Application of Evolutionary Algorithm to find the trade-off between Complexity of Software and its Deliverability

By

SIDDHARTH LAVANIA

College of Engineering

Department of Computer Science & Engineering

Submitted

IN PARTIAL FULFILLMENT OF THE REQUIREMENT OF THE DEGREE OF DOCTOR OF PHILOSOPHY

TO

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN
July, 2015

ACKNOWLEDGEMENT

First of all I would like to dedicate my work to almighty god for being with me all the time which empowered me to go ahead with this work.

I am highly indebted to my Internal Guide **Dr. Neelu Jyoti Ahuja,** Associate Professor & Head, Computing Research Institute, R & D Centre, U.P.E.S. who has guided me all through the completion of this PhD thesis. My enthusiasm in this subject is inspired by her profound knowledge in this area and her elegant research style. She has helped me with her wise advice, useful discussions, comments and facilities.

I am very grateful to my External Co-Guide **Prof. (Dr.) Manuj Darbari**. His incessant enthusiasm, support, knowledge and inspiration have given a new dimension to my work. Without his sustained and sincere efforts, this thesis would not have taken this shape.

I am very grateful to my External Guide **Dr. Imran Ali Siddiqui** as he helped me and guide me throughout the research process to overcome the various difficulties and challenges that were raised at various stages of the project.

I am thankful to **Prof. (Dr.) Shrihari**, Vice Chancellor, U.P.E.S, for his guidance related to research norms of the Institute at all stages.

I want to express my deepest sense of gratitude to **Prof. (Dr.) Kamal Bansal,** Dean, College of Engineering Studies, U.P.E.S., for his kind co-operation and motivation during my research work.

I am very thankful to **Prof. (Dr.) Manish Prateek**, Head, Centre for Information Technology for providing me guidance during crucial times.

I am thankful to **Er. Mahesh Aggrawal**, Managing Director, Goel Institution of Technology& Management, Lucknow for his kind support and motivation in encouraging research activities .I am also grateful to **Dr. A. K. Bhardawaj,** Director, GITM, Lucknow for his guidance.

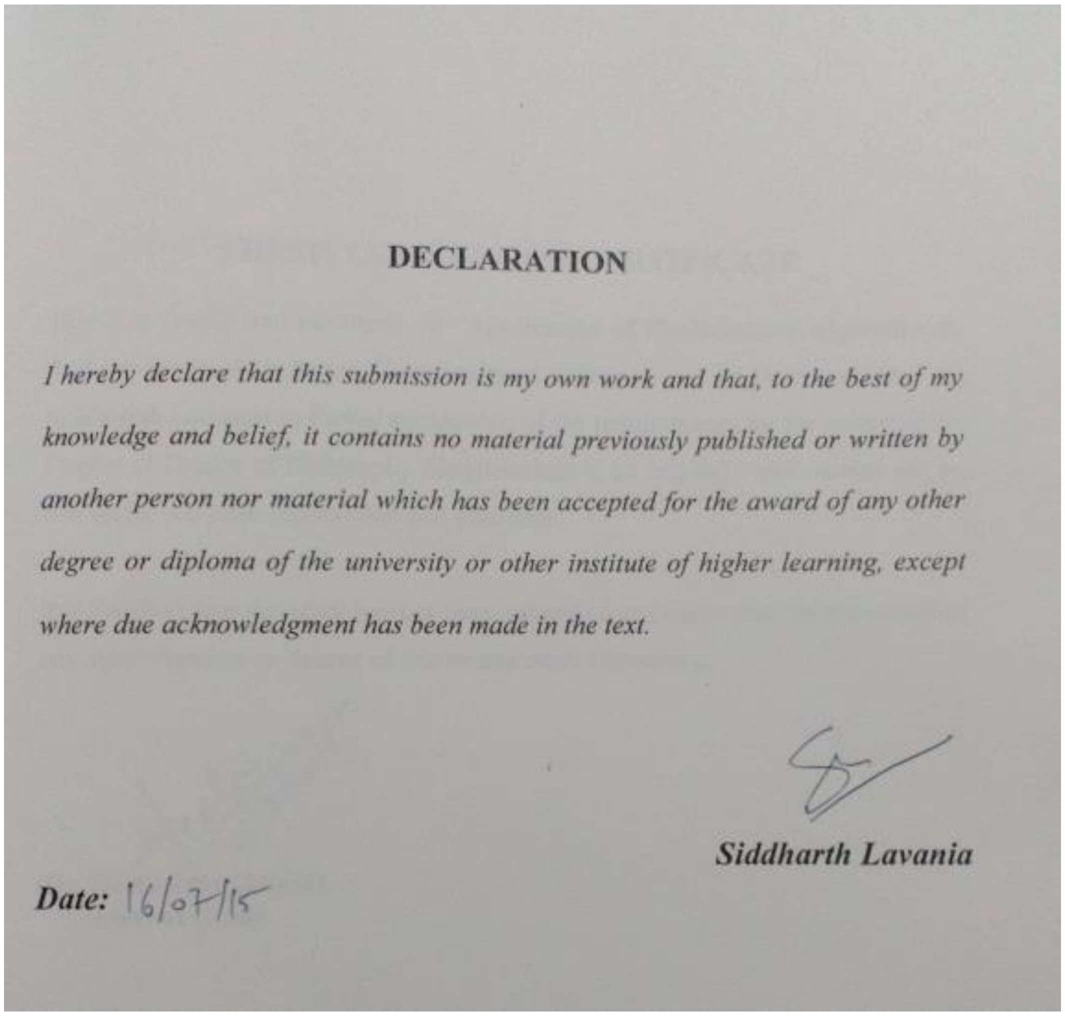
I would also like to thank **Dr. Namrata Dhanda**, Head, Department of Computer Science, Goel Institute of Technology and Management, Lucknow for her continuous support during my research work.

I wish to thank all my staff members and colleagues of the Computer Science/ Information Technology Department, Goel Institute of Technology and Management, Lucknow for their support and helpful suggestions.

I sincerely thank **Mr. Abhishank Maurya, Mr. Ankur Shukla, Mr. Baldev Verma, Mr. Vivek Dwivedi, Mr. Sushil Kumar** and **Mr. Siddharth Chaubey** for their everlasting support and cooperation throughout this work.

The vitality of this work was enhanced by the support I got from my family and friends. Last but not the least I would show my gratitude to my father **Dr. Anil Lavania,** my mother **Mrs. Pratibha Lavania,** my brother-in-law **Mr. Kunal Bhujbal**, my sister **Mrs. Parul Lavania Bhujbal** and my nephews **Ishaan** and **Vivaan** for their never ending support and motivation throughout my research work.

Siddharth Lavania



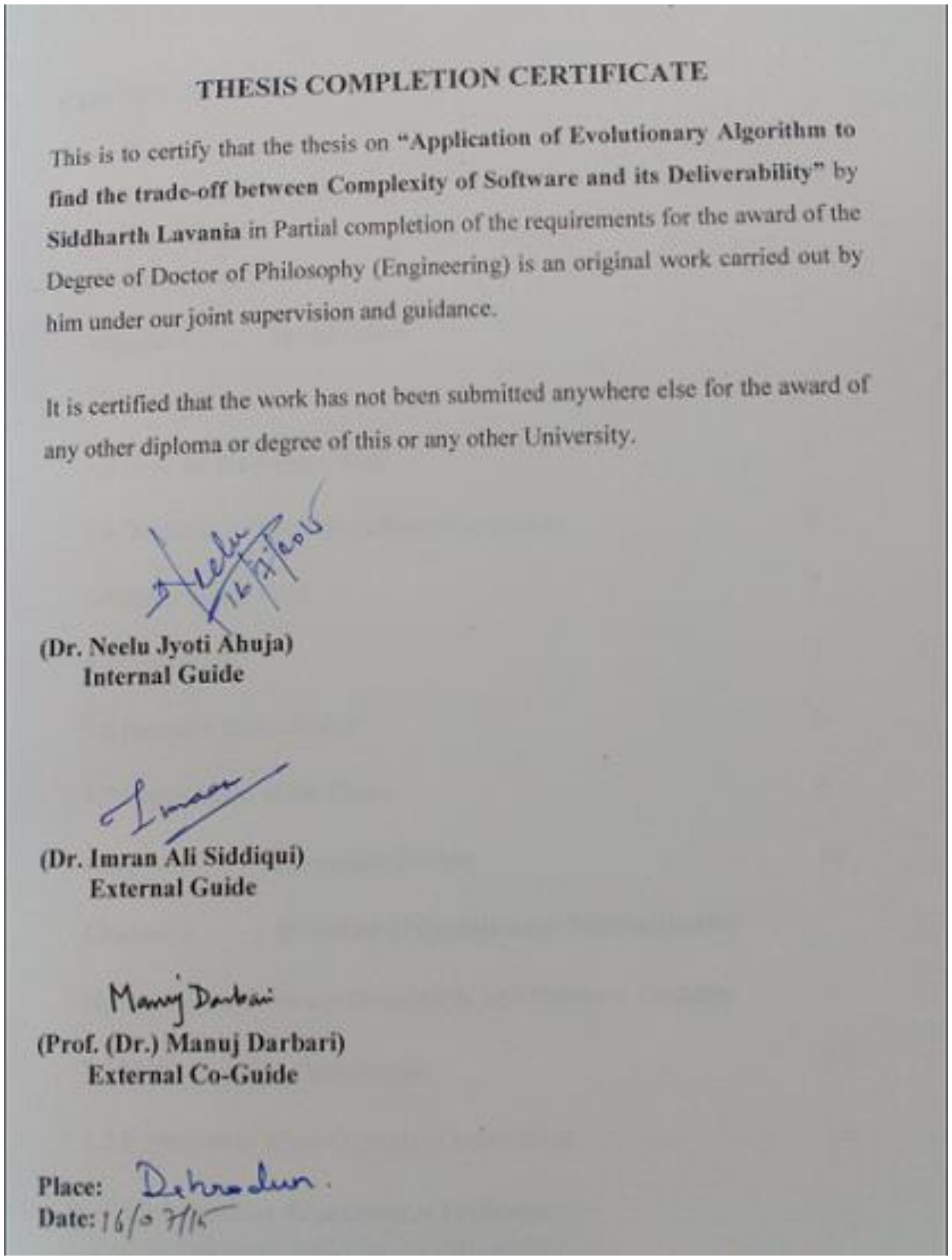
DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

[illegible]

Siddharth Lavania

Date: 16/07/15



THESIS COMPLETION CERTIFICATE

This is to certify that the thesis on **"Application of Evolutionary Algorithm to find the trade-off between Complexity of Software and its Deliverability"** by **Siddharth Lavania** in Partial completion of the requirements for the award of the Degree of Doctor of Philosophy (Engineering) is an original work carried out by him under our joint supervision and guidance.

It is certified that the work has not been submitted anywhere else for the award of any other diploma or degree of this or any other University.

[illegible]

(Dr. Neelu Jyoti Ahuja)
Internal Guide

[illegible]

(Dr. Imran Ali Siddiqui)
External Guide

Manuj Darbari

(Prof. (Dr.) Manuj Darbari)
External Co-Guide

Place: Dehradun.
Date: 16/07/15

CONTENTS

Application of Evolutionary Algorithm to find the trade-off between Complexity of Software and its Deliverability

EXECUTIVE SUMMARY

The term software complexity can be categorized in two ways. One is the code complexity which is not visible to the user and is the second one is the user interface complexity which is visible to the user. This research work is specifically about the second one that is user interface complexity aspect of any software.

The complexity of software with respect to the user varies from person to person. The same software at the same time could be a difficult to a person to work upon, very difficult for another person and at the same time could be simple for another person. This complexity or the comfort level varies because of social, economical and technical reasons. For example, an online shopping from a website could be a simple task for a person who has some prior knowledge about computer and internet but at the same time it is a complex procedure for those who are not familiar with the internet and computers. Similarly, if talk about the ERP software, that are being used by a lot of companies are having a different complexity level among the user in the same organization.

The high complexity of usage affects the market status of the software as the user finds it difficult to operate or to work upon and as a result, the software fails to deliver its maximum value to the organization which is the deliverability factor of the software.

It is the usability feature which plays a major role in having the product more sellable, but at the same time it should cater to all high level needs of the consumer. Although, there are number of companies developing certain guidelines about software development process, the major focus is on user-centred application development, software is evaluated with various tools like Cognitive tools and Complexity Matrices to find out the degree of acceptability amongst the users.

The deliverability of the software can further be described as the usability aspect of the software which is inversely proportional to the complexity of the software. The main problem arises to maintain a balance between complexity and deliverability, as both the quantities are very much inter-related, it is very difficult to raise the deliverability without increasing the complexity of the system.

