

**USE-CASE BASED EARLY SOFTWARE EFFORT
PREDICTION USING FUZZY LOGIC AND
GENETIC ALGORITHMS**

BY

MOHAMMED WAJAHAT KAMAL

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

MAY 2012

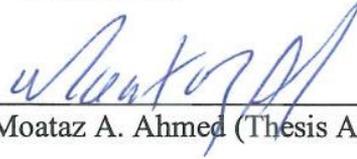
KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MOHAMMED WAJAHAT KAMAL** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee



Dr. Moataz A. Ahmed (Thesis Advisor)



Dr. Mohammad R. Alshayeb (Member)



Dr. Wasfi G. Al-Khatib (Member)



Dr. Adel F. Ahmed
Department Chairman



Dr. Salam A. Zummo
Dean of Graduate Studies

29/5/12
Date



Dedicated to

My parents, wife, sisters and brother

ACKNOWLEDGEMENT

Verily, all praise is to ALLAH, the creator and sustainer of the worlds, the ONE who deserves all gratitude and the ONE whom we turn for help. The successful completion of this thesis is solely attributed to the mercy of ALLAH.

The various facilities and resources provided by the King Fahd University of Petroleum and Minerals deserve special acknowledgment. This thesis would not have been possible without the help and guidance of several individuals who in one way or another contributed and extended their valuable assistance in course of this endeavor.

First and foremost, I owe my deepest gratitude to my thesis advisor Dr. Moataz Ahmed whose sincerity, knowledge and compassionate attitude is unquestionable. I thank him for being my inspiration, role model and an elder brother which helped me overcome the difficulties during the course of this research work. One simply cannot wish for a better and friendlier advisor.

I sincerely thank my committee members Dr. Mohammad Alshayeb and Dr. Wasfi Al-Khatib for their support, observations and helpful feedback.

The honest and valuable assistance of my dear friends Hamza Salami and Sameer Arastu deserve special mention. I also wish to thank all my friends in the student housing who made my stay at KFUPM memorable. May ALLAH bless them and help them in their future endeavors.

Finally, I would like to thank my family members for their continuous support, patience and prayers.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	IV
TABLE OF CONTENTS	VI
LIST OF TABLES	XIII
LIST OF FIGURES	XVI
THESIS ABSTRACT	XX
ملخص الرسالة	XXI
CHAPTER 1 INTRODUCTION	1
1.1. Research Questions	5
1.2. Main Contributions	7
1.3. Organization of the Thesis	8
CHAPTER 2 BACKGROUND	9
2.1. Use-Case based Effort Prediction	9
2.2. Fuzzy Logic	12
2.2.1. Fundamentals of Fuzzy Logic.....	15
2.2.2. Fuzzy Logic Systems	20
2.2.3. Adaptive Fuzzy Logic Systems	21

2.3. Genetic Algorithms	23
2.3.1. Genetic Algorithms: Fundamentals and Terminology.....	24
2.3.2. Process Flow in a Genetic Algorithm	28
2.3.3. Genetic Fuzzy Systems	28
CHAPTER 3 LITERATURE REVIEW.....	38
3.1. Use-Case based Effort Prediction Techniques.....	39
3.1.1. Use Case Points.....	39
3.1.2. Transactions	40
3.1.3. Paths.....	40
3.1.4. Extended Use Case Points	41
3.1.5. UCPm.....	41
3.1.6. Adapted Use Case Points	41
3.1.7. Use Case Size Points.....	42
3.1.8. Fuzzy Use Case Size Points	43
3.1.9. Simplified Use Case Points.....	43
3.1.10. Industrial Use Case Points	44
3.2. Comparison Criteria.....	44
3.2.1. Accuracy	45
3.2.2. Ease of Use	45
3.2.3. Use Case detail considerations.....	45
3.2.4. Factor Inclusion	46
3.2.5. Adaptability.....	46

3.2.6.	Handling Imprecision and Uncertainty	47
3.2.7.	Sensitivity	47
3.2.8.	Transparency.....	47
3.2.9.	Appropriate use of Productivity Factor.....	48
3.2.10.	Artifacts Considered	48
3.2.11.	Empirical Validations	49
3.3.	Comparison of Prediction Techniques.....	49
3.3.1.	Accuracy	50
3.3.2.	Ease of Use	51
3.3.3.	Use Case Detail Considerations.....	53
3.3.4.	Factor Inclusion	55
3.3.5.	Adaptability.....	57
3.3.6.	Handling Imprecision and Uncertainty.....	58
3.3.7.	Sensitivity	59
3.3.8.	Transparency.....	60
3.3.9.	Appropriate Use of Productivity Factor.....	61
3.3.10.	Artifacts Considered	63
3.3.11.	Empirical Validations	64
3.4.	Analysis.....	66

CHAPTER 4 RESEARCH APPROACH AND PROPOSED FRAMEWORKS..... 70

4.1.	Motivation and Research Approach.....	70
4.2.	' <i>f-UCP</i> ': Fuzzy Use Case Points Method	75

4.2.1.	Defining Antecedent and Consequent Fuzzy Sets	76
4.2.2.	Rule Base Formulation	77
4.2.3.	f-UCP Training	78
4.2.4.	f-UCP Validation	79
4.3.	The Proposed Adaptive Fuzzy Logic based Framework for Effort Prediction .	80
4.3.1.	Initializing the System	80
4.3.2.	Formulating the Rule Base.....	84
4.3.3.	Training the System	86
4.3.4.	Framework Validation	88
4.4.	The Simplified Adaptive Fuzzy Logic based Framework for Effort Prediction	89
4.4.1.	Initializing the System	90
4.4.2.	Formulating the Rule Base.....	90
4.4.3.	Training the System	92
4.4.4.	Framework Validation	92
4.5.	The Proposed Genetic Fuzzy System (GeFuSys-M) for evolving multi-layered architectures for Use-Case based Effort Prediction Systems.....	92
4.5.1.	Chromosome Design for GeFuSys-M	95
4.5.2.	The Genetic Learning Process	100

CHAPTER 5 EXAMINING THE IMPACT OF VARIOUS ALTERNATIVES ON PERFORMANCE..... 102

5.1.	Factor Analysis	103
5.1.1.	Principal Components Analysis on Experience Factors (EF).....	104

5.1.2. Principal Components Analysis on Technical Complexity Factors (TCF)	105
5.2. Training Algorithm	106
5.3. Genetic Learning of Rule Sets	110
5.3.1. Chromosome Structure	110
5.3.2. Other Genetic Learning Considerations	111
5.4. Choice of Fuzzy Inference System	112
5.4.1. Design details for comparing the Mamdani FIS and Sugeno FIS	114
5.5. Impact of Pairwise Combinations on the performance of the Simplified Effort Prediction Framework	117
5.6. Impact of Design Parameters on the performance of the Simplified Effort Prediction Framework	120
5.6.1. Method 1 for defining the parameters of the Gaussian membership functions	122
5.6.2. Method 2 for defining the parameters of the Gaussian membership functions	123

CHAPTER 6 EXPERIMENTS AND RESULTS..... 125

6.1. Prediction Accuracy	125
6.2. Artificial Data Generation	126
6.3. Experiment 1: Evaluating the prediction accuracy of the f-UCP model	128
6.3.1. Implementation Details	128
6.3.2. Results and Discussion	129

6.4. Experiment 2: Impact of TCF and EF on use-case based effort prediction.....	133
6.4.1. Implementation Details.....	133
6.4.2. Results and Discussion	133
6.5. Experiment 3: Evaluating the prediction accuracy of the proposed framework	
.....	139
6.5.1. Implementation Details.....	139
6.5.2. Results and Discussion	139
6.6. Experiment 4: Evaluating the prediction accuracy of the simplified framework	
.....	145
6.6.1. Implementation Details.....	145
6.6.2. Results and Discussion	145
6.7. Experiment 5: Impact of pairwise combinations on prediction accuracy.....	151
6.7.1. Results and Discussion	151
6.8. Experiment 6: Comparison of Mamdani type FLS vs. the Sugeno type FLS..	158
6.8.1. Results and Discussion	158
6.9. Experiment 7: Impact of design parameters on prediction accuracy.....	165
6.9.1. Results and Discussion	165
6.10. Experiment 8: Evaluating the system performance using genetic learning of	
rule sets	172
6.10.1. Implementation Details.....	172
6.10.2. Results and Discussion	172
6.11. Experiment 9: Impact of architecture on the effort prediction framework using	
GeFuSys-M.....	177

6.11.1. Implementation Details	177
6.11.2. Results and Discussion	177
CHAPTER 7 CONCLUSION.....	185
7.1. Introduction.....	185
7.2. Major Contributions.....	185
7.3. Limitations and Future Work.....	187
7.3.1. Limitations	187
7.3.2. Future Work	188
BIBLIOGRAPHY	189
CURRICULUM VITAE.....	202

LIST OF TABLES

Table 1: Classification of Algorithmic Models	4
Table 2: Evaluation of techniques based on ‘Accuracy’	50
Table 3: Evaluation of techniques based on ‘Ease of Use’	51
Table 4: Evaluation of techniques based on ‘Use Case Detail Considerations’	53
Table 5: Evaluation of techniques based on ‘Factor Inclusion’	55
Table 6: Evaluation of techniques based on ‘Adaptability’	57
Table 7: Evaluation of techniques based on ‘Handling Imprecision and Uncertainty’	58
Table 8: Evaluation of techniques based on ‘Sensitivity’	59
Table 9: Evaluation of techniques based on ‘Transparency’	60
Table 10: Evaluation of techniques based on ‘Appropriate Use of Productivity Factor’ .	61
Table 11: Evaluation of techniques based on ‘Artifacts Considered’	63
Table 12: Evaluation of techniques based on ‘Empirical Validations’	64
Table 13: Chromosome Description	96
Table 14: Chromosome Description	111
Table 15: Design Considerations for the Genetic Learning Process	112
Table 16: Comparison of Mamdani FIS and Sugeno FIS	114
Table 17: Summary of Prediction Quality on f-UCP using five different datasets, showing pred(25) and MARE values on training and testing datasets.....	130

Table 18: KMO and Bartlett's Test Results on TCF	135
Table 19: Total Variance Explained - TCF.....	135
Table 20: Pattern Matrix - TCF	136
Table 21: Component Correlation Matrix - TCF	136
Table 22: KMO and Bartlett's Test on EF	137
Table 23: Total Variance Explained - EF	137
Table 24: Pattern Matrix - EF.....	138
Table 25: Component Correlation Matrix - EF.....	138
Table 26: Summary of Prediction Quality on the Effort Prediction System using five different datasets, showing pred(25) and MARE values on training and testing datasets	141
Table 27: Summary of Prediction Quality on the Effort Prediction System using five different datasets, showing pred(25) and MARE values on training and testing datasets	146
Table 28: Summary of Prediction Quality on the Normal and Pairwise Effort Prediction Systems using five different datasets, showing pred(25) and MARE values on training and testing datasets	152
Table 29: Summary of Prediction Quality on the Mamdani and Sugeno Effort Prediction Systems using five different datasets, showing pred(25) and MARE values on training and testing datasets	159
Table 30: Summary of Prediction Quality on the Effort Prediction Systems (Parameter Design Method 1 and Method 2) using five different datasets, showing pred(25) and MARE values on training and testing datasets	166

Table 31: Summary of information about the genetic learning process, showing minimum error (MARE) values in the sub-experiments.....	173
Table 32: Summary of information about the genetic learning process of GeFuSys-M, showing minimum error (MARE) values in the sub-experiments.....	179

LIST OF FIGURES

Figure 1: Classification of antecedent ‘Number of Simple UC’ into fuzzy regions.....	16
Figure 2: Fuzzy Logic System with fuzzifier and defuzzifier	21
Figure 3: Process Flow in a Genetic Algorithm.....	30
Figure 4: Soft computing and learning in Fuzzy Systems	31
Figure 5: Hybridization in soft computing.....	33
Figure 6: Genetic Learning using Pittsburgh Rule Encoding Approach	36
Figure 7: Genetic Learning with Michigan Rule Encoding Approach	37
Figure 8: The Research Approach	74
Figure 9: Architecture of the f-UCP method	76
Figure 10: The Proposed Framework	81
Figure 11: The Proposed Simplified Framework for Effort Prediction	91
Figure 12: Chromosome Structure for GeFuSys-M.....	96
Figure 13: Adjacency matrix in case of 5 components.....	98
Figure 14: Adjacency matrix in case of 4 components.....	98
Figure 15: Chromosome Structure.....	111
Figure 16: Choices that need to be made before designing an FLS.....	122
Figure 17: Prediction of effort using trained f-UCP on training dataset.....	131
Figure 18: Prediction of effort using trained f-UCP on testing dataset	132

Figure 19: Average MARE graph of training the Effort Prediction System	142
Figure 20: Prediction of effort using the trained Effort Prediction System on training datasets	143
Figure 21: Prediction of effort using the trained Effort Prediction System on testing datasets	144
Figure 22: Average MARE graph of training the Effort Prediction System	147
Figure 23: Prediction of effort using the Effort Prediction System on training datasets	148
Figure 24: Prediction of effort using the Effort Prediction System on testing datasets..	149
Figure 25: Comparison of training and testing errors (MARE) on the Effort Prediction System.....	150
Figure 26: Average MARE graph of training the Normal and Pairwise Effort Prediction System.....	153
Figure 27: Prediction of effort on Normal and Pairwise Effort Prediction Systems using training datasets	154
Figure 28: Prediction of effort using Normal and Pairwise Effort Prediction Systems on testing datasets	155
Figure 29: Comparison of training and testing errors (MARE) on the Normal Effort Prediction System	156
Figure 30: Comparison of training and testing errors (MARE) on the Pairwise Effort Prediction System	157
Figure 31: Average MARE graph of training the Mamdani (Method 1) and Sugeno (Method 2) Effort Prediction Systems	160

Figure 32: Prediction of effort using the Mamdani and Sugeno Effort Prediction Systems on training datasets	161
Figure 33: Prediction of effort using the Mamdani and Sugeno Effort Prediction Systems on testing datasets	162
Figure 34: Comparison of training and testing errors (MARE) on Mamdani Effort Prediction System (Method 1)	163
Figure 35: Comparison of training and testing errors (MARE) on Sugeno Effort Prediction System (Method 2)	164
Figure 36: Average MARE graph of training the Effort Prediction Systems (Method 1 and Method 2)	167
Figure 37: Prediction of effort using the Effort Prediction Systems (Method 1 and Method 2) on training datasets.....	168
Figure 38: Prediction of effort using the Effort Prediction Systems (Method 1 and Method 2) on testing datasets	169
Figure 39: Comparison of training and testing errors (MARE) on the Effort Prediction System (Method 1).....	170
Figure 40: Comparison of training and testing errors (MARE) on the Effort Prediction System (Method 2).....	171
Figure 41: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 1	174
Figure 42: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 2	175

Figure 43: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 3	176
Figure 44: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 1	180
Figure 45: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 2	181
Figure 46: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 3	182
Figure 47: A four component sample architecture	183
Figure 48: A three component sample architecture	183
Figure 49: A three component sample architecture	184

THESIS ABSTRACT

Name: Mohammed Wajahat Kamal
Title: Use-Case based Early Software Effort Prediction using Fuzzy Logic and Genetic Algorithms
Major Field: Computer Science
Date of Degree: May 2012

An important and challenging activity, Software Development Effort Prediction involves dealing with imprecision, uncertainty and dearth of information in the early stages of software development. With the focus shifting more towards the use of machine learning techniques, predicting effort using Fuzzy Logic, Neural Networks, Genetic Algorithms or a combination of these has also been heavily considered by the research community. This thesis presents an adaptive fuzzy logic based framework for use-case based effort prediction capable of handling imprecision and incorporating expert opinions. Additionally, a simplified framework is conceptualized and empirical evaluations regarding the impact of various objectives are investigated which show that the proposed frameworks are promising and produce acceptable results. Since prediction accuracy of a fuzzy logic based effort prediction system is highly dependent on the system architecture, the development of a genetic-fuzzy tool to evolve different architectures provides results pertaining to the impact of architectural differences on the accuracy of effort prediction systems.

ملخص الرسالة

الاسم:	محمد وجهات كمال
عنوان الرسالة:	التقدير المبكر للجهد المطلوب لتطوير البرمجيات باستخدام المنطق الغامض إعتياداً علي مخطط الاستخدام.
التخصص:	علوم الحاسب الآلي والمعلومات
تاريخ التخرج:	جمادي الآخرة 1433 هـ

عملية تقدير المجهود المبذول في تطوير البرمجيات يعتبر عملية مهمة, وتواجهها تحديات كبيرة في ظل عدم اتضاح الرؤية وعدم دقة وتوافر المعلومات في تلك المراحل المبكرة لتطوير البرمجيات . مع وجود اتجاه بحثي كبير يركز علي استخدام تقنيات مثل المنطق الغامض (Fuzzy Logic), الشبكات العصبية, والخوارزمية الجينية, كل واحد بمفردها او الجمع بينها من اجل تقدير المجهود الذي تتطلبه عملية تطوير البرمجيات في المراحل المبكرة , في هذا العمل تم تقديم اطار عمل يعتمد علي المنطق الغامض (Fuzzy Logic) من اجل تقدير الجهد المطلوب لتطوير البرمجيات, اعتمادا علي مخططات الاستخدام (Use-Case), بحيث يتعامل اطار العمل المقدم هنا بكفاءة مع عدم وضوح الرؤية في بدايات مشاريع تطوير البرمجيات ويتم فيه دمج اراء الخبراء . تم شرح واختبار اطار العمل المقدم وكانت النتائج التي تم الحصول عليها واعده ومقبوله الي حد كبير . ولان تقدير المجهود المطلوب لتطوير البرمجيات باستخدام الالية المطروحة في هذا البحث يعتمد بصورة كبيرة علي معماريه النظام تم تطوير اداة برمجية تعتمد علي كل من الخوارزمية الجينية و المنطق الغامض (Fuzzy Logic) من اجل دراسة اثر استخدام عدد من المعماريات المختلفة في دقة تقدير الجهد المطلوب لتطوير البرمجيات.

CHAPTER 1

INTRODUCTION

Software Development Effort Prediction, more commonly referred to as Software Effort Estimation is basically a prediction procedure/methodology targeted at predicting the amount of effort (man/hour), cost and time required to actualize a software development task/job. Software Effort Prediction falls under the domain of the more abstract procedure of Software Cost Prediction. With regards to a particular software development project, the associated costs are related to hardware, training, traveling and mostly effort pertaining to the payment of software engineers or programmers[86]. As such it is noticeable that 'effort' is the predominant factor for predicting the cost of a software development project. In order to obtain accurate and reliable 'cost' estimates for software development projects, it is necessary to obtain accurate and reliable 'effort' estimates. Consequently, a lot of research has been carried out in the domain of software effort prediction. Software effort prediction spawned some of the first attempts at rigorous software measurement, so it is the oldest, most mature aspect of software metrics [81].

Narrowing down the aspect of Software Cost Prediction to Software Effort Prediction, the focus shifts on the practices and methodologies used in the field of Effort Prediction. The major factor in predicting the effort is the size of the software system being developed. Accurately predicting the size of a software system is proportional to accurately predicting the effort required to develop the complete system [40]. For this reason, various size metrics have been proposed by researchers in the academia and industry over the years. Typical size metrics are Lines of Code [12], Function Points [3], Use Case Points [42], Class Points [20], Feature Points, etc. In addition to the size of the software system, there are various other technical and non-technical factors involved in the effort prediction process. Videlicet, Effort Prediction is a complex procedure involving many factors and their interrelationships. Nevertheless, it is imperative for software development projects.

According to Boehm et al [10], the main goals of software cost and effort prediction are budgeting, tradeoff and risk analysis, project planning and control, and software improvement investment analysis. A good estimate can have many advantages for the project and understandably, a bad estimate can spell doom for a project. Underestimating the costs may result in management approving proposed systems that exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time, whereas overestimating may result in too many resources committed to the project, or, during contract bidding, result in not winning the contract, which can lead to loss of jobs [53]. Since, predicting effort is an activity which is done relatively early in the software development lifecycle, it becomes a challenging task to accurately predict effort

based on incomplete, crude, uncertain and imprecise inputs [45][53]. Moreover, early estimates of size, for example based on requirements specification, are the most difficult to obtain, and they are often the least accurate, because very little detail is known about the project and the product at its start [20]. Furthermore, available details are characterized as being imprecise and uncertain. It is also very difficult to model the relationships between various factors involved in the prediction process [57].

A variety of effort prediction models and metrics have been proposed to solve the problems associated with the prediction process. Unfortunately, there is not a single model which produces acceptable effort estimates in the early stages of the software development lifecycle. The issue is not with the design of the models but rather the uncertainty and imprecision involved in the overall process that makes it difficult to accurately predict effort. Broadly, the models are classified as algorithmic and non-algorithmic. Algorithmic Models such as COCOMO [10], COCOMO II [10], SLIM [71], Nelson Model, Wolverton Model, Doty Model, SoftCost, Price-S are some of the prominent effort prediction models in this domain. A brief discussion on the aforementioned models can be found in [53]. Algorithmic models are further classified as empirical and analytical models. An empirical model utilizes data from previous projects to evaluate the project at hand and derives the formulae from the analysis, whereas an analytical model uses formulae based on global assumptions [53]. A complete classification of the algorithmic models can be seen in Table 1. Non-algorithmic models comprise of the techniques and procedures related to Expert Judgment such as the Pert Technique and Delphi Technique. Analogy Costing is also one of the prominent effort

prediction methods commonly used. Other non-algorithmic models include Parkinson, Price-to-win, Top-down and Bottom-up procedures.

Table 1: Classification of Algorithmic Models

Algorithmic Models [53]					
	Linear	Multiplicative	Power function	Discrete	Others
Empirical	<i>Nelson</i>	<i>Walston-Felix</i>	<i>COCOMO</i>	<i>Aron</i>	<i>Price-S</i>
Analytical			<i>Putnam</i>		<i>SoftCost</i>

In the recent years, there has been a trend shifting towards incorporation of artificial intelligence and machine learning concepts in multifarious domains, the domain of Software Effort Prediction is no exception. More recently, there have been various attempts to incorporate machine learning techniques such as Fuzzy Logic, Neural Networks, Bayesian Belief Networks, Genetic Algorithms and a combination of these in various Effort Prediction Models. A detailed reference to these works can be found in [79][80][81]. Such models have tried to deal with the imprecise and uncertain data available for prediction.

Despite the availability of a plethora of models in the domain of software effort prediction, the domain still lacks an accomplished technique for accurately predicting effort [40]. As a consequence, there is a continuous cycle of developmental and research

works aimed at producing efficient and reliable models for predicting effort [13][14][21][34][40][41][64][70]. It is desirable to develop models which can produce accurate, efficient and reliable effort estimates in the early stages of software development. Considering the large scale of software development, minute improvements in the prediction accuracy is of significant importance. While developing new models, the focus lies in taking into consideration/deciding all the strengths and weaknesses of the available models and procedures. Decisions such as the choice of size metric, machine learning technique to be used and inclusion/exclusion of a number of internal factors are of significant importance. Another important deciding factor is to choose between purely algorithmic/mathematical models as opposed to a combination of expert opinion and mathematical models. More details pertaining to such considerations are discussed in the following sections.

1.1. Research Questions

The main aim of this research is to propose a framework for use-case based early effort prediction capable of dealing with imprecision and incorporating expert opinions which can produce accurate and reliable effort estimates.

In order to meet the targets of this research, initially a critical literature review of the domain was conducted. The literature review can be classified in two sets. The first focused more on general effort estimation models which resulted in narrowing down our problem to using a specific size predictor, namely Use Cases (more details can be found

in Chapter 2). The second focused more on literature pertaining to Use Case (UC) based effort prediction. Both surveys were conducted systematically, the latter of which has been recently published [40][41].

Based on the analysis of the critical literature review (see Section 3.4), the following research questions have been identified.

1. How can fuzzy logic be employed to enable the development of transparent use case based effort prediction models capable of incorporating expert opinions?
2. What are the factors that impact the accuracy of the effort prediction? Among the variety of factors considered by researchers in the prediction process, e.g., ‘technical complexity factors’ and ‘experience factors’, which factors can be ignored?
3. What is the impact of using some strategies, e.g., pairwise combinations, in the context of defining rules for a specific fuzzy logic based effort prediction system? Can pairwise combinations be used to avoid the frequently faced problem of ‘*curse of dimensionality*’ i.e. to reduce the exhaustive number of rules in a given fuzzy system?
4. Does the choice of a particular implementation of a fuzzy logic system affect the prediction accuracy of the Effort Prediction System? More specifically, what is the difference in prediction accuracy of the system when either Mamdani type fuzzy logic system or Sugeno type fuzzy logic system is used?

5. How can Genetic Algorithms help in realizing an efficient Effort Prediction System? Is it possible to use genetic learning of fuzzy rule based architectures to evolve efficient effort prediction systems?
6. Does the architecture of the prediction model have any effect on the model's prediction accuracy? Specifically, what is the impact of single-layer architecture vs. multi-layer architecture on the prediction accuracy of the Effort Prediction System?

1.2. Main Contributions

The main contributions of this work are as follows;

- i. Development of a use-case based effort prediction framework using fuzzy logic, capable of incorporating expert opinions and handling imprecision.
- ii. Identification and reduction of the 13 technical complexity factors and 8 experience factors to 6 and 5 respectively, based on the results obtained from performing Factor Analysis.
- iii. Investigating the impact of using pairwise combinations for defining rules for the fuzzy logic based effort prediction system on the prediction accuracy.
- iv. Comparison of prediction accuracies for the Effort Prediction System obtained using Mamdani type fuzzy logic system and Sugeno type fuzzy logic system.
- v. Investigating the impact of design parameters on the prediction accuracy of the Effort Prediction System.

- vi. An alternative proposal for a simplified use-case based effort prediction framework which includes input factors pertaining to actors and use-cases and excludes all the additional factors.
- vii. Development of a single-layer genetic fuzzy system for use-case based effort prediction, which gives the best prediction system in terms of prediction accuracy.
- viii. Design, development and implementation of a new chromosome structure for a generic multi-layer genetic fuzzy system (GeFuSys-M).
- ix. Fuzzifying the existing Use Case Points method to actualize an efficient model (f-UCP) on similar lines as “f-COCOMO”.

1.3. Organization of the Thesis

The thesis is organized as follows. Chapter 2 provides a background for the core knowledge areas of this thesis such as Use Case based Effort Prediction, Fuzzy Logic and Genetic Algorithms. Chapter 3 presents the literature review on Use Case based Effort Prediction techniques. Chapter 4 discusses the research approach and the proposed frameworks for effort prediction. Chapter 5 presents the experimental designs, followed by Chapter 6 which discusses the experimental results. Chapter 7 concludes the thesis, highlights the contributions and provides directions for future research.

CHAPTER 2

BACKGROUND

This chapter provides the necessary background related to the core knowledge areas associated with this work. Three major knowledge areas have been covered quite extensively; Effort Prediction based on Use Cases, Fuzzy Logic and Genetic Algorithms. An avid reader can find more information in the references used in the course of this chapter.

2.1. Use-Case based Effort Prediction

The history of using use cases for effort prediction started with the development of the Unified Modeling Language (UML) [76] by Jim Rumbaugh, Grady Booch, and Ivar Jacobson of Rational Software Corporation in mid-nineties [36]. Sometime later, UML was incorporated into the Rational Unified Process RUP by Rational Software. Meanwhile, Gustav Karner also of Rational Software Corporation developed an

estimating technique to predict the effort required based on Use Cases, much the same way as Function Points [3]. Karner's technique is known as Use Case Point (UCP) method [42] and is incorporated into RUP. It is the basic estimating technique for predicting effort based on use cases.

Use cases, as being available relatively early during the software development lifecycle, are expected to offer a good estimate of the size of the corresponding future system. This is of significant importance as effort estimates are required early in the software development lifecycle. Use cases are used to capture and describe the functional requirements of a software system. Use Case Models define the functional scope of the system. The Use Case model is relevant and valuable for early size measurement and predicting effort as it employs use cases as input. According to a survey conducted by Neil and Laplante [68], 50% of the software projects have their requirements presented as Use Cases. Moreover, as Moira Forbes [24] states, "*With very little effort, you can use this technique to get a very early gross estimate. And it will be just as accurate (or inaccurate) as any other method you could use at this early stage in the project*". This highlights the fact that the effort required to predicting or estimating the effort required in a software project should also be minimum. This is echoed by Ochodek *et al* [70] while stating the two kinds of useful research in this field, amongst which the following applies in this context; "*making effort estimation more accurate without increasing the time and money spent on effort estimation*".

Based on the aforementioned facts, the approach to estimate effort using Use Cases has gained popularity and subsequently the basic technique of UCP has gained more recognition. Consequently, many techniques based on UCP have been proposed since then, like Use Case Size Points [13], Extended Use Case Points [22], UCP modified [21], Adapted Use Case Points [64], Transactions [73] and Paths [73] to mention a few. A more detailed description of a few Use Case based effort prediction techniques will be presented in the following sub sections.

Along with the advantages of using these methods, several issues and concerns about these approaches have also been raised. Few of the problems are as follows; varying complexities in the use case models, adjusting the technical complexity factors and experience factors, classification of use cases and the overall construction of the UCP method. Additionally, there are few problems associated with using Use Cases as well [1][40]. First, there is no standardized style of writing a Use Case. The variations in the style and formality of writing a Use Case brings about many issues like how to measure the size of the Use Case, and how to classify the Use Case.

Second, an important issue with Use cases is the assessment of complexity of the Use Case. In a typical CRUD scenario (Create, Replace, Update, Delete), is it correct to consider the UC as one UC with four scenarios or one UC with one scenario, as all the other scenarios are so similar. Third, a UC represents an external actor's view. In case the system has states, it becomes necessary to define another model to represent this behavior which is quite complex. Fourth, granularity of Use Cases is another big issue. Questions

like these do not have specific answers: what is the optimum length and what are the details that should be mentioned while describing a UC? Fifth, most of the researchers complain about the non-consideration of non-functional requirements in the Use Case models.

This raises the question that, are UCs a good choice to depend on for estimating effort? The answer lies with the proper evaluation and investigation of these approaches. Different proposed approaches have tried to address some of these issues and many of them have ameliorated some problems as well, but there is not a single approach which addresses all issues satisfactorily. We discuss these approaches and compare them for analysis against some criteria in Chapter 3.

2.2. Fuzzy Logic

Human beings are undoubtedly gifted with the remarkable ability to reason and make decisions in highly imprecise, unstructured, uncertain and ambiguous environments. It is this ability which creates the differentiating factor between machines and humans. In order to realize the goal of Artificial Intelligence, which aims at minimizing differences between a human and a machine, many novel concepts have been conceived and implemented [77]. Terms such as ‘machine learning’ and ‘soft computing’ are not new paradigms, with the former encompassing a huge variety of techniques and the latter comprising majorly of techniques such as Fuzzy Logic [99][100], Artificial Neural Networks [57], Bayesian Networks [22] and Genetic Algorithms [19]. Another sub-field of artificial intelligence is ‘Computational Intelligence’ which comprises majorly of

Fuzzy Logic, Neural Networks and Evolutionary Computation [19]. The main aim of all these concepts is to bring machine level intelligence on par with human level intelligence, in other words making the machine capable enough to deal with imprecision, uncertainty and partiality of knowledge.

Fuzzy Logic, a novel idea first conceived by Professor Lotfi Zadeh in 1965 caused a huge paradigm shift in the domain of artificial intelligence. According to Zadeh [99], fuzzy logic is “*a logic which mirrors the remarkable capability of human mind to reason with information which is imprecise, uncertain and partially true*”. The main motivation behind the concept of fuzzy logic is to deal with imprecision and uncertainty.

Imprecision is the lack of exactness or defect in the accuracy of a certain measurable quantity. In terms of fuzzy logic, it refers to the vagueness associated with natural language. Instances of imprecision in statements are; “The weather is hot” and “The room temperature is very high”. From these statements, we have a subjective understanding about the terms ‘hot’ and ‘very high’, but we cannot quantify ‘hot’ as being more than 30 degrees Celsius or 35 degree Celsius. One can argue with the former and another can stick with later interpretation. This is what introduces vagueness in definition and ambiguity in interpretation, collectively termed as imprecision in natural language.

Uncertainty refers to unpredictability, indeterminacy and indefiniteness. It applies to the prediction of future events or measurement of known/unknown quantities. In the field of Effort Prediction, there are numerous factors involving measurements and predictions,

and as such there is a lot of uncertainty associated with the overall process. Typically, uncertainty arises because of the following three reasons; volume of work, theoretical ignorance and practical ignorance [60]. Furthermore, Mendel [60] identified four sources of uncertainty while building systems based on fuzzy logic, in addition to the three sources of uncertainty in the effort prediction process.

- *Uncertainty about the meanings of the words that are used in a rule*
- *Uncertainty about the consequent that is used in a rule*
- *Uncertainty about the measurements that activate the FLS*
- *Uncertainty about the data that are used to tune the parameters of a FLS*

A point to note is that the aforementioned four points will be more understandable to the reader provided they are familiar with the concepts of Fuzzy Logic Systems (FLS); nevertheless a description of the FLS is provided in the following sub-sections. Moreover, readers are advised to refer to [67] for a more detailed explanation regarding uncertainty. In what follows next, are the necessary background related to the fundamentals of Fuzzy Logic, a brief description of Fuzzy Logic Systems and Adaptive Fuzzy Logic Systems.

2.2.1. Fundamentals of Fuzzy Logic

This section introduces the basic concepts related to the theory and practice of fuzzy logic.

2.2.1.1. Fuzzy Sets and Membership Functions

Fuzzy Logic is based on the concept of fuzzy set theory which is an extension of the classical set theory. In classical set theory, the membership $\mu_A(x)$ of an element 'x' in a set 'A' is defined as $\mu_A(x) = 1$, if and only if **x belongs to A**, and the membership is defined as $\mu_A(x) = 0$, if and only if **x does not belong to A**. Such sets are also called crisp sets allowing only values of 0 or 1. Fuzzy set theory allows the concept of partial membership in sets, shown as follows;

$$\mu_A : X \rightarrow [0,1]$$

$\mu_A(x)$ is called the membership function and **X** is a reference set also called as Universe of discourse. Hence, $\mu_A(x)$ is interpreted as the degree of membership of **x** in fuzzy set **A**, where **x** belongs to **X**. A membership function (MF) is a curve that defines how each element in the input space is mapped to a membership value (or degree of membership) between 0 and 1. Amongst the many types of available membership functions, the commonly used MF's are Gaussian MF, triangular MF, trapezoidal MF and bell MF [60]. For sake of illustration, an example of membership functions for the input factor 'Number of Simple Use-Cases' is depicted in Figure 1.

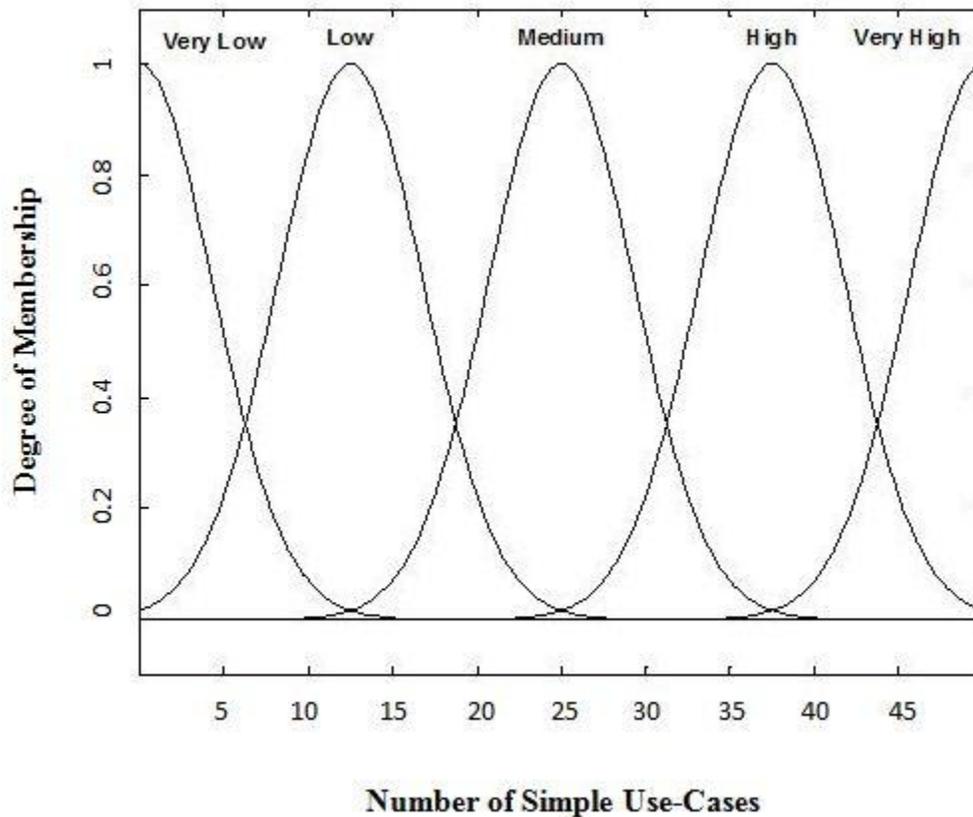


Figure 1: Classification of antecedent ‘Number of Simple UC’ into fuzzy regions

2.2.1.2. Linguistic Variables

Any defined system has variables and each variable has defined values to achieve the functionality of the system. Fuzzy Logic uses ‘linguistic variables’ and ‘linguistic values’ in place of the normal variables and their corresponding values. A linguistic variable is a variable which accepts values in terms of words, also called linguistic terms. These words or ‘linguistic terms’ are associated with a certain degree of membership in fuzzy sets.

