HANDLING IMPRECISION AND UNCERTAINTY IN SOFTWARE QUALITY MODELS

BY

QUAZI ABIDUR RAHMAN

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

June 2005

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **QUAZI ABIDUR RAHMAN** 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. Jarallah Al-Ghamdi (Chairman)

Dr. Moataz Ahmed (Co-Chairman)

Dr. Krishna Rao (Member)

Dr. Muhammad Shafique (Member)

Dr. Mohammad Al-Shayeb (Member)

Dr. Kanaan Faisal
Department Chairman

Dr. Mohammad Abdulaziz Al-Ohali
Dean of Graduate Studies

Date

28-6-2005

DEDICATION

This thesis is dedicated to

My Parents

Quazi Wahidur Rahman & Mosammat Noor Banu

Acknowledgments

All praise and thanks are due to Allah, the Greatest, the most Merciful, the Creator and Lord of everything. I am thankful to King Fahd University of Petroleum and Minerals for the support in my research.

I am grateful to Dr. Jarallah Al-Ghamdi, my thesis advisor for all his support throughout my thesis. My deepest appreciation goes to him for his constructive criticism and continuous encouragement. Besides his busy schedule, he always showed his interest in my thesis work. Especially I would like to thank him for motivating me to actively participate in Software Metrics Research Group (SMRG) which was a continuous source of research discussion and valuable feedbacks. I am really indebted to my co-advisor Dr. Moataz Ahmed for all his scholarly suggestions and critical review of each stage in my thesis progress. He showed real interest and motivated me from the very beginning of this work. Hours of discussion with him were the best learning experience in my life. My thanks also go to my committee members – Dr. Krishna Rao, Dr. Muhammad Shafique and Dr. Mohammad Al-Shayeb. All of them provided me with valuable feedbacks and guidelines to improve the quality of my work.

I appreciate all the members of SMRG specially Abubakar, Rufai and Sohel who were my good friends and guide in all the discussions of my thesis.

Finally, I believe all my success came due to the prayers of my parents and my wife. They were my continuous source of inspiration in all the hardship and struggle. They made me to realize that having a nice family back home is a real blessing from Allah.

Table of Contents

Table of Contents vi
List of Figures viii
List of Tables x
Thesis Abstract xi
ملخص الرسالة xii
Chapter 1 Introduction 1
1.1 Quality Models 3
1.2 Imprecision and Uncertainty in Quality Models 7
1.3 Problem Statement 12
1.4 Main Contributions 13
1.5 Organization of the Thesis 14
Chapter 2 Literature Survey 15
2.1 Algorithmic Approach 15
2.2 Non-Algorithmic Approach 17
Chapter 3 Imprecision, Uncertainty and Fuzzy Logic Systems 19
3.1 Imprecision and Type-1 Fuzzy Logic Systems 21
3.1.1 Fuzzy Sets and Linguistic Variables 21
3.1.2 Fuzzy Logic Systems 23
3.1.3 Adaptive Fuzzy Logic 25
3.2 Uncertainty in Fuzzy Logic Systems 26
3.3 Uncertainty and Type-2 Fuzzy Sets 28
3.4 Uncertainty and Type-2 Fuzzy Logic System 33
3.4.1 Probabilistic Models vs. Type-2 Fuzzy Logic Systems 34
3.5 Fuzzification in Type-2 Fuzzy Logic System 35
Chapter 4 Framework for Building Software Quality Models 40
4.1 Type-2 Fuzzy Logic and Four types of Uncertainties 41
4.1.1 Uncertainty about the meanings of the words that are used in a rule 41

4.1.2 Uncertainty about the consequent that is used in a rule 43
4.1.3 Uncertainty about the measurements that activate the FLS 45
4.1.4 Uncertainty about the data used to tune the parameters of a FLS 46
4.2 Type-2 fuzzy Logic based Framework to Build Software Quality Model 47
4.2.1 Initializing the Framework 49
4.2.2 Training the Framework 53
4.2.3 Using the Framework 55
4.3 Experimental Design and Validation 55
4.4 Validation of the Training Algorithm 56
Chapter 5 Experimental Results 63
5.1 Experiments with NASA Dataset 63
5.1.1 Results from OO Dataset 66
5.1.2 Results from Procedural Dataset 74
5.2 Experiments with Apache Dataset 80
Chapter 6 Conclusion 82
6.1 Contributions 82
6.2 Limitations 83
6.3 Future Work 84
References 86
Nomenclature 89

List of Figures

Figure 1: Different sources of knowledge in Software Quality Models 9
Figure 2: Membership functions for LCOM 23
Figure 3: Fuzzy logic system with fuzzifier and defuzzifier 24
Figure 4: A Type-1 triangular membership function 29
Figure 5: Blurred triangular membership function 30
Figure 6: Type-1 fuzzy sets 31
Figure 7: FOU s for the Figure 6 membership functions. 31
Figure 8: FOU for Gaussian primary membership function with uncertain mean ... 32
Figure 9: FOU for Gaussian primary membership function with uncertain standard deviation 33
Figure 10: Type -2 FLS 35
Figure 11: Different types of FLS – (a) singleton type-1, (b) non-singleton type-1, (c) singleton type-2, (d) non-singleton type-2 with type-1 inputs, (e) non-singleton type-2 with type-2 inputs 38
Figure 12: Different fuzzy sets which need to be combined [29] 42
Figure 13: Union operation to form a Type-2 fuzzy set [29] 42
Figure 14: Type-2 FLS based framework to build software quality models 48
Figure 15: Training with means (Type-2) and testing with CAMC (step size = 0.01) 58
Figure 16: Error Function 59
Figure 17: Training with means (Type-2) and testing with CAMC (step size = 0.1) 59
Figure 18: Training with means (Type-2) and testing with LCC (step size=0.1) 60
Figure 19: Training with means (Type-2) and testing with TCC (step size=0.1) 61
Figure 20: Training with means (Type-2) and testing with CCM (step size=0.1) ... 62
Figure 21: Average Testing MSE of OO project for 15 experiments with step size 0.01 68

Figure 22: Standard Deviation of Testing MSE of OO project for 15 experiments with step size 0.01 69
Figure 23: Average Testing MSE of OO project for 15 experiments with step size 0.2 70
Figure 24: Standard Deviation of Testing MSE of OO project for 15 experiments with step size 0.2 71
Figure 25: Comparison of Regression based models with FLS based models for OO Testing Data 73
Figure 26: Average Training MSE of procedural project for 15 experiments with step size 0.01 76
Figure 27: Standard Deviation of Training MSE of procedural project for 15 experiments with step size 0.01 77
Figure 28: Comparison of Regression based models with FLS based models for Procedural Testing Data 79
Figure 29: Comparison with the results from the paper [8] 81

List of Tables

Table 1: Different Fuzzy Logic Systems to handle different types of noise 39
Table 2: Uncertainties in FLS and Software Quality Model 41
Table 3: Survey of experts' opinion on linguistic relationship 45
Table 4: Training Dataset for one Attribute 51

Thesis Abstract

NAME: Quazi Abidur Rahman

TITLE: Handling Imprecision and Uncertainty in Software Quality Models

MAJOR FIELD: COMPUTER SCIENCE

DATE OF DEGREE: JUNE 2005

Imprecision and Uncertainty are two important issues that surround the different sources of knowledge to build software quality model. These issues have been discussed in details in this thesis. Four types of uncertainty have been identified surrounding four sources of knowledge. None of the existing approaches can handle imprecision and these four types of uncertainty together. This thesis developed a framework that is based on Type-2 fuzzy logic system to handle imprecision and uncertainty in software quality models. Experiments have been carried out to validate the framework. Software fault prediction model has been built as an instance of the framework using historical dataset. Experimental results have shown the superiority of Type-2 fuzzy logic based framework over regression based approach.

ملخص الرسالة

الاسم: قاضي عابد الرحمن

عنوان الرسالة: معالجة عدم الدقة و الارتياب في نماذج جودة البرامج

التخصص: علوم الحاسب الآلي

تاريخ التخرج: ربيع الثاني 1426

تعتبر عدم الدقة و الارتياب قضيتان رئيسيتان تحيطان بالمصادر المختلفة للمعرفة اللازمة لبناء نماذج جودة البرامج. لذلك تم إختيار معالجتهما موضوعاً لهذه الرسالة. وقد بدأت الرسالة بتحديد أربعة أنواع من الارتياب تحيط بمصادر المعرفة. والجدير بالذكر أنه حتى كتابة هذه الرسالة لاتوجد طريقة تستطيع التعامل مع عدم الدقة و تلك الأنواع الأربعة من الارتياب معا. ولقد تم في البحث المؤدي إلى هذه الرسالة تطوير هيكل مبني على النوع الثاني من نظام المنطق الغائم للتعامل مع عدم الدقة و الارتياب في نماذج جودة البرامج. وقد أجريت بعض التجارب للتحقق من نجاح الهيكل المطور. وكذلك تم في هذه الرسالة بناء نموذج للتنبؤ بأخطاء البرامج كتطبيق على هذا الهيكل باستخدام بيانات سابقة. وقد أثبتت نتائج التجارب تفوق الهيكل المطور والذي بُني على النوع الثاني من نظام المنطق الغائم على الطرق المبنية على تمثيل البيانات بالدوال.

Chapter 1

Introduction

Achieving a high level of software quality is the objective of most developers. It is no longer accepted to deliver poor quality products and then repair problems and deficiencies after they have been delivered to the customer. Accordingly, quality planning begins at an early stage in the software development process. A quality plan sets out the desired product qualities (a.k.a., *external quality attributes*). It should also define how they are to be assessed. It therefore defines what "high quality" software actually means for the product being developed. Software quality models provide such definitions along with means for prediction and assessment. Without a quality model, different engineers may work on in an opposing way so that different external quality attributes are optimized. There is a wide range of potential software external quality attributes, e.g., Safety, Security, Reliability, Understandability, Adaptability, Reusability, and Robustness [35]. In general, it is not possible for any system to be optimized for all potential attributes. A corresponding quality model is meant to define the critical and most significant quality attributes and show how they can be achieved. It may be that *reliability* is paramount and other attributes have to be scarified to achieve this.

It is often impossible to measure software external quality attributes directly. External attributes such as maintainability, understandability, and complexity are affected by many different factors and there are no straightforward metrics for them. Rather, we have to measure some *internal* attribute of software (such as its *size*) and assume that a relation exists between what we can measure and what we want to know. Ideally, there should be a clear and validated relationship between the internal and the external software attributes. External attributes are visible to the stakeholders (e.g., customers, users, and development project managers) of the product; internal attributes concern the developer of the product. In general, stakeholders (other than the developers) of software products care only about external quality attributes, but it is the internal attributes—which deal largely with the structure of the software—that help developers achieve the external qualities. For example, the internal quality of verifiability is necessary for achieving the external quality of reliability. In many cases, however, the qualities are related closely, and the distinction between internal and external is not sharp.

A software quality model is meant to define the different external attributes that are of interest to the customer along with their level of contributions; and the functional relationship between the external attributes that are to be predicted and assessed and the internal attributes which we can measure.

A major challenge that faces quality planners would be in building that part of the quality model that defines the contribution of the different internal attributes to the achievement of an external quality attribute. In other words, the challenge lies in building models which would predict/assess some external attribute based on the measurements of different internal attributes. The challenge is even amplified when trying to consider the imprecision and uncertainty issues surrounding the internal attributes measurements and the functional relationships between the attributes within the quality model. Unfortunately, well-known techniques such as regression analysis, artificial neural network, and Bayesian belief network etc cannot assist in dealing with these two issues.

1.1 Quality Models

The term *Quality Model* is defined in [15] as "the set of characteristics and relationship between them, which provides the basis for specifying quality requirements and evaluation quality". This set of characteristics has been defined in different ways by different quality model developers. Basically as we discussed in the previous section, quality models try to explore the relationship between internal and external attributes of software product, process or resources. Two of the earliest quality models are due to McCall [26] and Boehm [4] *et al.* In these models, the characteristics are quality

factors and quality criteria. The quality factors are high level external attributes which are the key attributes of the quality from the user's perspective. As these high-level quality factors are difficult to measure or predict, those are decomposed to measurable quality criteria. Quality models can be divided in two categories based on the approach which is used to build those [13]. These two approaches are defined in [13] as follows.

- "**The fixed model approach:** We assume that all important quality factors needed to monitor a project are a subset of those is a published model. To control and measure each attribute, we accept the model's associated criteria and metrics. Then we use data collected to determine the quality of the product
- **The 'define your own quality model' approach:** We accept the general philosophy that quality is composed of many attributes, but we do not adopt a given model's characterization of quality. Instead, we meet with prospective users to reach a consensus on which quality attributes are important for a given product. Together we decide on a decomposition (possibly guided by an existing model) in which we agree on specific relationships between them. Then we measure the quality attributes objectively to see if they meet specified, quantified targets."

Boehm and McCall models are typical examples of fixed quality models. Trendowicz and Punter [37] has done an excellent survey of different approaches of modeling quality. They have discussed three main requirements for appropriate quality modeling- *flexibility, reusability* and *transparency.* These three requirements are discussed here.

- **Flexibility:** The quality models should be flexible because it is context dependent. The possible contexts are company context, project context and process context. As each company has its own characteristics and requirements and different quality objectives, so the quality models need to be flexible enough to be applicable across different companies. Similarly different projects and processes have different quality requirements. Embedded systems may need a different quality model than the web application. Similarly process context reflects the characteristics of a software development process like its stability or availability of measurable objects in different process phases. Another important aspect of flexibility is the need of experts' assessment and people's experience to build quality models together with quantitative data.

- **Reusability:** Depending on the projects' similarity level, quality model should support the reuse of measurement data as well as

quality characteristics and their relationship. It enhances the accuracy and efficiency if the quality models incorporates experiences from past.

- **Transparency:** The quality model should be transparent so that the relationships between the characteristics have some rationale. And it also should allow the expert to directly interfere to model structure for any necessary modification.

[37] has done some critical review of fixed model approach and define-your-own-model approach based on these three requirements. It is very much evident that the fixed model approaches lack flexibility as the characteristics and their relationships are defined as constant. Fixed model approach also lacks transparency because it usually provides no logic behind how the characteristics are decomposed into sub-characteristics. Another main drawback of these models is their reliance only on quantitative measurements. These models are usually unable to make benefit from the qualitative data i.e., expert judgment.

Define-your-own-model approach has more flexibility in the sense that it does not impose any prescriptive set of characteristic. [37] has also defined the quality models as *directly-defined* and indirectly-defined. Project stakeholders define the characteristics and their sub-characteristics and their

relationship in the form of dependency graph in the directly-defined models. On the other hand, indirectly–defined models are usually generated automatically from the measurement data. The experts can control these quality models by selecting the appropriate techniques and some parameters to explore the relationship between internal and external attributes. Although in most of the cases, the quality relationships are represented in too complex way to understand. Bayesian Belief Network is an example of directly defined models. Statistical models and some artificial intelligence technique based models are examples of indirectly-defined models. We shall present a literature survey on different types of models in Chapter 2.

1.2 Imprecision and Uncertainty in Quality Models

As Wang noted [38], for most engineering systems, there are two important information sources: *sensors* which provide numerical measurements of variables, and *human experts* who provide linguistic instructions and descriptions about the system. Quality models are no exceptions in this sense. On the one hand, the sources of *knowledge* regarding the relationships between the different quality attributes (characteristics) are *numerical knowledge* from statistical analysis, and *linguistic knowledge* from human experts. For example, COCOMO provides a numerical knowledge about the relationship between the internal attribute that is the number of lines of code,

and the external attribute that is the effort; including the mode of the product as a factor. While experts may give similar knowledge but in linguistic form such as [2]

IF mode is Organic AND size is High THEN cost is Medium

IF mode is Semi-detached AND size is High THEN effort is a Little-High

IF mode is Embedded AND size is High THEN effort is High

IF mode is Organic AND size is Medium THEN effort is Low

…

Or in general,

IF mode is m_j *AND size is* s_i *THEN effort is* c_{ji} $\quad (1 \le i \le n, 1 \le j \le 3)$

Where m_j are the fuzzy values for the fuzzy variable mode, s_i $(1 \le i \le n)$ are the fuzzy values for the fuzzy variable *size*, and C_{ji} $(1 \le i \le n, 1 \le j \le 3)$ are the fuzzy values for fuzzy variable cost (effort).