This research is basically focuses on the user interface complexity of the software, various parameters that affects the complexity of the software and its trade-off relationship with the deliverability along with the usability parameters.

There are many researches and work done that shows that software's with very high complexity are very low in deliverability and hence are not popular in the market. That is, as the complexity of the software raises, the deliverability drops significantly, but in this research work, this trade-off relationship between the complexity and deliverability has been established with the concept of evolutionary multi-objective optimization which is novelty of this work and further there is framework developed using the fuzzy rule based systems (FRBS) with the help six identified parameters of software usability.

The framework has been verified using the hypothesis testing and a mathematical model has been developed for the support of the framework using rough set theory.

Step 1- Study of various software complexity issues, software deliverability issues and usability aspects of software.

In this step, the analysis has been done of the basic concepts of software complexity and by referring the various literatures available by the various researchers as well as big companies; the identification has been done about the fundamental criteria to analyze the complexity and deliverability of any software from user point of view. There are various models and schemes developed for checking the software quality improvement in terms of Flow of Data, Mean Time to Repair (MTTR) in addition to the Mean Time between Failure (MTBF), but there exists a huge gap in terms of usability of the software.

Usability of the software refers to the ease of use in driving the desired result. The ISO/DIS defined the term usability as "Degree to which a software package can be utilized by a specific user to attain specific objectives with maximum efficiency, satisfaction as well as effectiveness in a precise usage circumstance".

As functionality of the software increases the deliverability value to the client also increases but at the same time, complexity also increases. High software complexity leads to various issues such as lack of Adoption, more end-user training, more software Technical support, less likability of software, low user's performance and lack of customer satisfaction etc.

Complexity analysis involves breaking down a user task into a set of constituent steps and then calculating a complexity metric for each step in the task relative to the type of user.

The software deliverability can be considered as the degree of the usability factor provided to the user of the system by the software. The software deliverability should be high in order to attain maximum value from the software. The business value of any software is highly affected by the software deliverability factor which later imposes several constraints on the software developers.

The term software usability is actually the level of comfort or the ease with which a user can work on the software. As the discussion has been done earlier that the software complexity varies from person to person and from software to software in a well defined and constrained scenario, the software usability also varies accordingly. Higher the complexity, lower will be the usability aspect of that particular software product.

Step 2- Study of Evolutionary multi-objective optimization to establish the trade-off between software complexity and its deliverability.

In this step, various research papers and literature that focuses on the EMO has been analyzed and studied thoroughly in order to achieve the objective of this research work. One cannot identify a single point of solutions to optimize each objective simultaneously.

The job of handling multiple objective problems is known as multi-objective optimization. The trade-off surface's convexity is based on the fact that in what manner the objectives are scaled. As a result, the look for the best or an optimal solution is discarded from the observation in the case of one objective problem. In general, simplifying the multi-objective problems can be seen by decision making as well as searching. The primary step towards solving a multi- objective problem is the Pareto Optimality.

Step 3- Establishment of trade-off between software complexity and deliverability using EMO.

Here, there are two conflicting situations: software complexity and deliverability. In this particular case there is a need to increase up to maximum the deliverability and software's usability and minimize the software's complexity. A set of software and two set of questionnaires [Annexure 2, 3] has been developed for private and government financial institutions with varying range of complexity level. The software's complexity level has been performed by the process of complexity analysis. This particular software application has been used by three individual banks in the city named Lucknow.

The data has been collected from the employees after filling the questionnaires over a period of time which is basically the ratings the experiences that they faced during working on that software and then this data has been feed or entered into the JAVA based open tool known as 'GUAJE' which works on the basics of EMO and the results that are generated are promising. After analysis the results obtained from the tool it is concluded that software with higher usability factor or lower complexity level are much popular among the user which results into the higher acceptability of that particular software.

Step 4- Development of a framework using FRBS for the quantification of software complexity and its usability.

In this step, a framework has been developed using fuzzy rule base systems. The development of FRBS framework starts with Fuzzy Inference System (FIS). The input to FIS may be fuzzy or crisp but the output from FIS is always a fuzzy set as discussed in 3-Block Diagrams of expert systems. The basic step in FIS is to convert the crisp set into fuzzy input. This input is fed to the Rule Base which consists of Knowledge Extractor to generate the complex output set. Later on it is defuzzified to produce the crisp output of a particular event. Since, the calculation of the trade-off value for a software module is needed; the process begins with the basic building block of categorizing the clusters of software module.

Three rules have been identified that are applied to the framework. Also, there are six parameters of software usability that has been identified in the previous steps are used as inputs and the result has been taken in the form of software popularity.

This framework basically quantifies the value of software complexity, usability and popularity. The rule description of software complexity and usability has been done using Mamdani FIS. Based on the linguistic set, the output is classified into three broad categories: High, Low and Moderate ranging from 0, 1 and 2.

Step 5- Verification of the framework using hypothesis testing and the mathematical model in support of the model using rough set theory.

In this step, hypothesis testing has been used for the verification of the framework that has been developed. Also, there has been a mathematical model for the support of the framework using the rough set theory. The hypothesis testing is a very prominently used method of verification that is used in statistics.

In statistical hypothesis testing, a statistical inference is carried out based on the data that has been gathered from a research or survey carried out. If the occurrence of the result is predicted as unlikely according to the pre-calculated threshold probability also referred to the significance level, then the result is called as statistically significant in statistics. Ronald Fisher was the person who initiated the concept of "test of significance". The tests of significance are used to determine that which outcomes of a research will direct to a denial for a pre-specified significance level of the null hypothesis. This provides contribution in deciding whether the results contain sufficient information or not in order to cast disbelief on predictable insight, to establish the null hypothesis, considering the fact that the usual perception has been applied.

In order to analyze the relationship between the complexity of the software based on the various parameters like context shift, navigational guidance, input parameters and system feedback, The concept that is used here is rough set theory. Since two decades this approach is frequently used in the analyzing the relationship between various parameters.

Rough set methodology uses the concept of decision table consisting universe of discourse showing a relationship represented by two types of attributes: condition attribute and decision attribute. Basically it gives a concept of relationship between attributes called lower and upper approximation.

Step 6- Conclusion and future scope.

Complexity of the software varies from person to person. In an observation, it has been analyzed that when the software complexity increases, the usability aspect of the software drops significantly. The novelty of the current work is use of soft computing techniques in analyzing the trade-off between the complexity and the deliverability of the software. This research work has also involves the development of a framework using Fuzzy Rule based System (FRBS) for quantification of software complexity and usability aspects. In the final phase, the development of Fuzzy Inference System (FIS) by means of Expert Knowledge Base was done. In future, the work can be extended by enhancing the framework by identifying more usability aspects of the software and the application of the framework can be possible in various types of software to analyze the complexity aspect of the software to determine its future market potential.

List of Figures

List of Tables

List of Publications

1. Lavania, S., Darbari, M., Ahuja, N. J., & Shukla, P. K. (2012). The Application of Evolutionary Algorithm in Managing the Trade-Off between the Complexity of Software and Its Deliverables. *International Review on Computers & Software*, *7*(6), 2899-2903.

2. Lavania, S., Darbari, M., Ahuja, N. J., & Siddqui, I. A. (2014, February). Application of computational intelligence in measuring the elasticity between software complexity and deliverability. In *Advance Computing Conference (IACC), 2014 IEEE International* (pp. 1415-1418). IEEE.

3. Lavania, S., Darbari, M., Ahuja, N. J., & Siddqui, I. A. (2015). Genetic Algorithms-Fuzzy Based Trade-Off Adjustment between Software Complexity and Deliverability. In *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering* (pp. 15-18). Springer International Publishing.

CHAPTER 1
INTRODUCTION

1.1 PREAMBLE

This thesis discuss, in general terms, the application of soft computing algorithms (EMO) for designing a framework. Objectives of the thesis can be summarized as follows:

- Identification of Software Usability aspects
- Identification of software complexity metrics
- Establishing a trade-off between software complexity and its deliverability (usability aspects) by Evolutionary Multi-Objective Optimization (EMO).
- Development of a Framework by using Fuzzy Rule based System (FRBS) for quantification of software complexity and usability.

The work described herein not only concentrates on the software complexity part but it also contributes in the establishment of usability aspects of software. The present work proposes a framework to conquer the existing problems of establishing the trade-off relationship among software complexity and software deliverability. The motive of this thesis is to establish the trade-off using soft computing algorithms which is unique.

1.2 NEED FOR THE PRESENT WORK

Ever since the software development came into existence, there has been a rift between the software developers and users. Software developers in order to ease their development cycle, try to embed multiple features in a single module making the usability of the module tougher.

It is the usability feature which plays a major role in having the product more sellable, but at the same time it should cater to all high level needs of the consumer. Although, there are number of companies developing certain guidelines about software development process, the major focus is on user-centred application development, software is evaluated with various tools like cognitive tools and complexity Matrices to find out the degree of acceptability amongst the users.

The deliverability aspect of the software in most of the cases is contextual i.e. Analytical or Empirical Methods. The Analytical method depends upon potential interaction with the system and finding out the flaw in the system. Secondly, Empirical evaluation method which is based on the actual usage data.

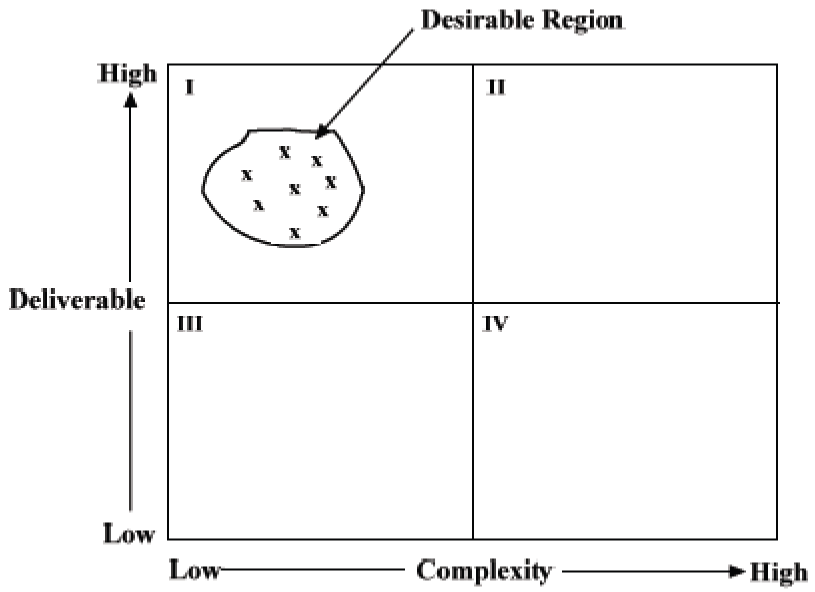


Figure 1.1: Deliverability v/s complexity

The main problem arises to maintain a balance between complexity and deliverability, as both the quantities are very much inter-related, it is very difficult to raise the deliverability without increasing the complexity of the system. The basic common criteria for deliverability are:

(i) Ease of use
(ii) Task Support
(iii) Navigation
(iv) Help
(v) Scalability with disturbing the ease of use.

There are various models and schemes developed for checking basic software quality improvement in terms of Flow of Data, Mean Time to Repair (MTTR) in addition to the Mean Time between Failure (MTBF), but there exists a huge gap in terms of usability of the software. Usability of the software refers to the ease of use in driving the desired result. The ISO/DIS defined the term usability as "Degree to which a software package can be utilized by a specific user to attain specific objectives with maximum efficiency, satisfaction as well as effectiveness in a precise usage circumstance". Relation between complexity and usability has an inverse relationship (Figure 1.2). As complexity raises the usability aspect of deliverable in terms of "Ease of Use" goes down.

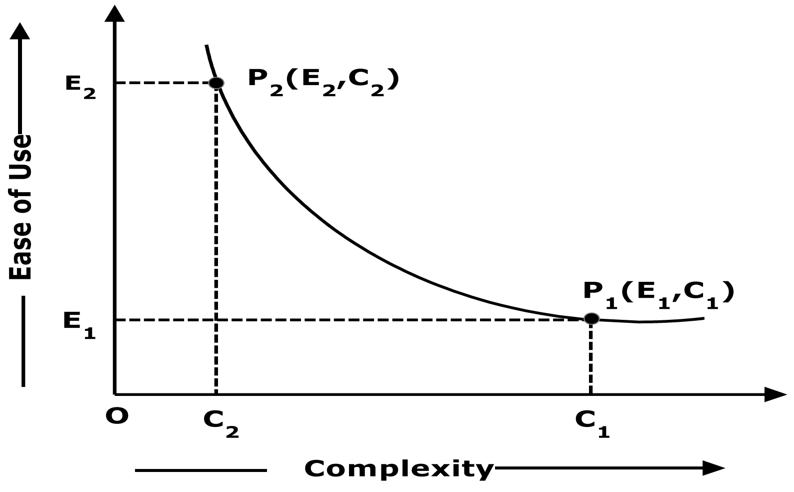


Figure 1.2: Plot between complexity and Ease of Use

As functionality of the software increases the deliverability value to the client also increases but at the same time, complexity also increases. There are chances that the user may not use some of the required functionality. In order to proceed with the problem, authors have used the concept of Expert System in calculating the elasticity between the two variables complexity and usability.