Consider an instance, where in we need to represent ‘temperature’ whose numerical value is 30 degree Celsius. The same sentence in terms of fuzzy logic can be represented by the following statements;

- *The temperature is high*, with degree of membership 0.85
- *The temperature is medium*, with degree of membership 0.5
- *The temperature is low*, with degree of membership 0.15

From the above statements, it is clear that statements in fuzzy logic are of the type “**Linguistic Variable is Linguistic Value**”, where ‘linguistic values’ can take subjective values from the ‘linguistic term sets’. An example of a ‘linguistic term set’ for the ‘linguistic variable’ *temperature* can be {very low, low, medium, high, very high}.

2.2.1.3. Logical Operators

Fuzzy Logic is a superset of Boolean Logic. The standard ‘AND’ and ‘OR’ operators of Boolean logic are replaced by the ‘MAX’ and ‘MIN’ operators respectively but the standard logical operations are the same.

For all x belongs to X , the intersection of two fuzzy sets A and B results in another fuzzy set with the membership function defined as follows;

$$\mu_{A \cap B}(x) = \min (\mu_A(x), \mu_B(x))$$

For all x belongs to X , the union of two fuzzy sets A and B results in another fuzzy set with the membership function defined as follows;

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

Other additional operators include the **t-norm** and **t-conorm** operators. For brevity purposes, the detailed explanation about these operators has not been included.

2.2.1.4. Fuzzy Rules

Rules in fuzzy logic assume the form of ‘**condition-action**’ pair, more precisely, ‘**if-then**’ pair. Typical fuzzy rules are of the form;

$$\mathbf{if} (x \text{ is } A) \mathbf{then} (y \text{ is } B)$$

‘ x ’ and ‘ y ’ are the *linguistic variables* from the input ranges X and Y (universes of discourse) respectively. ‘ A ’ and ‘ B ’ are *linguistic values* defined by fuzzy sets on the ranges X and Y respectively. The **if**-part of the rule is called ‘antecedent’ and the **then**-part is called the ‘consequent’.

The antecedent of the rule returns a single value between 0 and 1, which is also called as ‘support of the rule’. The consequent of the rule returns a fuzzy set which is assigned to the output. The output fuzzy set represented by a membership function is shaped according to the degree of support of the antecedents, and this process is called ‘**Implication**’. In other words, if the antecedent of a rule is true to a certain degree, then the consequent is also true to the same degree.

Fuzzy Logic Rules can have multiple antecedents and consequents. A representation of such a rule is as follows;

if (x is A) and (y is B) and (z is C) then (p is Q) and (r is S)

In case of multiple antecedents, the degree of support of a rule is calculated by applying the fuzzy logical operators and a single number between 0 and 1 is computed. In case of multiple consequents, the degree of support of a rule equally affects the shape of all consequent fuzzy sets.

2.2.1.5. Fuzzy Inference

The process of fuzzy inference utilizes all the fuzzy logic concepts such as fuzzy sets, membership functions, linguistic variables, operators and fuzzy rules. Fuzzy inference is the process of mapping elements from the input domain to the elements of the output domain using fuzzy logic.

The fuzzy inference process comprises of five main parts;

- Fuzzification of the input variables
- Computing the degree of support of each rule by using fuzzy operators (AND or OR) in the antecedent
- Deriving the shape of the output fuzzy sets by implication from the antecedent to the consequent

- Aggregating all the output fuzzy sets across all rules into a single fuzzy set
- Defuzzification of the output variables

Systems which perform the fuzzy inference procedures are called Fuzzy Inference Systems or more generally referred to as Fuzzy Logic Systems.

2.2.2. Fuzzy Logic Systems

Fuzzy Logic Systems (FLS), commonly referred to as Fuzzy Rule based Systems, Fuzzy Expert Systems and Fuzzy Control Systems are class of systems which contribute in realizing the goal of artificial intelligence. The multifarious domains in which Fuzzy Logic Systems have been implemented complement the fact that they are an efficient means to solve complex problems when compared with mathematical models. Examples of domains where FLS have been employed include control theory, robotics, data classification, automation, computer vision, expert systems and decision support systems.

Technically, FLS can be classified into three major types [95];

- Pure Fuzzy Logic Systems
- Takagi and Sugeno Fuzzy Logic Systems (Sugeno FLS) [92]
- Fuzzy Logic Systems with fuzzifier and defuzzifier (Mamdani FLS) [58]

Typically, a fuzzy logic system comprises of four major components;

- **Fuzzifier:** Performs conversion of the crisp input values to fuzzy sets.
- **Fuzzy Rule Base:** Stores all the fuzzy '*if-then*' rules. Also called knowledge base.

- **Fuzzy Inference Engine:** Performs the fuzzy inference procedure (see Section 2.2.1.5).
- **Defuzzifier:** Performs conversion of fuzzy output sets to crisp output.

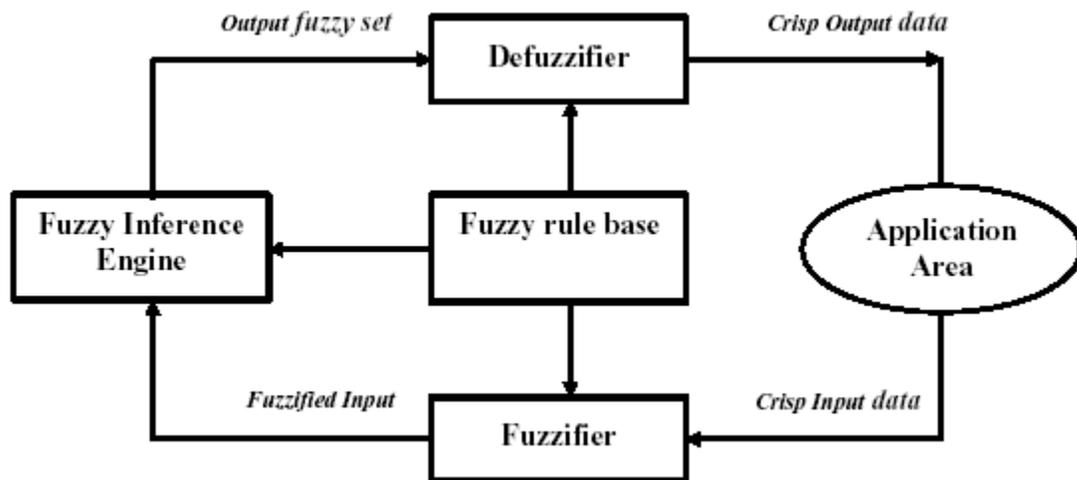


Figure 2: Fuzzy Logic System with fuzzifier and defuzzifier

Figure 2 presents a generic architecture of an FLS. More details pertaining to the architecture of an FLS can be found in [95].

2.2.3. Adaptive Fuzzy Logic Systems

Usually a predefined model is not present to fit a modeling scenario for a particular system. Only the numerical data (input-output pairs) for the system is available. In such a scenario, a generic fuzzy logic system can be modeled comprising of expert assisted design of membership functions and fuzzy rules. Training algorithms can then be applied to the FLS to fine tune the various parameters (shape of membership functions, position

of membership functions, number of rules in FLS, shape and position of consequent membership functions) associated with it. The numerical data assists in tuning the parameters.

This process of adapting a fuzzy system to a given set of data is called ‘training the system’ or ‘adapting the system’ or ‘parameter learning’. The training algorithms help the FLS to learn a given set of data it is modeling.

A proper definition of Adaptive Fuzzy Systems is given by Wang [95]; “*An adaptive fuzzy system is defined as a fuzzy logic system equipped with a training algorithm, where the fuzzy logic system is constructed from a set of fuzzy IF-THEN rules using fuzzy logic principles, and the training algorithms adjust the parameters of the fuzzy logic system based on numerical information*”.

Further, Wang [95] states two ways of realizing an Adaptive Fuzzy Logic System;

- Use of linguistic information (experts. knowledge) to develop an initial fuzzy logic system, and then adjust the parameters of the initial fuzzy logic system by using on numerical information.
- Use of numerical information and linguistic information to develop two separate fuzzy logic systems, and then final fuzzy logic system is obtained by averaging them.

The first method is the more commonly used one in practice. Many techniques comprising of training routines are also available for implementing an Adaptive FLS. One such example is ANFIS, an acronym for Adaptive Neuro Fuzzy Inference System, which uses the back propagation algorithm or a combination of back propagation and least squares algorithm to train the fuzzy logic system. A point worth noting is that ANFIS uses Sugeno type FLS to implement the training. Another example which uses Mamdani FLS is the training procedure prescribed by Mendel [60], wherein the means and standard deviations of the membership functions corresponding to the positions and shapes of membership functions are adjusted according to the input-output data pairs. Both of these methods have been utilized in course of this work and a comparison between the methods can also be seen in the following sections.

2.3. Genetic Algorithms

Genetic Algorithms (GAs) is a relatively old paradigm of evolutionary computation. Other recent paradigms include Swarm Intelligence, Ant Colony Optimization, Evolutionary Strategies and Particle Swarm Optimization. However, GAs is the most established among them all. Evolutionary Computation techniques use the concept of Darwinian principles and biological evolution to find highly optimized solutions for combinatorial problems.

GAs was first conceived by John Holland in 1975. Basically, GA's are general purpose search algorithms which use the principles of natural genetics to evolve solutions to problems [18]. GA's use three basic concepts of biological evolution namely; selection,

recombination and mutation to evolve a solution to a problem. The basic idea is to generate a set of possible solutions for a given problem and then apply the various GA operators related to selection, recombination and mutation to obtain a solution. Since, GA's produce enormous possibilities of solutions for a given problem, the probability of finding an optimal solution is very high. Because of this reason, genetic algorithms enjoy a special preference and are much favored in the domain of optimization problems.

The following subsections introduce the necessary background related to the fundamentals and terminology of GA's, a brief description of the process flow in GA's and a concise introduction about Genetic Fuzzy Systems.

2.3.1. Genetic Algorithms: Fundamentals and Terminology

2.3.1.1. Chromosome

Any GA starts with a set of candidate solutions. These solutions are encoded in a particular format. Most common methods of encoding are binary bit strings and real numbers. The structure which holds these encoded solutions is called a *Chromosome*.

2.3.1.2. Population

Each chromosome represents a particular solution for a given problem. A set of chromosomes representing a set of solutions is called *Population*. Each solution is also known as an *individual*, hence a population comprises of a set of individuals.

2.3.1.3. Fitness Function

The *fitness function* is the function which needs to be optimized. The aim of the GA is to minimize the error associated with applying the individuals (solutions) to the fitness function and maximizing the fitness value. The fitness function is also referred to as the *objective function*.

2.3.1.4. Generations

As the GA progresses, individuals in a population are evaluated based on their fitness value. Consequently, GA performs some computations (selection, recombination, mutation) to produce new populations from the old population. The successive populations are called *Generations*.

2.3.1.5. Parents and Children

In order to create new generations, the GA selects certain individuals from the current population (having maximum fitness) called *parents*, and creates new individuals in the new population called *children*. Few terminologies use *offspring* instead of children.

2.3.1.6. Selection

The process of selecting parents from a population based on fitness value is called *Selection*. This is based on Darwin's theory of 'survival of fittest', according to which the fittest parents should survive and create children. Many methods like *Boltzmann Selection*, *Ranking Selection*, *Roulette Wheel Selection*, *Elitist Selection* and *Tournament Selection* are available for the purpose of Selection in GA.

2.3.1.7. Recombination (Crossover)

The process of choosing two individuals (parent chromosomes) and swapping a segment of their encoded bits (real or binary) to produce children which are a combination of both the parents is called *Recombination*. In GA, this operation is called as *Crossover*. Commonly used types of crossover operators are *single point crossover*, *two point crossover* and *uniform crossover*.

2.3.1.8. Mutation

The process of swapping an encoded bit with a random value is called *Mutation*. Generally, in binary encoding of chromosomes, a bit is flipped from 0 to 1 or vice versa when mutation operator is applied. In case of real encoded chromosomes, a small value is added or subtracted to the existing value of a bit.

2.3.1.9. Summary of using Genetic Algorithms

Solving a particular optimization task using a GA requires the human designer to address the five following issues [19] which involve all the aforementioned fundamentals discussed so far.

- A genetic representation (chromosome) of candidate solutions
- A way to create an initial population of solutions
- An evaluation function (fitness function) which describes the quality of each individual
- Genetic operators (Selection, Crossover, Mutation) that generate new variants during reproduction
- Values for the parameters of the GA, such as population size, number of generations and probabilities (crossover probability, mutation probability) of applying genetic operators

2.3.2. Process Flow in a Genetic Algorithm

A genetic algorithm starts with an initial population of chromosomes which are randomly generated. A fitness function is devised so as to measure the optimality of a particular solution. In each generation, the fitness function is applied to all the individuals of a population and their respective fitness scores are computed. Based on the fitness scores, the most fit parent chromosomes are selected in order to reproduce children for the next generation. Selection is done based on the choice of the selection operator. After the parents are selected, crossover and mutation operators are applied depending on the choice of the operators. Crossover and mutation probabilities are specified and new children are created. The GA then evaluates the fitness of individuals of the new generation.

This process repeats in a continuous loop until a terminating criteria is reached. Generally referred to as the stopping condition, the termination criteria is a measure of the desired fitness (example: minimum error) and thus it defines when the GA should terminate its operation. A flowchart on the process flow of a genetic algorithm is presented in Figure 3 for better understanding.

2.3.3. Genetic Fuzzy Systems

Soft Computing techniques are meant to operate in an environment that is subject to uncertainty and imprecision [19]. According to Zadeh, *“the guiding principle of soft computing is to exploit the tolerance for imprecision, uncertainty, partial truth, and*

approximation to achieve tractability, robustness, low solution cost and better rapport with reality”.

Soft Computing techniques such as Fuzzy Logic, Neural Networks and Genetic Algorithms have proven to provide efficient methods of solving complex problems, due to which their popularity and usage in a wide variety of application domains is unquestionable. Fuzzy Logic, with its power of incorporating expert knowledge and linguistic representation of knowledge has been successfully applied in diversified applications and fields. As mentioned earlier, Fuzzy Logic Systems have been widely used in the fields of control theory, automation and data classification to mention a few. Despite their popularity and wide usage, fuzzy systems lacked learning and adaptation abilities in previous years, as a consequence of which hybridization between the soft computing techniques started gaining popularity.

Neural Networks and Genetic Algorithms provide learning capabilities to fuzzy logic systems as can be seen in Figure 4. Moreover, all techniques within the framework of soft computing are complementary and synergistic in nature. The most popular hybridization approach is that of Neuro-Fuzzy Systems which allow fuzzy systems to learn and adapt to various environments. Another popular hybridization approach is that of Genetic Fuzzy Systems which incorporate learning capabilities in fuzzy systems via genetic algorithms. Other hybrid approaches which are relatively less visible are Fuzzy-Neural Systems, Fuzzy-Evolutionary Algorithms, Genetic Neural Networks and Genetic Bayesian

Networks. Figure 5 illustrates the hybridization approaches in the soft computing framework.

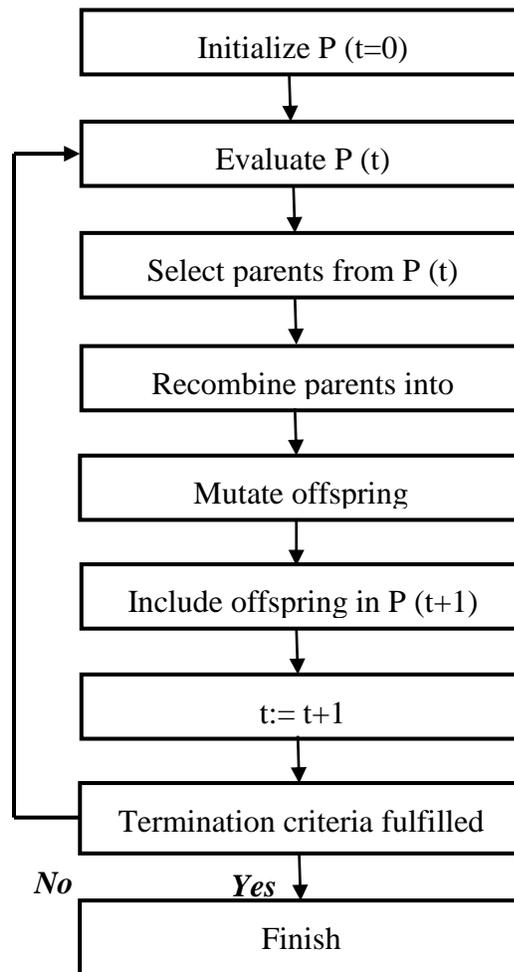


Figure 3: Process Flow in a Genetic Algorithm

A genetic fuzzy system (GFS) is essentially a fuzzy logic system supplemented by a GA based learning process. Genetic learning processes cover different levels of complexity according to the structural changes produced by the algorithm, from the simplest case of parameter optimization to the highest level of complexity of learning the rule set of a rule

based system [18]. Since, GFS deal with learning the rule set of a fuzzy logic system; they are also called Genetic Fuzzy Rule Based Systems (GFRBS). There are two main aspects in designing a GFS; firstly, determining which parts of the FRBS will be coded by the genetic model and secondly, determining the rule coding approach to be used in the genetic model. The following two sub-sections present a brief discussion about these two aspects.

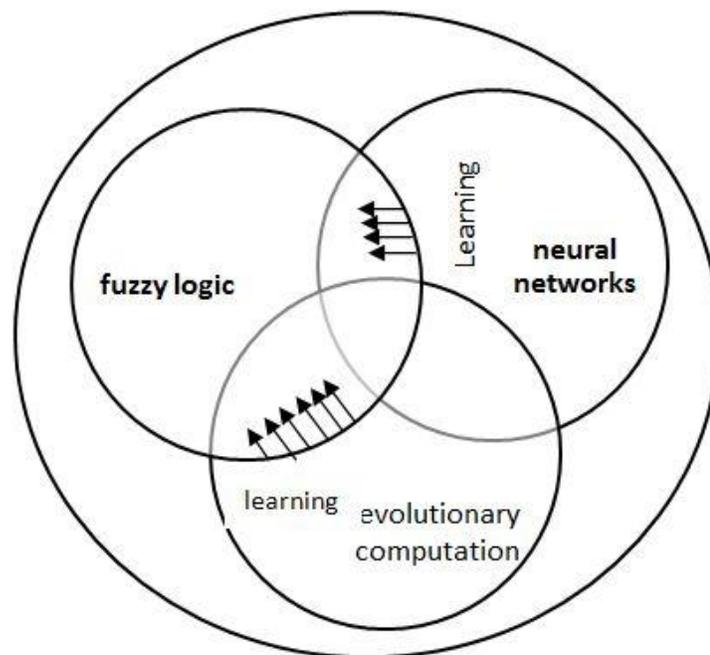


Figure 4: Soft computing and learning in Fuzzy Systems

2.3.3.1. Choice of FRBS components in genetic models

The principal factor behind using GA's for automatic learning of fuzzy rule based systems (FRBS) is that the problem of designing a rule set for an FRBS can be approached as a search problem where the focus is on finding the most optimal rule sets. The optimization eventualizes when the rule sets (more appropriately fuzzy models) are encoded as chromosomes and subjected to genetic learning. Herrera [31] states "*From the optimization point of view, to find an appropriate fuzzy model is equivalent to code it as a parameter structure and then to find the parameter values that give us the optimum for a concrete fitness function*". As such, it becomes obvious that one of the most important aspects in designing a GFS is to decide which parts of FRBS are subjected to optimization by GA. This is unfortunately not a simple task because of various concerns and tradeoffs.

An FRBS comprises of two main components; the database (DB) and the rule base (RB), collectively called the knowledge base (KB). A DB contains the definitions of the membership functions of the fuzzy sets, whereas the RB consists of the fuzzy rules (rule set). The decision pertaining to the inclusion of which parts of the FRBS should be optimized becomes a challenging issue. A tradeoff between *dimensionality* and *efficiency* needs to be resolved in order to decide with which parts of the FRBS should be included. A search space containing only the DB (sometimes RB) yields a smaller dimension, hence a faster and simple learning procedure. But the obtained solutions are not necessary optimal. On the other hand, a large search space comprising of the RB or the complete

KB leads to a higher dimensionality, hence a slower and complex learning procedure, but is more likely to produce optimal solutions. In this regards, two methodologies are available which offer some help in decision making considering the tradeoffs.

- **Genetic Tuning:** Tuning is more concerned with optimization of an existing FRBS. Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership functions [19].
- **Genetic Learning:** Learning constitutes an automated design method for fuzzy rule sets that starts from scratch. Learning processes perform a more elaborated search in the space of possible RBs or whole KBs and do not depend on a predefined set of rules [19].

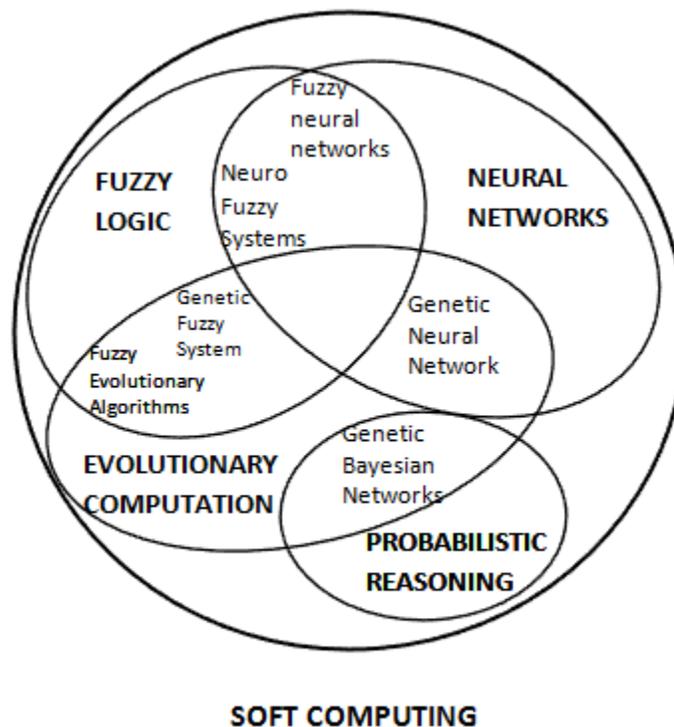


Figure 5: Hybridization in soft computing

There have been numerous works using both methodologies. Based on the comprehensive surveys by Cordon et al [18] and Herrera [31], the following approaches can be classified under either of the two methodologies.

- Genetic Tuning

- Genetic tuning of KB parameters
- Genetic adaptive inference systems
- Genetic adaptive defuzzification methods

- Genetic Learning

- Genetic KB learning
 - Genetic rule learning
 - Genetic rule selection
 - Genetic DB learning (Apriori and Embedded)
 - Simultaneous genetic learning of KB components
- Genetic learning of KB components and inference engine parameters

Furthermore, there are various proposals in each of the aforementioned approaches in the literature. A detailed explanation about all the approaches can be found in [18] [31].

2.3.3.2. Genetic Rule Coding

When it comes to genetic learning of rule sets in an FRBS, there are two major approaches that are commonly used for encoding the rules as individuals in a chromosome structure.

- Pittsburgh Approach [19]: In this approach, an individual comprises of a rule set. In other words, each individual represents a fuzzy logic system represented by a set of rules. The genetic learning proceeds by first maintaining a population of candidate solutions (rule sets representing different FLS) and then applying the various genetic operators to produce new generations of rule sets; thereby providing the most optimal rule set / FLS. In the Pittsburgh approach, “Chromosome = Rule Set”. The approach follows what is called as competition between individuals to yield the best individual (FLS). Figure 6 shows the pictorial representation of the Pittsburgh Approach.

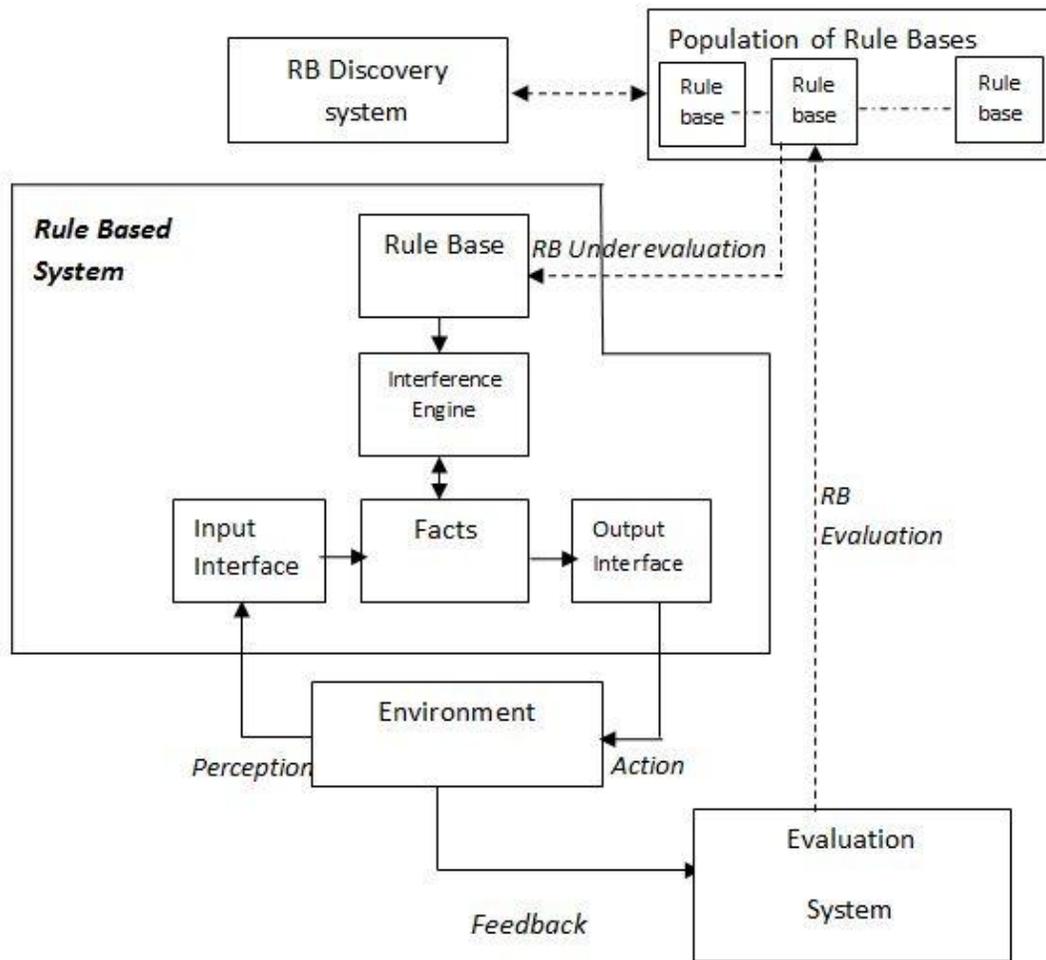


Figure 6: Genetic Learning using Pittsburgh Rule Encoding Approach

- Michigan Approach [19]: In this approach, an individual represents a single rule. Since each individual represents a rule, an entire population of individuals constitutes an FLS. Genetic learning proceeds by maintain a population of individuals (rules) and then applying selection, crossover and mutation to produce new generations of rules, thereby providing the most optimal population (FLS). In the Michigan approach, “Chromosome = Rule”. The approach follows

what is called as cooperation among individuals to produce the best population (FLS). Figure 7 represents the Michigan Approach.

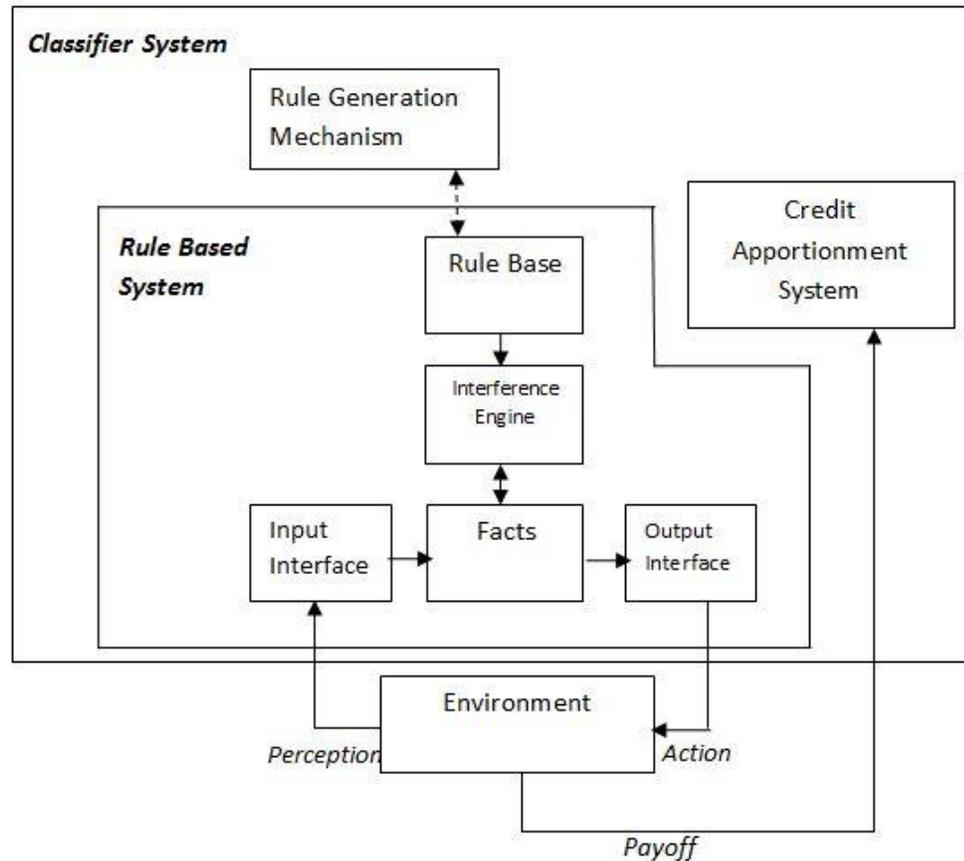


Figure 7: Genetic Learning with Michigan Rule Encoding Approach

Apart from the two aforementioned approaches, there are other approaches which have been used for genetic encoding of rules/rule sets in FRBS such as the '*Iterative Rule Learning*' [18] and '*Genetic Cooperative-Competitive Learning*' [18].

CHAPTER 3

LITERATURE REVIEW

In our bid to carry out a critical survey of the literature on using Use Cases for software development effort prediction, we discovered that a common ground for assessing and comparing these prediction techniques (see Section 3.1) was not available. Though a few related works are available, there is no significant contribution which explicitly offers an evaluation criterion for comparison and evaluates the proposed Use Case based metrics on a common platform [8][24][74][85]. Boehm [12] presented a set of useful criteria (attributes) for evaluating the utility of software cost models. The attributes targeted model-based estimation methods. Similarly, Saliu and Ahmed in their chapter of the book “Soft Computing in Software Engineering” [80] proposed a set of attributes; theirs targeted soft computing-based effort estimation models though. As such, no criteria were developed to target use case-based models. The primary goal of conducting the critical survey is to fill the void caused by the unavailability of such literature which can help practitioners in selecting appropriate metrics for their respective development efforts and

also guide researchers interested in developing new metrics in this domain. Accordingly, we identified a set of comparison attributes to be used in assessing and comparing various use case-based approaches for effort prediction which resulted in a published critical survey [40][41].

3.1. Use-Case based Effort Prediction Techniques

This section presents a brief survey about the various use-case based effort prediction techniques available in the literature.

3.1.1. Use Case Points

The basic technique proposed by Gustav Karner [42] for estimating effort based on Use Cases. The method assigns quantitative weights to actors based on actor classification as simple, average and complex. The sum of all the weighted actors in the system gives the Unadjusted Actor Weight (UAW). Similarly, Use Cases are classified according to their complexity and are assigned quantitative weights. The sum of all the Use Cases in the system gives the Unadjusted Use Case Weight (UUCW). The sum of UAW and UUCW gives the Unadjusted Use Case Points (UUCP). Then, a number of technical complexity factors and experience factors are weighted and are multiplied to the UUCP to yield Use Case Points (UCP). Finally, the obtained Use Case Points are multiplied by the Productivity Factor PF to give the final Effort Estimate. Critics claim Karner's method to be decent with the exception of the non-flexibility in adjusting the Productivity Factor which was later proved to be a major variable affecting the estimation process.

3.1.2. Transactions

A metric proposed by Robiolo et al [73] for estimating size of software based on the size of Use Cases. It depends on the textual description of a Use Case. A Transaction is defined by a stimulus by the Actor and response by the system. The sum of all the stimuli is the number of Transactions in a particular Use Case. Summing up the transactions for all the use cases in the entire system, the number of Transactions is calculated. In order to estimate the final effort, the Historical Mean Productivity technique was used by the authors [73]. Three major objectives using this metric and the following metric “Paths” were highlighted by the method which are simplifying the counting method, to obtain different units of measurement that individually may capture a single key aspect of software applications and reducing the estimation error.

3.1.3. Paths

Another metric proposed by [73] which pursues similar objectives as the “Transaction” metric. It is based on the concept of Cyclomatic complexity which identifies binary and multiple decisions in code. The same idea has been applied in terms of textual descriptions of Use Cases. The method is as follows; obtaining the complexity of each transaction. For obtaining the complexity of each transaction, first count the number of binary decisions, then identify the multiple decisions by counting the different pathways and subtract one from the number obtained. In the final step, for computing the complexity of each uses case, sum up the complexity value for each transaction.

3.1.4. Extended Use Case Points

The EUCP method proposed by Wang et al [22] contains three parts; first, refining the Use Case classification with fuzzy set theory. Second, using a learning Bayesian Belief Network BBN for getting the Unadjusted Use Case Points UUCP probability distribution. Third, using a BBN for generating the effort probability distribution derived from UCP. The contribution of this approach is a probabilistic cost estimation model obtained by integrating fuzzy set theory and Bayesian belief networks with the generic UCP method.

3.1.5. UCPm

UCPm is a slight modification of the Use Case Points method proposed by Sergey Diev [21]. The method stresses more on defining Actors and Use Cases comprehensively. The slight change from the basic UCP method is the calculation of the size of the software product. The “UUCP” obtained is multiplied with the technical complexity factor “TCF” to give the size of the software product. To the size, environmental factor “EF”, base system complexity factor “BSC” and pre-defined number of person-hours per use case point “R” are multiplied. Finally, supplementary effort factor is added to yield the final effort estimate of the software product. The supplementary effort may include activities like writing configuration management scripts or performing regression testing.

3.1.6. Adapted Use Case Points

The basic objective of this method proposed by Mohagheghi et al [64] is to develop a technique which fits the incremental model of software development and in situations

where requirements specifications are frequently changed. The method follows the structure of the UCP method but with major differences. All actors are assumed to be average without differences in classification. All the Use Cases are assumed to be complex and then later on they are decomposed to smaller use cases and classified as simple or average. The method includes the extended use cases as well and counts them as base use cases. Exceptional flows are also counted as average use cases. The method has very promising results and the major contributions are the adaptation of the UCP method for incremental development and identifying the impact of effort distribution profile on effort estimation results.

3.1.7. Use Case Size Points

This metric proposed by Braz and Vergilio [13] focuses on the internal structures of the Use Cases in depth and hence better captures the functionality. The primary factors considered in this metric are the Actors classification, pre-condition classification and post-condition classification, main scenarios, alternate scenarios, exception classification and the Adjustment Factor. The sum of all these factors gives the Unadjusted Use Case Size Points UUSP which is subsequently multiplied by the difference of the technical complexity factor and the experience factor. The results are compared with Function Points and UCP metrics.

3.1.8. Fuzzy Use Case Size Points

Another metric proposed by Braz and Vergilio [13]. The primary factors considered in this metric are the Actors classification, pre-condition classification and post-condition classification, main scenarios, alternate scenarios, exception classification and the Adjustment Factor. The sum of all these factors gives the Unadjusted Use Case Size Points UUSP which is subsequently multiplied by the difference of the technical complexity factor and the experience factor. The difference between USP and FUSP is in the use of the concept of Fuzzification and Defuzzification. This creates gradual classifications that better deal with uncertainty. Also, it reduces the human influence on the classification of the Use Case elements. The results obtained using this metric are slightly better than the Use Case Size Points metric.

3.1.9. Simplified Use Case Points

The main aim of this method proposed by M. Ochodek et al [70] is to simplify the UCP method and the process of Effort Estimation in general. This is not a completely defined metric. The approach used for realizing the objective is the cross validation procedure, which compares different variants of UCP with and without certain factors. Factor Analysis was also performed to investigate the possibility of reducing the adjustment factors. The results from this study include recommending a metric based on rejection of actor weights and rejection of 9 Technical Complexity Factors and 6 Experience Factors.