On the other hand, the information used to assess/predict quality using the quality model has also two sources: numerical information that is coming form the corresponding metrics, and linguistic information coming form the experts' judgment. Figure 1 shows the different sources of knowledge in software quality models.

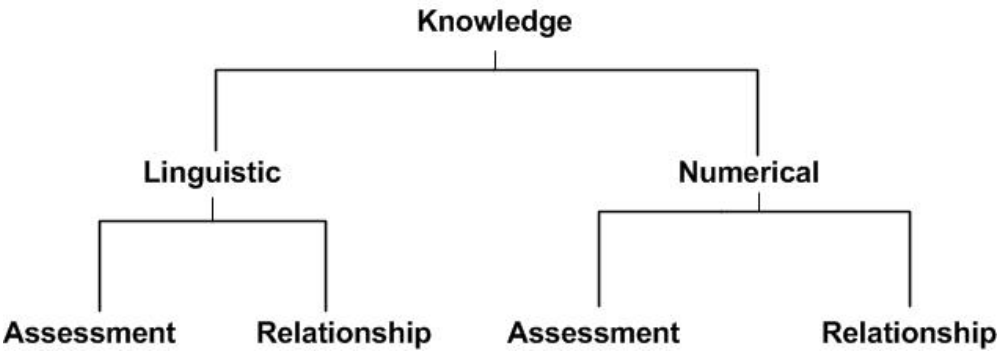

Figure 1: Different sources of knowledge in Software Quality Models

Obviously, traditional statistical regression analysis approaches can only make use of numerical information and have difficulty incorporating linguistic knowledge. Because so much human knowledge is available and valuable with regard to quality aspects as with other engineering systems, incorporating it into engineering systems in a systematic and efficient manner is very important.

However, as seen in the above COCOMO example, human knowledge is imprecise in nature and human being likes to represent knowledge using words i.e. linguistic variables. As another example, an expert would describe the relationship between coupling and reliability as "high coupling may produce high number of faults", as opposed to saying that "coupling values of 10 to 20 will produce 5 to 10 faults". One of the earlier works that talks about imprecision in software quality models is due to Ebert [9] [10]. Ebert mentioned that the metric values are usually continuous in nature and it is

hard to distinguish between good or bad measurement although these values are precise. He also suggested that the expert comments on the relation between internal and external attributes may cover these problems with the use of words rather than using precise numerical relation. Accordingly, successful quality models should take experts knowledge into consideration. Therefore, we propose a framework for building models based on both expert knowledge as well as historical data.

Another issue that would arise when trying to develop such a framework is that historical data as well as expert knowledge are surrounded by uncertainty. Uncertainty, however, has not been addressed by many of the previous works. Fenton and Nell [11] [12] raised this issue but did not discuss in details about the nature and cause of uncertainty.

As discussed earlier, expert knowledge with regard to the relationships that forms the quality model is represented in an imprecise way. Along with this imprecision, there are two associated uncertainties: *relationship uncertainty* and *assessment uncertainty*. The relationship uncertainty verily exists in the expert judgments regarding the nature of the relation between internal and external attributes. People generally differ in their judgments on the impact of certain internal attributes on a particular external attribute. For example, one expert may assert that "high coupling produces high number of faults"

while another may assert that “high coupling produces very high number of faults”. So, the impact of internal attributes is also uncertain. In other words, the “relationships” are not certain. With regard to the assessment uncertainty, experts may have slightly different optioning when judging artifacts’ quality. For example, considering the cohesion of a software component; one expert may rate it as highly cohesive, while another may rate it as moderately cohesive. For both types of uncertainty surrounding the expert knowledge, the definition or meaning of the words may be uncertain too. In the examples we have used words like low and high etc. Expert may mean different thing for the same word.

Uncertainty is not only surrounding the experts’ knowledge, as we have seen; it also surrounds the knowledge extracted from statistical analysis and measurements as well. Similar to the experts’ knowledge, numerical knowledge suffer from both relationships uncertainty and assessment uncertainty. As for the relationships uncertainty, it is mainly due to the laziness/ignorance and to some extent to the accuracy of the regression model used.

On the other hand, for the assessment uncertainty, there have been typically more than one metric proposed for assessing each quality attribute. Each metric tries to capture the correct measurement of an attribute considering

different factors. The accuracy of prediction models greatly depends on how the existing metrics capture which aspect of an attribute measurement. For example, there are different metrics proposed to calculate cohesion such as LCOM1, LCOM2, TCC, and DCI. We may never know which one of these exactly calculates cohesion. This sort of uncertainty in the measurements occurs mainly because, in most cases the definition of the metric itself is abstract and people try to instantiate this abstraction based on their own understanding. In summary, we can say that the two categories of uncertainty—the relationships uncertainty and the assessment uncertainty—surround both sources of knowledge. Accordingly, handling uncertainties is necessary for establishing more effective quality models.

1.3 Problem Statement

In the previous section, we have discussed the importance of the issues of imprecision and uncertainty in the domain of software quality. We shall see in Chapter 2 that some algorithmic and non-algorithmic approaches have been previously used to build software quality prediction models. Statistical models rely totally on historical data and so transparency is not present in this models. We can not explain the nature of relations between the internal and external attributes. Fuzzy logic based models are transparent but can not handle uncertainty and probabilistic models can deal with uncertainty but can

not build transparent models. Due to these shortcomings, none of these approaches can be used as a general framework to build software quality models where both imprecision and uncertainty will be handled together.

We investigated these problems in the existing approaches and set the objective of this thesis is to develop a framework which should be

- General: can be used to determine functional relation between arbitrary internal and external attributes
- Able to build transparent models: experts can incorporate their knowledge and modify the model based on some rationale
- Able to handle four types of uncertainty

1.4 Main Contributions

The main contributions of this work are as follows.

i) Defining four types of uncertainty in software quality models

ii) Developing a general framework which is able to build transparent models and can handle imprecision and four types of uncertainty in software quality models.

iii) Conducting experiments which compare some of the existing approaches with our proposed framework.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 presents the literature survey done to investigate other techniques for building software quality models. Chapter 3 discusses the preliminaries of type-1 and type-2 fuzzy logic. Chapter 4 presents our framework. Chapter 5 shows the experimental results. At the end, Chapter 6 concludes this thesis mentioning the contributions, limitations and future work.

Chapter 2

Literature Survey

Previously both algorithmic and non algorithmic techniques have been used to build quality models. Algorithmic approach uses the historical data to come up with a functional relationship. Non algorithmic approaches use expert judgment, probabilistic models and some other soft computing techniques to approximate the functional relation. Regression analysis is the most widely used algorithmic approach. The other algorithmic techniques are discriminant analysis, principal component analysis etc. Among non algorithmic techniques, probabilistic and soft-computing approaches are common. We tried to look at different techniques from the perspectives of imprecision, uncertainty, transparency and generality.

2.1 Algorithmic Approach

We found many works where different types of statistical regression analysis have been used to build software quality models. Most of the works concentrated on building software fault prediction models, where number of faults is predicted for individual software modules based on some internal attribute measurement. Some work on software cost estimation is also found in literature. [2] can be used as a nice summary of the works in this

direction. Basilli *et al.* tried to validate some OO metrics to use a quality indicator. They applied statistical univariate and multivariate analysis to conclude that some OO metrics can be useful to predict class fault proneness. Briand *et al.* used univariate and multivariate logistic regression to explore the relationship between design measures and software quality in object oriented systems [5]. They tried to predict the probability of fault in OO software module. From their developed models, the best model showed a percentage of correct classification higher than 80% and finds more than 90% of faulty classes. Chidamber, Darcy and Kemerer used some statistical correlation based exploratory analysis to conduct empirical investigation [7]. They reported that some OO metrics may be very useful to explain the variations in some external attributes like productivity, rework effort, design effort. Denaro and Pezze applied some multivariate regression analysis techniques to build fault prediction models [8]. They used data from Apache 1.3 as training set and Apache 2.0 as testing set. They tested the performance of the models and concluded that if the models are applied on software from homogeneous environment, they can perform well. Briand *et al.* has shown that MARS (Multivariate Adaptive Regression Analysis) based techniques can be very good candidate for building fault proneness models across object oriented software projects [6].

2.2 Non-Algorithmic Approach

Lanubile *et al.* applied Artificial Neural Network (ANN) with some other techniques like principal component analysis, logistic regression, logical classification, and discriminant analysis to classify fault prone software components [25]. But from their experiment, they found that no model is sound enough to discriminate between faulty and non-faulty modules. In 1994, Khoshgoftaar *et al.* introduced a neural network classification model for identifying high risk program modules [24]. They concluded that neural network provides a better management tool in software engineering environment. There are also other works in literature that use ANN to predict different software quality attributes [18]- [23].

Fenton and Nell first proposed using Bayesian Belief Net (BBN) for predicting software quality [11], [12]. Their research successfully pointed out some limitations of the existing prediction models. One of the limitations they reported is that none of the existing models care about the uncertainty factor. However, Fenton and Nell did not discuss the nature and types of uncertainty in software quality models.

Ebert in 1993 successfully focused on the imprecision issue in quality prediction models [10]. He proposed to use fuzzy logic based prediction model as it has the superiority over crisp classification techniques to deal

with imprecision. He also argued in favor of a fuzzy logic based model because it can incorporate expert knowledge in the model along with historical data. Ebert in 1997 published his experiment results in comparing among different techniques and the fuzzy classification technique [9]. He found that fuzzy classification outperforms some other techniques such as classification trees and factor based discriminant analysis. He also argued against using neural net because the neural nets rely only on experimental data. In the same line, Cha and Kwon have proposed fuzzy logic based model to predict error-prone software modules from inspection data [34].

Chapter 3

Imprecision, Uncertainty and Fuzzy Logic Systems

Definition of fuzziness and imprecision in this section are extracted from [14]. Fuzziness should not be confused with other forms of imprecision and uncertainty. There are several types of imprecision and uncertainty and fuzziness is just one aspect of it. Imprecision and uncertainty may be in the aspects of measurement, probability, or descriptions. Imprecision in measurement is associated with a lack of precise knowledge. Imprecision in description is the type of imprecision addressed by fuzzy logic. It is the ambiguity, vagueness, qualitativeness, or subjectivity in natural language (linguistic, lexical, or semantic uncertainty). It is the ambiguity found in the definition of a concept or the meaning of terms such as "tall building" or "low scores". It is also the ambiguity in human thinking, that is, perceptions and interpretations. Examples of statements that are fuzzy in nature are "Hemoglobin count is very low." And "Teddy is rather heavy compared to Ike." The nature of fuzziness and randomness are therefore quite different. They are different aspects of imprecision and uncertainty. The former conveys subjective human thinking, feelings, or language, and the latter indicates an objective statistic in the natural sciences. From the modeling

point of view, fuzzy models and statistical models also possess philosophically different kinds of information: fuzzy memberships represent similarities of objects to imprecisely defined properties, while probabilities convey information about relative frequencies. Thus, fuzziness deals with deterministic plausibility and not nondeterministic probability.

Uncertainty is a very important aspect of real human life. By the dictionary definition, it means "Not knowing with certainty, doubtful; not definitely known; such as cannot be definitely forecast; subject to chance; not to be depended on; changeable" [32]. This definition can be extended to the context of AI. Uncertainty in AI is "Given the knowledge base, current and previous percepts, if the agent still cannot answer a question regarding the domain, then this agent must act under uncertainty" [33]. This uncertainty occurs mainly due to three reasons:

1) Volume of work: It is too much work to list all the antecedents and consequences in the problem domain.
2) Lack of theoretical knowledge: We usually do not know enough about the domain to list every consideration.
3) Lack of experimental results: It may be that we do not have all the tests to run, or we do not want to run all the tests.

3.1 Imprecision and Type-1 Fuzzy Logic Systems

In this section, we will discuss the preliminaries of Type-1 fuzzy logic systems and also how imprecision issue is handled by Type-1 FLS.

3.1.1 Fuzzy Sets and Linguistic Variables

L. Zadeh defined fuzzy logic in the foreword of Wang's book [38] - "In a broader and much significant sense, fuzzy logic is coextensive with the theory of fuzzy sets, that is, classes of objects in which the transition from membership to non-membership is gradual rather than abrupt". So, before defining a fuzzy logic system, fuzzy sets and linguistic variables should be explored first. Linguistic Variables, Linguistic Values, Linguistic Terms: Just as numerical variables take numerical values, in fuzzy logic, linguistic variables take on linguistic values which are words (linguistic terms) with associated degrees of membership in the set. Thus, instead of a variable height assuming a numerical value of 1.75 meters, it is treated as a linguistic variable that may assume, for example, linguistic values of tall with a degree of membership of 0.92, "very short" with a degree of 0.06, or "very tall" with a degree of 0.7. This concept was introduced by Zadeh to provide a mean of approximate characterization of phenomena that are too complex or too ill-

defined to be amenable to description in conventional quantitative terms. Linguistic variables take on values defined in its term set - its set of linguistic terms. Linguistic terms are subjective categories for the linguistic variable. For example, for linguistic variable age, the term set T(age) may be defined as follows:

T(age) = { "young", "not young", "not so young", "very young", ..., "middle aged", "not middle aged", ..., "old", "not old", "very old", "more or less old", "quite old", ..., "not very young and not very old", ... }

Fuzzy Sets and Membership Functions: Each linguistic term is associated with a fuzzy set, each of which has a defined membership function (MF). Formally, a fuzzy set A in U is expressed as a set of ordered pairs

$$A = \{(x, m_A(x)) | x \text{ in } U\}$$

Here $m_A(x)$ is the membership function that gives the degree of membership of x. This indicates the degree to which x belongs in set A. Here U can be called the universe of discourse. Let's illustrate these concepts using an example. We know LCOM is a metric to measure the lack of cohesion in object oriented system. Figure 2 illustrates a linguistic variable LCOM with three associated linguistic terms namely "low", "medium" and "high". Each of these linguistic terms is associated with a fuzzy set defined by a

corresponding membership function. Anyway, the membership functions shown in the figure are just for illustration.

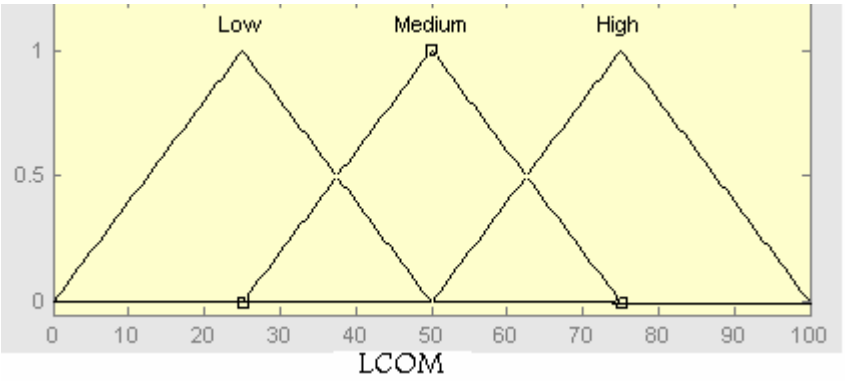

Figure 2: Membership functions for LCOM

There are many types of membership functions. Some of the more common ones are triangular MFs (such as the functions in the Figure 1), trapezoidal MFs, Gaussian MFs, and generalized bell MFs.