In general, complexity disturbs the ecological aspect of the messages in module. A complex information module can be represented in three dimensional formats as suggested in Albers [4, 6]: Knowledge Level, Detail Level and Cognitive Abilities. In order to increase all the three levels there will be a compromise with the Usability Aspect form HCI preview.

The deliverable aspects nowadays focuses on Human Centered application where customer's involvement plays a major role in design phase, but the customer always tells the requirement in the form of stories which looks much simpler during requirement gathering stage but when implemented on real scale. The complexity of the software increases considerably.

1.3 OUTCOMES OF HIGH SOFTWARE COMPLEXITY

High complexity in the software leads to various negative results. Following are the outcomes of high complexity in the software:

- Lack of Adoption
- More end-user training
- More software Technical support
- Less likability of Software
- Low user’s performance
- Lack of customer satisfaction

It is broadly known that enterprise resource planning (ERP) software systems put up with very complex user interfaces. The software complexity of these user interfaces negatively affects the usability aspect of these software systems.

Present study has revealed that a need exists to advance the overall usability of the ERP software systems. The Specific approach as well as the criterion for evaluating the usability aspect of ERP software products have not been developed or broadly published. This work proposes a set of heuristics that can be used to measure the usability of ERP systems and similar kinds of software systems.

This work gives the description about the complexity analysis; a quantitative approach to the software usability engineering which has been effectively used in a number of real-world software projects. The complexity analysis depends on finding and quantifying impediments that get in the way of easily learning and using software. The impediments – such as confusing user interfaces, long sequences of manual steps and cryptic error messages – are quantified by the measures named as “complexity metrics”.

The complexity metrics gives the easily-understood comparisons of usability between the steps in a task, overall tasks, releases, and the products. They are developed through thorough, exhaustive rating scales related with the following six aspects of software usability: navigational guidance, context shifts, error feedback input parameters, new concepts and system feedback. Even though the complexity analysis is a lighter-weight usability evaluation method as compared to the usability testing, the empirical results show that the complexity metrics are powerfully related to the usability testing time-on task measures.

The association between the complexity of software and its usability is contradictory in nature and hence can be represented by the following equation:

Maximize (deliverability, usability)

Or

Minimize (un-deliverability, complexity)

Equation 1

1.4 SCOPE OF THE WORK

To facilitate the trade-off among the software complexity and deliverability, the concept of Evolutionary Fuzzy Rule Generation using Messy Genetic Algorithm has been used. The multi-objective evolutionary algorithm is well known technique in finding out optimum solution in case of multiple goals. The problem which is single objective optimization in nature could have a single optimal solution while multi-objective generates multiple solutions produces the vectors representing the value of trade-off.

The work presents the issue of establishing the trade-off between the software complexity and its deliverability aspect. Based on the management of trade-off [8]-[10], the Popularity index of the software is determined.

1.5 OBJECTIVES

There are two objectives of the present work which are as follows:

1. Establishing a trade-off between software complexity and its deliverability (usability aspects) by Evolutionary Multi-Objective Optimization.
2. Development of a Framework by using Fuzzy Rule based System (FRBS) for the quantification of software complexity and its usability.

1.6 RESEARCH METHODOLOGY

The steps followed are mentioned below:

1. The first step covers the establishment of software complexity metrics to evaluate the complexity of any software or its particular application.

2. The second step involves the identification of software usability aspects to evaluate the deliverability of that particular software.

3. The third step involves the establishment of trade-off between the software complexity and deliverability using evolutionary multi-objective optimization (EMO).

4. The fourth step involves the development of framework using Fuzzy Rule Based System (FRBS) for the quantification of software complexity and its usability.

5. The last step deals with bringing the thesis to a Conclusion by justifying the work which is done till date.

1.7 ORGANISATION OF THE THESIS

Chapter 2 gives a brief introduction to evolutionary multi-objective optimization. It also highlights the various characteristics and components soft computing algorithms. Lastly, it discusses about software complexity metrics, its evaluation criteria and six aspects of software usability.

Chapter 3 presents literature review of the soft computing algorithms. It highlights the work by various researchers and companies in the form of their white papers to discuss the basic ideology of software deliverability criteria’s and usability index over the last few years.

Chapter 4 involves the completion of the first objective that is the establishment of trade-off between software complexity and deliverability with the help of the concept of EMO.

Chapter 5 involves the development of framework using Fuzzy Rule Based System (FRBS) for the quantification of software complexity and usability.

Chapter 6 presents the verification and validation of the developed framework using the concept of Hypothesis Testing and the mathematical model in support of the framework using rough set theory.

Chapter 7 draws the previous chapters to a conclusion and indicates how the deliverability of the software significantly drops with the increase in the complexity along with the development of framework for potential future developments.

CHAPTER 2

LITERATURE SURVEY

The literature available on software complexity, usability aspects, software deliverability issues, soft computing and evolutionary multi-objective optimization has been rigorously viewed and presented in this chapter.

The paper [1] analyse what implications of software usability has for development phase, giving particular concentration to the impact of the software quality attribute.

The Usability-Supporting Architectural Patterns (USAPs) helps to bridge the gap between software engineers and User Interface designers to develop the software architecture solution that fruitfully fulfils the usability requirements is discussed in [3]. Besides that the formulation of an idea on usability engineering, the concept on FRBS helps in deriving the trade-off relationship between complexity and usability.

The usability of open source software is often regarded as one reason for the restricted distribution. There is a review of the active facts of the software usability of open source software and examine how the features of open source development manipulate the usability of the software in [3].

The influence of ease in the development of software application is very well explained in [9]. This paper also highlights some points on the earlier software development process.

The constraints of interoperability in the field of fuzzy systems modelling using the concept of granularity of information are discussed in detail in [13]. This also describes the extension of the model in developing enterprise level applications.

The major focus on various factors on which usability of software is dependent, is provided in the work [14]. This provides the summary of various complexity models of software quality.

Some of the impediments have been used. The paper by Dia, Y. et al [37] also discusses some prominent issues on quantifying the complexity of IT service and management processes by highlighting some issues relating to the complexity of services quantification and its impact on user psychology. The Complexity issues relating to the usability engineering is very well discussed in [38]. This paper highlighted the quantitative approach to software usability engineering.

For the adaptation of the fuzzy rule systems through an on-line clustering are given in [44] that gives the basic idea about the fuzzy rule based systems and its adaptation on real life case scenarios. Another paper highlights the comparison of existing frameworks and others developed over a decade [45].

Another important contribution is there by the work of Hoffmann, F. et al that focuses on the study of classification rules and their applications using evolutionary algorithms [47].

The various applications in the framework of imbalanced data-sets that focus on the classification systems is given in [49]. This paper highlights expert systems and its applications that are being used to give the real time case study on the analysis of stock market using Neuro based fuzzy inference system [50].

A paper by Cordon, O. et al gives a detailed description and theoretical knowledge on genetic fuzzy systems [52]. The detailed description of design of evolutionary multi-objective systems using fuzzy systems based on rule based criteria is discussed in [53].

The detail description of the application of fuzzy rules and interpolative system reasoning for applications is [53]. The highlights on the kernel based granulation using fuzzy rule based system are given in [55].

The paper [56] highlights the issues of usability issues of ERP systems the common usability criteria's (Navigation, Learnability, Task Support, Customization and Presentation). The basic idea about the role of usability experts in finding the usability aspects and a real life case study on the two processes of ERP (SAP business one) of adding a customer in process a sales order is given.

The parameters of testing as well as quantifying the ERP Usability are described in [57]. It also gives the detailed idea about the research involves testing users as they worked with PeopleSoft on various usability aspects. Also, this paper enhances the usability criteria's on GOMS-KLM are found out as Usefulness, Ease of use, Acceptance and Satisfaction. A user based study was performed on studying the effects of the cellular phones and their prototypes as well as the task complexities on the usability and focuses on a case study based report [58].

The research work is highly motivated by the paper [59] which describes the promise as well as the performance measures of enterprise systems in higher education. It focuses on Ranking of potential obstacles and identifies the potential obstacles of system performance. Another study reveals the quality parameters of software and its metrics for the software quality evaluation [39, 60]. This gives the software quality parameters like Capability, Usability, Performance, Reliability, Installabilty, Maintainability, Durability, Availability, Structuredness and Efficiency.

The paper [61] has identifies the CFF's (Critical Failure Factors) of ERP and gives the EFA (Exploratory Factor Analysis) on three results (Total Failures, Partial Failure and Success). A quantitative approach to Usability Engineering highlights the issues of complexity analysis [62]. It also describes the latest version of the complexity analysis from the user's point of view. Also, this paper gives the basic idea the complexity analysis that involves breaking down a user task into a set of constituent steps and then calculating a complexity metric for each step in the task relevant to the type of user. This paper also highlights the complexity analysis as well as describes the latest version of the complexity analysis.

On the usability side, the white paper by Microsoft deals with the usability engineering aspect of software design by highlighting a framework for checking the usability component of any software [63].

The philosophy of user centered design with respect to the usability in software design is given in [64]. The examination of some decisive causes of the software complexity and its impact on user experience and in order to push the product in market, software firms are focusing on usability.

CHAPTER 3

OVERVIEW OF EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION, SOFTWARE COMPLEXITY AND SOFTWARE USABILITY

3.1 MULTI-OBJECTIVE OPTIMIZATION

The process of concurrently optimizing more than one objective contradictory with each other subject to some constraints is called multi-objective optimization. There is a need to take a decision that is optimal in the company of trade-offs among more than one conflicting goals. One cannot identify a single point of solutions to optimize each objective simultaneously [22].

Optimization is a mathematical discipline that concerns to find out a substitute solution with most cost effective and maximum achievable performance under the limitations by maximizing the required factors and lowering the not required factors [33]. The meaning of maximization is to trying to achieve the uppermost or maximum outcome or result will generated with no regard to expense or cost. These days, optimization comprises a broad diversity of techniques commencing artificial intelligence and is utilized to perk up the business processes in almost all the industries.

3.2 EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION (EMO)

The term MO is, without loss of generality, there is no single solution and it should be best when measured on all objectives/ solution [14-19]. It minimizes n number of the components f_k, where if there exists no, perfect, feasible, unique answer however a minimization crisis, dominance can be given as:-

$iff!\ u, v: \ , !\ \gamma: (0, 1), \#\ w : P : w \leq \square u + (1 - \square)v ,$ *Equation 1*

where the disparity applies component wise or else, P will be non-convex. Likewise, for concavity:

The definition of Concavity can be given as: a *well defined non-dominated set P will be concave iff*

$!\ u, v : P , !\ \gamma : (0, 1) , \#\ w : P :: \square u + (1 - \square)v\ p< w .$ *Equation 4*

The surfaces attained after establishing the trade-off might not be concave or convex. Therefore, the regions of local concavity as well as local convexity can generally be recognized in such trade-off surfaces.

The trade-off surface's convexity is based on the fact that in what manner the objectives are scaled. As a result, the look for the best or an optimal solution is discarded from what comes as an observation in the case of one objective problem. Generally there is a requirement of decision making as well as searching. To facilitate the decision building and searching, there are four different approaches are acknowledged.

3.2.1 POSTERIOR ARTICULATION OF PREFERENCES

In order to discover every promising answer of the non conquered set, utilization of the user first choice can be done to decide the largely suitable known as decision making following the search. A large number of techniques are available that enable to discover the solution space [23]. The great reward with these kinds of methods is that the obtained solution is self-determining of the decision maker’s preferences. The study has only to be execute ones, since the Pareto set would not vary providing the problem description are unaffected. Though, a few of these methods bear from a huge computational load.

3.2.2 PROGRESSIVE ARTICULATION OF PREFERENCES

At each step in this preference, a partial preference piece of the important information is delivered by the authorized decision maker to the start the process of optimization. The process of optimization and decision making happen at interleaved steps, thus it inward information and produces improved alternatives. Also, the decision-maker is not constantly giving input throughout the operation of the algorithm.

3.3 NO PREFERENCE

This approach is useful in solving a problem as well as provides a solution to the Decision Maker. This method doesn’t consider any preference information. This Min-Max formulation has the fundamental of minimization of the calculated relative distance that is starting from a given candidate solution up to the desired utopian solution. Also, there is no preference related information from the decision- maker is essential. Though, the outcome is merely a single point on the obtained Pareto front, which the Decision Maker must have to recognize as the last and final generated solution.