3.1.10. Industrial Use Case Points

The IUCP method proposed by Edward Carroll [14] is not a defined metric but an amalgamation of different industrial practices used in association with the UCP method to increase the accuracy and reliability of the estimation procedure. The main contribution of this method is the inclusion of the Risk Factor and additional effort for activities other than the development of the software product. Also, in depth analysis of few factors like Performance Analysis, Deliverable Analysis, Schedule Analysis, Defect Analysis, Causal Analysis and Quantitative Management Analysis is mentioned. The importance of using a Process Improvement Cycle is also highlighted.

3.2. Comparison Criteria

To compare the various proposed prediction techniques, we developed a comparison criteria consisting of eleven attributes, which were chosen carefully to accommodate all the pros and cons of using those techniques. A point worth mentioning is that “*Transparency*” was not included in the critical survey by Kamal et al [40]. Since, “*Transparency*” has a significant role in developing dependable and efficient effort prediction models; we included it in our complete comparison criterion. The qualified comparison attributes and their descriptions are presented in the following sub-sections.

3.2.1. Accuracy

The degree of precision or correctness obtained in estimating the effort with reference to a particular approach is termed as Accuracy. It is basically obtained by comparing the effort estimated with the actual effort and checking for deviations. A higher accuracy of an approach validates the efficiency of that approach. Better accuracy implies better reliability [1]. It should be noted that comparing estimation accuracy of various approaches is not easy pertaining to reasons such as different datasets, different definitions of similar terms and different goals of estimation accuracy [26].

3.2.2. Ease of Use

This attribute implies simplicity of use. How easy it is to use a particular technique/approach? A fact that should be understood is that, the effort required in estimating effort for software development should be minimal. What is the use of a technique which itself requires a lot of time and effort? [41]. Preferably, the approach used should be simple enough to be implemented in a reasonable time frame as Bente Anda [8] states that the UCP method requires little technical insight and effort and hence makes it easy to use in early stages.

3.2.3. Use Case detail considerations

The level of detail considered in evaluating a particular Use Case before using it in the estimation process is important for various reasons. Issues like the granularity of Use

Cases, number of scenarios in a Use Case, inclusion of Extended Use Cases with the Base Use Cases, classification of Use Cases as simple and complex are commonly debated among various researchers for the Use Case based estimation methods [21][64][85]. This is a valuable attribute for comparing the different approaches related to Use Case based methods.

3.2.4. Factor Inclusion

The effort estimation calculated using the basic UCP method considers various Experience factors and Technical Complexity factors [42]. The variety of other Use Case based approaches we have considered, discard few of these factors and consider them unrequired for the estimation process, whereas few of the approaches consider some additional factors [64][70]. The attribute will help in analyzing the approaches and contribute in specifying the optimal factors to be considered in the estimation process.

3.2.5. Adaptability

The capability of the model or method to adjust according to new environments and fit the incremental style of development practices is termed as Adaptability of the model. “Incremental or evolutionary development approaches have become dominant. Requirements are changed in successive releases, working environments are shifted and this has been accepted as a core factor in software development” [64]. A method or a

model should be adaptive to these changes and if it is otherwise, then the model will have limited usability value.

3.2.6. Handling Imprecision and Uncertainty

Quite a common aspect in all the software development practices is to take account of the imprecision and uncertainty associated with the processes. We know that there is a reasonable imprecision in estimating the size of software and a lot of uncertainty in predicting various factors associated with developing software [67]. A model which considers these factors is better than a model which doesn't.

3.2.7. Sensitivity

The receptiveness or responsiveness to an input stimulus is called sensitivity. In terms of software development, a model in which the change in estimated effort with respect to a small change in the input values is large or significant is termed as a sensitive model. In Effort Estimation, it is desirable to have low sensitivity models.

3.2.8. Transparency

The visibility of the underlying effort prediction model is termed as transparency. It is desirable to have transparent models as it would provide the experts the ability to

incorporate their opinions based on their knowledge and experience. Empirical research studies have shown prediction models coupled with expert opinions to be better than the prediction systems or the expert alone [57].

3.2.9. Appropriate use of Productivity Factor

The conversion of estimated points based on Use Cases to Effort requires the multiplication of a factor called productivity factor whose units are person-hours. Initially, Karner [42] proposed a productivity factor value of 20 person-hours, which later turned out to be variable for different projects. An appropriate use of the productivity factor results in close to accurate estimations and reduces the deviations. This is a valuable attribute to distinguish between the available approaches.

3.2.10. Artifacts Considered

This attribute reflects the artifacts that are considered in the implementation of a particular technique or metric. Effort Estimation using Use Cases considers all the functional requirements in a satisfactory way, but a major complaint against the use of this method is that the non- functional requirements are not considered extensively. But, if the artifacts pertaining to non-functional requirements like estimating for reports, schedule spreadsheets, staffing concerns are considered [14], then the method could have

a valid defense. The use of artifacts considered by different models is helpful in comparing them.

3.2.11. Empirical Validations

The evaluation and validation of a metric or a model in general is essential. If the model is validated, then the validation criteria and the dataset on which it is validated are considered. Datasets from the industry are considered more reliable than student datasets or datasets from open sources [1]. The empirical validation of a model adds to its credibility as well.

3.3. Comparison of Prediction Techniques

The comparisons have been presented in tabulated form for sake of simplicity and ease of understanding. Each table is followed by a short discussion which summarizes the tabulated information and provides recommendations for the use of certain techniques with respect to the attributes. It is noteworthy that subjective ratings have been used for comparing the various techniques.

3.3.1. Accuracy

Table 2: Evaluation of techniques based on ‘Accuracy’

Metric	Comments
UCP [42]	Relatively good accuracy and promising results. More accurate than expert estimates in few cases and almost equally accurate in some other cases.
Transactions [73]	Good accuracy, close to UCP, lower variability of prediction error, high correlation with actual effort.
Paths [73]	Better accuracy than Transactions and UCP, lower deviation from actual effort, high correlation with actual effort.
EUCP [22]	Better accuracy than UCP as they use Fuzzification and a Bayesian Belief Network to train the system.
UCPm [21]	Relatively good accuracy, less calculations required in the method.
AUCP [64]	Very good accuracy, effort calculated using AUCP for release 1 and release 2 were 21% and 17% lower than Actual Effort.
USP [13]	Competent accuracy compared to others, but lower error rates.
FUSP [13]	Competent accuracy results with lower error rates, a fuzzified form of USP with minor changes in results.
SUCP [70]	Slight improvement in accuracy. Discarding TCF and EF doesn't cause a negative effect in prediction of effort.
IUCP [14]	Perhaps the most efficient and accurate results. Using the process improvement loop, the deviation in prediction has been cut down to 9%, which is a very significant contribution.

Discussion: Even after evaluating all metrics based on their respective results, terming a certain metric better than others is not justified because of many reasons such as different data sets used, differences in the nature of the software projects, environmental and expertise differences, etc. Nevertheless, it is recommendable to use metrics which use machine learning techniques like FUSP. Additionally, the use of industrial practices in the estimation process improves the accuracy of the method. Hence, the use of IUCP is also recommendable.

3.3.2. Ease of Use

Table 3: Evaluation of techniques based on ‘Ease of Use’

Metric	Comments
UCP [42]	Very easy to compute effort using UCP. It can be done at the early stages of the development of the life cycle. A rough estimate can also be made just by mental calculation.
Transactions [73]	An easy method involving counting the number of transactions in each Use Case and subsequently the total in a system.
Paths [73]	A relatively complex method to use, involving obtaining the complexity of a transaction by summing up the number of binary decisions and identification and summing up of multiple decisions.
EUCP [22]	A complex method involving fuzzifying the inputs and training the Bayesian Belief Network for estimating effort and consequently defuzzifying the output to obtain a crisp value.
UCPm [21]	An easy method, almost similar to UCP; the only difference being size is calculated as the product of Unadjusted Use Case Weights and the sum of Technical Complexity factors.
AUCP [64]	A complex method compared to other approaches. Involves computing modified Unadjusted Use Case Weights and uses many additional factors such as Adaptation Adjustment Factor (AAF), and Equivalent Modification Factor (EMF) which itself comprises of 6 other factors.
USP [13]	A fairly simple method to calculate the effort. Only lengthy part is to consider the details of use cases and classify them appropriately.
FUSP [13]	A simple method, slightly complex than USP because of the Fuzzification of inputs and Defuzzification of outputs respectively.
SUCP [70]	A method simpler than conventional UCP, this reduces the number of Technical Complexity Factors and Experience Factors by limiting them to 6 only.
IUCP [14]	A simple method similar to UCP, with the additional overhead of calculating for non-functional requirements like documenting reports, spread sheets, etc.

Discussion: Almost all the metrics are subjectively rated equally in terms of ‘Ease of Use’, with the exception of Paths and AUCP metrics. It is intuitive that since the basic UCP method is quite simple in terms of use, a metric or method which deviates from the norms and structure of the basic method is bound to be relatively complex. Though the EUCP method is mentioned as complex, the rationale can be to consider the metrics which

use soft computing methods as relatively more time consuming rather than terming them as complex to use. We recommend SUCP as the metric easiest to use compared to the others with UCP coming a close second.

3.3.3. Use Case Detail Considerations

Table 4: Evaluation of techniques based on ‘Use Case Detail Considerations’

Metric	Comments
UCP [42]	Only considers the complexity classification of a Use Case by counting the number of transactions in a Use Case. Classified as simple, average and complex.
Transactions [73]	Considers only the stimulus by an actor and response by the system, by counting the number of transactions. No other details are considered.
Paths [73]	Identifies binary and multiple decisions in a Use Case. Sums up the number of binary and multiple decisions in a Use Case and consequently for the entire system. No other details are considered.
EUCP [22]	The Use Case classification is refined by considering detailed aspects of a Use Case such as User Interface Screens, pre-conditions, primary scenario, alternative scenario, exception scenario, post-conditions.
UCPm [21]	High level of detail is considered. Scoping of actors, classification of Use Cases as zero weight use cases, duplicated use cases, background process use cases, report use cases. Also considers the granularity of use cases.
AUCP [64]	Initially all Use Cases as considered complex, then are broken down to simple and average based on transactions. Include extended Use Cases as base Use Cases and exceptional flows in a Use Case are also assigned a weight factor of 2.
USP [13]	A detailed classification comprising of pre-conditions, post-conditions, main scenarios, alternate scenarios and exceptional scenarios.
FUSP [13]	The Use Case detailed classification comprises of pre-conditions, post-conditions, main scenarios, alternate scenarios and exceptional scenarios.
SUCP [70]	Considers the complexity classification of a Use Case by counting the number of transactions in a Use Case. Additionally, the cardinality of Use Cases is computed.
IUCP [14]	Similar to UCP, IUCP does not consider any extra Use Case details except the complexity classification.

Discussion: This is perhaps a very important and valuable attribute for distinguishing the strengths and weaknesses of the available metrics. Majority of the metrics base their calculations of size on the number of transactions in a Use Case without considering other details related with use cases. If the metrics were to be ranked according to this attribute or recommended on this basis, Use Case Size Point ‘USP’ would win the

evaluation followed by UCPm and AUCP. The reason for this ranking is quite visible in the tabulated information. USP considers almost all the details associated with a Use Case. UCPm takes it to a further level by classifying use cases by varying levels but misses including the pre-conditions and post-conditions.

3.3.4. Factor Inclusion

Table 5: Evaluation of techniques based on ‘Factor Inclusion’

Metric	Comments
UCP [42]	Includes Actor weights and Use Case weights. Also includes 13 Technical Complexity Factors and 8 Experience Factors.
Transactions [73]	No use of Actor weights and Use Case weights. Does not include any Technical Complexity Factors and Experience Factors.
Paths [73]	No use of Actor weights and Use Case weights. Does not include any Technical Complexity Factors and Experience Factors.
EUCP [22]	Includes Actor weights, Use Case weights, 13 Technical Complexity Factors and 8 Experience Factors.
UCPm [21]	Includes Actor weights, Use Case weights, 13 Technical Complexity Factors, 8 Experience Factors. Additionally, UCPm includes Base System Complexity factor and Supplementary Effort Factor.
AUCP [64]	Actor Weights and Use Case weights are included. All the Technical Complexity Factors and Experience Factors are discarded. Includes new factors such as Adaptation Adjustment Factor (AAF), Equivalent Modification Factor (EMF), and Overhead Factor (OF).
USP [13]	Actor weights and Use Case weights are included as per the detailed Use Case classification. Additionally, 14 Technical Complexity factors and 5 Environmental Factors are included.
FUSP [13]	Actor weights and Use Case weights are included. 14 Technical Complexity Factors and 5 Environmental Factors are included.
SUCP [70]	Discards Actor weights and includes only Use Case weights. 9 out of 13 Technical Complexity factors and 6 out of 8 Experience Factors are discarded.
IUCP [14]	Includes Actor weights and Use Case weights. Also includes 13 Technical Complexity Factors and 8 Experience Factors.

Discussion: Perhaps the most debated attribute which can involve lot of future work. The issue is to find the optimum number of factors that are to be considered while estimating effort. Many metrics agree with the standardized thirteen technical complexity factors and the eight experience or environmental factors as proposed by the basic UCP method. SUCP discards nine technical complexity factors and six experience factors. UCPm keeps all the standard factors same but includes additional factors. Few metrics like

Transactions, Paths and AUCP discard all the standardized factors but the latter makes up for the non-inclusion by using new factors such as AAF, EMF and OF. As such, we cannot recommend any metric to be the best in terms of this attribute.

3.3.5. Adaptability

Table 6: Evaluation of techniques based on ‘Adaptability’

Metric	Comments
UCP [42]	Very simple and adaptable method. Fits any Use Case modeling environment easily.
Transactions [73]	An adaptable method, worked well with 13 different projects under different environments. Fits the dynamic model of software development. Only needs counting the number of transactions.
Paths [73]	Fairly adaptable. Depends on calculating the complexity of Use cases. Slight difficulty expected in adapting to environments with less experienced teams.
EUCP [22]	Less adaptive as compared with other metrics because of the involvement of the training BBN.
UCPm [21]	Fairly adaptable to different environments. Difficulty with less experienced teams for estimating effort.
AUCP [64]	Perhaps the most adaptable metric. The aim of realizing this metric was to fit the incremental model of development and support environments where Extreme Programming is used.
USP [13]	Slightly less adaptable relatively. The adjustment factors need to be calibrated with each and every changing project and environment.
FUSP [13]	Same as the USP method. Less adaptable relatively.
SUCP [70]	Adaptable in many environments. Applied to 14 industrial and academia projects with relative ease and promising results were obtained. Removal of few factors supports adaptability.
IUCP [14]	A very adaptable metric, perhaps because of the feedback loop and its ability to fit into any mode of operation and environment. The metric has been custom designed to fit any model of development.

Discussion: Almost all metrics qualify well for this attribute. Few of them are more adaptable in terms of their structure, ease of use and lesser difficulty with new and inexperienced teams. An interesting observation is that, the use of soft computing methods like in the case of EUCP, where a learning Bayesian Belief Network is incorporated in the estimation process, it made the metric relatively less adaptable to different working environments. But the validity of this observation can be debatable. AUCP is the most recommended metric in terms of Adaptability.

3.3.6. Handling Imprecision and Uncertainty

Table 7: Evaluation of techniques based on ‘Handling Imprecision and Uncertainty’

Metric	Comments
UCP [42]	Doesn't handle imprecision, though it manages to deal with uncertainty up to some extent.
Transactions [73]	Doesn't handle imprecision nor uncertainty.
Paths [73]	It is not designed to handle imprecision and uncertainty.
EUCP [22]	Handles imprecision and uncertainty fairly because of the use of Fuzzy logic and additionally because of the learning Bayesian Belief Network.
UCPm [21]	Not capable of handling imprecision and uncertainty.
AUCP [64]	Does not handle imprecision, but the metric deals with uncertainty satisfactorily.
USP [13]	Is not capable of handling both imprecision and uncertainty.
FUSP [13]	The fuzzified version of USP, and hence it handles imprecision and uncertainty quite well.
SUCP [70]	Does not handle imprecision, nor does it handle uncertainty.
IUCP [14]	A metric tailored to deal with uncertainties but cannot handle imprecision.

Discussion: Another important factor for evaluation. It is much desirable that in a process like estimation of effort and cost where loads of uncertainty is possible and imprecise estimates are quite common, a metric should account for both the afore-mentioned factors. Unfortunately, most of the metrics don't account for both imprecision and uncertainty. Few of them such as UCP, AUCP and IUCP are capable of dealing with uncertainties but not imprecision. EUCP and FUSP, since they use soft computing techniques account reasonably well for both imprecision and uncertainty and are recommended for use.

3.3.7. Sensitivity

Table 8: Evaluation of techniques based on ‘Sensitivity’

Metric	Comments
UCP [42]	The metric is less sensitive to input changes. Can accommodate noise reasonably well.
Transactions [73]	Is less sensitive to changes. A small change to the input i.e. the increase or decrease in the number of transactions of a Use Case will not adversely impact the effort estimated.
Paths [73]	Is moderately sensitive when compared to Transactions metric. If the Use Case details are changed, the number of binary decisions and multiple decisions change considerably. This affects the final estimated effort.
EUCP [22]	Less sensitive because of the Fuzzification and Defuzzification process. Accommodates noise levels easily.
UCPm [21]	Less sensitive as the input factors don’t impact the final estimated effort much.
AUCP [64]	A moderately sensitive metric. AUCP incorporates many factors because of which, a slight change in some factors may result in considerable changes to the final estimated effort.
USP [13]	Less sensitive to changes.
FUSP [13]	A slightly less sensitive metric than the USP. It accounts for varying levels of input changes.
SUCP [70]	A lesser sensitive metric. Almost similar to the conventional UCP metric.
IUCP [14]	Not sensitive to input changes. Works the dynamic way and hence accounts for changes anywhere in the process lifecycle.

Discussion: A much desirable attribute for comparison in many fields and not just effort estimation, Sensitivity like ‘Use Case Details Consideration’ can distinguish between metrics in a very proper way. Unfortunately, it is very difficult to distinguish between the available metrics because of lack of information related with the sensitiveness of the metric inputs and outputs. Nevertheless, few metrics have been classified as lowly sensitive and moderately sensitive. It is worth noting that, using soft computing approaches can minimize the sensitivity of a metric considerably. The IUCP can be recommended for use if Sensitivity is the main concern.

3.3.8. Transparency

Table 9: Evaluation of techniques based on ‘Transparency’

Metric	Comments
UCP [42]	UCP is not transparent. The equations of the UCP method don’t give any idea about the way UCP is calculated. As such experts cannot calibrate the factor values of UCP
Transactions [73]	Not transparent. The calculation of size is based on the number of transactions and the final effort is calculated based on Historical Mean Productivity.
Paths [73]	Not transparent. The calculation of size is based on the number of paths and the final effort is calculated based on Historical Mean Productivity.
EUCP [22]	Not transparent. Even though EUCP uses the Bayesian Belief Network for training the prediction system, the visibility of the underlying process is minimal.
UCPm [21]	Not transparent enough. Just allows the expert to calibrate few factors but as a whole the effect of calibrating those factors cannot be determined.
AUCP [64]	AUCP is not transparent, as it follows the UCP method and its associated equations with few modifications.
USP [13]	Not transparent. All the use cases are classified and size is calculated based on training from the historical data.
FUSP [13]	Not transparent. The size and effort are calculated based on historical data.
SUCP [70]	Not transparent. Doesn’t allow for any calibrations within the process.
IUCP [14]	IUCP is not transparent. It has the basic equations of the UCP method and only adopts few additional industrial practices, which don’t account for transparency.

Discussion: Transparency is a very important factor in effort prediction processes. A metric or a method can be termed as fully transparent if its underlying model is clear enough to be understood and allows the experts to calibrate the input values while knowing what the corresponding results will be obtained. But unfortunately, none of the metrics have taken into account this factor.

3.3.9. Appropriate Use of Productivity Factor

Table 10: Evaluation of techniques based on ‘Appropriate Use of Productivity Factor’

Metric	Comments
UCP [42]	Karner described the method and fixed the productivity factor at 20 man-hours per Use Case Point.
Transactions [73]	Effort calculation is based on Historical Mean productivity technique. No involvement of Productivity Factor.
Paths [73]	Effort Estimation is based on Historical Mean productivity technique. No involvement of Productivity Factor.
EUCP [22]	Not much use of the productivity factor. All the calculations are based on adjusting other factors.
UCPm [21]	Uses the productivity factor specified by the conventional UCP method.
AUCP [64]	Productivity factor of 36 man-hours per Use Case is used in addition to other adjustment factors such as AAF, EMF and OF. In case of the overhead factor (OF) not being used, the use of 72 man-hours as productivity factor has been prescribed.
USP [13]	A productivity factor of 26 man-hours is used as per the calculations.
FUSP [13]	Productivity factor of 26 man-hours has been used.
SUCP [70]	Productivity factor of 20 man-hours, 28 man-hours and 36 man-hours has been used as per the requirement of the project under consideration which is appropriate.
IUCP [14]	Productivity factor of 20 man-hours and 28 man-hours has been used as other adjustments are taken care of by the risk adjustment factor and factors like estimating for reports.

Discussion: With respect to Use Case based effort estimation, this attribute has a vital contribution in the comparative analysis. Earlier when the estimation of effort based on use cases was in its infancy, there were quite significant variations in estimated effort even though the technical complexity factors and experience factors were properly adjusted. The reason which came in the focus after many years was the inappropriate use of Productivity Factor. Since, Karner proposed a 20 person-hour per use case; it was not changed for quite some time until variations with it resulted in more accurate effort

estimates. SUCP can be recommended for use as it allows variable use of the Productivity Factor with respect to the project. The use of IUCP is also recommended as it provides freedom to the estimators for selecting the appropriate Productivity Factor.

3.3.10. Artifacts Considered

Table 11: Evaluation of techniques based on ‘Artifacts Considered’

Metric	Comments
UCP [42]	Does not take into account any additional artifacts.
Transactions [73]	Does not consider any additional artifacts. Deals with the functional requirements only.
Paths [73]	No consideration of additional artifacts.
EUCP [22]	No additional artifacts considered.
UCPm [21]	No additional artifacts are considered.
AUCP [64]	Considered artifacts related to non-functional requirements of the process lifecycle like availability, performance and security.
USP [13]	No consideration of additional artifacts.
FUSP [13]	No additional artifacts are considered.
SUCP [70]	Additional artifacts are not considered.
IUCP [14]	A lot many artifacts have been considered by the IUCP metric. Artifacts like estimating for reports, risk management artifacts, artifacts dealing with performance analysis, deliverable analysis, schedulable analysis and defect analysis are considered.

Discussion: In terms of this study, artifacts imply the inclusion of non-functional requirements in the effort estimation process. As tabulated in the above tables, most of the metrics do not consider any additional artifacts with the exception of the AUCP and the IUCP. AUCP considers important non-functional requirements such as performance and security. IUCP also considers non-functional requirements in addition to including lesser effect artifacts such as Reports documentation etc. As such, both AUCP and IUCP are recommended for use.

3.3.11. Empirical Validations

Table 12: Evaluation of techniques based on ‘Empirical Validations’

Metric	Comments
UCP [42]	Many empirical validations are available for the use of traditional UCP approach. Many authors have validated the UCP procedure empirically using both Industry datasets as well as Student datasets.
Transactions [73]	Empirically validated using datasets comprising of 13 small business projects distributed across 3 different contexts; an Undergraduate Academic Environment, System and Technology Department at Austral University and a level 4 CMM certified company. The projects are also distributed implementation wise as well.
Paths [73]	The same datasets used to validate the Transactions metric were used.
EUCP [22]	Validated using two industry projects in a Chinese company of 500 employees. Since results show some inconsistency, more evaluation needs to be done with the metric.
UCPm [21]	Not validated using any dataset. The proposed metric is a result of analysis carried out over 50 projects in a period of 2 years as reported.
AUCP [64]	The results of applying this metric were validated using a telecom project of Ericsson and across 2 releases. The authors report more case studies that validated the AUCP metric but information about them has not been specified explicitly.
USP [13]	A case study was done to validate the results of this metric using a real project database of a private company. The metric was validated against Function Points and traditional UCP.
FUSP [13]	Same case study as was used by the USP metric. FUSP was validated against Function Points, traditional UCP and USP itself. Differences between USP and FUSP were also highlighted. The use of these metric needs more validations and more experiments needs to be done.
SUCP [70]	Empirically validated against 7 industrial projects and 7 other projects from the Poznan University of Technology. The range of the actual effort was 277 man-hours to 3593 man-hours. Promising results were obtained. Additionally, a framework was built to evaluate the estimation accuracy of all the 14 projects using this metric.
IUCP [14]	The metric has been validated over a continuous period of 5 years, consisting of 200 projects in a CMM level 5 company. The results are astonishing as the feedback loop helped in reaching 9% deviation with reference to the Actual Effort for 95% of the company’s projects.

Discussion: The attribute where in all the metrics are on par with each other. It is interesting to note that all the metrics have been extensively validated using Industrial

data sets. As such, we cannot underestimate the evaluations of the proposed metrics in any manner.

3.4. Analysis

Based on the critical literature review and after drawing comparisons between the various Use Case based metrics on a common ground, several shortcomings arose which were anticipated. The comparison brought forth many weak links in the Use Case based estimation process and at the same time highlighted many advantages of using it. The following analysis is based on the evaluation attributes used in the comparison.

Nearly all the metrics have been validated using either industry datasets or student datasets. This is an onus for the validity of the efficiency and accuracy of the metrics. This is well complemented by the fact that most of them have competent and reliable effort estimates. Most of the proposed metrics are easy to use which makes them more liable to be favored over other techniques and metrics which provide similar results. Adaptability, in terms of usage of the metrics is noteworthy considering that almost all metrics qualify as being fairly adaptable and the case studies involving them verify the fact. Few metrics consider detail classification of the Use Cases with respect to complexity by considering all the aspects related to the implementation of Use Case. Metrics which capture the details are definitely more useful and efficient than metrics which do not consider detailed classification. Also, the inclusion and exclusion of the technical complexity factors and experience factors showed varied results. Mostly, it was generalized that the exclusion of few factors does not have negative impact on the estimation of effort. Many metrics considered the technical complexity factors to be overlapped and hence discarded many such factors.

Sensitivity is an attribute which could not be properly addressed in the comparison. It is due to the fact that enough information was not available to distinguish the metrics from being highly sensitive and lowly sensitive. It is desirable to have metrics and techniques which have low level of sensitivity. Based on our comparison, few metrics were found to be lowly sensitive and few moderately sensitive. Productivity factor is an important concern while estimating effort using Use Cases. It is an important contributor for the conversion of the metric in terms of size to effort. Appropriate use of this factor affects the final estimated results. The degree of correlation between estimated effort and Actual effort can be established satisfactorily if the productivity factor is rightly used. Most of the proposed approaches don't consider the importance of this factor and focus more on other adjustment factors. One of the most important weaknesses of Use Case based approaches was the non-consideration of the non-functional requirements associated with software development processes. Though few metrics attempted to incorporate the artifacts pertaining to non-functional requirements, it is not enough. Any software process depends on both functional and non-functional requirements. A metric or technique which does not consider additional artifacts will have varying levels of deviation in the estimated effort.

The two most important and perhaps the negative factors in terms of using Use Case based metrics are the non-transparency of effort prediction processes and the inability to deal with imprecision and uncertainty. These two attributes show the vulnerability of the Use Case based approach when compared with other approaches. Transparency in effort prediction processes is a major issue as it reflects the visibility of the prediction process

to experts and software engineers. Collaboration between the experts and the prediction system is highly recommended as the experts can use their knowledge and experience to improve the prediction process. Usually, prediction systems coupled with expert opinions are more mature and better off than the standalone prediction systems [57]. Most of the compared metrics do not account for imprecision with the slight exception of the metrics using fuzzy logic and other machine learning techniques. With the prediction processes accommodating expert opinions, the imprecision only increases. It is desirable to have prediction systems that can handle imprecision. Fuzzy Logic can be employed to handle such imprecision. Uncertainty, however, did not seem to have caught enough attention; future research is needed to consider the uncertainty associated with measurements provided by the different metrics.

The important requirement is that the negative aspects which expose the vulnerability of Use Cases should be addressed. In the same context, if a standardized approach is established to write Use Cases, many issues would be minimized. Alternately, each organization can come up with their own standards of writing Use Cases and keep a check on the standards so that, the estimation process can be generalized using Use Cases. Lastly, using the process improvement lifecycle as a feedback loop to learn and incorporate efficient techniques should be prescribed by organizations so as to reap the benefits of efficient and accurate effort prediction. Causal Analyses and Quantitative Management Analysis of the reports documented should be carried out on a periodic interval to ensure continuous improvement.

Despite few shortcomings and negative aspects, the detailed comparison and evaluations support the fact that predicting effort using use-cases is justified and that they can be successfully used in the software effort prediction process. The primary aspect that strongly justifies the use of use-cases for software development effort prediction is the early availability of use-cases in the software development life cycle. This is a value adder in terms of the effort prediction process in the sense that it is desirable to have effort prediction models which can aid in the early prediction of effort. Moreover, the applicability of fuzzy logic can help in evolving transparent and adaptive effort prediction models capable of handling imprecision and incorporating expert opinions, thereby helping in overcoming the majority of the shortcomings as deduced from the outcomes of the literature survey.

CHAPTER 4

RESEARCH APPROACH AND PROPOSED FRAMEWORKS

This chapter follows up the discussion from the previous chapters and provides us the motivation for the work, various proposals and technical details pertaining to the proposed frameworks.

4.1. Motivation and Research Approach

The discussion from the previous sections highlights the fact that, effort prediction is a complex activity involving many difficulties such as dealing with imprecision, accounting for uncertainty, and involvement of experts to produce a reasonably accurate effort estimate. The impact of producing accurate effort estimates is also clearly visible in

terms of project proffering, acceptance, scheduling, execution and profit/loss for business entities. The presence of numerous algorithmic and non-algorithmic models for effort prediction have certainly added value and improvements in this area, but the scope for further improvements and development still prevails. This is justified by the ongoing research in this domain and is even more highlighted by the attempts from researchers to incorporate machine learning and other related techniques to produce improvements in the effort prediction accuracy.

The hierarchical break down of the domain from Cost Estimation to Effort Estimation, and subsequently from Effort Estimation to Size Estimation brings forth the diversity of approaches and methodologies designed and developed over the years. A vast number of techniques for effort estimation and an equally vast number of metrics and techniques for size estimation can be found in the literature. As mentioned in Section 1.2, the results of the first survey show the presence of machine learning techniques in various approaches as early as in 1992. Zonglian and Xihui [101] presented the idea of fuzzifying the COCOMO model for effort estimation in 1992 which is famously called '*f-COCOMO*'.

Dividing the effort prediction techniques into two categories namely '*use-case based techniques*' and '*non- use-case based techniques*', one can draw a clear distinction portraying the wide gap in the incorporation of machine learning techniques between them. Apart from the work of Zonglian and Xihui [101], other prominent works in the '*non-use-case based techniques*' category which utilized machine learning are

[33][35][54][57][65][67][78][81][88]. Liang and Noore [54] presented a proposal for a multi-stage software estimation model using Fuzzy Logic. Saliu [81] presented an adaptive fuzzy logic based framework for effort prediction. The main aim of Saliu's work was to deal with imprecision. Muzaffar [67] proposed a novel framework for effort prediction aimed at dealing with imprecision and uncertainty using the concept of type-2 fuzzy logic. Comparing the aforementioned approaches with the '*use-case based techniques*' category, there are only two works which have tried to incorporate machine learning [13] [22]. Braz and Vergilio [13] proposed a new size metric called USP and later incorporated fuzzy logic with it to create FUSP. The USP and FUSP focus mainly on including the details of the use cases to produce the size estimate. Fan et al [22] used a combination of fuzzy logic and bayesian belief networks to evolve a new framework for effort prediction based on use cases.

As such, it is clear that there have been very few attempts to incorporate techniques such as fuzzy logic, neural networks, Bayesian networks and genetic algorithms in the '*use-case based techniques*' category. Surprisingly, the much acclaimed use-case based technique proposed by Karner [42], the Use Case Points method has not been subjected to the collaboration of machine learning techniques. This leads to the idea of modifying the existing UCP method (*f-UCP*) by incorporating fuzzy logic on similar lines as the '*f-COCOMO*' model [101].

Following up the discussion from Section 3.3, it is obvious that two major shortcomings that need to be dealt with are the non-transparency of effort prediction processes and the inability to deal with imprecision and uncertainty. Fuzzy Logic, along with its power of approximate reasoning can help build transparent effort prediction models capable of incorporating expert opinions and dealing with imprecision and uncertainties. Additionally, Kamal et al [40] states in the future work that the impact of the technical complexity factors (TCF) and experience factors (EF) on the prediction accuracy needs to be evaluated.

With the above stated facts exhibiting a clear insight about the motivation of the work, the research approach follows a well-structured and goal based methodology. Initially, a proposal for integrating fuzzy logic with the existing UCP method (viz a viz '*f-UCP*') can be seen in the immediately following section. To study the impact of TCF and EF on the prediction process and to decide between including/excluding few factors, dimension reduction (Factor Analysis) on TCF and EF is performed. Then, the proposed adaptive fuzzy logic based framework is presented keeping in mind the post-implementation observations obtained from '*f-UCP*' and '*Factor Analysis*'.

Following the proposed framework is another alternative framework for effort prediction which is aimed at simplifying the effort prediction process by removing all additional factors other than the main factors pertaining to use-case information (Actors and Use Cases). This simplified framework is then extensively evaluated for a variety of

objectives including the impact of pairwise combinations strategy for defining rules in a fuzzy logic system, impact of design parameters and the choice of fuzzy logic system to be used (Mamdani vs. Sugeno). More details regarding these evaluations can be seen in Chapter 5.

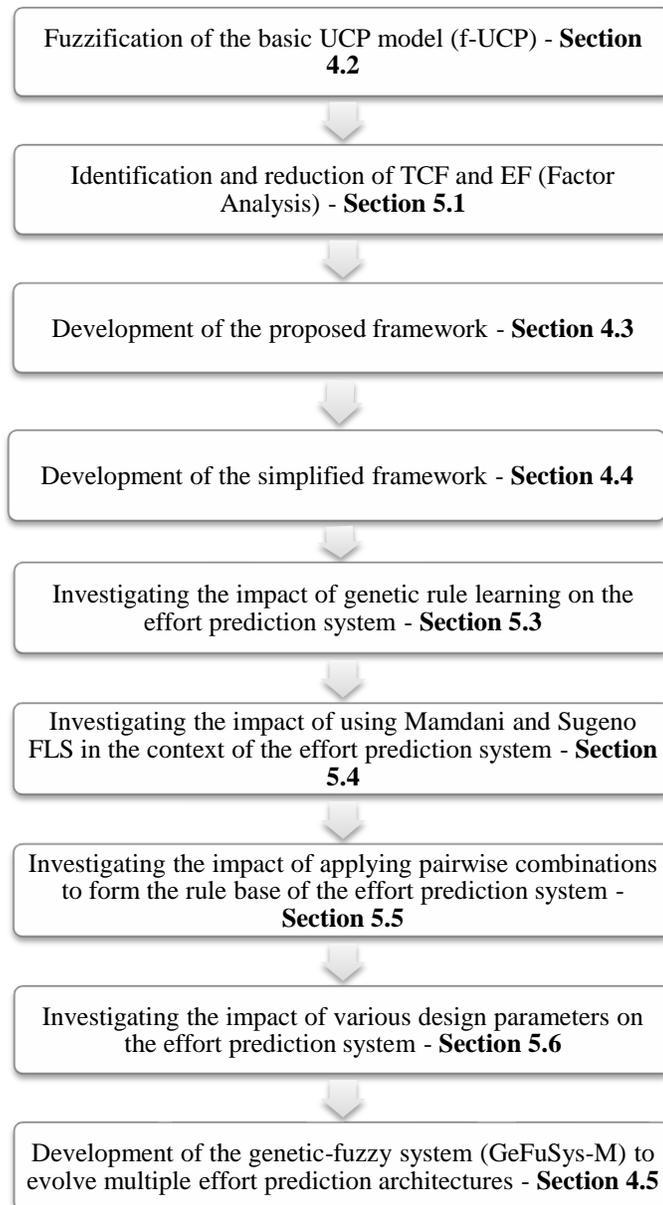


Figure 8: The Research Approach

Finally, the power of Genetic Fuzzy Systems is realized by developing a genetic learning model for the simplified effort prediction framework. Summing up, a specially designed chromosome structure for implementing a generic multi-layer genetic fuzzy system (*GeFuSys-M*) for effort prediction is developed and implemented. For sake of illustration, the research approach is presented as a flow chart in Figure 8.

4.2. '*f-UCP*': Fuzzy Use Case Points Method

The UCP method proposed by Karner [42] is the most popular method of effort prediction based on Use Cases. The aim of fuzzifying the UCP method is to replace the existing mathematical model with a fuzzy model. The added advantage is that, fuzzification of UCP factors will provide a gradual and continuous classification for experts to choose between 2 values for a particular factor. Another reason to actualize the *f-UCP* method is to try to improve the prediction of effort by building a transparent system which can aid the experts to incorporate their opinions. To design a fuzzy UCP model, we need to use fuzzy logic systems (FLS) as components coupled in an architecture representing a complete UCP method. The architecture for *f-UCP* is presented in Figure 9.