3.1.2 Fuzzy Logic Systems

Fuzzy logic systems (FLS) are name for the systems which have a direct relationship with fuzzy concepts (e.g., fuzzy sets, linguistic variables and so on) and fuzzy logic. The most popular fuzzy logic systems in the literature may be classified into three types: pure fuzzy logic systems, Takagi and Sugeno's fuzzy system, and fuzzy logic system with fuzzifier and defuzzifier. As most of the engineering applications produce crisp data as input and expects crisp data as output, the last type is the most widely used

one. Figure 3 shows the basic configuration of a fuzzy logic system with fuzzifier and defuzzifier. This type of fuzzy logic system was first proposed by Mamdani. It has been successfully applied to a variety of industrial processes and consumer products. The main fours components' functions are as follows.

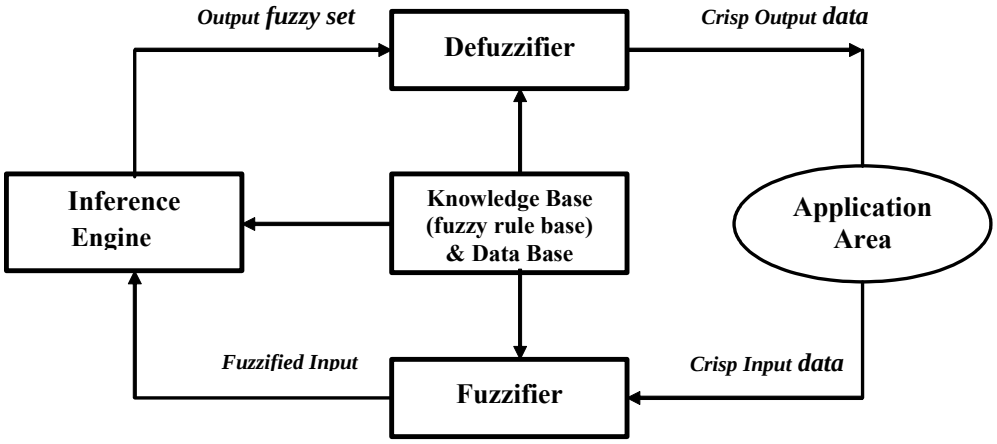

Figure 3: Fuzzy logic system with fuzzifier and defuzzifier

Fuzzifier: Fuzzifier does a mapping from crisp input to a fuzzy set.

Fuzzy Rule Base: Fuzzy logic systems use fuzzy IF-THEN rules. A fuzzy IFTHEN rule is of the form

"IF $X_1 = A_1$ and $X_2 = A_2$... and $X_n = A_n$ T HEN Y = B"

where X_i and Y are linguistic variables and A_i and B are linguistic terms. The IF part is the antecedent or premise, while the THEN part is the consequence or conclusion. An example of a fuzzy IF-THEN rule is

"IF LCOM = Low THEN FAULT =High".

In a fuzzy logic system, the collection of fuzzy IF-THEN rules is stored in the fuzzy rule base which is referred to by the inference engine when processing inputs.

Fuzzy Inference Engine: Once all crisp input values have been fuzzified into their respective linguistic values, the inference engine will access the fuzzy rule base of the fuzzy expert system to derive linguistic values for the intermediate as well as the output linguistic variables. The two main steps in

the inference process are aggregation and composition. Aggregation is the process of computing for the values of the IF (antecedent) part of the rules while composition is the process of computing for the values of the THEN (consequent) part of the rules.

Defuzzifier: Defuzzifier does a mapping from the fuzzy output to the crisp output The details of the above four components can be found in Wang's book [38].

3.1.3 Adaptive Fuzzy Logic

The definition of adaptive fuzzy system given by Wang in his book [38] is a good one and easy to understand - "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". Here parameters are the necessary values to construct the membership functions. Membership functions are adjusted by a set of input-output pairs. This is adaptive in the sense that the necessary changes are made only locally to the affecting membership functions whereas trainable neural networks globally adjust all the weights. So, adaptive fuzzy logic is a nice way of combining linguistic and numerical information, which can be done in two ways [38]-

- Use linguistic information to construct an initial fuzzy logic system, and then adjust the parameters of the initial fuzzy logic system based on numerical information.
- Use numerical information and linguistic information to construct two separate fuzzy logic systems, and then average them to obtain the final fuzzy logic system.

3.2 Uncertainty in Fuzzy Logic Systems

We discussed general concepts of uncertainty in the beginning of this chapter. This concept has been clarified in our discussion of uncertainty in the context of our problem domain in Section 1.2.

Mendel [27] has noted that uncertainty also exists while building and using typical fuzzy logic systems. He has described four sources of uncertainty. Those are summarized here.

i. *Uncertainty about the meanings of the words that are used in a rule*. This is the uncertainty with the membership functions because membership functions represent words in a FLS. It can be both antecedents and consequents.

ii. *Uncertainty about the consequent that is used in a rule.* This is the uncertainty with the rule itself. A rule in FLS describes the impact of the antecedents on the consequent. Expert may vary in their opinion to decide this nature of impact.

iii. *Uncertainty about the measurements that activate the FLS.* This is the uncertainty with the crisp input values or measurements that activates the FLS systems. These measurements may be noisy or corrupted. This noise can again be in a certain range or totally uncertain meaning stationary or non-stationary.

iv. *Uncertainty about the data that are used to tune the parameters of a FLS*. This is the uncertainty with the measurements again. But these measurements are used to train the FLS as opposed to that of (iii) which are used to activate the FLS.

3.3 Uncertainty and Type-2 Fuzzy Sets

Mendel has proposed using Type-2 fuzzy sets and Type-2 fuzzy logic systems to deal with the four types of uncertainty discussed in the previous section. Type-2 fuzzy sets were first proposed by Zadeh [39] in 1975. But the characterization of type-2 fuzzy sets was first done by Mendel and Liang in 1999 [28]. They characterized type-2 fuzzy sets using the concept of footprint of uncertainty and upper and lower membership functions. Actually type-2 fuzzy sets are three dimensional whereas type-1 is two dimensional. This extra dimension lets uncertainty to be handled by type- 2 fuzzy sets. We will now see the definition of type-2 fuzzy sets and how they can help to model uncertainty. We use the definition and figures from Mendel's book [27]. Type-2 fuzzy sets help us to deal with the first source of uncertainty i.e. uncertainty about the meaning of the words. Type-1 fuzzy sets can not deal with this type of uncertainty because degree of membership is considered as certain in type-1 fuzzy sets. On the other hand, the blurred area i.e. the second dimension in a type-2 fuzzy set adapts the concept of uncertainty. Mendel calls this blurred area as *footprint of uncertainty (FOU).*Actually here the concept is to consider different degree of membership for each of the values in the universe of discourse. Fuzzy sets are used to represent word or linguistic variables and people really differ in how to interpret a particular

word. So, the concept of 2nd dimension in type-2 fuzzy set provides this flexibility to incorporate different person's view in a fuzzy set. We will discuss more on this issue in the later part of this thesis.

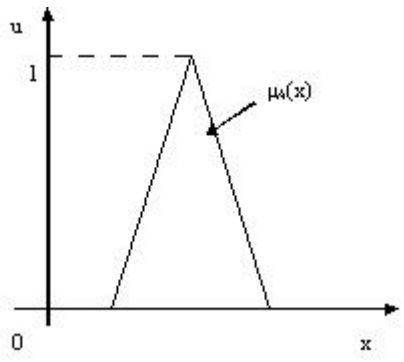

Figure 4: A Type-1 triangular membership function

Let's imagine blurring the type-1 membership function depicted in Figure 4 by shifting the points on the triangle either to left or to right and not necessarily by the same amounts, as in Figure 5. Then at a specific value of x, say x′, there no longer is a single value for the membership function; instead the membership function takes on values wherever the vertical line intersects the blur.

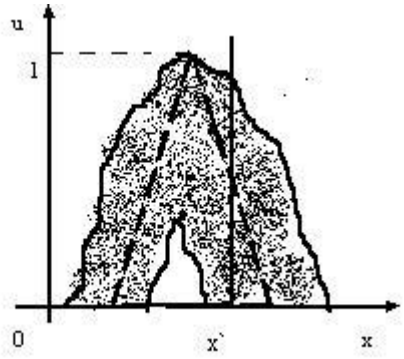

Figure 5: Blurred triangular membership function

Those values need not all be weighted the same; hence, we can assign an amplitude distribution to all of those points. Doing this for all $x \in X$, we create a three-dimensional membership function- a type-2 membership function- that characterizes a type-2 fuzzy set. Type-2 membership functions have same constraint of type-1 membership functions. The degree of membership along the second dimension is always in the interval [0, 1]. The amplitude distribution i.e. the values along the 3rd dimension also lay between the interval [0, 1]. So, it is clear that if the blur disappears, then a type-2 membership function must reduce to a type-1 membership function.

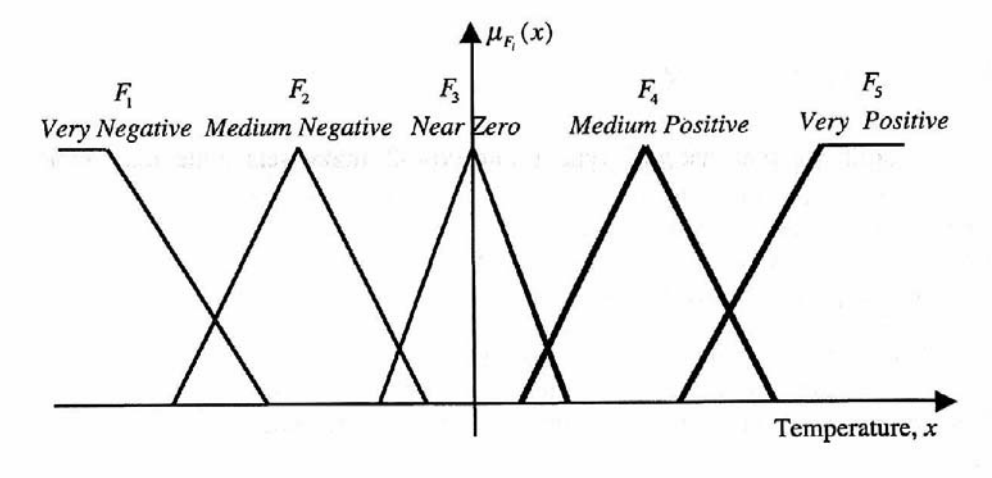

Figure 6: Type-1 fuzzy sets

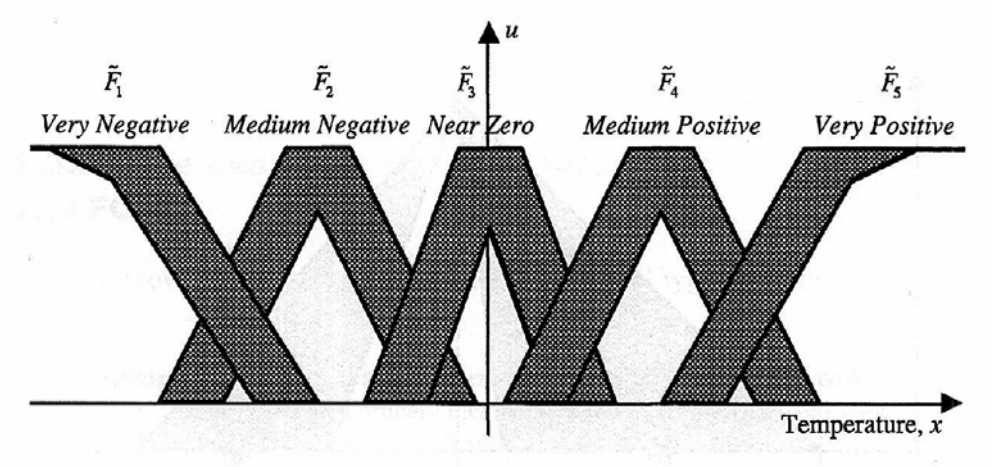

Figure 7: FOU s for the Figure 6 membership functions.

Figure 6 shows some triangular membership functions and is the FOU for those membership functions [27]. The shaded or blurred area is our FOU i.e. the second dimension that helps to deal with uncertainty. We see in the figure that this FOU is uniformly shaded. It means that that at each point in the FOU, the membership degree is one. This type of membership functions

are called interval type-2 membership functions. Imposing this constraint helps to build the fuzzy logic system but it also poses some limitations.

We have used Gaussian membership functions in our experiments as Mendel used these to build the fuzzy logic systems. Now, let's see some examples on type-2 Gaussian membership functions. Let's consider the case of a Gaussian membership function having a fixed standard deviation, σ, and an uncertain mean that takes on values in $[m_1, m_2]$. Figure 8 is an example.

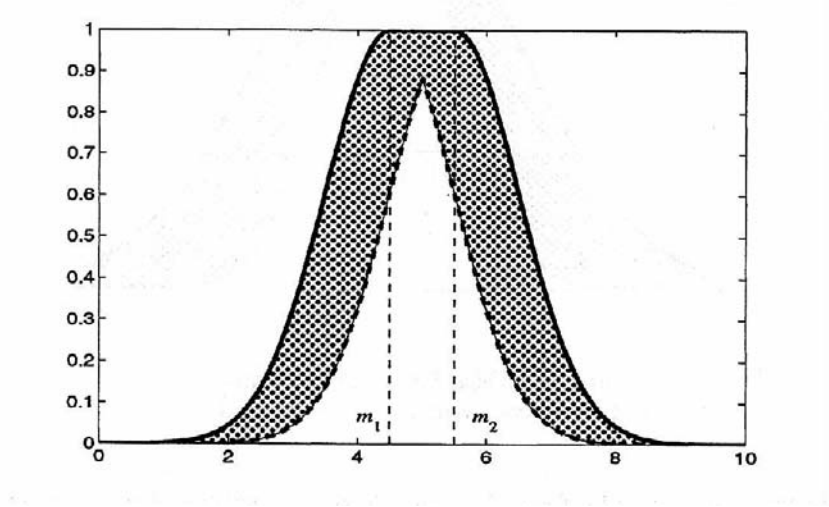

Figure 8: FOU for Gaussian primary membership function with uncertain mean

Similarly, let's consider the case of a Gaussian membership function having a fixed mean, m, and an uncertain standard deviation that takes on values in $[\sigma_1, \sigma_2]$. Figure 9 is an example.

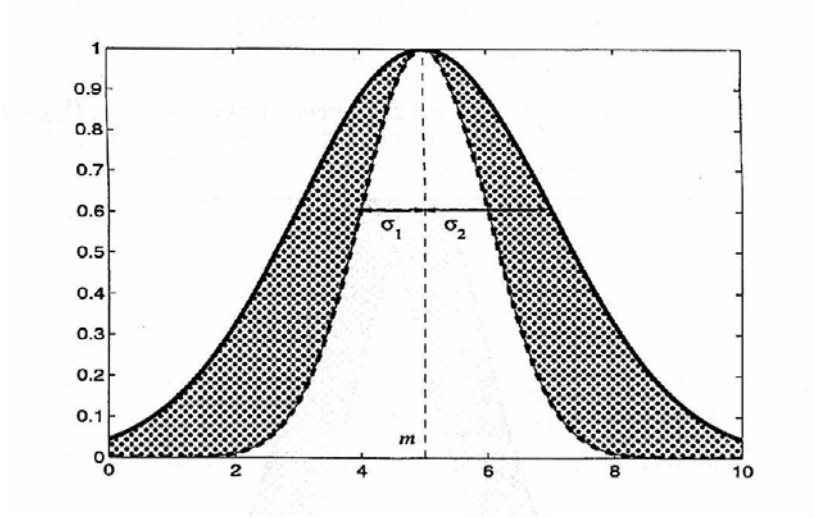

Figure 9: FOU for Gaussian primary membership function with uncertain standard deviation

It is easy to see here that both the Gaussian membership functions are of interval type-2 as the shading is uniform. Mendel developed fuzzy logic systems using these two types of Gaussian membership function.