Through the provision of the single objectives diverse weightings and altering the exponent in the distance formulation, dissimilar points on the obtained Pareto front might be established [24]-[26]. Though, then the preference related information provided from the decision-maker afterwards is required. In this scenario, this kind of formulation isn’t generally considered in industrial design. Yet, this Min-Max formation could be considered jointly along with the other similar methods in order to uncover various points on the desired Pareto front for the interactive methods.

3.4 CONSTRAINTS OF OPTIMIZATION

The most suitable solution to a given real world problem might be controlled by a series of real world limitations forced of the desired decision variable. There are following two categories of constraints:

3.4.1 DOMAIN CONSTRAINTS

The first category is the Domain Constraints, which states the domain of the description of required objective function. Considering the case of control systems, the closed loop system steadiness will be specified as a case of a mentioned domain constraint, since the major and significant performance measures are not very clear for the unstable systems.

3.4.2 PREFERENCE CONSTRAINTS

This imposes additional limits on the solution of the problem. The known stability margin, in case, represents a (subjective) liking of the decision maker.

It is completely supposed that there is no less than a single point in the given U that fulfils each and every constraint, even though in the preparation that cannot for all time be certain. While the defined constraints may not be all concurrently fulfilled, the major issue is regularly seemed to recognize not a single solution as it occurs. In order to probably slow down the preference constraints, there is a range of constraints violated along with the level to which every defined constraint is desecrated are then engaged into consideration.

The Constraints may be considered as objectives of maximum priority, which have to be mutually satisfied prior to the optimization of the left over. Fulfilling several violated inequality constraints are obviously the multi-objective problem of minimizing the related functions awaiting the provided values (aim) is achieved. The idea of non inferiority is as a result, willingly appropriate and even predominantly suitable while constraints are themselves non commensurable.
In contrast, the problems categorized by well defined soft – objectives simply, are regularly reorganized as the constrained optimization problems (single objective) so that it can be resolved.

3.5 VISUALIZATION AND PERFORMANCE ASSESSMENT ISSUES

This is related with the graphical image of the trade-off data, that is, in case, if there are single or multiple sets.

The major objective that is present here is to correspond to the individual decision maker, the valuable information regarding to the most excellent established trade-off on the whole non-dominated solutions. In contrast, for achieving within reach into how fine an optimizer can be likely to execute upon any specified crisis, the extracted data by several long and tedious optimization runs have to be measured in its totality, which means this is wrong to believe that only the whole non-dominated solutions will be taken into consideration. As a result, a visualization process is being considered; the process of Visualization of the trade-off data commencing single run methods for posterior as well as progressive articulation of preferences that call for the trade-off information be expressed in front of the human decision maker in a shape that should be without difficulty understand [27].

Therefore, in order to attain the smooth image of the Pareto-set, the interpolating between the data is not usually right, primary as here there is usually zero assurance that it would truly be even, as well as next as actual generated solutions related to those in-between objective vectors, yet if they be present, but still not acknowledged.

Maximum, one possibly will wish to draft an edge unravelling those important points in objective space that are equals or dominates. This kind of edge can as well be viewed are identified to be achievable, providing the data is given [28].

While evaluating an optimizer, the individual is generally worried for the worth value of the resultant solutions that are capable to fabricate, as well as with the sum of calculation attempt it needed. Additionally, evaluation of how probable a single run is to generate high-quality outputs is also important.
If there is only one objective, the value of the end result of an optimization run is calculated straight by the objective value related with the finest solution established. Given that this is a single, scalar value, the division of the quality of end results generated by numerous runs could be effortlessly seen by the scatter plots, histograms and empirical distribution functions etc.

In case, if there exists multiple that is more than one objective, a dissimilar illustration is essential. A general method, also known as the approach of parallel coordinates, including the integrating an integer index i to every individual objective and then representing every non-dominated point with the help of a line linking the objective points. Along with this illustration, rival objectives having successive indices outcome in the crossing of lines, while non- concurrent lines point out non-rival objectives. Even though the sequencing of the objectives might be routinely resolute upon on the foundation of a number of measure of struggle (so that struggle will be at its maximum level among neighbouring objectives), being capable to vary this sequencing interactively is valuable as well, and not hard to execute.

In case, if there are many objectives, though, runs will usually generate changeable number of approximations. The importance of the trade-off description generated by each run depends on two factors. The first is the non-dominated points established and on how well they cover it up [30]. Merely the suggestions of how fine the individual objective points establish in every run inclined to be, however the valuable information on matter that how they inclined to be dispersed alongside the trade-off surface is gone. There are following two factors:

- Population based search techniques simulating the behaviour of natural evolution.
- Dealing with complex search spaces having robust and powerful search mechanism.

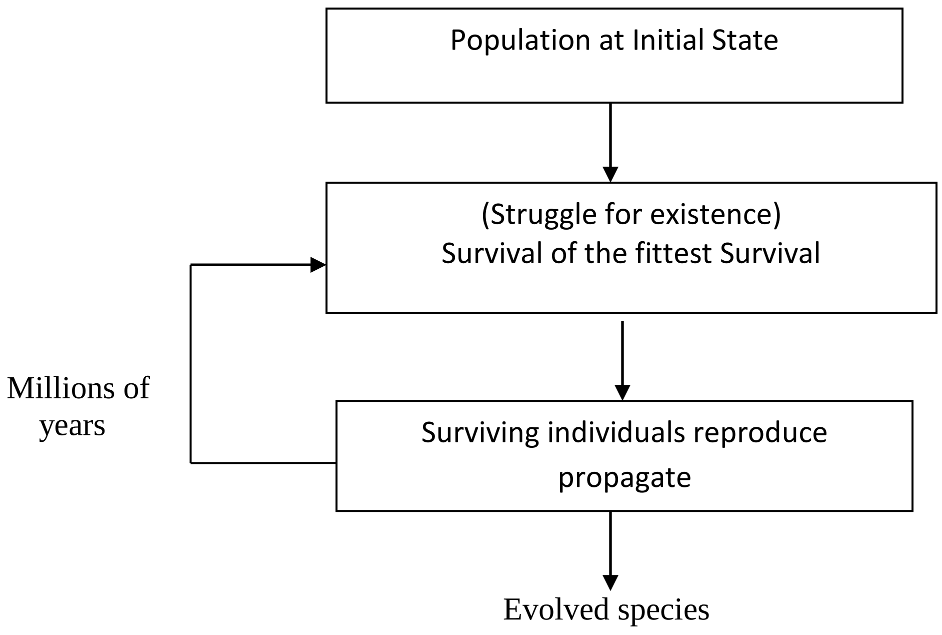

Population at Initial State
(Struggle for existence)
Survival of the fittest Survival
Millions of years
Surviving individuals reproduce propagate
Evolved species

Figure 3.1 Natural Evolution

3.6 SOFTWARE COMPLEXITY

Complexity issue related to "Ease of Use" in the line with the end user. A high skilled user may find some task very easy than low skilled (related to computer related knowledge) user [6]. Complexity of the system varies from mere installation of the software to complexity verifying online in real time mode. It provides a relative comparison between the levels of difficulty in software products. For example, in Indian scenario, there are number of consumers who have high end systems but lacks of usage of all the software systems. It is divided into three parts Personal, Social and Technological [7]-[11]. The Personal factors identifies the traits of user depending upon his/her knowledge skills, user is able to perceive things and develops adaptability accordingly. When design a software module for user interface personal traits are given very high priority. Social factor deals with peer environment and their skill set.

3.6.1 COMPLEXITY ANALYSIS

It is a new approach to evaluating the usability of software that combines many of the advantages of both usability testing and usability inspection methods. Complexity analysis provides metrics that quantify usability and that are highly correlated to results gathered through usability testing [37]. In addition, complexity analysis shares the lightweight characteristics of usability inspection methods, enabling teams that develops software to frequently evaluate software usability throughout the development process.

Complexity analysis involves breaking down a user task into a set of constituent steps and then calculating a complexity metric for each step in the task relative to the type of user [64]. For example, Table 1 shows the complexity metric for each step of a fictitious installation task for the user role “social-networking parent”. The nine steps of the installation task in Table 1 are listed sequentially in the left-hand column, and their corresponding complexity metrics are in the right-hand column. The complexity metric for a step is a measure of how difficult it is for the targeted user to complete that step. You can think of the complexity metric as an inverse measure of usability – the higher the complexity metric, the lower the usability for that step. Therefore, “lower is better” when it comes to complexity metrics.

The overall complexity metric for a task is the sum of the complexity metrics for its constituent steps. Therefore, the complexity metric for the installation task depicted in Table 1 is 79.

The steps of an Installation Task with its complexity metric are shown in the table 3.1.

Table 3.1: Complexity metrics for the steps of a software installation task

S. No.	**Step**	**Complexity Measure**
1	Turn off firewall	15
2	Execute setup file	11
3	Select license option	3
4	Select type of installation	3
5	Specify installation directory	5
6	Confirm user name	11
7	Install program files	11
8	Specify preferences	3
9	Turn on firewall	17
	Total	**79**

3.7 SOFTWARE DELIVERABILITY (USABILITY ASPECT)

The software deliverability can be considered as the degree of the usability factor provided to the user of the system by the software. The software deliverability should be high in order to attain maximum value from the software. The business value of any software is highly affected by the software deliverability factor which later imposes several constraints on the software developers.

The term software usability is actually the level of comfort or the ease with which a user can work on the software. As discussed earlier that the software complexity varies from person to person and from software to software in a well defined and constrained scenario, the software usability also varies accordingly. Higher the complexity, lower will be the usability aspect of that particular software product.

Higher usability factor enabled software has higher market space capturing probability as comparison to those software products which has low software usability value. This directly affects the market status of the software as well as the company.

The usability factors of software can be categorized as:

- Efficiency:
- Learnability
- Satisfaction
- Memorability:
- Errors

The next factor Memorability can be defined as when users go back to the design following a period of not using it, how effortlessly can they re-establish the expertise? And the last factor satisfaction can be explained as how pleasing is it to utilize the design?

Six Basic Factors of Software Usability:-

1. Context Shifts
2. Navigational Guidance
3. Input parameters
4. System feedback

5. Error feedback
6. New concepts

1**.** ***Context shift*** occurs when the user crosses user interface, tool or product boundaries in order to perform a step.

*2**.*** ***Navigational guidance*** refers to the support provided to a user for proceedings into a step (from the previous step) and through the step.

3. ***Input parameters*** are data supplied by the user to complete the step

4. ***System feedback*** is the system response to the user actions for a given step. Examples of system feedback include progress indication dialog boxes, confirmation of command execution and system generated reports.

5. ***Error feedback*** is the system response to common error situations the user may encounter.

6. ***New Concepts*** refers to background information on a specific topic that the user needs to understand in order to perform a step, and that the user has encountered for the first time in the context of their current task(s).

CHAPTER 4

THE ESTABLISHMENT OF THE TRADE- OFF BETWEEN THE COMPLEXITY OF THE SOFTWARE AND ITS DELIVERABILITY BY THE EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

The situations that are conflicting in nature with each other in terms of their goals can be very well represented by the evolutionary multi-objective optimization. The solutions that are generated are called Pareto-optimal solutions. Here, there are have two conflicting situations: software complexity and deliverability.

In this particular case there is a need for maximizing the deliverability aspect of the software as well as the Simplicity factor of the software which is represented by $g(x) = (d(x), s(x))$.The multi-objective formulation of complexity and usability can be given as:

$$Maximize \{f_3(d), f_4(u)\}$$

$$Minimize \{f_1(c), f_2(p)\}$$

Equation 8

Where f_1 (c) is the complexity function of the software.

f_2 (p) is the poor usability function of the software.

f_3 (d) is the deliverability function of the software.

f_4 (u) is the usability function of the software.

4.1 COMPLEXITY AND DELIVERABILITY ANALYSIS USING THE REAL DATA [ANNEXURE A]

Table 4.1: A snapshot of data generated after applying complexity and deliverability metrics

Bank – A		Bank – B		Bank – C	
CS-1	DLS-1	CS-2	DLS-2	CS-3	DLS-3
31	81	61	39	85	28
34	76	71	36	92	25
43	71	76	31	95	20
44	63	81	28	101	21
41	61	91	24	106	19

Where,

CS – Complexity of Software

DLS – Deliverability of Software

A set of software and two set of questionnaires [Annexure 2, 3] has been developed for private and government financial institutions with varying range of complexity level. The software's complexity level has been performed by the process of complexity analysis. This particular software application has been used by three individual banks in the city named Lucknow.

