

# **f-SEE**

## **Fuzzy Software Estimation Environment**

**Submitted in Partial Fulfillment for the Award of the Degree of  
Bachelor of Technology  
(Computer Science and Engineering)**

**Guru Gobind Singh Indraprastha University  
New Delhi, India  
2006-2007**

## CONTENTS

<u>S. No.</u>	<u>Title</u>	<u>Page No.</u>
1	Motivation	1
2	Software Engineering Practices	2
3	Software Cost Estimation Models	15
4	Function Point and Class Point Measures	26
5	Soft Computing and Fuzzy Logic	38
6	Fuzzy Software Estimation Framework	63
7	Features and Software Considerations for f-SEE	102
8	Future Scope	138
9	References	140

## **CERTIFICATE**

This is to certify that **Aditi Kapoor (0191502703)**, **Tapasya Patki (0731502703)** and **Pratibha Pandey (0141502703)** have carried out their project work entitled “**f-SEE: Fuzzy Software Estimation Environment**” as a partial requirement for the award of Bachelor of Technology Degree in Computer Science and Engineering by Guru Gobind Singh Indraprastha University, New Delhi and is a record of bonafide work carried out and completed under my supervision and guidance during the academic session from **2006-2007**.

**Ms. Sonali Bhatnagar**

(Project Mentor)

**Mr. V. Kumar**

(HoD)

**Department of Computer Science and Engineering**

**Maharaja Surajmal Institute of Technology**

**Guru Gobind Singh Indraprastha University**

**New Delhi**

**2006-2007**

## **ACKNOWLEDGEMENTS**

It is a pleasure to acknowledge the guidance and help we have received at various stages in the development of this project. We are grateful to Mr. V. Kumar, HoD (Computer Science and Engineering), for having given us an opportunity to work on fuzzy logic and software engineering.

Our sincere thanks goes to Ms. Sonali Bhatnagar for believing in us, for providing immense motivation all through, and for explaining software management and fuzzy logic concepts to us. Without her guidance, this project would not have been such a success.

We wish to extend a special thanks to all the faculty members who evaluated our project during the review process and provided us constructive feedback. Their inputs have been vital in shaping the project.

Finally, we wish to thank the laboratory assistants and library staff for rendering excellent infrastructural facilities and for aiding us with the extensive literature references that were required for this project.

An earnest thanks is also due to our colleagues, friends and family members for their support.

**Aditi Kapoor**

**Tapasya Patki**

**Pratibha Pandey**

## **SYNOPSIS**

Software engineering is concerned with the conception, development and verification of a software system. This discipline deals with identifying, defining, implementing and verifying the characteristics of a software product. These characteristics include attributes such as desired functionality, maintainability, testability, ease-of-use, portability, and reliability. Software engineering addresses these characteristics by preparing design and technical specification documents, which when adhered to, will result in software that can be verified to meet these requirements. **Software Estimation** is an important area that falls under the *planning* procedure for any software. For a given set of requirements, it is desirable to know how much it will cost to develop the software and how much time and effort the actual development will take. Estimation of resources, cost, and schedule for a software project requires experience, access to good historical information, and the courage to commit to quantitative predictions when qualitative information is all that exists.

**Cost models** provide direct estimates of effort. These models typically have a primary cost factor such as size and a number of secondary adjustment factors or *cost drivers*. Cost drivers are characteristics of the project, process, products, or resources that influence effort. Boehm derived a cost model called COCOMO (Constructive Cost Model) using data from a large set of projects at TRW, a consulting firm based in California. COCOMO is a relatively straightforward model based on inputs relating to the size of the system and a number of cost drivers that affect productivity. The original COCOMO is a collection of three models: a *Basic* model that is applied early in the project, an *Intermediate* model that is applied after requirements are specified, and an *Advanced* model that is applied after design is complete.

Fuzzy Logic, a sub-discipline of Soft Computing, was conceived by Prof. Lofti Zadeh. It is a form of logic that is tolerant to imprecision, partial truth and uncertainty, and is hence able to capture the linguistic domains of software evaluation and modeling. Software cost estimation deals with the planning and tracking of software projects. This calls for

predictions of the likely amount of effort, time and staffing levels required to build a software system. Existing models for this prediction process rely on accurate estimate of either size of the software in terms of the lines of code, number of user screens etc., or on the functionality or feature based analysis. Algorithmic models such as COCOMO, have failed to present suitable solutions, due to their inability to capture the complex set of relationships (e.g., the influence of each variable in a model on the overall predication made using the model) that are evident in several software development environments. While these existing models can be successful in a certain limited environments, they are not flexible enough to adapt to new environments. These models cannot handle categorical data (i.e. data that is specified over a range of values) and lack the reasoning capabilities. Fuzzy Logic, with its offerings of a powerful linguistic representation can easily incorporate imprecision in inputs and outputs, while providing a more expert-knowledge based approach to model building. We propose a framework for an interactive, integrated estimation environment that we refer to as “f-SEE- Fuzzy Software Estimation Environment”. The platform used is Visual C#.

# **CHAPTER 1**

## **MOTIVATION**

---

A recent report undertaken by National ICT, Australia and IBM Australia, published in IEEE Software, talks about misleading software metrics and unsound estimation analysis procedures. This report clearly highlights that uncertainty and unclear advice in the software planning phase results in the adoption of incorrect data aggregation and analysis techniques in the software cost modeling phase. Existing models have been successful in the pre-Internet and the pre-Offshoring era, where software was not a global phenomenon, and when computers were merely used for automation of simple processes. The last few decades, however, have seen a boom in the software industry, with many Small and Medium Enterprises (SMEs) and Multi National Companies (MNCs) joining the league of software consultancy and software services. These companies build significantly complex software systems with constantly changing requirements. Conventional models are intolerant to uncertainty, partial truth and approximations, thus causing disappointment and collapse of trust amongst the interacting groups. Thus, there is a strong urge to resort to cost estimation models that translate to real-life scenarios and function on linguistic scales, instead of the absolute numeric scales of the conventional models. Soft Computing addresses most of these issues, and Fuzzy Logic being one of the most popular and established disciplines of Soft Computing, has been used to deal with software cost estimation in this work.

## CHAPTER 2

---

### SOFTWARE ENGINEERING PRACTICES

---

#### **2.1 Introduction to Software Engineering**

Computer software is formally defined as a set of programs and procedures required to enable a computer to perform a specific task, as opposed to the physical components of the system (hardware). John W. Tukey first used the term “software” in this sense in 1957. Software has historically been considered an intermediary between electronic hardware and data, which later the hardware processes according to the sequence of instructions defined by the software.

The early decades of the emergence of information technology focused on hardware requirements of a system rather than the software or the set of programs providing functionality to the system. This was because initially the user and the programmer were the same person: the users typically were the scientists and engineers who helped build the hardware. Early programs were generally written in machine or assembly language and involved very little logical complexity. The computer-based systems were used for “number crunching” applications that were scientific, with little emphasis on user-friendliness and sophistication of interface. Gradually, hardware became more powerful and cheaper and the application specific needs were satisfied by software, which was now written in higher-level languages. People who were not hardware engineers increasingly started writing software.

This trend led to the software crisis, and the completion and delivery of software projects became problematic. Software was found to be unreliable, non-maintainable and non-transportable. There were many instances of customer dissatisfaction. The programmers and management professionals started to realize that developing software merely with the programmer’s viewpoint and relying on the belief that such software would “work” was not enough. Rather, software needed to be ‘engineered’. Software Engineering was thus identified as an important discipline. Initial work in this area was put forth by Fritz Bauer in 1969, where he stated that there was a need for ‘the establishment and use of sound engineering principles in order to obtain, economically, software that is reliable and



works efficiently on real machines'. Thus, software no longer meant source code. It now included the inspiration that led to choosing a certain set of data structures of the program, the customer requirements, and the documents that described the system and the expected usage in detail.

IEEE now defines software engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software. Software Engineering tools are used to support the tasks by automating the tasks or parts of the tasks.

### **Software Life Cycle**

The period of time from when a software product is conceived and to when the software is no longer used, is termed as the Software Life Cycle. The software life cycle typically includes phases such as concept, requirements, design, implementation, test, installation, checkout, maintenance, and sometimes, retirement. These phases may overlap or be performed iteratively, based on life cycle model in use. Choice of an appropriate software life cycle model for a particular project depends on factors such as clarity of the problem, technology to be used, time at hand to develop and implement the project, etc.

## **2.2 Software Estimation Modeling**

Software project development comprises of a combination of engineering and management activities that are closely interleaved with one another. The selected life cycle model governs the engineering activities. Management activities, on the other hand, support and control the execution of the engineering activities.

Software Project Management involves three basic phases:

1. Project Planning
2. Project Monitoring and Control
3. Project Termination

*Project Planning* entails all activities that must be performed before starting the actual development work. It is a major management activity and it plays a vital role in determining the actual nature of the developed software project. Before a project can

begin, the manager and the software team must estimate the work to be done, the resources that will be required, and the time that will elapse from the start to the finish. Estimation begins with a description of the scope of the product. Software Scope describes the data and control to be processed, function, performance, constraints, interfaces and reliability. Until the scope is *bounded*, it is not possible to obtain a meaningful estimate. The problem is decomposed into sub problems, and each of these is estimated using historical data and experience as guidelines. The basic goal of planning is thus to look into the future, and identify the tasks that need to be accomplished to complete the project successfully, and handle the scheduling and the resource allocation of these tasks. A good plan is flexible enough to tackle unforeseen events that inevitably occur in a large software project. Social, economic and political factors must be taken into account for a realistic plan. The input to a planning process is the desired specification. The output is a *project plan*, which is a document that provides the various phases of the development process.

*Project Monitoring and Control* deals with the actual execution and updating of the project's plan. The progress of the plan is monitored periodically. It is essential to identify 'stages' within a project to aid the control and monitoring procedures. The plan may need to be modified depending upon various dynamic factors that come into the picture during the course of software development.

*Project Termination* stage involves the verification and validation activities. It also accounts for delivering according to the promised Quality Assurance Plan. *Verification* is the process of determining whether or not the products of a given phase of software development fulfill the specifications established earlier (i.e. during previous phases). Validation, on the other hand, is the process of evaluating the developed software to ensure compliance with the software requirements. Testing is a common method of validation. The verification and validation activities, together, are generally referred to as *V&V activities*. The major V&V activities for software development are inspection, reviews and testing (both static and dynamic). Testing is an activity that can be performed only on the source code. Inspection is a more general activity and can be applied to any

work product, including the source code. It is thus a formal evaluation technique in which software requirements, design or code are examined in detail to detect faults, violations of development standard, and other problems.

Software Estimation is an important area that falls under the planning procedure for any software. For a given set of requirements, it is desirable to know how much it will cost to develop the software, and how much time the development will take. Estimation of resources, cost, and schedule for a software project requires experience, access to good historical information, and the skill set for quantitative predictions when qualitative information is all that exists. Resources in a software-planning task can be conceptualized in the form of a pyramid. The *development environment* – hardware and software tools – sits at the foundation of the resources pyramid and provides infrastructure to support the development effort. At higher level, we encounter reusable *software components* – software building blocks that can dramatically reduce development costs and accelerates delivery. At the top of the pyramid is the primary resource – *the people*. The bulk of the cost of software development is due to the human effort, and most cost estimation methods focus on this aspect and give estimates in terms of *person-months*.

Accurate cost estimation is important because:

- It can help to classify and prioritize development projects with respect to an overall business plan.
- It can be used to determine what resources to commit to the project and how well these resources will be used.
- It can be used to assess the impact of changes and support replanning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs.

Software cost estimation involves the determination of one or more of the following estimates:

- a. Effort (usually in person-months)
- b. Project Development Time (in calendar time)
- c. Productivity

Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost. Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in person-months of the programmers, analysts and project managers. This effort estimate can be converted into a unit cost figure by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required.

Most cost models are based on the size measure, such as Lines of Code (LOC) and Function Point (FP), obtained from size estimation. The accuracy of size estimation directly impacts the accuracy of cost estimation. Size estimation, an internal attribute is of prime importance in this project report. It has been used in several effort/cost models as a predictor of the effort, duration and cost needed to design and implement the software.

The various size measures so computed can be applied to find:

- *productivity* in terms of person-months, and
- *quality* which gives the number of defects per unit in a software with respect to the requirements, design, coding and user documentation phases .

Time estimate for a module refers to the time a software engineer thinks it might take to complete the coding of that module. The unit of time could be hours or days or man-months or man-years. Thus time estimate can be considered under the *productivity* applications of size measures.

For instance, a module in the structure chart could have time estimate that reads as [5-10-20], 5 representing Optimistic Time, 10 representing Most Likely Time and 20 representing Pessimistic Time. In this case, Optimistic time is the time in which a

particular module could have been completed if everything went well and there have been no complications. A rule of thumb can be that there should be only one chance in ten of accomplishing the coding in less time than the optimistic time estimate.

Experienced software engineers would have thumb rules to estimate the size of the code before they write the code. This capability to predict the size of the code has been gained over a period of programming experience.

The time size estimation also includes complexity and program size among various other factors. For example it is known that the programmers can deliver a few lines of code per day. There is some limit to speed of a programmer so any development size estimate that is supposed to be independent of program size will be wrong if the program turns out to be larger than that can be constructed during estimation period. The size so estimated should be language and platform independent. Experienced software engineers are capable of estimating the time requirement for coding because they understand the complexity of the components of the programming task. They know their competence to code a particular logic using certain data structures, in a given language. They could visualize the sub modules in their programming task and arrive at a time estimate for the task. On the other hand, novice software engineers find it difficult to estimate the time requirement because of the lack of experience in understanding the complexity of the programming task.

A methodology should be designed that allows software engineers to explicitly spell out the different components of a module and provide time estimates for the development of each of the component. It could then be used to monitor the progress of code development and evaluate the developed program with respect to the time schedule.

These are some guidelines, which should be taken into account while undertaking the task of size estimation:

1. Analyze similar past projects to generate the historical data needed to estimate the size of new software projects. Relying on memory is not effective and leads to poor estimates.

2. Include data gathered from software and test team members. Early project team involvement not only serves to improve the accuracy of the estimate, but also prepares the team for the eventual project start date.
3. Keep in mind that experience is the key to effective software size estimation. The larger the project, the more is the experience required to make a good estimate.
4. Use multiple estimation techniques. During the initial estimation stage, the comparative results of different estimation techniques provide the best estimate.
5. Revise the initial size estimate as new information becomes available. During the design phase as the major software pieces come into focus, each module can be estimated separately, the sum of which reflects a revised, more accurate estimate.

To produce better estimates, we must improve our understanding of these project attributes and their causal relationships, model the impact of evolving environment, and develop effective ways of measuring software complexity.

At the initial stage of a project, there is high uncertainty about these project attributes. The estimate produced at this stage is inevitably inaccurate, as the accuracy depends highly on the amount of reliable information available to the estimator. If the effort estimate is on upper side, the other competitive bidder will win over us and get the contract. On the other hand, if our effort estimates are on lower side, we will have to incur loss in executing the project. Thus, it is a conflicting situation i.e. if we are not awarded the contract for the development project, we will never get an opportunity to actually design it to improve upon the initial estimation models. Hence, it is of paramount importance that we give as exact effort estimation as possible LOC has been widely used for this modeling purpose. As we learn more about the project during analysis and later design stages, the uncertainties are reduced and more accurate estimates can be made. Most models produce acceptable level results without regard to this uncertainty. They need to be enhanced to produce a range of estimates and their probabilities.

Although estimating is as much art as it is science, this important activity cannot be conducted in a haphazard manner. Estimation carries inherent risk and this risk can lead to uncertainty. These estimates are made within a certain limited time frame at the beginning of the project and must be updated regularly as the project progresses.

Estimation Risk is measured by the degree of uncertainty in the quantitative estimates established for resources, cost and schedule. If project scope is poorly understood or project requirements are vulnerable, uncertainty and risk become dangerously high. The software planner should thus demand the completeness of function, performance and interface definitions.

The *Degree of Structural Uncertainty* also has an effect on estimation risk. In this context, structure refers to the degree to which the requirements have been solidified, the ease with which the functions have been compartmentalized, and the hierarchical nature of the information that must be processed. Another significant factor that can influence the estimation risk is the *availability of historical information*. By backtracking, we can emulate things that worked and improve the areas where the problems arose.

### **2.3 Capability Maturity Model**

Capability Maturity Model (CMM) was developed by the SEI at Carnegie Mellon University in Pittsburgh. It has been used extensively for avionics software and government projects, in North America, Europe, Asia, Australia, South America, and Africa. CMM broadly refers to a process improvement approach that is based on a process model. CMM also refers specifically to the first such model, developed by the Software Engineering Institute (SEI) in the mid-1980s, as well as the family of process models that followed. A process model is a structured collection of practices that describe the characteristics of effective processes; the practices included are those proven by experience to be effective.

The Capability Maturity Model can be used to assess an organization against a scale of five process maturity levels. Each level ranks the organization according to its standardization of processes in the subject area being assessed. The subject areas can be as diverse as software engineering, systems engineering, project management, risk management, system acquisition, information technology (IT) services and personnel management.

## **Structure of CMM**

### *Maturity Levels*

A layered framework providing a progression to the discipline needed to engage in continuous improvement (It is important to state here that an organization develops the ability to assess the impact of a new practice, technology, or tool on their activity. Hence it is not a matter of adopting these, rather it is a matter of determining how innovative efforts influence existing practices. This really empowers projects, teams, and organizations by giving them the foundation to support reasoned choice.)

### *Key Process Areas*

Key process area (KPA) identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important.

### *Goals*

The goals of a key process area summarize the states that must exist for that key process area to have been implemented in an effective and lasting way. The extent to which the goals have been accomplished is an indicator of how much capability the organization has established at that maturity level. The goals signify the scope, boundaries, and intent of each key process area.

### *Common Features*

Common features include practices that implement and institutionalize a key process area. These five types of common features include: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation.

### *Key Practices*

The key practices describe the elements of infrastructure and practice that contribute most effectively to the implementation and institutionalization of the key process areas.

## **Levels of CMM**

There are five levels of the CMM. According to the SEI,

*Predictability, effectiveness, and control of an organization's software processes are believed to improve as the organization moves up these five levels. While not rigorous, the empirical evidence to date supports this belief."*



**Level 1- Initial**

At maturity level 1, processes are usually ad hoc and the organization usually does not provide a stable environment. In spite of this ad hoc, chaotic environment, maturity level 1 organizations often produce products and services that work; however, they frequently exceed the budget and schedule of their projects. Maturity level 1 organizations are characterized by a tendency to over commit, abandon processes in the time of crisis, and not be able to repeat their past successes again. Level 1 software project success depends on having high quality people.

**Level 2- Repeatable**

At maturity level 2, software development successes are repeatable. The processes may not repeat for all the projects in the organization. The organization may use some basic project management to track cost and schedule. Process discipline helps ensure that existing practices are retained during times of stress. When these practices are in place, projects are performed and managed according to their documented plans. Basic project management processes are established to track cost, schedule, and functionality. The minimum process discipline is in place to repeat earlier successes on projects with similar applications and scope. There is still a significant risk of exceeding cost and time estimates.

**Level 3 – Defined**

The organization's set of standard processes, which is the basis for level 3, is established and improved over time. These standard processes are used to establish consistency across the organization. Projects establish their defined processes by the organization's set of standard processes according to tailoring guidelines. A critical distinction between level 2 and level 3 is the scope of standards, process descriptions, and procedures. At level 2, the standards, process descriptions, and procedures may be quite different in each specific instance of the process (for example, on a particular project). At level 3, the standards, process descriptions, and procedures for a project are tailored from the organization's set of standard processes to suit a particular project or organizational unit.

#### **Level 4 – Managed**

Organizations at this level set quantitative quality goals for both software process and software maintenance. Subprocesses are selected that significantly contribute to overall process performance. These selected subprocesses are controlled using statistical and other quantitative techniques. A critical distinction between maturity level 3 and maturity level 4 is the predictability of process performance. At maturity level 4, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable. At maturity level 3, processes are only qualitatively predictable.

#### **Level 5 – Optimizing**

Maturity level 5 focuses on continually improving process performance through both incremental and innovative technological improvements. Quantitative process-improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement. The effects of deployed process improvements are measured and evaluated against the quantitative process-improvement objectives. Both the defined processes and the organization's set of standard processes are targets of measurable improvement activities. Optimizing processes that are nimble, adaptable and innovative depends on the participation of an empowered workforce aligned with the business values and objectives of the organization. The organization's ability to rapidly respond to changes and opportunities is enhanced by finding ways to accelerate and share learning. critical distinction between maturity level 4 and maturity level 5 is the type of process variation addressed. At maturity level 4, processes are concerned with addressing special causes of process variation and providing statistical predictability of the results. Though processes may produce predictable results, the results may be insufficient to achieve the established objectives. At maturity level 5, processes are concerned with addressing common causes of process variation and changing the process (that is, shifting the mean of the process performance) to improve process performance (while maintaining statistical probability) to achieve the established quantitative process-improvement objectives.

## Capability Maturity Model Integration

The CMMI is the successor of the CMM. The CMM was developed from 1987 until 1997. In 2002 version 1.1 of the CMMI was released: v1.2 followed in August 2006. The goal of the CMMI project is to improve usability of maturity models for software engineering and other disciplines, by integrating many different models into one framework. It was created by members of industry, government and the SEI.

The CMMI comes with two different representations - staged and continuous. The staged model, which groups process areas into 5 *maturity* levels, was also used in the ancestor software development CMM, and is the representation used to achieve a "CMMI Level Rating" from a SCAMPI appraisal. The continuous representation, which was used in the ancestor systems engineering CMM, defines *capability* levels within each profile. The differences in the representations are solely organizational; the content is equivalent. The CMMI uses a common structure to describe each of the 22 process areas (PAs). A process area has 1 to 4 *goals*, and each goal is comprised of *practices*. Within the 22 PAs these are called *specific* goals and practices, as they describe activities that are specific to a single PA. There is one additional set of goals and practices that apply in common across all of the PAs; these are called *generic* goals and practices.

CMMI should be adapted to each individual company; therefore companies are not "certified." A company is appraised (e.g. with an appraisal method like SCAMPI) at a certain level of CMMI.

In this chapter we have examined the fundamental issues of software engineering that are of relevance to the estimation effort modeling. The next chapter outlines the Cost Estimation aspects.

## CHAPTER 3

---

### SOFTWARE COST ESTIMATION MODELS

---

#### **3.1 Introduction**

In the previous chapter we have seen the importance of assessment of effort estimation. There are two types of models that have been used to estimate software cost: *cost models* and *constraint models*.

##### **(i) Cost Models**

Cost models provide direct estimates of effort. These models typically have a primary cost factor such as size and a number of secondary adjustment factors or *cost drivers*. Cost drivers are characteristics of the project, process, products, or resources that influence effort. Cost drivers are used to adjust the preliminary estimate provided by the primary cost factor.

A typical cost model is derived using regression analysis on data collected from past software projects. Effort is plotted against the primary cost factor for a series of projects. The line of best fit is then calculated among the data points. If the primary cost factor were a perfect predictor of effort, then every point on the graph would lie on the line of best fit. In reality however, there is usually a significant residual error. It is therefore necessary to identify the factors that cause variation between predicted and actual effort. These parameters are added to the model as cost drivers.

The overall structure of regression-based models takes the form:

$$E = A + B x S^C$$

where  $A$ ,  $B$ , and  $C$  are empirically derived constants,  $E$  is effort in person months, and  $S$  is the primary input (typically either LOC or FP).

The following are some examples of cost models using LOC as a primary input:

**Table 1 LOC Based Models**

$E = 5.2 \times (\text{KLOC})^{0.91}$	Walston-Felix Model
$E = 5.5 + 0.73 \times (\text{KLOC})^{1.16}$	Bailey-Basili Model
$E = 3.2 \times (\text{KLOC})^{1.05}$	COCOMO Basic Model
$E = 5.288 \times (\text{KLOC})^{1.047}$	Doty Model for KLOC > 9

Cost models using FP as a primary input include:

**Table 2 FP Based Models**

$E = -12.39 + 0.0545 \text{ FP}$	Albrecht and Gaffney Model
$E = 60.62 \times 7.728 \times 10^{-8} \text{ FP}^3$	Kemerer Model
$E = 585.7 + 15.12 \text{ FP}$	Matson, Barnett, and Mellichamp Model

**(ii) Constraint Models**

Constraint models demonstrate the relationship over time between two or more parameters of effort, duration, or staffing level. The RCA PRICE S model and Putnam’s SLIM model are two examples of constraint models.

**3.2 Constructive Cost Models**

The most fundamental cost models used for software cost estimation are COCOMO and COCOMO – II and we have identified them for further study in the context of this project.

**COCOMO ’81**

Boehm derived a cost model called COCOMO (Constructive Cost Model) using data from a large set of projects at TRW, a consulting firm based in California. COCOMO is a relatively straightforward model based on inputs relating to the size of the system and a number of cost drivers that affect productivity. The original COCOMO model was first

published in 1981. Boehm and his colleagues have since defined an updated COCOMO, called COCOMO II that accounts for recent changes in software engineering technology.

### **Original COCOMO**

The original COCOMO is a collection of three models: a *Basic* model that is applied early in the project, an *Intermediate* model that is applied after requirements are specified, and an *Advanced* model that is applied after design is complete. All three models take the form:

$$E = aS^b \times EAF$$

where  $E$  is effort in person months,  $S$  is size measured in thousands of lines of code (KLOC), and  $EAF$  is an effort adjustment factor (equal to 1 in the Basic model). The factors  $a$  and  $b$  depend on the development mode. Boehm has defined three development modes:

1. *Organic mode* – relatively simple projects in which small teams work to a set of informal requirements (i.e. thermal transfer program developed for a heat transfer group).
2. *Semi-detached mode* – an intermediate project in which mixed teams must work to a set of rigid and less than rigid requirements (i.e. a transaction processing system with fixed requirements for terminal hardware and software).
3. *Embedded mode* – a project that must operate within a tight set of constraints (ie. flight control software for aircraft).

### **Basic COCOMO**

The Basic COCOMO model computes effort as a function of program size. The Basic COCOMO equation is:

$$E = a (KLOC)^b$$

The factors  $a$  and  $b$  for the Basic COCOMO model are shown in Table 3.

**Table 3 Effort for three modes of Basic COCOMO**

Mode	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

### **Intermediate COCOMO**

The Intermediate COCOMO model computes effort as a function of program size and a set of cost drivers. The Intermediate COCOMO equation is:

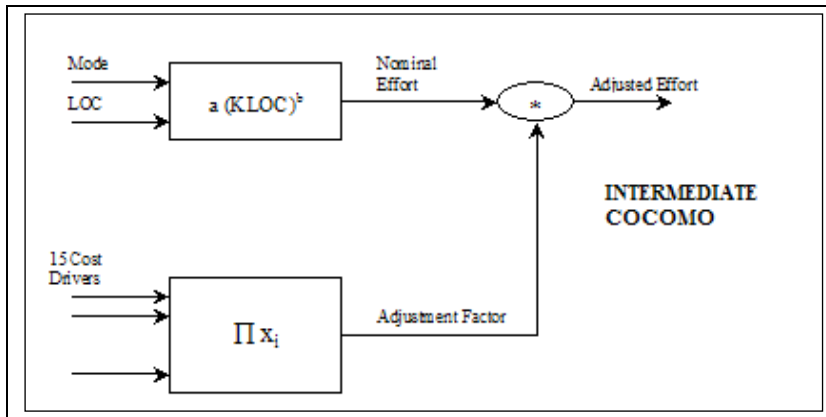
$$E = a (KLOC)^b \times EAF$$

The factors  $a$  and  $b$  for the Intermediate COCOMO model are shown in Table 4.

**Table 4 Effort parameters for three modes of Intermediate COCOMO**

Mode	a	b
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

The effort adjustment factor (EAF) is calculated using 15 cost drivers. The cost drivers are grouped into four categories: *product*, *computer*, *personnel*, and *project*. Each cost driver is rated on a six-point ordinal scale ranging from low to high importance. Based on the rating, an effort multiplier is determined using Table 5. The product of all effort multipliers is the EAF. The figure illustrates the basic process of the COCOMO model.



**Table 5. Software Development Effort Multipliers**

Cost Driver	Description	Rating					
		Very Low	Low	Nominal	High	Very High	Extra High
<i>Product</i>							
RELY	Required software reliability	0.75	0.88	1.00	1.15	1.40	-
DATA	Database size	-	0.94	1.00	1.08	1.16	-
CPLX	Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
<i>Computer</i>							
TIME	Execution time constraint	-	-	1.00	1.11	1.30	1.66
STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.56
VIRT	Virtual machine volatility	-	0.87	1.00	1.15	1.30	-
TURN	Computer turnaround time	-	0.87	1.00	1.07	1.15	-
<i>Personnel</i>							
ACAP	Analyst capability	1.46	1.19	1.00	0.86	0.71	-
AEXP	Applications experience	1.29	1.13	1.00	0.91	0.82	-
PCAP	Programmer capability	1.42	1.17	1.00	0.86	0.70	-
VEXP	Virtual machine experience	1.21	1.10	1.00	0.90	-	-
LEXP	Language experience	1.14	1.07	1.00	0.95	-	-
<i>Project</i>							
MODP	Modern programming practices	1.24	1.10	1.00	0.91	0.82	-
TOOL	Software Tools	1.24	1.10	1.00	0.91	0.83	-
SCED	Development Schedule	1.23	1.08	1.00	1.04	1.10	-

**Advanced COCOMO**



The Advanced COCOMO model computes effort as a function of program size and a set of cost drivers weighted according to each phase of the software lifecycle. The Advanced model applies the Intermediate model at the component level, and then a phase-based approach is used to consolidate the estimate.

The 4 phases used in the detailed COCOMO model are: requirements planning and product design (RPD), detailed design (DD), code and unit test (CUT), and integration and test (IT). Each cost driver is broken down by phase as in the example shown in Table 6.

**Table 6 Analyst capability effort multiplier for Detailed COCOMO**

Cost Driver	Rating	RPD	DD	CUT	IT
ACAP	Very Low	1.80	1.35	1.35	1.50
	Low	0.85	0.85	0.85	1.20
	Nominal	1.00	1.00	1.00	1.00
	High	0.75	0.90	0.90	0.85
	Very High	0.55	0.75	0.75	0.70

Estimates made for each module are combined into subsystems and eventually an overall project estimate. Using the detailed cost drivers, an estimate is determined for each phase of the lifecycle.

### 3.3 COCOMO II

Whereas COCOMO is reasonably well matched to custom, build-to-specification software projects, COCOMO II is useful for a much wider collection of techniques and technologies. COCOMO II provides up-to-date support for business software, object-oriented software, software created via spiral or evolutionary development models, and software developed using commercial-off-the-shelf application composition utilities. COCOMO II includes the *Application Composition* model (for early prototyping efforts) and the more detailed *Early Design* and *Post-Architecture* models (for subsequent portions of the lifecycle).