3.4 Uncertainty and Type-2 Fuzzy Logic System

Although Zadeh proposed the concept of type-2 fuzzy sets [39], Karnik and Mendel [17] for the first time extended the concept of type-2 fuzzy sets to build type-2 fuzzy logic systems. In this section, we will describe the main components of a type-2 fuzzy logic system and we will also see how the uncertainty issues are considered. But before that, we add a subsection here to discuss the concept of probabilistic models vs. type-2 fuzzy logic systems

which is due to Mendel [27]. This discussion is essential because probabilistic model is considered to be the best option to deal with uncertainty.

3.4.1 Probabilistic Models vs. Type-2 Fuzzy Logic Systems

Mendel [27] discussed the similarity or differences between type-2 FLS and probabilistic models that may help to understand more about how random uncertainty is modeled in type-2 FLS. Probabilistic models represent random uncertainties using probability density functions (pdf). As many moments a pdf can use, it can model uncertainty better. For example, if the pdf is Gaussian, it has two moments- mean and variance. This second order i.e. variance tries to provide an understating about the dispersion about the mean. Although it is difficult to compare a FLS with a pdf, from these moments point of view some analogy may be found. A type-1 FLS produces a defuzzified output which may be compared to first order moment i.e. mean of a pdf. This defuzzified output considers the result as fully certain. On the other hand, the output of a type-2 FLS is a type reduced set with two interval endpoints. The second order moment of a pdf is used as a confidence interval and similarly type reduced interval set can be thought as a linguistic confidence interval. As the uncertainty increases, this interval set also increases. So, conceptually type-2 FLS is analogous to the probabilistic

models from the perspective of the first and second order moments of a pdf. Mendel [27] also mentioned about the superiority of type-2 FLS over probabilistic model when data does not agree with the a priori knowledge of the pdf. He suggests using framework of a type-2 FLS when probabilistic models cannot be used because of system complexities such as non-linearity, time-variability or non-stationarity.

3.5 Fuzzification in Type-2 Fuzzy Logic System

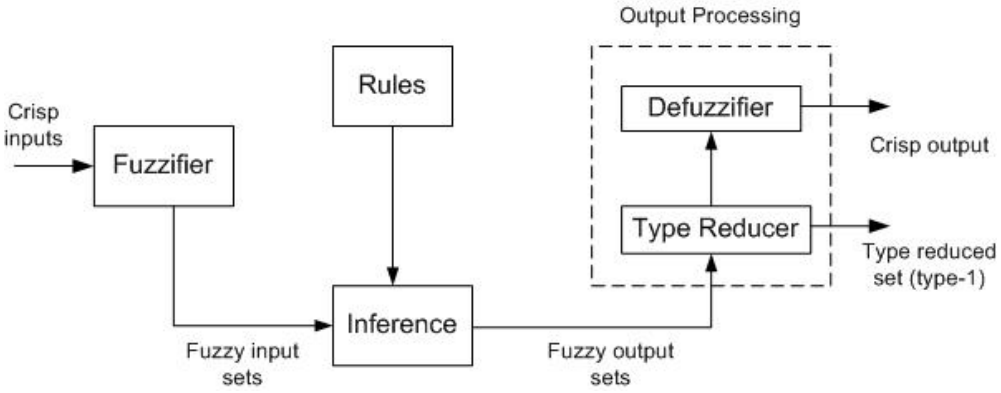

Figure 10: Type -2 FLS

A fuzzy logic system is considered to be type-2 as long as any one of its antecedent or consequent sets is type-2. All the components of Figure 10 have been discussed in details by Mendel [27]. Fuzzifier is one of the most important components from the aspect of uncertainty. Here we shall discuss fuzzification because it helps to handle uncertainty.

Fuzzification can be done in mainly two ways- singleton and non-singleton. Singleton fuzzification considers the measurement that activates the FLS to be certain and noise free. Non-singleton considers the input crisp measurement to be uncertain. In singleton, the result of fuzzification is a fuzzy singleton i.e., only at the input measurement, the membership function has a value of 1. On the other hand, conceptually, a non-singleton fuzzifier implies that the given input value is the most likely value to be correct one from all the values in its immediately neighborhood; however, because the input is corrupted by noise, neighboring points are also likely to be the correct value, but to a lesser degree. So, fuzzy membership function is used for fuzzification where the fuzzy membership function is centered at the measurement value. This non-singleton fuzzification can also be done in two ways – Type-1 and Type-2 based on the type of fuzzy sets used for fuzzification. When the noise is stationary, we can use the type-1 non-singleton fuzzification and when the noise is non-stationary, we can use type-2 non-singleton. Based on different types of fuzzification and different types of antecedent fuzzy sets, Mendel has developed 5 different fuzzy logic systems [27]. Those five different FLS s are –

a) Singleton type-1

b) Non-singleton type-1

c) Singleton type-2

d) Non-singleton type-2 with type-1 inputs

e) Non-singleton type-2 with type-2 inputs

Figure 11 shows a pictorial description of these 5 different fuzzy logic systems [27].

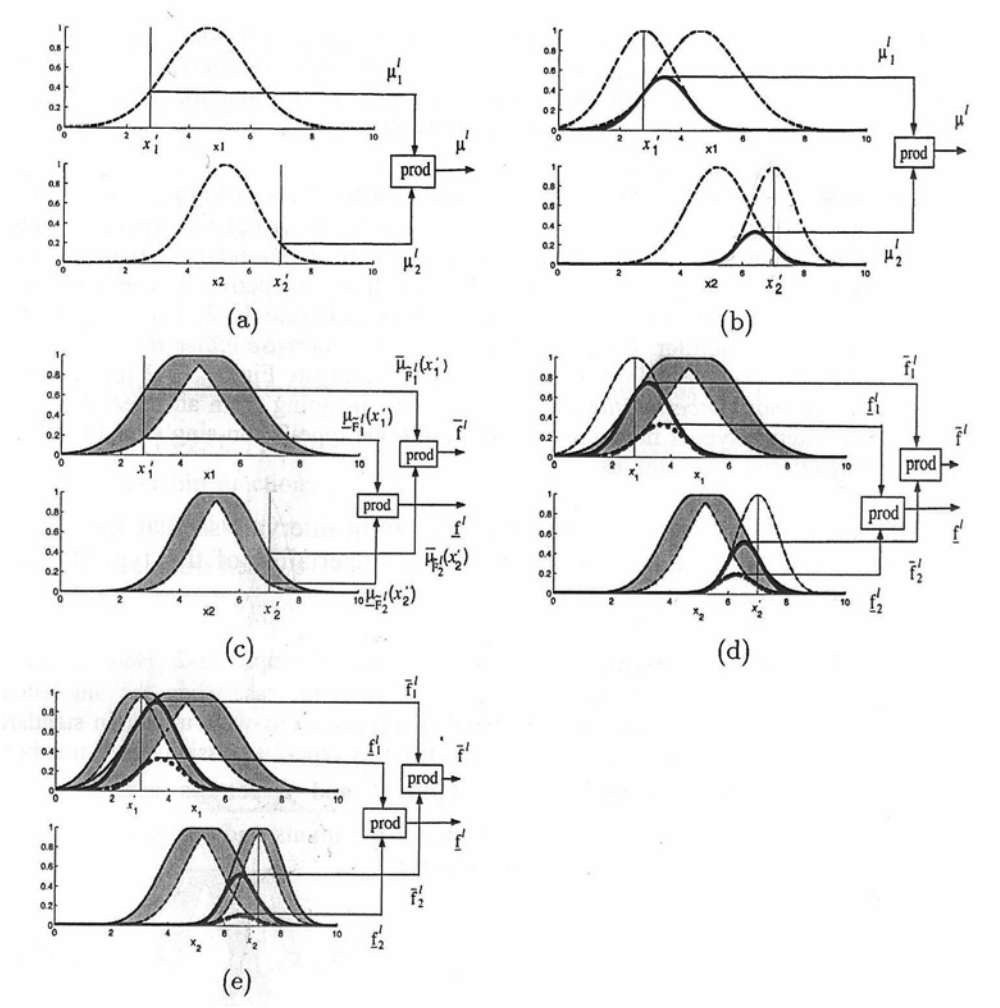

Figure 11: Different types of FLS – (a) singleton type-1, (b) non-singleton type-1, (c) singleton type-2, (d) non-singleton type-2 with type-1 inputs, (e) non-singleton type-2 with type-2 inputs

Mendel has shown in Table 1, which type of noise i.e. uncertainty can be handled by which FLS.

Table 1: Different Fuzzy Logic Systems to handle different types of noise

Type of FLS	Measurement Noise	Training and Testing Data	Measurements that is used after building the FLS
Singleton type-1	None	Noise Free	Noise Free
Non-singleton type-1	Stationary	Noisy	Noisy
Singleton type-2	Stationary	Noisy	Noise Free
Type-1 non-singleton type-2	Stationary	Noisy	Noisy
Type-2 non-singleton type-2	Non-Stationary	Noisy	Noisy

Chapter 4

Framework for Building Software Quality Models

In this chapter we shall present our framework to build software quality models which takes care of all types of uncertainty and imprecision. Already we have demonstrated that fuzzy logic is good enough to handle imprecision in software quality models and have discussed Mendel's approach [27] to deal with uncertainty. In Section 1.2, we discussed different types of uncertainty that should be considered while developing software quality models. In Section 3.2 we have discussed the four types of uncertainty mentioned by Mendel in a fuzzy logic system. Before approaching to build the framework, we need to show that there is a mapping between the uncertainty discussed by Mendel and our findings in software quality models. Table 2 shows a summary of this mapping.

It is evident from the Table 2 that if we build our framework based on Type-2 FLS, we can solve the uncertainty problem in our software quality domain. In the previous Chapter, we have presented the basic concepts of type-2 fuzzy logic. Now we shall see how those concepts help us to deal with four types of uncertainty mentioned by Mendel. Then we shall present our framework.

Table 2: Uncertainties in FLS and Software Quality Model

Uncertainty in Software Quality Models	Uncertainty in FLS	Example
Linguistic Assessment	Meaning of the word	Expert judgment on the measurement of an internal attribute
Linguistic Relationship	Consequent	How the internal attributes contribute to the external attribute
Numerical Assessment	Measurement to activate FLS	Different metrics to measure a particular attribute
Numerical Relationship	Data to build the FLS	Rely only on historical data to build a model

4.1 Type-2 Fuzzy Logic and Four types of Uncertainties

In this Section we shall see how the four types of uncertainty mentioned in Section 3.2 can be solved using Mendel's approach [27].

4.1.1 Uncertainty about the meanings of the words that are used in a rule

From our discussion of type-2 fuzzy sets, it is evident that type-2 fuzzy sets can help us to handle this uncertainty. Actually people interpret the same word differently. For example, if we have a range of values 0-10 and ask people about the word 'LOW', we expect to get different sub ranges for LOW. For example, some may say 0-2 is low or some may say 0-3 is low. We know that we can represent any interpretation of the word using a type-1 fuzzy set. Now, we need to combine different type-1 fuzzy sets to form one

type-2 fuzzy set to represent several expert opinions in one word. Mendel [29] proposed to use union operation to combine different type-1 or type-2 fuzzy sets to form a type-2 fuzzy set.

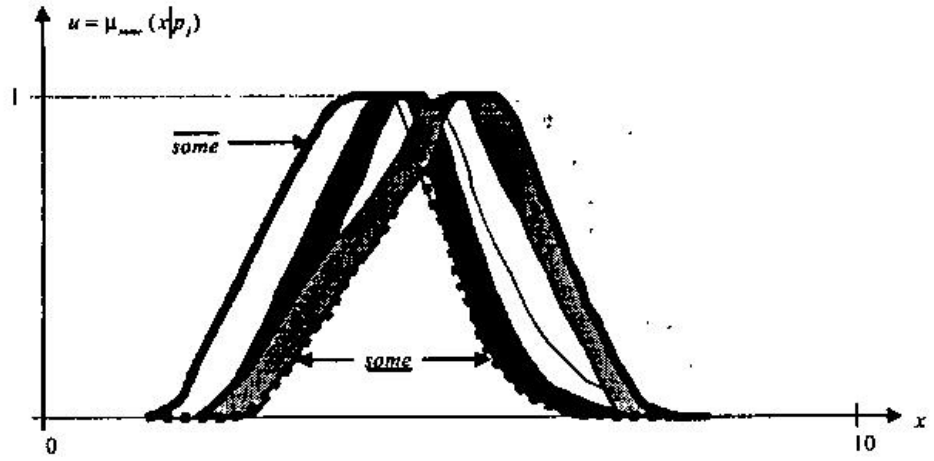

Figure 12: Different fuzzy sets which need to be combined [29]

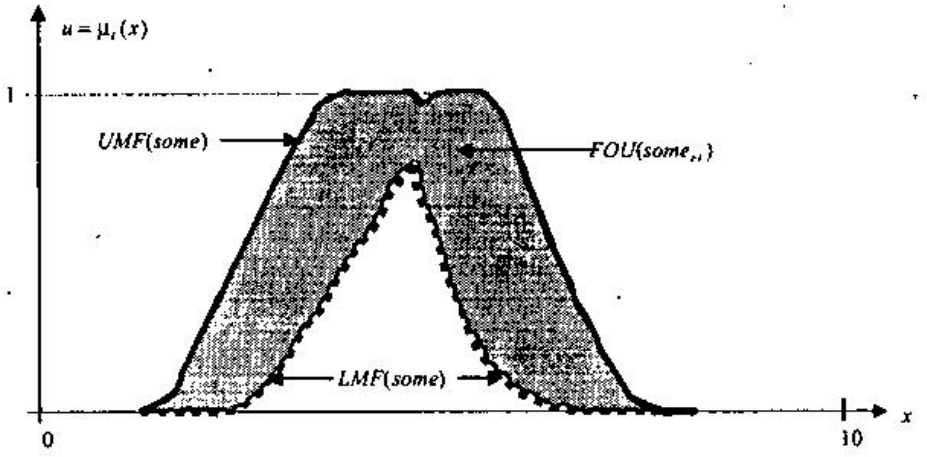

Figure 13: Union operation to form a Type-2 fuzzy set [29]

We can conduct a survey to get opinions from different experts. But interval type-2 fuzzy sets with uniform shading i.e. FOU is used in the FLS developed by Mendel. It has a problem when the expert opinions are not uniformly distributed. And usually more experts have close opinion while few can have opinion that is far. In this case uniform shading of type-2 fuzzy set is a limitation. Still, considering the computational complexity of general type-2 fuzzy sets, interval tpe-2 fuzzy set is the right choice so far. There is another way of handling this situation—we can drop some experts' opinion as outliers if those are really far from most of the others.

This uncertainty is very much similar to our defined linguistic assessment uncertainty which is the difference of opinion of experts while assessing a software artifact. So we can collect experts' assessment of the measure of an artifact in form of type-1 fuzzy sets. And then union operation of those type-1 fuzzy set can produce a type-2 fuzzy set which will represent this assessment uncertainty.

4.1.2 Uncertainty about the consequent that is used in a rule

Survey again should be conducted among the experts to reach some conclusion about the consequent of a rule. Mendel [27] has discussed in details how this type of survey can be formulated. To reflect the result of the

survey at the output of fuzzy logic systems, he has proposed three possibilities-

a. Keep the response chosen by the largest number of experts.

b. Find a weighted average of rule consequents for each rule

c. Preserve the distributions of the expert responses for each rule

Mendel has chosen solution (b) as the most appropriate one and derived the defuzzification method which accomplishes this task.

If we want to formulate a fuzzy logic system to build quality models, then the consequent is the external attribute. And the fuzzy rules are the linguistic relationships between the internal and external attributes. As experts have different viewpoint about the impact of a group of internal attributes on a particular external attribute i.e. consequent, this is the linguistic relationship uncertainty. We can conduct a survey among the experts about the fuzzy rules to solve this problem. Let us consider a survey among 10 experts on the relationship between the internal attributes coupling and cohesion and the external attribute reliability. Table 3 shows an example of such survey.