The data has been collected from the employees after filling the questionnaires over a period of time which is basically the ratings the experiences that they faced during working on that software and then this data has been feed or entered into the JAVA based open tool known as 'GUAJE' which works on the basics of EMO and the results that are generated are promising.

After analysis the results obtained from the tool it is concluded that software with higher usability factor or lower complexity level are much popular among the user which results into the higher acceptability of that particular software.

4.2 RESULTS OBTAINED FROM EMO FRAMEWORK (GUAJE)

This tool GUAJE implements the methodology of fuzzy modelling called Highly Interpretable Linguistic Knowledge (HILK) that is focused on yielding a high-quality interpretability-accuracy trade-off thanks to combining expert and induced knowledge in a common framework.

This tool contains a computational environment for creating interpretable as well as accurate fuzzy systems through integrating several pre-existing open source tools, taking profit from the major advantages of each individual tool by analogy with the major idea primary to Soft Computing. In reality, it is an upgraded version of the free software known as KBCT (Knowledge Base Configuration Tool).

On applying the data on this tool, following results have been carried out.

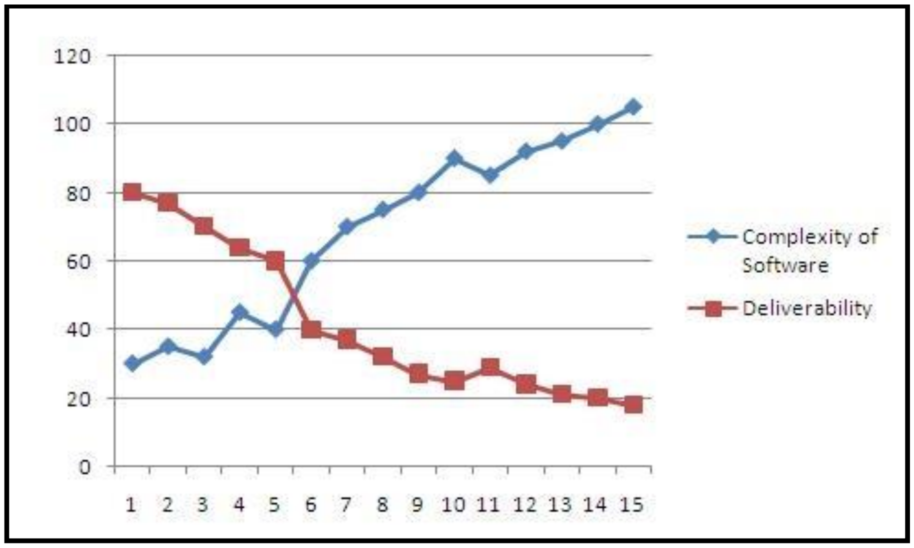


Figure 4.1: Complexity v/s deliverability

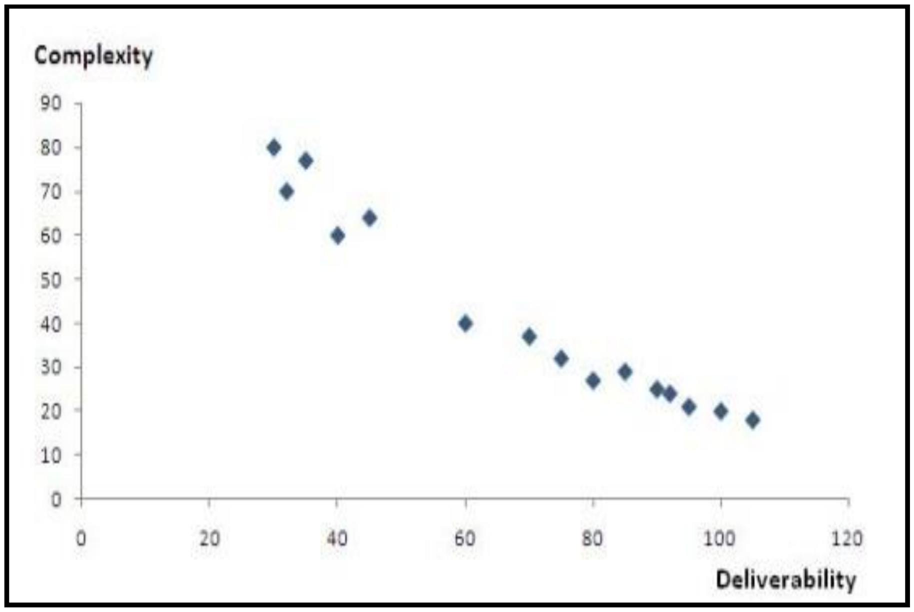


Figure 4.2: The Pareto Front (deliverability and complexity)

4.3 CONCLUSION

The plot between complexity and deliverability shows that deliverability of the software drops significantly with the rise in the complexity of the software. After analysis the results obtained from the tool it is concluded that software with higher usability factor or lower complexity level are much popular among the user which results into the higher acceptability of that particular software.

CHAPTER 5

DEVELOPMENT OF A FRAMEWORK USING FUZZY RULE BASE SYSTEMS (FRBS) FOR THE QUANTIFICATION OF COMPLEXITY OF THE SOFTWARE AND ITS USABILITY

The development of FRBS framework starts with Fuzzy Inference System (FIS). The input to FIS may be fuzzy or crisp but the output from FIS is always a fuzzy set as discussed in 3-Block Diagrams of expert systems. The basic step in FIS is to convert the crisp set into fuzzy input. This input is fed to the Rule Base which consists of Knowledge Extractor to generate the complex output set. Later on it is defuzzified to produce the crisp output of a particular event [35, 36]. Since, there is a need to calculate the trade-off value for a software module. The process starts with the basic building block of categorizing the clusters of software module.

5.1 FUZZY RULE BASED SYSTEMS (FRBS)

An Expert System consists of Knowledge Accumulator, Fuzzy Inference System (FIS) and External Variables (Fine Tuning Variables).

Fuzzy Rules Contains linguistic values [46, 47] which are supported by their intensity using IF-THEN-ELSE condition with other linguistic variable. Fuzzy Rule implication can be two kinds of logic inferences: modus ponens and modus tollens. A simple statement like: "If complexity of software is HIGH, then deliverability is LOW". Since "Unix operating system is complex" according to modus ponens can be infer that "Unix Deliverability is LOW" while according to modus tollens "If Unix operating system is NOT Complex" can be infer that "Unix is Highly Deliverable". FRBS helps in generating a fuzzy model which consists of mapping functionality between set of input variables and set of output variables.

Complex Input problems are simplified in terms of linguistic variables to generate Fuzzy Rule Based. These Rules are matched with GA. These rules iterated and finally refined to eliminate GA generator encodes one complete set of duplicity using Post-Processing stage. Each chromosome generated from Fuzzy Rule.

Each generation was mutated by selecting 20% of the Parent Population, and then these selected individuals were again mutated with a probability of 0.15.

For the above situation, the fitness functions as a component of complexity, deliverability and popularity are obtained.

Finally the Plot as shown in figure 4 which as the popularity of the software (i.e. its deliverability features) increases steadily and plateau at complexity is generated.

$$f = \begin{cases} 1 \text{ ; if Deliverability} < 0 \\ 0.5 \text{ ; if Deliverability} > 0 \\ \dfrac{\text{High} - x}{\text{High-Low}} \text{ ; if Low} \leq x \leq \text{High ; Complexity} > 0\text{,Popularity} > 0 \\ 2 - \dfrac{\text{Complexity}}{\text{Complexity x Popularity}} \text{ ;} \qquad \text{Popularity} > 0 \\ 0 \text{ ; if Complexity} > \text{Deliverability} \end{cases}$$

Equation 9

A general framework of Expert System consists of Knowledge Accumulator Fuzzy Inference System and External Variables (Fine Tuning Variables).

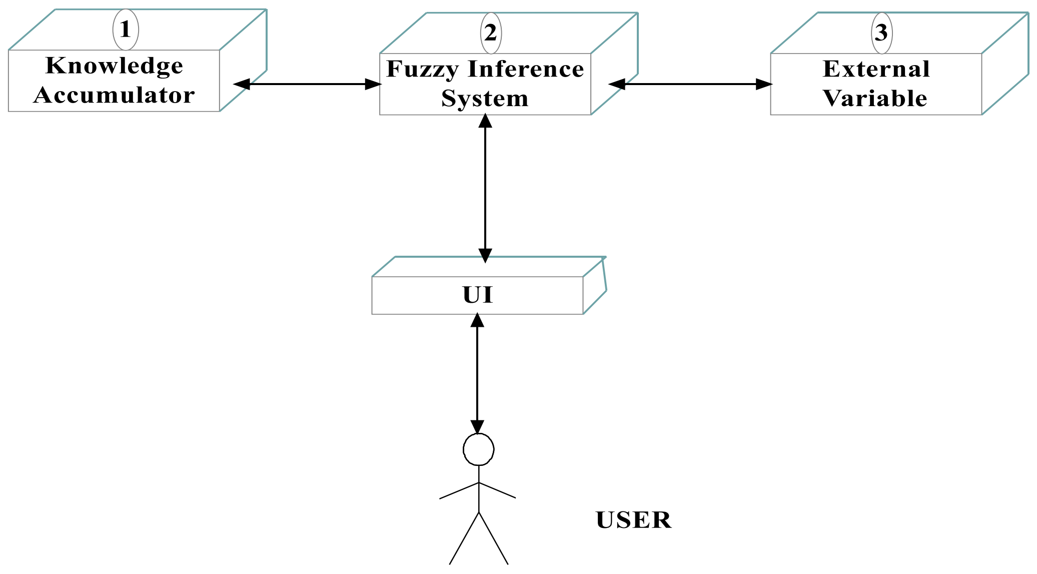


Figure 5.1: Three Block Architecture of Expert System

Knowledge Accumulator gathers knowledge of multiple human experts. This knowledge is fed into the Fuzzy Inference System (FIS) supported by any real time variable. The output is either generated by Sugeno or Mamdani FIS which is converted to the user interface for results and analysis.

5.2 DEVELOPMENT OF THE FRAMEWORK USING FRBS

The development of FRBS framework starts with Fuzzy Inference System (FIS) [51, 52]. The input to FIS may be fuzzy or crisp but the output from FIS is always a fuzzy set as discussed in 3-Block Diagrams of expert systems. The basic step in FIS is to convert the crisp set into fuzzy input. This input is fed to the Rule Base which consists of Knowledge Extractor to generate the complex output set. Later on it is defuzzified to produce the crisp output of a particular event. Since, there is requirement to calculate the trade-off value for a software module. The work begins with the basic building block of categorizing the clusters of software module.

Earlier identified factors on which FRBS will be applied are:

1. Navigational Guidance
2. Context Shifts
3. System feedback
4. Input parameters
5. New concepts
6. Error feedback

These factors will be the Input for the framework and the usability will be in terms of complexity (High, Medium, and Low)

A simple clustering is achieved by using Fuzzy Decision Tree. From this tree there will be the formation of the Rule base (Figure 3).

Now there are three basic categories of rules:

Rule I: If Complexity is Moderate, Usability is also Moderate, Software is Popular.

Rule II: If Complexity is Very High, Usability is Poor, Software is Not Popular.

Rule III: If Complexity is Low, Usability is High, Software is very Popular.

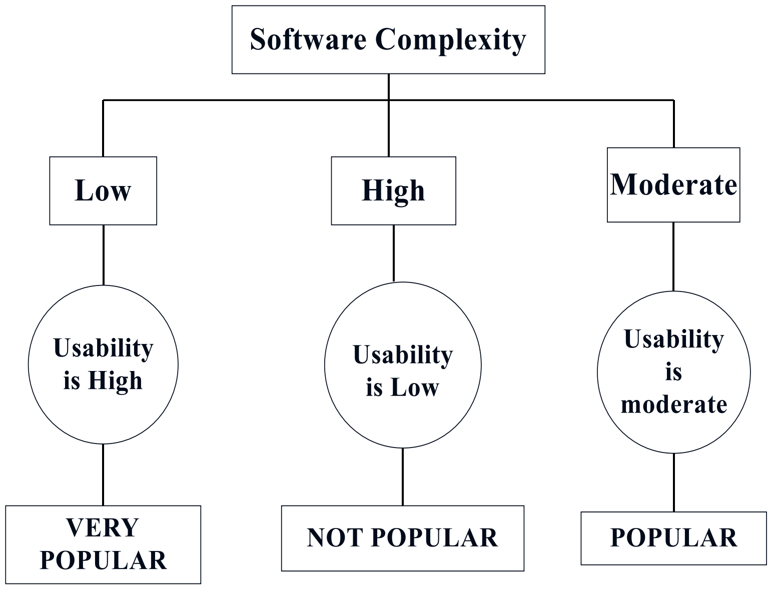


Figure 5.2: Tree hierarchy of Rule-Base

The extension of the basic the above Rule Base of Tree Hierarchy into Fuzzy Inference System in the work is being done.