## The Application Composition Model

The Application Composition model is used in prototyping to resolve potential high-risk issues such as user interfaces, software/system interaction, performance, or technology maturity. *Object points* are used for sizing rather than the traditional LOC metric.

An initial size measure is determined by counting the number of screens, reports, and third-generation components that will be used in the application. Each object is classified as simple, medium, or difficult using the guidelines shown in Tables 7 and 8.

Table 7 Object point complexity levels for screens

Number of views contained	Number and source of data tables		
	Total <4	Total <8	Total 8+
< 3	simple	simple	medium
3 to 7	simple	medium	difficult
8 +	medium	difficult	difficult

Table 8 Object point complexity levels for reports

Number of views contained	Number and source of data tables		
	Total <4	Total <8	Total 8+
< 3	Simple	simple	medium
3 to 7	Simple	medium	difficult
8 +	Medium	difficult	difficult

The number in each cell is then weighted according to Table 9. The weights represent the relative effort required to implement an instance of that complexity level.

Table 9 Complexity weights for object points

Object type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	-	-	10

The weighted instances are summed to provide a single object point number. Reuse is then taken into account. Assuming that  $r\%$  of the objects will be reused from previous projects, the number of new object points (NOP) is calculated to be:

$$NOP = (\text{object points}) \times (100 - r) / 100$$

A productivity rate (PROD) is determined using Table 10.

**Table 10 Average productivity rates**

<b>Developers' experience and capability</b>	<b>Very Low</b>	<b>Low</b>	<b>Nominal</b>	<b>High</b>	<b>Very High</b>
ICASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

Effort can then be estimated using the following equation:

$$E = NOP / PROD$$

### **The Early Design Model**

The Early Design model is used to evaluate alternative software/system architectures and concepts of operation. An unadjusted function point count (UFC) is used for sizing. This value is converted to LOC using tables such as those published by Jones, excerpted in Table 11.

**Table 11 Programming language levels and ranges of source code per function point**

Language	Level	Min	Mode	Max
Machine language	0.10	-	640	-
Assembly	1.00	237	320	416
C	2.50	60	128	170
RPGII	5.50	40	58	85
C++	6.00	40	55	140
Visual C++	9.50	-	34	-
PowerBuilder	20.00	-	16	-
Excel	57.00	-	5.5	-

The Early Design model equation is:

$$E = a (KLOC) \times EAF$$

where  $a$  is a constant, provisionally set to 2.45. The effort adjustment factor (EAF) is calculated as in the original COCOMO model using the 7 cost drivers shown in Table 12. The Early Design cost drivers are obtained by combining the Post-Architecture cost drivers shown in Table 13.

Table 12 Early Design Cost Drivers

Cost Driver	Description	Counterpart Architecture Cost Driver	Combined Post- Architecture Cost Driver
RCPX	Product reliability and complexity	RELY, DATA, CPLX, DOCU	
RUSE	Required reuse	RUSE	
PDIF	Platform difficulty	TIME, STOR, PVOL	
PERS	Personnel capability	ACAP, PCAP, PCON	
PREX	Personnel experience	AEXP, PEXP, LTEX	
FCIL	Facilities	TOOL, SITE	
SCED	Schedule	SCED	

## The Post-Architecture Model

The Post-Architecture model is used during the actual development and maintenance of a product. Function points or LOC can be used for sizing, with modifiers for reuse and software breakage. Boehm advocates the set of guidelines proposed by The Software Engineering Institute in counting lines of code. The Post-Architecture model includes a set of 17 cost drivers and a set of 5 factors determining the projects scaling component. The 5 factors replace the development modes (organic, semidetached, embedded) of the original COCOMO model.

The Post-Architecture model equation is:

$$E = a (KLOC)^b \times EAF$$

where  $a$  is set to 2.55 and  $b$  is calculated as:

$$b = 1.01 + 0.01 \times SUM(W_i)$$

where  $W$  is the set of 5 scale factors shown in Table 13.

Table 13 COCOMO II scale factors

<b>W(i)</b>	<b>Very Low</b>	<b>Low</b>	<b>Nominal</b>	<b>High</b>	<b>Very High</b>	<b>Extra High</b>
Precedentedness	4.05	3.24	2.42	1.62	0.81	0.00
Development/Flexibility	6.07	4.86	3.64	2.43	1.21	0.00
Architecture/Risk Resolution	4.22	3.38	2.53	1.69	0.84	0.00
Team Cohesion	4.94	3.95	2.97	1.98	0.99	0.00
Process Maturity	4.54	3.64	2.73	1.82	0.91	0.00

The EAF is calculated using the 17 cost drivers shown in Table 14.

**Table 14 Post-Architecture Cost Drivers**

Cost Driver	Description	Rating					
		Very Low	Low	Nominal	High	Very High	Extra High
<i>Product</i>							
RELY	Required software reliability	0.75	0.88	1.00	1.15	1.39	-
DATA	Database size	-	0.93	1.00	1.09	1.19	-
CPLX	Product complexity	0.70	0.88	1.00	1.15	1.30	1.66
RUSE	Required reusability		0.91	1.00	1.14	1.29	1.49
DOCU	Documentation		0.95	1.00	1.06	1.13	
<i>Platform</i>							
TIME	Execution time constraint	-	-	1.00	1.11	1.31	1.67
STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.57
PVOL	Platform volatility	-	0.87	1.00	1.15	1.30	-
<i>Personnel</i>							
ACAP	Analyst capability	1.50	1.22	1.00	0.83	0.67	-
PCAP	Programmer capability	1.37	1.16	1.00	0.87	0.74	-
PCON	Personnel continuity	1.24	1.10	1.00	0.92	0.84	-
AEXP	Applications experience	1.22	1.10	1.00	0.89	0.81	-
PEXP	Platform experience	1.25	1.12	1.00	0.88	0.81	-
LTEX	Language and tool experience	1.22	1.10	1.00	0.91	0.84	
<i>Project</i>							
TOOL	Software Tools	1.24	1.12	1.00	0.86	0.72	-
SITE	Multisite development	1.25	1.10	1.00	0.92	0.84	0.78
SCED	Development Schedule	1.29	1.10	1.00	1.00	1.00	-

## CHAPTER 4

---

### FUNCTION POINT AND CLASS POINT MEASURES

---

#### **4.1 Software Metrics**

Measurements in the physical world can be categorized in two ways: Direct Measures (e.g. the length of a bolt) and indirect measures (e.g. the 'quality' of the bolts produced, measured by counting the rejects). An analogy from this basic concept can be used to classify the software metrics as well. *Direct Measures*, thus include the cost and effort applied. These define the parameters like Lines of Code (LOC) produced, execution speed, memory size, and defects reported over some fixed time interval. Indirect Measures of the product include the functionality, quality, complexity, efficiency, reliability, maintainability etc. The direct measures can be determined easily, as long as specific conventions for measurement are established in advance. The indirect measures, however, are more difficult to assess. The *software metrics domain* can be partitioned into process, project and product metrics. Product metrics, are generally private to an individual, and are generally combined to develop project metrics that are public to a software team. Project metrics are then consolidated to create process metrics that are public to the software organization as a whole. The need for metrics is particularly acute when an organization is adopting a new technology for which the established practices have not been developed.

Taxonomy of the Software Metrics is presented below:

1. Size-Oriented Metrics
2. Function-Oriented Metrics
3. Extended Function Point Metrics

Normalizing the quality and/or productivity measures by considering the *size* of the software that has been produced derives *Size-Oriented Metrics*. Generally a table of size-oriented measures is constructed from the records of the software organization. The table

lists each software development project that has been completed over the past few years and the corresponding measures (e.g. LOC, Effort, Errors, Defects, People, Cost, Pages of Documentation) of that project. All the activities like analysis, design, coding and testing are accounted for. In order to develop metrics that can be assimilated with similar metrics from other projects, the LOC parameter is chosen for normalization. From the rudimentary data contained in the table, a set of simple size-oriented metrics is generated.

*Size-Oriented Metrics* are not universally accepted as the best way to measure the software development process. The controversy swirls around the use of LOC as a key measure. The proponents claim that LOC is an artifact of all the software estimation models, whereas the opponents argue that LOC measures are programming language dependent, and that their use in estimation requires a level of detail that may be difficult to achieve at the planning stage.

*Function-Oriented Metrics*, use a measure of functionality delivered by the application as a normalization value. Since *functionality* is an indirect measure, it must be derived indirectly from existing direct measures. We define a *function point* for this purpose. Function points are deduced using an empirical relationship based on countable (i.e. direct) measures of the software's information domain and assessments of the software complexity.

## **4.2 Function Point Approach**

Function points are computed by first calculating an *unadjusted function point count* (UFC). Counts are made for the following categories:

1. *Number of user inputs* – those items provided by the user that describe distinct application-oriented data (such as file names and menu selections). Inputs should be distinguished from inquiries.
2. *Number of user outputs* – those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than the individual components of these).



3. *Number of user inquiries* – an inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output.
4. *Number of logical files* – each logical master file, i.e., a logical grouping of data that may be a part of large database or a separate file, is counted.
5. *Number of external interfaces* – all machine readable interfaces (e.g., data files on storage media) that are used to transfer information to another system are counted.

**Table 15 Function point complexity weights**

Measurement Parameter	Weighting Factors		
	Simple	Average	Complex
<b>Number of User Inputs</b>	<b>3</b>	<b>4</b>	<b>6</b>
<b>Number of User Outputs</b>	<b>4</b>	<b>5</b>	<b>7</b>
<b>Number of User Inquiries</b>	<b>3</b>	<b>4</b>	<b>6</b>
<b>Number of Logical Files</b>	<b>7</b>	<b>10</b>	<b>15</b>
<b>Number of External Interfaces</b>	<b>5</b>	<b>7</b>	<b>10</b>

Each count is multiplied by its corresponding complexity weight and the results are summed to provide the UFC. The *adjusted function point count* (FP) is calculated by multiplying the UFC by a *technical complexity factor* (TCF). Components of the TCF are listed in Table 16.

Table 16 Components of the Technical Complexity Factor

F1	Reliable back-up and recovery	F2	Data communications
F3	Distributed functions	F4	Performance
F5	Heavily used configuration	F6	Online data entry
F7	Operational ease	F8	Online update
F9	Complex interface	F10	Complex processing
F11	Reusability	F12	Installation ease
F13	Multiple sites	F14	Facilitate change

Each component is rated from 0 to 5, where 0 means the component has no influence on the system and 5 means the component is essential. The TCF can then be calculated as:

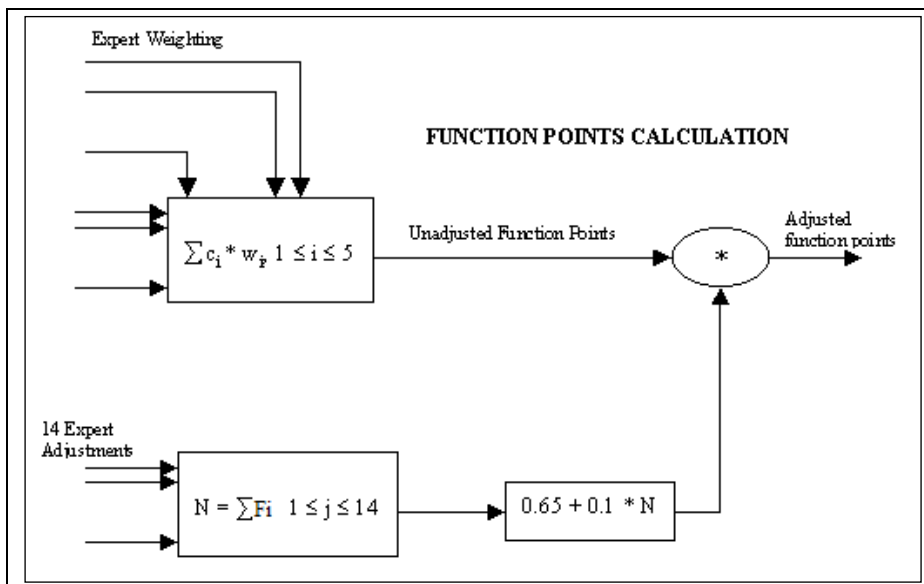
$$TCF = 0.65 + 0.01 (SUM (F_i))$$

The factor varies from 0.65 (if each  $F_i$  is set to 0) to 1.35 (if each  $F_i$  is set to 5).

The final function point calculation is:

$$FP = UFC \times TCF$$

The figure illustrates this process of Function Points Calculations.



**Extended Function Point Metrics** were proposed to accommodate both data dimensions as well as control (functional and behavioral) dimensions. A major limitation of the Function Point approach is that control dimensions are not emphasized upon which makes the approach inadequate for many engineering and embedded system problems.

A **Feature Point** is a superset of the function point measure that can be applied to problems, which are high algorithmic complexity. To compute a feature point, information domain values are again counted and weighted as discussed in the previous section. In addition, the feature point metric accounts for a new software characteristic – algorithms. An algorithm is defined as a bounded computational problem that is included in a specific computer program. The data dimension is evaluated using retained data (e.g.,

files) and the external data (e.g., inputs, outputs and inquiries). The function dimension is measured by considering the number of internal operations required to transform the input to the output. The behavioral dimensions include the states and state transitions. Together, the functional and the behavioral dimension determine the control dimension.

### **Benefits of Function Point Analysis**

Organizations that adopt Function Point Analysis as software metric realize many benefits including: improved project estimating; understanding project and maintenance productivity; managing changing project requirements; and gathering user requirements. Each of these is discussed below. Estimating software projects is as much an art as a science. While there are several environmental factors that need to be considered in estimating projects, two key data points are essential. The first is the size of the deliverable. The second addresses how much of the deliverable can be produced within a defined period of time. Size can be derived from Function Points, as described above. The second requirement for estimating is determining how long it takes to produce a function point. This delivery rate can be calculated based on past project performance or by using industry benchmarks. The delivery rate is expressed in function points per hour (FP/Hr) and can be applied to similar proposed projects to estimate effort

Project Hours = estimated project function points FP/Hr.

*Productivity measurement* is a natural output of Function Points Analysis. Since function points are technology independent they can be used as a vehicle to compare productivity across dissimilar tools and platforms. More importantly, they can be used to establish a productivity rate (i.e. FP/Hr) for a specific tool set and platform. Once productivity rates are established they can be used for project estimating as described above and tracked over time to determine the impact continuous process improvement initiatives have on productivity.

In addition to delivery productivity, function points can be used to evaluate the support requirements for maintaining systems. In this analysis, productivity is determined by

calculating the number of function points one individual can support for a given system in a year (i.e. FP/FTE year). When compared with other systems, these rates help to identify which systems require the most support. The resulting analysis helps an organization develop a maintenance and replacement strategy for those systems that have high maintenance requirements.

Managing Change of Scope for an in-process project is another key benefit of Function Point Analysis. Once a project has been approved and the function point count has been established, it becomes a relatively easy task to identify, track and communicate new and changing requirements. As requests come in from users for new displays or capabilities, function point counts are developed and applied against the rate. This result is then used to determine the impact on budget and effort. The user and the project team can then determine the importance of the request against its impact on budget and schedule. At the conclusion of the project the final function point count can be evaluated against the initial estimate to determine the effectiveness of requirements gathering techniques. This analysis helps to identify opportunities to improve the requirements definition process.

Organizations that adopt Function Point Analysis as software metric realize many benefits including: improved project estimating; understanding project and maintenance productivity.

FPA has become generally accepted as an effective way to

- Estimate a software project's size (and in part, duration)
- Establish productivity rates in function points per hour
- Evaluate support requirements
- Estimate system change costs
- Normalize the comparison of software modules.
- Managing changing project requirements; and gathering user requirements.

Function Points measures systems from a functional perspective they are independent of technology. Regardless of language, development method, or hardware platform used, the number of function points for a system will remain constant. The only variable is the amount of effort needed to deliver a given set of function points; therefore, Function

Point Analysis can be used to determine whether a tool, an environment, a language is more productive compared with others within an organization or among organizations. This is a critical point and one of the greatest values of Function Point Analysis.

The task of counting function points should be included as part of the overall project plan. That is, counting function points should be scheduled and planned. The first function point count should be developed to provide sizing used for estimating.

### **4.3 Class Point Approach**

The Class Point approach is analogous to the Function point approach in its technique. It has been developed to aid the size estimation in object-oriented systems. In this approach, the basic unit is a class. Thus the entities that are counted and weighed in this case are classes. Although in some ways Class Point may be considered an extension of Function Point with Object Oriented features. But there is no one-to-one mapping between the logical files and transactions of Function Point Approach and the classes and methods of Class Point Approach. This is because of the differences between the procedural and Object Oriented features. This is of great value as it encompasses the advantages of various earlier approaches in size estimation and overcomes the drawbacks.

#### **Class Point Algorithm**

The Class Point methodology can be expressed in algorithmic form by implementing the following steps:

1. Process the information available for size estimation. This further includes the following sub-steps-:
  - a. Identify and organize the classes in to groups depending upon their application domain.
  - b. Evaluate the complexity level of each class.
  - c. Estimate the Total Unadjusted Class Point (TUCP).
2. Estimate the technical complexity factor.
3. Evaluate the final Class Point value based on the results of the above steps.

## **Taxonomy of classes**

The basic unit of class point approach is classes. In the first step these classes are identified and organized in groups. They can be categorized on the basis of their application into 4 groups namely:

1. Problem Domain Type (PDT)

They contain classes representing real-world entities in the application domain of the system

2. Human Interaction Type (HIT)

They are designed to satisfy the need for information visualization and human-computer interaction

3. Data Management Type (DMT)

The DMT component encompasses the classes that offer functionality for data storage and retrieval.

4. Task Management type (TMT)

TMT classes are designed for task management purposes, thus they are responsible for the definition and control of tasks such as *Manage-Emergency-Control* and *Report-Emergency-Control*. Moreover, such components also include classes responsible for communication between subsystems allocated to different computers, and classes responsible for the communication with external systems. As a matter of fact, Message and Connection are typical classes falling within this component is assigned a high complexity level.

## **Evaluation of complexity of each class**

The behavior of each class component is taken into account in order to evaluate its complexity level. There are two measures for finding the complexity of any system: CP1 and CP2. The difference between the CP1 and CP2 measures lies in the way such a

complexity level is determined. In particular, in CP1 the number of external methods and the number of services requested are taken into account; whereas, in CP2, the number of attributes is also exploited.

**The Number of External Methods (NEM)** measures the size of the interface of a class and is determined by the number of locally defined public methods. The Number of Services Requested (NSR) provides a measure of the interconnection of system components.

Both measures are available in a distributed information system for design documentation. Indeed, activities that characterize any OO design process include the identification of the classes with their attributes and methods, and the construction of interaction (or collaboration) diagrams showing which external services are needed for a class to perform the expected tasks. Thus CP1 is generally applicable only in the estimation of preliminary stages. CP2 on the other hand, also depends upon the **Number of Attributes (NOA)**. It is thus more often used for refining the existing information. This is done when more information about the software is available, for example when the number of attributes is known. In both measures, CP1 and CP2, after the complexity has been evaluated, it weighed based on its type and level of complexity. It can be assigned levels like: HIGH, MEDIUM and LOW.

This is illustrated in the table given below:

	0 - 4 <i>NEM</i>	5 - 8 <i>NEM</i>	$\geq 9$ <i>NEM</i>
0 - 1 <i>NSR</i>	Low	Low	Average
2 - 3 <i>NSR</i>	Low	Average	High
$\geq 4$ <i>NSR</i>	Average	High	High

In the same way complexity levels for CP2 are also formed. In this case, for each range of NOA values, a table similar to the one above is constructed in which a particular value gives the level based on all three characteristics.

**Estimate the TUCP (Total Unadjusted Class Point factor)**

To find the TUCP, the value of complexity of each level is multiplied by the weight assigned to the particular type and level to which it belongs. The TUCP is computed as the weighted total of the four components of the application:

$$TUCP = \sum_{i=1}^4 \sum_{j=1}^3 w_{ij} \times x_{ij},$$

where  $x_{ij}$  is the number of classes of component type  $i$  (Problem Domain, Human Interaction etc.) with complexity level  $j$  (Low, Average, High), and  $w_{ij}$  is the weighting value for type  $i$  and complexity level  $j$ .

Evaluating the TUCP					
System Component Type	Description	Complexity			
		Low	Average	High	Total
PDT	Problem Domain	...*3=...	...*6=...	...*10=...	...
HIT	Human Interaction	...*4=...	...*7=...	...*12=...	...
DMT	Data Management	...*5=...	...*8=...	...*13=...	...
TMT	Task Management	...*4=...	...*6=...	...*9=...	...
<b>TUCP</b>		<b>Total Unadjusted Class Point</b>			

### Find the TCF (Technical Complexity Factor)

The TCF depends upon the TDI (Total Degree of Influence) as shown by the formula.

$$TCF = 0.55 + 0.01 * TDI$$

As it can be seen, even when there is no TDI (or TDI=0), TCF still exist. This is due to the basic complexity which exists in each and every case. Thus to find the TCF, we need to calculate TDI. TDI in turn is derived by taking the sum total of all the degrees of influence of various predetermined factors.



ID	System Characteristic	DI	ID	System Characteristic	DI
C1	Data Communication	...	C10	Reusability	...
C2	Distributed Functions	...	C11	Installation ease	...
C3	Performance	...	C12	Operational ease	...
C4	Heavily used configuration	...	C13	Multiple sites	...
C5	Transaction rate	...	C14	Facilitation of change	...
C6	Online data entry	...	C15	User Adaptivity	...
C7	End-user efficiency	...	C16	Rapid Prototyping	...
C8	Online update	...	C17	Multiuser Interactivity	...
C9	Complex processing	...	C18	Multiple Interfaces	...
<b>TDI</b>	<b>Total Degree of Influence</b>				...

Given below are examples to illustrate how to assign degree of influence.

Guidelines to Determine the Influence Degree of the Introduced Factors	
<b>Adaptivity</b>	<p>0 – No adaptivity capability is required for the given application.</p> <p>1 – The system should be able to adjust the form of output in response to a change in input. The system should have a limited variety of behaviors.</p> <p>2 - The system should be able to have a dialogue record, which allows it to respond to sequences of stimuli rather than just individual input.</p> <p>3 - The system should be able to have a dialogue record, which allows it to adapt in response to a history of the interaction.</p> <p>4 - The system should be able to monitor the effects of the adaptation on the subsequent interaction and evaluate this through trial and error, by selecting from a range of possible outputs for any given input.</p> <p>5 - The system should have inference mechanisms to abstract from the dialogue record and capture a design or intentional interpretation of the interaction.</p>
<b>Rapid Prototyping</b>	<p>0 – No prototyping is required for the given application.</p> <p>1 – Paper mock-ups are required to realize scenario tests.</p> <p>2 – A user interface prototype is required, which presents an interface for some use cases, with no functionality, to realize a usability test.</p> <p>3 – A user interface prototype is required, which presents an interface for most use cases, with no functionality, to realize a usability test.</p> <p>4 – A functional prototype, implementing key aspects of the system, is required to receive feedback from end-users.</p> <p>5 – A functional prototype, implementing key aspects of the system, is required to receive feedback from end-users. The modified version will be resubmitted to users' validation.</p>
<b>Multiple Interfaces</b>	<p>A score for each of the following items</p> <ul style="list-style-type: none"> <li>• A lexical customization is required, where user can only adjust the position of some widgets on the screen or redefine command names. The structure of the interaction is unchanged.</li> <li>• Different interfaces for novice and experienced users are required.</li> <li>• Different interfaces are required based on users' cultural, educational, and employment background.</li> <li>• Accessible interfaces are required to cope with different types of disabilities.</li> <li>• Different interaction platforms have to be considered.</li> </ul>
<b>Multiuser Interactivity</b>	<p>A score for each of the following items</p> <ul style="list-style-type: none"> <li>• The system should provide replicated windows on multiple workstations.</li> <li>• Heterogeneous hardware environments should be considered.</li> <li>• The system should provide simple control of the order of interaction among users.</li> <li>• The system should provide control of the simultaneity of interaction among users.</li> <li>• The system should provide sophisticated process synchronization and control of the order and simultaneity of interaction among users.</li> </ul>

**Evaluate the final CP**

The final value of the Adjusted Class Point (CP) is obtained by multiplying the TUCP value by TCF

$$CP = TUCP * TCF$$

In this chapter we have seen the function point and class point approaches for effort estimation. Chapter 2 described the COCOMO models. It has become evident that these conventional modeling have to handle many uncertainties and hence we introduce the basics of fuzzy sets and fuzzy logic in the next chapter to discuss vague and uncertainty handling.

## CHAPTER 5

---

### SOFT COMPUTING AND FUZZY LOGIC

---

#### **5.1 Introductory Soft Computing Concepts**

The modern computer industry is now shifting towards what is known as ‘*soft*’ computing, and a deviation from the conventional use of computer systems from zero-IQ machines to systems that can go hand-in-hand with human thought processes is being discovered. The preliminary research associated with soft computing includes areas such as Fuzzy Logic, Rough Set Theory, Neural Networks, Genetic Algorithms, Chaos and Fractals. We now need to advance a step further in this respect to introduce ‘intelligent’ or ‘thinking’ computer systems, which can be applied in various aspects of day-to-day life. Technology and science always go hand in hand; hence the need is felt to tap the potentials of the *science* of Soft Computing and to utilize its principles *technically* for the betterment of human existence.

Conventional computing techniques, i.e. hard computing techniques require precisely stated analytical models, which are valid for ideal situations. These also require a lot of computation time. Real life situations, on the other hand are non-ideal and more generic in nature; and thus they cannot be dealt with entirely by the hard computing techniques. Practical problems have a pervasive element of imprecision and uncertainty. Premises and guiding principles of hard computing include precision, certainty and rigor. Recognition problems (handwriting, speech, objects, images etc.), mobile robot coordination, forecasting, combinational problems etc do not lend themselves to precise solutions, thus introducing the need for soft computing.

As opposed to the conventional techniques, soft computing is tolerant to imprecision, uncertainty, partial truth and approximation. We can easily state that the role model for soft computing is the human mind. The basic principle behind soft computing is to utilize the tolerance for imprecision, uncertainty, partial truth and approximation to achieve tractability, robustness and low solution cost.

The basic ideas underlying soft computing in its current manifestation have links to many earlier influences; among them is Zadeh's 1965 paper on fuzzy sets. The inclusion of neural computing and genetic computing in soft computing came at a later point.

Soft Computing includes an emerging and more-or-less established family of problem stating and problem-solving methods that attempt to mimic the intelligence found in nature.

Some unique properties of Soft Computing include:

- Learning from experimental data
- Deriving their power of generalization from approximating or interpolating from these previously 'learned' inputs to produce outputs from unseen inputs
- Embedding existing structured human knowledge (experience, expertise, heuristics) into workable mathematics

At this point in time, the principal constituents of Soft Computing (SC) are Fuzzy Logic (FL), Neural Computing (NC), Evolutionary Computation (EC), Machine Learning (ML) and Probabilistic Reasoning (PR), with the latter subsuming belief networks, chaos theory and parts of learning theory. However, it is important to note that soft computing is not a concoction. It is rather a partnership in which each of the partners contributes a dissimilar methodology for addressing problems in its domain. In this standpoint, the principal constituent methodologies in SC are complimentary rather than competitive. In addition, soft computing may be viewed as a foundation component for the promising field of conceptual intelligence.

Current applications of Soft Computing include handwriting recognition, manufacture of automotive systems, image processing, data compression, decision-support systems, fuzzy control and neuro-fuzzy control. The successful applications of soft computing suggest that the impact of soft computing will be felt increasingly in the coming years in addressing issues of reducing 'cognitive load' of end user. The influence of soft

computing will eventually increase beyond science and engineering, providing the Machine Intelligence Quotient (MIQ) as the performance parameter of hardware/software systems. It represents a significant paradigm shift in the intentions of computing, which will reflect the fact that human mind, unlike present day computers, possesses the remarkable ability to store and process information lacking in categoricity.

## **5.2 Fuzzy Sets and Fuzzy Logic**

Fuzzy logic is a branch of Soft Computing which deals with a system which tolerant to imprecision, partial truth, uncertainty and approximation. It allows for a gradation of values instead of discrete values. The concept of Fuzzy Logic (FL) was conceived by Lotfi Zadeh, a professor at the University of California at Berkley, and presented not as a control methodology, but as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. In the context of Information Systems, Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control and decision-making. He proposed a comprehensive theory of approximate reasoning based on Fuzzy Logic in which truth-values are linguistic and the rules are expressed as fuzzy propositions. Approximate reasoning can be viewed as a process by which a possible imprecise conclusion is deduced from a collection of imprecise premises expressed in linguistic terms and fuzzy sets. Historically, FL applications were seen in the control system domain as successful commercial products. However, trends for FL applications in IT and computer science have been also reported in the literature. Professor Watanabe suggested Fuzzy Hardware Inference Engine chip and Fuzzy Microprocessor approach.

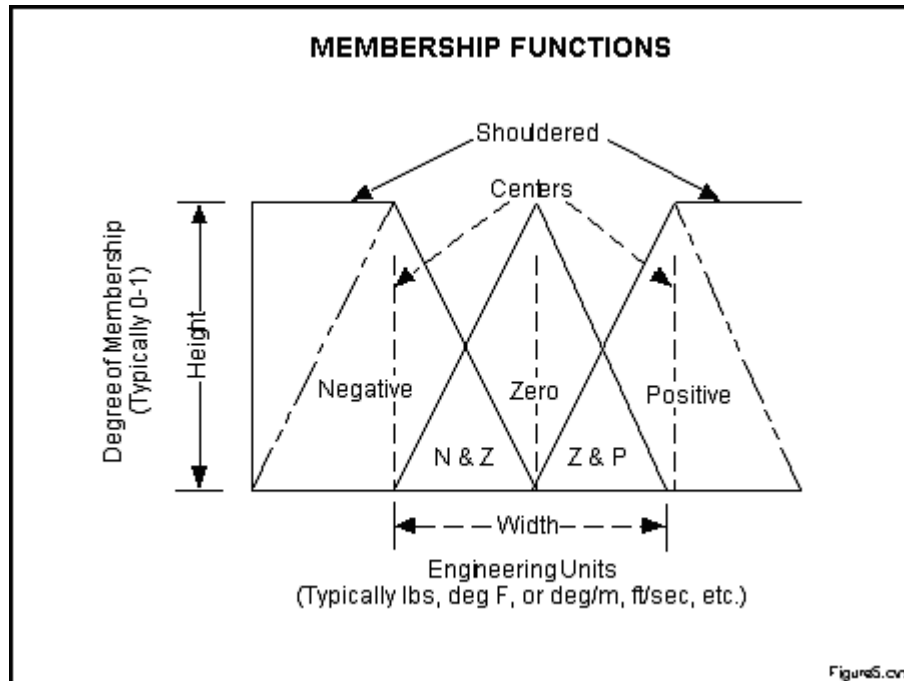
A fuzzy subset  $A$  of a (crisp) set  $X$  is characterized by assigning to each element  $x$  of  $X$  the *degree of membership* of  $x$  in  $A$  (e.g.  $X$  is a group of people,  $A$  the fuzzy set of *old* people in  $X$ ). Now if  $X$  is a set of propositions then its elements may be assigned their *degree of truth*, which may be “absolutely true,” “absolutely false” or some *intermediate* truth degree: a proposition may be more true than another proposition. This is obvious in the

case of vague (imprecise) propositions like “this person is old” (beautiful, rich, etc.). In the analogy to various definitions of operations on fuzzy sets (intersection, union, complement) one may ask how propositions can be combined by *connectives* (conjunction, disjunction, negation) and if the truth degree of a composed proposition is determined by the truth degrees of its components, i.e. if the connectives have their corresponding *truth functions* (like truth tables of classical logic). Saying “yes” (which is the mainstream of fuzzy logic) one accepts the truth-functional approach; this makes fuzzy logic to something distinctly *different from probability theory* since the latter is not truth-functional (the probability of conjunction of two propositions is *not determined* by the probabilities of those propositions).