The *f-UCP* method is designed using Sugeno type of fuzzy inference system as opposed to the more commonly used Mamdani type fuzzy inference system. More details regarding the differences between the two types of inference systems can be seen in the next chapter. Designing a fuzzy logic system requires four steps. The first step in building

a fuzzy logic system hereafter referred to as FLS; is to define the fuzzy sets for all input/internal and output/external attributes. The second step is to formulate the rule base using the linguistic variables for each fuzzy set. The third step is training the FLS to refine the linguistic relationships in the rule base. The fourth step is to validate the performance of the FLS using test data. We will discuss the four steps in the following sequel.

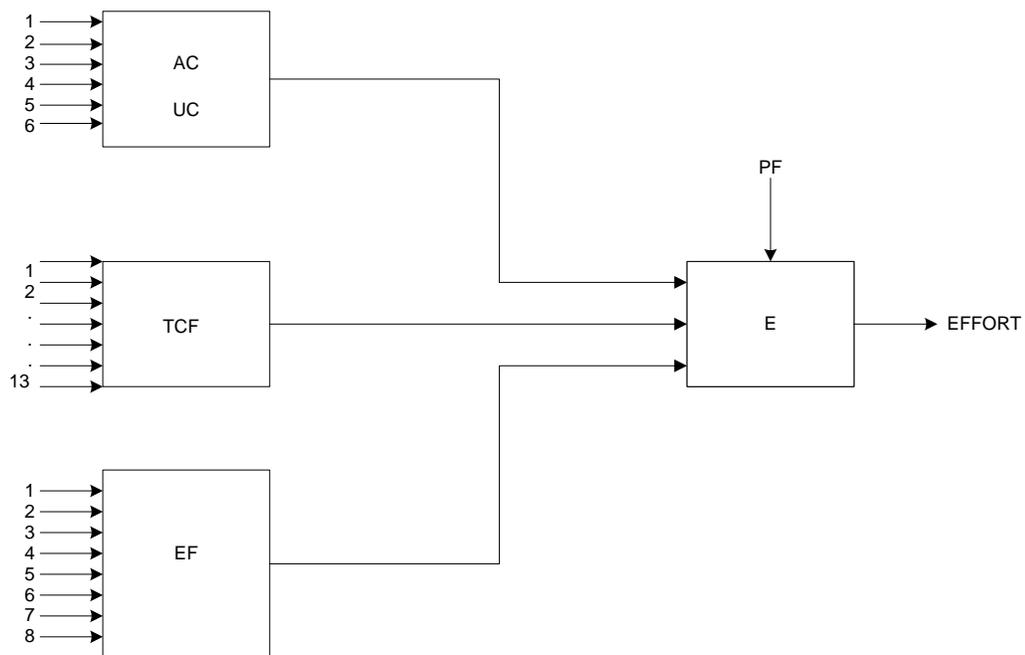


Figure 9: Architecture of the f-UCP method

4.2.1. Defining Antecedent and Consequent Fuzzy Sets

The internal and external attributes of the system under consideration are classified into fuzzy sets based on either expert opinion or by analysis of numerical data sets. From the

architectural design of the *f-UCP* method, it is visible that the number of inputs to the components in the first layer is large. The last component in the second layer has relatively lesser number of inputs. Nevertheless, we use 3 membership functions for each input attribute. The type of the membership functions used is Gaussian. The membership functions overlap initially and are shouldered at the interval boundaries of the antecedents.

4.2.2. Rule Base Formulation

There are quite a few approaches which are commonly used for formulating the rules of an FLS. One such method is to consider all the possible combinations of antecedent fuzzy sets to create a complete rule base. Even though this approach has advantages, the disadvantage is that, it creates a large rule base when the number of inputs is large or the number of membership functions used is large. In *f-UCP*, each component is an individual FLS and since there are a large number of inputs to each component, there would be an explosion of rules in the rule base.

To resolve the problem of accommodating a large rule base, a clustering technique called ‘Subtractive Clustering’ is used. The subtractive clustering method extracts rules that model the data behavior. The method (see, algorithm 4-1) assumes each data point to be a potential cluster center and calculates a measure of the likelihood that each data point would define the cluster center, based on the density of surrounding data points.

A point to be noted is that, the subtractive clustering method is used for formulating the rules of the first 3 components in the first layer. The last component (Effort) in the second layer does not use subtractive clustering since it has only 4 inputs. As a result, all possible combinations are used to define the rule base of the last component, which imply the presence of 81 rules ($3 \times 3 \times 3 \times 3$).

Algorithm 4-1: Subtractive Clustering Method

1. Selects the data point with the highest potential to be the first cluster center.
2. Removes all data points in the vicinity of the first cluster center (as determined by radii), in order to determine the next data cluster and its center location.
3. Iterate on this process until all of the data is within radii of a cluster center

4.2.3. f-UCP Training

The most important aspect of realizing an efficient *f-UCP* method is related to the training of the system. Training corresponds to the refinement of linguistic relationships in the rule base by adapting the parameters associated with the membership functions. In the context of *f-UCP*, the adaptive neuro-fuzzy inference system (ANFIS) is used for training the system.

Referring to the discussion in Section 2.3.3 wherein it is stated that neural networks and genetic algorithms provide learning capabilities to Fuzzy Logic Systems, ANFIS is an example of one such system which uses the adaptive learning techniques of neural networks in the context of fuzzy systems to learn information about a data set. In essence, ANFIS is concerned with tuning/training the parameters of membership functions belonging to a particular FLS. ANFIS allows two methods of training the FLS; a back propagation method and a combination of back propagation method with least squares method, the latter of which is called the Hybrid method of ANFIS training.

In *f-UCP*, all the 4 components are subjected to training using ANFIS. The training data set is prepared before the start of the training procedure. The training data set comprises of 70% of the available data.

4.2.4. f-UCP Validation

Finally, the complete system comprising of all the 4 FLS is activated. Testing data can be used to validate the performance of the proposed *f-UCP* method. The testing data set is 30% of the available data. The mean absolute relative error (described in the next section) is used for validating the performance of *f-UCP*.

4.3. The Proposed Adaptive Fuzzy Logic based Framework for Effort Prediction

The proposed framework aims at resolving all the issues and observations collectively obtained from the critical literature review, the *f-UCP* implementation and factor analysis. The proposed framework is depicted in Figure 10. The framework consists of two layers and 5 components, wherein each component is an individual FLS. The architecture presented in Figure 10 shows 4 components in the first layer and 1 component in the second layer. The outputs of the first layer are propagated as inputs to the last component in the second layer which produces the actual output i.e. the predicted effort. In this section, the details pertaining to the use of the proposed framework to initialize, formulate, train and validate the effort prediction systems are explicitly discussed. For sake of brevity, the discussion in the following sub-sections is kept general and is applicable to all the 5 components in the framework. For illustration purposes, few sub-sections have examples related to a specific component.

4.3.1. Initializing the System

Each component in the effort prediction system has certain input attributes and a single output attribute. Initializing the system corresponds to initializing the membership functions for the antecedent and consequent fuzzy sets for each individual component (FLS). In this thesis, we have used type-1 singleton FLS for all the components and the definitions of the antecedent and consequent membership functions have been obtained

using numerical analysis of data sets. Initializing the membership functions requires deciding on 3 major aspects;

1. Type of membership function to be used,
2. Number of membership functions to be used corresponding to the division of the variable interval into the number of regions/fuzzy sets,
3. Selecting the parameters for the membership functions.

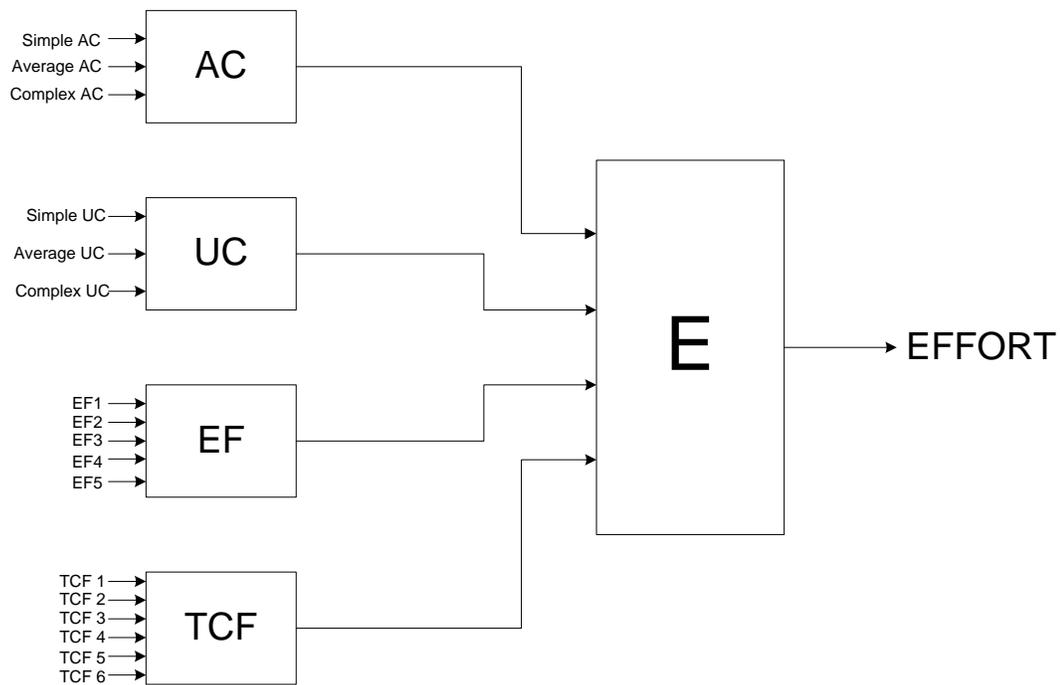


Figure 10: The Proposed Framework

As such, type-1 Gaussian membership functions have been used for both the antecedents and consequents. All the input attributes and output attributes of the five components

have been divided into 3 fuzzy sets, corresponding to 3 membership functions for each antecedent and consequent. A point to be noted is that, the terms membership function and fuzzy sets are used interchangeably and hence, antecedent fuzzy sets or antecedent membership functions imply the same.

After the variable (antecedent inputs) interval is divided into 3 regions, i.e. 3 antecedent fuzzy sets, we need to make sure that the 3 adjacent antecedent fuzzy sets overlap initially. This is done by making the tails of fuzzy sets lie at the mean of the adjacent fuzzy sets. The initial overlap helps in exploiting the power of fuzzy logic to handle data that lies in between the fuzzy sets intervals [81]. Also, the fuzzy sets that lie at the interval boundaries are shouldered.

When it comes to initializing the parameters for the membership functions, we follow the approach prescribed by Mendel [60]. This approach is suitable for designing type-1 singleton FLS which use back propagation methods for training the FLS. For the antecedents, there are two parameters which need to be defined while using a Gaussian membership function; mean (M) and standard deviation (σ) and for the consequents, one parameter needs to be defined; center of consequent membership function (c).

To define the antecedent parameters based on the numerical data set, the first step is to calculate the *mean* (M) and *standard deviation* (σ) values for each input attribute. Then

depending on the number of fuzzy sets for each input attribute, define the means of the membership functions ‘M’ as follows;

$$M_{mf1} = M - \alpha.\sigma$$

$$M_{mf2} = M$$

$$M_{mf3} = M + \alpha.\sigma$$

Where, ‘ α ’ is a constant that should be defined properly (using experience) so as to cover the complete interval range of a particular input attribute. M_{mfi} refers to the mean of the i^{th} membership function. The standard deviation values for all the 3 membership functions are kept to the same value of ‘ σ ’.

To define the consequent parameters based on the numerical data set, i.e. the centers of consequent fuzzy sets, the lowest and highest values for a particular output attribute are extracted from the numerical data. Then the range is calculated as the difference between the highest and lowest value. Dividing the range by the number of membership functions minus 1, gives the increment factor. To get the initial values for the center of consequents (c), start with the lowest value, keep adding the increment factor until all the consequent fuzzy sets have the center (c) values. While defining these values, one has to be careful to cover the domain interval of the output attributes.

4.3.2. Formulating the Rule Base

An integral part of the FLS is the rule base. The definitions of the *system parameters* and the *rule base* aid in realizing the process of fuzzy inference. In this context, we have used the approach described in [67][81], but with modifications. Both the approaches [67][81] use all possible combinations of the antecedent fuzzy sets to define the rule base. Moreover, the number of consequent fuzzy sets is equal to the number of the rules, with each consequent fuzzy set having a distinct center of consequents (*c*) value. This is similar to the One-pass method [60] of FLS design where the number of consequent fuzzy sets is equal to the number of rules in an FLS.

In our approach (back propagation method), we use all possible combinations of the antecedent fuzzy sets to define the rule base, but we have a fixed number of consequent fuzzy sets in an FLS corresponding to a fixed center of consequents (*c*) values. Since, each rule in an FLS should have a certain ‘*c*’ value, a random value from among the fixed ‘*c*’ values is chosen.

Typical rules for component 1 (Actors) are of the form;

- If **simpleAC** is *low* and **averageAC** is *low* and **complexAC** is *low*, then **ACTORS** is *low*
- If **simpleAC** is *low* and **averageAC** is *medium* and **complexAC** is *low*, then **ACTORS** is *low*
- If **simpleAC** is *medium* and **averageAC** is *low* and **complexAC** is *high*, then **ACTORS** is *medium*

- If **simpleAC** is *medium* and **averageAC** is *high* and **complexAC** is *high*, then **ACTORS** is *high*
- If **simpleAC** is *medium* and **averageAC** is *low* and **complexAC** is *high*, then **ACTORS** is *medium*

A point worth mentioning is that, *low*, *medium* and *high* correspond to the antecedent membership functions M_{mf1} , M_{mf2} and M_{mf3} respectively. Since, we consider all possible combinations of antecedent fuzzy sets to form the rule base; the total number of rules in an FLS is given by the product of the number of fuzzy sets for each input attribute in an FLS. In terms of the proposed framework, for component 1 (Actors), there are 3 input attributes namely; simpleAC, averageAC and complexAC. Each input attribute has 3 fuzzy sets which means that component 1 (Actors FLS) has 27 ($3 \times 3 \times 3$) rules. This method of formulating the rules applies to all the individual components in the overall effort prediction system.

In some cases, for an FLS, the number of input attributes is large or the number of fuzzy sets for input attributes is large. This leads to an explosion in the number of rules in the rule base which brings forth many difficulties in designing and implementing the FLS. This problem is called as the '*curse of dimensionality*' and is commonly faced by researchers mainly due to the former issue of large number of input attributes for and FLS. Section 5.5 presents an approach 'pairwise combinations' to check whether it can successfully resolve the issue of curse of dimensionality.

4.3.3. Training the System

In the context of the framework, training the system refers to the training of the rules in an FLS so as to improve the accuracy of predicting the output. After defining the fuzzy sets and formulating the fuzzy rule base, the third step is to train the FLS. Training the FLS is required to refine the linguistic relationships in the rule base. In this thesis, training is realized using back propagation algorithm, wherein the training proceeds by propagating the inputs through the FLS and modifying the parameters of various membership functions based on computed error and steepest descent approach, see Algorithm 4-2.

Before starting with the training procedure, training data sets need to be prepared from the available data sets. The training data set comprises of 70% of the available data. Additionally, because of the dearth of industrial data sets, artificial data sets were generated. More details pertaining to the generation of artificial data sets can be found in Chapter 6.

Algorithm 4-2: Training algorithm for tuning a singleton type-1 FLS

Given N input-output training samples $(x^{(i)} : y^{(i)})$, $i = 1 \dots N$. The objective is to minimize the error function for 'k' training epochs. The error function is computed as:

$$e^{(i)} = \frac{1}{2} (f(x^{(i)}) - y^{(i)})^2 \quad i = 1 \dots N$$

Steps

1. Initialize all the parameters.
2. Set the counter, ep, of the training epoch to zero i.e. ep=0.
3. Set the counter, i, of the training data to one. i.e., i=1.
4. Apply the means of inputs with their corresponding standard deviation to the singleton type-1 FLS and compute the output $f(x^{(i)})$.
5. Compute the output error (relative) as: $e = \frac{f(x^{(i)}) - y^{(i)}}{y^{(i)}}$
6. Tune the means and standard deviations of the antecedent membership functions and the centers of consequents using steepest descent algorithm for the error function.

The training algorithm is essentially the same as initially proposed by Mendel [60] and used by Muzaffar [67] and Rahman [72].

4.3.4. Framework Validation

Once the prediction system has been trained, the last step is to validate the performance. This is done by testing the system on testing data sets. The testing data sets are prepared from the available data sets and comprise of 70% of the available data.

For testing purposes, the trained system consisting of the modified parameters (antecedent means, antecedent standard deviations, center of consequents) is used and the testing data is applied to get the predicted output. Both the training and testing are carried out in terms of the overall prediction system, i.e. for all the 5 components. The output of the last component (*predicted effort*) and the *actual effort* values in the testing data set are used to calculate the error which gives a measure of the system's prediction accuracy and the overall validity of the proposed framework. The *mean absolute relative error* (MARE) has been used for obtaining the prediction accuracy. The *mean absolute relative error* is defined in Section 6.1. Thus, by using the testing data and the error measures, the validity of the framework can be established.

4.4. The Simplified Adaptive Fuzzy Logic based Framework for Effort Prediction

The proposed framework in Section 4.3 consists of a multi-layered architecture, the reason being the presence of a variety of input attributes pertaining to the technical complexity factors (TCF) and experience factors (EF). As a result, the prediction system was divided into multiple components and layers. Following from the results of the literature review, especially the result of the work by Ochodek et al [70], which states that the difference in the prediction accuracy is insignificant whether or not the TCF and EF are considered, we thought of designing a simple framework for effort prediction. Moreover, the aim of Ochodek et al [70] also, was to simplify the process of effort prediction based on *use case points*. They use *multiple regression analysis* to prove the result that TCF and EF affect the final prediction of effort minimally.

The proposed simplified framework differs from the previous framework in terms of the number of components and the number of attributes. The simplified framework has a single component consisting of 6 input attributes which include 3 inputs pertaining to the Actors and 3 inputs pertaining to the Use Cases. The output attribute is the Effort (predicted effort). The simplified framework is depicted in Figure 11. In what follows, are the details related to initializing, training and activating the simplified effort prediction framework.

4.4.1. Initializing the System

With regards to initializing the system, the sequel follows from the previous discussion in Section 4.3.1. Type-1 singleton FLS is used and Gaussian membership functions are chosen for the input attributes. Each input attribute has 3 membership functions. The parameters related to the definition of membership functions (means of antecedents, standard deviations of antecedents, centers of consequents) are initialized in a similar manner as described in Section 4.3.1.

4.4.2. Formulating the Rule Base

The prediction system consists of a single component having 6 input attributes and each input attribute has 3 membership functions. Since, we follow the approach used in the previous framework; we have 729 rules in the rule base corresponding to all the possible combinations of the antecedent membership functions. Examples of rules in the rule base are as follows;

- If **simpleAC** is *low* and **averageAC** is *low* and **complexAC** is *low* and **simpleUC** is *low* and **averageUC** is *low* and **complexUC** is *low* then **EFFORT** is *low*
- If **simpleAC** is *low* and **averageAC** is *medium* and **complexAC** is *high* and **simpleUC** is *low* and **averageUC** is *medium* and **complexUC** is *medium* then **EFFORT** is *medium*
- If **simpleAC** is *medium* and **averageAC** is *low* and **complexAC** is *high* and **simpleUC** is *high* and **averageUC** is *low* and **complexUC** is *low* then **EFFORT** is *medium*

- If **simpleAC** is *high* and **averageAC** is *high* and **complexAC** is *medium* and **simpleUC** is *low* and **averageUC** is *medium* and **complexUC** is *high* then **EFFORT** is *high*
- If **simpleAC** is *high* and **averageAC** is *low* and **complexAC** is *medium* and **simpleUC** is *medium* and **averageUC** is *low* and **complexUC** is *high* then **EFFORT** is *medium*

A point to note is that this is quite a large rule base in terms of implementing an FLS. More commonly, the number of input attributes is lesser in practice as can be seen in Muzaffar's [67] and Saliu's [81] work. Typically, in industrial applications of FLS, expert opinions are used to define the rule base of a fuzzy logic system which helps in reducing the number of rules considerably. The experts use their experience in deciding upon including the important rules and discarding the unnecessary ones. This aids in realizing simple, yet efficient systems.

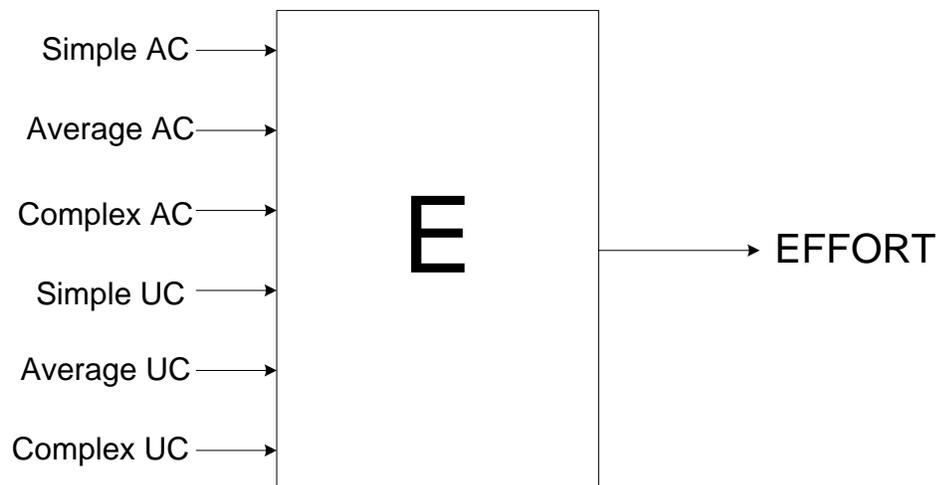


Figure 11: The Proposed Simplified Framework for Effort Prediction

4.4.3. Training the System

The prediction system under consideration is trained for refining the linguistic relationships in the rule base on similar lines as the previous framework. Back propagation (Steepest Descent Approach) algorithm is used to train the system. The training algorithm (Algorithm 4-2) can be seen in Section 4.3.3. The training data set is extracted from the available data set. The training data set contains 70% of the available data set.

4.4.4. Framework Validation

The simplified framework is validated by testing the prediction system with testing data. The testing data is obtained from the available data set and is chosen to be 30% of the available data. MARE is used as the error measure and helps in obtaining the prediction accuracy.

4.5. The Proposed Genetic Fuzzy System (GeFuSys-M) for evolving multi-layered architectures for Use-Case based Effort Prediction Systems

A fuzzy logic system augmented by a genetic learning process makes it more efficient than a plain FLS which just performs the process of fuzzy inference by utilizing a defined rule base to produce outputs given a set of inputs. From Section 2.3.3, we note that

genetic learning processes cover different levels of complexity according to the structural changes produced by the algorithm, from the simplest case of parameter optimization to the highest level of complexity of learning the rule set of a rule based system. Fuzzy Systems with genetic learning capabilities to learn the rule sets are also called Genetic Fuzzy Rule based Systems (GFRBS). A complete design and implementation of a GFRBS in the context of the simplified framework for effort prediction can be seen in Section 5.3.

Based on the observations obtained from implementing the genetic learning process for the simplified effort prediction framework, the idea of designing a special chromosome structure for building a multi-layered genetic fuzzy system (GeFuSys-M) was conceived. The main theme of designing such a system is to exploit the power of genetic learning to generate an exhaustive number of effort prediction systems and then return the most optimal effort prediction system. The generated effort prediction systems differ from each other in terms of number of inputs used, number of components within a system, division of inputs into each component, number of rules in each component, interconnection between the components, and rule sets for each component.

GeFuSys-M is a complex genetic fuzzy system with an even more complex chromosome structure which caters to a large number of requirements of a prediction system designer. Following are the requirements that are expected to be fulfilled by GeFuSys-M in the context of effort prediction systems;

1. GeFuSys-M selects the number of inputs to be used in a particular effort prediction system. Suppose a designer specifies a set of inputs to be used for designing an effort prediction system, GeFuSys-M selects either a subset of the total inputs or the complete set as per the genetic learning process.
2. GeFuSys-M defines the number of components (individual FLS) in a particular effort prediction system. Depending on the number of inputs selected for that particular system, GeFuSys-M defines the number of components based on the genetic learning process.
3. GeFuSys-M divides the selected inputs into the defined components. Assuming that there are 'X' inputs to be divided across 'Y' components, GeFuSys-M offers a genetic learning based solution for the task.
4. GeFuSys-M is responsible for defining the number of rules for each component. Depending on the number of inputs in a component, GeFuSys-M defines the number of rules to be accommodated in the particular component. The rules are initialized randomly.
5. Most importantly, GeFuSys-M gives various architectures for effort prediction systems. This is achieved by depending on the learning process to give different interconnections between the components.
6. Lastly, GeFuSys-M gives the best set of rules for each component within the system and consequently the best overall effort prediction system.

A point worth mentioning is that, GeFuSys-M is a generic system, in other words a framework which is not fixed to be used in the context of effort prediction alone, but rather to any application domain utilizing fuzzy systems. The idea is to play with a

plethora of design possibilities for building a fuzzy logic system. Since, genetic algorithms and genetic fuzzy systems have been extensively discussed in Chapter 2 of this thesis; we can proceed directly to the chromosome structure of GeFuSys-M.

4.5.1. Chromosome Design for GeFuSys-M

The most important step in designing a genetic fuzzy system is designing the chromosome itself. A large number of GFS differ in the way their chromosome structures are designed. As we know that, genetic algorithms start with an initial population of candidate solutions encoded as chromosomes, it is imperative that an efficient chromosome structure be designed which can lead to optimal solutions. The chromosome structure is depicted in Figure 12. Table 13 presents a detailed description of the chromosome structure.

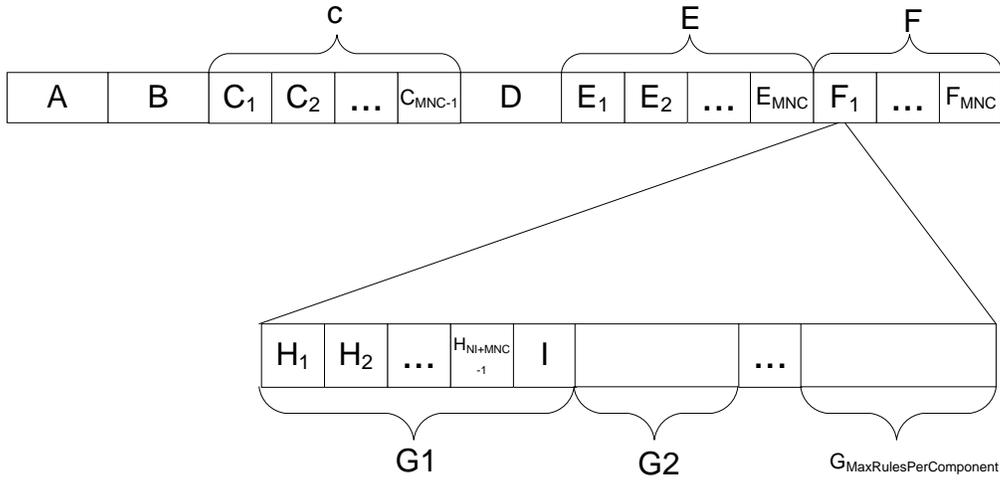


Figure 12: Chromosome Structure for GeFuSys-M

Table 13: Chromosome Description

Chromosome Part	Size in Bits	Description
A	numBitsPerm	This part selects an index from a list of $2^{\text{numBitsPerm}}$ index values. A pre-defined array whose size is $2^{\text{numBitsPerm}}$ consists of permutation wise arrangement of the set of inputs for a particular system. ‘numBitsPerm’ is the number of bits used for making the permutation array and it defines the number of permutations the array should hold. If numBitsPerm = 16, and the number of inputs = 27, then the permutation array holds 65536 permutation wise arrangements of 27 inputs.
B	$\text{numBitsInput} = \log_2(\text{numInputs})$	This part gives the number of inputs selected from the total set of inputs, i.e. from ‘numInputs’. ‘numBitsInput’ corresponds to the size in bits in the chromosome structure reserved for the selected inputs.
C = C ₁ , C ₂ ... C _{MNC-1}	$(\text{MNC} - 1) * \text{numBitsInput}$	This part indicates how many inputs go to each of the components. ‘MNC’ stands for maximum number of components. It helps us to put an upper limit on the number of components we wish to accommodate in our

		system.
D	$(MNC-2)(MNC-1)/2$	This part gives the different architectures possible in the system by defining interconnections between the components by utilizing an adjacency matrix (see Section 4.5.1.1).
$E = E_1, E_2 \dots$ E_{MNC}	MNC * numBitsMRPC	This part gives the number of rules for each component. The number of rules in each component does not exceed the limit defined by MRPC. MRPC stands for ‘Maximum number of Rules per Component’. As such, ‘numBitsMRPC’ allocates the size for accommodating the rules in the chromosome structure.
$H = H_1, H_2 \dots$ $H_{numInputs+MNC-1}$	$(numInputs + MNC - 1) *$ numBitsInputMFs	This part gives the values of the input membership functions (low, medium, high) for data inputs. Additionally, in case of a multi layered architecture being produced by the system, this part gives the values of the input membership functions for those inputs which are outputs of the previous components and are being fed as inputs to the next component. ‘numBitsInputMFs’ is the size in bits required to accommodate the values of the input membership functions.
I	numBitsOutputMFs	This part gives the values of the output membership functions (low, medium, high) for data outputs. ‘numBitsOutputMFs’ is the size in bits required to accommodate the values of the output membership functions.
$G = G_1, G_2 \dots$ G_{MRPC}	MRPC * (H + I)	(G = H + I) is one complete rule. A collection of such rules i.e. $G_1, G_2 \dots G_{MRPC}$ gives one complete Rule Set. Hence, this part of the chromosome gives the ‘Rule Set’ for one component.
$F = F_1, F_2 \dots$ F_{MNC}	MNC * G	This part of the chromosome gives the Rule Sets of all the components in the complete system.

4.5.1.1. Adjacency Matrix for interconnections between components

One of the most important and challenging aspect of designing the chromosome is to allow the learning process of GeFuSys-M to generate a variety of architectures for the effort prediction system. To embed this functionality in GeFuSys-M, we used the concept of adjacency matrix. An upper triangular adjacency matrix for defining the interconnections between the components is used. The structure of the adjacency matrix

is for 5 components and 4 components is depicted in Figure 13 and Figure 14 respectively.

	2	3	4
1	0	0	1
2		1	0
3			0

Figure 13: Adjacency matrix in case of 5 components

	2	3
1	0	1
2		1

Figure 14: Adjacency matrix in case of 4 components

Before interpreting the contents of the adjacency matrix, we first determine its size in terms of the chromosome structure. The assumption is that, the last component is fixed at the end of the architecture and hence the output of the immediately preceding component will always be the input to the last component. So, we do not need to know the interconnections for the last 2 components, as a consequence why we have the number of rows in the adjacency matrix as '1 to MNC-2'. And since, if any of the previous

components are not connected to each other, it implies that they are connected to the last component. Hence, the number of columns in the adjacency matrix starts from '2 to MNC-1'. The size of the adjacency matrix is calculated based on the number of entries in the matrix.

$$\text{Number of entries} = 1 + 2 + 3 + \dots + (MNC-2)$$

Using the formula for sum of 'n' numbers, i.e. $n(n+1)/2$;

$$\text{Size of matrix} = (MNC-2)(MNC-1)/2$$

The method of interpreting the contents of the adjacency matrix is as follows;

Let the entry in the adjacency matrix be a vector of row entry and column entry corresponding to (i, j).

If $(i, j) = 1$, \Rightarrow the i^{th} component is connected to component 'j', which also implies that the output of the i^{th} component is input to component 'j'.

If $(i, j) = 0$, \Rightarrow the i^{th} component is not connected to component 'j'.

A point to note that, if any component is not connected to any other component in the adjacency matrix, it means that the component is connected to the last component in the architecture, which is always fixed.

4.5.2. The Genetic Learning Process

The genetic learning process begins with a random population, i.e. a set of randomly generated chromosomes. Each individual in a population corresponds to one complete effort prediction system. Subsequently, the fitness value for each individual in the population is obtained. The fitness function used is the Mamdani type-1 FLS which takes the rule set and the inputs used from the chromosome and evaluates the prediction accuracy of the system. The fitness value corresponds to the Mean Absolute Relative Error (MARE) which is calculated in the fitness function itself. Once a generation is completed, the genetic learning process applies operators such as Selection, Crossover and Mutation to generate the population for the next generation.

Selection: *Stochastic uniform selection* is used in the context of GeFuSys-M. Stochastic uniform selection lays out a line in which each parent corresponds to a section of the line of length proportional to its scaled value. The selection algorithm moves along the line in steps of equal size. At each step, the algorithm allocates a parent from the section it lands on.

Crossover: *Scattered crossover* is used in the context of GeFuSys-M. It creates a random binary vector of 1's and 0's and selects the genes from the first parent where the vector is a 1, and the genes from the second parent where the vector is a 0, and combines the genes to form the child. The crossover probability is kept at 0.8. For example, if p1 and p2 are the parents given as follows;

p1 = [a z j k n l y t]

p2 = [1 2 3 4 5 6 7 8]

Binary vector = [1 1 0 0 1 0 0 0], the function returns the following child:

Child = [a z 3 4 n 6 7 8]

Mutation: *Gaussian mutation* is used, which adds a random number taken from a Gaussian distribution with mean 0 to each entry of the parent vector. The standard deviation of this distribution is determined by the parameters *Scale* and *Shrink*. The *Scale* parameter determines the standard deviation at the first generation. The *Shrink* parameter controls how the standard deviation shrinks as generations go by.

The genetic learning process continues until the end of the specified generations. The best fitness value corresponding to the minimum MARE value is reported from among the complete learning phase. The prediction system which gives the best fitness value is returned as the most optimal effort prediction system. An important point is that, GeFuSys-M during its learning phase exhaustively searches for various possibilities of effort prediction system pertaining to different architectures, different inputs, different components and different rule bases. Overall, GeFuSys-M is an attempt to create a powerful tool which aims at realizing an efficient GFS which can cater to the needs of the FLS designers irrespective of the application domain. The application of GeFuSys-M in the domain of Effort Prediction has yielded encouraging and promising results which can be seen in Chapter 6.

CHAPTER 5

EXAMINING THE IMPACT OF VARIOUS ALTERNATIVES ON PERFORMANCE

In this chapter, we present the important aspects of the proposed frameworks related to the training algorithm used, the choice of the fuzzy inference system, and the definition of design parameters for training the proposed frameworks. All the aforementioned aspects play a vital role in designing FLS based effort prediction systems. We proceed by first presenting a brief discussion on factor analysis which played a key role in the development of the proposed effort prediction framework, see Section 4.3. A genetic learning model for learning the rule sets of an FLS has been presented in the context of the simplified effort prediction framework.

5.1. Factor Analysis

The term Factor Analysis refers to a widely used statistical technique for dimension reduction and data classification. The main applications are to reduce the number of variables, and to classify the variables by detecting structure in the relationships between variables. Factor analytic techniques are broadly classified into two categories; Exploratory Factor Analysis and Confirmatory Factor Analysis. Exploratory factor analysis aims at finding new structural relationships between factors, whereas Confirmatory Factor Analysis aims at determining whether or not the factors conform to the pre-defined structural relationship. One of the most commonly used forms of exploratory factor analysis is the ‘Principal Components Analysis’ which has been used in the context of this thesis.

Principal Components Analysis (PCA) is a descriptive statistical technique that employs the concept of orthogonal transformations to convert a set of possibly correlated variables into a set of linearly uncorrelated variables. The resulting sets of uncorrelated variables are called ‘principal components’. Usually, the number of principal components is less than the number of original variables. PCA extracts the components in such a way that the first principal component accounts for the largest possible variance. This is continued further, wherein the second principal component accounts for the maximum variance amongst the remaining components and at the same time is orthogonal to the first component.

We applied PCA to reduce the number of ‘Experience Factors’ and ‘Technical Complexity Factors’ pertaining to the design of the adaptive fuzzy logic based framework for effort prediction. As a consequence of developing the *f-UCP* method, the observation that large number of inputs to any FLS resulted in an explosion of rules (curse of dimensionality), led us to reason about ways of reducing the number of rules. Subsequently, the idea of dimension reduction was conceived which is realized by the use of Principal Components Analysis to reduce the number of inputs to FLS, thereby reducing the number of rules.

5.1.1. Principal Components Analysis on Experience Factors (EF)

PCA was conducted to reduce the number of variables corresponding to the 8 Experience Factors. Kaiser-Meyer-Olkin (KMO) Measure of Sampling Adequacy and Bartlett’s test of Sphericity was conducted and a KMO value of 0.652 was found. A KMO value close to one indicates that correlations are tightly coupled and as a result the PCA should produce distinct and reliable factors/components. Additionally, a Chi Square value of 307.174 with 10 degrees of freedom and a significance value of 0.000 was reported which establishes that there are non-zero correlations between the 8 EF variables and hence factors exist.