5.3 QUANTIFICATION OF INPUT VALUES

5.3.1 INPUTS:

1-Complexity (2, 1 and 0 as High, Moderate and Low respectively)

2-Usability (2, 1 and 0 as High, Moderate and Low respectively)

5.3.2 OUTPUT:

Popularity (Ranging from 0 to 10)

The following values has been put into the framework and by applying the three rules that are mentioned above, a framework has been developed which has the input in the form of complexity and usability and the output in the form of popularity.

5.4 FRAMEWORK DEVELOPED USING FRBS

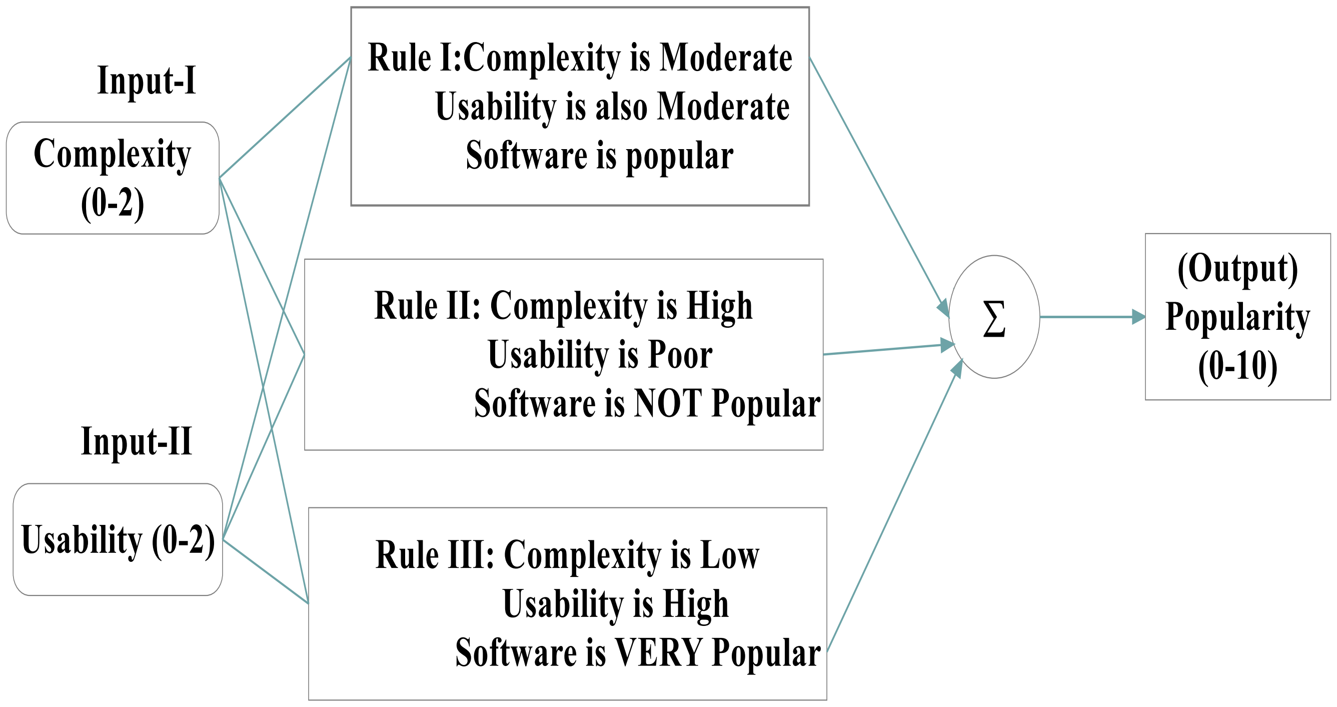
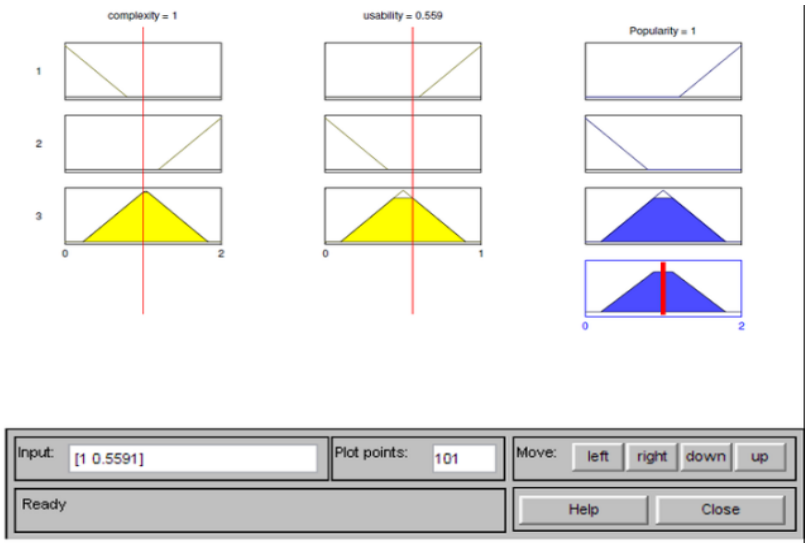

Input-I
Complexity (0-2)
Input-II
Usability (0-2)
Rule I:Complexity is Moderate Usability is also Moderate Software is popular
Rule II: Complexity is High Usability is Poor Software is NOT Popular
Rule III: Complexity is Low Usability is High Software is VERY Popular
∑
(Output) Popularity (0-10)

Figure 5.3: Framework developed using FRBS for the quantification of Software complexity and deliverability

The above figure illustrates the rule description of software complexity and usability using Mamdani FIS. Based on the linguistic set, the output is classified into three broad categories: High, Low and Moderate ranging from 0, 1 and 2.

5.5 SURFACE PLOT OF SOFTWARE COMPLEXITY, USABILITY & POPULARITY



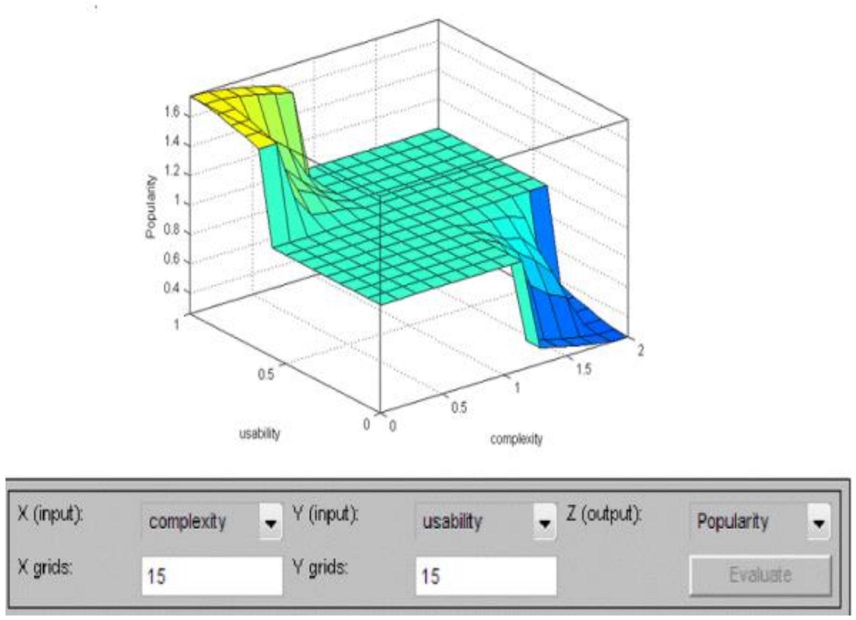
Input: [1 0.5591]

Plot points: 101

Move:

Ready

Figure 5.4: Snapshot of deriving a trade-off value of complexity and deliverability



X (input): complexity Y (input): usability Z (output): Popularity

X grids: 15 Y grids: 15

Figure 5.5: Snapshot of Surface plot of software complexity and usability

The surface plot for the above rule base shows the movement of the spike when software complexity is low and usability is high on the VERY popular side of the 3-D graph.

5.6 CONCLUSION

A framework has been developed which takes two inputs (Complexity and Usability) ranging from 0 to 2 and gives the output in the form of popularity ranging from 0 to 10. This framework uses three rules that have been established and quantifies the software attributes like complexity, usability and popularity.

CHAPTER 6

VERIFICATION OF THE FRAMEWORK USING HYPOTHESIS TESTING

6.1 INTRODUCTION TO HYPOTHESIS TESTING

Hypothesis testing is a process of making a choice between several competing hypotheses about probability distribution on the basis of the observed data distribution. Hypothesis Testing is a very prominently used method of verification that is used in statistics. In statistical hypothesis testing there is a statistical inference based on the data that has been gathered from a research or survey carried out. If the occurrence of the result is predicted as unlikely according to the pre-calculated threshold probability also referred to the significance level, then the result is called as statistically significant in statistics. Ronald Fisher was the person who initiated the concept of "test of significance". The tests of significance are used to determine that which outcomes of a research will direct to a denial for a pre-specified significance level of the null hypothesis. This provides contribution in deciding whether the results contain sufficient information or not in order to cast disbelief on predictable insight, to establish the null hypothesis, considering the fact that the usual perception has been applied. The critical region of the hypothesis test is defined to be the collection of all the outcomes that will cause the null hypothesis to be redundant in comparison to the alternative hypothesis.

Hypothesis testing is referred to as statistical or confirmatory data analysis as it has pre-defined hypotheses, in disparity to the exploratory method of data analysis that might not have pre-specified hypotheses.

One of the vital parts of the statistical inference is the setting up of the hypothesis and then testing the hypothesis. For formulating a test like this, some theory has to be set forward and that theory may be supposed to be accurate or it can be used as a source for the argument and then proved later. For example, claiming that a particular medicine for a particular ailment is better than the existing one.

6.2 STEPS OF HYPOTHESIS TESTING

The Hypothesis Testing is performed in following steps:

Step 1: Identify the hypothesis or claim that needs to be proved. For instance, if there is a need to determine that majority of users prefer less complex software in comparison to high complex software.

Step 2: Decide upon the criterion on the basis of which the user will decide whether the hypothesis being claimed upon is true or false. In a way, it can be said that in this step, the defined threshold value for deciding the truth or falsity of the hypothesis.

Step 3: The third step involves selecting a sample population and measuring the sample mean.

Step 4: In the last step, there is a comparison of the sample mean obtained in Step 3 above with the expected threshold that has been defined in Step 2. If there is a small difference jammed between the two means: the sample mean and the population mean, then the hypothesis is true else it is false.

For every problem under the consideration, the decision is based upon an issue that is of interest to us. Then there are two distinguishing claims that can be made about the issue that is termed as the hypothesis: one of them is the Null Hypothesis denoted by H0 and the other one is the alternative or the substitute hypothesis denoted by H1. The above said hypothesis a not observed on an identical basis, exceptional consideration has been given to the Null Hypothesis.

There are following two general situations:

1. The experimentation has been done in order to do the confirmation in opposition to it is adequately strong. For example,
   H0: Suppose that there is no distinction in flavour of Diet Pepsi and Pepsi against H1: Distinction between the two exists.

2. If either of the two hypotheses stated above is simple enough, there is a provision of more preference in comparison to the other complicated one so that the latter one is not adopted until and unless there exists an adequate amount of confirmation in support of the alternate hypothesis. For instance, it is very simple to declare that no variation in the taste or flavor exists between Diet Pepsi and Pepsi instead of saying that there exists a variation.

The assumptions or hypotheses are the statements that are very prominently used regarding the population parameters such as variance, expected value etc. For instance, the Null Hypothesis H0 can be the accepted value of the weight of eighteen year old boys in a population is not different from that of eighteen year old girls. A hypothesis can also be a statement that concerns a distributional figure of an attribute of interest.

The result of a hypothesis test is "Do not refuse H0" or "Reject H0 in favour of H1".

In order to evaluate the behaviour of a population that is too large or inaccessible, the use of inference statistics to study the behaviour in a sample of population as it allows us to do a more accurate study. Samples are used for evaluation as they are linked to the attributes of the population. The standard of the sample mean will be approximately equal to the value of the population mean, if an arbitrary sample is selected from a population. The method in which there is a need to make a decision about samples to study about attributes of a particular population is known as Hypothesis Testing. Hypothesis Testing is a regular approach to verify the claims or facts regarding an assembly or population.

6.3 HYPOTHESIS TESTING

In order to confirm our representation Hypothesis Testing was performed of our framework on a total of 100 Test samples (n=100). Here, the deliberated level of satisfaction and establish that mean to be equivalent to 70% (M=70) (70+10) i.e., μ =10. After calculating one independent sample Z-test, preserving of the Null Hypothesis is done.