Fuzzy sets are an extension of classical set theory and are used in fuzzy logic. In classical set theory the membership of elements in relation to a set is assessed in binary terms according to a crisp condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in relation to a set; this is described with the aid of a membership function . Fuzzy sets are an extension of classical set theory since, for a certain universe, a membership function may act as an indicator function, mapping all elements to either 1 or 0, as in the classical notion.

The theory of binary logic is based on the assumption of crisp membership of an element to a certain set. An element  $x$  thus either *belongs to* (i.e. has a membership value of 1) or *doesn't belong to* (i.e. has a membership value of 0) a particular set  $X$ . Conventional logic systems can be extended to encompass normalized values in the range of  $(0,1)$ , thus introducing the notion of *partial membership* of an element to a particular set. Such a logic system allows us to represent variables in a natural form with infinite degrees of membership is referred to as Fuzzy Logic System. The variable in a fuzzy system is generally described linguistically prior to its mathematical description, as it is more important to visualize a problem in totality to devise a practical solution.

Fuzzy membership functions can be illustrated as follows:



A fuzzy set  $F$ , on a collection of objects,  $X$ , is a mapping

$$\mu_F(x): X \rightarrow [0, a]$$

Here,  $\mu_F(x)$  indicates the extent to which  $x \in X$  has the attribute  $F$ , thus it is the *membership function*. In general, we use a normalized fuzzy domain set, for which

$$a = \sup \mu_F(x) = 1$$

The membership function can be generated with the help of mathematical equations. Typically, it can be in trapezoidal form, triangular form or in the form of S or  $\pi$ -curve.

The *support* of a fuzzy set,  $F$ ,  $S(F)$  is the crisp set of all  $x \in X$  such that  $\mu(x) > 0$ .

The three basic logical operations of intersection, union and complementation can be performed on fuzzy sets as well.

1. The membership  $\mu_C(x)$  of the *intersection*  $C = A \cap B$  satisfies for each  $x \in X$ ,

$$\mu_C(x) = \min \{ \mu_A(x), \mu_B(x) \}$$

2. The membership  $\mu_C(x)$  of the *union*  $C = A \cup B$  satisfies for each  $x \in X$

$$\mu_C(x) = \max \{\mu_A(x), \mu_B(x)\}$$

3. The membership  $\mu_C(x)$  of the *complementation*  $C = \bar{A}$  satisfies for each  $x \in X$ ,

$$\mu_C(x) = 1 - \mu_A(x)$$

Properties of classical sets are very important to consider because of their influence on the mathematical manipulation. Some of these properties are listed below.

*Commutativity:*

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

*Associativity:*

$$A \cup (B \cap C) = (A \cup B) \cap C$$

$$A \cap (B \cup C) = (A \cap B) \cup C$$

*Distributivity:*

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

*Idempotency:*

$$A \cup A = A$$

$$A \cap A = A$$

*Identity:*

$$A \cup \phi = A$$

$$A \cap U = A$$

$$A \cap \phi = \phi$$

$$A \cup U = U$$

Excluded middle laws are very important since they are the only set operations that are not valid for both classical and fuzzy sets. Excluded middle laws consist of two laws. The first, known as *Law of Excluded Middle*, deals with the union of a set  $A$  and its



complement. The second law, known as *Law of Contradiction*, represents the intersection of a set  $A$  and its complement.

The following equations describe these laws:

***Law of Excluded Middle***

$$A \cup \bar{A} = U$$

***Law of Contradiction***

$$A \cap \bar{A} = \phi$$

Some of the possible membership functions are:

- (a) the  $\Gamma$ -function: an increasing membership function with straight lines;
- (b) the L-function: a decreasing function with straight lines;
- (c)  $\Lambda$ -function: a triangular function with straight lines;
- (d) the singleton: a membership function with a membership function value 1 for only one value and the rest is zero.
- (e) There are many other possible functions such as trapezoidal, Gaussian, sigmoidal or even arbitrary. These are much more popular.

**Alpha-Cut Fuzzy Sets**

It is the crisp domain in which we perform all computations with today's computers.

Given a fuzzy set  $\underline{A}$ , the alpha-cut (or lambda cut) set of  $\underline{A}$  is defined by

$$A_\alpha = \{x \in U \mid \mu_{\underline{A}}(x) \geq \alpha\}$$

Note that by virtue of the condition on  $\mu_{\underline{A}}(x)$  in above equation, i.e., a common property, the set  $A_\alpha$  is now a crisp set. In fact, any fuzzy set can be converted to an infinite number of cut sets.

### Extension Principle

In fuzzy sets, just as in crisp sets, one needs to find means to extend the domain of a function, i.e., given a fuzzy set  $\tilde{A}$  and a function  $f(\cdot)$ , then what is the value of function  $f(\tilde{A})$ ? This notion is called the *extension principle*.

Let the function  $f$  be defined by

$$f:U \rightarrow V$$

where  $U$  and  $V$  are domain and range sets, respectively. Define a fuzzy set  $\tilde{A} \subset U$  as,

$$\tilde{A} = \left\{ \frac{u_1}{\mu_1} + \frac{u_2}{\mu_2} + \frac{u_n}{\mu_n} \right\}$$

Then the extension principle asserts that the function  $f$  is a fuzzy set, as well, which is defined below:

$$\tilde{B} = \left\{ \frac{v_1}{\mu_1} + \frac{v_2}{\mu_2} + \frac{v_n}{\mu_n} \right\}$$

The complexity of the extension principle would increase when more than one member of  $u_1 \times u_2$  is mapped to only one member of  $v$ ; one would take the maximum membership grades of these members in the fuzzy set  $\tilde{A}$ .

### Hedges

The linguistic *hedge* is an operation that modifies the meaning of a term or more generally, of a fuzzy set. The distribution of membership of the membership function as indicated above can be modified so that the concept captured by the modified fuzzy term is stronger (concentrated) or weaker (dilated) than the original term. If  $A$  is a fuzzy set then the modifier  $m$  generates the (composite) term  $B = m(A)$ . The linguistic hedge comprises concentration and dilation. These are discussed below:

Concentration:

The operation of concentration on F set results in a fuzzy subset of F that the reduction in the magnitude of the grade of membership of an element in F is relatively small for those values which have high membership and relatively large for those which have low membership. The operation of concentration is defined by:

$$CON(F) = F^p$$

$$\mu_{con, A}(x) = (\mu_A(x))^p, \text{ where } p > 1$$

Dilation:

The operation of dilation on F set results in a fuzzy subset of F that the increase in the magnitude of the grade of membership of an element in F is relatively small for those values which have high membership and relatively large for those which have low membership. The operation of dilation is defined by:

$$DIL(F) = F^q$$

$$\mu_{dil, A}(x) = (\mu_A(x))^q, \text{ where } q < 1$$

Defuzzification is the process of producing a quantifiable result in fuzzy logic. Typically, a fuzzy system will have a number of rules that transform a number of variables into a "fuzzy" result, that is, the result is described in terms of membership in fuzzy sets. For example, rules designed to decide how much pressure to apply might result in "Decrease Pressure (15%), Maintain Pressure (34%), Increase Pressure (72%)". Defuzzification would transform this result into a single number indicating the change in pressure. The simplest but least useful defuzzification method is to choose the set with the highest membership, in this case, "Increase Pressure" since it has a 72% membership, and ignore the others, and convert this 72% to some number. The problem with this approach is that

it loses information. The rules that called for decreasing or maintaining pressure might as well have not been there in this case.

A useful defuzzification technique must first add the results of the rules together in some way. The most typical fuzzy set membership function has the graph of a triangle. Now, if this triangle were to be cut in a straight horizontal line somewhere between the top and the bottom, and the top portion were to be removed, the remaining portion forms a trapezoid. The first step of defuzzification typically "chops off" parts of the graphs to form trapezoids (or other shapes if the initial shapes were not triangles). For example, if the output has "Decrease Pressure (15%)", then this triangle will be cut 15% the way up from the bottom. In the most common technique, all of these trapezoids are then superimposed one upon another, forming a single geometric shape. Then, the centroid of this shape, called the *fuzzy centroid*, is calculated. The  $x$  coordinate of the centroid is the defuzzified value.

$$\text{Centroid}_{\text{Fuzzy}} = (\sum d_i w_i) / (\sum w_i)$$

#### **T-norm:**

A T-norm is a function  $T: [0, 1] \times [0, 1] \rightarrow [0, 1]$  which satisfies the following properties:

- Commutativity:  $T(a, b) = T(b, a)$
- Monotonicity:  $T(a, b) \leq T(c, d)$  if  $a \leq c$  and  $b \leq d$
- Associativity:  $T(a, T(b, c)) = T(T(a, b), c)$
- The number 1 acts as identity element:  $T(a, 1) = a$

Minimum T-norm is defined by:

$$\boxed{T_{\min}(a, b) = \min\{a, b\}}$$

Product T-norm is defined by:

$$\boxed{T_{\text{prod}}(a, b) = a \cdot b}$$

### **T-Conorm (S-norm):**

- Dual to T-norms under the order-reversing operation which assigns  $1 - x$  to  $x$  on  $[0, 1]$
- Generalizes De Morgan's laws
- Given a T-norm, the complementary Conorm is defined by:

$$\perp(a, b) = 1 - T(1 - a, 1 - b)$$

Maximum T-conorm is defined by :

$$\perp_{\max}(a, b) = \max\{a, b\}$$

Probabilistic sum T-conorm is defined by:

$$\perp_{\text{sum}}(a, b) = a + b - a \cdot b$$

### **Fuzzy Control System**

FL offers several unique features that make it a particularly good choice for many control problems:

1. It is inherently robust since it does not require precise, noise-free inputs and can be programmed to fail safely if a feedback sensor quits or is destroyed. The output control is a smooth control function despite a wide range of input variations.
2. Since the FL controller processes user-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance. New sensors can easily be incorporated into the system simply by generating appropriate governing rules.
3. FL is not limited to a few feedback inputs and one or two control outputs, nor is it necessary to measure or compute rate-of-change parameters in order for it to be implemented. Any sensor data that provides some indication of a system's actions and reactions is sufficient. This allows the sensors to be inexpensive and imprecise thus keeping the overall system cost and complexity low.

4. Because of the rule-based operation, any reasonable number of inputs can be processed (1-8 or more) and numerous outputs (1-4 or more) generated, although defining the rule base quickly becomes complex if too many inputs and outputs are chosen for a single implementation since rules defining their interrelations must also be defined. It would be better to break the control system into smaller chunks and use several smaller FL controllers distributed on the system, each with more limited responsibilities.
5. FL can control nonlinear systems that would be difficult or impossible to model mathematically. This opens doors for control systems that would normally be deemed unfeasible for automation.

Fuzzy Logic Control (FLC) is an algorithm. This develops a process control as fuzzy relation information on the condition of the process to be controlled and the control action. The essence of fuzzy control algorithm is a conditional statement between a fuzzy input variable B.

This is expressed by a linguistic implication statement such as :-

**A → B** (condition A implies condition B),

This may be written as

**IF A THEN B**

There is an equivalency between this expression and the relation obtained by a Cartesian multiplication i.e.

**R=A \* B ≡ IF A THEN B.**

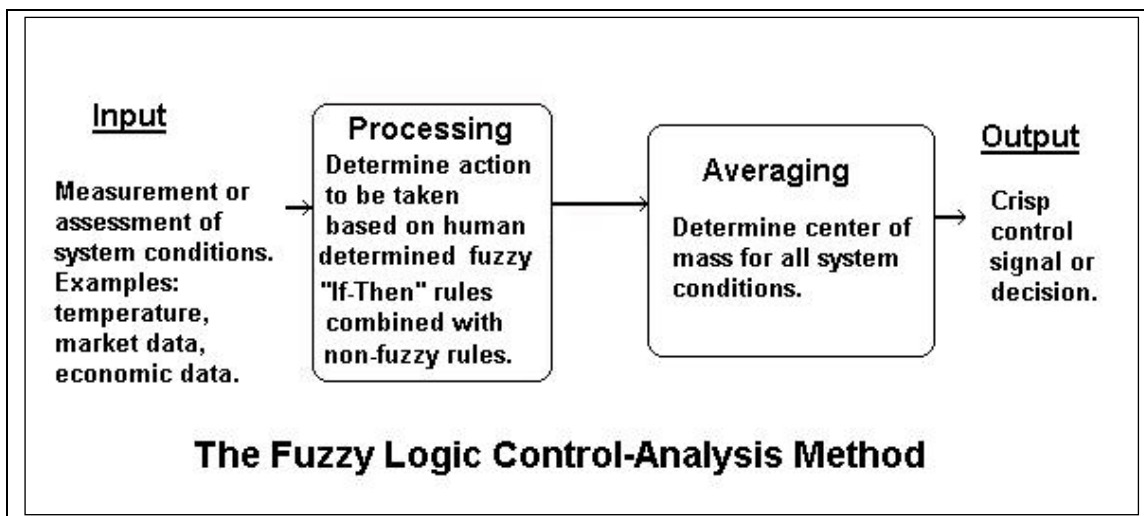
A fuzzy variable is expressed through a fuzzy set which in turn is defined by a membership function  $\mu$ . The fuzzy variable may be continuous or discrete. A continuous variable can be quantized and expressed as if it were discrete.

Fuzzy production rules are used for knowledge representation. The general formulation of fuzzy production rule is

**If  $F_i$  (CF = x) then  $C_i$  (CF = y)**

Where  $F_i$  represents the antecedent portion of the rule containing fuzzy quantifiers and  $C_i$  represents the consequent. The CF values give the confidence measures usually varying from 0 to 1.

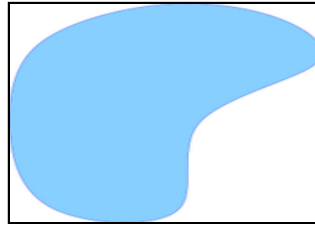
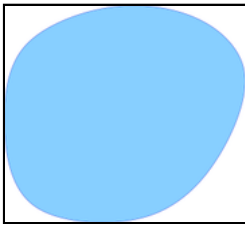
The Fuzzy Control Logic process, described above, can be illustrated as shown in the following diagram:



### 5.3 Fuzzy Numbers

A **fuzzy number** is a **convex, normalized** fuzzy set  $\tilde{A} \subseteq \mathbb{R}$  whose membership function is at least segmentally **continuous** and has the functional value  $\mu_A(x) = 1$  at precisely one element. This can further be explained by the following terms in relation to fuzzy numbers :

- **Convex:** In Euclidean space, an object is convex if for every pair of points within the object, every point on the straight line segment that joins them is also within the object. For example, a solid cube is convex, but anything that is hollow or has a dent in it is not convex. This can be explained by the figure given below comparing the convex and non-convex shapes.



CONVEX FUNCTION

NONCONVEX FUNCTION

- **Normalized:** A normalized set is one in which values are fixed within a specific range. For example, 0 to 1 instead of random variations.
- **Continuous:** In mathematics, a continuous function is a function for which, intuitively, small changes in the input result in small changes in the output.
- **Fuzzy:** Being part of the fuzzy set is what makes a fuzzy number an ordinary number whose precise value is somewhat uncertain. Thus being fuzzy allows it to have approximations.

A fuzzy number is a thus quantity whose value is imprecise, rather than being exact like single-valued numbers. Any fuzzy number can be thought of as a function whose domain is a specified set (usually the set of real numbers, and whose range is the span of non-negative real numbers between, and including, 0 and 1000. Each numerical value in the domain is assigned a specific "grade of membership" where 0 represents the smallest possible grade, and 1000 is the largest possible grade. In many respects, fuzzy numbers depict the physical world more realistically than single-valued numbers.

The number has three main properties, the minimum possible value of fuzzy number are called Minimum, the maximum possible value - Maximum, and most possible value –



Best. The peak, minimum, and maximum, also describe the apex, left corner and right corner of the membership function

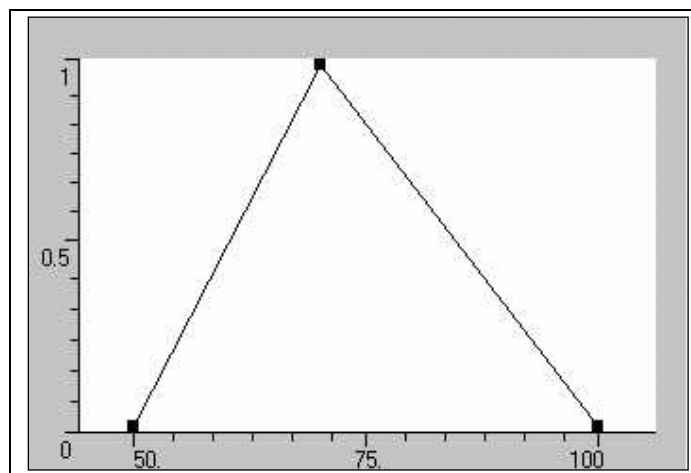
Fuzzy numbers can also be called fuzzy sets on which certain restrictions and distinctive denotations have been applied:

For fuzzy set  $A$ ,

1.  $A(x)$  is a function mapping , often a subset of  $\mathbb{R}$ , into  $[0,1]$
2. The value of  $A(x)$  is called its *membership value* in  $A$ , denoted by  $\alpha$
3. For some  $x \in \Omega$ ,  $\bar{A}(x)=1$ ; that is, fuzzy numbers are *normalized* to 1. The set  $\{x \mid \bar{A}(x)=1\}$  is called the *core*. If  $x$  is a singleton it may be called a *vertex*.
4. The membership function  $A$  must have  $C0$  continuity; i.e., be connected, have no breaks.
5. The interval for which  $A(x) > 0$ , say  $[a,c]$ ,  $a < \text{the core} < c$ , is called the *support* of the fuzzy number. In this case,  $a$  ( $c$ ) is called the left (right) support. Some authors allow the support to be an open interval. The membership function from  $a$  to the core (the core to  $c$ ) must be monotonically increasing (decreasing).

There are two main ways of representing fuzzy numbers:

1. By creating a set of pairs
2. By using a set of Belief graphs. For example, the fuzzy number  $Z$  could have the following Belief graph:



Each fuzzy number has a centroid. It is determined as the balancing point of Belief graph. Centroid is used to display fuzzy number in the sheet cell.

When working with fuzzy numbers and performing fuzzy arithmetic, we should use a large universal space because the intervals over which fuzzy numbers are defined, widen as arithmetic operations are performed. Also, like traditional numbers, fuzzy numbers can be negative or positive, so the universal space should be symmetric around zero.

### **Hedges and Fuzzy Numbers**

Fuzzy numbers are most easily specified by using modifying words called hedges. For example:

nearly 2 = 2 +/- 5%
about 2 = 2 +/- 10%
roughly 2 = 2 +/- 25%
crudely 2 = 2 +/- 50%

All these dispersions are measured at the 50% confidence level. That is, we are at least 50% sure that a number belongs to about 2 with a confidence of 500 or greater if it lies within 10% of 2, or between 1.8 and 2.2.

So far we have two ways of representing uncertainties: by confidence levels attached to a specific datum, such as the value of some real-world variable; and by the use of fuzzy sets of word descriptors. We now consider how to represent an uncertain number.

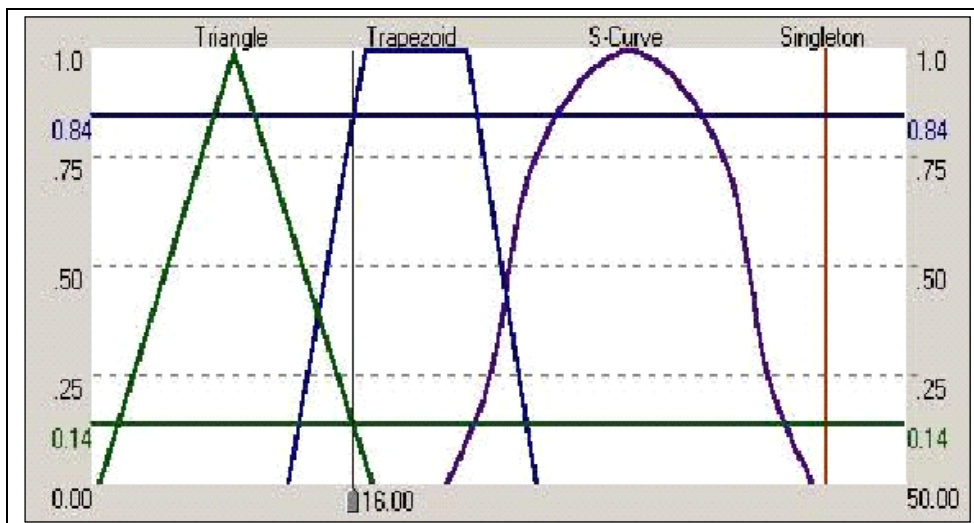
Suppose a fuzzy set is given members as all the real numbers. Since there is infinity of real numbers, this fuzzy set has infinity of members. As in all fuzzy sets, to each real number there will be attached a grade of membership.

There are many cases when an input number may be only approximately known, or may be subject to uncertainties even if a precise value is available. In these cases, fuzzy numbers are useful. Like membership functions, fuzzy numbers have shape. There are many cases when an input number may be only approximately known, or may be subject to uncertainties even if a precise value is available.

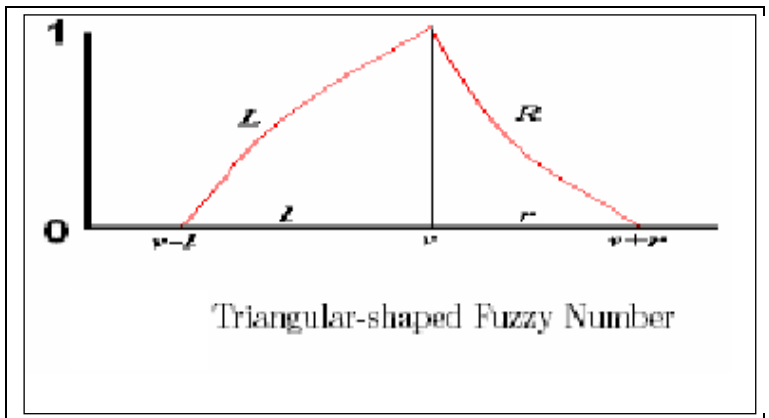
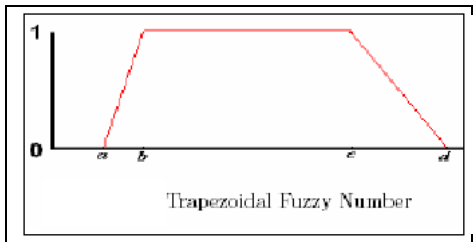
Fuzzy numbers are also useful when we believe that the true value of a decimal number falls within a known range, for example +/- 10%.

Like membership functions, fuzzy numbers have shape restrictions. Usually, the confidence is zero in small numbers, rises to full confidence, and then declines again to zero. (Technically, such numbers are called convex.) There may also be special fuzzy numbers which start or end up with full confidence, as in some membership functions; this is reasonable.

A fuzzy number can be linear, curvilinear shaped or bell shaped. This is illustrated in the figure comparing the various shapes:

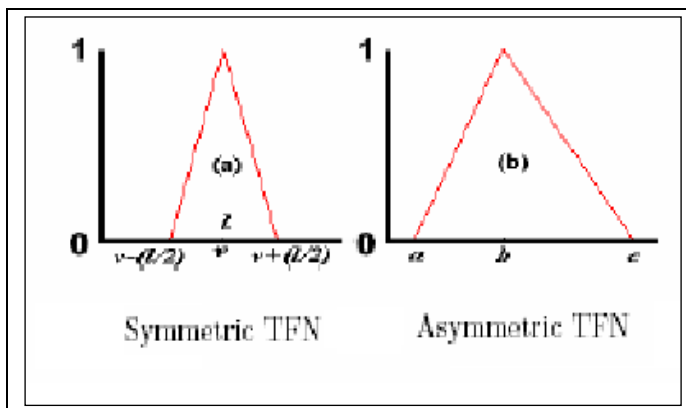


Trapezoidal and Triangular functions for fuzzy numbers are separately, being more popular as compared to other functions in case of fuzzy numbers.



Many functions, like triangular, can further be classified as:

1. Symmetric functions and,
2. Asymmetric functions



As shown above, some fuzzy number representations can be triangular and trapezoidal. Triangular can be symmetric or asymmetric. In symmetric representation,  $(v, l)$  are taken where  $v$  gives the position of the vertex and  $l$  gives the support distance where perpendicular bisects it. A notation for asymmetric is  $(a/b/c)$  where  $b$  is the vertex and  $(a, c)$  gives the support (i.e. the base of the triangle). If the core (like vertex  $b$ ) is not a single value, then it gives rise to trapezoidal fuzzy numbers. It uses the notation  $(a/b/c/d)$  with  $[a, d]$  as support and  $[b, c]$  as base.

Fuzzy numbers define various concepts using partial membership functions. Amongst various membership functions, most often used in this case are, triangular and parabolic. These are represented by  $(\alpha, m, \beta)$  where  $m$  is the modal value and  $\alpha$  and  $\beta$  represent the left and right boundary values.

*Triangular Fuzzy Numbers:*

A triangular fuzzy number (TFN)  $K$ , is described by a triplet  $\{\alpha, m, \beta\}$  where  $m$  is the modal value of the fuzzy number  $K$ , and  $\alpha$  and  $\beta$  are its left and right boundary values, respectively.  $K = \text{TFN} \{\alpha, m, \beta\}$

*Parabolic Fuzzy Numbers:*

Parabolic fuzzy number (PFN) is composed of two parabolic segments of the membership grades delimited by modal and boundary values. It is described by a triplet PFN  $\{\alpha, m, \beta\}$  with modal value  $m$ , and parameters of the left and right boundaries  $\alpha$  and  $\beta$ .  $K = \text{PFN} \{\alpha, m, \beta\}$

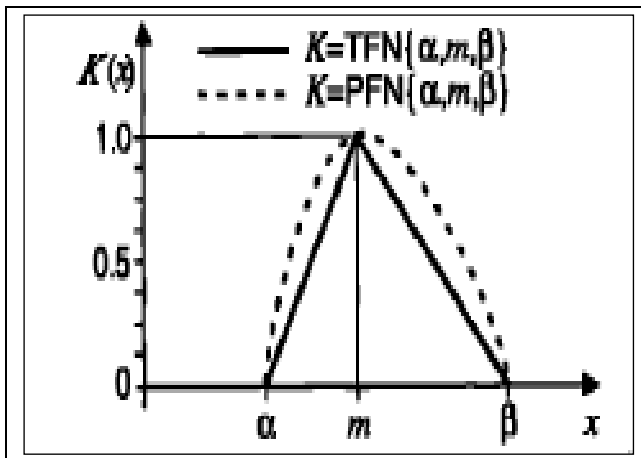
Equations of the Triangular Fuzzy Number:

$$K(x) = \begin{cases} \frac{x-\alpha}{m-\alpha} & \text{for } x \in [\alpha, m] \\ \frac{\beta-x}{\beta-m} & \text{for } x \in [m, \beta] \\ 0 & \text{for } x \notin [\alpha, \beta] \end{cases}$$

Equation of the Parabolic Fuzzy Number:

$$K(x) = \begin{cases} 1 - \frac{(x-\alpha)^2}{(m-\alpha)^2} & \text{for } x \in [\alpha, m] \\ 1 - \frac{(\beta-x)^2}{(\beta-m)^2} & \text{for } x \in [m, \beta] \\ 0 & \text{for } x \in [\alpha, \beta] \end{cases}$$

Figure Showing a TFN and a PFN:



Fuzzy numbers are special fuzzy sets representing uncertain quantitative information. They are convex and normal, usually with single modal value. They are also associated with some vagueness or *fuzziness*. For evaluation of uncertainty, relative fuzziness is defined as:

$$f(A) = \frac{\int_{\alpha}^{\beta} A(x) dx}{m}$$

where more the relative fuzziness, more uncertain A is.

Computations involving fuzzy numbers are carried out in the setting of *fuzzy arithmetic*. It dwells on the extension principle. The extension principle deals with the generalization of this mapping to the case of arguments being fuzzy numbers that is  $B = f(A)$ , where A and B are fuzzy numbers.

If  $C = f(A, B)$  then its membership function can be defined by:

$$C(z) = \sup_{x,y \in \mathbb{R}: z=f(x,y)} [\min(A(x), B(y))]$$

### Comparing Fuzzy Numbers

In computer programming, it is quite common to compare two numbers in various ways. For example, we might want to know if some input number X equals some certain value: in almost any computer language, we can write an instruction like this:

If x = 200 then (do something)

Similarly, we can write a rule to do the same thing:

Rule: IF x = 200 THEN (do something)

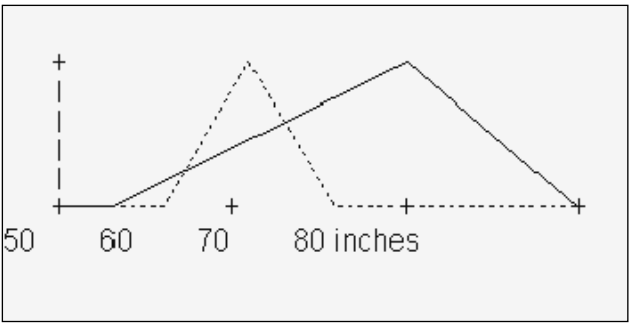
In both cases, if x is just slightly off (say x is 199.99) the instruction will not be executed and the rule will not fire.