Since, rotation optimizes the component structure; we used the Direct Oblimin Rotation method along with the principal components analysis. The ‘maximum iterations for

convergence' value was set to 10. The rotations converged in 3 iterations and yielded 5 variables classified under 2 components. EF-1, EF-2, EF-3 fall under the first component and EF-7 and EF-8 fall under the second component. The comprehensive set of results for PCA on Experience Factors can be seen in Chapter 6.

5.1.2. Principal Components Analysis on Technical Complexity Factors (TCF)

PCA was conducted to reduce the number of variables corresponding to the 13 Technical Complexity Factors. Kaiser-Meyer-Olkin (KMO) Measure of Sampling Adequacy and Bartlett's test of Sphericity was conducted. A KMO value of 0.459 was reported. Even though the KMO value is not close to one, there is finite possibility that PCA would produce distinct factors. A Chi Square value of 368.514 with 15 degrees of freedom and a significance value of 0.000 were reported.

The Direct Oblimin Rotation method was used and the 'maximum iterations for convergence' value was set to 10. The rotations converged in 4 iterations and yielded 6 variables classified under 3 components. TCF-2 and TCF-3 fall under the first component; TCF-5 and TCF-9 fall under the second component and the third component includes TCF-6 and TCF-12. The comprehensive set of results for PCA on Technical Complexity Factors can be seen in Chapter 6.

5.2. Training Algorithm

In this thesis, back propagation (Steepest Descent Approach) algorithm has been used in the context of training the FLS based effort prediction system. Both the proposed frameworks utilize the same training algorithm as mentioned in Section 4.3.3 and Section 4.4.3. In the back propagation method, neither of the antecedent or consequent parameters is fixed ahead of time [60]. A steepest descent method is used to tune the antecedent and consequent parameters. A point worth mentioning is that, the following discussion related to the steepest descent approach of tuning the antecedent and consequent parameters via a back propagation algorithm is essentially the same as prescribed by Mendel [60] and used by Muzaffar [67].

Consider a type-1 singleton fuzzy logic system with Gaussian membership functions which uses max-product composition, product implication and height defuzzification.

Such an FLS is expressed by the following equation:

$$y(x^{(i)}) = f_s(x^{(i)}) = \frac{\sum_{l=1}^M y^{-l} \prod_{k=1}^p \exp \left[-\frac{(x_k^{(i)} - m_{F_k^l})^2}{(2\sigma_{F_k^l}^2)} \right]}{\sum_{l=1}^M \prod_{k=1}^p \exp \left[-\frac{(x_k^{(i)} - m_{F_k^l})^2}{(2\sigma_{F_k^l}^2)} \right]} \quad i = 1, \dots, N \quad (5-1)$$

Where, ‘M’ is number of rules, ‘p’ is number of antecedents and ‘N’ is number of data points.

Given an input-output training pair $(x^{(i)} : y^{(i)})$, the aim is to design an FLS such that the following error function is minimized:

$$e^{(i)} = \frac{1}{2} [f_s(x^{(i)}) - y^{(i)}]^2 \quad i = 1, \dots, N \quad (5-2)$$

From equation (5-1), it is clear that, f_s is completely characterized by the following 3 parameters;

- \bar{y}^{-l} (point at which rule l has the highest degree of membership),
- $m_{F_k^l}$ (mean of k^{th} antecedent in rule l),
- $\sigma_{F_k^l}$ (Standard deviation of k^{th} antecedent in rule l).

In order to minimize error function in (5-2), the steepest descent approach can be applied. The steepest descent approach helps in obtaining the following recursions to update all the design parameters of this FLS ($k = 1, \dots, p, l = 1, \dots, M$ and $i = 0, 1, \dots$):

$$\bar{y}^{-l}(i+1) = \bar{y}^{-l}(i) - \alpha_y [f_s(x^{(i)}) - y^{(i)}] \phi_l(x^{(i)}) \quad (5-3)$$

$$m_{F_k^l}(i+1) = m_{F_k^l}(i) - \alpha_m [f_s(x^{(i)}) - y^{(i)}] [\bar{y}^{-l}(i) - f_s(x^{(i)})] \times \frac{[x_k^{(i)} - m_{F_k^l}(i)]}{\sigma_{F_k^l}^2(i)} \phi_l(x^{(i)}) \quad (5-4)$$

$$\sigma_{F_k^l}(i+1) = \sigma_{F_k^l}(i) - \alpha_\sigma [f_s(x^{(i)}) - y^{(i)}] [\bar{y}^{-l}(i) - f_s(x^{(i)})] \times \frac{[x_k^{(i)} - m_{F_k^l}(i)]^2}{\sigma_{F_k^l}^3(i)} \phi_l(x^{(i)}) \quad (5-5)$$

Mendel [60] states that, since \bar{y}^{-l} , $m_{F_k^l}$ and $\sigma_{F_k^l}$ are parameters associated with the membership functions for meaningful variables; it is possible to obtain good initial values for them. In the worst case, these parameters can be chosen randomly. The point is that, smartly initializing the parameters causes the back propagation algorithm to converge faster and randomly initializing the parameters causes the algorithm to converge slower.

The back propagation algorithm is presented as follows;

Algorithm 5-1: Back Propagation Algorithm for Type-1 FLS

1. Initialize the parameters of all the membership functions for all the rules, $m_{F_k^l}(0)$, $\bar{y}^l(0)$ and $\sigma_{F_k^l}(0)$.
2. Choose the learning parameters, α_m , α_y and α_σ . Usually they are chosen to be the same, say α .
3. Set some end criterion to achieve convergence.
4. **Repeat**
 - i. **for all** data points $(x^{(i)} : y^{(i)}) i = 1, \dots, N$
 - a. Propagate the next data point through the FLS and compute $f_s(x^{(i)})$
 - b. Compute error as: $e = y^{(i)} - f_s(x^{(i)})$
 - c. Update the parameters of the membership functions using (5-3), (5-4) and (5-5).
 - ii. end for (*end for each input-output pair*)
 - iii. Compute the mean absolute relative error (MARE) as in (5-6).
 - iv. Test the end criterion. If satisfied break.

Until (*end for each epoch*)

The mean absolute relative error is given as follows;

$$\text{MARE} = \frac{|y^{(i)} - f_s(x^{(i)})|}{y^{(i)}} \quad (5-6)$$

5.3. Genetic Learning of Rule Sets

Genetic Fuzzy Systems (GFS) have been extensively discussed in Section 2.3 and Section 4.5 of Chapter 2 and Chapter 4 respectively. As such, it is clear that fuzzy systems with genetic learning capabilities to learn the rule sets are called Genetic Fuzzy Rule based Systems (GFRBS). This section presents a design of one such GFRBS in the context of the proposed simplified framework for effort prediction, see Section 4.4. The actual theme behind the concept of developing the GFRBS is to design a genetic learning model for learning the rules in a rule set. In this context, each rule set corresponds to a single FLS, which implies that the genetic learning process generates an exhaustive number of FLS with different rule sets and finally returns the most optimal FLS for effort prediction.

5.3.1. Chromosome Structure

As mentioned earlier, chromosome is the most important part of the genetic process, since the genetic algorithms start with an initial population of solutions which are encoded as chromosomes. The chromosome structure for the GFRBS under consideration is depicted in Figure 15.

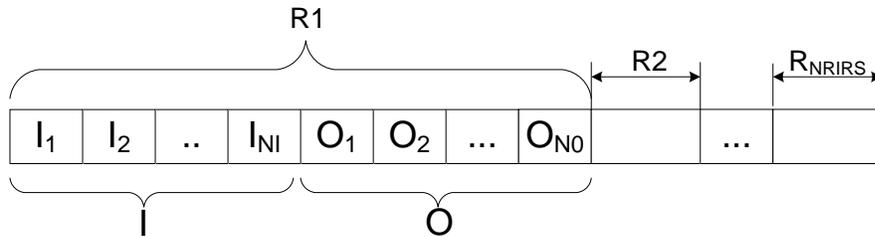


Figure 15: Chromosome Structure

Table 14: Chromosome Description

Chromosome Part	Size in Bits	Description
$I = I_1, I_2 \dots I_{NI}$	$NI \cdot \text{numBitsPerInput}^*$	This part gives the values of the input membership functions (low, medium, high) for data inputs. 'numBitsPerInput' is the size in bits required to accommodate the values of the input membership functions.
$O = O_1, O_2 \dots O_{NO}$	$NO \cdot \text{numBitsPerOutput}^*$	This part gives the values of the output membership functions (low, medium, high) for data outputs. 'numBitsPerOutput' is the size in bits required to accommodate the values of the output membership functions. Usually, we have 1 output, but the chromosome has space for accommodating multiple outputs.
$R = R_1, R_2 \dots R_{NRIRS}$	$NRIRS \cdot (I + O)$	$(R = I + O)$ is one rule. This part gives the complete rule set made of NRIRS rules. 'NRIRS' is an abbreviation for the number of rules in a rule set.

5.3.2. Other Genetic Learning Considerations

The genetic learning process begins with a population of candidate solutions, i.e. a set of randomly generated chromosomes. Each individual in a population corresponds to one

complete effort prediction system with one complete rule set. Subsequently, the fitness value for each individual in the population is obtained and the population for the next generation is produced. This activity proceeds until the maximum number of generations is reached. In the end, GA returns the FLS with the most optimal rule set. The various design considerations for implementing the genetic learning process are presented in Table 15.

Table 15: Design Considerations for the Genetic Learning Process

Design Consideration	Value
Fitness Function	Mamdani type-1 FLS
Measure of Fitness Value	Mean Absolute Relative Error (MARE)
Selection Function	Stochastic Uniform Selection
Crossover Function	Scattered Crossover
Crossover Probability	0.8
Mutation Function	Gaussian Mutation

5.4. Choice of Fuzzy Inference System

Fuzzy Inference Systems (FIS) can be seen as functions which perform universal approximation and conform to the laws of the Universal Approximation theorem [37]. FIS maps the input space to the output space of a model by approximating the model input-output data. There are basically two types of Fuzzy Inference Systems; Mamdani-type and Sugeno-type.

Mamdani-type FIS are more popular and widely used because of the relatively simple structure of the inference model and also because of the interpretability of the rule base [28]. In a Mamdani FIS, both the antecedent membership functions and the consequent membership functions are fuzzy sets which add to the interpretability of the rules. The Mamdani FIS uses the centroid method for defuzzifying the consequent fuzzy sets and converting them to crisp values. Moreover, Mamdani FIS can be used for MIMO (multiple input multiple output) systems and MISO (multiple input single output) systems [29].

Sugeno-type FIS are acknowledged widely to be more computationally efficient than the Mamdani FIS. The reason for this is that Sugeno FIS uses a linear or constant function for computing the output instead of the fuzzy membership functions. This means that there are an equal number of parameters for the rule consequents as the number of the input rule antecedents. This corresponds to higher degree of freedom in terms of system design. However, Sugeno FIS can only be used for MISO (multiple input single output) systems. The Sugeno FIS uses the weighted average method of computing the crisp output.

Given the differences between both the methods, the main motivation of this comparison is to find the impact of Mamdani and Sugeno FIS on the effort prediction system and also to find out which type of FIS is more suitable and efficient for predicting effort in the highly imprecise and uncertain environment of software development. Even though there

are previous studies [28][29][37][59] which compare both the methods, the underlying models used belong to completely different domains like resonant frequency calculation of rectangular micro strip antennas, space fault detection system, evaluation of quality of experience of audio-visual haptic application and modeling visual perception lab data. A more theoretical and general comparison between the Mamdani and Sugeno FIS is presented in Table 16.

Table 16: Comparison of Mamdani FIS and Sugeno FIS

Mamdani-type FIS	Sugeno-type FIS
More expressive power and interpretability	Less expressive power and interpretability
Output is expressed as fuzzy membership functions	Output is a mathematical function; linear or constant
Output is converted to crisp value by centroid Defuzzification method	Output is calculated by using the Weighted Average method
Less flexibility in system design	More flexibility in system design
Supports MIMO and MISO systems	Supports only MISO systems
Relatively lower computational efficiency	Higher computational efficiency
Relatively higher computation time	Lower computation time

5.4.1. Design details for comparing the Mamdani FIS and Sugeno FIS

Both the FIS have been compared in the context of the simplified framework for effort prediction. The effort prediction system has 6 inputs and 1 output. The process of fuzzy inference starts with the construction of the initial FIS. Construction refers to the process of defining the fuzzy membership functions for the input and outputs, and also to the development of the fuzzy rule base. This is followed by training the Fuzzy Inference System by employing a dataset of input-output pairs. In the end, the trained Fuzzy

Inference System is subjected to testing by employing another dataset which is discussed in the following subsections.

5.4.1.1. Membership Functions

Antecedent MFs: Initially 3 membership functions were chosen for each of the 6 input attributes. The MF type used is Gaussian. The advantage of using Gaussian Membership function is that it allows obtaining smooth, differentiable surfaces of fuzzy models.

Consequent MFs: 3 membership functions were chosen for the output attribute as well. The MF type is Gaussian. This is only defined for the Mamdani FIS and not the Sugeno FIS as it uses a mathematical linear or constant function to compute the output.

5.4.1.2. Rules Formulation

For both the Mamdani FIS and Sugeno FIS, the complete sets of rules were generated. The number of rules in both the cases is 729 ($3 \times 3 \times 3 \times 3 \times 3 \times 3$).

5.4.1.3. Training the FIS

In case of Mamdani FIS, the training algorithm (Algorithm 4-2) used in Section 4.3.3 is used. The result of training is that the initially defined membership functions are tuned according to the input-output data pairs. The training algorithm approximates the initialized FIS structure to the desired FIS structure according to the training data. The actual effect of training is the change in the shape and location of the membership functions. The change in shape is attributed to the change in the standard deviations of the initialized membership functions, whereas the change in the location of the membership function is attributed to the change in means of the initialized membership functions.

In case of Sugeno FIS, the ANFIS (adaptive neuro-fuzzy inference system) function is used to train the initialized FIS. ANFIS uses the back propagation algorithm or a combination of the least squares method and the back propagation to approximate the FIS structure according to the input-output data pairs. For the sake of a fair comparison, we use only the back propagation algorithm for training the effort prediction system. The output membership functions are not present in Sugeno FIS, hence ANFIS allows us to choose between a linear or constant output (a constant output is chosen). The antecedent membership functions are tuned similarly to the Mamdani FIS by the changes in the shape and locations.

5.4.1.4. Testing the FIS

Both the systems are subjected to the hold-out cross validation form of testing. The trained FIS are provided with inputs which predict the output values. The predicted values are compared with the actual output values for calculating performance error measures such as the Mean Absolute Relative Error (MARE). This helps in comparing the prediction accuracy of the Mamdani FIS and the Sugeno FIS.

5.5. Impact of Pairwise Combinations on the performance of the Simplified Effort Prediction Framework

Pairwise Combinations is basically a testing strategy used to generate test cases for testing of software systems. Testing all pairwise (2-way) interactions between input components helps to reveal the combinatorial explosion problem and can ensure the detection of 50 – 97 percent of faults [48]. Although using pairwise testing gives a good percentage of reduction in fault coverage, empirical studies show that pairwise testing is not sufficient enough for highly interactive systems and constructing a minimum test set for combinatorial interaction is still a NP complete problem [48].

To understand the concept of pairwise testing, the example used by Ghazi et al [25] is presented as follows;

Consider different versions of an online portal system developed in two scripting languages (JavaScript and VbScript) and uses Apache, IIS, or PWS webserver. A user

may view the portal using Netscape or Internet Explorer. In this example the system can be considered to be composed of instances of three component classes namely Scripting Language, Webserver, and Browser. For the system tester, two sample test configurations could be {JavaScript, Internet Explorer, and Apache} and {VbScript, Internet Explorer, IIS}. In other words, the system tester will need to test such different configurations to make sure that the system works properly in all configurations and not just some. It is obvious that the total number of configurations in this case is $2 \times 3 \times 2 = 12$ different configurations.

As the complexity of the system to be tested increases, the total number of configurations also increases. It becomes a very difficult task for system testers to design test configurations for the complete system. Pairwise Testing is one such strategy which aids in minimizing the number of configurations to be tested. Pairwise Testing is widely used in the industry and as such there are numerous tools available.

Analogous to the problem of testing is the problem of defining rules in the FLS. The curse of dimensionality is a widely faced problem in the field of Fuzzy Logic Systems. A brief discussion on this problem can be seen in Section 4.3.2. We have investigated the impact of using pairwise combinations for defining rules on the accuracy of the effort prediction system generated using the proposed simplified framework.

The effort prediction system as initialized in Section 4.4.1 is used. The only difference is in the formulation of the rule base. The actual system contains 729 rules corresponding to

3 membership functions for 6 inputs. When pairwise combination is used, the rule base contains only 14 rules which are generated from a tool which uses combinatorial algorithm for pairwise testing. The rules are as follows;

1. If **simpleAC** is *low* and **averageAC** is *low* and **complexAC** is *low* and **simpleUC** is *low* and **averageUC** is *low* and **complexUC** is *low* then **EFFORT** is *low*
2. If **simpleAC** is *low* and **averageAC** is *low* and **complexAC** is *medium* and **simpleUC** is *high* and **averageUC** is *medium* and **complexUC** is *low* then **EFFORT** is *medium*
3. If **simpleAC** is *low* and **averageAC** is *medium* and **complexAC** is *low* and **simpleUC** is *low* and **averageUC** is *high* and **complexUC** is *medium* then **EFFORT** is *medium*
4. If **simpleAC** is *low* and **averageAC** is *medium* and **complexAC** is *medium* and **simpleUC** is *medium* and **averageUC** is *medium* and **complexUC** is *medium* then **EFFORT** is *medium*
5. If **simpleAC** is *low* and **averageAC** is *high* and **complexAC** is *high* and **simpleUC** is *high* and **averageUC** is *high* and **complexUC** is *high* then **EFFORT** is *high*
6. If **simpleAC** is *medium* and **averageAC** is *low* and **complexAC** is *low* and **simpleUC** is *medium* and **averageUC** is *medium* and **complexUC** is *high* then **EFFORT** is *medium*
7. If **simpleAC** is *medium* and **averageAC** is *low* and **complexAC** is *high* and **simpleUC** is *high* and **averageUC** is *high* and **complexUC** is *medium* then **EFFORT** is *medium*
8. If **simpleAC** is *medium* and **averageAC** is *medium* and **complexAC** is *medium* and **simpleUC** is *low* and **averageUC** is *high* and **complexUC** is *low* then **EFFORT** is *medium*
9. If **simpleAC** is *medium* and **averageAC** is *medium* and **complexAC** is *high* and **simpleUC** is *medium* and **averageUC** is *low* and **complexUC** is *low* then **EFFORT** is *medium*

10. If **simpleAC** is *medium* and **averageAC** is *high* and **complexAC** is *low* and **simpleUC** is *high* and **averageUC** is *low* and **complexUC** is *medium* then **EFFORT** is *medium*
11. If **simpleAC** is *medium* and **averageAC** is *high* and **complexAC** is *medium* and **simpleUC** is *low* and **averageUC** is *medium* and **complexUC** is *high* then **EFFORT** is *high*
12. If **simpleAC** is *high* and **averageAC** is *low* and **complexAC** is *high* and **simpleUC** is *low* and **averageUC** is *medium* and **complexUC** is *medium* then **EFFORT** is *medium*
13. If **simpleAC** is *high* and **averageAC** is *medium* and **complexAC** is *medium* and **simpleUC** is *high* and **averageUC** is *low* and **complexUC** is *high* then **EFFORT** is *medium*
14. If **simpleAC** is *high* and **averageAC** is *high* and **complexAC** is *low* and **simpleUC** is *medium* and **averageUC** is *high* and **complexUC** is *low* then **EFFORT** is *high*

The effort prediction system is trained using these 14 rules and the accuracy of predicting the output effort is tested. The results of the impact of pairwise combinations can be seen in Chapter 6.

5.6. Impact of Design Parameters on the performance of the Simplified Effort Prediction Framework

Designing an FLS is not an easy task. Various considerations need to be kept in mind while designing an FLS. According to Mendel [60], we must decide on the kind of fuzzification (singleton or non-singleton), types of membership functions (triangular, trapezoidal, Gaussian, piece-wise linear), parameters of membership functions,

composition (max-min, max-product), implication (minimum, product), and defuzzifier (centroid, center of sums, height, modified height). Different combinations of the aforementioned possibilities can result in 131,072 different FLS. Figure 16 shows the choices that need to be made before designing an FLS.

The parameters of the FLS can affect the behavior of the FLS. Muzaffar [67] investigated the impact of the design parameters such as the Height Defuzzification vs. Modified Height Defuzzification and triangular membership functions vs. Gaussian membership functions.

In the context of our work, we investigate the impact of defining the parameters of the membership functions. In course of this work, we came across two different methods of defining the parameters (means of antecedents MFs, standard deviation of antecedent MFs and centers of consequent MFs) related to the Gaussian membership functions. The first method is the same as described in Section 4.3.1. Nevertheless, both the methods are described in the sequel.

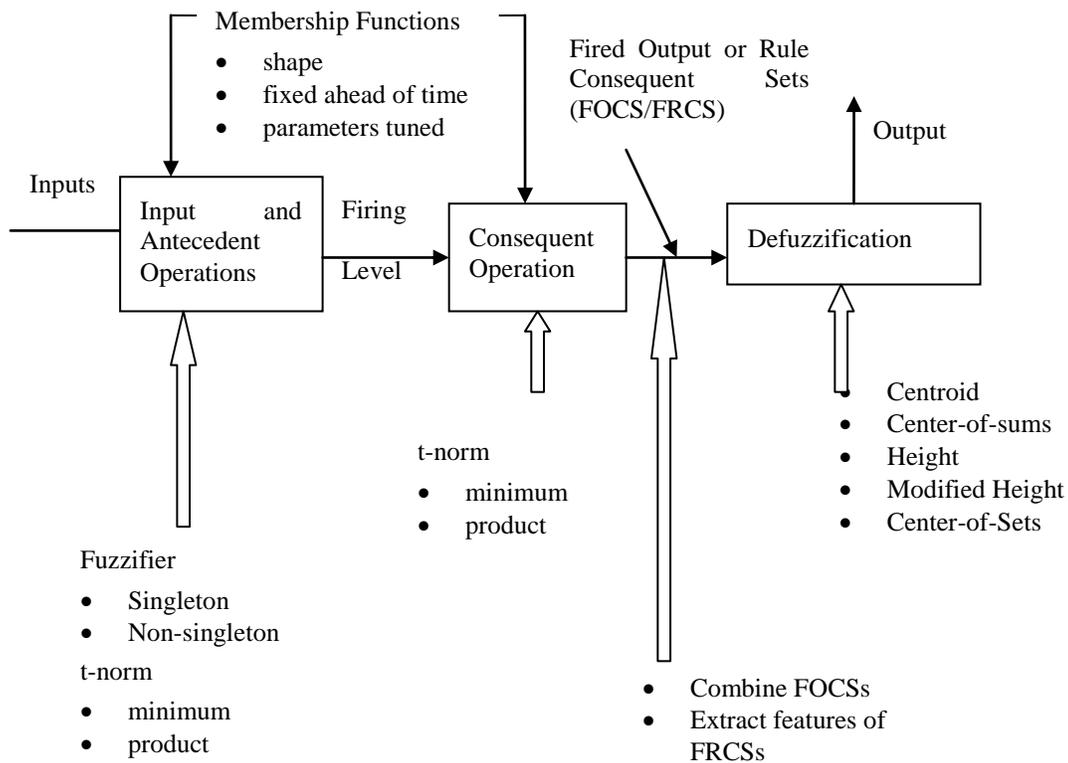


Figure 16: Choices that need to be made before designing an FLS

5.6.1. Method 1 for defining the parameters of the Gaussian membership functions

To define the antecedent parameters based on the numerical data set, the first step is to calculate the *mean* (M) and *standard deviation* (σ) values for each input attribute. Then depending on the number of fuzzy sets for each input attribute, define the means of the membership functions ‘M’ as follows;

$$M_{mf1} = M - \alpha.\sigma$$

$$M_{mf2} = M$$

$$M_{mf3} = M + \alpha.\sigma$$

Where, ‘ α ’ is a constant that should be defined properly (using experience) so as to cover the complete interval range of a particular input attribute. M_{mfi} refers to the mean of the i^{th} membership function. The standard deviation values for all the 3 membership functions are kept to the same value of ‘ σ ’.

To define the consequent parameters based on the numerical data set, i.e. the centers of consequent fuzzy sets, the lowest and highest values for a particular output attribute are extracted from the numerical data. Then the range is calculated as the difference between the highest and lowest value. Dividing the range by the number of membership functions minus 1, gives the increment factor. To get the initial values for the center of consequents (c), start with the lowest value, keep adding the increment factor until all the consequent fuzzy sets have the center (c) values. While defining these values, one has to be careful to cover the domain interval of the output attributes.

5.6.2. Method 2 for defining the parameters of the Gaussian membership functions

In method 2, based on the numerical data sets, the means of the membership functions are defined as follows;

M_{mf1} = lowest value in the data set

M_{mf2} = (lowest value + highest value) / 2

M_{mf3} = highest value in the data set

The standard deviation values are calculated as follows;

σ = (highest value – lowest value) / 8

Finally, the centers of consequent values are calculated similar to the way the means for the antecedents are calculated. The difference is in the random selection of these values corresponding to different rules. Suppose, we have a system which uses 3 membership functions for the output attribute and has 81 rules. Since, each rule should have a center of consequent value, and we have just 3 values of centers of consequents, each rule takes a random value from among the 3 values.

Both the methods are applied to an effort prediction system pertaining to the simplified effort prediction framework and results are reported for both the cases.

CHAPTER 6

EXPERIMENTS AND RESULTS

This chapter presents the experimental information and implementation details with regards to the proposed frameworks and other related studies discussed in the previous chapters. Due to unavailability of industrial datasets for validating the proposed frameworks, artificial datasets have been used. The algorithm for artificial data generation is presented in the course of this chapter. The results have been analyzed and discussed based on the Mean Absolute Relative Error (MARE) and prediction accuracy.

6.1. Prediction Accuracy

Prediction Accuracy or prediction at level 'q', i.e. $\text{pred}(q)$ is a quantitative measure that helps in measuring the prediction power of the effort prediction systems by comparing the predicted values and the actual values.

Given a set of ‘n’ projects, and let ‘k’ be the number of projects whose mean absolute relative error is less than equal to q, then prediction accuracy is given as follows;

$$\mathit{pred}(q) = k / n$$

An acceptable level for mean absolute relative error is something less than or equal to 0.25 as suggested by Conte et al [17]. For example, if $\mathit{pred}(0.25)$ is 0.72, then 72% of the predicted values fall within 25% of the original values.

The mean absolute relative error (MARE) is given as follows;

$$\text{MARE} = \frac{|\mathit{actual\ effort} - \mathit{predicted\ effort}|}{\mathit{actual\ effort}}$$

6.2. Artificial Data Generation

While dealing with systems which utilize the concept of machine learning techniques, it is imperative that sufficient data be available for training and testing these systems. Historical dataset plays a vital role in training and testing adaptive FLS based prediction systems generated using the frameworks [67]. Unfortunately, such data (use case based effort prediction) is not readily available in the public domain or other data repositories.

Henceforth, the need for generating artificial datasets arises which is realized by algorithm 6-1. Due to the wide acceptance of the mathematical equations in the UCP method, they have been utilized for generation of artificial data. A point worth mentioning is that the proposed frameworks are general and can accept any factor

measure as inputs. The data generation algorithm follows from the approach adopted by Ahmed et al [2] and Muzaffar et al [66] wherein the COCOMO equations have been employed to generate artificial data.

Algorithm 6-1: Generation of artificial data for training and validation data sets

1. Generate random values for a desired number of data points, say 'K', in the domain intervals [EF⁻ EF⁺], [TCF⁻ TCF⁺], [AC⁻ AC⁺], [UC⁻ UC⁺], [PF⁻ PF⁺] using Uniform Distribution for all the 8 EF factors, 13 TCF factors, 3 types of Actors (simple, average, complex), 3 types of Use Cases (simple, average, complex) and productivity factor PF.

2. For each factor value generated in step 1, compute the Efactor, Tfactor, UAW, UUCW values by multiplying each factor by its weight factor as follows;

$$\Sigma (EF \text{ Factor Value}) * (\text{Weighting Factor}) * = Efactor$$

$$\Sigma (TCF \text{ Factor Value}) * (\text{Weighting Factor}) * = Tfactor$$

$$\Sigma (\text{Actor Factor Value}) * (\text{Weighting Factor}) * = UAW$$

$$\Sigma (\text{Use Case Factor Value}) * (\text{Weighting Factor}) * = UUCW$$

3. For each data point, compute the final EF, final TCF, and UCP values using the following equations obtained from the UCP method;

$$(-0.03 * Efactor) + 1.4 = EF$$

$$(0.01 * Tfactor) + 0.6 = TCF$$

$$(UAW + UUCW) * EF * TCF = UCP$$

4. For each data point, compute the EFFORT value as follows;

$$UCP * PF = EFFORT$$

5. Repeat steps 2 to 4 until 'K' data points have been generated, with each data point consisting of attributes in step 1 as the inputs and EFFORT as the output.
6. Partition the 'K' data points into training and testing data sets. The ratio of training data sets to the testing data sets is 70:30.

6.3. Experiment 1: Evaluating the prediction accuracy of the f-UCP model

This experiment deals with the performance evaluation of the proposed f-UCP model discussed in Chapter 4, for handling imprecision in software development effort prediction. In the following sub sections, the implementation details and the results (tabular values and graphs) are presented.

6.3.1. Implementation Details

The f-UCP model is implemented using Sugeno type of Fuzzy Logic System which applies to all the 4 components in the architectural design. The number of membership functions used for each component is 3 and the type is Gaussian shouldered. Since, the number of inputs to the components in the 1st layer is large, subtractive clustering (see algorithm 4-1 in Section 4.2.2) is used to define the rule base of the 3 components. The last component in the 2nd layer has 4 inputs and hence all possible combinations of inputs are used to define the rule base. The f-UCP model is trained using ANFIS (Adaptive Neuro Fuzzy Inference System) which utilizes a combination of back propagation and least squares estimation learning. The experiment is run for 5 times with different data sets each time. The number of data points in each data set is 100. The training and testing data are in the ratio 70:30.

6.3.2. Results and Discussion

The f-UCP model is a complete fuzzy model which replaces the existing mathematical model of UCP. As such, prior to fuzzifying the UCP model, positive outcomes corresponding to a low margin of error was expected. The extremely low MARE values in the training part (see Table 17) highlight the efficiency of ANFIS in training the system and a high average testing prediction accuracy of 95.33% confirms that fuzzifying the UCP model (f-UCP) has positive outcomes. This also establishes the fact that f-UCP produces acceptable results and is better than the mathematical UCP model.

Table 17: Summary of Prediction Quality on f-UCP using five different datasets, showing pred(25) and MARE values on training and testing datasets

Experimental Run	Training		Testing	
	pred(25)	MARE	pred(25)	MARE
1	100	$4.8593 * 10^{-7}$	93.3407	0.0628
2	100	$8.2661 * 10^{-7}$	100	0.0487
3	100	$5.1713 * 10^{-7}$	93.3407	0.0637
4	100	$6.1311 * 10^{-7}$	90.0111	0.1046
5	100	$4.5803 * 10^{-7}$	100	0.0499

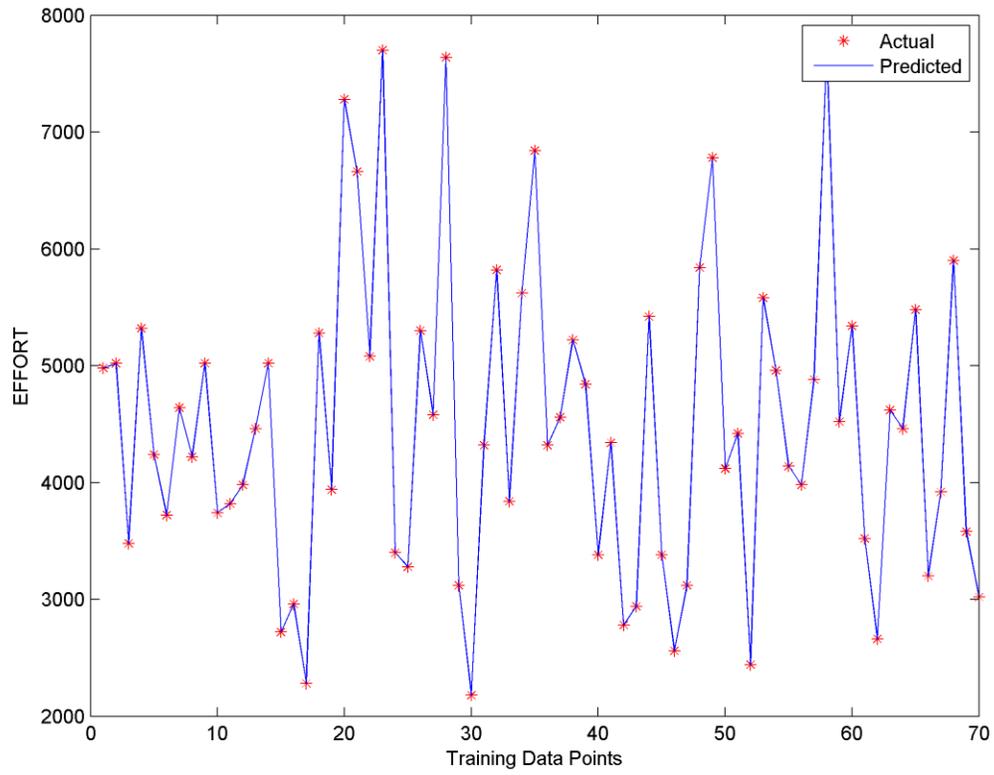


Figure 17: Prediction of effort using trained f-UCP on training dataset

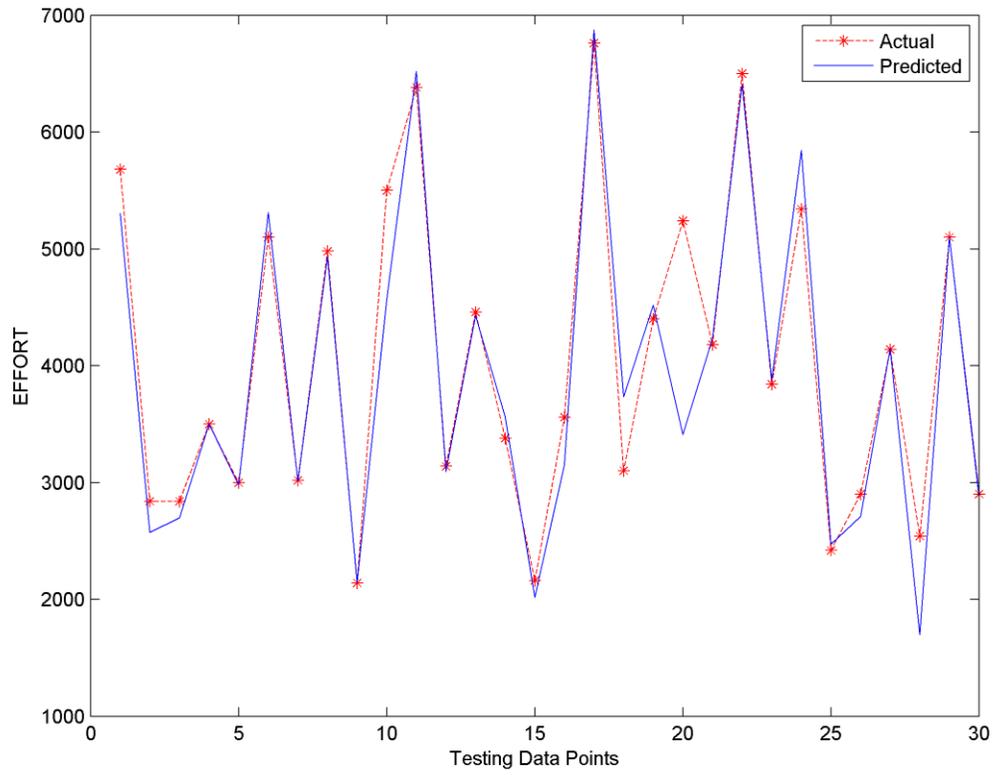


Figure 18: Prediction of effort using trained f-UCP on testing dataset

6.4. Experiment 2: Impact of TCF and EF on use-case based effort prediction

This experiment aims at investigating the impact of technical complexity factors and experience factors on effort prediction which is achieved by using factor analytic techniques to resolve and reduce the factors into components.

6.4.1. Implementation Details

The Principal Components Analysis method is used for reducing the number of factors and resolving them into individual components. Kaiser Meyer Olkin test and Bartlett's test of Sphericity is used, the details for which can be found in Section 5.1.1 and Section 5.1.2. For rotations, the Direct Oblimin method is adopted and the maximum number of iterations for convergence is kept to be 4 and 3 for TCF and EF respectively.