(M=70%) at a 0.05 significance level (α=0.05). The trace the sample mean as 90% (M=90) is present.

STEP I: STATE THE HYPOTHESIS

The process begins with defining the population mean's value in a Null Hypothesis, which is considered as true. The Null Hypothesis H0 is a statement relative to a population parameter, like the population mean, that is hypothetical to be true. It is the preliminary assumption.

STEP II: LAY DOWN THE CRITERIA OF DECISION

In order to set criteria for a decision, there is a declaration of the level of impact for the test. During hypothesis testing, the collection of data is done to exemplify that the null hypothesis is false, depending upon the probability of choosing a sample mean from the population (the criterion is the likelihood). In behavioural research analysis, the significance level is usually fixed at 5% in. If the probability of achieving the sample mean is not as much as 5%.

The level of significance or the significance level refers to a standard upon which a decision is to be made with regards to the value settled in a Null Hypothesis. The criterion depends upon on the possibility of getting a statistic calculated in a sample in case the settled value in the null hypothesis is true.

The level of significance is 0.05, which makes $\alpha=0.05$. Now, in order to uncover the chance of a sample mean from a given population, the method which is taken is of standard normal distribution by placing standard normal distribution of Z-scores that are frequently cut offs or defined as critical values for the sample mean values lower than 5% probability of occurrence. After this, split the alpha value in half in a non-conditional two tailed, so that an identical proportion of area is placed in lower and upper tails.

Dividing α in half: $\alpha/2=0.05/2=0.0250$ in each tail.

The region ahead of the critical value of the hypothesis is the rejection region.

STEP III: THE TEST STATISTIC CALCULATION

A test statistic aids us in determining the number of standard deviations or the distance between the sample mean and the population mean. The larger is the test statistic's value; greater will be the distance, or the figure of the standard deviation. The determination of a sample mean from the population mean to test statistics value is considered to construct a decision in Step 4. In this stage the judgement of the generated value to the critical values occurs.

Z statistics: Z obtained = $M - \mu / \sigma M$

Where Z and $\sigma M = \sigma / \sqrt{n}$ statistics is inference statistics that is applied to resolve on the amount of standard deviations in the standard normal distribution.

The test statistic's value is the resulting value. In order to formulate the decision, the value of resultant statistics is compared with the critical values.

$\sigma M = \sigma/\sqrt{n} = 10/\sqrt{100} = 1$

Z obtained = $90-80/10 = 1$

STEP IV: COMPOSE A DECISION

The test statistic's computed value is used to compose the decision regarding null hypothesis. The result depends upon the possibility of getting a sample means, taking into consideration that the value known to Null Hypothesis will be true providing the value obtained in the sample mean is lower than 5% and then there is a decision of discarding the null hypothesis. However, if the probability of getting a sample mean is more than 5% while the null hypothesis is assumed to be true, then there comes the decision to maintain the Null Hypothesis. Apart from these, the following two decisions could be taken by the analyst:

- *Denial of the Null Hypothesis. In this case the sample mean is related with a low likelihood of occurrence when the null hypothesis is correct.*
- *Retention of the Null Hypothesis. In this case the sample mean is related with a high likelihood of occurrence while the null hypothesis is correct.*

The likelihood of obtaining a sample mean, taking into account that the value defined in the null hypothesis is true, is settled by the probability value p. The value of p ranges from 0 to 1 and can never be negative. In the next step, the settling of the probability of generating a sample mean is done and at that point there is a need to make a decision to discard the value defined in the null hypothesis, which is settled down at 5% in behavioural research.

In order to derive a conclusion, there is a need to place the value of p side by side to the criterion that has been set in Step 2. The probability of obtaining a sample result is p, in view of the fact that the value defined in the Null Hypothesis is true. The p-value obtained for generating a sample result is compared to the significance level.

A decision made related to a value defined in null hypothesis is explained using statistical significance. When the null hypothesis is discarded, the user is arrived at the significance and when the null hypothesis is retained, there will not be a success in attaining the significance.

Null hypothesis is discarded when the p value is lower than 5% ($p < .05$). Also, when the value of $p = .05$, the conclusion is still to discard the null hypothesis. However, in the case when the value of p is larger than 5% ($p > .05$), then there is a need to make a decision to retain the null hypothesis. Significance is mainly the decision of discarding or retaining.

There will not be a success to get to significance and the decision is to keep hold of this stage to compose a decision by comparing it with the critical value. The Null Hypothesis is refused if the generated value exceeds a critical value.

Table 6.1: Four Outcomes to make a Decision

	DECISION	
Truth in the Population	Retain the Null	Reject the Null
Truth	Correct	Error-α
Falsity	Error-β	Correct

In Step 4, there is point arrived where the decision whether to keep hold of or discard the null hypothesis takes place. As the evaluation of a sample and not the total population is taken place, it is likely that the conclusion may be incorrect. Table 6.1 above shows that there are four decision options regarding the falsity and truth of the decision that constructs concerning a null hypothesis:

- *The decision regarding retaining of the null hypothesis might be right.*
- *The decision regarding retaining of the null hypothesis might be incorrect.*
- *The decision regarding discarding of the null hypothesis might be right.*
- *The decision regarding discarding of the null hypothesis might be incorrect.*

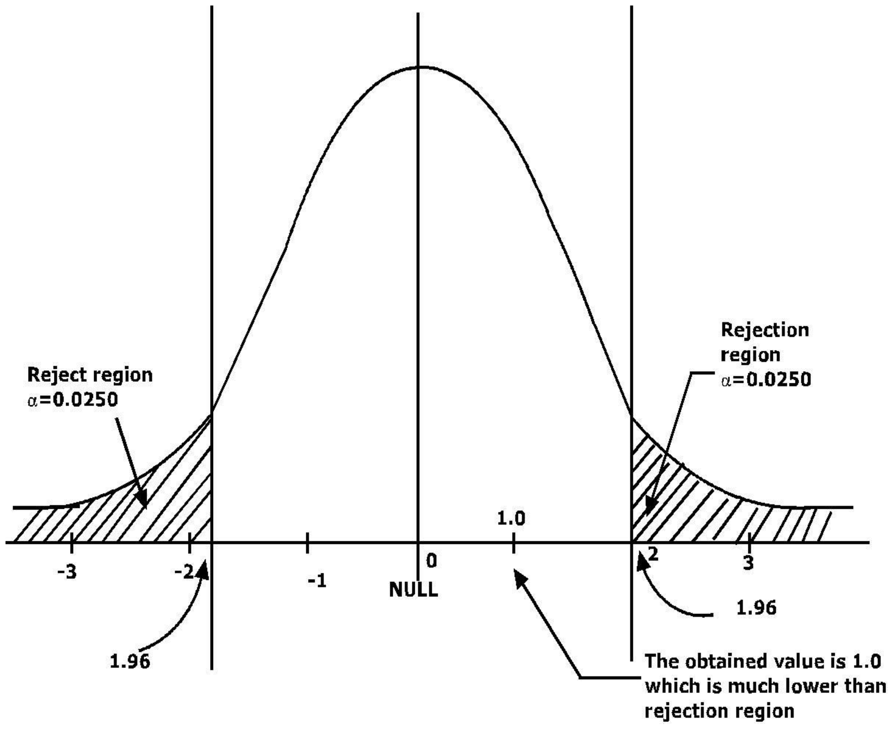


Figure 6.1: Acceptance of the Hypothesis

From Fig 6.1 there is a bringing to a close point that the framework has a reception of 70% supporting the Null Hypothesis.

6.4 MATHEMATICAL MODEL OF SIX FACTORS OF SOFTWARE USABILITY USING ROUGH SET THEORY

In order to analyze the relationship between the complexity of the software based on the various parameters like context shift, navigational guidance, input parameters and system feedback, the concept of rough set theory is used. Since two decades this approach is frequently used in the analyzing the relationship between various parameters.

Rough set methodology uses the concept of decision table consisting universe of discourse showing a relationship represented by two types of attributes. Basically it gives a concept of relationship between attributes called lower and upper approximate called the "Information System" given as 'S' where

S = (D,C,U)

Where

The point of interest is in the object set X ≤ U.

If, in case, A $\underline{C}$ Q determines a binary relation A_+ or U the n the relationship is called as indiscernibility relation.

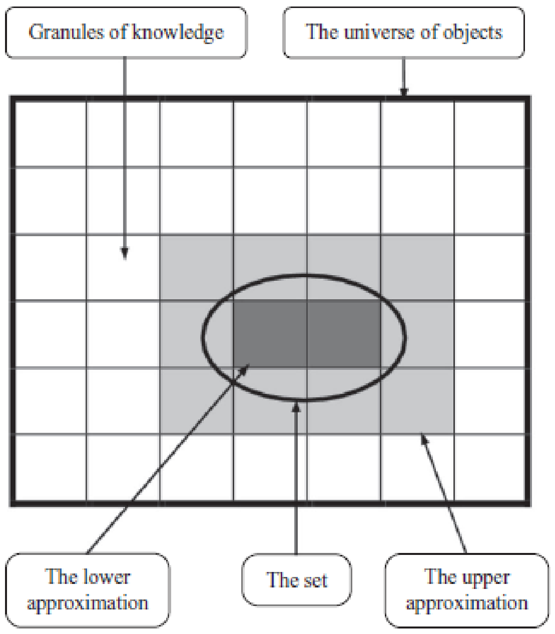


Figure 6.2: Basic Diagram of rough set theory using granular computing

Applying the above condition there is need to discriminate the factor which affects the "ease of use" factor.

Let,

U = Ease of use

A = Set of Navigational guidance

B = Levels of Navigational guidance

Table 6.2: Six levels of navigational guidance rating with example

NAVIGATIONAL GUIDANCE RATING	**EXAMPLE**
Level 1 (well constructed user interface navigation)	A step consists of completing the page of a wizard where the user is presented with one primary path for completing the step for given task.
Level 2 (basic user interface navigation)	A step consists of completing a user interface with the choice of several paths. The user is provided with textual guidance in the interface on how to complete the step for given task.
Level 3 (task oriented documentation)	A command-line step is fully documented in a procedural “step-by-step” description of the overall task in a User’s Guide manual.

Level 4 (basic documentation)	A command-line step is documented in a procedural “step-by-step” description of the task in a User’s Guide. However, the detailed syntax for this command is missing from the User’s Guide. Users need to search for the syntax detail in a Command Reference Manual.
Level 5 (unsupported navigation)	A step is not covered by product documentation, requiring the user to seek assistance through newsgroups, blogs, or product support channels.

Then

$U/A = \{X_1, X_2 \ldots\ldots\ldots X_n\}$

$U/B = \{Y_a, Y_b, \ldots\ldots\ldots Y_m\}$

Denote the parameters of “Ease of use”.

To determine the extent of partition of U/B^+ the definition

For example when A = {Ease of Use} and B = {Context shift}, then

$POS\_REG_A(B) = A_*(\{1, 2, 3, 4\}) \cup A_*(\{5, 6, 7\}) = \{1, 2, 3, 4, 5, 6, 7\} = U.$

Similarly, $BND\_REG_A(B) = \{1, 4, 5, 6, 7\}$ and $NEG\_REG_A(B) = Ø$.

Now there is a position to define several quantitative measures to relate two sets of attributes.

The measure is defined as

$$\Upsilon_A(B) = \frac{|\,\mathrm{POS\_\ REG_A(B)}|}{|\mathrm{U}|}$$ Equation 10

Clearly, $0 \leq \Upsilon_A(B) \leq 1$

$\Upsilon_A(B) = 1$,

$\Upsilon_A(B) = 0$,

In other situations there is roughly dependency.

In above example, there is an observation that B is totally dependent on A. This is because every B-granule, B $\underline{\subset}$ A, is a composed set of union of A-granules. The dependency measure is 1 in such cases.

Here to define another measure of dependency alternative, the use of discriminant index is taken for consideration.

6.5 DISCRIMINANT INDEX

The discriminant index $\beta_A(B)$ is defined as

$$\beta_A(B) = \frac{|\,\mathrm{POS\_\ REG_A(B)} \cup \mathrm{NEG\_REG_A(B)}|}{|\mathrm{U}|} = \frac{\mathrm{U-BND\_\ REG(B)}|}{|\mathrm{U}|}$$

Equation 11

For example, when A = Ease of use and B = Context Shift, then $\beta_A(B) = 5/7$.

Here the point is to be noted that when the boundary region of B with respect to A is empty, the discriminant index is 1. The definition can also be given of another measure of importance called significance of B on the set of all condition attribute without a.