In fuzzy reasoning, we are often not concerned with precise equality, but would like to be able to say

IF x is approximately 200 THEN (do something)

The use of fuzzy numbers permits us to do this. This can be done by using approximate numerical comparisons like:  $\sim <$  (approximately less than),  $\sim < =$  (approximately less than or equal to),  $\sim =$  (approximately equal) and so on..

Suppose we are comparing two fuzzy numbers, as shown in figure:



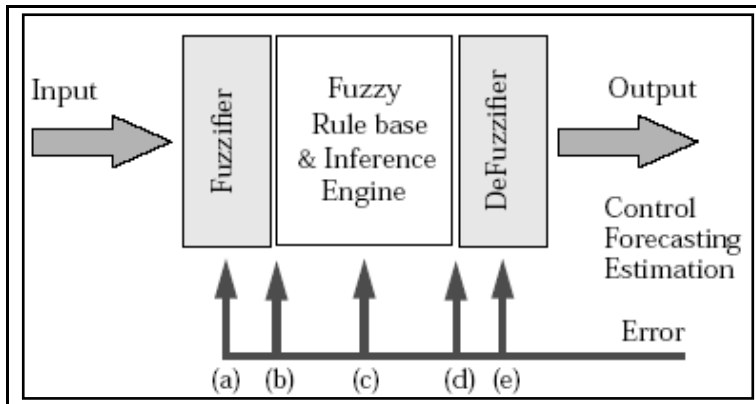
We see that the two curves cross at two points. The confidence that the comparison holds is the greater of the confidences at these intersection points, in this case 600.

## **5.4 Fuzzy Inference**

Applications of Fuzzy Logic in simulation and modeling are based on fuzzy inference mechanism. A Fuzzy Inferencing System (FIS) uses fuzzy sets to make decisions or draw conclusions. Assuming that there is a particular problem that cannot (at all or with difficulty) be tackled by conventional methods such as by developing a mathematical model, after some process (e.g. knowledge acquisition from an expert in the domain) the 'base' fuzzy sets that describe the problem are determined. These might be, for example, how a doctor describes the temperature of a patient as low, normal or high and that the patient's temperature is a factor in the diagnostic process. The rules (usually of an IF...THEN... nature (if-then)) are thus determined. These rules then have to be combined in some way referred to as *rule composition*. Finally conclusions have to be drawn - *defuzzification*.

A general fuzzy inference system consists of three parts. A crisp input is fuzzified by input membership functions and processed by a fuzzy logic interpretation of a set of fuzzy rules. This is followed by the defuzzification stage resulting in a crisp output. The rule base is typically crafted by an expert.





An approach that we can define for a Fuzzy Inferencing System as:

- the *base fuzzy sets* that are to be used, as defined by their membership functions;
- the *rules* that combine the fuzzy sets;
- the fuzzy *composition* of the rules;
- the *Defuzzification* of the solution fuzzy set.

There are two types of fuzzy inference systems that can be implemented:

- Mamdani-type
- Sugeno-type

*Mamdani-type inference*, as defined for Fuzzy Logic Toolbox, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It is possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. This type of output is sometimes known as a *singleton* output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, you use the weighted average of a few data points.

*Sugeno-type systems* support this type of model. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.

Before concluding this chapter, it is necessary to discuss some misconceptions about Fuzzy Logic. These have been highlighted below.

(i) Fuzzy logic is the same as "imprecise logic"

Fuzzy logic is not any less precise than any other form of logic: it is an organized and mathematical method of handling *inherently* imprecise concepts. The concept of "coldness" cannot be expressed in an equation, because although temperature is a quantity, "coldness" is not. However, people have an idea of what "cold" is, and agree that something cannot be "cold" at N degrees but "not cold" at N+1 degrees — a concept classical logic cannot easily handle due to the principle of bivalence.

(ii) Fuzzy logic is a new way of expressing probability

Fuzzy logic and probability refer to different kinds of uncertainty. Fuzzy logic is specifically designed to deal with imprecision of facts (fuzzy logic statements), while probability deals with chances of that happening (*but still considering the result to be precise*). However, this is a point of controversy. Many statisticians are persuaded by the work of Bruno de Finetti that only one kind of mathematical uncertainty is needed and thus fuzzy logic is unnecessary. On the other hand, Bart Kosko argues that probability is a sub theory of fuzzy logic, as probability only handles one kind of uncertainty. He also claims to have proven a theorem demonstrating that Bayes' theorem can be derived from the concept of fuzzy subethood. Lotfi Zadeh, the creator of fuzzy logic, argues that fuzzy logic is different in character from probability, and is not a replacement for it. He has created a fuzzy alternative to probability, which he calls possibility theory. Other controversial approaches to uncertainty include Dempster-Shafer theory and rough sets.

(iii) Fuzzy logic will be difficult to scale to larger problems

In a widely circulated and highly controversial paper, Charles Elkan in 1993 commented that "*...there are few, if any, published reports of expert systems in real-world use that reason about uncertainty using fuzzy logic. It appears that the limitations of fuzzy logic have not been detrimental in control applications because current fuzzy controllers are*

*far simpler than other knowledge-based systems. In future, the technical limitations of fuzzy logic can be expected to become important in practice, and work on fuzzy controllers will also encounter several problems of scale already known for other knowledge-based systems".* Reactions to Elkan's paper are many and varied, from claims that he is simply mistaken, to others who accept that he has identified important limitations of fuzzy logic that need to be addressed by system designers. In fact, fuzzy logic wasn't largely used at that time, and today it is used to solve very complex problems in the AI area. Probably the scalability and complexity of the *fuzzy* system will depend more on its implementation than on the theory of fuzzy logic.

## **CHAPTER 6**

---

### **FUZZY SOFTWARE ESTIMATION FRAMEWORK**

---

#### **6.1 Introduction**

In the previous chapters, we have seen the need to provide support for cost estimation models to handle uncertainties and vagueness arising out of several practices used in software development process. In our project, Software Effort Estimation and Design is developed using classical and fuzzy models. During the course of our project, we went through various papers and articles so as to select the best possible methodology for implementing Fuzzy Software Estimation. The fuzzy models implemented in our software are based on the models established and proposed in these papers. Hence, we are providing a short overview of the ideas presented by various authors which have been used by us in the **fuzzy Software Estimation Environment (f-SEE)** package.

#### **PAPER 1:**

##### **Estimation of f-COCOMO Model Parameters Using Optimization Techniques**

Leonard J. Jowers, James J. Buckley, and Kevin D. Reilly

#### **Summary**

The paper is concerned with improving the project outcome using COCOMO. It allows for uncertainty by use of fuzzy logic. We start with a crisp COCOMO model which depends on interpretation of a set of linguistic variables to create a set of crisp parameters. There are always a number of parameters in the system whose values are not known precisely. These parameters need to be estimated and their estimators contain uncertainties. We model these uncertainties using fuzzy numbers and fuzzy arithmetic. It is primarily based of the principle of fuzzy extension which effective in COCOMO estimation. Thus the COCOMO model changes into a fuzzy COCOMO model. This uncertainty in the parameter values computes a larger uncertainty in the COCOMO result.

If a project parameter is fuzzy, the associated COCOMO Model also becomes a fuzzy COCOMO Model (f-COCOMO Model) with a fuzzy result i.e: schedule and effort. A dictated fuzzy schedule and budget can be used to improve an f-COCOMO Model, or plan a project. By application of constraints created by dictated fuzzy results, and back propagation, better estimates of project parameters are obtainable. Such a project scenario is presented in this paper and the method is applied to demonstrate its use.

There is more than one way in which fuzziness occurs in computation:

- perception-based (linguistic) and
- measurement-based (numerical) fuzziness.

### **Fuzzy Numbers**

Some fuzzy number representations can be triangular and trapezoidal. Triangular can be symmetric or asymmetric. In symmetric representation, (v, l) are taken where v gives the position of the vertex and l gives the support distance where perpendicular bisects it. A notation for asymmetric is (a/b/c) where b is the vertex and (a, c) gives the support (i.e. the base of the triangle). If the core (like vertex b) is not a single value, then it gives rise to trapezoidal fuzzy numbers. It uses the notation (a/b/c/d) with [a, d] as support and [b, c] as base.

Fuzzy arithmetic is primarily based on two methods:

#### 1. Extension Principle

If A and B are two fuzzy numbers, then C is calculated as the supremum of the min of these two. This is shown in the given equations:

$$\bar{C} = \bar{A} + \bar{B} : \bar{C}(z) = \sup_{x,y} \{ \min(\bar{A}(x), \bar{B}(y)) \mid x + y = z \} .$$

$$\bar{C} = \bar{A} - \bar{B} : \bar{C}(z) = \sup_{x,y} \{ \min(\bar{A}(x), \bar{B}(y)) \mid x - y = z \} .$$

$$\bar{C} = \bar{A} \cdot \bar{B} : \bar{C}(z) = \sup_{x,y} \{ \min(\bar{A}(x), \bar{B}(y)) \mid x \cdot y = z \} ,$$

#### 2. Interval arithmetic on alpha cuts.

Alpha-cuts are slices through a fuzzy set producing crisp (non-fuzzy) sets.

Given a fuzzy set  $\tilde{A}$ , the alpha-cut (or lambda cut) set of  $\tilde{A}$  is defined by

$$A_\alpha = \{x \mid \mu_{\tilde{A}}(x) \geq \alpha\}$$

Note that by virtue of the condition on  $\mu_{\tilde{A}}(x)$  in above equation, i.e., a common property, the set  $A_\alpha$  is now a crisp set. In fact, any fuzzy set can be converted to an infinite number of cut sets.

Alpha cut sets are calculated for the following equations:

$$\overline{C} = \overline{A} + \overline{B}: \overline{C}[\alpha] = [c_L, c_R] = [a_L, a_R] + [b_L, b_R] = [a_L + b_L, a_R + b_R].$$

$$\overline{C} = \overline{A} - \overline{B}: \overline{C}[\alpha] = [c_L, c_R] = [a_L, a_R] - [b_L, b_R] = [a_L - b_R, a_R - b_L].$$

### Fuzzy COCOMO

COCOMO II being more popular, is taken into consideration in the paper. the basic equations are given as follows:

$$PM = A \times Size^E \times \prod_{i=1}^n EM_i + PM_{Auto},$$

$$E = B + 0.01 \times \sum_{j=1}^5 SF_j.$$

In this case, fuzzification is done for all parameters which are inexactly known using the TFN values. Another concept implemented in that of inverse problem.

### Fuzzy Estimators

In this paper only one method of fuzzy estimation that is expert opinion is considered. Then a value “b” is obtained from experts. This is then fuzzified using triangular fuzzy number concept. It is assumed that  $b_1$  is the pessimistic value,  $b_3$  is the optimistic value and  $b_2$  is the most likely value. Thus a TFN is constructed by using  $b=(b_1 /b_2 /b_3 )$  for b. This is one approach towards fuzzy estimators.

### The Inverse Problem

It can be explained as:

“Given an effort budget, where should resources be directed to increase possibility of staying within budget?”

A research based on Monte Carlo method is done for this problem. And additional evaluations are carried out by a defined defuzzification scheme. The Monte Carlo method takes as input an incomplete set of parameters (linguistic) and the target fuzzy solution (ie. the effort and the development time).It then optimizes them in such a way so as to resolve the unknown fuzzy parameters and make a complete set available.

## PAPER 2

### On the Use of Fuzzy Regression in Parametric Software Estimation Models: Integrating Imprecision in COCOMO Cost Drivers

F. Javier. Crespo, Miguel-Ángel Silicia, Juan J. Cuadrado

#### Summary

Parametric software estimating models are based on inherently imprecise and uncertain input variables. These use mathematical models elaborated from regression techniques to obtain effort of development estimates. In this paper, preliminary results on using fuzzy inputs to f-regression have been reported. It shows that fuzzy regression is able to obtain estimation models with similar predictive properties than existing basic estimation models.

The limitations of COCOMO in crisp set evaluation arises due to the imprecision of the limited set of labels and the uncertainty of the human approximate judgments about abstract concepts of the various parameters. In this paper the concept of fuzzy regression is shown to overcome the limitations. Fuzzy regression analysis approaches can be roughly categorized in two groups:

1. Classical fuzzy regression –it is based on the assumption that deviations are due to the fuzziness of parameters
2. Fuzziness in the experimental points while using a crisp model.

In this paper, the second method is implemented. It takes into consideration the imperfection in the input assessment.

The standard COCOMO is given below:

$$e = a \cdot size^b \cdot M$$

where e is the effort,

a and b are constants

and M is the product of the cost drivers ranked on linguistic ordinal scales,

it can be given as:

$$M = \prod_{i=1}^{15} c_i$$



This paper takes into account the kind and impact of uncertainty of each of these cost drivers and calculates an upper bound error for them. Only an upper bound is calculated because of diversity of imperfection of different parameters.

The *f-regression* method is based on obtaining a coefficient vector  $a$  for a given crisp model, assuming that experimental points (both inputs and actual outputs) are modeled as fuzzy numbers  $\mu_{Q_i}(x_i, y)$  modeled as the product *T-norm* of the fuzzy numbers representing inputs and outputs. Then, a similarity measure  $M_i(a)$  with function  $f$  (for coefficient vector

$a$ ) for a given fuzzy point  $Q_i$  is defined as follows:

$$M_i(a) = \sup[\mu_{Q_i}(x, f(x, a))]$$

In addition, an aggregation operator  $M$  is used to compute the similarity of a set of fuzzy points to a given coefficient vector for function  $f$ . Its aggregation operation is given as follows:

$$MA(a) = \frac{1}{N} \sum_i M_i(a)$$

The optimal coefficient vector  $a^*$  is obtained by random search techniques as described by above equation.

Thus, the same ordinal linguistic scale for cost drivers is mapped to different non-regular intervals that suggest a loss of information when casting input variables to the numbers.

Another limitation of this paper lies in requiring further research in fuzzy formulation of input values, which may eventually result in more accurate models, e.g. cost drivers selected in COCOMO are heterogeneous in uncertainty terms, so that different forms of membership functions would be required for a more realistic modeling.

## PAPER 3

### f-COCOMO : Fuzzy Constructive Cost Model in Software Engineering

Fei Zonglian, Liu Xihui

#### Summary

In this paper, the fuzzy Constructive Cost Model (f-COCOMO) is presented. This involves overcoming the limitations of the existing COCOMO and making it more tolerant to imprecision using a domain of Soft Computing called Fuzzy Logic. It shows how allowing for uncertainties leads to improved decision making. It provides a reasonable estimation of manpower and development time.

Though the analysis methodology of COCOMO takes the uncertainties of randomness into consideration; yet, the uncertainty of fuzziness involved in the process of analysis is not taken into account. In these traditional COCOMO models, linguistic values are used for representation and user convenience but for calculations, these are converted to numerical values. These are crisp values. But for linguistic values, use of fuzzy numbers is generally more efficient.

#### Basic COCOMO implementation

It gives an equation (1) for estimating the number of Man-Months (MM) required for a specified software product in terms of the number of thousands of delivered source instructions (KDSI) and equation (2) for calculating the required development time.

$$MM = 2.4 (KDSI)^{1.05} \quad (1)$$

$$TDEV = 2.5 (MM)^{0.38} \quad (2)$$

These equations can be transformed into fuzzy ones to ease the process of decision making. This is done by using relational algebra. This is done by first providing intervals of values instead of discrete values and then defining the exponential operation for those intervals.

This is shown in the equation below:

$$I^P = [a^P, b^P]$$

### **Intermediate COCOMO implementation**

The intermediate COCOMO has greater accuracy and is suitable for cost estimation in the more detailed stages compared to Basic COCOMO. In this case, 15 additional are defined. Instead of assigning discrete values to the natural language values of these parameters, they are implemented using intervals in fuzzy logic. This thus led to fuzzy adjustment factor.

### **Fuzzy Decision Making**

In this paper, a comprehensive evaluation system is provided as shown below:

1. Factor set  $U = \{u_1, u_2, \dots, u_n\}$ , where  $u_i$  ( $i=1, 2, \dots, n$ ) are associated factors involved.
2. Evaluation set  $V = \{v_1, v_2, \dots, v_m\}$ , where  $v_j$  ( $j=1, 2, \dots, m$ ) represent the results of evaluation of the system.
3. Fuzzy relationship matrix (R)
4. Weight Set (a)

Comprehensive evaluation

$$b = a * R,$$

Where \* gives the composition operator.

“b” is the result of the evaluation and  $b_j$  gives the degree of membership.

In the given paper, this concept is explained by taking an example of selection of powerful computer. Here “b” is evaluated and it is verified that the result is more reasonable due to fuzzy application.

## **PAPER 4**

### **FULSOME**

Stephen G. MacDonell, Andrew R. Gray, and James M. Calvert

#### **Summary**

This paper discusses the benefits of using fuzzy logic modeling for software metric models. Fuzzy Logic has joined both traditional and robust statistical techniques, regression and classification trees, case based reasoning and neural networks, model based attempts for efficient software development.

Advantages of using fuzzy logic:

- using expert knowledge based system
- using linguistic values before actual values are known
- allowing less precise estimates for models

Function Point Approach can be implemented using fuzzy logic as it is well documented and has quality control.

Software and guidelines leads to adopting fuzzy logic scheme prominently. Conventionally the models fail to predict the software effort before coding, in some cases like COCOMO the size measure is transformed before hand .Fuzzy logic implementation of software metrics helps by use of single model, ability to cope with small or non existent data sets, robustness to data quality and use of fuzzy logic as means of communicating project management issues.

Using a single model through out the entire development model is advantageous as it takes linguistic inputs in the form of small or large and gives output as precise crisp values .In multiple models the requirements to predict effort is less but it leads to fluctuation and inconsistency.

Effort prediction from a fuzzy logic model can be made in different stages of development cycle with different levels of precision. This paper suggests some of the levels of precision in estimating development effort

Fuzzy models can be easily constructed with a small sample of data to validate the models

FULSOME provides software metrics developer using a series of tools like to input data, membership function, rule creation, inferencing and online explanation of fuzzy logic. Few of the membership functions used are Gaussian, Bell, Trapezoidal, Triangular, Sigmoidal, Tnorm, Tconorm and defuzzification strategies.

This paper has generated an algorithm for membership functions. This involves selecting a function and finding the center and making it the center of cluster. This can be explained by the following algorithm as given in the paper:

1. Select an appropriate mathematically defined function for the membership functions of the variable of interest ( $i$ ), say  $f_i(z)$
2. Select the number of membership functions that are desired for that particular variable,  $m_i$  functions for variable  $i$
3. Call each of the  $m_i$  functions  $f_{ij}([z])$  where  $j = 1 \dots m_i$  and  $[X_i]$  is an array of parameters defining that particular function (usually a center and width parameter are defined, either explicitly or implicitly)
4. Using one-dimensional fuzzy c-means clustering on the data set find the  $m_a$  cluster centers, from the available data
5. Sort the cluster centers into monotonic
6. Set the membership function center for  $f_a J$ , generally represented as one of the parameters in the array  $[z]$ , to the cluster center  $ctJ$
7. Set the membership function widths for  $f_{ij}$  in  $[x]$  such that  $f_{tn}([c z n, \dots]) = 1$ , or as close as possible for the chosen  $f(z)$  where this can not be achieved exactly (for example for triangular membership functions each function can be defined using three points,  $a$ ,  $b$ , and  $c$  where  $a$  is the center of the next smaller functions and  $c$  is the center of the next larger function)

## **PAPER 5**

### **On Generating FC Fuzzy Rule Systems from Data Using Evolution Strategies**

Yaochu Jin, Member, IEEE, Werner von Seelen, and Bernhard Sendhoff

#### **Summary**

Software sizing is an important management activity for both customer and developer that is characterized by uncertainty. Fuzzy system modeling offers a means to capture and logically reason with uncertainty. This paper investigates the application of fuzzy modeling techniques to two of the most widely used software effort prediction models, the Constructive Cost Model and the Function Points model.

#### **SOFTWARE PREDICTION MODELS**

##### **A. Current Methods**

Current methods of software effort prediction include

- 1) expert opinion,
- 2) 2) analogy,
- 3) decomposition and summation , and
- 4) algorithmic models which attempt to relate input cost drivers to output effort (cost)

Algorithmic modeling is preferred by management because the inputs are quantifiable: and it is a repeatable process. These models use linear regression to correlate cost drivers, based on historical data, to the effort required.

##### **B. Constructive Cost Model**

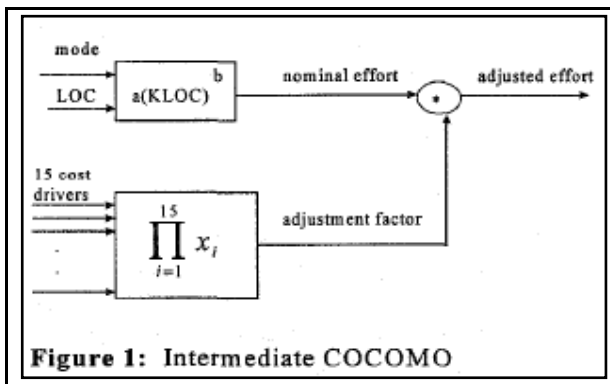
COCOMO, developed by Boehm while at TRW, is used in three phases. Basic COCOMO has two inputs: the mode and LOC estimate (in thousands of delivered source code), and produces a nominal effort estimate using the nonlinear equation.

$$effort = a(KLOC)^b$$

**Table 1: Mode Coefficients and Exponents**

<u>mode</u>	<u>a</u>	<u>b</u>
organic	2.4	1.05
semi-detached	3	1.12
embedded	3.6	1.2

Intermediate COCOMO uses the product of 15 adjustment factors [Table 21, determined by the user, and the nominal effort (from Basic COCOMO) to produce an adjusted effort estimate.

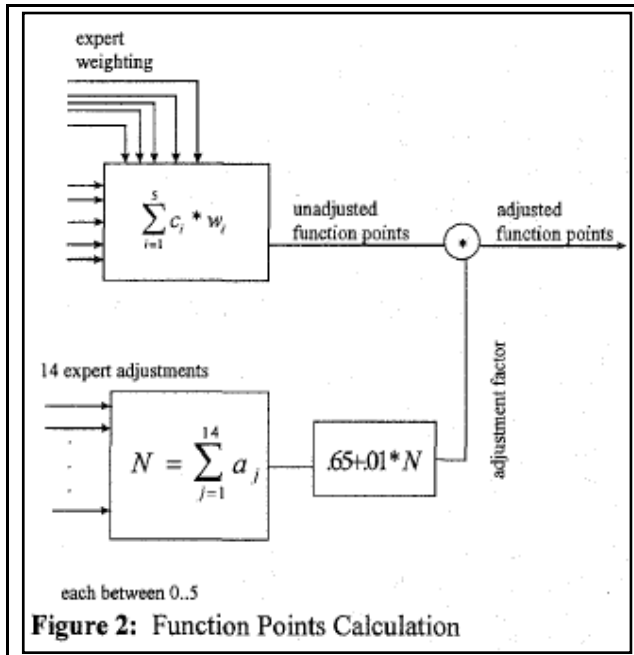


**Figure 1: Intermediate COCOMO**

### C. Function Points

Functionality factors in each of five categories are counted and given an expert weighting

Functional Factor	Weights for		
	low	avg.	high
# user inputs	3	4	6
# user outputs	4	5	7
# user inquiries	3	4	6
# logical files	7	10	15
# external interfaces	5	7	10



## FUZZY MODELING APPROACH

An alternate approach to the effort prediction problem is to use fuzzy systems that perform a mapping between linguistic terms such as “medium complexity” and “high cost”. Fuzzy systems are able to capture uncertainty associated with independent input variables (cost drivers) and output variables (cost/effort) using fuzzy set theory. Fuzzy logic is used as an approximate reasoning technique.

A fuzzy expert system has three parts:

- fuzzification of inputs,
- imprecise reasoning with a fuzzy rule bank, and
- defuzzification for final output.

Inputs and outputs can be either linguistic or numeric. Fuzzification involves finding the membership of an input variable with a linguistic term.

Some of the benefits of using fuzzy modeling techniques for software effort prediction:

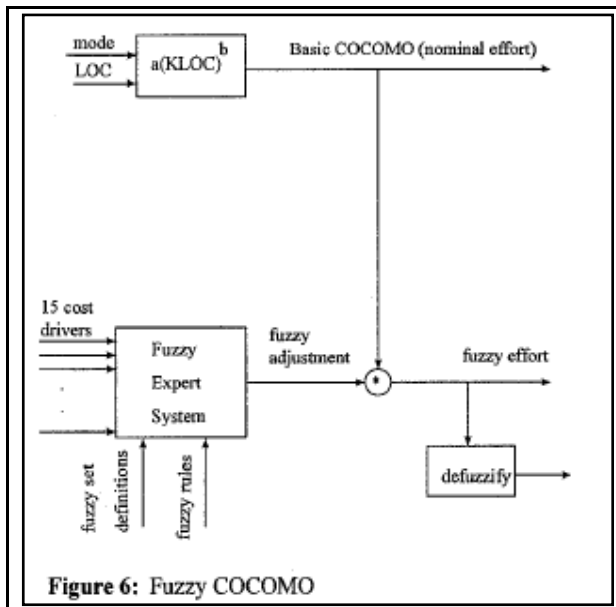
- intuitive nature of linguistic inputs and output
- possible reduction of input variables
- approximate reasoning ability



- better performance early in sizing task
- reduces dependence on historical data
- reduces commitment on crisp prediction

### Fuzzy COCOMO and Fuzzy Function Points Models

One way to deal with the imprecision in these models involves inserting a fuzzy expert system to calculate the adjustment factor.



A second alternative involves reducing the number of inputs of both models and inserting the fuzzy expert system at the top layer. For COCOMO, this would eliminate the equation calculation. For functions points, it would eliminate the function points calculation.

## PAPER 6

### *f*<sup>2</sup> COCOMO: Estimating Software Project Effort and Cost

Allan Caine and Anne Banks Pidduck

#### Summary

The Constructive Cost Model, COCOMO, was developed to estimate the effort and cost to complete a software project. All business enterprises involved in developing software must know their costs to maintain their long-term viability. COCOMO measures the size of the project in lines of code. When the size of the project is measured in function points, COCOMO uses a function points to lines of code converter. By experimentation, this research paper shows that software project data can be analyzed on a programming language basis. The different programming languages are reflected in the constants *a* and *b*. A model can be derived for each programming language by simply separating the data on a programming language basis. Suppose we have project data which relates function points to project effort. If that data is separated by programming language and analyzed, then a function point model, *f*<sup>2</sup> COCOMO, can be derived for each programming language. One advantage of this model is that it eliminates the errors introduced by arbitrary function point indices and replaces them with constants that are scientifically verifiable.

The formula for COCOMO is:

$$SM = aS^b \times EAF,$$

where *SM* is effort, *S* is lines of code, *EAF* is effort adjustment factor and *a* and *b* are constants.

Similarly for COCOMO II, the formula is:

$$SM_2 = 2.94S^{(0.91+W)}$$

The COCOMO 81 and COCOMO II models both models have a major deficiency that they cannot take function points as a direct input. Yet function points are a superior

metric to measuring project size compared to lines of code. Instead, a function point index is used to convert the function points to an equivalent number of lines of code using a standard converter. Unfortunately, the values of these indices are not universally agreed upon and any errors in the values of these indices materially affect the estimate of the software project effort and consequently the estimated cost of producing the software.

In the model proposed in this paper, function points,  $EM$ ,  $W$ , and possibly other model inputs are taken as a direct input. The function point to lines of code converter is not used. The box labeled ' $f^2$  COCOMO' is the mathematical formula, which computes  $SM2$  from function points,  $EM$ ,  $W$ , and possibly other model inputs. In this, experiments were carried out by re-computing the values of constants  $a$  and  $b$  in the formula given earlier. It uses the Gauss-Newton method. This begins by assuming a form of the solution and taking an initial guess of the parameters  $a$  and  $b$ . Iteratively, the method computes the 'goodness-of-fit' of the current parameters, and computes new and better parameters. As it iterates, the method (hopefully) converges upon the 'correct' solution.

## **PAPER 7**

### **Software Cost Estimation with Fuzzy Models**

Petr Musilek, Witold Pedrycz, Giancarlo Succi, Marek Reformat

#### **Summary**

Estimation of effort/cost required for development of software products is inherently associated with uncertainty. This paper deals with a fuzzy set-based generalization of the COCOMO model (f-COCOMO). Rather than using a single number, the software size can be regarded as a fuzzy set (fuzzy number) yielding the cost estimate also in form of a fuzzy set. The paper includes detailed results with this regard by relating fuzzy sets of project size with the fuzzy set of effort. The analysis is carried out for several commonly encountered classes of membership functions (such as triangular and parabolic fuzzy sets). Here the emphasis is on a way of propagation of uncertainty and ensuing visualization of the resulting effort (cost). In the simplest augment the model by admitting, software systems to belong partially to the three main categories (namely embedded, semidetached and organic). In general, these models are based on measuring certain size or function related attributes of the software and relating these measurements to the cost or effort necessary for its development.

Fuzzy sets (as opposed to standard interval analysis) create a more flexible, highly versatile development environment. Firstly, they help articulate the estimates and their essence (e.g., by exploiting fuzzy numbers described by asymmetric membership functions). Secondly, they generate a feedback as to the resulting uncertainty (granularity) of the results.

The reason for extending traditional cost estimation models using fuzzy logic stems from the vagueness present in all the data entering cost estimation process: size, function points, development modes, and other metrics and attributes are matter .of (informed) guessing rather than exact measurements.

In particular, in this paper, the basic COCOMO model is extended. The reason for this lies in it being most simple, plausible for extension and availability for database. The basic equation used is:

$E = aK^b$  which relates effort E to KDSI,K

But size of the project, especially during the earlier stages, is matter of estimation. The correctness and precision of such estimates are limited. Using fuzzy sets, size of a software project can be specified by distribution of its possible values. Fuzzy numbers define various concepts using partial membership functions. Amongst various membership functions, most often used in this case are, triangular and parabolic. These are represented by  $(\alpha, m, \beta)$  where m is the modal value and  $\alpha$  and  $\beta$  represent the left and right boundary values.

Equations of the triangular function:

$$K(x) = \begin{cases} \frac{x-\alpha}{m-\alpha} & \text{for } x \in [\alpha, m] \\ \frac{\beta-x}{\beta-m} & \text{for } x \in [m, \beta] \\ 0 & \text{for } x \notin [\alpha, \beta] \end{cases}$$

Equations of the parabolic function:

$$K(x) = \begin{cases} 1 - \frac{(x-\alpha)^2}{(m-\alpha)^2} & \text{for } x \in [\alpha, m] \\ 1 - \frac{(\beta-x)^2}{(\beta-m)^2} & \text{for } x \in [m, \beta] \\ 0 & \text{for } x \notin [\alpha, \beta] \end{cases}$$

Fuzzy numbers are special fuzzy sets representing uncertain quantitative information. They are convex and normal, usually with single modal value. They are also associated with some vagueness or *fuzziness*. For evaluation of uncertainty, relative fuzziness is defined as:

$$f(A) = \frac{\int_{\alpha}^{\beta} A(x) dx}{m}$$

where more the relative fuzziness, more uncertain A is.