6.4.2. Results and Discussion

The KMO test and Bartlett's test values (Table 18 and Table 22) obtained are significant and imply that distinct and reliable factors can be produced as a result of conducting PCA. The reduction of TCF and EF factors from 13 and 8 to 6 and 5 respectively can be seen in the pattern matrix generated (Table 20 and Table 24). A point worth mentioning is that, the reduction of factors according to the principal component analysis is purely dependent on the data at hand. As such it cannot be generalized that the specific factors which have been removed as a consequence of PCA do not have any influence of the

effort prediction process. The inclusion and exclusion of the various factors varies with the nature of the available data.

Table 18: KMO and Bartlett's Test Results on TCF

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.459
	Approx. Chi-Square	368.514
Bartlett's Test of Sphericity	df	15
	Sig.	.000

Table 19: Total Variance Explained - TCF

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	1.806	30.107	30.107	1.806	30.107	30.107
2	1.479	24.654	54.761	1.479	24.654	54.761
3	1.041	17.342	72.103	1.041	17.342	72.103
4	.899	14.990	87.093			
5	.523	8.721	95.814			
6	.251	4.186	100.000			

Table 20: Pattern Matrix - TCF

	Component		
	1	2	3
VAR00002	.924		
VAR00003	.912		
VAR00005		.855	
VAR00009		.821	
VAR00006			.835
VAR00012			.623

Table 21: Component Correlation Matrix - TCF

Component	1	2	3
1	1.000	.048	.061
2	.048	1.000	.144
3	.061	.144	1.000

Table 22: KMO and Bartlett's Test on EF

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.652
	Approx. Chi-Square	307.174
Bartlett's Test of Sphericity	df	10
	Sig.	.000

Table 23: Total Variance Explained - EF

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	2.093	41.854	41.854	2.093	41.854	41.854
2	1.143	22.850	64.704	1.143	22.850	64.704
3	.877	17.540	82.244			
4	.535	10.693	92.937			
5	.353	7.063	100.000			

Table 24: Pattern Matrix - EF

	Component	
	1	2
VAR00001	.810	
VAR00002	.830	
VAR00003	.856	
VAR00007		.740
VAR00008		.750

Table 25: Component Correlation Matrix - EF

Component	1	2
1	1.000	.053
2	.053	1.000

6.5. Experiment 3: Evaluating the prediction accuracy of the proposed framework

This experiment deals with validating the performance of the proposed adaptive use-case based effort prediction framework using fuzzy logic. More details regarding the framework can be found in Chapter 4.

6.5.1. Implementation Details

The proposed framework is implemented using Mamdani type of Fuzzy Logic System which applies to all the 5 components in the architectural design. The number of membership functions used for each component is 3 and the type is Gaussian shouldered. All possible combinations of inputs are used to define the rule base for all the components in the system. The system is trained using a back propagation algorithm (Steepest Descent Approach). The experiment is run for 5 times with different data sets each time. The number of data points in each data set is 140. The training and testing data are 100 and 40 respectively.

6.5.2. Results and Discussion

Minimum training and testing error values (MARE) of 7.11% and 9.94% have been reported. Additionally, the average testing prediction accuracy is 89.56% which highlights the performance of the prediction system obtained from the proposed framework. This highlights the fact that the proposed framework can be used for evolving

efficient effort prediction systems according to the needs and requirements of the users. Moreover, the obtained effort prediction systems are transparent and allow the users to incorporate their judgment and experience to further tune the system in terms of its operation which corresponds to the tuning of the rule base of the system. In essence, the users have the discretion to either include or exclude a particular rule from the rule base or the users may include or exclude a particular input factor from the system, thereby making the system more adaptive and receptive to the users' requirements.

Table 26: Summary of Prediction Quality on the Effort Prediction System using five different datasets, showing pred(25) and MARE values on training and testing datasets

Experimental Run	Training		Testing	
	pred(25)	MARE	pred(25)	MARE
1	99.0001	0.0711	87.8121	0.1180
2	95.0005	0.0940	90.0062	0.1142
3	95.0005	0.0881	92.5047	0.1003
4	95.0005	0.1095	92.5047	0.0994
5	94.0004	0.0995	85.0094	0.1734

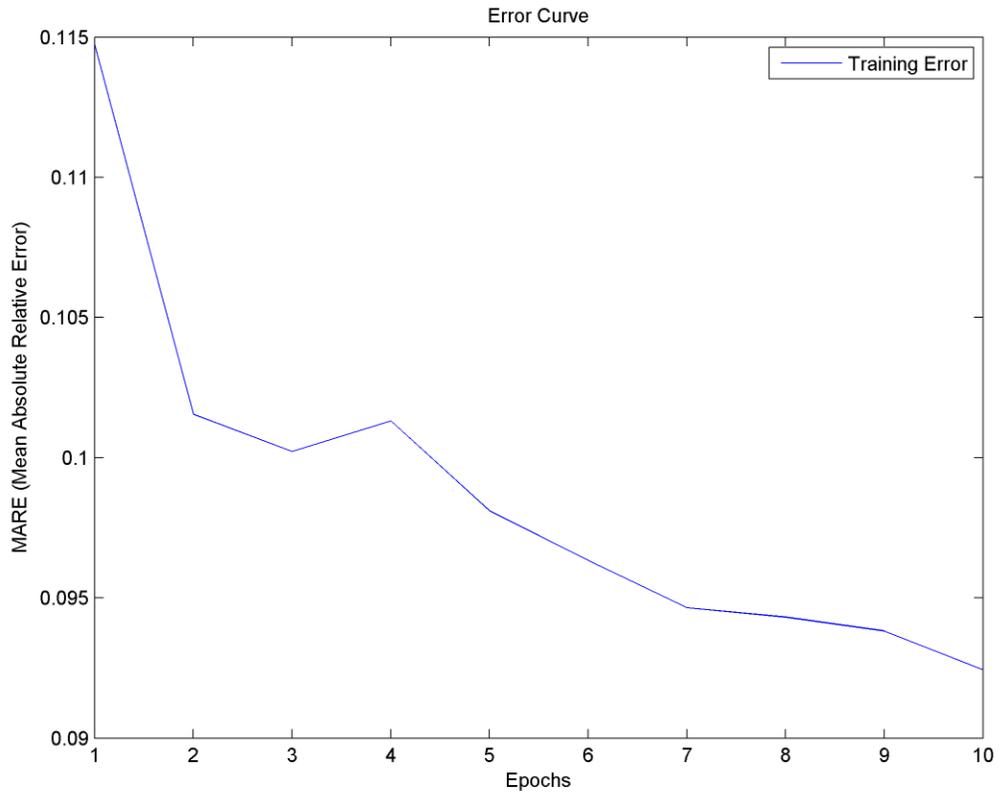


Figure 19: Average MARE graph of training the Effort Prediction System

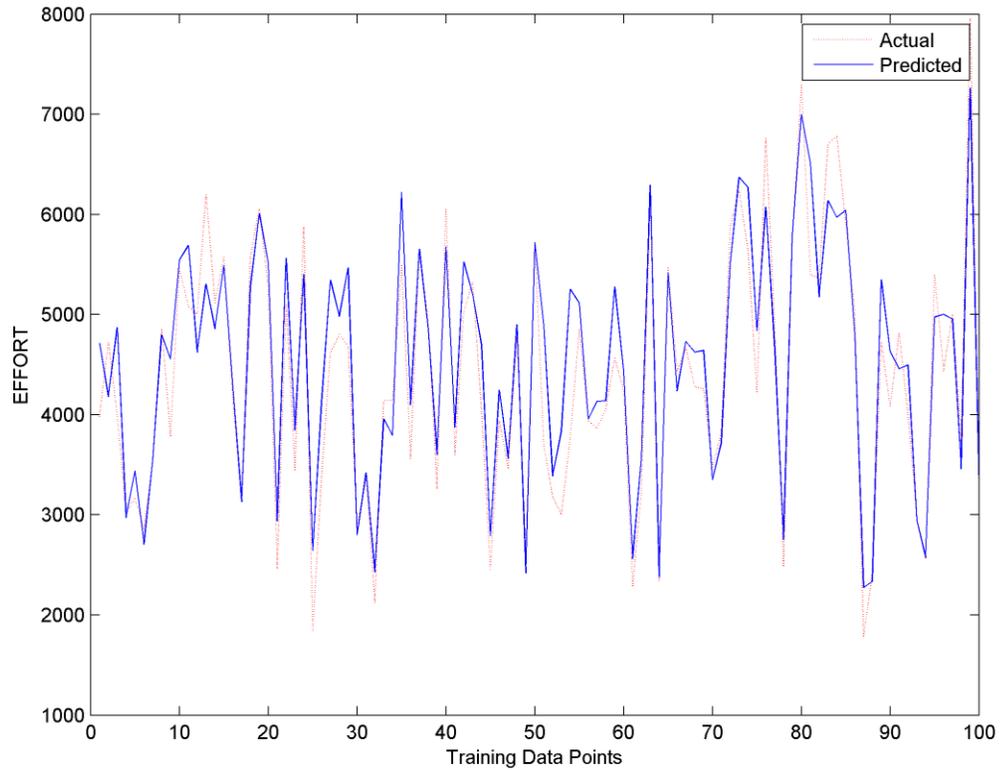


Figure 20: Prediction of effort using the trained Effort Prediction System on training datasets

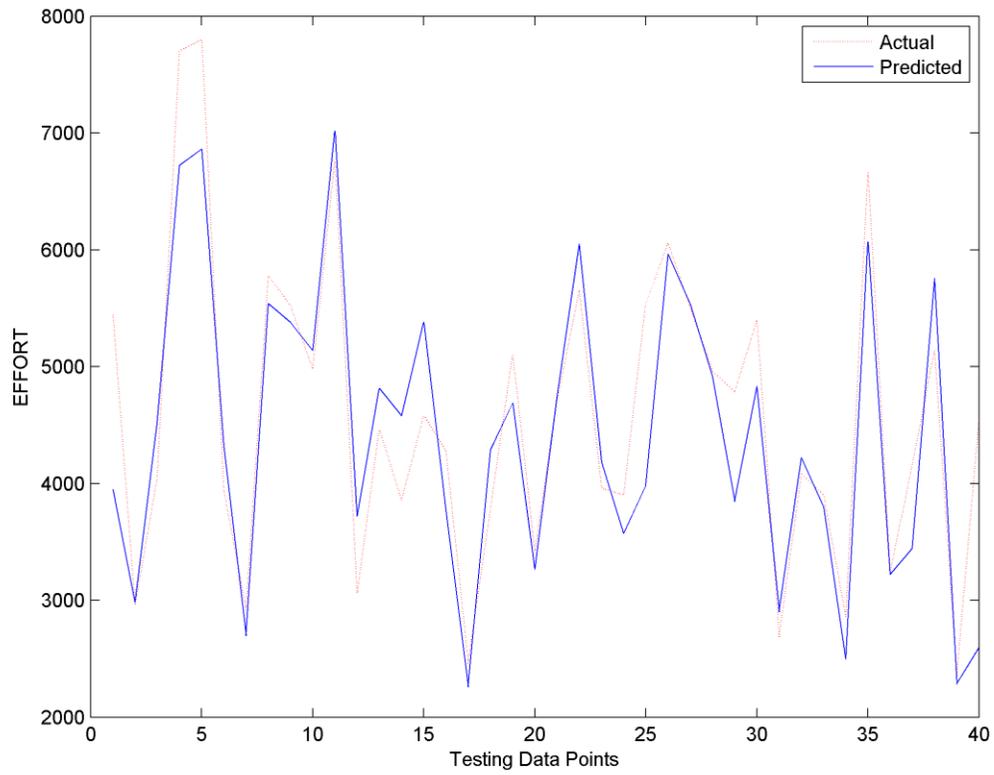


Figure 21: Prediction of effort using the trained Effort Prediction System on testing datasets

6.6. Experiment 4: Evaluating the prediction accuracy of the simplified framework

This experiment aims at validating the performance of the proposed simplified adaptive use-case based effort prediction framework using fuzzy logic. The framework includes just the Use Case elements such as Actors and Use Case information.

6.6.1. Implementation Details

The simplified framework is implemented using Mamdani type of Fuzzy Logic System. The number of membership functions used is 3 and the type is Gaussian shouldered. All possible combinations of inputs are used to define the rule base for the system. The system is trained using a back propagation algorithm (Steepest Descent Approach). The experiment is run for 5 times with different data sets each time. The number of data points in each data set is 100. The training and testing data are in the ratio 70:30.

6.6.2. Results and Discussion

Minimum training and testing error values (MARE) of 7.28% and 16.87% have been reported. Additionally, the average testing prediction accuracy is 74.67% which is acceptable and showcases the performance of the prediction system obtained from the simplified framework. A point to be noted is that the simplified framework differs from the proposed framework just in terms of the exclusion of additional factors.

Table 27: Summary of Prediction Quality on the Effort Prediction System using five different datasets, showing pred(25) and MARE values on training and testing datasets

Experimental Run	Training		Testing	
	pred(25)	MARE	pred(25)	MARE
1	97.1434	0.0889	73.3629	0.1687
2	95.7152	0.0735	76.6926	0.1985
3	95.7152	0.0777	76.6926	0.1816
4	97.1434	0.0728	63.3740	0.2017
5	95.7152	0.0892	83.3518	0.1860

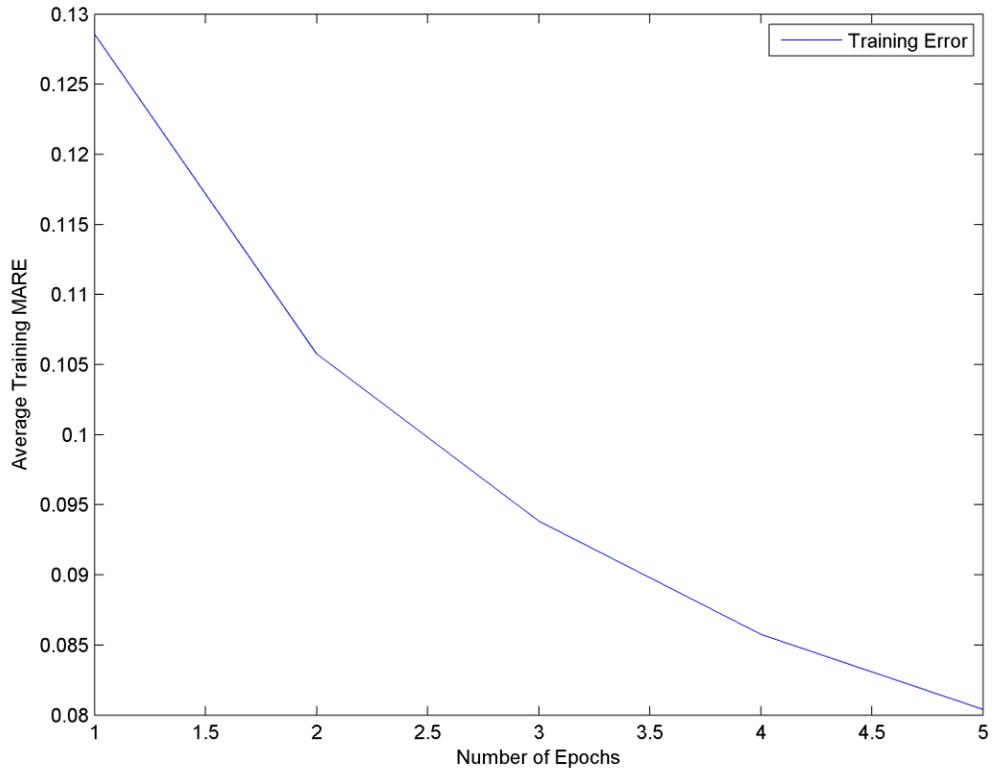


Figure 22: Average MARE graph of training the Effort Prediction System

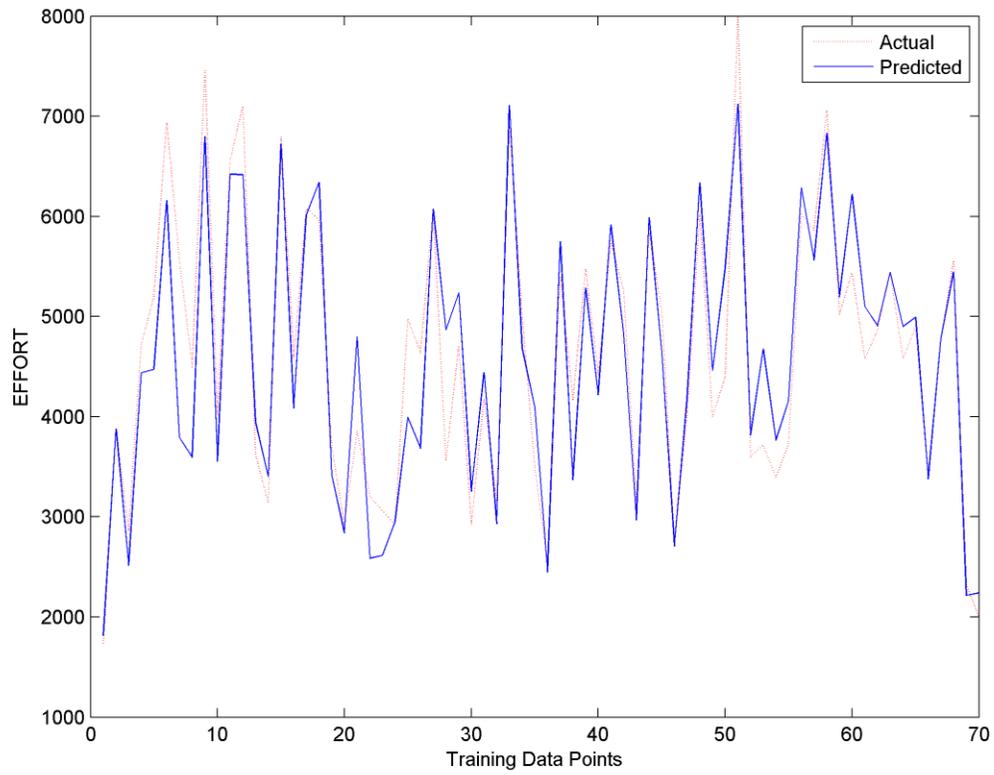


Figure 23: Prediction of effort using the Effort Prediction System on training datasets

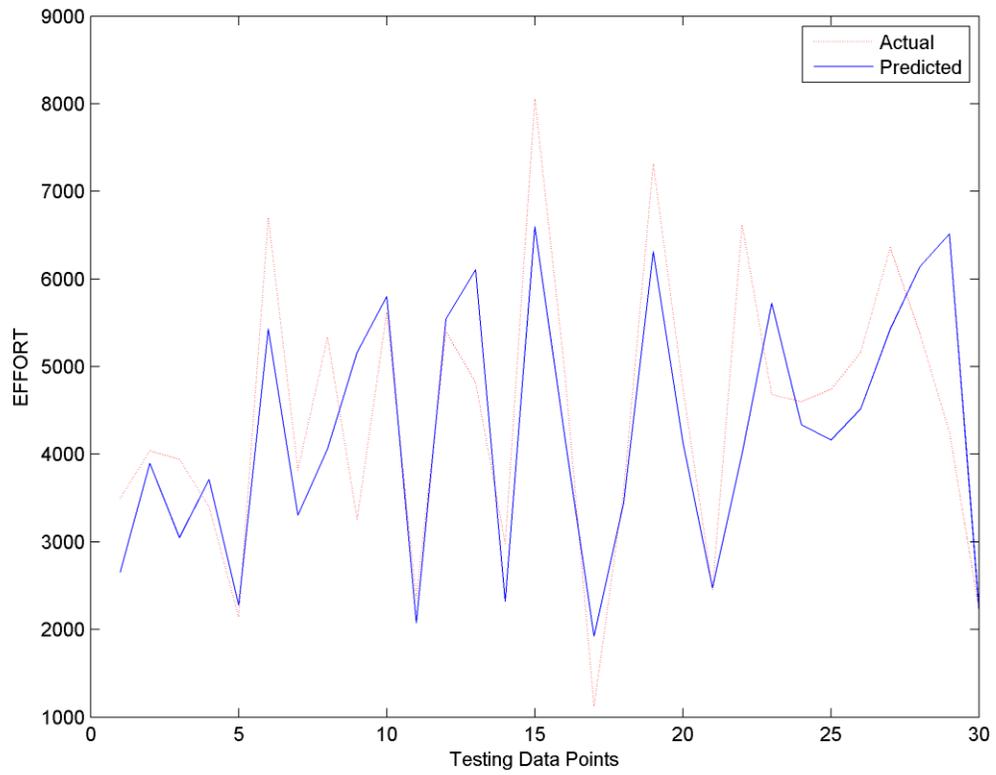


Figure 24: Prediction of effort using the Effort Prediction System on testing datasets

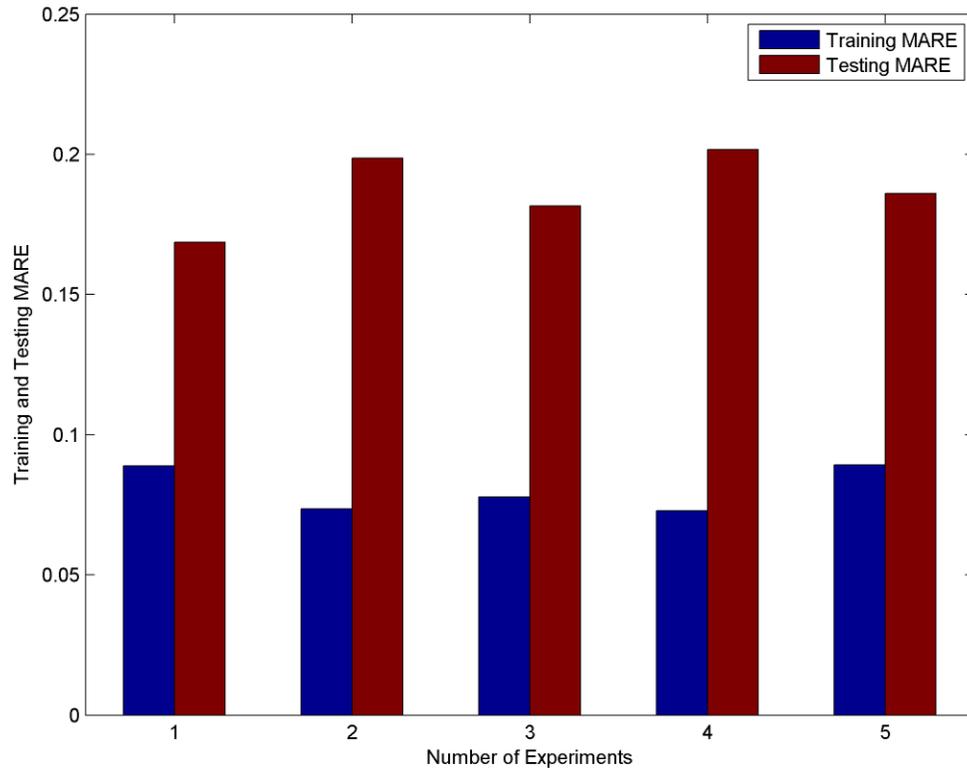


Figure 25: Comparison of training and testing errors (MARE) on the Effort Prediction System

6.7. Experiment 5: Impact of pairwise combinations on prediction accuracy

This experiment is concerned with the comparison of effort prediction systems in the context of the simplified framework; one using all possible combinations of inputs to create the rule base and the other using pairwise combinations to create the rule base. As mentioned earlier, the predictions systems being tested are obtained from the simplified framework; needless to say the implementation details are the same (refer to Section 6.6.1) and need no explicit further details.

6.7.1. Results and Discussion

It is evident from Table 28 that the normal prediction system using all possible combinations of inputs to define the rule base outperforms the other prediction system using pairwise combinations. The prediction accuracies (training and testing) clearly exhibit the difference in the performance of both the systems. Moreover, the average training error graph (Figure 26) also highlights the superiority of the normal prediction system as opposed to the pairwise based prediction system. Even though pairwise combinations are an effective strategy for producing minimal number of test cases in the software testing phase, it is not an effective approach in the context of software effort prediction systems. Overall, from the results obtained, it can be safely concluded that the pairwise combinations are ineffective in terms of minimizing the rules in the rule bases for fuzzy based effort prediction systems.

Table 28: Summary of Prediction Quality on the Normal and Pairwise Effort Prediction Systems using five different datasets, showing pred(25) and MARE values on training and testing datasets

Experimental Run	Normal				Pairwise			
	Training		Testing		Training		Testing	
	pred(25)	MARE	pred(25)	MARE	pred(25)	MARE	pred(25)	MARE
1	95.7152	0.0977	80.0222	0.1861	47.1536	0.2762	46.7259	0.2900
2	100	0.0795	63.3740	0.1934	72.8627	0.1776	66.7037	0.1999
3	100	0.0575	80.0222	0.1943	80.0041	0.1650	70.0333	0.2098
4	95.7152	0.0818	63.3740	0.2065	65.7213	0.2161	66.7037	0.2603
5	92.8586	0.0910	86.6815	0.1354	58.5799	0.2945	63.3740	0.2587

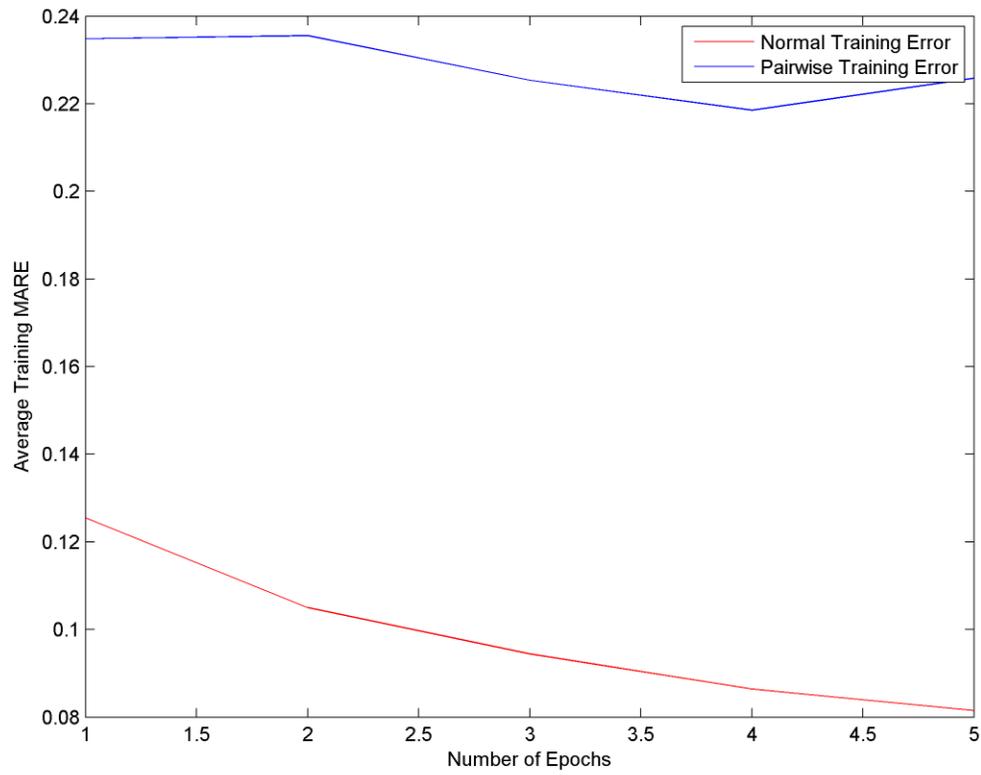


Figure 26: Average MARE graph of training the Normal and Pairwise Effort Prediction System

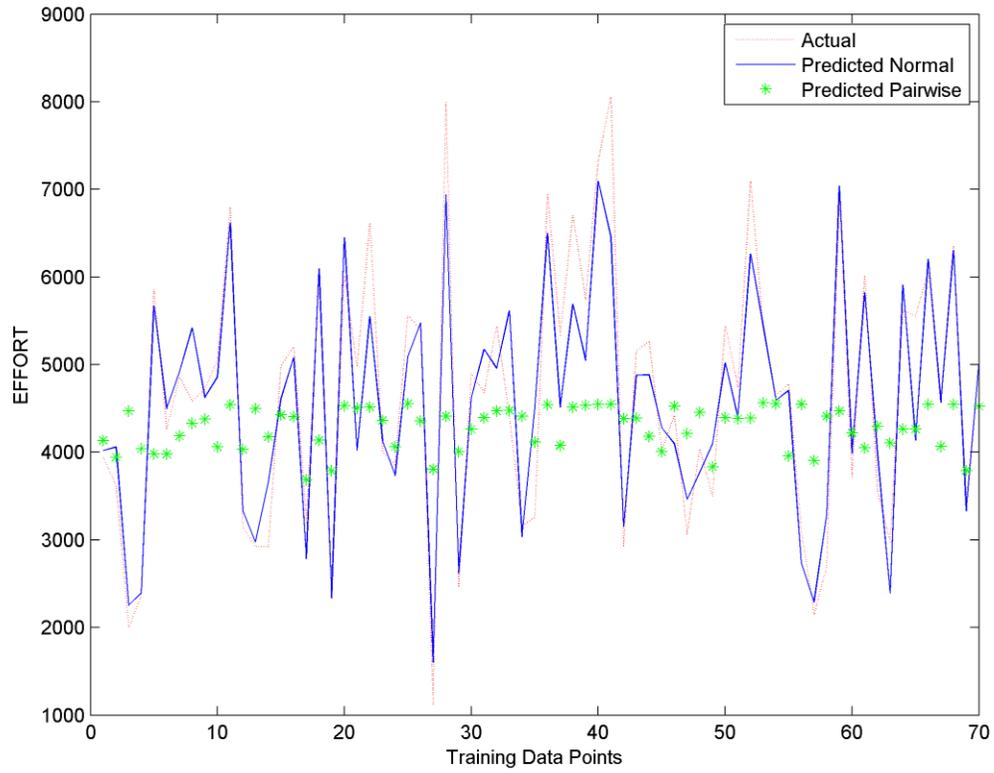


Figure 27: Prediction of effort on Normal and Pairwise Effort Prediction Systems using training datasets

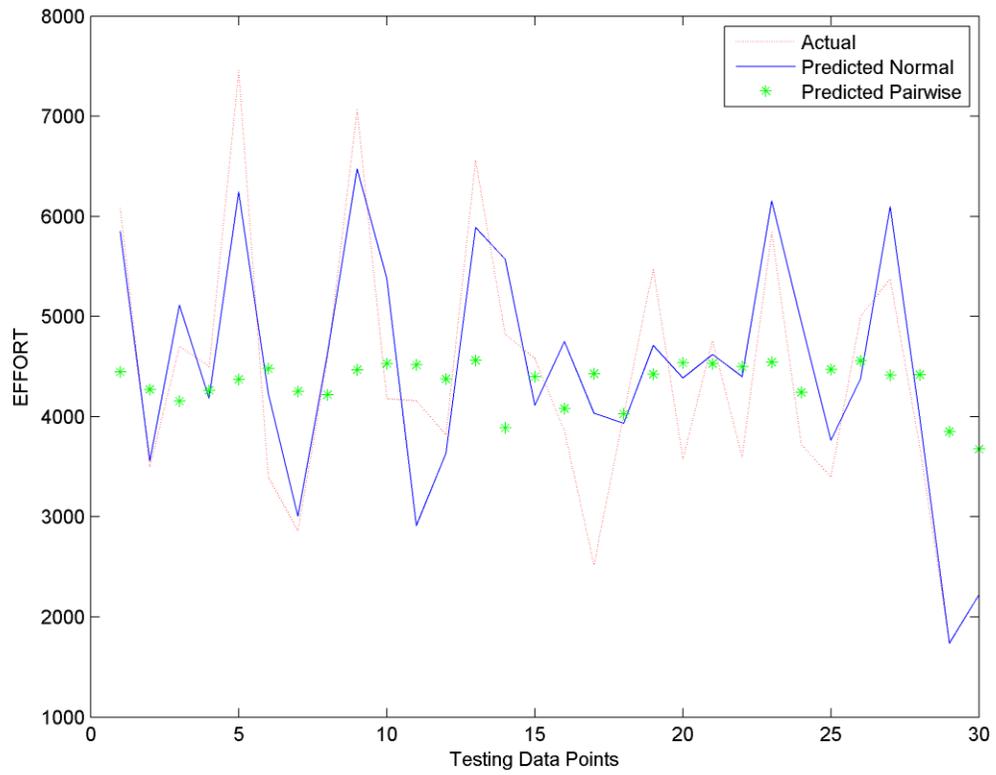


Figure 28: Prediction of effort using Normal and Pairwise Effort Prediction Systems on testing datasets

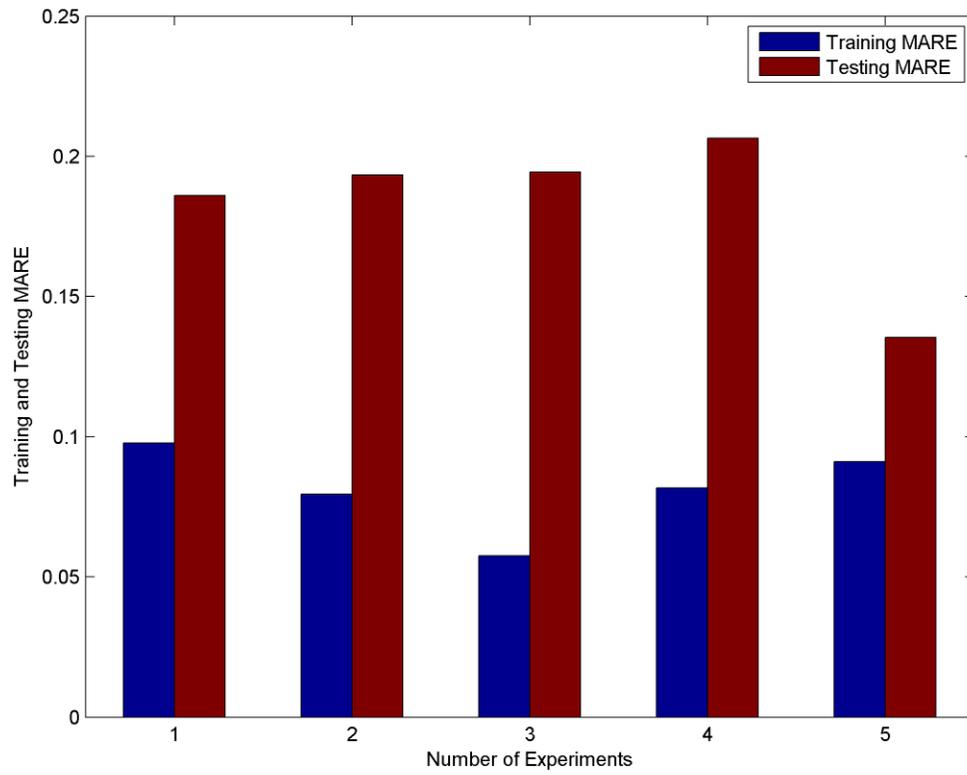


Figure 29: Comparison of training and testing errors (MARE) on the Normal Effort Prediction System

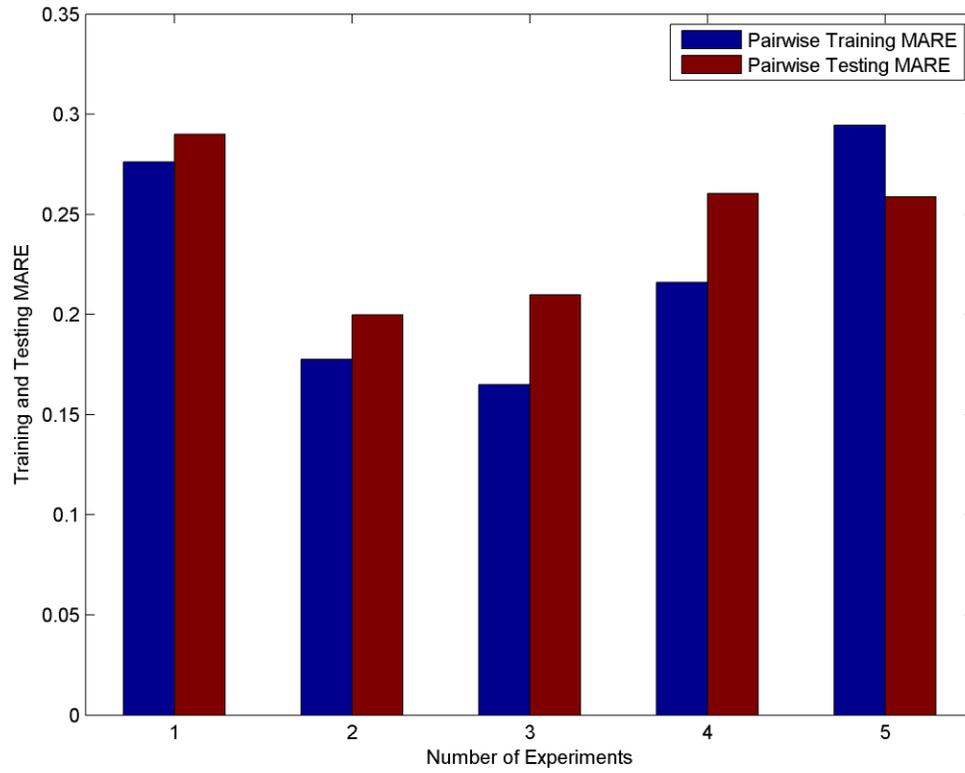


Figure 30: Comparison of training and testing errors (MARE) on the Pairwise Effort Prediction System

6.8. Experiment 6: Comparison of Mamdani type FLS vs. the Sugeno type FLS

This experiment is concerned with evaluating the performances of the prediction systems utilizing Mamdani and Sugeno type of fuzzy logic systems (FLS) in the context of the simplified framework. The implementation details for this experiment can be found in Section 6.6.1.