Table 3: Survey of experts' opinion on linguistic relationship

IF	Reliability is Low	Reliability is Medium	Reliability is High
Coupling is High and Cohesion is High	6	3	1
Coupling is low and Cohesion is medium	4	2	4

4.1.3 Uncertainty about the measurements that activate the FLS

We have seen in Section 3.5, how non-singleton fuzzification helps us to deal with input noise. This input noise is the uncertainty about the measurements that we use to activate the FLS. When the noise is stationary, we can use type-1 non-singleton fuzzification. If the noise is non-stationary, we should use type-2 non-singleton fuzzification.

We have seen in Table 2 that this type of uncertainty is similar to the numerical assessment uncertainty in software quality models. This uncertainty needs some explanation and the concept of noise in fuzzy logic system has a difference with our defined uncertainty. Numerical assessment uncertainty comes from the different inconsistent measures of the same internal attribute. A Master thesis work here in KFUPM has shown that different cohesion measurements to measure class cohesion have real inconsistency among them [1]. Noise in the fuzzy logic system defined by

Mendel has a basic difference with our numerical assessment uncertainty. Noise in general sense comes from a noisy sensor i.e. if the same thing is measured using a noisy sensor for more than once, then the sensor will give different measurements at different times. If the metrics are our sensors, then there is no noise because software metric always will give the same numerical value. Our uncertainty concept lies in the existence of different metrics for the same internal attribute. Still we can use non-singleton fuzzification to solve this problem. Mendel modeled the input noise using interval type-2 Gaussian fuzzy set with uncertain standard deviation. But as in our case, we have no concept of stationary or non-stationary noise; we shall use type-1 non-singleton fuzzification where the mean of the Gaussian membership functions will be the mean of different metric values of a particular internal attribute and the standard deviation will be the standard deviation of these different metric values.

4.1.4 Uncertainty about the data used to tune the parameters of a FLS

Here the data means the training data. Training data can be noisy or uncertain. For example, at each data point we may have more than one value available. This noise should be handled while building the FLS. Type-2 fuzzy sets help us to handle this type of uncertainty. Again for example, if we consider Gaussian membership function, then uncertain mean can

represent this noise in the training data. Usually there are two steps of building a FLS- initializing and training. While initializing, we should define the type-2 fuzzy MF in such a way so that it represents the noise. While training, the training data is used as the measurements to activate the FLS and to adjust the type-2 membership functions. So, we can use non-singleton fuzzification to handle the noise in this stage.

Uncertainty in the training data has different sources in software quality model. The first one is the uncertainty we discussed in the previous section i.e., different metric values for measuring the same attribute. Another is laziness or ignorance i.e., there may be other factors we don't know which affect the relationship. The first one can be dealt with non-singleton fuzzification. While we train the framework, we should use the mean and standard deviation of the internal attribute measurements as input. Type-2 fuzzy set can help to deal with the second issue. We may represent different experts' opinion using type-2 fuzzy set or we can derive the type-2 fuzzy sets from the uncertain numerical data.

4.2 Type-2 fuzzy Logic based Framework to Build Software Quality Model

The core structure of our framework is based on type-2 FLS which has all the components as in Figure 10. Internal and external attributes and their

relationships are the main sources of knowledge in our framework. So, we developed the framework as in Figure 14.

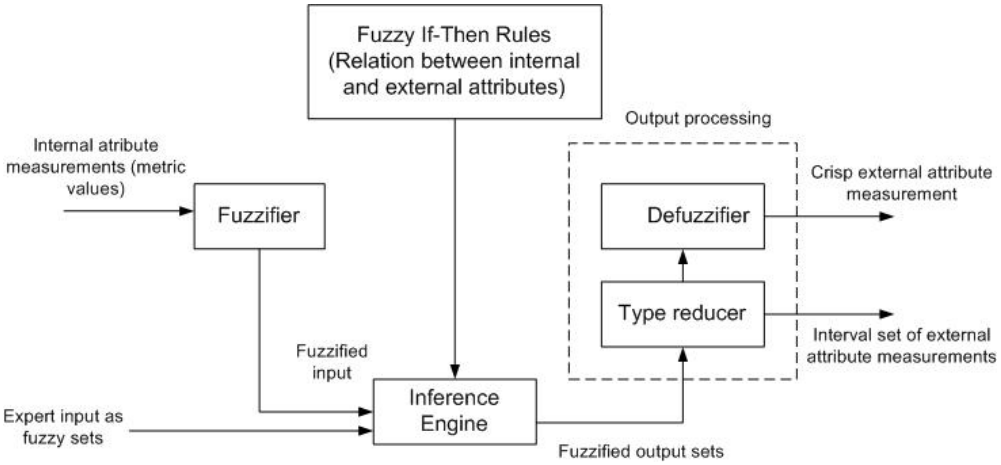

Figure 14: Type-2 FLS based framework to build software quality models

The fuzzifier takes the crisp metrics values as input. These crisp metric values are the different measurements of the internal attributes. We shall see later how these different values can be used as input. The output of fuzzifier is the fuzzified measurements which will be the input to the inference engine. Expert assessment of a software artifact in a form of fuzzy set also can be input to the inference engine. The fuzzy rules are also input to the inference engine. In our framework, fuzzy rules are the relationship between internal and external attributes. The resultant of the inference engine is type-2 fuzzy output sets which can be reduced to type-1 fuzzy set by the type reducer. This type reduced fuzzy set in our framework is an interval set which gives the predicted external attribute measurement as a possible range of values.

The defuzzifier calculates the average of this interval set to produce the predicted crisp external attribute measurement.

Developing such framework usually has three main steps. We shall also define developing process of our framework in these three steps. The steps are

i) Initializing the Framework

ii) Training the Framework

iii) Using the Framework

We shall discuss these three steps in details in the following three subsections.

4.2.1 Initializing the Framework

We have discussed type-2 fuzzy logic system in the previous chapter. We know about the components of a typical fuzzy logic system. Now to initialize our framework, we need to define those components from the perspective of software quality models. Initializing a FLS means initialization of its antecedents, consequents and the fuzzy rules. These components of a fuzzy logic system can be initialized either from the numerical dataset or from the expert opinion. Before delving more into this, let us see what will be the antecedents and consequents in our framework.

- Internal attributes are the antecedents
- External attribute is the consequent

Our framework will support one external attribute to be assessed or predicted based on several internal attributes. If-Then rules will form the rule base using these internal and external attributes.

First let us look at the issue of initialization from numerical data. We expect that we shall have one or more measurements available in the dataset for each internal or external attribute. Our framework will define the initial fuzzy sets for both antecedents and the consequent from this dataset. To use the FLS developed by Mendel [27], we consider our antecedent and consequent membership functions to be type-2 Gaussian with uncertain mean and the input membership functions will be type-2 Gaussian with uncertain standard deviation. Let us suppose that we need to initialize F fuzzy sets for the attribute A. Each attribute has m measurements. In the training dataset, we have attribute measurements for n software modules. Table 4 shows the structure of the training dataset for one attribute.

Table 4: Training Dataset for one Attribute

Module No	Measurement 1	Measurement 2	… …	Measurement n	Mean of Measurements of each Module	Standard Deviation of Measurements of each Module
1	A_{11}	A_{12}		A_{1m}	μ_1	σ_1
2	A_{21}	A_{22}		A_{2m}	μ_2	σ_2
. .	. .	. .	. .	. .	. .	. .
n	A_{n1}	A_{n2}		A_{nm}	μ_n	σ_n

Now, we have to calculate the followings-

M1 = Minimum ($\mu_1, \mu_2 \ldots \mu_n$)

M2 = Maximum ($\mu_1, \mu_2 \ldots \mu_n$)

M = Mean ($\mu_1, \mu_2 \ldots \mu_n$)

S = Standard Deviation ($\mu_1, \mu_2 \ldots \mu_n$)

R1 = Minimum ($\sigma_1, \sigma_2 \ldots \sigma_n$)

R2 = Maximum ($\sigma_1, \sigma_2 \ldots \sigma_n$)

$R = \text{Mean}(\sigma_1, \sigma_2 \ldots \sigma_n)$

$T = (M2-M1)/(F-1)$

Now if U_{i1} and U_{i2} are the uncertain means for i-th fuzzy set, then define the means and standard deviations of F fuzzy sets as follows-

$U_{i1} = M1 + (i-1)T - 0.5*R$

$U_{i2} = M1 + (i-1)T + 0.5*R$

Here $i = 1 \ldots F$

Standard Deviation of all the fuzzy sets = K*S

Here K is a positive constant. K should be chosen such that the membership functions cover the whole universe of discourse.

Expert comments can be used to initialize all these parameters. Whenever we shall have more than one expert to define a fuzzy set, we can use the approach discussed in Section 4.1.1.

The rule base can be initialized by an expert or considering all the possible combinations of fuzzy sets of internal attributes. If we have X internal attributes and F fuzzy sets for each internal attribute, then number of rules can be maximum F^X if in each rule we use all the internal attributes as antecedents.

4.2.2 Training the Framework

After initializing the FLS, our framework supports an optional step – training. If someone wants to rely totally on the expert comment to build the quality model, he does not need any training. But if it is needed to use the historical data, then training is the second step in our framework. The historical dataset will contain the measurements of different internal and external attributes. We have discussed in the previous section how this data should be organized and used to initialize the FLS. The same dataset will be used as training data. It will contain the input-output pair where the inputs are the internal attribute measurements and the output is the external attribute measurement. If we have more than one measurement for any attribute, then we shall use the mean of those measurements i.e. from Table 4 we shall use μ_i as the input or output measurement for a particular attribute *i* of a software module. μ_i will be the mean of non-singleton input type-1 Gaussian membership function. The standard deviation of the measurements i.e. σ_i will be the standard deviation of the non-singleton input type-1 Gaussian membership function. Following is the training algorithm.

- Given N input-output training samples *(x(t):y(t)), t=1,...,N*
- Objective is to minimize the error function for E training Epochs
  - E(t) = {[f(x(t)) – y(t)]/y(t)}^2, t=1,…N

- The steps
  - Initialize all the parameters (as we discussed before)
  - Set the counter, e, of the training epoch to zero i.e. e=0
  - Set the counter, t, of the training data to one. i.e., t=1
  - Apply the means of the internal attribute measurements with their corresponding standard deviation to the type-1 non-singleton type-2 FLS (see Chapter 11 of Mendel's book). Mendel has used same standard deviation for all the input MF. But in our framework, we used different standard deviation for each input.
  - While defuzzification, use average the average response from the survey for the consequents (see section 10.12 of Mendel's book)
  - Tune the uncertain means of the antecedent membership functions and the consequents also using steepest descent algorithm for the error function (see chapter 11 of Mendel's book). Don't tune the input standard deviations.
  - Set t=t+1. If t = N+1, go to next step otherwise apply the next input

- Set e=e+1. If e=E, Stop; otherwise start a new epoch

4.2.3 Using the Framework

Using our framework is straightforward. If we want to use numerical data as input, we shall use the mean and standard deviation of the existing measurements for each internal attribute where this mean and standard deviation will be the mean and standard deviation of input non-singleton type-1 fuzzy set. If we use expert comment as the input, then the expert comment should be given as input to the system in the form of type-2 Gaussian fuzzy set with uncertain standard deviation.

4.3 Experimental Design and Validation

Validation is a very important requirement to show that any newly proposed framework really works. We also tried to validate our framework. For validating the framework, we need to conduct experiments. In this section we shall discuss how we design our experiment to validate the framework.

We want to prove mainly two things from the validation. The first one is that our framework's training works fine i.e., with numerical dataset, the framework can train well to achieve a lower value of error function. The second thing is to show that the framework gives better performance than the other existing approaches. The second objective of validation is often

difficult to achieve due to the insufficiency of available data. Still, we can do some sort of comparisons using artificial dataset. The first objective can easily be achieved, because artificial dataset can be used to train the framework. Actually the ultimate purpose of the framework is function approximation i.e. approximate the functional relation between internal and external attributes. So, if we generate artificial dataset, then still there will be some sort of functional relation. So, if we can validate our framework using this artificial dataset, it is expected that while actual dataset is provided, it will also work fine.

For training the framework, we need uncertain numerical measurements of internal attributes and the measurement of the external attribute. We have such data from [1]. But unfortunately this dataset does not have any external attribute measurements. So we used artificial external attributes. To compare our framework with other approaches, we used data from NASA. But this dataset has no uncertain measurements. We discuss the details of these experiments and the results in Chapter 5.

4.4 Validation of the Training Algorithm

Here, we now discuss the validation of the training algorithm of the framework. As we mentioned earlier, the dataset from [1] has uncertain internal attribute measurements. All the measurements available there, are

cohesion metric values. Different cohesion metrics have been calculated for many classes from different object oriented systems. We took 50 classes from this dataset and for each class we considered the metric values of four cohesion metrics. Those are CCM, TCC, LCC, CAMC. We randomly generated the one external attribute value for each class. Then we applied our framework on this dataset and trained with different step size. Step size is an important parameter of steepest descent algorithm. The step size determines how quickly or slowly the error function is minimized. Figure 15 and Figure 17 show the relative Mean Squared Error (MSE) while training with step sizes 0.01 and 0.1 respectively. It also shows the testing MSE while we test with CAMC metric after training with uncertain data. It is very much evident from these figures that lower step size makes the learning procedure slow, whereas higher step size makes it faster. Figure 15 needs some more explanation regarding the ups and downs towards the end of learning epochs. This may occur if the error function has some local minima or maxima and in that case small step size may be caught in those regions. The example of such error function is shown in Figure 16. Another important conclusion we may draw from the figures that the uncertainty is properly handled through our framework because when we test with one of the cohesion metrics, the error function is also minimized. To check the same result with other metrics we tested with LCC, TCC and CCM also. Figure 18, Figure 19 and Figure 20

shows the test result with the cohesion metrics LCC, TCC and CCM respectively. All these figures agree with our conclusion that the framework is learning the uncertainty in the dataset properly. In all cases, while testing with a single cohesion metric, the testing curve has same pattern as the training curve.