Computations involving fuzzy numbers are carried out in the setting of *fuzzy arithmetic*. It dwells on the extension principle. The extension principle deals with the generalization of this mapping to the case of arguments being fuzzy numbers that is  $B = f(A)$ , where A and B are fuzzy numbers.

If  $C = f(A, B)$  then its membership function can be defined by:

$$C(z) = \sup_{x,y \in \mathbb{R}: z=f(x,y)} [\min(A(x), B(y))]$$

This principle is extended to give rise to simple fCOCOMO. Here the input variable (size K) is a fuzzy set (fuzzy number), so is the effort E and a, b are crisp values. Its extended equation is as follows:

$$E(e) = \sup_{x \in \mathbb{R}: x = \alpha^b} [K(x) = K((\frac{e}{\alpha})^{1/b})]$$

Using the above equation and the equation of the triangular function, we get:

$$E(e) = \begin{cases} \frac{\left(\frac{e}{a}\right)^{1/b} - \alpha}{m - \alpha} & \text{for } e \in [a\alpha^b, am^b] \\ \frac{\beta - \left(\frac{e}{a}\right)^{1/b}}{\beta - m} & \text{for } e \in [am^b, a\beta^b] \\ 0 & \text{for } e \notin [a\alpha^b, a\beta^b] \end{cases}$$

This directly estimates effort from size of software.

Another application is when the software project may concern a system whose membership to one of the three system categories is not obvious. In this case values of a, b are also taken as fuzzy sets. This leads to equation as shown below:

$$E(e) = \max_{a,b \in \mathbb{R}} \left\{ \min \left[ K \left( \left( \frac{e}{a} \right)^{1/b} \right), A(a), B(b) \right] \right\}$$

The methodology of fCOCOMO, as presented in this paper can be applied to other models of software cost estimation also.

## **PAPER 8**

### **COCOMO Cost Model using Fuzzy Logic**

Ali Idri, Alain Abran, Laila Kijri

#### **Summary**

In this paper COCOMO'81 and in particular its intermediate model is studied with view to its implementation using fuzzy logic.

The intermediate model is emphasized upon because:

- It is the most widely use version
- Database for only the simple and intermediate models are available
- The simple model doesn't take enough factors for validation
- Accuracy is greater than the simple model.

The work effort formula is:

$$MM=A*SIZE^B \sum_{j=1}^{15} C_{ij}$$

where MM are man months, SIZE is size of software in KDSI,

A,B are constants specific to project mode

$C_{ij}$  is the effort multiplier with jth selecting range and ith cost driver.

Assignment of linguistic values uses conventional quantization. So no parameter can occupy more than class even if it exists at the boundary of 2. This is because values are taken as intervals with abrupt discrete change between levels instead of gradual. To overcome this, fuzzy sets are used:

- More general
- Mimic human linguistic ways
- Changes are gradual and not abrupt between levels

Intermediate COCMO is evaluated in same way but effort multipliers are taken as fuzzy sets. The gradation is less sensitive to changes in inputs.

Hence, in this case accuracy sensitive is to changes in inputs.

## **PAPER 9**

### **On Generating FC Fuzzy Rule Systems from Data Using Evolution Strategies**

Yaochu Jin, Member, IEEE, Werner von Seelen, and Bernhard Sendhoff

#### **Summary**

Sophisticated fuzzy rule systems are supposed to be flexible, complete, consistent, and compact (FC3). Flexibility, completeness and consistency are essential for fuzzy systems to exhibit an excellent performance and to have a clear physical meaning, while compactness is crucial when the number of the input variables increases. However, the completeness and consistency conditions are often violated if a fuzzy system is generated from data collected from real world applications.

The structure of the fuzzy rules, which determines the compactness of the fuzzy systems, is evolved along with the parameters of the fuzzy systems. Special attention has been paid to the completeness and consistency of the rule base. The completeness is guaranteed by checking the completeness of the fuzzy partitioning of input variables and the completeness of the rule structure. An index of inconsistency is suggested with the help of a fuzzy similarity measure, which can prevent the algorithm from generating rules that seriously contradict with each other or with the heuristic knowledge. Soft T-norm and BADD Defuzzification are introduced and optimized to increase the flexibility of the fuzzy system.

#### **Incompleteness**

A common problem concerning adjustment of the membership parameters is that the shape of the membership functions is adjusted so drastically that either some of the fuzzy subsets lose their corresponding physical meanings, or the fuzzy subsets do not cover the whole space of the input variable. This is called Incompleteness.

The proposed approach is advantageous over the other methods in the following respects.

- 1) The fuzzy system is compact and efficient because the number of the fuzzy rules is greatly reduced



- 2) The fuzzy system is complete and no seriously conflicting rules will be generated, which contributes to the improvement of the generalization ability of the fuzzy system and guarantees that the knowledge acquired by the fuzzy rules is physically sound,
- 3) The fuzzy system is expected to exhibit a better flexibility because soft fuzzy operators are incorporated and optimized.

### Basic Formulas of Fuzzy Systems

$R_i$ : If  $x_1$  is  $A_{i1}$  and  $x_2$  is  $A_{i2}$  and  $\dots$  and  $x_n$  is  $A_{in}$ ,  
then  $y$  is  $B_i$ ;  $i = 1, 2, \dots, N$

$$R_i = (A_{i1} \times A_{i2} \times \dots \times A_{in}) \times B_i$$

which is a fuzzy set whose membership function is described by

$$R_i(x_1, x_2, \dots, x_n, y) \\ = A_{i1}(x_1)T A_{i2}(x_2)T \dots T A_{in}(x_n)T B_i(y)$$

where  $T$  means the T-norm operator. Based on sup-star composition, the overall fuzzy relation of the fuzzy system in terms of membership function can be written as follows:

$$R(x_1, x_2, \dots, x_n, y) = \bigvee_{i=1}^N R_i(x_1, x_2, \dots, x_n, y)$$

The output of a Takagi–Sugeno rule system is in the following form:

$$y = \frac{\sum_{i=1}^N \{T_{j=1}^n A_{ij}(x_j) \cdot f_i(x_1, x_2, \dots, x_n)\}}{\sum_{i=1}^N T_{j=1}^n A_{ij}(x_j)}$$

The consequent part of the Takagi–Sugeno rules is often simplified to a constant, in which case the output of the Takagi–Sugeno rules can be written as follows:

$$y = \frac{\sum_{i=1}^N \{T_{j=1}^n A_{ij}(x_j) p_{i0}\}}{\sum_{i=1}^N T_{j=1}^n A_{ij}(x_j)}$$

## Flexible Fuzzy Operators

The soft T-norm and BADD defuzzifier are very flexible They can be expressed in the following

$$\begin{aligned} \text{Soft T-norm: } \tilde{T}(x_1, x_2, \dots, x_n) &= (1 - \alpha) \frac{1}{n} \sum_{i=1}^n x_i \\ &+ \alpha T(x_1, x_2, \dots, x_n) \end{aligned} \quad (10)$$

$$\text{BADD defuzzifier: } y = \frac{\sum_{i=1}^N \{ [T_{j=1}^n A_{ij}(x_j)]^\delta p_{i0} \}}{\sum_{i=1}^N [T_{j=1}^n A_{ij}(x_j)]^\delta} \quad (11)$$

## Completeness of the Fuzzy Systems

A fuzzy system is said to be complete if

1. Fuzzy partitioning of each input variable is complete;
2. Rule structure of the fuzzy system is complete.

The over fitting of the fuzzy membership functions results in the following consequences:

1. Fuzzy partitioning become incomplete
2. Physical meaning of some fuzzy subsets may be blurred, that is to say, the fuzzy subsets lack distinguishability

A fuzzy similarity measure indicates the degree to which two fuzzy sets are equal

In our approach, the fuzzy similarity measure is used to preserve the completeness of the fuzzy partitionings of the input variables and to preserve the distinguishability of the fuzzy subsets. For any two fuzzy sets A and B the fuzzy similarity measure is defined by:

$$\begin{aligned} S(A, B) &= \frac{M(A \cap B)}{M(A \cup B)} \\ &= \frac{M(A \cap B)}{M(A) + M(B) - M(A \cap B)} \end{aligned}$$

where M(A) is called the size of fuzzy set A and can be calculated as follows:

$$M(A) = \int_{-\infty}^{+\infty} A(x) dx$$

A fuzzy rule system may still be incomplete even if the fuzzy partitioning of the input variables are complete. This happens when the rule structure is incomplete, i.e., some of the fuzzy subsets are not used by the rule system, which is often the case in the course of rule structure optimization.

### Consistency of the Fuzzy Systems

Therefore, fuzzy rules are regarded as inconsistent, if

1. They have very similar premise parts, but possess rather different consequents, and
2. They conflict with the expert knowledge or heuristics.

Two fuzzy rules may contradict with each other even if they do not have the same premise, on the other hand, it is hard to say that two rules are inconsistent if their premise parts have little similarity.

$$R_i: \text{ If } x_1 \text{ is } A_{i1}(x_1) \text{ and } x_2 \text{ is } A_{i2}(x_2) \text{ and } \dots \text{ and } x_n \text{ is } A_{in}(x_n), \text{ then } y \text{ is } B_i(y)$$

$$R_k: \text{ If } x_1 \text{ is } A_{k1}(x_1) \text{ and } x_2 \text{ is } A_{k2}(x_2) \text{ and } \dots \text{ and } x_n \text{ is } A_{kn}(x_n), \text{ then } y \text{ is } B_k(y)$$

Then SRP and SRC of these two rules are defined as follows:

$$SRP(i, k) = \min_{j=1}^n S(A_{ij}, A_{kj})$$

$$SRC(i, k) = S(B_i, B_k)$$

where  $n$  is the total number of the input variables and  $S$  is the fuzzy similarity measure of fuzzy sets and as defined in (13). Then the consistency of rule and is defined by:

$$Cons(R(i), R(k)) = \exp \left\{ - \frac{\left( \frac{SRP(i, k)}{SRC(i, k)} - 1.0 \right)^2}{\left( \frac{1}{SRP(i, k)} \right)^2} \right\}$$

## **PAPER 10**

### **An Approach to Rule-Based Knowledge Extraction**

Yaochu Jin ,Werner von Seelens and Bernhard Sendhoff

#### **Summary**

The extraction of easily interpretable knowledge from the large amount of data measured in experiments is well desirable. A fuzzy rule system is first generated and optimized using evolution strategies. This fuzzy system is then converted to an RBF neural network to refine the obtained knowledge. In order to extract understandable fuzzy rules from the trained RBF network, a neural network regularization technique called adaptive weight sharing is developed.

**Interpretability** of a fuzzy system usually involves the following aspects. Firstly, the fuzzy partitioning for each input variable of the fuzzy system should be complete and different fuzzy subsets in a fuzzy partitioning should be well distinguishable

1. One direct method to achieve this is to limit the range of the parameters of membership functions during learning [8]. This can be achieved more flexibly with the help of a fuzzy similarity measure
2. Secondly, the number of fuzzy subsets in a fuzzy partitioning should be limited and each fuzzy subset should have one unique membership function, to which a proper physical meaning can be assigned.
3. Thirdly, fuzzy rules in the rule base should be consistent. Traditionally, this means that fuzzy rules with the same premise should have the same consequent

A fuzzy system is first generated by virtue of evolution strategies. Then we convert the fuzzy system to an RBF neural network for further training to refine the acquired knowledge. After this learning stage, the RBF neural network cannot be directly converted back to a clearly interpretable fuzzy system because there may be numerous fuzzy subsets in a fuzzy partitioning which are hard to distinguish and hard to assign

proper linguistic values to. To solve this problem, a network regularization algorithm called adaptive weight sharing is developed to train the RBF neural network further so that some of the basis functions as well as the output weights in the RBF network share certain values. As a result, a well interpretable fuzzy system can again be obtained. This method has proved to be successful by simulation studies on the Mackey-Glass time series.

### **Fuzzy System Generation and Optimization**

*Using Evolution Strategies Completeness conditions:*

Suppose an input variable of a fuzzy system  $x$  is partitioned into  $M$  fuzzy subsets represented by  $A_1(x), A_2(x), \dots, A_M(x)$  on the universe of discourse  $U$ , then the partitioning is complete if the following condition holds:

$$\forall x \in U \sum_{1 \leq i \leq M} A_i(x) > 0$$

In the optimization of fuzzy systems based on evolutionary algorithms or neural networks, it is often the case that either the fuzzy partitionings are incomplete or different fuzzy subsets in a fuzzy partitioning lack good distinguishability. To avoid this, we require that every two neighbouring fuzzy sets should satisfy the following constraints:

$$\delta_1 \leq S(A_i, A_{i+1}) \leq \delta_2$$

where  $S(A_i, A_{i+1})$  is called the fuzzy similarity measure between the two fuzzy subsets  $A_i$  and  $A_{i+1}$ ,  $\delta_1$  and  $\delta_2$  are two thresholds of the fuzzy similarity measure, where  $\delta_1$  should be greater than zero to keep the fuzzy partitioning complete and  $\delta_2$  should be sufficiently smaller than  $\delta_1$  to ensure good distinguishability. The fuzzy similarity measure is defined by:

$$S(A_i, A_{i+1}) = \frac{M(A_i \cap A_{i+1})}{M(A_i) + M(A_{i+1}) - M(A_i \cap A_{i+1})}$$

where  $M(\cdot)$  is called the size of the fuzzy set. If fuzzy set  $A(x)$  has a Gaussian membership function with center  $\mu$  and width (or variance)  $\sigma$ , then  $M(A(x))$  can be calculated as:

$$M(A(x)) = \int_{-\infty}^{+\infty} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right) dx$$

It is noticed that if  $S(A_i, A_{i+1})$  equals 1, the two fuzzy sets overlap completely, i.e.  $A_i$  and  $A_{i+1}$  are equal. On the other hand, they do not overlap if  $S(A_i, A_{i+1}) = 0$ .

### Consistency of fuzzy systems

It is easy to imagine that two fuzzy rules are inconsistent if they have the same if-part but different then-parts. However, we argue that two fuzzy rules may also be inconsistent even if their if-parts are different. To evaluate the consistency of two arbitrary fuzzy rules, definitions of Similarity of Rule Premise (SRP) and Similarity of Rule Consequent (SRC) are given.

$$R_i : \text{If } x_1 \text{ is } A_{i1}(x_1) \text{ and } x_2 \text{ is } A_{i2}(x_2) \text{ and } \dots \text{ and } x_n \text{ is } A_{in}(x_n), \text{ then } y \text{ is } B_i(y)$$

$$R_k : \text{If } x_1 \text{ is } A_{k1}(x_1) \text{ and } x_2 \text{ is } A_{k2}(x_2) \text{ and } \dots \text{ and } x_n \text{ is } A_{kn}(x_n), \text{ then } y \text{ is } B_k(y)$$

Then SRP and SRC between rule  $i$  and rule  $k$  are defined in terms of the fuzzy similarity measure as follows:

$$SRP(i, k) = \bigwedge_{j=1}^n S(A_{ij}, A_{kj})$$

$$SRC(i, k) = S(B_i, B_k)$$

where  $n$  is the total number of the input variables. The consistency between rule  $R(i)$  and  $R(k)$  can now be defined as:

$$Cons(R(i), R(k)) = \exp \left\{ - \frac{(\frac{SRP(i,k)}{SRC(i,k)} - 1.0)^2}{(\frac{1}{SRP(i,k)})^2} \right\}$$

Thus, inconsistency is given by:

$$f_{Incons} = \sum_{i=1}^N \sum_{\substack{1 \leq k \leq N \\ k \neq i}} [1.0 - Cons(R(i), R(k))]$$

where  $N$  is the total number of rules.

Since they impose additional restrictions in generating fuzzy systems, they could be treated as a means of regularization

### **ES based fuzzy rule generation and optimization**

The quality of the fuzzy system can be evaluated by the following cost function:

$$f = f_E + \xi \cdot f_{Incons} + f_{Incomp}$$

### **Conversion of the Fuzzy System to an RBF Network**

The significance of converting fuzzy systems to neural networks lies in two aspects. On the one hand, a fuzzy system can be refined taking advantage of the learning ability of neural networks. On the other hand, the structure of a neural network can be determined and prior knowledge can be incorporated into the network with the help of a fuzzy system.

An interpretable fuzzy system and an RBF neural network are equivalent if the following conditions hold:

1. Both the fuzzy system and the neural network have Gaussian basis function
2. The number of fuzzy rules is equal to the number of receptive field units ( or hidden nodes) in the RBF network.
3. The fuzzy system is either a zero-order Takagi-Sugeno model or a Mamdani model. If a Mamdani model is used, the corresponding defuzzification method should be the simplified weighted average
4. The output of the RBF neural network should be normalized.
5. The receptive field units in the RBF network are allowed to have different variances.
6. Centers and variances from different receptive field units but for the same input variable should share certain values, which could construct a complete and well distinguishable fuzzy partitioning.

In this section, we convert the fuzzy system generated by evolution strategies to an RBF network for further training using the learning algorithm of the neural network. The final input-output relationship of the fuzzy system with  $n$  inputs and one output is expressed as follows:

$$y = \frac{\sum_{j=1}^N [w_j \prod_{i=1}^{m_j} \exp(-\frac{(x_i - \mu_{ij})^2}{\sigma_{ij}^2})]}{\sum_{j=1}^N \prod_{i=1}^{m_j} \exp(-\frac{(x_i - \mu_{ij})^2}{\sigma_{ij}^2})}$$

Consequently, a conventional learning algorithm based on the gradient method can be directly applied.

### **Extraction of Fuzzy Rules by Regularization**

To extract meaningful fuzzy rules from the trained neural network, we introduce here a novel weight sharing regularization technique. This technique enables the output weights and the parameters of the basis functions of the RBF network to share some certain values so that each fuzzy partitioning has a proper number of fuzzy subsets with well distinguishable membership functions.



## **PAPER 11**

### **Design Of Fuzzy Controllers**

Jan Jantzen

#### **Summary**

**Fuzzy controllers** are used to control consumer products, such as washing machines, video cameras, and rice cookers, as well as industrial processes, such as cement kilns, underground trains, and robots. Fuzzy control is a control method based on fuzzy logic.

Fuzzy control can be described simply as "control with sentences rather than equations".

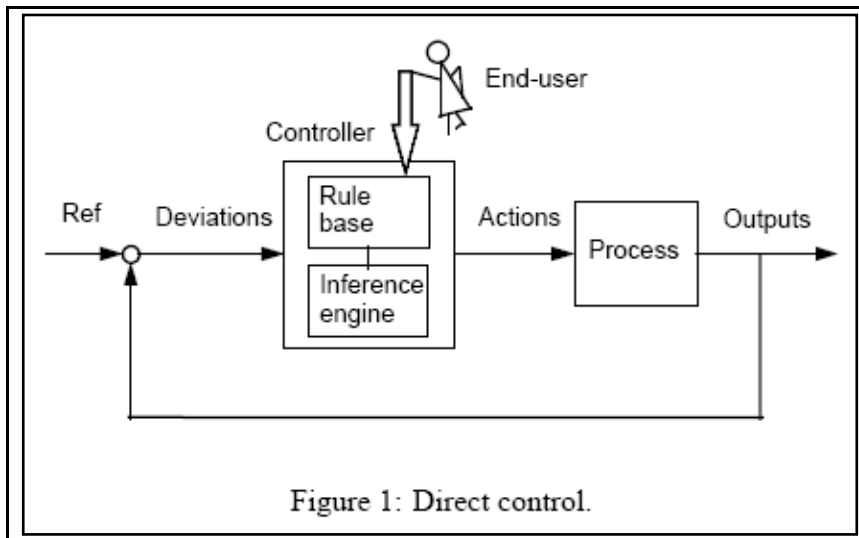
Design of a fuzzy controller requires more design decisions than usual, for example regarding rule base, inference engine, defuzzification, and data pre- and post processing.

The approach here is based on a three step design procedure that builds on PID control:

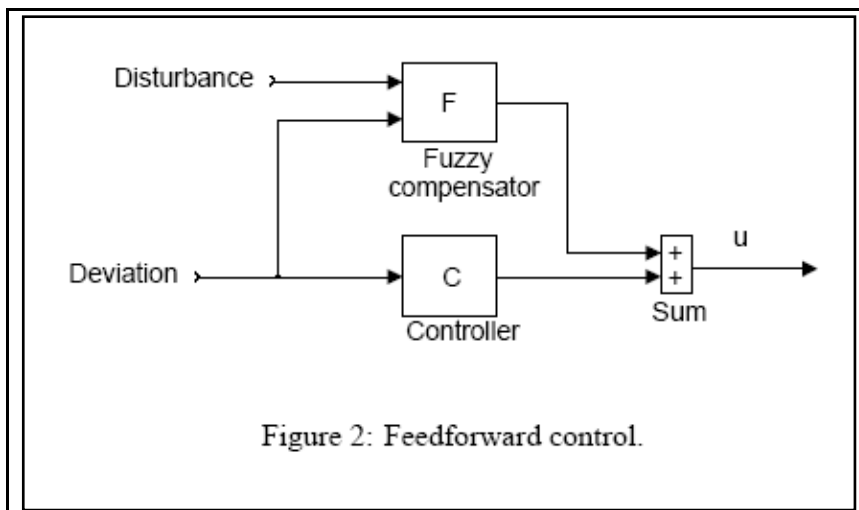
1. Start with a PID controller.
2. Insert an equivalent, linear fuzzy controller.
3. Make it gradually nonlinear.

In a rule based controller the control strategy is stored in a more or less natural language. The control strategy is isolated in a rule base opposed to an equation-based description. A rule based controller is easy to understand and easy to maintain for a non-specialist end-user. The computer is able to execute the rules and compute a control signal depending on the measured inputs errors and change in errors.

One control scheme is Direct Control where the fuzzy controller is in the forward path in a feedback control system. The process output is compared with a reference, and if there is a deviation, the controller takes action according to the control strategy.



In Feed forward Control, a measurable disturbance is being compensated. It requires a good model, but if a mathematical model is difficult or expensive to obtain, a fuzzy model may be useful.



Fuzzy rules are also used to correct tuning parameters in parameter adaptive control schemes (Fig. 3). A gain scheduling controller contains a linear controller whose parameters are changed as a function of the operating point in a preprogrammed way. Sensor measurements are used as scheduling variable that govern the change of the controller parameters, often by means of a table look-up.

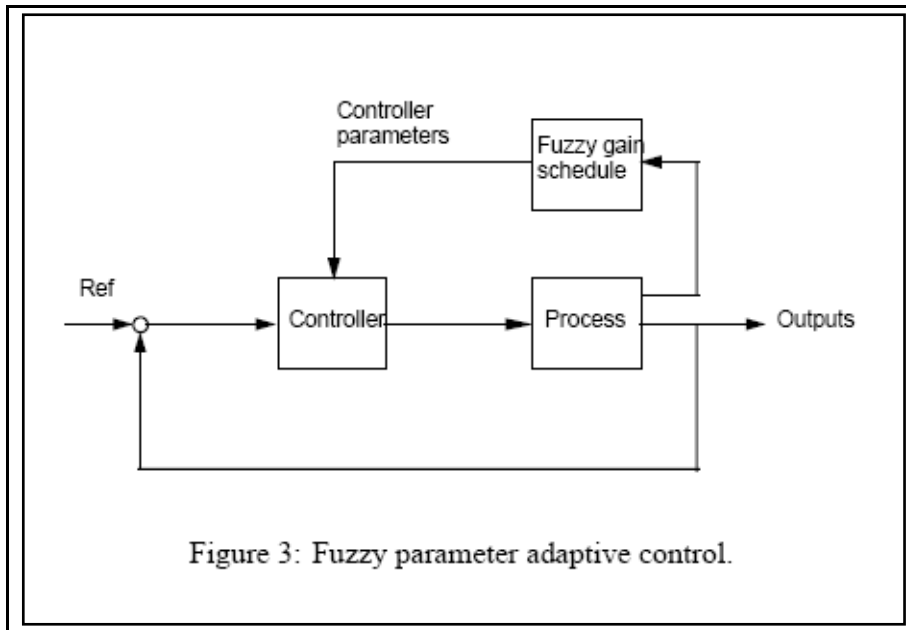


Figure 3: Fuzzy parameter adaptive control.

Stability concerns the system's ability to converge or stay close to equilibrium. A stable linear system will converge to the equilibrium asymptotically no matter where the system states Variables start.

There are at least four main sources for finding control rules. The most common approach to establishing such a collection of rules of thumb is to question experts or operators using a carefully organized questionnaire.

- Based on the operator's control knowledge- Fuzzy *if – then* rules can be deduced from observations of an operator's control actions or a log book. The rules express input-output relationships.
- Based on the fuzzy model on the process- This method is restricted to relatively low order systems, but it provides an explicit solution assuming that fuzzy models of the open and closed loop systems are available.
- Based on learning- The self-organizing controller is an example of a controller that finds the rules itself. Neural networks are another possibility.

### Structure of Fuzzy Controller

There are specific components characteristic of a fuzzy controller to support a design procedure.

In the block diagram in Fig. 4, the controller is between a preprocessing block and a post-processing block.

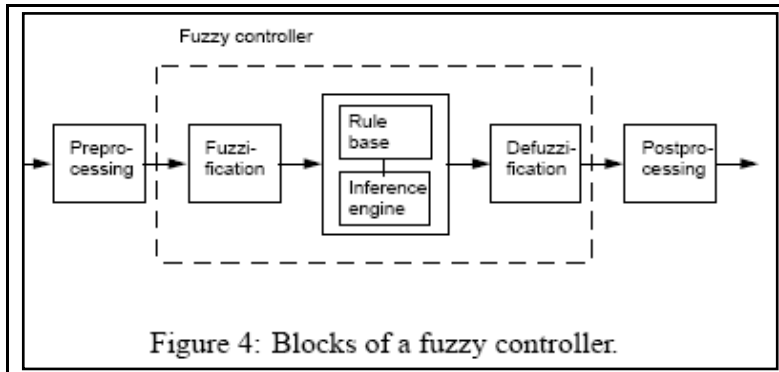


Figure 4: Blocks of a fuzzy controller.

### Preprocessor

A preprocessor, conditions the measurements before they enter the controller. Examples of preprocessing are:

- Quantisation in connection with sampling or rounding to integers;
- Normalization or scaling onto a particular, standard range;
- Filtering in order to remove noise;
- Averaging to obtain long term or short term tendencies;
- A combination of several measurements to obtain key indicators; and
- Differentiation and integration or their discrete equivalences.

A quantizer is necessary to convert the incoming values in order to find the best level in a discrete universe.

### Fuzzification

The first block inside the controller is fuzzification, which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block thus matches the input data with the conditions of the rules to determine how well the condition of each rule matches that particular input instance.

## **Rule Base**

The rules may use several variables both in the condition and the conclusion of the rules. The controllers can therefore be applied to both multi-input-multi-output (MIMO) problems and single-input-single-output (SISO) problems.

### *Rule Format*

Basically a linguistic controller contains rules in the if-then format, but they can be presented in different formats.

1. If error is Neg and change in error is Neg then output is NB
2. If error is Neg and change in error is Zero then output is NM
3. If error is Neg and change in error is Pos then output is Zero
4. If error is Zero and change in error is Neg then output is NM
5. If error is Zero and change in error is Zero then output is Zero
6. If error is Zero and change in error is Pos then output is PM
7. If error is Pos and change in error is Neg then output is Zero
8. If error is Pos and change in error is Zero then output is PM
9. If error is Pos and change in error is Pos then output is PB

In case the table has an empty cell, it is an indication of a missing rule, and this format is useful for checking completeness.

## **Connectives**

Here the lines are connected using if-then-else, if and only if The connectives “and” and “or” are always defined in pairs, for example,  $A \text{ and } B = \min(A, B)$

## **Modifiers**

A linguistic modifier, is an operation that modifies the meaning of a term.

## **Universe**

Elements of a fuzzy set are taken from a Universe of discourse or just universe.

The universe contains all elements that can come into consideration

### **Membership Function**

Every element in the universe of discourse is a member of a fuzzy set to some grade, maybe even zero. The set of elements that have a non-zero membership is called the support of the fuzzy set. The function that ties a number to each element of the universe is called the Membership function

### **Inference Engine**

The rules reflect the strategy that the control signal should be a combination of the reference error and the change in error, a fuzzy proportional-derivative controller. For each rule, the inference engine looks up the membership values in the condition of the rule. Following are the operations:

- Aggregation
- Activation
- Accumulation

### **Defuzzification**

The resulting fuzzy set must be converted to a number that can be sent to the process as a control signal. This operation is called Defuzzification. There are several Defuzzification methods:

- Center of gravity
- Center of Gravity for singleton
- Bisector of Area
- Mean of Maxima
- Left most maximum and right most maximum

### **Postprocessing**

The postprocessing block often contains an output gain that can be tuned, and sometimes also an integrator.

### **Table based controller**

In a table based controller the relation between all input combinations and their corresponding outputs are arranged in a table.

With two inputs and one output, the table is a two-dimensional look-up table. The array implementation improves execution speed, without too much searching.

### **Input output mapping**

The controllers have the input families in the if -column and the output families in the then column. The results depend on the choice of design.

## PAPER 12

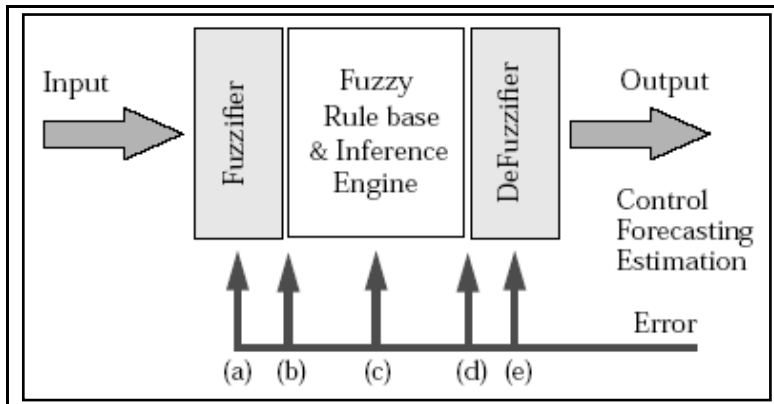
### Adaptation of Fuzzy Inferencing: A Survey

Payman Arabshahi, Robert J. Marks II, and Russell Reed

#### Summary

Fuzzy inference has numerous applications, ranging from control to forecasting. A number of researchers have suggested how such systems can be tuned during application to enhance inference performance. Inference parameters that can be tuned include the central tendency and dispersion of the input and output fuzzy membership functions, the rule base, the cardinality of the fuzzy membership function sets, the shapes of the membership functions and the parameters of the fuzzy AND and OR operations. In this paper, an overview of these tuning procedures is given. An extensive bibliography is provided of recent literature on the topic.