6.8.1. Results and Discussion

The experiment produces controversial results in the essence that, the Sugeno type of FLS outperforms the Mamdani FLS in terms of training (Figure 32), whereas the Mamdani FLS clearly outperforms the Sugeno FLS in terms of testing (Figure 33). As such it becomes difficult to establish the superiority of one system over the other which is in conformation to the results of other comparisons which state that the choice of a particular system is dependent on the application domain and application data.

Table 29: Summary of Prediction Quality on the Mamdani and Sugeno Effort Prediction Systems using five different datasets, showing pred(25) and MARE values on training and testing datasets

Experimental Run	Mamdani				Sugeno			
	Training		Testing		Training		Testing	
	pred(25)	MARE	pred(25)	MARE	pred(25)	MARE	pred(25)	MARE
1	92.8586	0.1033	70.0333	0.2174	100	$6.1017 * 10^{-8}$	33.4073	0.4598
2	91.4303	0.1108	73.3629	0.1584	100	$5.7148 * 10^{-8}$	23.4184	0.4148
3	97.1434	0.0830	76.6926	0.1544	100	$8.5959 * 10^{-8}$	30.0777	0.4416
4	94.2869	0.1153	60.0444	0.2136	100	$5.7706 * 10^{-8}$	16.7592	0.5274
5	95.7152	0.0796	80.0222	0.1500	100	$8.4250 * 10^{-8}$	16.7592	0.4918

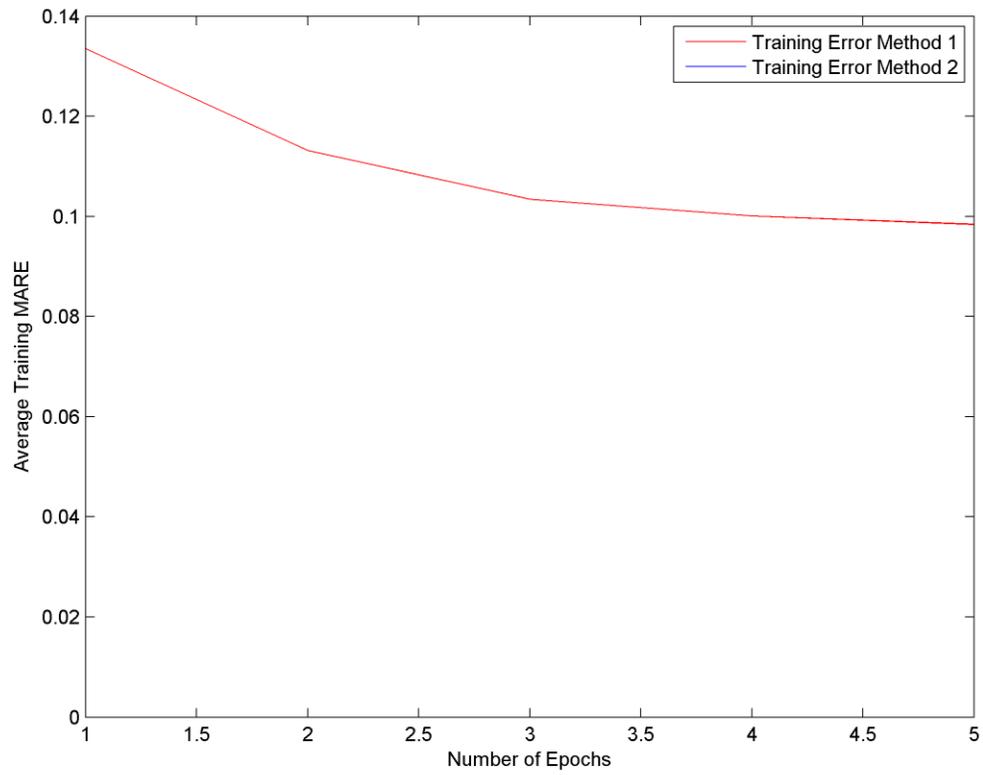


Figure 31: Average MARE graph of training the Mamdani (Method 1) and Sugeno (Method 2) Effort Prediction Systems

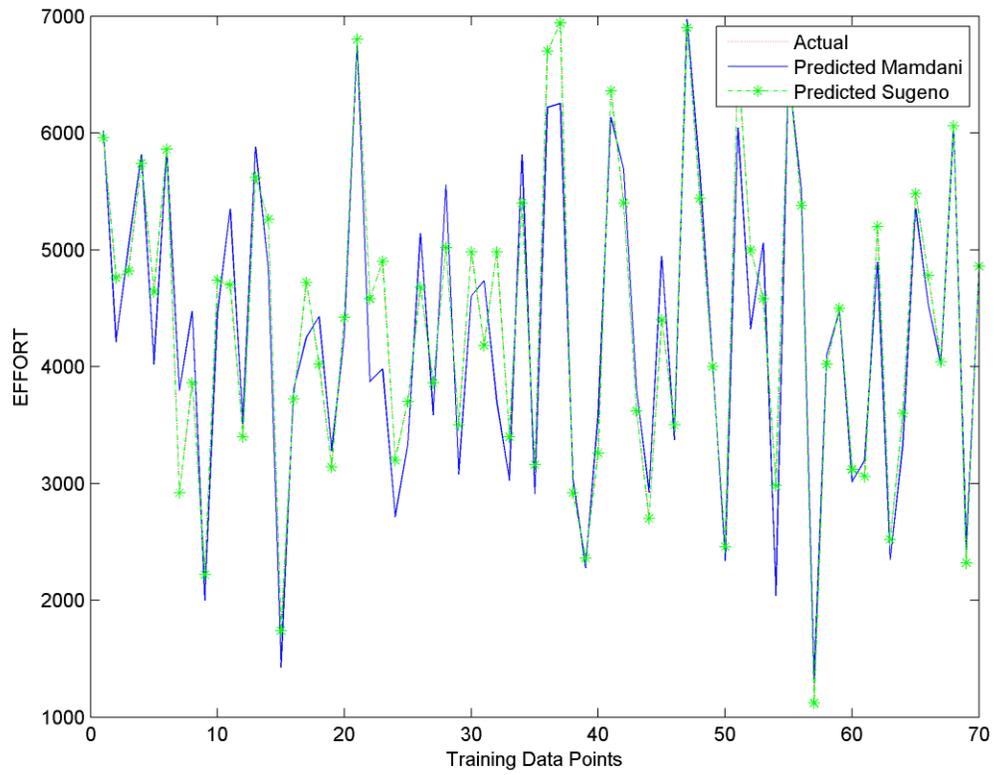


Figure 32: Prediction of effort using the Mamdani and Sugeno Effort Prediction Systems on training datasets

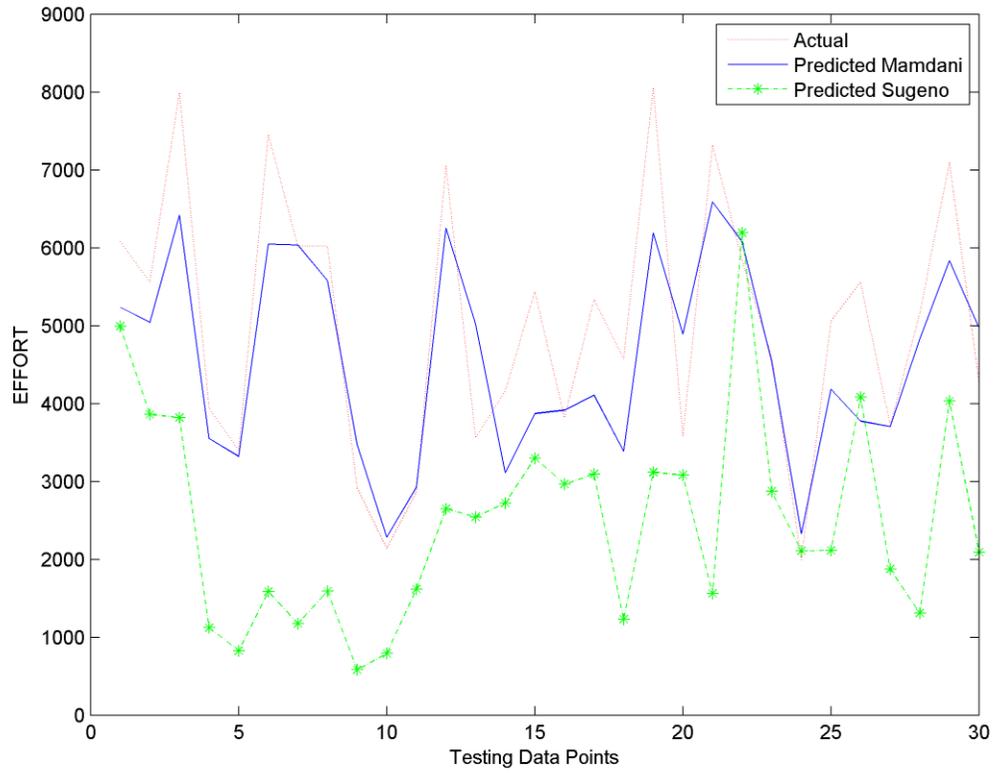


Figure 33: Prediction of effort using the Mamdani and Sugeno Effort Prediction Systems on testing datasets

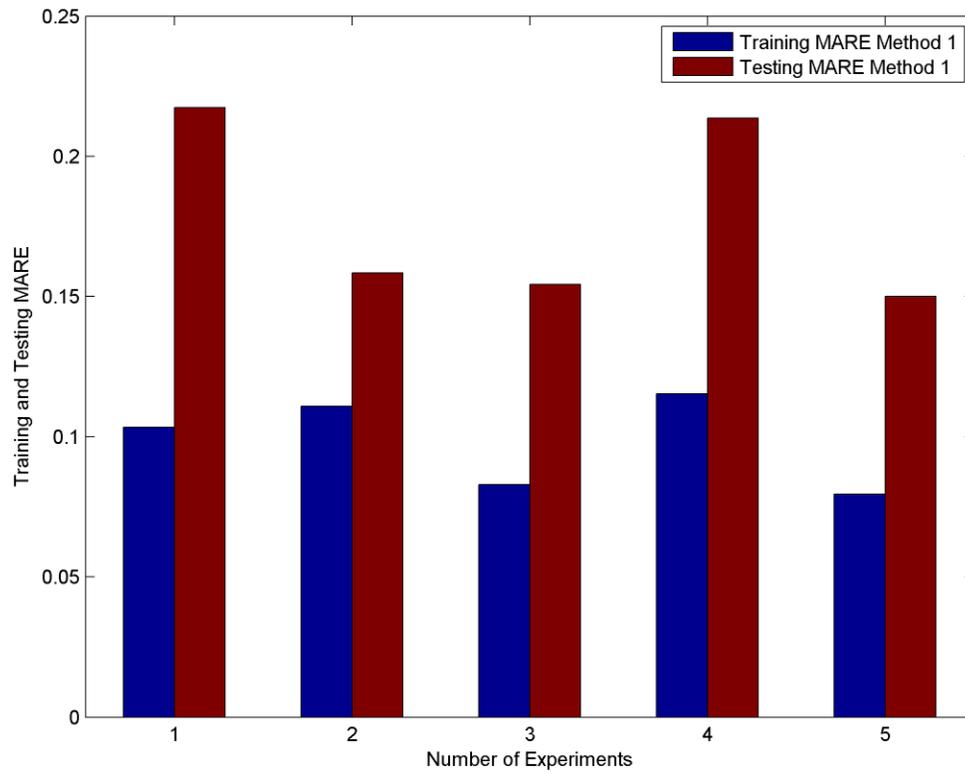


Figure 34: Comparison of training and testing errors (MARE) on Mamdani Effort Prediction System (Method 1)

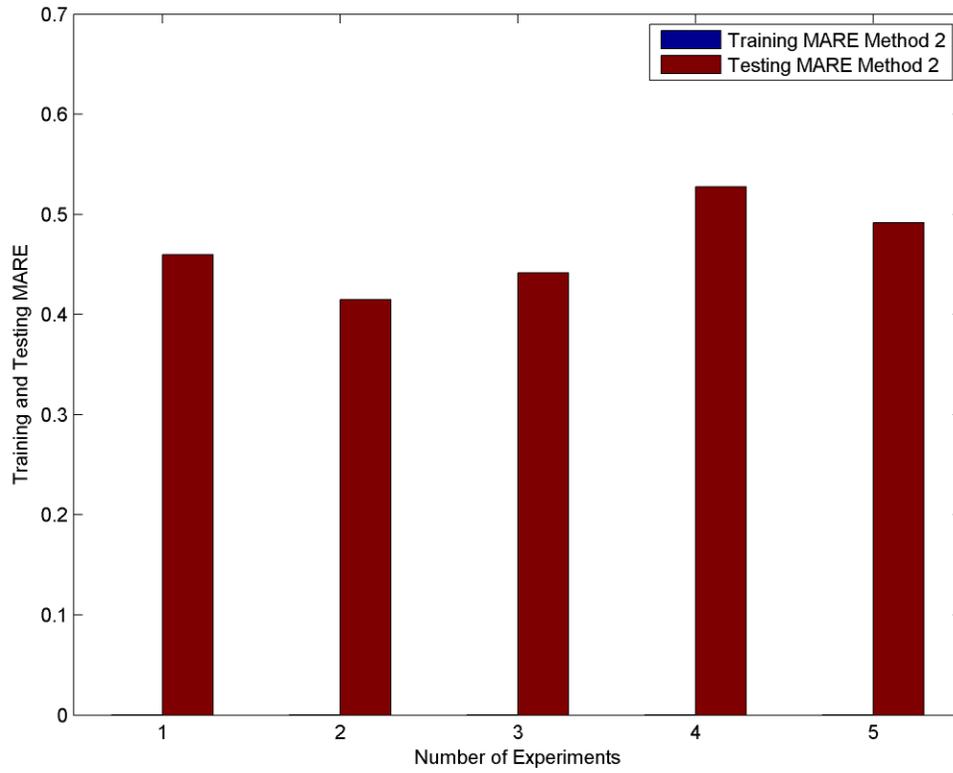


Figure 35: Comparison of training and testing errors (MARE) on Sugeno Effort Prediction System (Method 2)

6.9. Experiment 7: Impact of design parameters on prediction accuracy

This experiment deals with evaluating the performance of the prediction system in the context of the simplified framework when two different methods for initializing the design parameters are used. More details pertaining to both the methods for initializing the design parameters are present in Section 5.6 of Chapter 5. The implementation details for this experiment can be found in Section 6.6.1.

6.9.1. Results and Discussion

It is evident from Table 30 that the system with method 1 of initializing the design parameters performs relatively better than the second system. The difference in performance is not very large in terms of prediction accuracies; wherein the average testing prediction accuracy of system 1 is 78.01% and the average testing prediction accuracy of system 2 is 76.68%. Additionally, the average training error graph (Figure 36) also follows a similar observation with less difference in training error (MARE) values. Overall, it can be safely stated that the investigation pertaining to the impact of design parameters on the fuzzy systems stands valid, and the method of defining/initializing the design parameters does have an effect on the final result of the performance of the fuzzy system, which in this case is the prediction accuracy of the effort prediction system.

Table 30: Summary of Prediction Quality on the Effort Prediction Systems (Parameter Design Method 1 and Method 2) using five different datasets, showing pred(25) and MARE values on training and testing datasets

Experimental Run	Method 1				Method 2			
	Training		Testing		Training		Testing	
	pred(25)	MARE	pred(25)	MARE	pred(25)	MARE	pred(25)	MARE
1	95.7152	0.1121	73.3629	0.1732	71.4344	0.1995	80.0222	0.1767
2	92.8586	0.0998	86.6815	0.1469	81.4324	0.1480	83.3518	0.1495
3	97.1434	0.0911	83.3518	0.1490	84.2889	0.1333	93.3407	0.1374
4	95.7152	0.0963	66.7037	0.2235	84.2889	0.1461	63.3740	0.1819
5	94.2869	0.0973	80.0222	0.1638	77.1475	0.1781	63.3740	0.2132

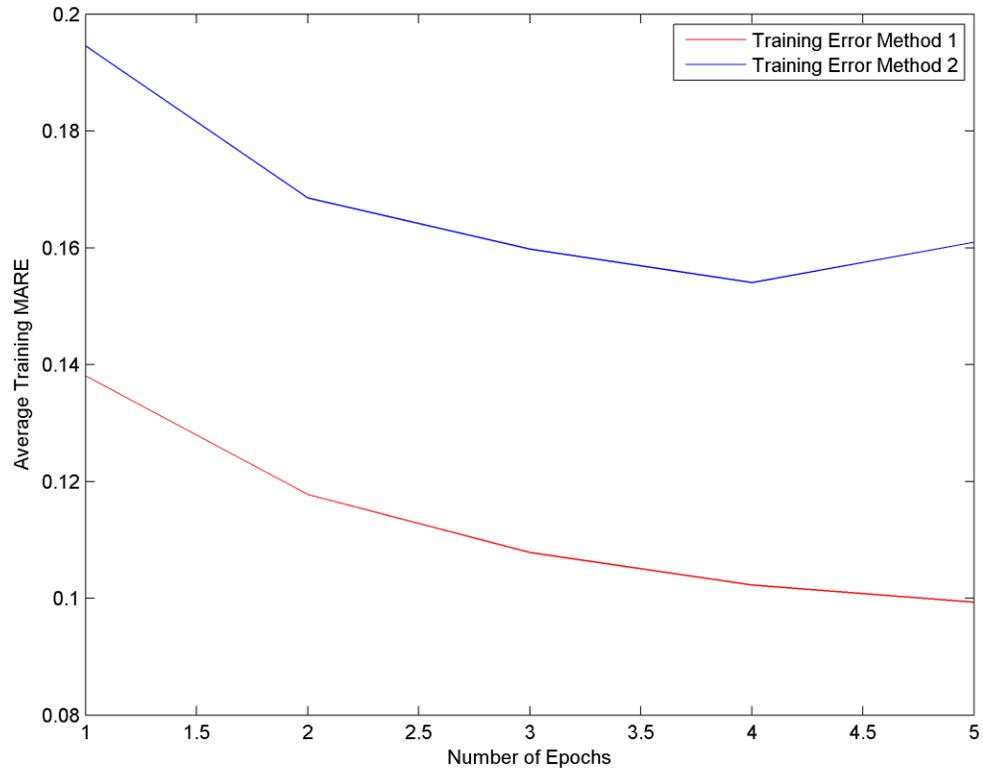


Figure 36: Average MARE graph of training the Effort Prediction Systems (Method 1 and Method 2)

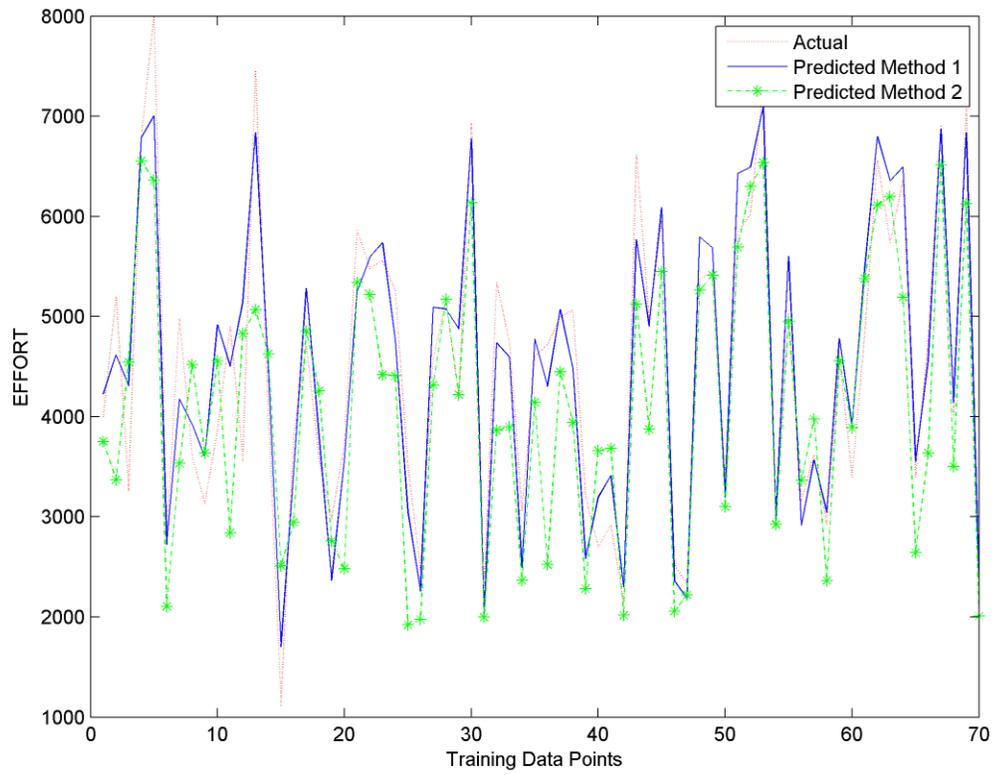


Figure 37: Prediction of effort using the Effort Prediction Systems (Method 1 and Method 2) on training datasets

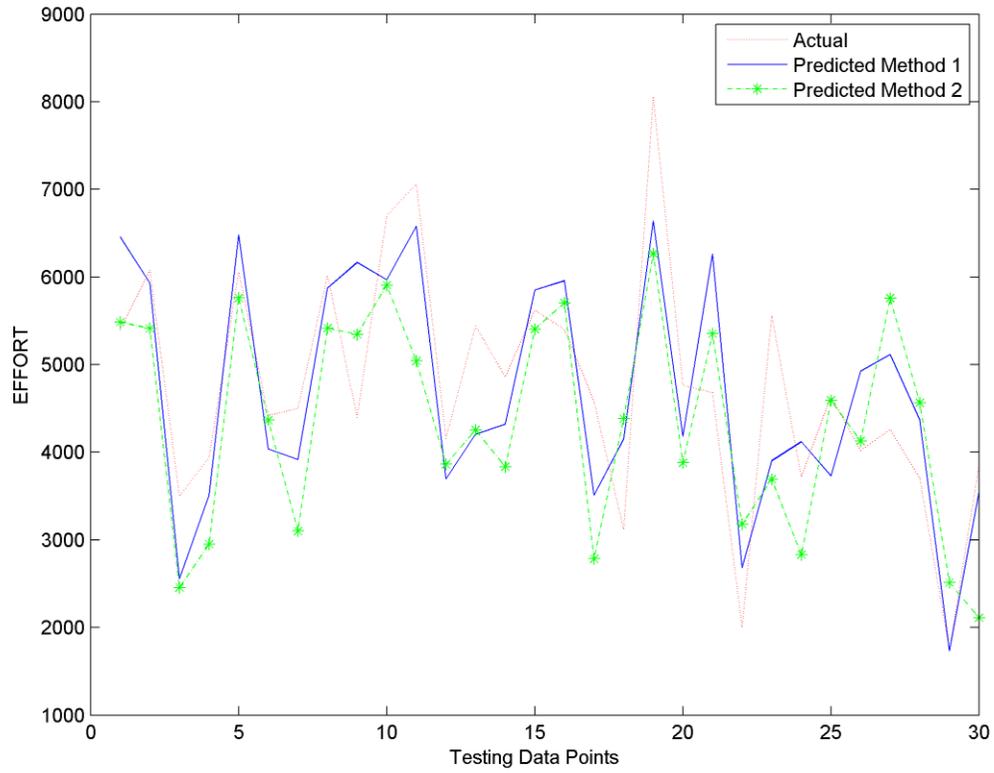


Figure 38: Prediction of effort using the Effort Prediction Systems (Method 1 and Method 2) on testing datasets

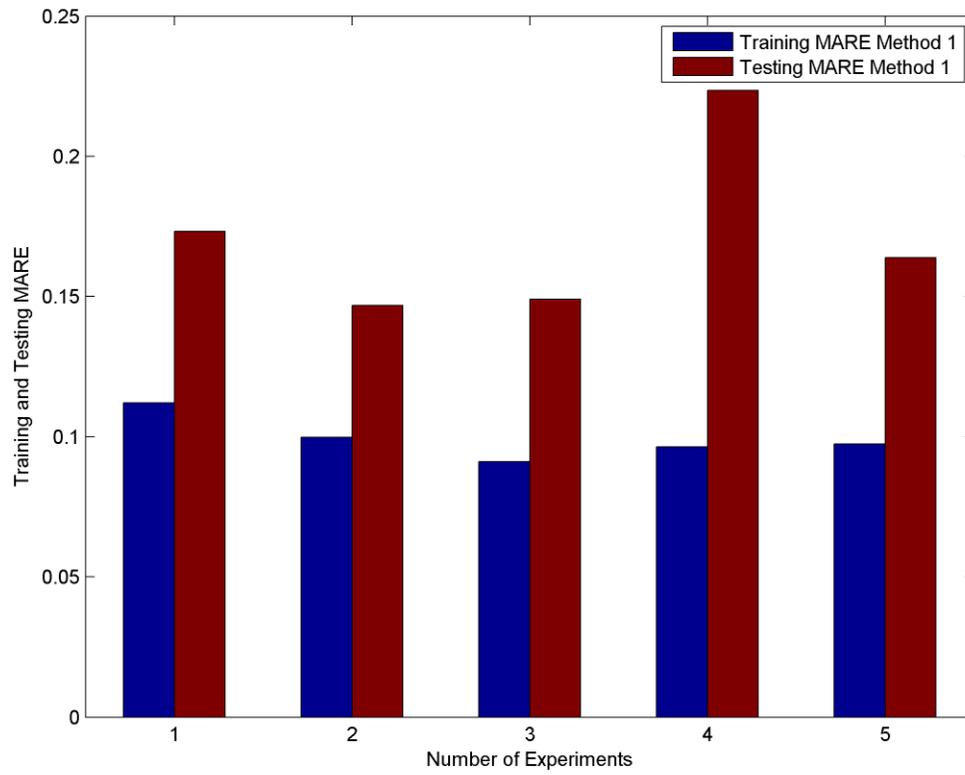


Figure 39: Comparison of training and testing errors (MARE) on the Effort Prediction System (Method 1)

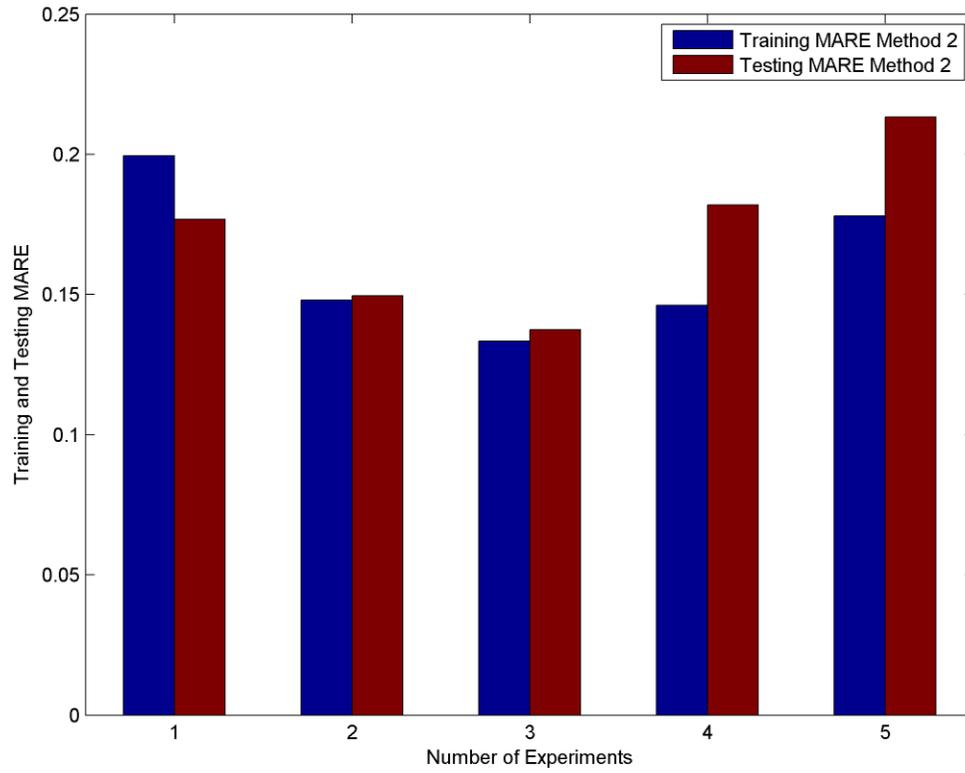


Figure 40: Comparison of training and testing errors (MARE) on the Effort Prediction System (Method 2)

6.10. Experiment 8: Evaluating the system performance using genetic learning of rule sets

This experiment deals with the evaluation of a genetic learning based prediction system where the focus is on learning the rule sets to evolve an optimal rule set which corresponds to an optimal prediction system.

6.10.1. Implementation Details

The prediction system used is essentially the same as the one obtained from the simplified framework. Genetic parameters such as size of population, number of generations, and type of selection, crossover, mutation and probabilities associated with them are initialized. The data set consists of 100 data points. There are 3 sub-experiments (different combinations of parameters of genetic algorithms). Each sub-experiment is run for 5 repetitions.

6.10.2. Results and Discussion

The minimum value of mean absolute relative error obtained in the 3 sub-experiments is 10.76%, 10.22% and 10.58% respectively. Moreover, the standard deviation is very less which shows more confidence in the error values obtained. Therefore, the use of genetic learning for evolving rule bases in the context of effort prediction systems is credible.

Table 31: Summary of information about the genetic learning process, showing minimum error (MARE) values in the sub-experiments

Serial Number	Size Of Population	Number Of Generations	Number Of Repetitions	Number Of Rules in Rule Set	Minimum Error (MARE)	Graph
1	100	100	5	100	0.1076	Figure 6-25
2	100	200	5	100	0.1022	Figure 6-26
3	100	200	5	200	0.1058	Figure 6-27

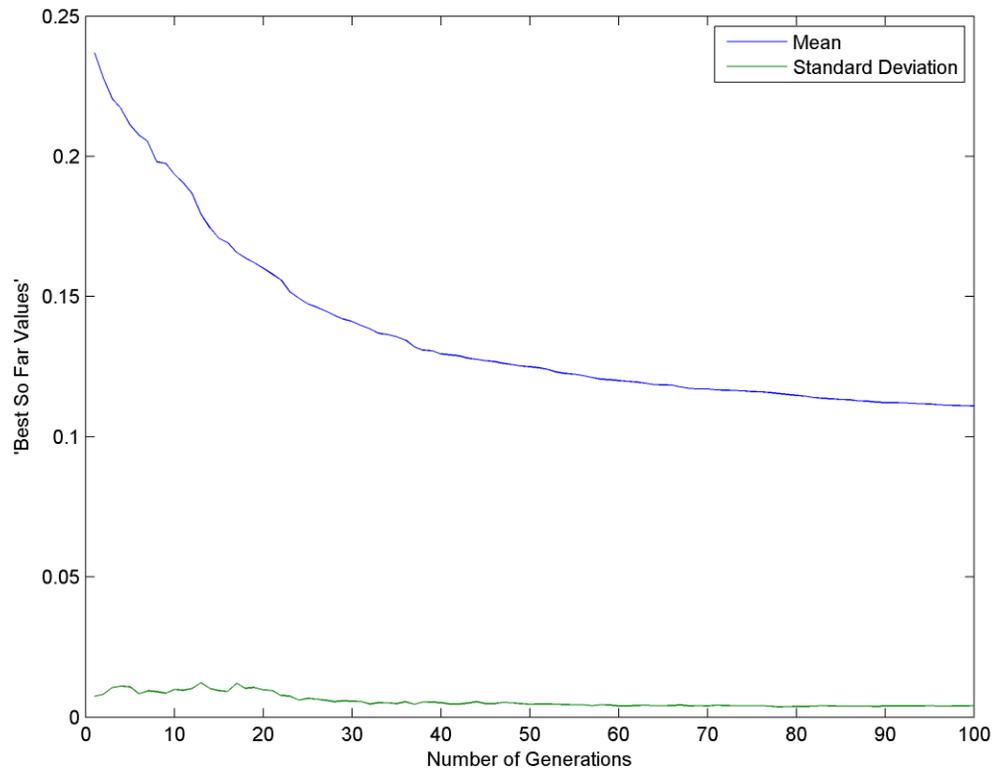


Figure 41: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 1

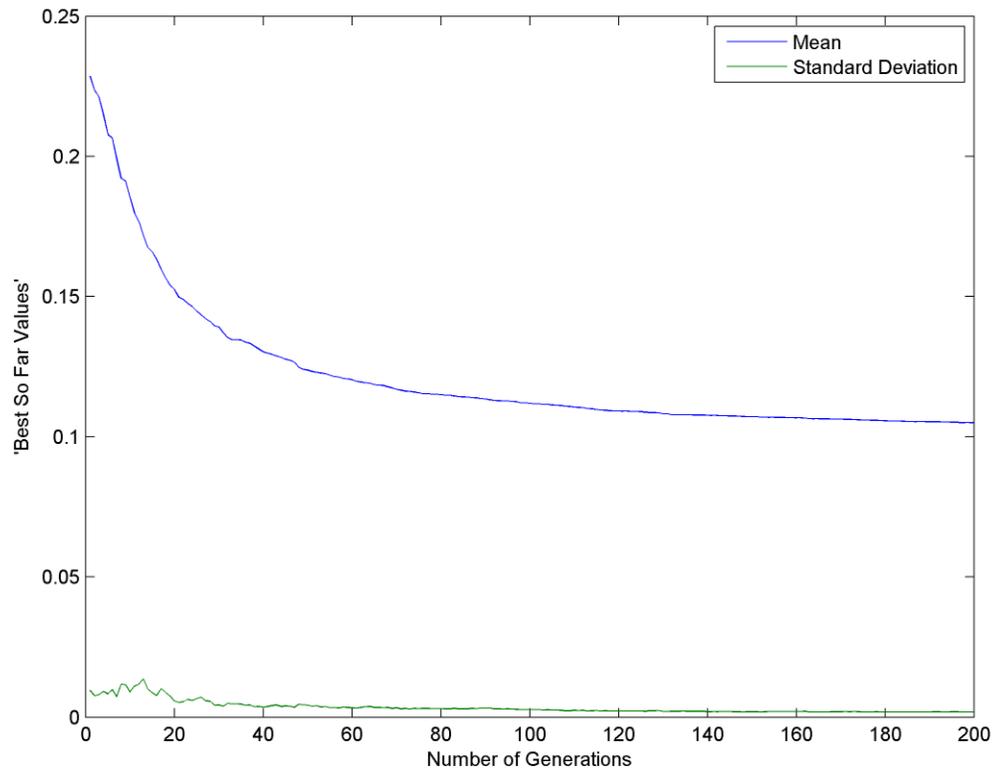


Figure 42: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 2

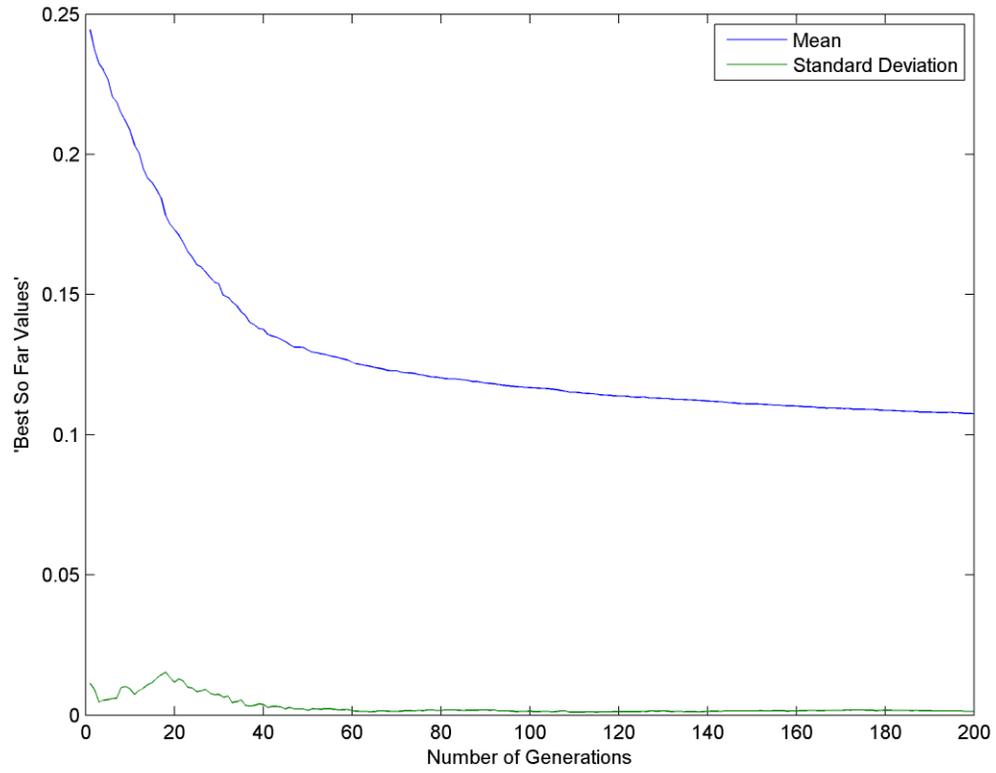


Figure 43: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 3

6.11. Experiment 9: Impact of architecture on the effort prediction framework using GeFuSys-M

This experiment deals with investigating the performance of various systems (different architectures) by measuring the MARE values of each system obtained from GeFuSys-M. More details pertaining to the design of GeFuSys-M can be found in Section 4.5. For sake of illustration, few sample architectures (Figure 47, Figure 48 and Figure 49) obtained from GeFuSys-M have been included at the end of this section.