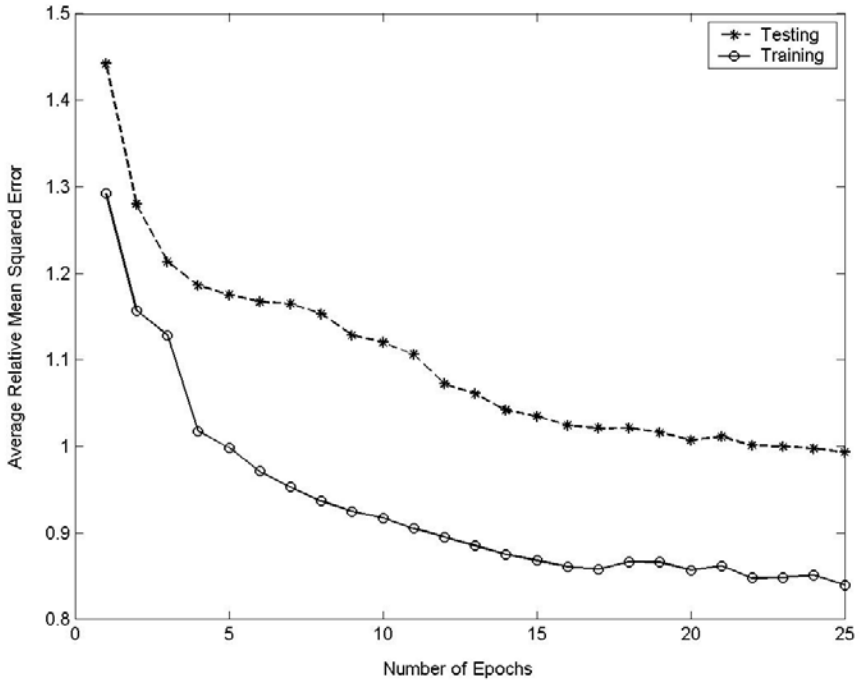

Figure 15: Training with means (Type-2) and testing with CAMC (step size = 0.01)

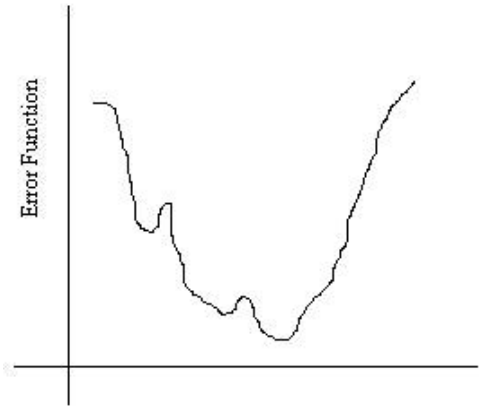

Figure 16: Error Function

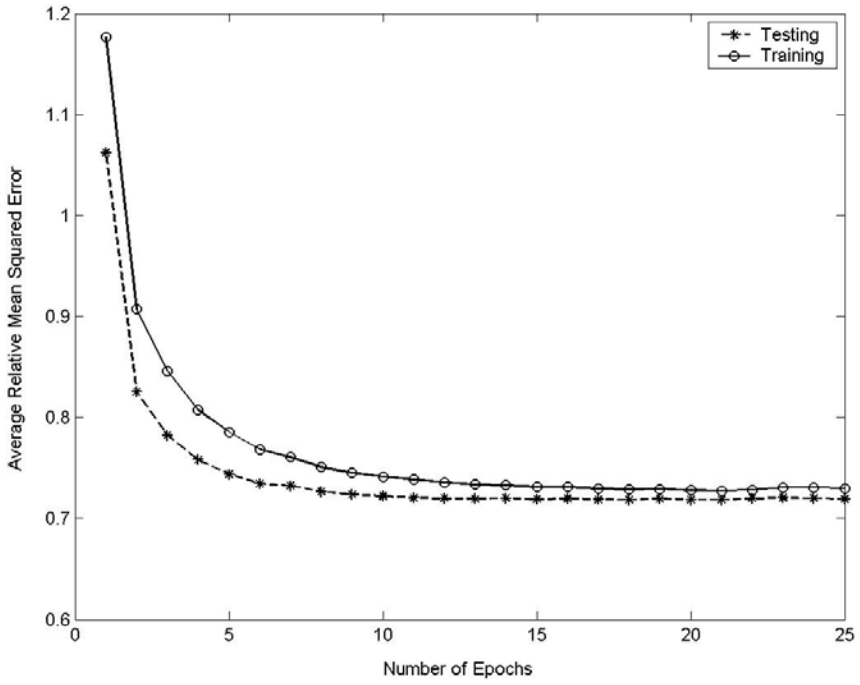

Figure 17: Training with means (Type-2) and testing with CAMC (step size = 0.1)

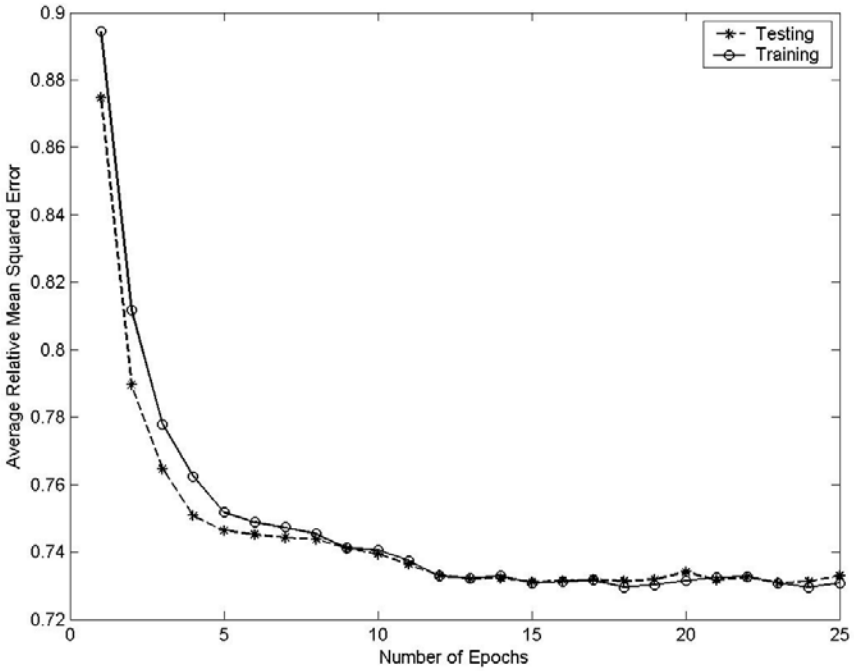

Figure 18: Training with means (Type-2) and testing with LCC (step size=0.1)

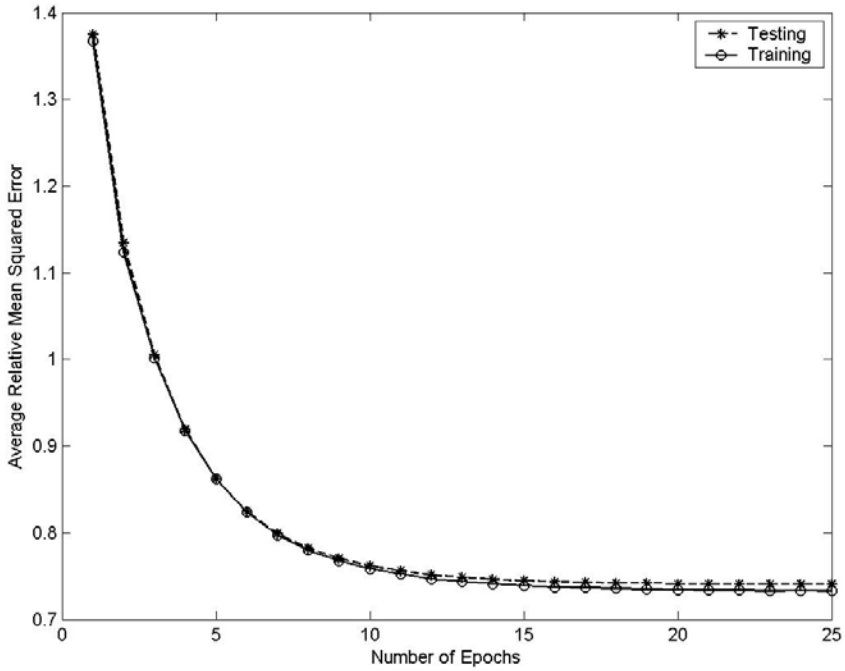

Figure 19: Training with means (Type-2) and testing with TCC (step size=0.1)

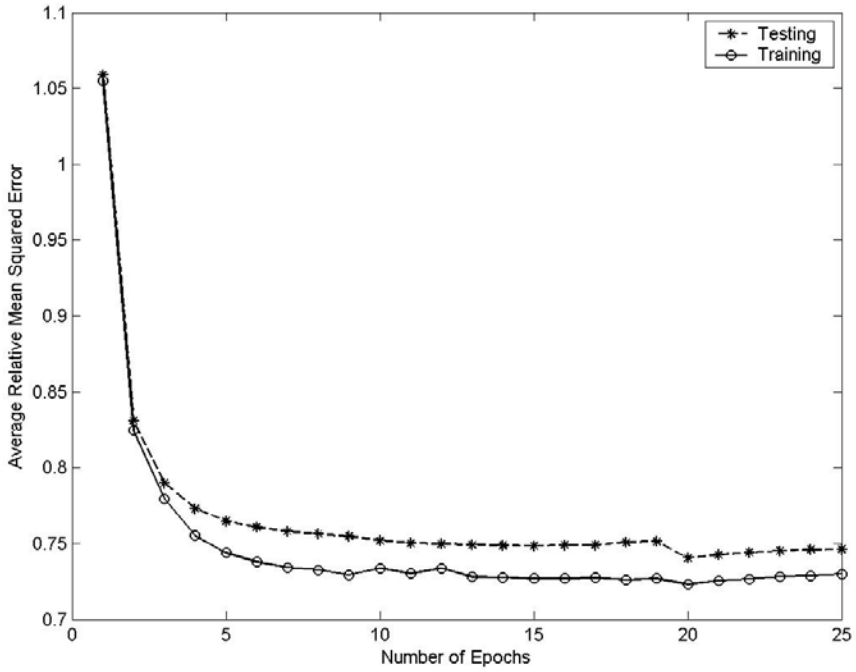

Figure 20: Training with means (Type-2) and testing with CCM (step size=0.1)

Chapter 5

Experimental Results

We implemented our proposed framework to build software fault prediction model. As we saw in Chapter 2, people have widely used software fault prediction models to verify their model building approach. There is another reason behind our choice of fault prediction. That is the availability of data. To get historical data in public domain is not very easy and for the experiments data should come from reliable source. We got some data which helped us to build fault prediction models and to compare the performance of our framework with that of other existing techniques. We have divided this chapter in three sections based on different types of available data. Two sources of data are used. One is NASA IV&V MDP [30] another is from the Giovanni Denaro, author of the paper on software fault proneness models [8].

5.1 Experiments with NASA Dataset

NASA dataset has few projects developed in C and one project developed in C++. For all the projects, there are several metrics data calculated from different modules of the projects. The very much useful aspect of this dataset is that the number of defects or faults detected for each module has been

stored in the dataset. For the OO project i.e., project developed is C++ we have considered each class as a module. As there is only one such project, we have divided the classes into two sets randomly. One set was used as the training set and another for testing. For the procedural projects i.e., projects developed in C, we could choose training set and test set from different projects as we have more than one project. We did use only numerical data for training and testing.

We implemented type-1 and type-2 fuzzy logic based models and compared their performances. This comparison is important because; although both type of fuzzy logics have the ability to take expert comments into consideration, uncertainty is the main issue that should create a difference between the performances. Actually in NASA data, we cannot apparently see any type of uncertainty except only uncertainty with numerical relationship. So, we expect type-2 FLS to perform better if we could have data that associated all types of uncertainties. We also tried to compare the performance with that of regression based model. But it is difficult to reach some conclusion with this sort of comparison. Because, the underlying shortcoming of the regression based models is the one we discussed in Chapter 1. Regression based models are not universal function approximator. So, we need to define the type of the model and there could be numerous types. In our experiments we used linear and polynomial statistical

regression models to compare their performance. Actually this comparison between fuzzy logic based model and regression based models is just for experimental purpose, because theoretically also it is evident that regression based models can not handle imprecision and uncertainty.

For all experiments, we used the type-2 fuzzy logic system developed by Mendel [27]. We used his source code also which is provided for free to use. He used Gaussian membership functions with uncertain mean for antecedents and consequents and Gaussian membership functions with uncertain standard deviation for inputs. We also used type-1 fuzzy logic system developed by Mendel to build the fault prediction model. We used the same set of rules to build fault prediction model using type-1 and type-2 fuzzy logic systems. We used mean squared error as the performance indicator of the fault prediction models. While developing this type of adaptive models, another issue plays a vital role in the results i.e., the step size. Higher step size may lead to worse performance rather than converging. On the other hand very low step size may lead to a slow convergence. So, we did experiments with different step sizes. As it is not our main objective to find out a suitable step size, we just did experiment with several step sizes and show some of the results that may help us to explain the performance of the framework. Suitable step size really differs in different problems and

experiments with different values will be required to find the most suitable one.

5.1.1 Results from OO Dataset

The object oriented project has several metrics data in the dataset. We used CBO and LCOM metrics to build our fault prediction model. In the model, CBO and LCOM are the antecedents and number of faults is the consequent. For each antecedent, we considered 2 membership functions. So we have total 4 rules. For each rule the consequent membership functions are different and random. So we have 4 consequent membership functions. The rules look like as given below.

If CBO is MF^1_{CBO} and LCOM is MF^1_{LCOM} then Number of Faults is MF^1_{NOF}

If CBO is MF^1_{CBO} and LCOM is MF^2_{LCOM} then Number of Faults is MF^2_{NOF}

If CBO is MF^2_{CBO} and LCOM is MF^1_{LCOM} then Number of Faults is MF^3_{NOF}

If CBO is MF^2_{CBO} and LCOM is MF^2_{LCOM} then Number of Faults is MF^4_{NOF}

In these rules MF^i_v represents the *i- th* membership function for variable *v*.

In the object oriented project, we have total 144 classes. We chose randomly 75 classes for training and rest 69 classes for testing. As the consequent fuzzy set is random, we conducted the experiment with training dataset 15 times and plotted the average and standard deviation of the relative Mean

Squared Error after testing. This gives us more confidence in the result. Figure 21 and Figure 22 shows average and standard deviation respectively. The result doesn't look to be promising for type-2 fuzzy logic based framework. Type-1 fuzzy logic is performing better. To have an explanation on this issue, we conducted the same experiment with step size 0.2. With step size 0.01, our framework is converging still while Type-1 has already converged. After some more epochs, it is expected to converge totally.

Figure 23 and Figure 24 shows the average and standard deviation of MSE with step size 0.2. It is evident from these two figures that with high step size, after few epochs Type-2 FLS based framework converges to the same level of Type-1. This happened mainly because we had no uncertainty in the training or testing dataset. Type-2 FLS considers that uncertainty is there, but if the dataset is certain, then after few epochs it starts performing same as Type-1. If we could have data with uncertainty i.e., each attribute with several measurements, then it was possible to show the superiority of Type-2 FLS over Type-1.