A general fuzzy inference system consists of three parts



The familiar operations to arrive at the output are as follows.

1. Perform a pairwise fuzzy intersection  $T$ , on each of the membership values of  $x_1$  and  $x_2$  in  $\mu_1$  and  $\mu_2$  for every rule with consequent  $n_k$ , forming activation values  $\zeta$ :

$$\zeta_{lm}^k = T_{l,m \in S_k} (\mu_1^l(x_1), \mu_2^m(x_2))$$

Let us assume that the ( $T$ -norm) operator  $T$  itself is parameterized by  $a$ , i.e.,  $T = T(a)$ .



2. Collect activation values for like output membership functions and perform a fuzzy union  $T_{-}$ , where  $T_{-} = T_{-}(b)$

$$w_k = T_{-}^{*} (\zeta_{lm}^k)$$

3. These values are defuzzified to generate the output estimated value,  $f(x_1; x_2)$ , by computing the centroid of the composite membership function  $\mu$ :

$$\mu = \sum_{k=1}^K w_k v^k$$

$$y(x_1, x_2) = \frac{\sum_{k=1}^K w_k c_k A_k}{\sum_{k=1}^K w_k A_k},$$

where

$$A_k = \int v^k(x) dx, \quad c_k = \frac{\int x v^k(x) dx}{\int v^k(x) dx}.$$

$A_k$  and  $c_k$  are, respectively, the area and centroid of the consequent membership function  $n_k$ .

### Adaptation in Fuzzy Inference Systems

All of the stages of the fuzzy inference system are affected by the choice of certain parameters. A list follows.

#### A. The Fuzzifier

The fuzzifier maps the input onto the possibility domain and has the following parameters:

1. The number of membership functions.
2. The shape of the membership functions (*e.g.* triangle, Gaussian, etc.)
3. The Central tendency (*e.g.* center of mass) and dispersion (*e.g.* standard deviation, bandwidth, or range) of the membership function.

#### B. The Inference Engine

The inference engine is the system “decision maker” and determines how the system interprets the fuzzy linguistics. Its parameters are those of the aggregation operators. which provide interpretation of connectives “AND” and “Or”.

### C. The Defuzzifier

The defuzzification stage maps fuzzy consequents into crisp output values. Its design requires choice of

1. The number of membership functions.
2. The shape of membership functions.
3. The definition of fuzzy implication, *i.e.*, how the value of the consequents from the inference engine impact the output membership functions prior to defuzzification.
4. A measure of central tendency of the consequent altered output membership functions. The center of mass is typically used, although use of medians and modes can also be used to arrive at the crisp output.

## 6.2 f-SEE Analysis and Design Models

The above literature survey was useful in formulating the framework for f-COCOMO and Inference Based Analysis Models and the Design Models used in f-SEE. It helped us to choose effective default values for the various fuzzy variables used in f-SEE. The default rule base provided by us is also based on the study of these papers from the literature. We have used two fuzzy number based models and we have also developed a skeleton expert system for fuzzy inference based cost estimation. Rule Base Editor and Membership Function Editors have been provided to enable the user to experiment with values other than the default values. All the membership functions in f-SEE are triangular in nature, as they pertain to software engineering practices and are widely adopted in the industry. A main reason for this is that they provide a peak, singleton value, which allows us to specify a crisp set, if required.

## CHAPTER 7

---

### FEATURES AND SOFTWARE CONSIDERATIONS FOR f-SEE

---

#### 7.1 Introduction

**f-SEE** is a user-friendly, integrated, graphics-based Software Estimation Environment. Developed in Visual C#, it provides the user with over 40 interactive forms. The user is prompted accordingly to rectify an input error. f-SEE has two modes of operation:

1. Analysis
2. Design

The modes are selected using Menu Options. The software provides graceful degradability in all cases, by means of various error handling mechanisms. In the Analysis Mode, the user can feed in the information s/he has about the software and employ the models provided for software estimation. The models have been classified at three levels- Conventional Models, which include the Basic COCOMO Model, the Intermediate COCOMO Model and the Function Point Measure; Fuzzy Models, which include the Basic f-COCOMO Model, the Intermediate f-COCOMO Model and the Fuzzy Inference Based Estimation Model; and Object-Oriented Models, which includes the Class Point Measure. The Conventional Models have been suggested in the 1980s by Prof. Barry Boehm of the University of South Carolina. The f-COCOMO Models (Basic and Intermediate) have been based on the lines of the model suggested by Musilek et al.

The Design Mode is a special crucial feature of f-SEE that offers the user design guidance and support. This mode of operation allows the user to validate all the membership functions of the fuzzy variables used in the inference engine (KLOC, Complexity and Effort). It also helps the user to improve his/her membership function. The module checks for two main characteristics – Incompleteness and Lack of Distinguishability. Incompleteness is exhibited by the fuzzy variable if certain values from its Universe of Discourse are left unmapped on the  $\mu$ -axis. The fuzzy variable exhibits Lack of Distinguishability when the physical meaning of the fuzzy subsets defined on the Universe of Discourse is blurred. The Design Mode also provides Rule Base Validation. Fuzzy Rules are regarded as inconsistent if they have very similar premise parts and rather different consequents. Fuzzy membership graphic display is provided in f-SEE for

pictorial representation. A Summary Sheet showing a rule-by-rule similarity based report is generated by the software.

### **Major Highlights of f-SEE:**

- It is an easy-to-use, user-friendly, GUI based software package
- It provides graceful degradability at all times
- It caters for analysis as well as design in the same environment
- It serves as a comparator tool for the various estimation models of analysis
- It can serve in the industry as a deliverable, as well as a research software for independent study, thus making it very versatile specially for SMEs who cannot afford expensive tools
- It allows the user to plot all the membership functions associated with the fuzzy variables to the scale
- It also allows the user to save all the validated membership functions to the Inference Engine for future use
- It provides a Membership Function Editor and a Rule Base Editor, thus equipping the user with the facility of modifying the inferencing process to suit his/her needs
- It provides a basic Primer for users who are not well-equipped with the basics of Software Engineering and Fuzzy Logic

### **Analysis Mode**

#### **Conventional Models**

- Basic COCOMO
- Intermediate COCOMO
- Function Point Measure
- Fuzzy Models
  - Basic f-COCOMO
  - Intermediate f-COCOMO
  - Fuzzy Inference Based Estimation
- Object-Oriented Models

- Class Point Measure

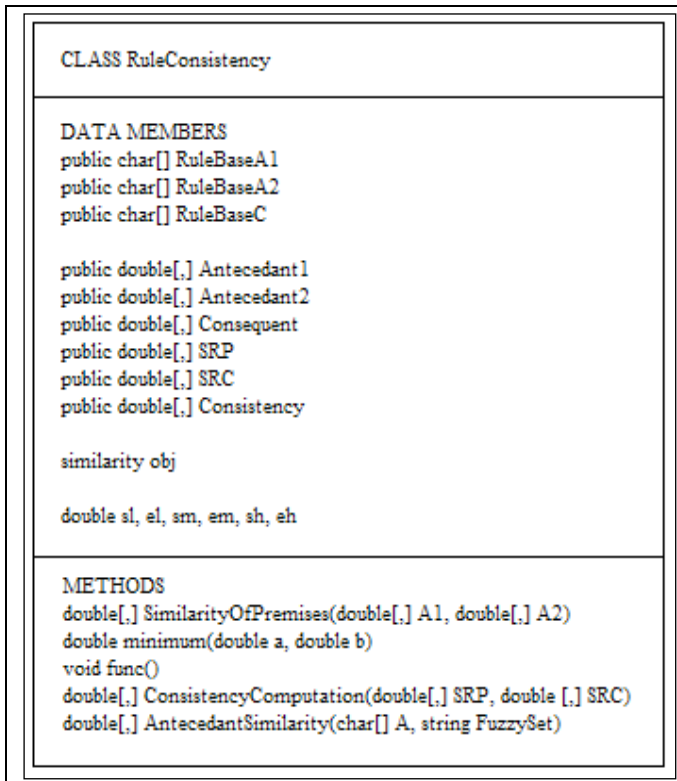
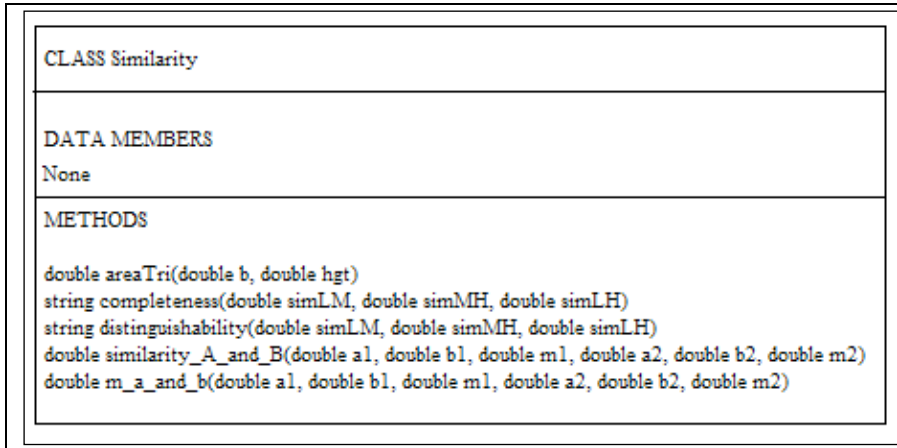
### **Design Mode**

- KLOC Validation
- Complexity Validation
- Effort Validation
- Rule Validation

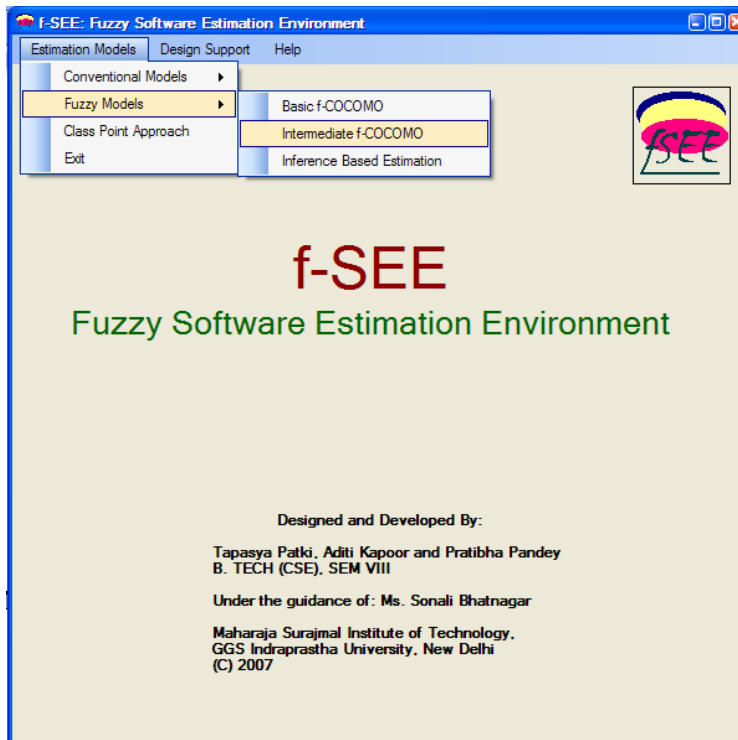
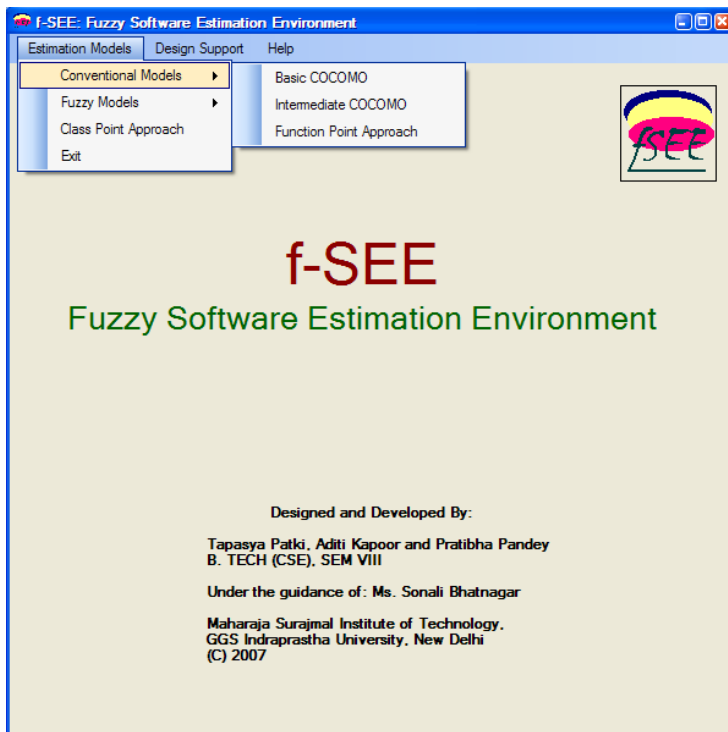
Programming language selection for f-SEE development was a very important consideration. We had Java and C# as two major options and we selected Visual Express C# environment primarily due to its ease of availability from Microsoft and the emerging popularity of .NET platforms. We found several web sites providing reference articles for understanding C# language. Although Visual C# has been used for development of f-SEE, it is not required to deploy f-SEE. Thus, the SME units need not have Visual C# installed on their PCs. For deploying f-SEE, we have to first install .NET framework (version 2), if it is not already installed e.g. in Windows 98 and Windows ME. However, Windows Xp which is being used by several SME units is already .NET framework enabled. The executable version installs using setup.exe file. In order to get insight into the potentials of f-SEE, several screen shots are included in this report.

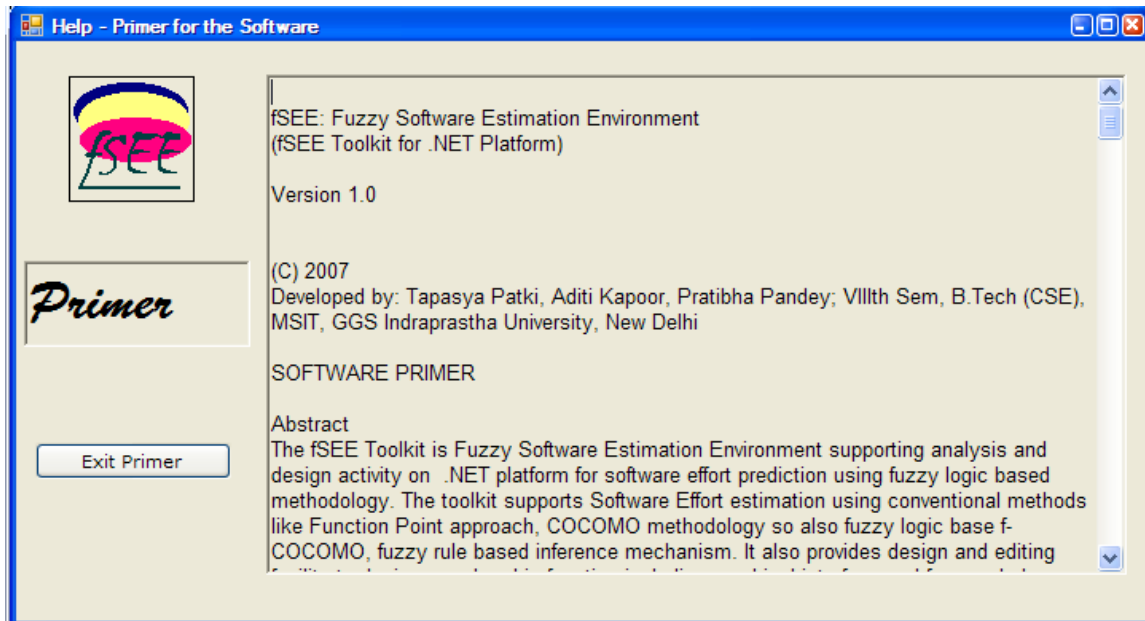
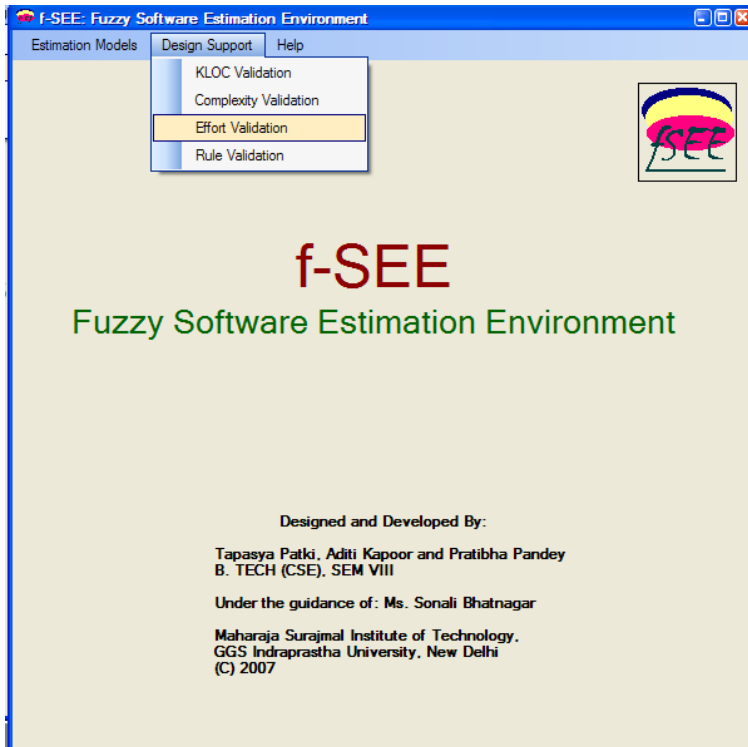
## 7.2 UML Diagrams for f-SEE Classes

We have used the Visual C# environment for developing the software. Over 40 forms have been developed, and the environment defines partial classes for these forms accordingly. In addition to the forms, we have added two classes for catering to similarity operations and Rule Consistency operations. We are providing UML diagrams for these two classes.



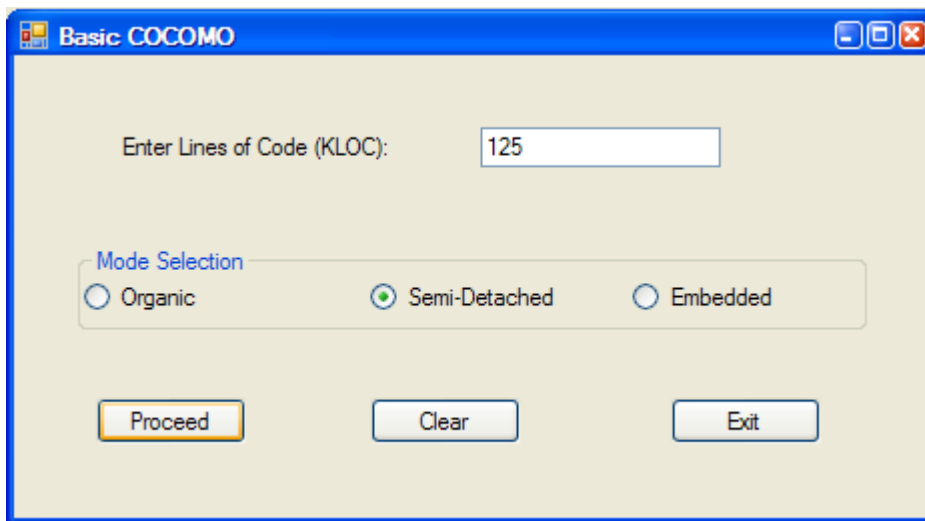
## 7.3 f-SEE Screen Shots



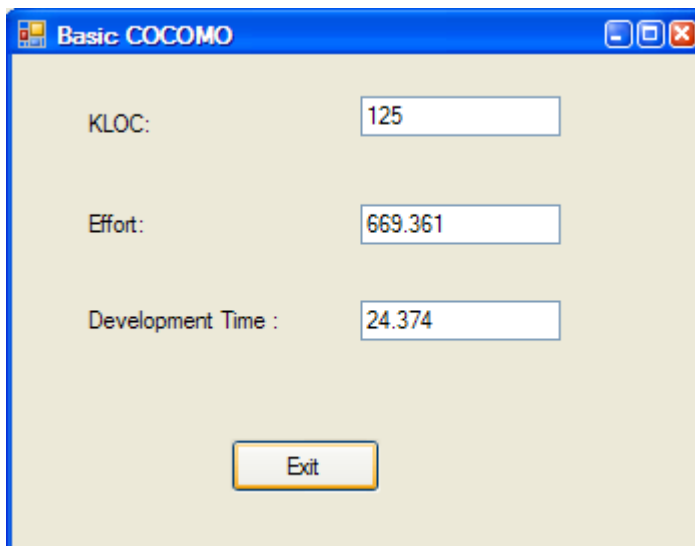




## Basic COCOMO



The screenshot shows the 'Basic COCOMO' application window. It features a text input field for 'Enter Lines of Code (KLOC):' containing the value '125'. Below this is a 'Mode Selection' section with three radio buttons: 'Organic', 'Semi-Detached' (which is selected), and 'Embedded'. At the bottom of the window are three buttons: 'Proceed', 'Clear', and 'Exit'.



The screenshot shows the 'Basic COCOMO' application window displaying the results of the calculation. It contains three text input fields with the following values: 'KLOC:' with '125', 'Effort:' with '669.361', and 'Development Time :' with '24.374'. A single 'Exit' button is located at the bottom center of the window.

## Intermediate COCOMO

The screenshot shows a window titled "Intermediate COCOMO". It features a text input field labeled "Enter Lines of Code (KLOC):" containing the value "120". Below this is a "Mode Selection" section with three radio buttons: "Organic", "Semi-Detached", and "Embedded". The "Embedded" option is selected. At the bottom of the window are three buttons: "Proceed", "Clear", and "Exit".

The screenshot shows a window titled "Intermediate COCOMO- Cost Drivers". It is titled "PRODUCT ATTRIBUTES" and contains three sections of radio button options:

- Required Software Reliability:** Options are "Very Low", "Low", "Nominal", "High", and "Very High". The "High" option is selected.
- Database Size:** Options are "Low", "Nominal", "High", and "Very High". The "Nominal" option is selected.
- Product Complexity:** Options are "Very Low", "Low", "Nominal", "High", "Very High", and "Extra High". The "High" option is selected.

At the bottom of the window are three buttons: "Proceed", "Clear", and "Exit".

## Intermediate COCOMO

Intermediate COCOMO- Cost Drivers

Execution Time Constraint

Nominal  High  Very High  Extra High

Main Storage Constraint

Nominal  High  Very High  Extra High

Virtual Machine Volatility

Low  Nominal  High  Very High

Computer Turnaround Time

Low  Nominal  High  Very High

Intermediate COCOMO- Cost Drivers

PERSONNEL ATTRIBUTES

Analyst Capability

Very Low  Low  Nominal  High  Very High

Application Experience

Very Low  Low  Nominal  High  Very High

Programmer Capability

Very Low  Low  Nominal  High  Very High

Virtual Machine Experience

Very Low  Low  Nominal  High

Programming Language Experience

Very Low  Low  Nominal  High

## Intermediate COCOMO

Intermediate COCOMO- Cost Drivers

PROJECT ATTRIBUTES

Modern Programming Practices

Very Low  Low  Nominal  High  Very High

Use of Software Tools

Very Low  Low  Nominal  High  Very High

Required Development Schedule

Very Low  Low  Nominal  High  Very High

Proceed Clear Exit

Intermediate COCOMO

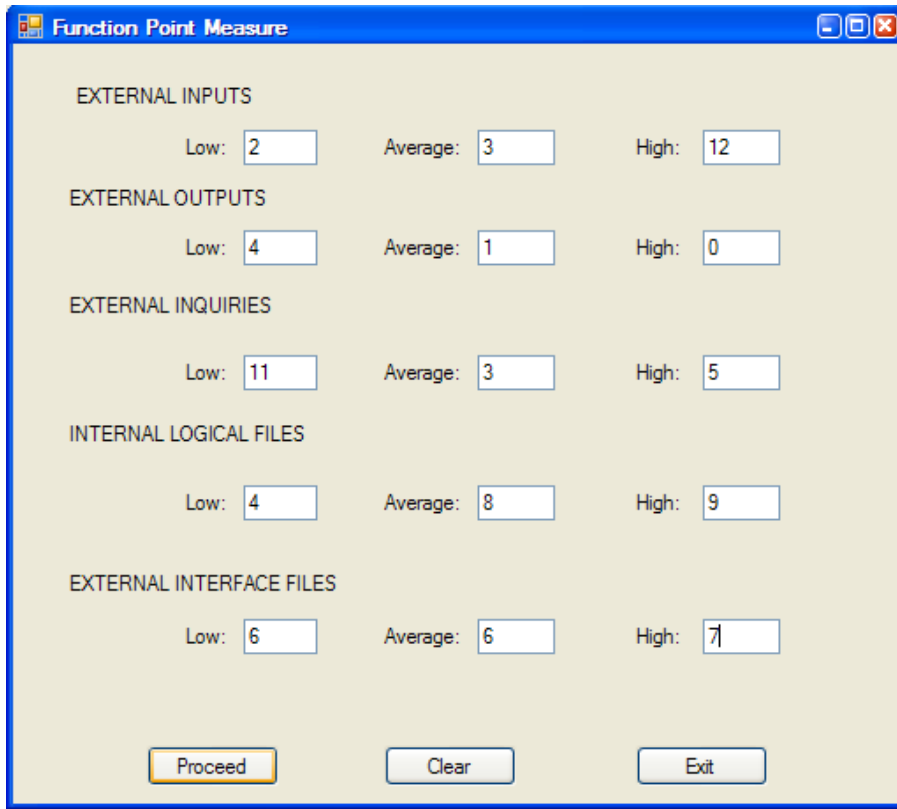
KLOC: 120

Effort: 1915.869

Development Time : 28.074

Exit

## Function Point Measure



Function Point Measure

EXTERNAL INPUTS

Low:  Average:  High:

EXTERNAL OUTPUTS

Low:  Average:  High:

EXTERNAL INQUIRIES

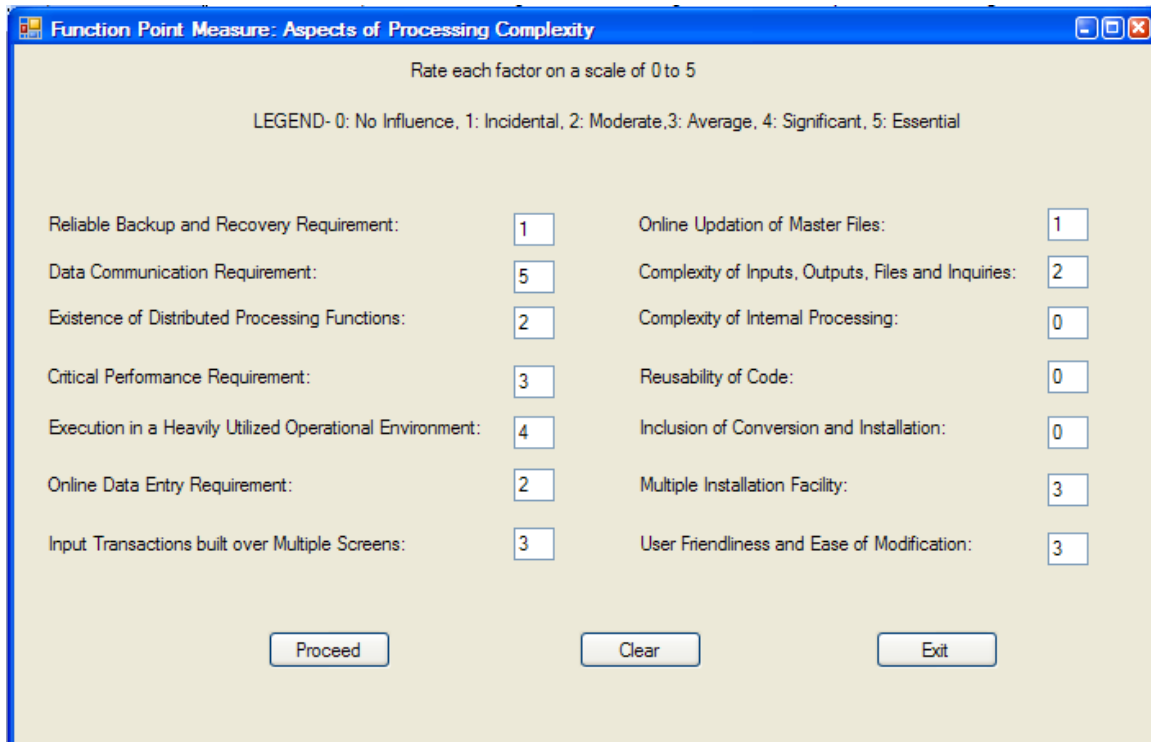
Low:  Average:  High:

INTERNAL LOGICAL FILES

Low:  Average:  High:

EXTERNAL INTERFACE FILES

Low:  Average:  High:



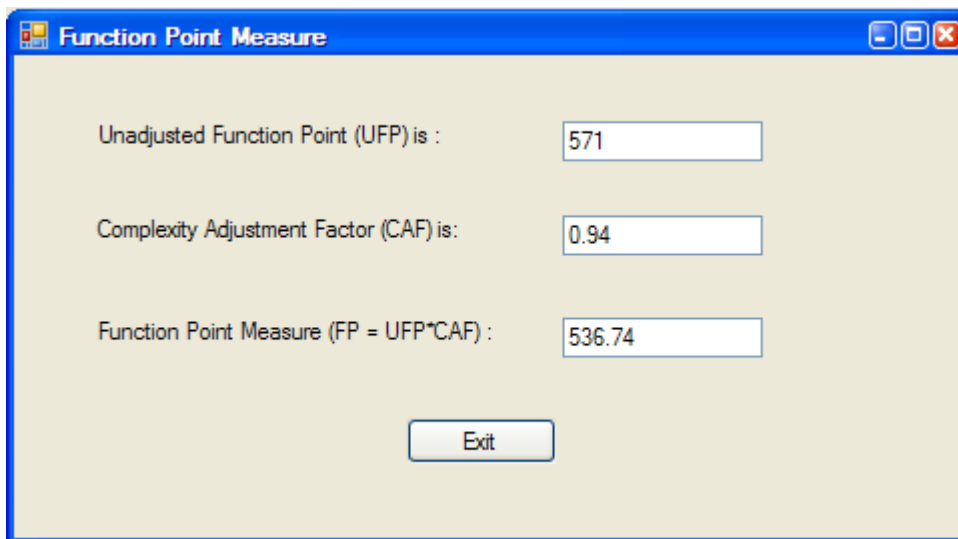
Function Point Measure: Aspects of Processing Complexity