6.11.1. Implementation Details

The system is initialized by setting the genetic parameters. The data set used consists of 100 data points. There are 3 sub-experiments (different combinations of parameters of genetic algorithms). Each sub-experiment is run for 5 repetitions.

6.11.2. Results and Discussion

The ‘Best so far’ graphs for 3 sub-experiments yield consistent results, as can be seen from the minimum value of mean absolute relative error obtained in the 3 sub-experiments which are 25.61%, 25.41% and 25.51% respectively. Also, the minimal standard deviation values obtained show more confidence in the error values obtained. Even though the MARE values are not comparable with those obtained from the prediction systems implemented using the proposed framework (see Section 6.5.2), they are promising because the genetic learning process evolves prediction systems based on

different architectures and different rule sets, whereas the proposed framework produces systems based on back propagation learning and adaptation of learning parameters. A point worth mentioning is that, future work on incorporating back propagation method of parameter learning within GeFuSys-M will probably yield more efficient use-case based effort prediction systems in terms of prediction accuracy. Moreover, the results conforms the claims that the architecture of the effort prediction systems does have an impact on its accuracy.

Table 32: Summary of information about the genetic learning process of GeFuSys-M, showing minimum error (MARE) values in the sub-experiments

Serial Number	Size Of Population	Number Of Generations	Number Of Repetitions	Maximum number of Rules Per Component	Minimum Error (MARE)	Graph
1	100	100	5	100	0.2561	Figure 6-28
2	100	50	5	200	0.2541	Figure 6-29
3	200	50	5	200	0.2551	Figure 6-30

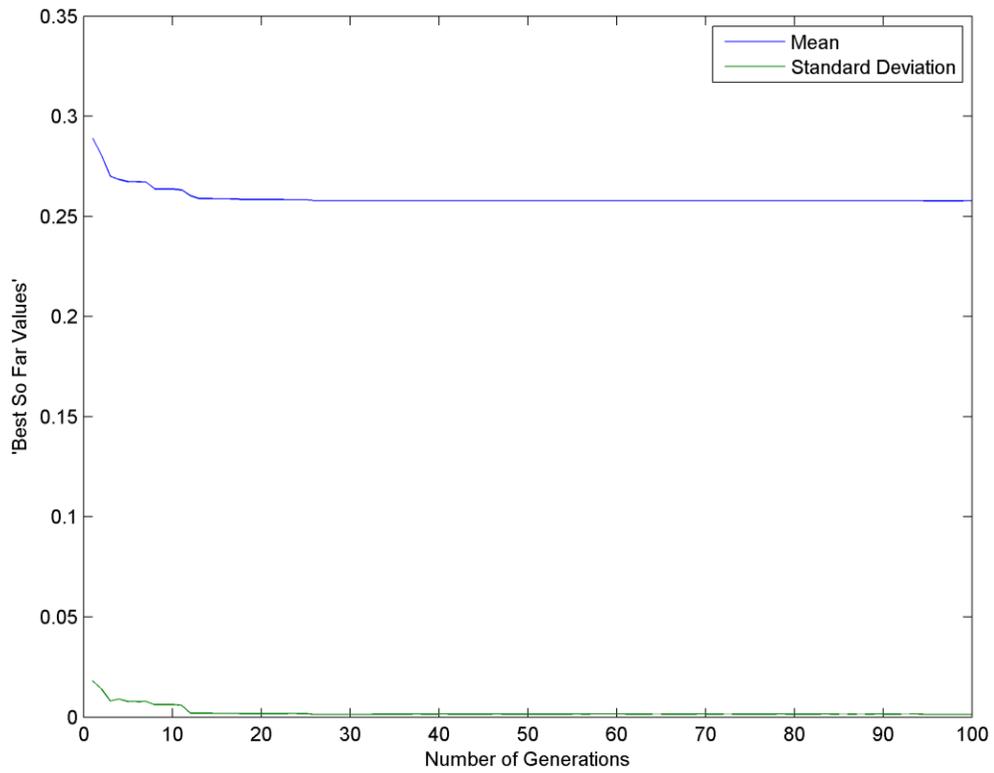


Figure 44: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 1

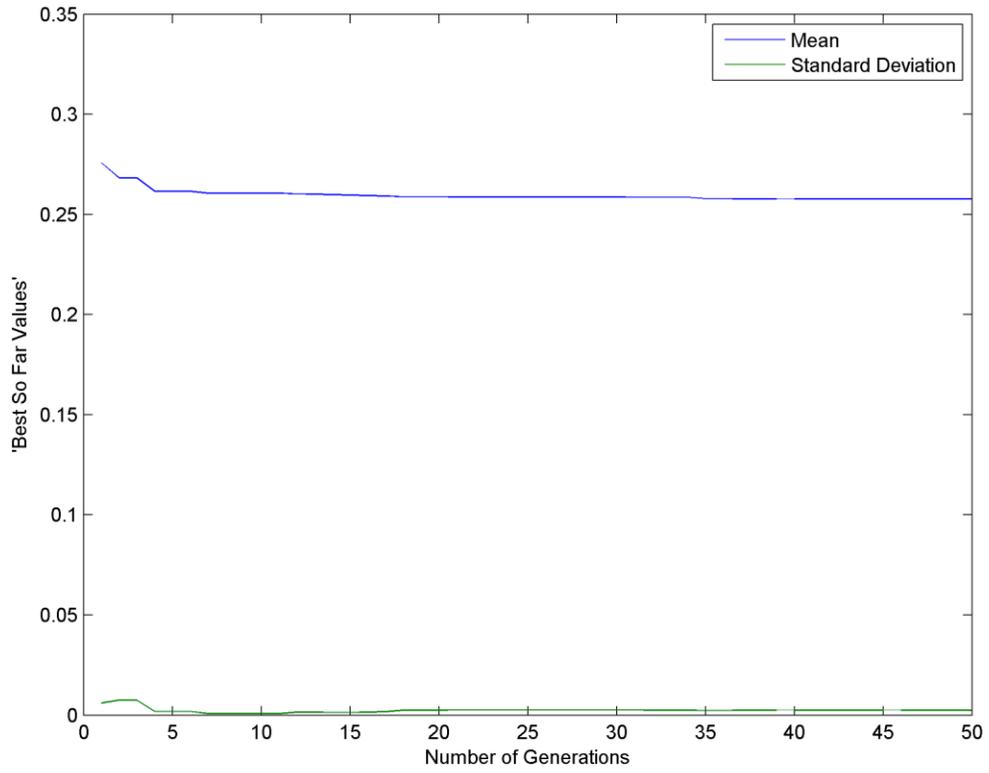


Figure 45: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 2

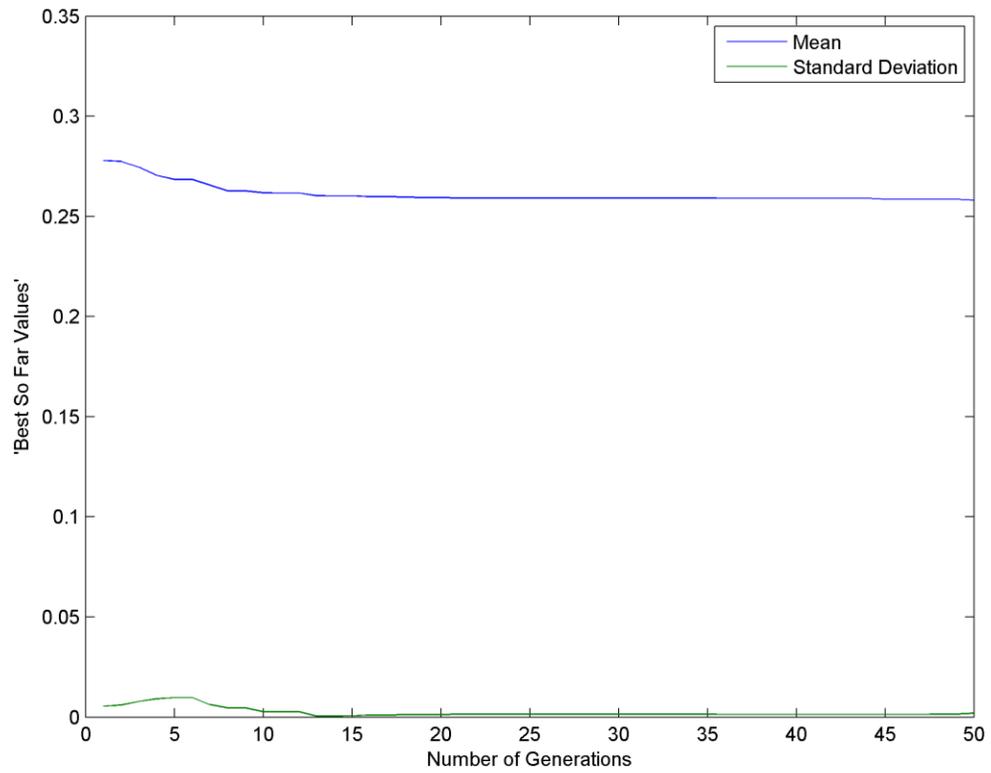


Figure 46: Best So Far Graph showing the error values (MARE) on the learning dataset for sub-experiment 3

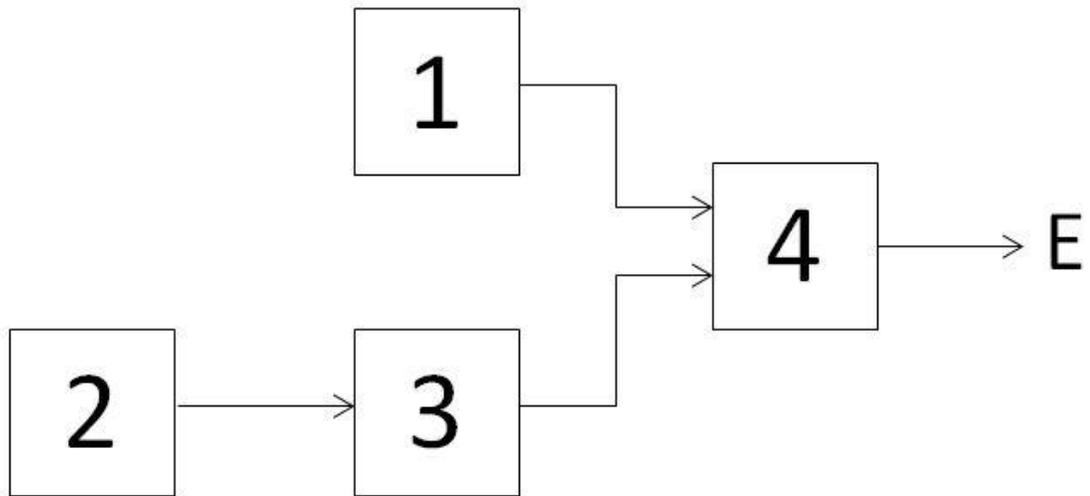


Figure 47: A four component sample architecture

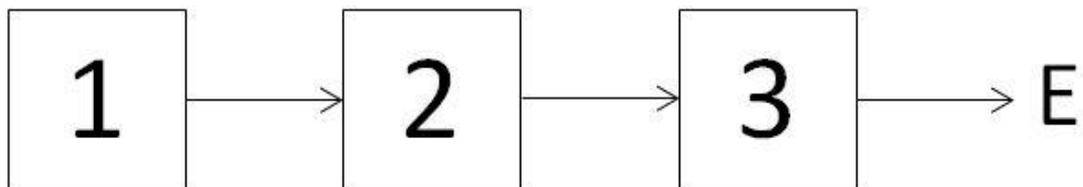


Figure 48: A three component sample architecture

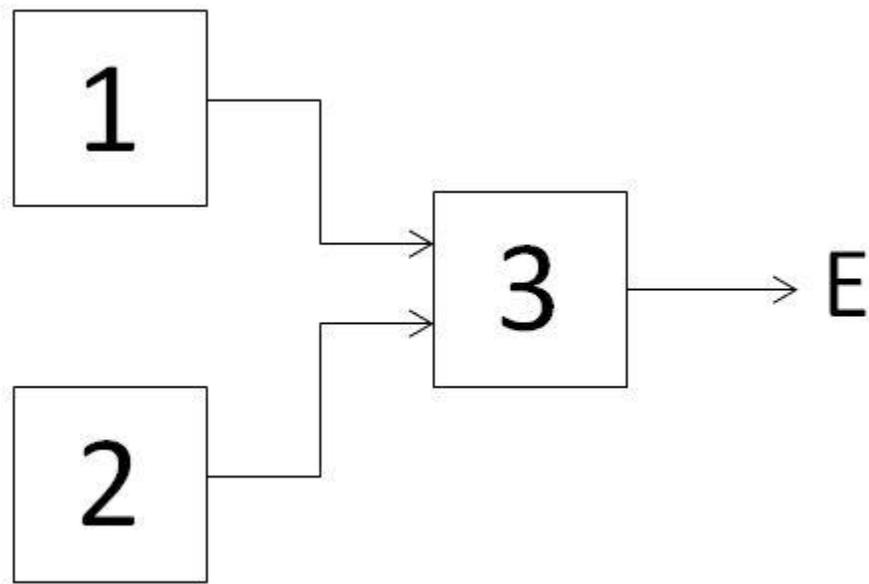


Figure 49: A three component sample architecture

CHAPTER 7

CONCLUSION

7.1. Introduction

The preceding chapters have presented the proposed frameworks and their validations in terms of experimental results. In this chapter, the conclusions viz a viz the major contributions of the investigations are presented in Section 7.2 and ideas for future work on effort prediction with use cases using fuzzy logic and genetic algorithms in Section 7.3.

7.2. Major Contributions

The research work carried out in the course of this quest to provide answers to the research questions framed in the initial phase of this thesis resulted in the following contributions.

1. Development of a use-case based effort prediction framework using fuzzy logic, capable of incorporating expert opinions and handling imprecision. Fuzzy Logic with its power of approximate reasoning provides a transparent system which allows the experts to tune the rules of the effort prediction system. Not only do the experts have freedom to tune the classification of factors, but also differentiate in the operating aspect as well by including or excluding a particular input factor from the rule.
2. Identification and reduction of the 13 technical complexity factors and 8 experience factors to 6 and 5 respectively, based on the results obtained from performing Factor Analysis.
3. Fuzzifying the existing Use Case Points method to actualize an efficient model (f-UCP) on similar lines as "*f-COCOMO*".
4. Development of a simplified use-case based effort prediction framework using fuzzy logic, capable of incorporating expert opinions and handling imprecision.
5. Investigation of the impact of using pairwise combinations for defining rules for the fuzzy logic based effort prediction system on the prediction accuracy.
6. Comparison of prediction accuracies for the effort prediction system obtained using Mamdani type fuzzy logic system and Sugeno type fuzzy logic system.
7. Investigation of the impact of design parameters on the prediction accuracy of the Effort Prediction System.

8. Development of a single-layer genetic fuzzy system for use-case based effort prediction, which gives the best rule base, in other words, the best fuzzy system in terms of prediction accuracy.
9. Development of a genetic-fuzzy tool (GeFuSys-M) for evolving multiple architectures in the context of use-case based effort prediction systems, which includes design and implementation of a new chromosome structure for GeFuSys-M. The tool GeFuSys-M has been applied in the context of use-case based effort prediction and encouraging results have been reported.

7.3. Limitations and Future Work

It is very ambitious and challenging to take into account all the aspects associated with the problem of use-case based effort prediction using machine learning, given the limited time allocated for the thesis work. As such, we have highlighted some limitations and ideas for future research in the following sequel;

7.3.1. Limitations

1. Subjective evaluation of the available use case based effort prediction metrics and models led us to find the shortcomings in the use case based effort prediction process, but not the actual head to head comparisons because no rating scheme has been used.
2. Inability of the proposed effort prediction framework to deal with uncertainty.

3. Evaluating the performance of the proposed frameworks and other investigations based on industrial datasets rather than the artificially generated datasets which have been used because of the shortage of industrial (real) dataset.

7.3.2. Future Work

1. Development of formal metrics based on our generic set of attributes to evaluate the use case based effort prediction metrics which can involve quantitative evaluations.
2. Development of use-case based effort prediction frameworks capable of dealing with both imprecision and uncertainty using type-2 fuzzy logic [66].
3. Investigating the prospect of using different membership functions other than the Gaussian MF used for the fuzzy logic based proposed frameworks. (example: Triangular MF)
4. Studying the impact of training algorithms on the performance of the proposed frameworks. Back propagation (Steepest Descent Approach) has been used now, which can be replaced by some other 'heuristic based' approach as studied by Muzaffar [67].

BIBLIOGRAPHY

- [1] Ahmad, I.: “A Probabilistic Size Proxy for Software Effort Estimation: A Framework,” Master Thesis, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, 2008.
- [2] Ahmed, M., A., Saliu, M., O. and Al-Ghamdi, J.: “Adaptive Fuzzy Logic Based Framework for Software Development Effort Prediction”, Information and Software Technology, Vol. 47, No. 1, January, 2005, pp. 31-48.
- [3] Albrecht, A., J. and Gaffney, J., E.: “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation,” IEEE Transactions on Software Engineering, vol. SE-9, Nov. 1983, pp. 639-648.
- [4] Albrecht, A., J.: “Measuring Application Development Productivity”, In Proceedings of the IBM Applications Development Symposium, SHARE-Guide, 1979, pp. 83-92.
- [5] Anda, B., Angelvik, E., and Ribu, K.: “Improving ion Practices by Applying Use Case Models Estimation”, The 4th International Conference on Product Focused Software Process Improvement (PROFES), Finland, December 9-11, pp. 383-397, LNCS 2559, Springer-Verlag, 2002.

- [6] Anda, B.: “Comparing Use Case based Estimates with Expert Estimates”, The 2002 Conference on Empirical Assessment in Software Engineering (EASE), Keele, United Kingdom, April 8-10, 2002.
- [7] Anda, B., Dreiem, H., Sjøberg, D., Jørgensen, M.: “Estimating software development effort based on use cases-experiences from industry”, Fourth International Conference on the UML, 2001, pp. 487–504.
- [8] Anda, B., Benestad, H., C. and Hove, S., E.: “A Multiple-Case Study of Effort Estimation based on Use Case Points,” pp. 1-20.
- [9] Angelis, L., Stamelos, I. and Morisio, M.: “Building a Software Cost Estimation Model Based on Categorical Data”, The 7th IEEE International Software Metric Symposium, London, England, 2001.
- [10] Boehm, B., Abts, C., and Chulani, S.: “Software Development Cost Estimation Approaches: A Survey”, University of Southern California Centre for Software Engineering, Technical Report, USC-CSE-2000-505, 2000.
- [11] Boehm, B., Clark, B., Horowitz, E., Madachy, R., Shelby, R., and Westland C.: “Cost Models for Future Software Life Cycle Processes: COCOMO 2.0”, Annals of Software Engineering, 1995.
- [12] Boehm, B. W.: “Software Engineering Economics”, Englewood Cliffs, NJ, Prentice-Hall, 1981.
- [13] Braz., M., R. and Vergilio, S., R.: “Software Effort Estimation Based on Use Cases”, COMPSAC (1), 2006, pp.221-228.

- [14] Carroll, E., R.: “Estimating Software Based on Use Case Points”, Proc. 2005 Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 05), pp. 257-265, ACM Press.
- [15] Chai, Y., Jia, L., Zhang, Z.: “Mamdani Model based Adaptive Neural Fuzzy Inference System and its Application in Traffic Level of Service Evaluation”, 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE.
- [16] Chen, Y., Boehm, B., W., Madachy, R., and Valerdi, R.: “An Empirical Study of eServices Product UML Sizing Metrics”, In Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2004), IEEE-CS Order No. P2165, Redondo Beach CA, USA, August 19-20, 2004, pp. 199-206.
- [17] Conte, S., Dunsmore, H. and Shen, V.: “Software Engineering Metrics and Models”, Benjamin-Cummings, Menlo Park, CA, 1986.
- [18] Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: “Ten years of genetic fuzzy systems: current framework and new trends”, Fuzzy Sets and Systems (2004) 141:5–31
- [19] Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L.: “Genetic Fuzzy Systems”, (Book), Advances in Fuzzy Systems – Applications and Theory, Vol.19, World Scientific, Singapore, 2001.
- [20] Costagliola, G., Ferrucci, F., Tortora, G. and Vitiello, G.: “Class Point: An Approach for the Size Estimation of Object-Oriented Systems”, IEEE

Transactions on Software Engineering, Volume 31, Issue 1, January, 2005, pp. 52-74.

- [21] Diev, S.: "Use Cases Modeling and Software Estimation: Applying Use Case Points", Software Engineering Notes, ACM, vol. 31, 2006, pp. 1-4.
- [22] Fan, W., Xiaohu, Y., Xiaochun, Z., Lu, C.: "Extended Use Case Points Method for Software Cost Estimation", Computational Intelligence and Software Engineering, 2009. CiSE 2009. Page(s): 1 - 5, Volume: Issue: 11-13 Dec. 2009.
- [23] Fenton, N., E., and Pfleeger, S., L.: "Software Metrics: A rigorous and Practical Approach", Second Edition, PWS Publishing Company, 1997.
- [24] Forbes, M.: "Use Case Survey (2009): Towards Adopting Enterprise", 2009, pp. 1-11.
- [25] Ghazi, S.A., Ahmed, M.A.: "Pair-wise test coverage using genetic algorithms," Evolutionary Computation, 2003. CEC '03. The 2003 Congress on , vol.2, no., pp. 1420- 1424 Vol.2, 8-12 Dec. 2003.
- [26] Grimstad, S., and Jørgensen, M.: "A framework for the analysis of software cost estimation accuracy", Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering - ISESE '06, 2006, p. 58.
- [27] Grimstad, S., Jorgensen, M., and Ostvold, K., M.: "Software Effort Estimation Terminology – The Tower of Babel", Information and Software Technology (2006), 302-310

- [28] Guney, K., Sarikaya, N.: “Comparison of Mamdani and Sugeno Inference System Models for Resonant Frequency Calculation of Rectangular Micro strip Antennas”, *Progress in Electromagnetics Research B*, Vol. 12, 81-104, 2009.
- [29] Hamam, A., Georganas, N.D.: “A Comparison of Mamdani and Sugeno Fuzzy Inference Systems for Evaluating the Quality of Experience of Haptic-Audio-Visual Applications”, *IEEE-HAVE’2008*, Ottawa, Canada.
- [30] Hastings, T., E. and Sajeev, A., S., M.: “A Vector-Based Approach to Software Size Measurement and Effort Estimation”, *IEEE Transactions on Software Engineering*, Volume 27, Issue 4, April 2001, pp. 337-350.
- [31] Herrera, F., “Genetic fuzzy systems: taxonomy, current research trends and prospects”, *Evolutionary Intelligence* (2008), 1: 27-46, Springer-Verlag 2008.
- [32] Herrera, F., Lozano, M., and Verdegay, J.L., “A learning process for fuzzy control rules using genetic algorithms”, *Fuzzy Sets and Systems* (1998), 100:143–151
- [33] Hodgkinson, A.C. and Garratt, P. W.: “A NeuroFuzzy cost estimator”, In *Proceedings of the 3rd International Conference on Software Engineering and Applications - SAE*, 1999, pp. 401-406.
- [34] Ibarra, G., J., Vilain, P.: “Software Estimation based on Use Case Size”, 2010, *Brazilian Symposium on Software Engineering*, IEEE.
- [35] Idri, A. and Abran, A.: “COCOMO Cost Model Using Fuzzy Logic”, *The 7th International Conference on Fuzzy Theory and Technology*, Atlantic City, New Jersey, March 2000.
- [36] Jacobson, I., Booch, G., and Rumbaugh, J.: “The Unified Software Development Process”, Addison Wesley.

- [37] Jassbi, J.J., Serra, P.J.A., Ribeiro, R.A., Donati, A.: “A Comparison of Mamdani and Sugeno Inference Systems for a Space Fault Detection System”, Contract No: 18989/05/NL/MV, European Space Agency.
- [38] Jenson, R. L. & Bartley, J. W.: “Parametric Estimation of Programming Effort: An Object-Oriented Model”, *Journal of Systems and Software*, Vol. 15, 1991, pp. 107-114.
- [39] Jorgensen, M., and Molokken, K.: “Combination of software development effort prediction intervals: Why, when and how?”, In *Proceedings of the Fourteenth IEEE Conference on Software Engineering and Knowledge Engineering (SEKE'02)*, Ischia, Italy, 2002, pp. 425-428.
- [40] Kamal, M.W., Ahmed, M.A., and El-Attar, M.: “Use Case-Based Effort Estimation Approaches : A Comparison Criteria”, *ICSECS 2011, Part III, CCIS 181*, J.M. Zain et al, eds., Springer-Verlag Berlin Heidelberg, 2011, pp. 735-754.
- [41] Kamal, M.W., and Ahmed, M.A.: “A Proposed Framework for Use Case based Effort Estimation using Fuzzy Logic: Building upon the outcomes of a Systematic Literature Review”, *International Journal of New Computer Architectures and Their Applications, IJNCAA*, Vol.1(4): 976 -999, SDIWC, 2011, ISSN: 2220-9085.
- [42] Karner, G.: “Resource Estimation for Objectory Projects”, *Objectory Systems*, 1993.
- [43] Kathleen Peters: “Software Project Estimation”, *Software Productivity Center Inc.*, 1999.

- [44] Kauffman, R. and Kumar, R.: “Modeling Estimation Expertise in Object Based CASE Environments”, Stern School of Business Report, New York University, January, 1993.
- [45] Kirsopp, C., Shepperd, M., J. and Hart, J.: “Search Heuristics, Case-Based Reasoning and Software Project Effort Prediction”, Genetic and Evolutionary Computing Conference (GECCO 2002), New York, AAAI, 2002.
- [46] Kirsten Ribu: “Estimating Object-Oriented Software Projects with Use Cases”, Master of Science Thesis, Department of Informatics, University of Oslo, Oslo, Norway, November 7, 2001.
- [47] Kitchenham, B., Pfleeger, S., L. and Fenton, N.: “Towards a Framework for Software Measurement Validation”, IEEE Transaction on Software Engineering, Vol. 21, No. 12, 1995, pp. 929-944.
- [48] Klaib, M.F.J., Muthuraman, S., and Noraziah, A.: “A Parallel Tree Based Strategy for T-Way Combinatorial Interaction Testing”, ICSECS 2011, Part III, CCIS 181, J.M. Zain et al, eds., Springer-Verlag Berlin Heidelberg, 2011, pp. 91 – 98.
- [49] Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S. and Maegawa, Y.: “Estimating Effort by Use Case Points: Method, Tool and Case Study”, In Proceedings of the 10th International Symposium on Software Metrics, (METRICS'04), Volume 00, September, 2004.
- [50] Lai, R. and Huang, S., J.: “A Model for Estimating the Size of a Formal Communication Protocol Specification and Its Implementation,” vol. 29, 2003, pp. 46-62.

- [51] Laranjeira, L.: “Software Size Estimation of Object-Oriented Systems”, IEEE Transactions on Software Engineering, Vol. 16, No. 5, May, 1990, pp: 510 – 522.
- [52] Larsen, P., M.: “Industrial Applications of Fuzzy Logic Control”, International Journal of Man-Machine Studies, Vol. 12, No. 1, 1980, pp. 3-10.
- [53] Leung, H., Fan, Z.: “Software Cost Estimation,” Handbook of Software Engineering and Knowledge Engineering, Vol. 2, January 2002, pp. 1-14.
- [54] Liang, T. and Noore, A.: “Multistage Software Estimation”, Proceedings of the 35th Southeastern Symposium on System Theory, 16-18 March, 2003, pp. 232 – 236.
- [55] Lockheed, M.: “Advanced Concept Center training materials”, Object Oriented Size and Cost Estimation, 1994.
- [56] MacDonell, S., G., and Gray A., R.: “A Comparison of Modeling Techniques for Software Development Effort Prediction”, In Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems, Denedin, Newzealand, Springer-Verlag (1997), 869-872.
- [57] Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., and Webster, S.: “An Investigation of Machine Learning Based Prediction Systems Background to Research,” 1999, pp. 1-21.
- [58] Mamdani, E., H.: “Applications of Fuzzy Algorithms for Simple Dynamic Plant”, Proc. IEEE 121, 12 (1974).
- [59] Meitzler, T.J., Sohn, E.: “A Comparison of Mamdani and Sugeno Methods for Modeling Visual Perception Lab Data”, IEEE-NAFIPS, North American Fuzzy Information Processing Society, 2005.

- [60] Mendel, J., M.: “Uncertain Rule-Based Fuzzy Logic Systems”, Prentice-Hall, Upper Saddle River, NJ 07458, 2001.
- [61] Mendel, J., M. and Liang, Q.: “Pictorial Comparison of Type-1 and Type-2 Fuzzy Logic Systems”, In Proceedings of IASTED International Conference on Intelligent Systems & Control, Santa Barbara, CA, Oct., 1999.
- [62] Milicic, D., and Wohlin, C.: “Distribution Patterns of Effort Estimations”, Proceedings of the 30th EUROMICRO Conference (EUROMICRO’04).
- [63] Minkiewicz, A. F.: “Objective Measures”, Software Development, June 1997, pp: 43-47.
- [64] Mohagheghi, P., Anda, B., Conradi, and R.: “Effort Estimation of use cases for incremental large-scale software development”, in: Proceedings of the 27th International Conference on Software Engineering, 2005, pp. 303–311.
- [65] Musilek, P., Pedryez, W., Succi, G. and Reformat, M.: “Software Cost Estimation with Fuzzy Models”, Applied Computing Review, Vol. 8, No. 2, pp. 24-29, 2000.
- [66] Muzaffar, Z., and Ahmed, M., A.: “Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems”, Information and Software Technology, vol. 52, Jan. 2010, pp. 92-109.
- [67] Muzaffar, Z.: “Adaptive Fuzzy Logic based Framework for handling Imprecision and Uncertainty in Software Development”, Master Thesis, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, 2006.
- [68] Neill, C., J. and Laplante, P., A.: “Requirements Engineering: The State of the Practice,” IEEE Software, Vol. 20, No. 6, November-December 2003. pp. 40-45.

- [69] Nunes, N., J., Constantine, L., and Kazman, R.: “iUCP: Estimating Interactive-Software Project Size with Enhanced Use-Case Points”, IEEE Software, 2011.
- [70] Ochodek, M., Nawrocki, J., and Kwarciak, K.: “Simplifying effort estimation based on Use Case Points”, Information and Software Technology, vol. 53, Mar. 2011, pp. 200-213.
- [71] Putnam, L. H.: “A General Empirical Solution to the Macro Software Sizing and Estimating Problem”, IEEE Transactions on Software Engineering, Vol. 4., No. 4, pp. 345-361, 1978.
- [72] Rahman, Q. A.: “Dealing with Imprecision and Uncertainty while Developing Software Quality Models”, Master’s Thesis, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, 2005.
- [73] Robiolo, G., Badano, C., Orosco, R.: “Transactions and Paths: two use case based metrics which improve the early effort estimation”, Third International Symposium on Empirical Software Engineering and Measurement, 2009, pp.422–425.
- [74] Robiolo, G., and Orosco, R.: “Employing use cases to early estimate effort with simpler metrics,” Innovations in Systems and Software Engineering, vol. 4, Feb. 2008, pp. 31-43.
- [75] Royce, W.: “Software Project Management: A Unified Framework”, Addison Wesley, 1998.

- [76] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenzen, W.: “Object-Oriented Modeling and Design”, Prentice-Hall, Rational Software Corporation, 1997b, Unified Modeling Language, Version 1.1, 1991.
- [77] Russell, S. and Norvig, P.: “Artificial Intelligence: A Modern Approach”, 1st Edition, Prentice-Hall Inc., 1995.
- [78] Ryder, J.: “Fuzzy modeling of software effort prediction”, In IEEE Information Technology Conference, 1998, pages 53–56.
- [79] Saliu, M., Ahmed, M. and AlGhamdi, J.: “Towards Adaptive Soft Computing Based Software Effort Prediction”, In IEEE Meeting of the Fuzzy Information Processing NAFIPS, Volume 1, 27-30 June, 2004, pp. 16-21.
- [80] Saliu, M.O., and Ahmed, M.A., “Soft Computing Based Effort Prediction Systems – A Survey”, A Chapter in E. Damiani, L. C. Jain, and M. Madravio (EDs), Soft Computing in Software Engineering, Springer-Verlag Publisher, July 2004, ISBN 3-540-22030-5.
- [81] Saliu, M.: “Adaptive Fuzzy Logic Based Framework for Software Development Effort Prediction”, Master Thesis, Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, 2003.
- [82] Schneider, G., Winters, J.P.: “Applying Use Cases – A practical guide”, 2nd ed. Addison-Wesley, 2001.
- [83] Schofield C.: “Non Algorithmic Effort Estimation Techniques”. Technical Report, Department of Computing, Bournemouth University, England, TR98-01, March 1998.

- [84] Sindre, G., Opdahl, A., L.: “Eliciting Security Requirements with misuse cases”, *Requirements Engineering* (2005), 10: 34-44, Springer-Verlag, London Limited.
- [85] Smith, J.: “The Estimation of Effort Based on Use Case”, IBM Rational Software, White Paper, 1999.
- [86] Somerville, I.: “Software Engineering”, sixth edition, 2001, Pearson Education Limited.
- [87] “Software estimation, benchmarking, productivity, risk analysis, and cost information for software developers and business”, <http://www.isbsg.org/>
- [88] Srinivasan, K., and Fisher, D.: “Machine Learning Approaches to Estimating Software Development Effort”, *IEEE Transactions on Software Engineering*, Vol. 21, No. 2, 1995.
- [89] Strike, K., El-Emam, K. and Madhavji M.: “Software Cost Estimation with Incomplete Data”, *IEEE Transactions on Software Engineering*, Vol. 27, No. 10, Oct. 2001.
- [90] Stutzke, R., D.: “Software Estimation Technology: A Survey”, *IEEE Software Engineering Project Management*, 1997.
- [91] Symons, C., R.: “Software Sizing and Estimating: Mk II FPA”, Chichester, England, John Wiley, 1991.
- [92] Takagi, T., Sugeno, M., “Fuzzy identification of systems and its application to modeling and control”, *IEEE 1985, Trans Syst Man Cybern* 15(1):116–132
- [93] Teologlou, G.: “Measuring Object Oriented Software with Predictive Object Points”, In 10th Conference on European Software Control and Metrics, May 1999.

- [94] Walston, C., E., and Felix C., P.: "A Method of Programming Measurement and Estimation", IBM Systems Journal, vol. 16, no. 1, 1977, pp. 54-73.
- [95] Wang, L.: "Adaptive Fuzzy System and Control: Design and Stability Analysis", Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1994.
- [96] Wu, L.: "The comparison of the Software Cost Estimating Methods", University of Calgary, Calgary, Canada, 1997, http://sern.ucalgary.ca/courses/seng/621/W97/wul/seng621_11.html.
- [97] Yahya, M., Ahmad, R., and Lee, S.: "Impact of CMMI Based Software Process Maturity on COCOMO II's Effort Estimation", The International Arab Journal of Information Technology, Vol. 7, No. 2, April 2010
- [98] Zadeh, L., A.: "The concept of a Linguistic Variable and Its Application to Approximate Reasoning-1", Information Sciences, Volume. 8, 1975, pp. 199-249.
- [99] Zadeh, L., A.: "Fuzzy Sets", Information and Control 8, 1965, pp. 338-353.
- [100] Zimmermann, H.: "Fuzzy Set Theory and Its Applications", Kluwer Academic Publishers, Third Edition, 1996.
- [101] Zonglian, F. and Xihui, L.: "f-COCOMO: Fuzzy Constructive Cost Model in Software Engineering", Proceedings of IEEE International Conference on Fuzzy Systems (IEEE 1992), pp. 331-337.

CURRICULUM VITAE

- Name: Mohammed Wajahat Kamal
- Nationality: Indian
- Languages: English, Hindi, Urdu, Arabic
- Education
 - Master of Science in Computer Science, King Fahd University of Petroleum and Minerals, May 2012, KSA
 - Bachelor of Engineering in Information Technology, Osmania University, May 2009, India
- Work Experience
 - Dell International Services
 - MS Group of Institutions
- Research Interests
 - Software Effort Prediction
 - Machine Learning
 - Management Information Systems
- Contact Details
 - Permanent Address: 3-8-72, Chintalkunta Checkpost, Hyderabad, India
 - E-mail Address: wajju.999@gmail.com