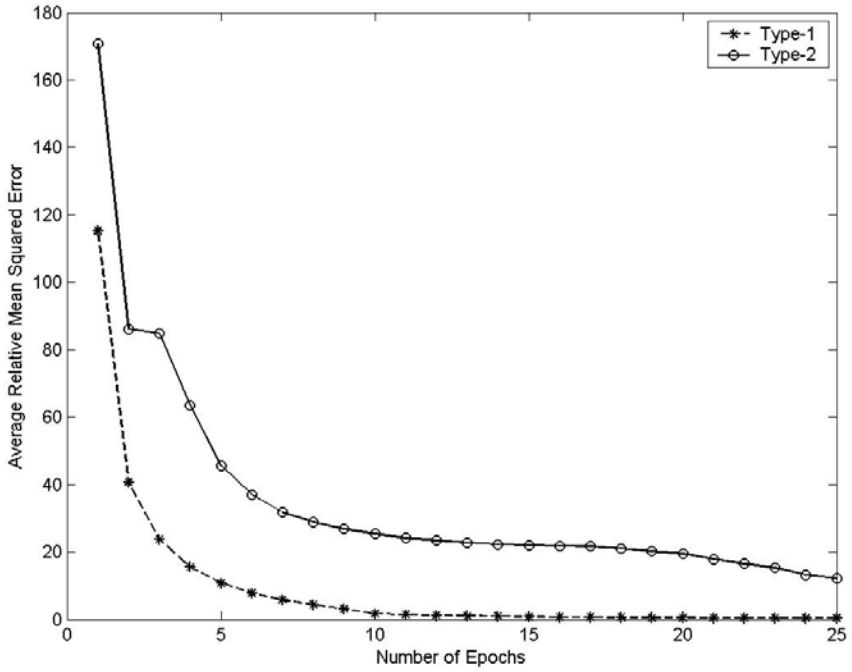

Figure 21: Average Testing MSE of OO project for 15 experiments with step size 0.01

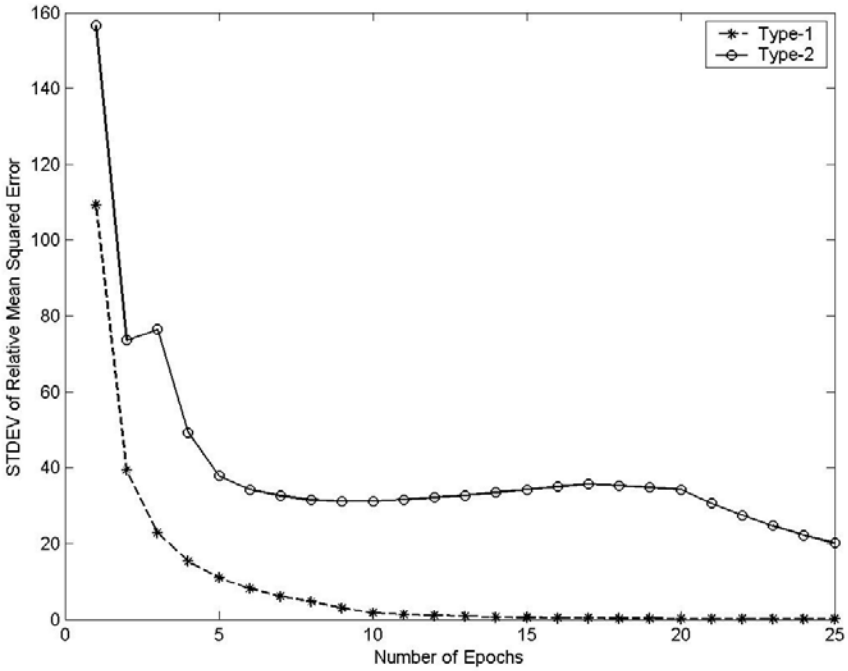

Figure 22: Standard Deviation of Testing MSE of OO project for 15 experiments with step size 0.01

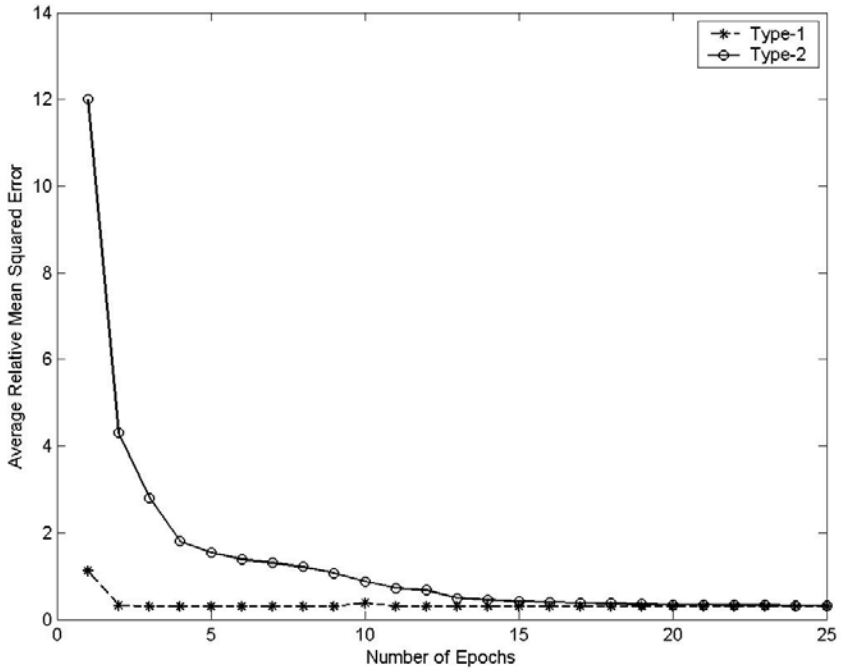

Figure 23: Average Testing MSE of OO project for 15 experiments with step size 0.2

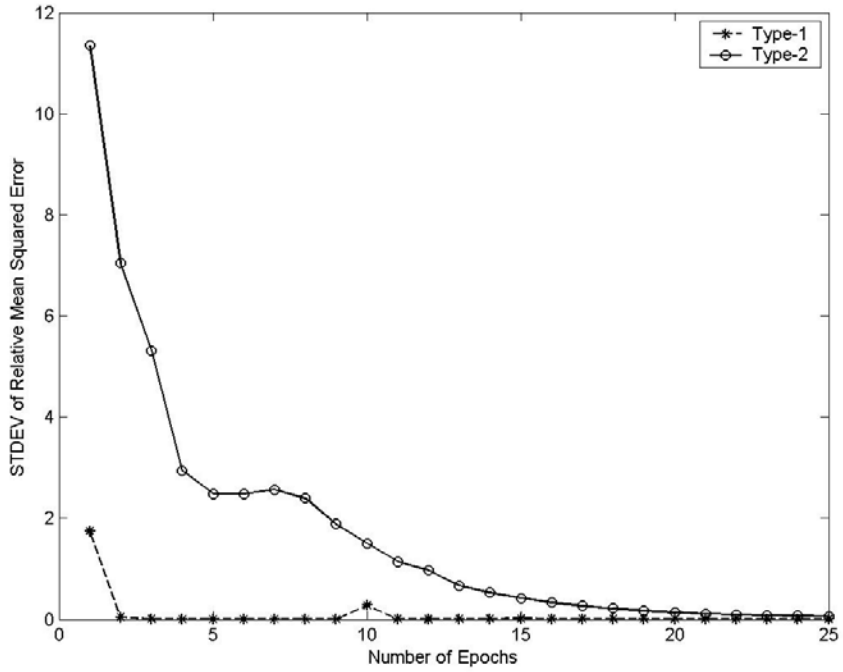

Figure 24: Standard Deviation of Testing MSE of OO project for 15 experiments with step size 0.2

We see the same type of graphs with test data also. Our explanation regarding the relation between the step size and the uncertainty is more evident from Figure 23. All the figure of standard deviation also gives us confidence that at least at higher step size, type-2 has better standard deviation.

We built statistical regression based model also with the same training and testing dataset. We used both linear and non-linear regression. For non linear we used 2^{nd} degree polynomial regression. We calculated the MSE for both training and testing dataset. While comparing the MSE of regression based models with that of FLS based models, we considered the minimum that we could get from FLS based model irrespective of the step size. Figure 25 shows the comparison for the testing dataset. The result looks inspiring for the Type-2 FLS based framework. Some may argue that we just experimented with two types of regression based models. Actually this is the strength of FLS based models over regression based model. Fuzzy logic systems are universal function approximator and we discussed this issue earlier in this chapter.

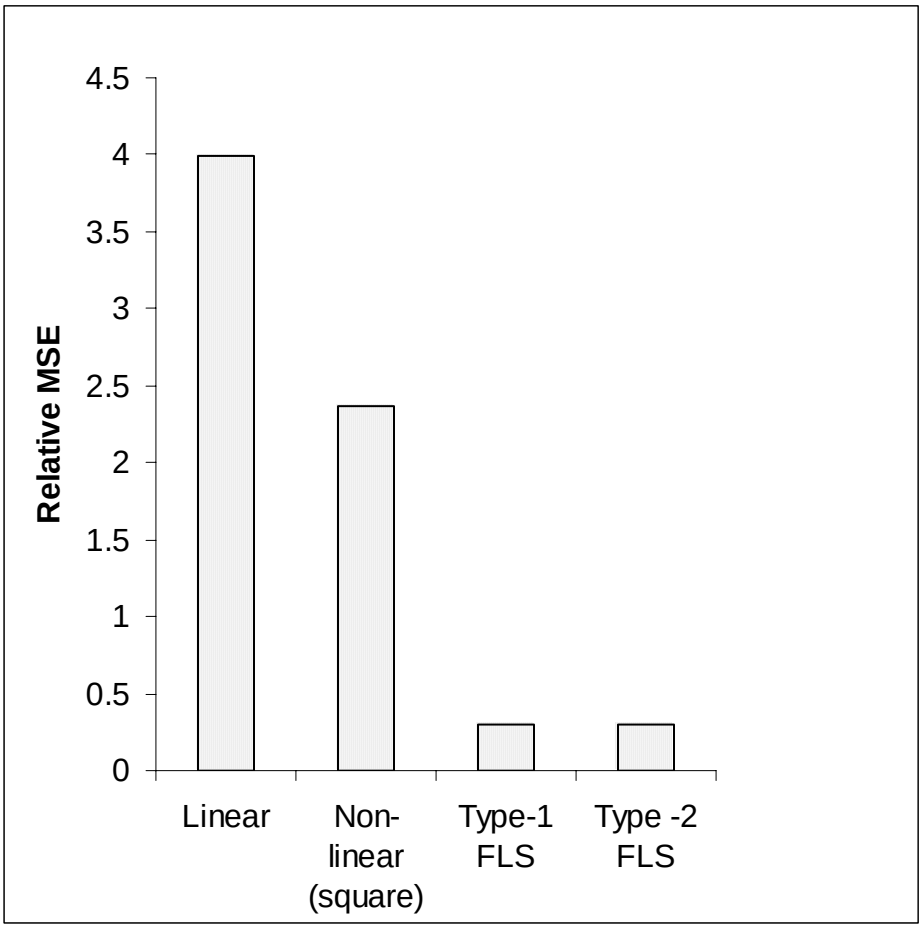

Figure 25: Comparison of Regression based models with FLS based models for OO Testing Data

5.1.2 Results from Procedural Dataset

As we have more than one project written in C, we could choose our training and testing dataset from two different projects. The training and testing both projects have more than 1000 modules. But most of them have zero faults. So, we considered to choose only those modules which have at least 1 fault. Now our training set contains 81 modules and test set contains 73 modules. Halstead Error Estimate (HEE) and Cyclomatic Complexity (CC) are two metrics that we used as our internal attribute measurements. So, in our model HEE and CC are antecedents and Number of Faults (NOF) is the consequent. For antecedents, we considered two membership functions and for each consequent we considered different and random membership function. Here are our rules.

If HEE is MF^1_{HEE} and CC is MF^1_{CC} then Number of Faults is MF^1_{NOF}

If HEE is MF^1_{HEE} and CC is MF^2_{CC} then Number of Faults is MF^2_{NOF}

If HEE is MF^2_{HEE} and CC is MF^1_{CC} then Number of Faults is MF^3_{NOF}

If HEE is MF^2_{HEE} and CC is MF^2_{CC} then Number of Faults is MF^4_{NOF}

In these rules MF^i_v represents the *i- th* membership function for variable *v*.

We have conducted the same sets of experiments with step sizes 0.01 and 0.1 as we did for OO dataset. We shall present the results of average and

standard deviation of 15 experiments for both training and testing. Figure 26 and Figure 27 shows the average and standard deviation of MSE with training dataset. The step size is 0.01 here. We see here that with step size 0.01, after 25 epochs, Type-2 converged at the same level with Type-2. We did not do more experiments with high step size because with step size 0.01 we could get good performance for both Type-1 and Type-2. Here the data has same shortcoming as the OO data had. We have no uncertainty and thats why after some epochs Type-2 FLS should perform as good as Type-1 FLS.

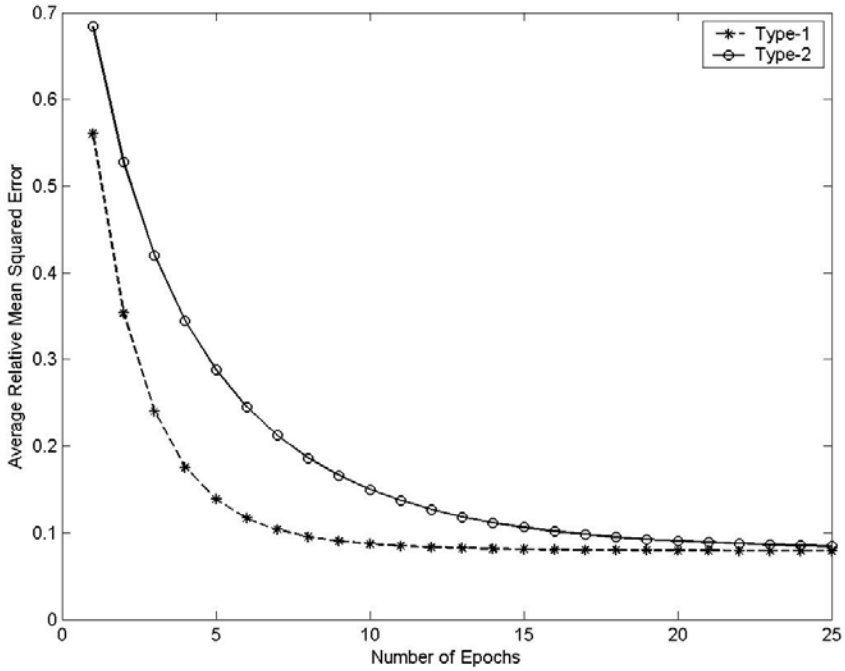

Figure 26: Average Training MSE of procedural project for 15 experiments with step size 0.01

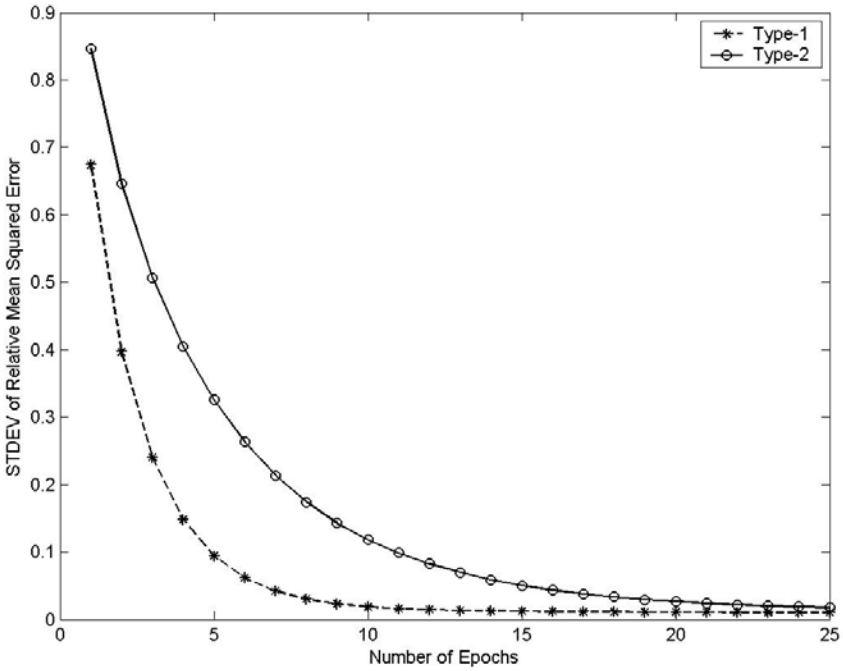

Figure 27: Standard Deviation of Training MSE of procedural project for 15 experiments with step size 0.01

We also conducted some experiments on this procedural dataset to compare our results with the results of regression based models. We did it only for test dataset.

Figure 28 shows this comparison. FLS based models outperformed regression based models. The explanation of this result is again the same as we explained in the case of OO dataset. FLS based models are universal function approximator but regression based models are not.

None of the above experiments with OO and procedural dataset uses expert comments. We could not use it due to the lack of available data from the experts. We hope to get better performance with our framework if we could use different expert judgments.