Rate each factor on a scale of 0 to 5

LEGEND- 0: No Influence, 1: Incidental, 2: Moderate, 3: Average, 4: Significant, 5: Essential

Reliable Backup and Recovery Requirement:	<input type="text" value="1"/>	Online Updation of Master Files:	<input type="text" value="1"/>
Data Communication Requirement:	<input type="text" value="5"/>	Complexity of Inputs, Outputs, Files and Inquiries:	<input type="text" value="2"/>
Existence of Distributed Processing Functions:	<input type="text" value="2"/>	Complexity of Internal Processing:	<input type="text" value="0"/>
Critical Performance Requirement:	<input type="text" value="3"/>	Reusability of Code:	<input type="text" value="0"/>
Execution in a Heavily Utilized Operational Environment:	<input type="text" value="4"/>	Inclusion of Conversion and Installation:	<input type="text" value="0"/>
Online Data Entry Requirement:	<input type="text" value="2"/>	Multiple Installation Facility:	<input type="text" value="3"/>
Input Transactions built over Multiple Screens:	<input type="text" value="3"/>	User Friendliness and Ease of Modification:	<input type="text" value="3"/>

## Function Point Measure

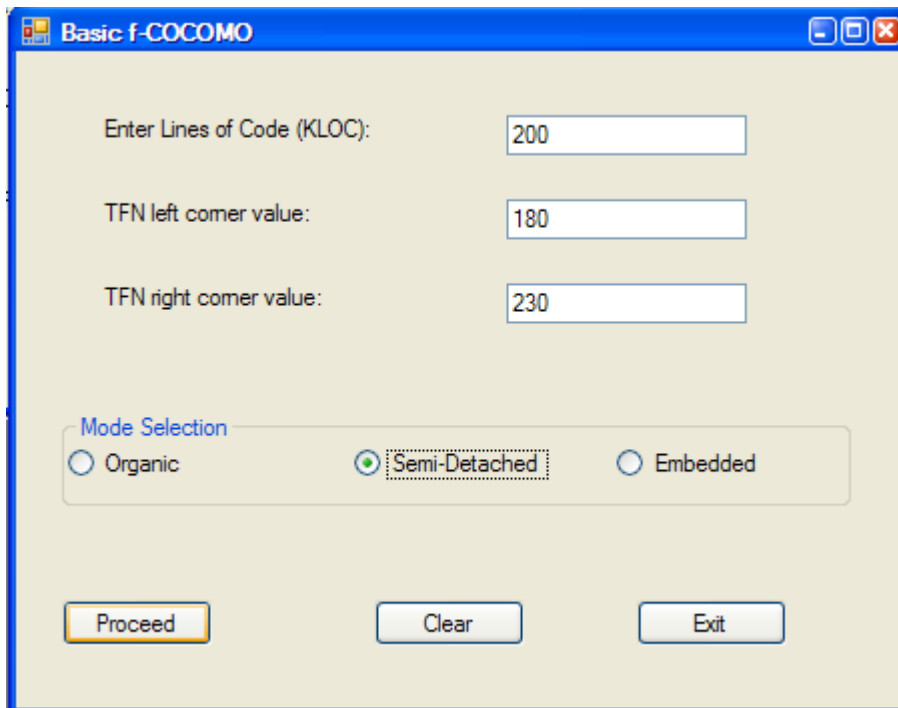


The screenshot shows a window titled "Function Point Measure" with a blue title bar and standard window controls. The main area has a light beige background and contains three rows of text with corresponding input fields:

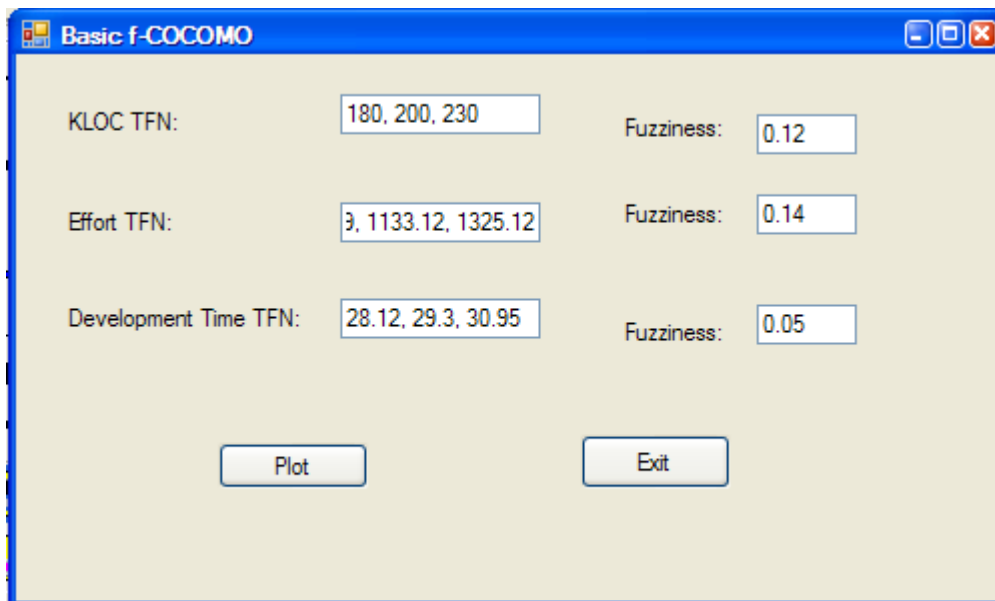
- Unadjusted Function Point (UFP) is :
- Complexity Adjustment Factor (CAF) is:
- Function Point Measure (FP = UFP\*CAF) :

At the bottom center, there is an "Exit" button.

## Basic f-COCOMO

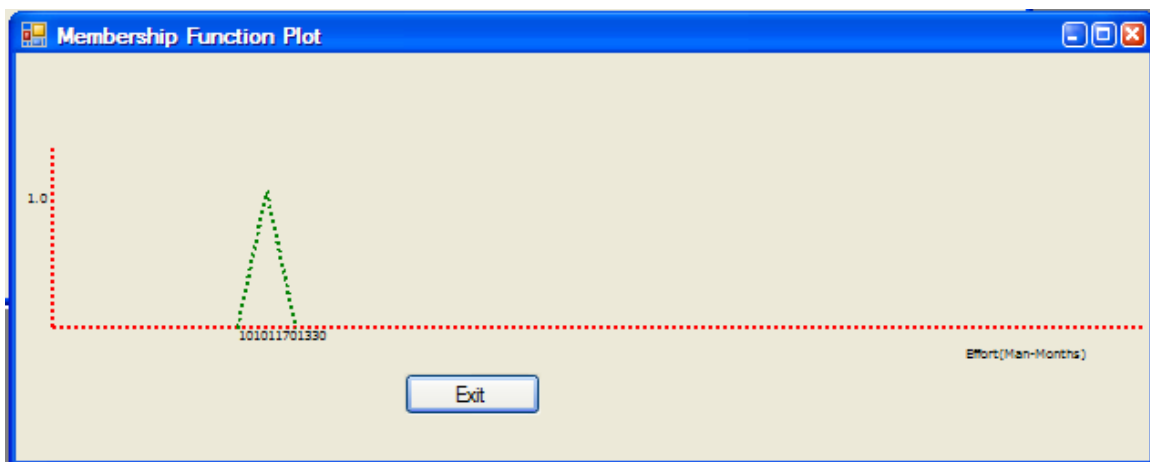
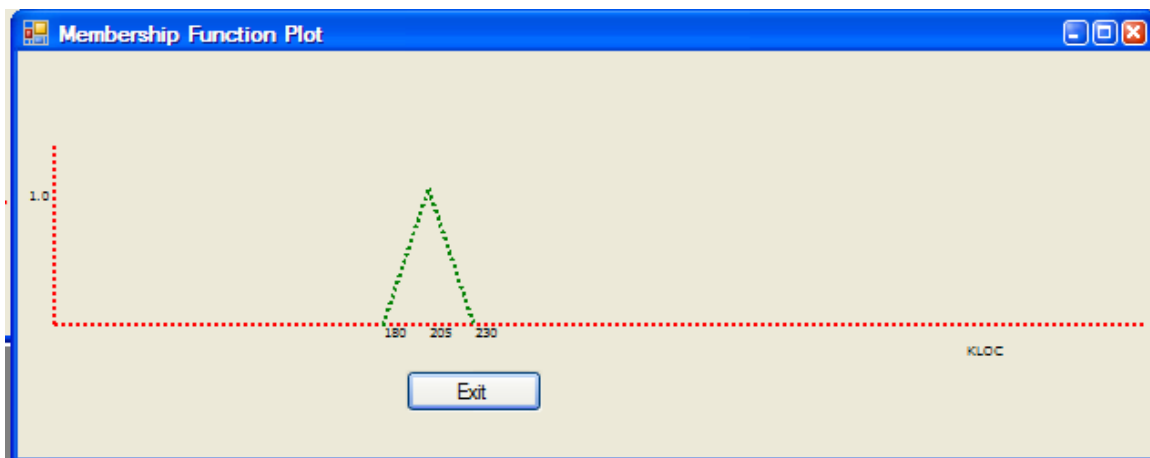
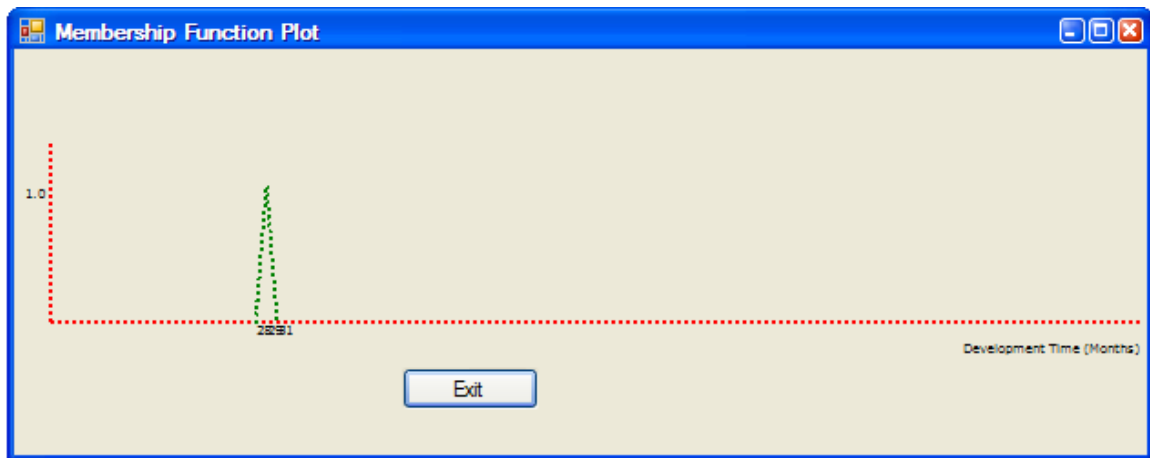


The screenshot shows the 'Basic f-COCOMO' application window. It has a blue title bar with the text 'Basic f-COCOMO' and standard window control buttons (minimize, maximize, close). The main area is light beige and contains three input fields for numerical values: 'Enter Lines of Code (KLOC):' with the value '200', 'TFN left comer value:' with '180', and 'TFN right comer value:' with '230'. Below these is a 'Mode Selection' section with three radio buttons: 'Organic' (unselected), 'Semi-Detached' (selected), and 'Embedded' (unselected). At the bottom, there are three buttons: 'Proceed' (highlighted in yellow), 'Clear', and 'Exit'.



The screenshot shows the 'Basic f-COCOMO' application window displaying calculated results. The title bar and window controls are the same as in the previous screenshot. The main area is light beige and contains three rows of data, each with a label, a list of values in a text box, and a 'Fuzziness' value in another text box: 'KLOC TFN:' with values '180, 200, 230' and 'Fuzziness: 0.12'; 'Effort TFN:' with values '3, 1133.12, 1325.12' and 'Fuzziness: 0.14'; and 'Development Time TFN:' with values '28.12, 29.3, 30.95' and 'Fuzziness: 0.05'. At the bottom, there are two buttons: 'Plot' and 'Exit'.

## Basic f-COCOMO





## Intermediate f-COCOMO

The screenshot shows a window titled "Intermediate f-COCOMO". It contains three input fields for numerical values: "Enter Lines of Code (KLOC):" with the value 120, "TFN Left Corner Value:" with the value 80, and "TFN Right Corner Value:" with the value 140. Below these fields is a "Mode Selection" section with three radio buttons: "Organic" (which is selected), "Semi-Detached", and "Embedded". At the bottom of the window are three buttons: "Proceed", "Clear", and "Exit".

The screenshot shows a window titled "IFC: Cost Drivers - Product Attributes". It is titled "PRODUCT ATTRIBUTES" and contains three sections of radio button options. The first section, "Required Software Reliability", has options: Very Low, Low (selected), Nominal, High, and Very High. The second section, "Database Size", has options: Low, Nominal, High (selected), and Very High. The third section, "Product Complexity", has options: Very Low, Low, Nominal, High, Very High (selected), and Extra High. At the bottom of the window are three buttons: "Proceed", "Clear", and "Exit".

## Intermediate f-COCOMO

IFC: Cost Drivers- Computer Attributes

COMPUTER ATTRIBUTES

Execution Time Constraint

Nominal  High  Very High  Extra High

Main Storage Constraint

Nominal  High  Very High  Extra High

Virtual Machine Volatility

Low  Nominal  High  Very High

Computer Turnaround Time

Low  Nominal  High  Very High

IFC: Cost Drivers- Personnel Attributes

PERSONNEL ATTRIBUTES

Analyst Capability

Very Low  Low  Nominal  High  Very High

Application Experience

Very Low  Low  Nominal  High  Very High

Programmer Capability

Very Low  Low  Nominal  High  Very High

Virtual Machine Experience

Very Low  Low  Nominal  High

Programming Language Experience

Very Low  Low  Nominal  High

## Intermediate f-COCOMO

IFC: Cost Drivers- Project Attributes

PROJECT ATTRIBUTES

Modem Programming Practices

Very Low    Low    Nominal    High    Very High

Use of Software Tools

Very Low    Low    Nominal    High    Very High

Required Development Schedule

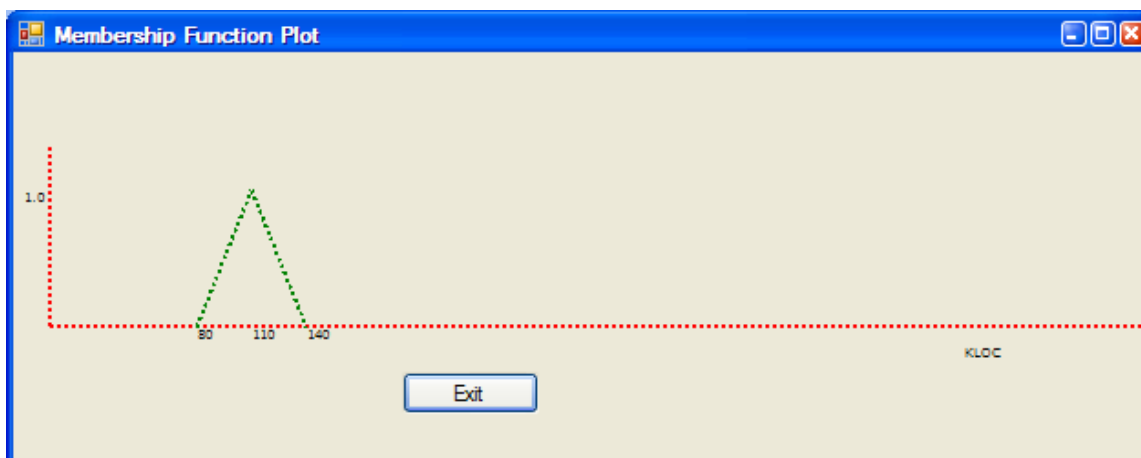
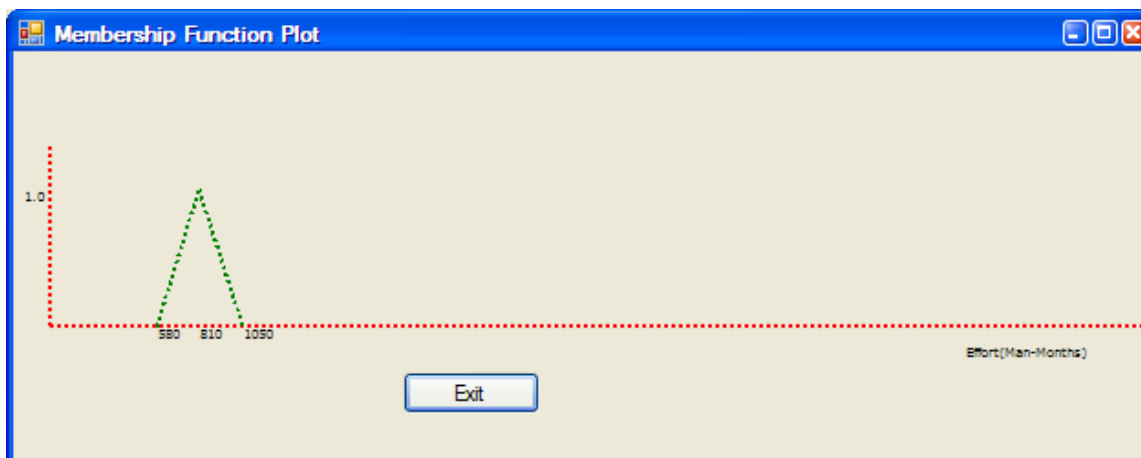
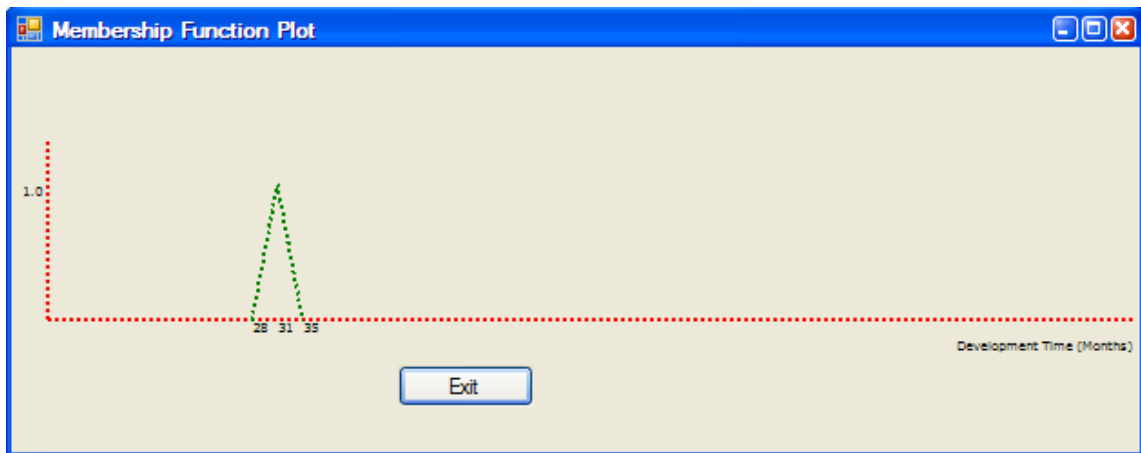
Very Low    Low    Nominal    High    Very High

Intermediate f-COCOMO

KLOC TFN:	<input type="text" value="80, 120, 140"/>	Fuzziness:	<input type="text" value="0.25"/>
Effort TFN:	<input type="text" value="581.4, 889.95, 104€"/>	Fuzziness:	<input type="text" value="0.26"/>
Development Time TFN:	<input type="text" value="28.08, 33.01, 35.11"/>	Fuzziness:	<input type="text" value="0.11"/>

## Intermediate f-COCOMO



## Fuzzy Inference Based Estimation

The screenshot shows a window titled "Inference Based Estimation". It contains two input fields: "Enter Lines of Code (KLOC):" with the value "200" and "Enter Complexity (Scale- 0 to 10):" with the value "4". To the right of the complexity field are two buttons: "Edit Membership Functions" and "Edit Rule Base". Below these fields is a radio button labeled "Triangular Membership Function" which is selected. At the bottom of the window are three buttons: "Proceed", "Clear", and "Exit".

The screenshot shows a window titled "Triangular Membership Function Editor: Effort". It contains six input fields for defining membership functions: "Start Point (X-Axis) of Low Curve:" (0), "End Point (X-Axis) of Low Curve:" (400), "Start Point (X-Axis) of Medium Curve:" (115), "End Point (X-Axis) of Medium Curve:" (2945), "Start Point (X-Axis) of High Curve:" (1455), and "End Point (X-Axis) of High Curve:" (4850). Below these fields are three buttons: "Save", "Clear", and "Default Values". At the bottom of the window, there is a note: "Note: Range is 0 to 6500 for Effort".

## Fuzzy Inference Based Estimation

**Triangular Membership Function Editor - Complexity**

Start Point (X-Axis) of Low Curve:

End Point (X-Axis) of Low Curve:

Start Point (X-Axis) of Medium Curve:

End Point (X-Axis) of Medium Curve:

Start Point (X-Axis) of High Curve:

End Point (X-Axis) of High Curve:

Note: Range is 1 to 10 for Complexity

**Triangular Membership Function Editor: KLOC**

Start Point (X-Axis) of Small Curve:

End Point (X-Axis) of Small Curve:

Start Point (X-Axis) of Medium Curve:

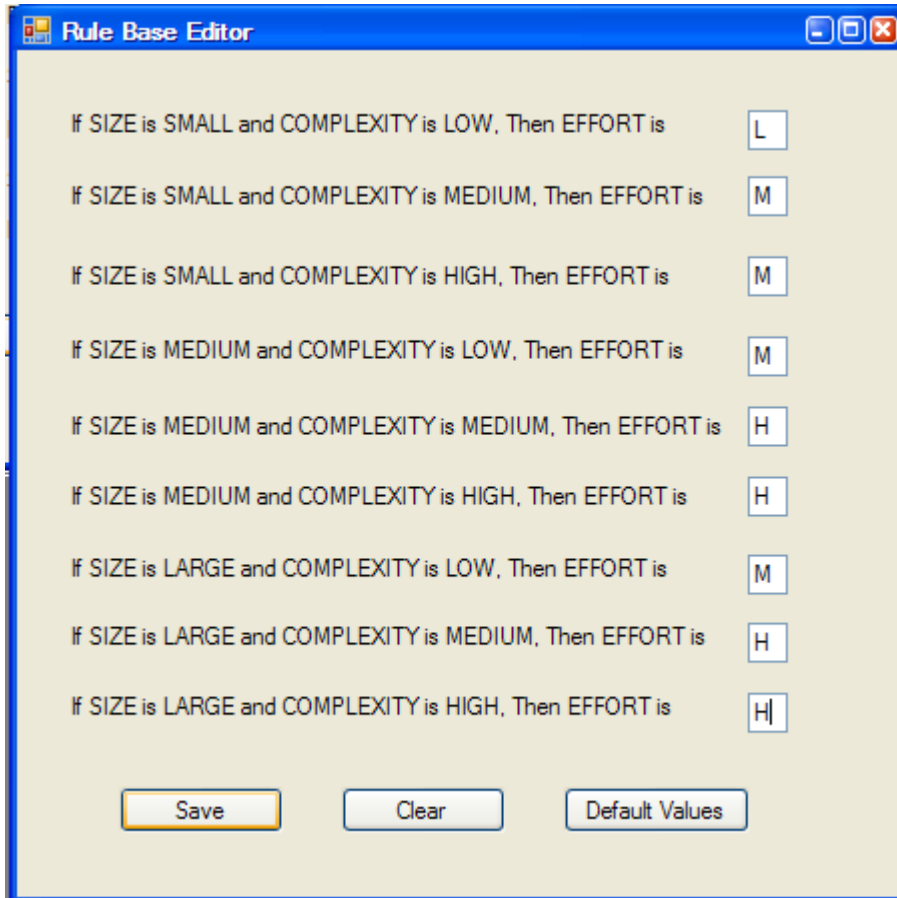
End Point (X-Axis) of Medium Curve:

Start Point (X-Axis) of Large Curve:

End Point (X-Axis) of Large Curve:

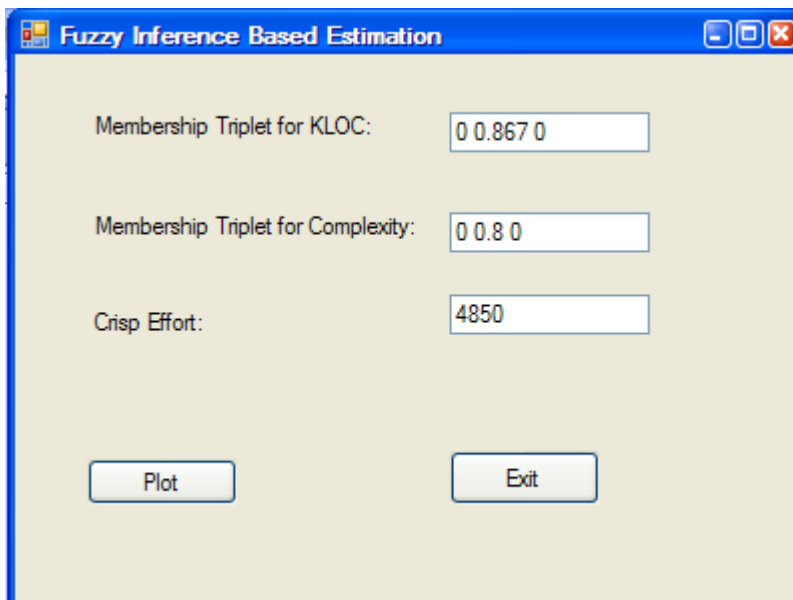
Note: Range is 0 to 500 KLOC

## Fuzzy Inference Based Estimation



**Rule Base Editor**

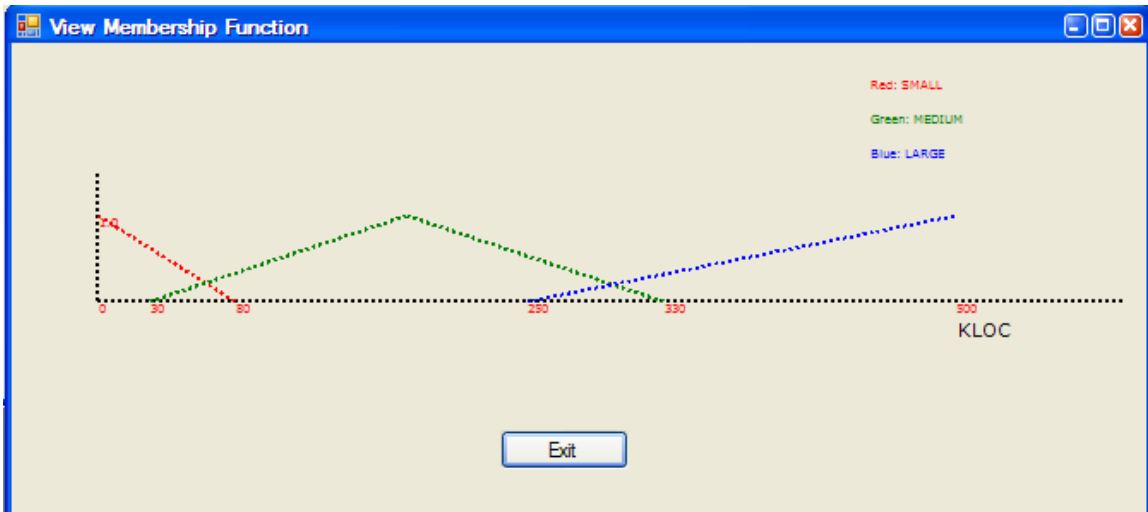
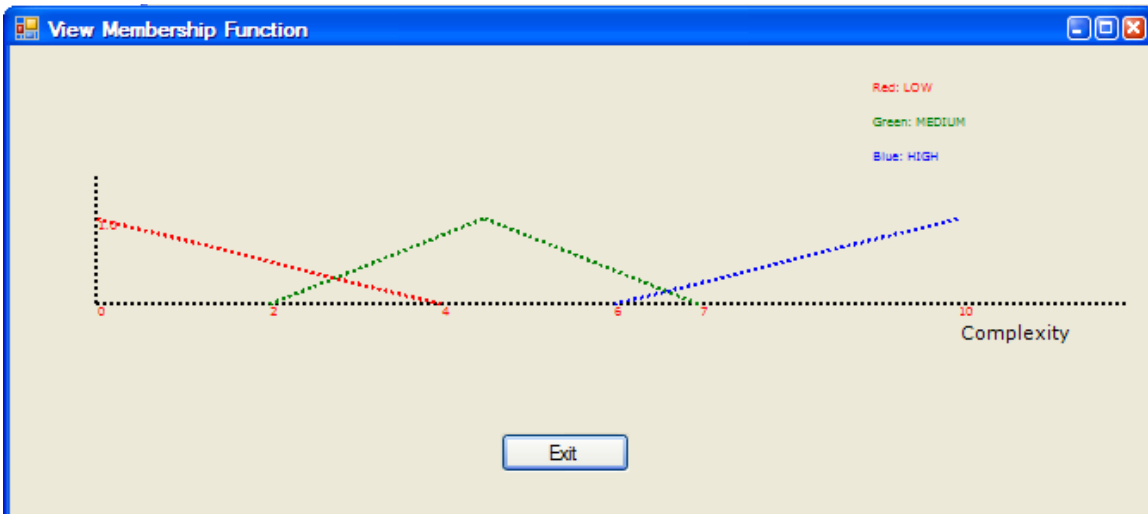
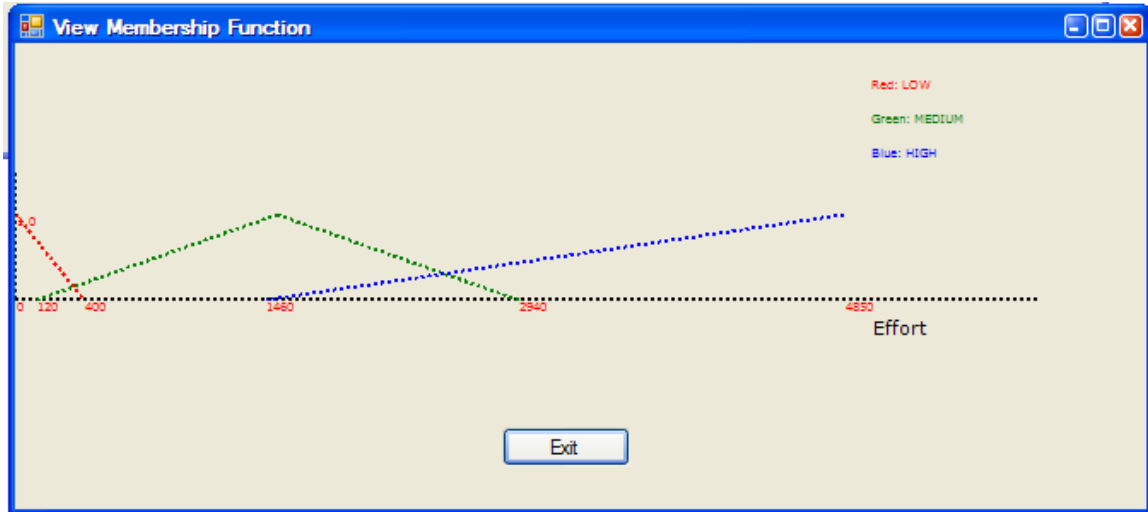
If SIZE is SMALL and COMPLEXITY is LOW, Then EFFORT is	L
If SIZE is SMALL and COMPLEXITY is MEDIUM, Then EFFORT is	M
If SIZE is SMALL and COMPLEXITY is HIGH, Then EFFORT is	M
If SIZE is MEDIUM and COMPLEXITY is LOW, Then EFFORT is	M
If SIZE is MEDIUM and COMPLEXITY is MEDIUM, Then EFFORT is	H
If SIZE is MEDIUM and COMPLEXITY is HIGH, Then EFFORT is	H
If SIZE is LARGE and COMPLEXITY is LOW, Then EFFORT is	M
If SIZE is LARGE and COMPLEXITY is MEDIUM, Then EFFORT is	H
If SIZE is LARGE and COMPLEXITY is HIGH, Then EFFORT is	H



**Fuzzy Inference Based Estimation**

Membership Triplet for KLOC:	<input type="text" value="0 0.867 0"/>
Membership Triplet for Complexity:	<input type="text" value="0 0.8 0"/>
Crisp Effort:	<input type="text" value="4850"/>

## Fuzzy Inference Based Estimation





## Class Point Measure

**Class Point Measure**

PROBLEM DOMAIN

Low:  Average:  High:

HUMAN INTERACTION TYPE

Low:  Average:  High:

DATA MANAGEMENT TYPE

Low:  Average:  High:

TASK MANAGEMENT TYPE

Low:  Average:  High:

**Class Point Measure: Aspects of Processing Complexity**

Rate each factor on a scale of 0 to 5

LEGEND- 0: No Influence, 1: Incidental, 2: Moderate, 3: Average, 4: Significant, 5: Essential

Data Communication:	<input type="text" value="1"/>	Reusability:	<input type="text" value="5"/>
Distributed Functions:	<input type="text" value="3"/>	Installation Ease:	<input type="text" value="5"/>
Performance:	<input type="text" value="2"/>	Operational Ease:	<input type="text" value="5"/>
Heavily used Configuration:	<input type="text" value="3"/>	Multiple Sites:	<input type="text" value="4"/>
Transaction Rate:	<input type="text" value="3"/>	Facilitation of Change:	<input type="text" value="3"/>
Online Data Entry:	<input type="text" value="0"/>	User Adaptivity:	<input type="text" value="4"/>
End User Efficiency:	<input type="text" value="0"/>	Rapid Prototyping:	<input type="text" value="1"/>
Online Update:	<input type="text" value="2"/>	Multuser Interactivity:	<input type="text" value="2"/>
Complex Processing:	<input type="text" value="3"/>	Multiple Interfaces:	<input type="text" value="4"/>

## Class Point Measure

Class Point Measure

Total Unadjusted Class Point (TUCP) is : 501

Technical Complexity Factor (TCF) is: 1.05

Class Point Measure (CP = TUCP\*TCF) : 526.05

Exit

# KLOC Validation

**KLOC Validation**

Start Point (X-Axis) of Small Curve:

End Point (X-Axis) of Small Curve:

Start Point (X-Axis) of Medium Curve:

End Point (X-Axis) of Medium Curve:

Start Point (X-Axis) of Large Curve:

End Point (X-Axis) of Large Curve:

Note: Range is 0 to 500 KLOC

**KLOC Validation**

SIMILARITY INDICES:

Small-Medium

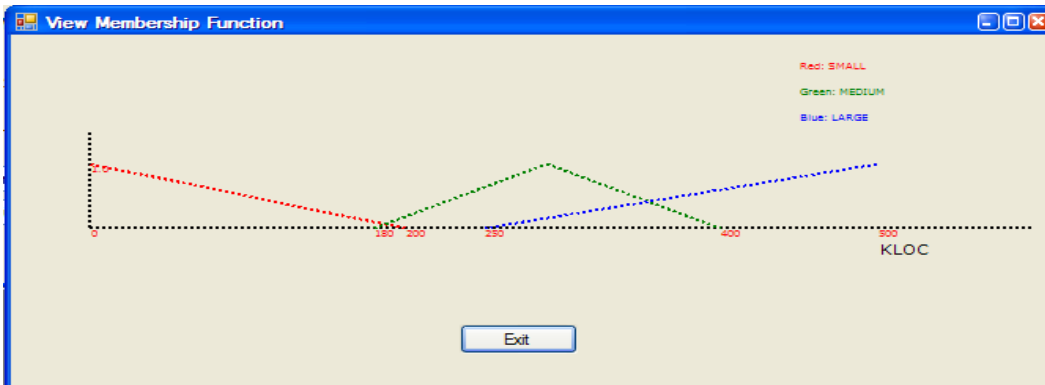
Medium-Large

Small-Large

COMMENTS ON KLOC FUZZY PARTITION:

Poor Overlap Region for Low-Medium, Medium-High Partition is

Low-Medium Distinguishability is Acceptable Medium-High Distir



## Complexity Validation

**Complexity Validation**

Start Point (X-Axis) of Small Curve:

End Point (X-Axis) of Small Curve:

Start Point (X-Axis) of Medium Curve:

End Point (X-Axis) of Medium Curve:

Start Point (X-Axis) of Large Curve:

End Point (X-Axis) of Large Curve:

Note: Range is 0 to 10 for Complexity

**Complexity Validation**

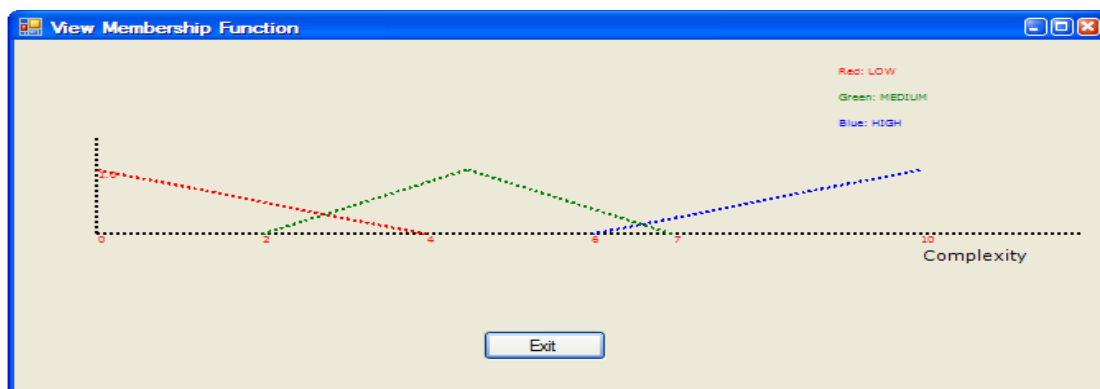
SIMILARITY INDICES:

Low-Medium:

Medium-High:

Low-High:

COMMENTS ON COMPLEXITY FUZZY PARTITION:



## Effort Validation

**Effort Validation**

Start Point (X-Axis) of Small Curve:

End Point (X-Axis) of Small Curve:

Start Point (X-Axis) of Medium Curve:

End Point (X-Axis) of Medium Curve:

Start Point (X-Axis) of Large Curve:

End Point (X-Axis) of Large Curve:

Note: Range is 0 to 5000 for Effort

**Effort Validation**

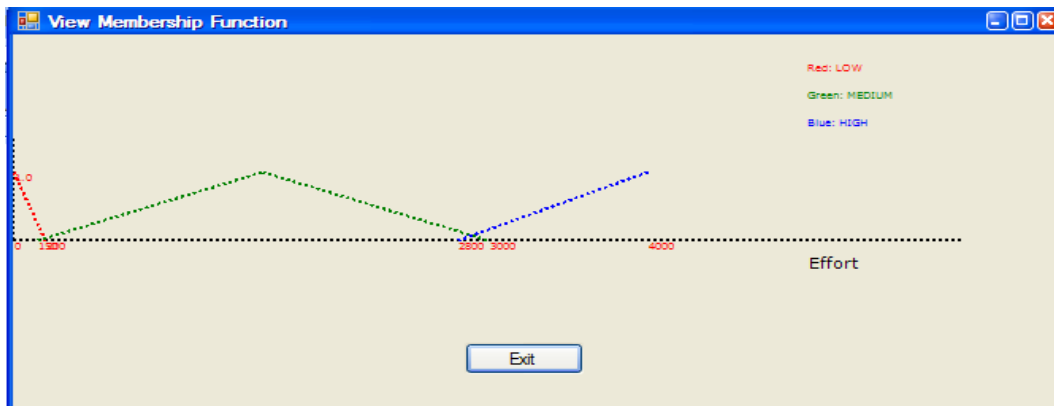
SIMILARITY INDICES:

Low-Medium

Medium-High

Low-High

COMMENTS ON EFFORT FUZZY PARTITION:



## Rule Base Validation

The screenshot shows a 'Rule Validation' dialog box with three tabs: 'Rule Base', 'Consistency Summary', and 'Rule Redundancy'. The 'Rule Base' tab is active, displaying a list of nine rules. Each rule is followed by a dropdown menu for the effort level. The effort levels are: Rule 1 (LOW), Rule 2 (MEDIUM), Rule 3 (MEDIUM), Rule 4 (MEDIUM), Rule 5 (HIGH), Rule 6 (HIGH), Rule 7 (MEDIUM), Rule 8 (HIGH), and Rule 9 (HIGH). At the bottom of the dialog, there are three buttons: 'Default Values', 'Save', and 'Exit'.

Rule	Condition	Effort
RULE 1	If SIZE is SMALL and COMPLEXITY is LOW, then EFFORT is:	LOW
RULE 2	If SIZE is SMALL and COMPLEXITY is MEDIUM, then EFFORT is:	MEDIUM
RULE 3	If SIZE is SMALL and COMPLEXITY is HIGH, then EFFORT is:	MEDIUM
RULE 4	If SIZE is AVERAGE and COMPLEXITY is LOW, then EFFORT is:	MEDIUM
RULE 5	If SIZE is AVERAGE and COMPLEXITY is MEDIUM, then EFFORT is:	HIGH
RULE 6	If SIZE is AVERAGE and COMPLEXITY is HIGH, then EFFORT is:	HIGH
RULE 7	If SIZE is LARGE and COMPLEXITY is LOW, then EFFORT is:	MEDIUM
RULE 8	If SIZE is LARGE and COMPLEXITY is MEDIUM, then EFFORT is:	HIGH
RULE 9	If SIZE is LARGE and COMPLEXITY is HIGH, then EFFORT is:	HIGH

## Rule Base Validation

The screenshot displays the 'Rule Validation' application window. It features three tabs: 'Rule Base' (selected), 'Consistency Summary', and 'Rule Redundancy'. The main area contains nine rules, each with a condition and a corresponding effort level selected in a dropdown menu:

- RULE 1: If SIZE is SMALL and COMPLEXITY is LOW, then EFFORT is: **LOW**
- RULE 2: If SIZE is SMALL and COMPLEXITY is MEDIUM, then EFFORT is: **MEDIUM**
- RULE 3: If SIZE is SMALL and COMPLEXITY is HIGH, then EFFORT is: **MEDIUM**
- RULE 4: If SIZE is AVERAGE and COMPLEXITY is LOW, then EFFORT is: **MEDIUM**
- RULE 5: If SIZE is **SMALL** and COMPLEXITY is LOW, then EFFORT is: **MEDIUM**
- RULE 6: If SIZE is **SMALL** and COMPLEXITY is MEDIUM, then EFFORT is: **MEDIUM**
- RULE 7: If SIZE is **SMALL** and COMPLEXITY is HIGH, then EFFORT is: **MEDIUM**
- RULE 8: If SIZE is LARGE and COMPLEXITY is MEDIUM, then EFFORT is: **HIGH**
- RULE 9: If SIZE is LARGE and COMPLEXITY is HIGH, then EFFORT is: **HIGH**

A 'Success!' dialog box is overlaid on the window, containing the message: 'The values have been saved successfully. You may now load the Summary Report for these values.' and an 'OK' button. At the bottom of the main window, there are buttons for 'Default Values', 'Save', and 'Exit'.

## Rule Base Validation

The screenshot shows a software window titled "Rule Validation" with three tabs: "Rule Base", "Consistency Summary", and "Rule Redundancy". The "Consistency Summary" tab is active. A "Load Summary" button is located above a table. The table has columns for "Rule" and "R1" through "R5". Below the table is a scrollbar and a "Save to Inference Engine" button. At the bottom of the window is an "Exit" button.

Rule	R1	R2	R3	R4	R5
R1	1	0.9097	1	0.999	0
R2	0.9097	1	0.9997	0.9992	0.9997
R3	1	0.9997	1	1	0.9998
R4	0.999	0.9992	1	1	1
R5	0	0.9997	0.9998	1	1
R6	1	0.9998	0.9997	1	0.9997
R7	1	1	1	0.9992	0.9996
R8	1	1	1	0.9996	0.9992
R9	1	1	1	1	0.9997
▶▶					



## Rule Base Validation

The screenshot displays the 'Rule Validation' application window. It features three tabs: 'Rule Base', 'Consistency Summary', and 'Rule Redundancy'. The 'Rule Base' tab is active, showing a 'Load Summary' button and a table of rule values. A 'Success!' dialog box is overlaid on the table, indicating that the values have been successfully stored to the Rule Base of the Fuzzy Inference Based Estimation Model. Below the table, there are 'Save to Inference Engine' and 'Exit' buttons.

Rule	R1	R2	R3	R4	R5
R1	1	0.9097	1	0.999	0
R2	0.9097	1	0.9997	0.9992	0.9997
R3	1	0.9997	1	1	0.9998
R4	0.999	0.9992	1	1	1
R5	0	0.9997	0.9998	1	1
R6					
R7					
R8					
R9					
*					

Success!  
The values have been successfully stored to the Rule Base of the Fuzzy Inference Based Estimation Model!

OK

Save to Inference Engine

Exit

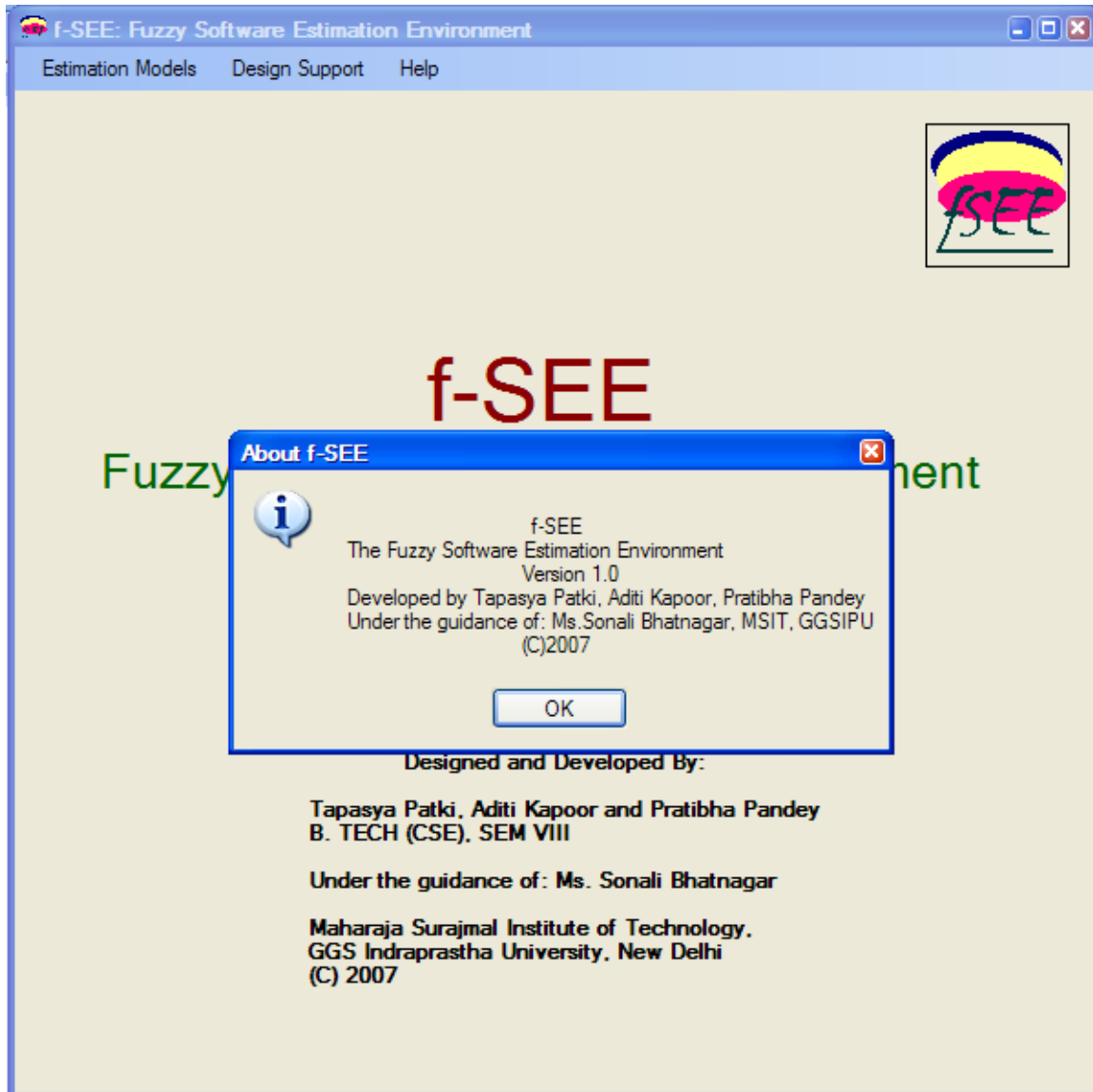
# Rule Base Validation

Select Rule for Redundancy Assessment: RULE 4 RULE 5 RULE 6 Proceed

	Rule:Rule	KLOC: Antecedent 1 Similarity	Complexity: Antecedent 2 Similarity	Rule Premise Consistency	Effort: Consequent Consistency	Rule Consistency
	8 : 1	0	0.073	0	0	1
	8 : 2	0	1	0	0.08	1
	8 : 3	0	0.017	0	0.08	1
	8 : 4	0.03	0.073	0.03	0.08	0.9996
	8 : 5	0.03	1	0.03	1	0.9992
	8 : 6	0.03	0.017	0.017	1	0.9997
	8 : 7	1	0.073	0.073	0.08	1
	8 : 8	1	1	1	1	1
	8 : 9	1	0.017	0.017	1	0.9997
▶*						

Exit

## About f-SEE



## 7.4 f-SEE Source Code File Sizes Summary Details

```
C:\tap8semprj\fSEE\fSEETap\fSEETap>dir *.cs
```

```
Volume in drive C has no label.
```

```
Volume Serial Number is 74A5-A460
```

```
Directory of C:\tap8semprj\fSEE\fSEETap\fSEETap
```

```
04/22/2007  12:48 AM                3,628 Form1.cs
04/22/2007  12:36 AM            19,603 Form1.Designer.cs
04/25/2007  08:15 PM                7,663 Form10.cs
04/25/2007  07:47 PM            7,558 Form10.Designer.cs
04/22/2007  09:49 PM                3,657 Form11.cs
03/21/2007  11:32 PM                2,165 Form11.Designer.cs
03/25/2007  11:47 AM                1,819 Form12.cs
03/23/2007  10:14 PM                5,874 Form12.Designer.cs
03/25/2007  12:04 PM                4,202 Form13.cs
03/25/2007  12:01 PM            10,467 Form13.Designer.cs
03/23/2007  10:45 PM                1,773 Form14.cs
03/23/2007  02:45 PM            9,955 Form14.Designer.cs
03/23/2007  04:09 PM                6,021 Form15.cs
03/23/2007  04:09 PM            12,345 Form15.Designer.cs
03/25/2007  11:46 AM                6,410 Form16.cs
03/25/2007  11:36 AM                2,048 Form16.Designer.cs
03/25/2007  11:37 AM                2,187 Form17.cs
03/25/2007  10:53 AM                7,250 Form17.Designer.cs
03/24/2007  11:53 AM                 745 Form18.cs
03/24/2007  11:53 AM                5,092 Form18.Designer.cs
03/25/2007  10:50 AM                2,048 Form19.cs
03/25/2007  10:50 AM                7,190 Form19.Designer.cs
04/25/2007  07:38 PM                5,135 Form2.cs
```

03/18/2007	07:50 PM	9,791	Form2.Designer.cs
03/24/2007	05:03 PM	3,188	Form20.cs
03/24/2007	04:57 PM	16,652	Form20.Designer.cs
03/24/2007	05:10 PM	3,554	Form21.cs
03/24/2007	05:07 PM	18,201	Form21.Designer.cs
03/24/2007	05:23 PM	4,603	Form22.cs
03/24/2007	05:23 PM	23,830	Form22.Designer.cs
03/24/2007	09:57 PM	3,376	Form23.cs
03/24/2007	09:57 PM	16,761	Form23.Designer.cs
03/24/2007	10:11 PM	952	Form24.cs
03/24/2007	10:12 PM	5,113	Form24.Designer.cs
03/25/2007	11:37 AM	1,843	Form25.cs
03/25/2007	11:27 AM	9,950	Form25.Designer.cs
03/25/2007	10:09 PM	336	Form26.cs
03/25/2007	10:09 PM	1,408	Form26.Designer.cs
04/26/2007	11:03 AM	11,170	Form27.cs
03/27/2007	12:16 AM	10,559	Form27.Designer.cs
03/31/2007	05:59 PM	1,701	Form28.cs
03/31/2007	05:57 PM	8,256	Form28.Designer.cs
04/26/2007	11:20 AM	5,500	Form29.cs
03/27/2007	12:58 AM	10,639	Form29.Designer.cs
04/22/2007	09:53 PM	2,307	Form3.cs
03/21/2007	08:13 PM	9,031	Form3.Designer.cs
03/31/2007	05:59 PM	1,682	Form30.cs
03/31/2007	05:59 PM	8,268	Form30.Designer.cs
04/26/2007	11:24 AM	5,414	Form31.cs
03/27/2007	01:17 AM	10,564	Form31.Designer.cs
03/31/2007	06:04 PM	1,882	Form32.cs
03/31/2007	06:02 PM	8,260	Form32.Designer.cs
04/26/2007	12:54 PM	8,515	Form33.cs
04/21/2007	09:55 PM	22,292	Form33.Designer.cs
04/26/2007	12:34 PM	11,365	Form34.cs

04/20/2007	08:14 PM	19,316	Form34.Designer.cs
04/26/2007	11:26 AM	20,120	Form35.cs
04/26/2007	11:26 AM	22,957	Form35.Designer.cs
04/26/2007	12:37 PM	862	Form36.cs
04/20/2007	08:26 PM	5,024	Form36.Designer.cs
04/26/2007	12:50 PM	6,395	Form37.cs
04/21/2007	09:53 PM	17,872	Form37.Designer.cs
04/26/2007	12:38 PM	14,136	Form38.cs
04/22/2007	12:00 AM	23,110	Form38.Designer.cs
04/26/2007	12:38 PM	882	Form39.cs
04/21/2007	11:46 PM	5,039	Form39.Designer.cs
04/25/2007	07:43 PM	4,544	Form4.cs
03/19/2007	08:32 PM	9,833	Form4.Designer.cs
04/26/2007	11:31 AM	536	Form40.cs
04/26/2007	11:31 AM	5,394	Form40.Designer.cs
03/22/2007	12:04 AM	3,595	Form5.cs
03/22/2007	12:04 AM	17,187	Form5.Designer.cs
03/22/2007	12:04 AM	3,593	Form6.cs
03/17/2007	04:43 PM	18,516	Form6.Designer.cs
03/24/2007	05:16 PM	5,114	Form7.cs
03/24/2007	05:16 PM	24,660	Form7.Designer.cs
03/22/2007	12:03 AM	3,400	Form8.cs
03/17/2007	07:07 PM	17,038	Form8.Designer.cs
04/25/2007	07:44 PM	2,665	Form9.cs
03/21/2007	11:49 PM	8,612	Form9.Designer.cs
03/11/2007	10:18 PM	474	Program.cs
	81 File(s)	652,672	bytes

## **CHAPTER 8**

### **FUTURE SCOPE**

---

f-SEE is an integrated software package for effort estimation (with design support) for software projects. Its usefulness in the initial phase of software projects tender bidding process is of paramount significance since it computes effort assessment using conventional as well as fuzzy models. f-SEE uses concepts of fuzzy set representation for the COCOMO model. It has also provisions to compare analysis using function point and class point conventional models. We discuss the future scope of work along the above lines.

The existing fuzzy model uses the popular triangular representation of fuzzy membership functions. It needs to be examined and compared whether the trapezoidal and S/Pi representation of membership functions will be more effective or the popular triangular membership function is adequate, especially during contract tender bidding process. f-SEE could be accordingly augmented to represent other forms of fuzzy membership functions.

The function point and class point modules use the required data as a user-supplied input. We have interacted with some software specialists and the opinion was that there should be a mechanism to assist the software professionals on this front also. We recommend that similar to Rational Rose philosophy of UML modeling, f-SEE could be provided with a front-end interface. Thus, the existing f-SEE GUI should be tied with the module to UML analysis of software thereby providing an extra resource for data input in addition to the existing user-supplied inputs for function / class point modules.

As suggested by Kitchenham and Heffery, misleading metrics and unsound analyses are the causes of concern amongst large software development houses like IBM, Australia. We see considerable future applications of f-SEE to expand on the fuzzy rule base modeling using AI techniques like evolutionary computing and genetic algorithm. It might be argued that because function point and class point approaches are not that useful

during the tender bidding process, these may be separated from the f-SEE. However, we strongly feel that fuzzy logic rule base could be a great source to overcome the lacuna of function point / class point methodologies in future. A work in this direction will be useful for software developers as well as outsourcing agencies that can have a joint, validated approach for assessing web enabled projects.



## CHAPTER 9

---

### REFERENCES

---

1. Aggarwal K.K., Singh Y., *Software Engineering*, New Age International Publishers, 2006
2. Arabshahi P., Marks R.J., Reed R., *Adaptation of Fuzzy Inferencing: A Survey*, Proc. IEEE/Nagoya Univ. Workshop, Learning and Adaptive Systems, Nagoya, Japan, Nov. 1993
3. Caine, A., and Pidducks A.B.,  $f^2$  COCOMO: *Estimating Software Project Effort and Cost*, Proceedings of the 6th International Workshop on Economic-Driven Software Engineering Research (EDSER-6), Edinburgh, Scotland: IEEE, 2004.
4. Chen W., Saif M., *A Novel Fuzzy System with Dynamic Rule Base*, IEEE Trans. on Fuzzy Systems, Vol. 13, No. 5, pp. 569-582, October 2005
5. Costagaliola G., and Tortora G., *Class Point: An Approach for the Size Estimation of Object-Oriented Systems*, IEEE Trans. On software Engineering, Vol. 31, No. 1, pp 52-74, Jan. 2005
6. Fei Z., *f-COCOMO: Fuzzy Constructive Cost Model in Software Engineering*, IEEE International Conference on Fuzzy Systems, pp. 331-337, March, 1992
7. Idri A., Abran A., and Kijri L., COCOMO Cost Model using Fuzzy Logic, 7<sup>th</sup> International Conference on Fuzzy Theory & Technology, NJ, 2000
8. Jantzen J., *Design of Fuzzy Controllers*, Technical University of Denmark, 98-E-864, 1998

9. Javier F.C., Sicilia M.A., and Cuadrado J.J., *On the Use of Fuzzy Regression in Parametric Software Estimation Models: Integrating Imprecision in COCOMO Cost Drivers*, accessed at <http://www.inf.uc3m.es>
10. Jin Y., von Seelen W., and Sendhoff B., Jin Y., *An Approach to Rule-Based Knowledge Extraction*, Proceedings of IEEE International Conference on Fuzzy Systems, Vol 2 pp. 1188-1193, May 1998
11. Jin Y., von Seelen W., and Sendhoff B., *On Generating  $FC^3$  Fuzzy rule systems from Data Using Evolution Strategies*, IEEE Trans. On Systems, Man and Cybernetics, Part B: Cybernetics, Vol. 29, No. 6, June, pp. 376-386, pp. 829-845, Dec. 1999
12. Jowers L.J., James J.B., and Reilly K.D., *Estimation of f-COCOMO Model Parameters using Optimization Techniques*, Oct 2006
13. Kapoor A., Patki T, Khurana S. – *Analytical Methodologies in Soft Computing Part-I: Exposure to cyber forensic Software tools Part II*, Training Report Submitted to DIT, July 2005
14. Kapoor A., Pandey P., *Fuzzy Implementation of Class Point Approach*, Training Report Submitted to DIT, July 2006
15. Kitchenham B., and Jeffery D.R., *Misleading Metrics and Unsound Analysis*, *IEEE Software*, pp. 73-78, Mar-Apr, 2007.
16. Klir G., and Folger T., *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, January 1988
17. MacDonell S.G., Gray A.R., and Calvert J.M., *FULSOME: Fuzzy Logic for Software Metrics Practitioners and Researchers*, ICONIP -1999, Proceedings of International conference on Neural Information Processing, IEEE, pp. 308-313, 1999

18. Musilek P., Pedrycz W., Succi G., and Reformat M., *Software Cost Estimation with Fuzzy Models*, ACM SIGAPP Applied computing Review, Vol. 8, Issue 2, 2000
19. Patki Tapasya, and Khurana Swati , *Software Cost Estimation and Software Obfuscation: A Fuzzy Logic Perspective*, Technical Report Submitted to Department of Information Technology, New Delhi, July 2006
20. Rajsekaran S., and Pai G.A.V, *Neural Networks, Fuzzy Logic, and Genetic Algorithms – Synthesis and Applications*, Prentice Hall India, 2006
21. Setnes M., Babuska R., Kaymak U., and Nauta Lemke H.R., *Similarity Measures in fuzzy Rule Base Simplification*, IEEE Trans. On Systems, Man and Cybernetics, Part B: Cybernetics, Vol. 28, No. 3, June, pp. 376-386, 1998
22. Zadeh, L. A., *Fuzzy Sets*, Information and Control, (8), pp. 338-353, 1965
23. Zadeh, L. A., *A Fuzzy-Algorithmic Approach to the Definition of Complex or Imprecise Concepts*, Int. Journal Man-machine Studies, (8), pp. 249-291, 1976