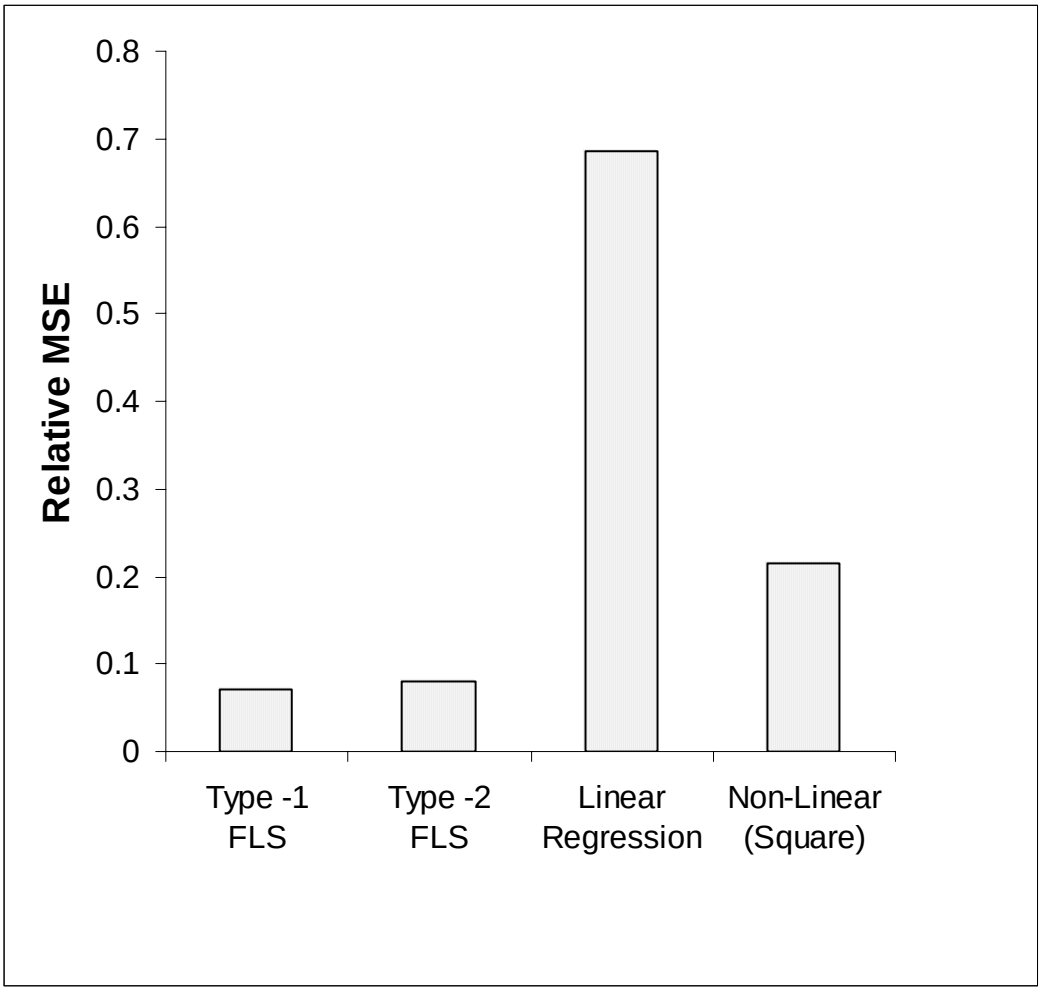

Figure 28: Comparison of Regression based models with FLS based models for Procedural Testing Data

5.2 Experiments with Apache Dataset

All the experimental results we showed in Section 5.1, were solely based on our own experiments. As we could not find any published result for the NASA dataset, we are not able to compare our work with others using this dataset. Fortunately we got the dataset used by the authors of [8]. The authors have extracted their relevant data fields from the public data of Apache web server. They used Data from Apache 1.3 as the training set and Apache 2.0 as the test set. Every C file has been considered as a module. Authors have calculated some procedural metrics for each module and extracted number of faults from the CVS repository. They have chosen different fault prediction models among all possible models based on some criteria. The models are built using multivariate regression analysis. The models differ in the use of metrics as independent variables. Different models use different subset of metrics. For each model, they have predicted the fault proneness as output from the test dataset. Then the modules of test dataset are sorted in descending order based on the predicted output. The percentage of modules from this sorted order have been calculated which are accountable for different percentages of known faults e.g., from the sorted order first *x%* modules are responsible for *y%* actual faults. If we want to compare two models, for the same value of *y*, the model which produces

smaller value of *x*, is said to be better to predict *y%* of actual faults. In the paper [8], the authors have shown the results of 8 different models. We have built our type-2 fuzzy logic based model using the metrics that the authors used to build the direct model with best overall completeness. Figure 29 shows the comparison between our model and the regression based model from the paper. It is evident that our type-2 FLS based model has a consistent better performance.

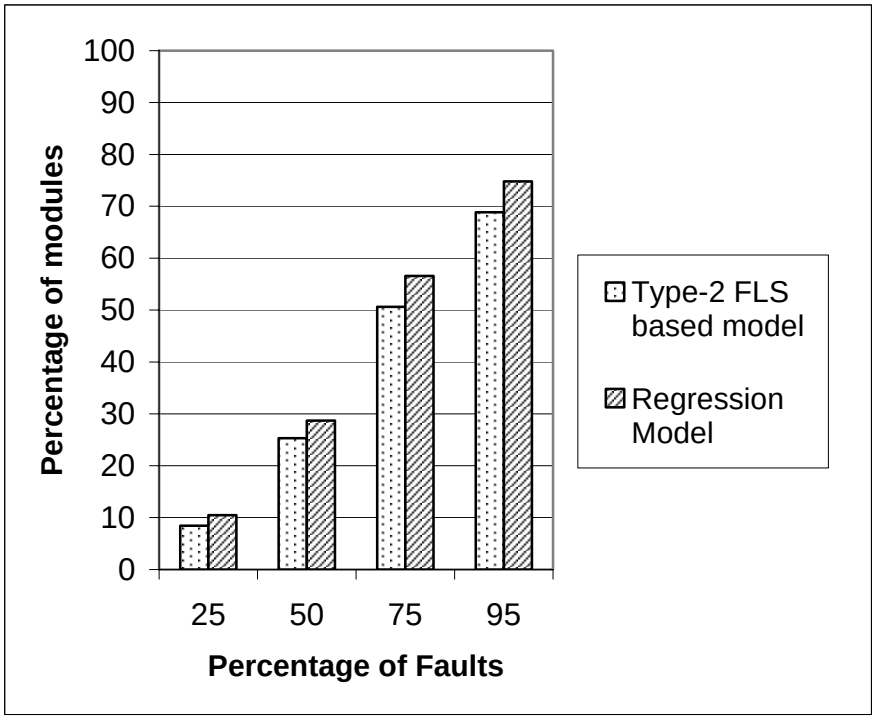

Figure 29: Comparison with the results from the paper [8]

Chapter 6

Conclusion

In this chapter, we summarize our contributions and limitations and conclude the thesis with some future work directions.

6.1 Contributions

Our investigation, research and experiments conclude the following contributions of this thesis.

i) Sources of imprecision and four different types of uncertainties have been defined in software quality models

ii) A literature survey has been done in the domain of software quality models

iii) Type-2 fuzzy logic has been investigated as a solution to the uncertainty problems

iv) It has been shown that there is a nice mapping between the uncertainties discussed by Mendel [29] and our definitions of uncertainty in software quality models.

v) Type-2 fuzzy logic based general framework has been built to handle imprecision and uncertainty in software quality models and different steps of this framework has been discussed in details. Conceptually it has been shown to deal with all types of uncertainties.

vi) Validation of the framework has been done using some artificial dataset.

vii) Software fault prediction models have been built as an instance of the framework. Experiments have been done using this model to compare type-2 fuzzy logic based framework with type-1 FLS and regression based approaches.

6.2 Limitations

i) We could not conclude the performance of our framework fully because of the lack of sufficient data. The data we used had no uncertainty in the attribute measurements and we did not try to build the model based on expert comments. These are the two limitations of this work. Still, most of the results look to be promising. Appropriate justification has been given to explain the results.

ii) Mendel's type-2 FLS has some limitations and we built our framework based on these limitations. For example, FLS can only deal with interval type-2 Gaussian membership functions. It specially imposes the limitation while combining the experts opinion to deal with linguistic assessment uncertainty.

6.3 Future Work

Here we point to some the possible future works in the same research direction of this thesis

i. More experiments can be done if data is available with uncertain numerical attribute measurements

ii. The framework can be validated using expert data

iii. Type-2 FLS developed by Mendel still has some limitations which have been discussed in the previous sections. Some future work can be directed towards this direction to make the framework more robust

iv. Software fault prediction models have been built in this work for experiment. Other models like cost estimation model may be built with this framework.

v. We mainly focused on the framework, not on the attributes and their relationship. This framework opens the door to work with different software attributes to explore their relationship and thus come up with different quality models.

References

[1] Abubakar A., *Implementation and validation of object-oriented design level cohesion metrics*, Master thesis in Computer Science, King Fahd University of Petroleum and Minerals, Saudi Arabia, 2005.

[2] Ahmed M., Saliu, M., and AlGhamdi J., "Adaptive Fuzzy Logic Based Framework for Software Development Effort Prediction," *Journal of Information and Software Technology (IST)*, Vol 47/1 pp 31-48, January, 2005.

[3] Basilli V, Briand L., Melo W., "A Validation of ObjectOriented Design Metrics as Quality Indicators", *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, October 1996

[4] Boehm, B.W., Brown, J.R., Kaspar, J.R. *et al.*, "Characteristics of Software Quality", *TRW Series of Software Technology*, Amsterdam, North Holland, 1978

[5] Briand L., et al., "Exploring the relationship between design measures and software quality in object oriented systems", *The Journal of Systems and Software* 51, pp 245-273, 2000.

[6] Briand, L., Melo, W., And Wust, J. "Assessing the applicability of fault-proneness models across object-oriented software projects". *IEEE Transactions on Software Engineering* vol. 28, 7 (July 2002).

[7] Chidamber S., Darcy D., Kemerer C., "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis", *IEEE Transactions on Software Engineering,* Vol.24, No. 8, August 1998

[8] Denaro, G., And Pezze, M. "An empirical evaluation of fault-proneness models". In *Proc. Int'l. Conf. Software Engineering* (Florida, USA, May 2002).

[9] Ebert, C. "Experiences with criticality predictions in software development". *ACM SIGSOFT Software Engineering Notes* 22, 6 (Nov. 1997).

[10] Ebert, C. "Rule based fuzzy classification for software quality control". *Fuzzy Sets and Systems* 63 (1993).

[11] Fenton, N., And Nell, M. "Software metrics and risk". In *Proc. 2nd European Software Measurement Conf*. (Oct. 1999).

[12] Fenton, N., And Nell, M. "Software metrics: Roadmap". In *Proc. Conf. Of Software Engineering* (May 2000).

[13] Fenton, N.E., Pfleeger, S.L., *Software Metrics: A rigorous and practical approach*, PWS Publishing Company, Boston, USA, 1997

[14] http://www.comp.nus.edu.sg/pris/FuzzyLogic/DescriptionDetailed2.html

[15] ISO/SEC 14598 International standard, *Standard for Information technology- Software product evaluation - Part 1: General overview*.

[16] Kanmani S., *et al.*, "Object Oriented Software Quality Prediction Using General Regression Neural Networks", *ACM SIGSOFT Engineering Notes*, Vol. 29, No. 5, September 2004

[17] Karnik N.N., *An Introduction to Type-2 Fuzzy Logic Systems*, Univ. of Southern California, LA, CA, June 1998; http://sipi.usc.edu/~mendel/report

[18] Khoshgoftaar T. M, and R.M.Szabo, "Detecting Program Modules with Low Testability", *Proceedings of the International Conference on Software Maintenance*, pp. 242-250, 1995.

[19] Khoshgoftaar T. M, and R.M.Szabo, "Improving Neural Network Predictions of Software Quality using Principal Components Analysis", *Proceedings of IEEE International World Congress on Computational Intelligence*, pp. 3295- 3300, 1994.

[20] Khoshgoftaar T. M., A.S.Pandya and H.M More, "A Neural Network Approach for Predicting Software Development Faults", *Proceedings of 3rd IEEE ISSRE*, pp. 83-89, 1992.

[21] Khoshgoftaar T. M., E.B.Allen, and Z.Xu, "Predicting Testability of Program Modules using Neural Networks", *Proceedings of 3rd IEEE Symposium on Applied Specific Systems and Software Engineering Techniques*, pp. 57-62, March, 2000.

[22] Khoshgoftaar T. M., E.B.Allen, J.P.Hidepohl, and S.J.Aud, "Application of Neural Networks to Software Quality Modeling of a very Large-Scale Telecommunications Systems", *IEEE Transactions on Neural Networks*, vol 8, no 4, July pp. 902-909, 1997

[23] Khoshgoftaar T. M., R.M.Szabo and P.J.Guasti, "Exploring the Behavior of Neural Network Software Quality Models", *Software Engineering Journal* May, pp. 89-96, 1995.

[24] Khoshgoftaar, T., Lanning, D., and Pandya, A. "A comparative study of pattern recognition techniques for quality evaluation of telecommunication software". *IEEE Journal on Selected Areas of Communication* (1994).

[25] Lanubile, F., Lonigro, A., And Visaggio, G. "Comparing models for identifying fault-prone software components". *In Proc. Int'l Conf. Software Eng. and Knowledge Eng*. (USA, June 1995).

[26] McCall, J.A., Richards, P.A., and Walters, G.F., *Factors in Software Quality*, RADC TR-77-369, 1977. Vols I, II, III, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977.

[27] Mendel M., *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions,* Prentice Hall Inc., Upper Saddle River, NJ 07458, 2000

[28] Mendel, J.M. and Q. Liang, "Pictorial Comparison of Type-1 and Type-2 Fuzzy Logic Systems", in *Proc. IASTED Int'l Conference on Intelligent Systems & Control*, Santa Barbara, CA, Oct., 1999.

[29] Mendel, J.M., "Fuzzy Sets for Words: A new Beginning", *The IEEE conference on Fuzzy Systems*, 2003

[30] NASA IV&V Metrics Data Program. http://mdp.ivv.nasa.gov/index.html

[31] Quah T. S, and M.M.T.Thewin, "Application of Neural Networks for Software Quality Prediction using Object-Oriented Metrics", *Proceedings of the International Conference on Software Maintenance* (ICSM'03), Vol 3, 2003

[32] Russell, Geddes, And Grosset. *Webster's New Dictionary and Thesauru,*. Geddes and Grosset Ltd, New Lanark, Scotland, 1990.

[33] Russell, S., And Norvig, P. *Artificial Intelligence: A Modern Approach*,1st ed. Prentice Hall Inc., 1995.

[34] So, S., Cha, S., And Kwon, Y. "Empirical evaluation of a fuzzy logic based software quality prediction model". *Fuzzy Sets and Systems* (2002).

[35] Sommerville, Ian, *Software Engineering*, Addison Wesley, 2004

[36] Thewin M. M, and T.S.Quah, "Application of Neural Networks for Predicting Software Development Faults using O-O Design Metrics", *Proceedings of 9th International Conference on Neural Information Processing (ICONIP 2002*), vol5, pp 2312-2316, 2003.

[37] Trendowicz A. and Punter T., "Quality modeling for software product lines"

[38] Wang, L., *Adaptive Fuzzy System and Control: Design and Stability Analysis*, Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632, 1994.

[39] Zadeh, L.A., "The concept of a Linguistic Variable and Its Application to Approximate Reasoning-1", *Information Sciences*, vol. 8, pp. 199-249, 1975

Nomenclature

SQM: Software Quality Model

FLS: Fuzzy Logic System

MF: Membership Function

MSE: Mean Squared Error

LCOM: Lack of Cohesion Measure

CAMC: Cohesion among Methods of Classes

TCC: Tight Class Cohesion

LCC: Loose Class Cohesion

Vita

Quazi Abidur Rahman from Bangladesh has been a Master student in Computer Science in King Fahd University of Petroleum and Minerals (KFUPM) since September 2002. He finished his B.Sc. in Computer Science and Engineering from Khulna University, Bangladesh in 1999. After finishing his Bachelor, he has been in teaching profession in different institutes. Before joining KFUPM, he had been working in the Islamic University of Technology (IUT), Dhaka, Bangladesh as a Lecturer. He has been in vacation from IUT while pursuing his MS at KFUPM. His research interest includes soft computing, empirical software engineering, software metrics and algorithms. He can be reached through his e-mail – qabidn@ccse.kfupm.edu.sa .