6.6 SIGNIFICANCE

The equation can be further extended the significance to a subset A also. When there are few elements in the positive regions, it is not useful to have the dependency and discriminant indices. Under such circumstances the significance measure becomes useful.

Let us elaborate the above example by using some more variables on “Ease of Use”. Having four condition attributes given as.

Table 6.3: Six identified factors of software usability

Condition Attribute	**Context Shift**	**Navigation guidance**	**Input parameter**	**System feedback**	**Error Feedback**
1	1	2	2	1	1
2	1	2	3	2	1
3	1	2	2	3	1
4	2	2	2	1	1
5	2	3	2	2	2
6	1	3	2	1	1
7	1	2	3	1	2
8	2	3	1	2	1
9	1	2	2	2	1
10	1	1	3	2	1
11	2	1	2	2	2
12	1	1	2	3	1

Let us determine equivalence class of condition attribute context shift is denoted by context shift.

Consider another example where there are four conditions attributes named as a, b, c, and d, and e is decision attribute. All the tuples together constitutes U.

Let us determine equivalent classes for each of the individual condition attributes. The equivalence relation to a (condition attribute) can be denoted with a^+. There are two distinct values of a in V_a. Hence U is partitioned into two classes by a^+.

The representation of these classes is done by identifying the corresponding attribute values, for example by $a^{+}_{a=1}$ we mean the equivalent class corresponding to the value 1. Thus, the following different partitions are generated:

For the attribute a, the different granules are the following.

$a^{+}_{a=1} = \{1, 2, 3, 6, 7, 9, 10, 12\}$;
$a^{+}_{a=2} = \{4, 5, 8, 11\}$

Similarly, for attribute b:

$b^{+}_{b=1} = \{10, 11, 12\}$;
$b^{+}_{b=2} = \{1, 2\ 3, 4, 7, 9\}$;
$b^{+}_{b=3} = \{5, 6, 8\}$;

For the attribute c:

$c^{+}_{c=1} = \{8\}$;
$c^{+}_{c=2} = \{1, 3, 4, 5, 6, 9, 11, 12\}$;
$c^{+}_{c=3} = \{2, 7, 1\}$;

For the attribute d

$d^{+}_{d=1} = \{1, 4, 5, 6, 7\}$;
$d^{+}_{d=2} = \{ 2, 5, 8, 9, 10, 11\}$;
$d^{+}_{d=3} = \{3, 12\}$;

For the set X = {1, 2, 3, 4, 8, 9, 10, 12}, which is a granule for e =1, the upper and lower approximation of X

Boundary of X with respect to a is given by

BND_a (x) = a*(X) – a ~~x~~ (X) = {1, 2,........, 12};

Similarly,

BND_a (x) = b*(X) – b ~~x~~ (X) = {1, 2,........., 12};

BND_a (x) = c*(X) – c ~~x~~ (X) = {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12}.

BND_a (x) = d*(X) – c~~x~~ (X) = {1, 2, 4, 5, 6, 7, 8, 9, 10, 11,}.

The other partition of U with respect to e = 12 is the set Y = {5, 7, 11}

a_* (Y) = {1, 2, 3,........., 11, 12}.

b_* (Y) = {1, 2, 3, ..., 12}.

c_* (Y) = {1, 2, 3, ..., 11, 12}.

d_* (Y) = Ø, d_* (Y) = {1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12}.

Boundary of Y with respect to a is given by

BND_a (Y) = a*(Y)- a_* (Y) = {1, 2, 3, 4, 5, 6, 8, 7, 9, 10, 11, 12}.

Similarly,

BND_b (Y) = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}.

BND_c (Y) = {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12}.

$BND_d (Y) = \{1, 2, 4, 5, 6, 7, 9, 10, 11\}$.

$POS\_REG_a (\{e\}) = a_* (X) \cup a_* (Y) = Ø$

$BND\_REG_a (\{e\}) = BND_a (X) \cup BND_a (Y) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$.

$NEG\_REG_a (\{e\}) =Ø$

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

The conclusion of the work can be summarized in following points:

- Complexity of the software varies from person to person.

- In an observation, it has been analyzed that when the software complexity increases, the usability aspect of the software drops significantly.

- It is concluded that systems having lower complexity level as well as at the same time with high aspects of usability are preferred by maximum users, which results into the higher deliverability factor the software application.

- The novelty of the current work is use of soft computing techniques in analyzing the trade-off among the software's complexity and software's deliverability.

- The second phase of the work done was development of a framework using Fuzzy Rule based System (FRBS) for quantification of software complexity and usability aspects.

- In the final phase, the development of Fuzzy Inference System (FIS) using Expert Knowledge Base was done.

In future, the work can be extended by applying this framework for further elaborating the software complexity and deliverability aspects and by using more concepts of soft computing. The work can be further extended by enhancing the framework by identifying more usability aspects of the software and the application of the framework can be possible in various types of software to analyze the complexity aspect of the software to determine its future market potential.

CHAPTER 8
REFERENCES

[1] Juristo, N., Moreno, A. M., & Sanchez-Segura, M. I. (2007). Analyzing the impact of usability on software design. Journal of Systems and Software, 80(9), 1506-1516.

[2] Carvajal, L. (2009, August). Usability-enabling guidelines: a design pattern and software plug-in solution. In *Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium* (pp. 9-12). ACM.

[3] Nichols, D., & Twidale, M. (2003). The usability of open source software. *First Monday*, *8*(1).

[4] Poore, J. H., Walton, G. H., & Whittaker, J. A. (2000). A constraint-based approach to the representation of software usage models. *Information and Software Technology*, *42*(12), 825-833.

[5] Coulier, W., Garijo, F., Gomez, J., Pavon, J., Kearney, P., Massonet, P., & Fuentes, R. (2004). MESSAGE: a Methodology for the Development of Agent-based Applications, en Methodologies and Software Engineering for Agent Systems—The Agent-Oriented Software Engineering Handbook.

[6] J. K. Nurminen (2003). By Using the software complexity measures to analyze the algorithms—an experiment with the shortest-paths algorithms. *Computers and Operations Research*, *30*(8), 1121-1134.

[7] Y. Wang, & V. Chiew (2011). The Empirical studies on the functional complexity of the software in very large-scale software systems. International Journal of Software Science & Computational Intelligence, Volume 3 Issue 3.

[8] E. Allen, B., Gottipati, S., and R. Govindarajan, (2007). Measuring the size, complexity, and the coupling of the hyper graph abstractions of a software: An information-theory approach. *Software Quality Journal*, *15*(2), 179-212.

[9] M. Burgin, and N. C. Debnath (2003). Complexity of the Algorithms and the Software Metrics. In *Computers and Their Applications* (pp. 259-262).

[10] A. Sharma, and D. S. Kushwaha, (2010). Early estimation of the software complexity by requirement engineering documents. *ACM SIGSOFT Software Engineering Notes*, *35*(5), 1-7.

[11] D. S. Kushwaha, and A. K. Misra (2006, February). A complexity measure based on the information enclosed in the software. In *Proceedings of the 5th WSEAS International Conference on Software*

Engineering, Parallel and Distributed Systems (pp. 187-195). World Scientific and Engineering Academy and Society (WSEAS).

[12] P.K. Shukla and S. P. Tripathi (2011). A Survey on the Interpretability-Accuracy (I-A) The Trade-off in the Evolutionary Fuzzy Systems, 2011 5th International Conference on Genetic and Evolutionary Computation (ICGEC), Xiamen, pp. 97-101, 29 Aug.-1 Sept. 2011.

[13] M. B. Gorzalczany and F. Rudziriski , Accuracy vs. Interpretability of the Fuzzy Rule Based Classifiers: an evolutionary approach, Swarm and the Evolutionary Computation, LNCS, 7296/2012, 222-230 (2012)

[14] C. A. Coello, D. A. Veldhuizen and G. B. Lamont, The Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic publishers, New York, (2002)

[15] C. Coello, L. Abraham, and R. Jain, The Recent trends in Evolutionary Multi-Objective Optimization: Theoretical Advances and Applications (Springer-Verlag, London, 2005)7-32 (2005).

[16] Coello, C, G. Toscano, & E. Mezura, The Current and Future Research Trends in Evolutionary Multi-Objective Optimization, in: M. Grana, R. Duro, A. d' Anjou, P. P. Wang (Eds.) Information processing and Evolutionary Algorithms: From Industrial Applications to Academic Speculations (Spring-Verlag, London, 2005) 213-231 (2005).

[17] C. Coello, Evolutionary Multi-Objective Optimization: The Historical View of the Field, IEEE computational Intelligence Magazine, 1:1, 28-36 (2006).

[18] D.E Goldberg, The Genetic algorithms in the search optimization and machine learning, Addison Wesley Publishing Company Reading Massachusetts, 1989.

[19] H.P. Schwefel, The Evolution and the optimization of seeking, John Wiley & Sons, New York, 1995.

[20] J. R Koza, Genetic Programming on the Programming of the Computers by Means of Natural Selection, The MIT Press, Cambridge, Massachusetts, (1992).

[21] L. J. Fogel, Artificial Intelligence by means of simulated evolution, John Wiley, New York, 1966.

[22] C. Coello, R. Jain, & Abraham, L., The Recent trends in the evolutionary multi-objective optimization. Theoretical Advances and Applications (Springer-Verlag, London, 2005)7-32 (2005).

[23] Coello, C. A., Veldhuizen, D.A., and Lamont, G. B., Evolutionary Algorithms for solving the multi-objective problems, Kluwer Academic Publishers, New York, (2002).

[24] Srinivas, N., & Deb, K., Multi-objective optimization using non-dominated sorting in genetic algorithms, Evolutionary Computation, 2(3), pp. 221-248, 1994.

[25] Horn, J., and Goldberg,E., A niche Pareto genetic algorithm for the multi-objective optimization, in: proc. Ist IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 1, 82-87, 1994.

[26] Fonseca, C. M., & Fleming, P. J., Genetic algorithms for multi-objective optimization: formulation, discussion and generalization, in proc. 5th International Conference on Genetic Algorithms, 416-423, (1993).

[27] Zitzler, E., & Thiele, L., Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation, 3 (4), 257-271, (1999).

[28] Zitzler, E., Laumanns, M., & Thiele, L., SPEA2: Improving the strength pareto evolutionary algorithms, Technical Report 103, Computer Engineering & Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001).

[29] Knowles, J. D., & Corne, D. W., Approximating the non-dominated front using the Pareto achieved evolution strategy, Evolutionary Computation, 8 (2), 149-172 (2000).

[30] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T., A fast and elitist Multi-objective genetic algorithm: NSGA II, IEEE Transactions on Evolutionary Computation, 6 (2), 182-197, (2002).

[31] Erickson, M., Mayer, A., & Horn, J., The Niched Pareto Genetic Algorithms applied to the design of ground water remediation system, Ist International Conference on Evolutionary Multi Criteria Optimization, 681-695, Springer-Verlag, LNCS, No. 1993 (2001).

[32] Corne, D. W., Knowles, J. D., & Oates, M. J., The pareto envelop based selection algorithm for multi-objective optimization, In Proc. VI Conference of Parallel Problem Solving from Nature, pp. 839-848, Paris, France, Springer LNCS 1917 (2000).

[33] Coello, C. A. C., & Pulido, G. T., A micro genetic algorithm for Multi objective optimization, in: proc. First International Conference on Evolutionary Multi-Criteria Optimization, pp. 126-140, LNCS 1993 (2001).

[34] Coello, C. A. C., & Pulido, G. T., Multi-objective optimization using a micro-genetic algorithm, Proc. Genetic and Evolutionary Computation Conference (GECCO' 2001), pp. 274-282, Morgan Kaufmann Publishers (2001).

[35] Thift, P., Fuzzy Logic Synthesis with Genetic Algorithms, in Proc 4th Int. Conf. Genetic Algorithms (ICGA), San Diego, CA, pp 509-51301991.

[36] James, M., Mahfonf, M., & Linkens, D.A., Elicitation and fine tuning of Fuzzy Control rules using symbiotic evolution, Fuzzy states and systems, Elsevier, 2004.

[37] Dia, Y., & Keller, Quantifying the complexity of IT Services Management Processes, IBM Research Report, 2006.

[38] Sobiesiak, R., & Keefe, T.O., Complexity Analysis: A Quantitative Approach to Usability Engineering, IBM Design: papers and Presentations, 2009.

[39] Dia, Y., & Sobiesiak, R., Quantifying Software Usability through Complexity Analysis, IBM Design: papers and Presentations, 2010.

[40] Shukla, P. K., & Tripathi, S.P., Interpretability issues in Evolutionary Multi-Objective Fuzzy knowledge Base Systems, 7th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), ABV-IIITM, Gwalior, India, 14-16 December, 2012.(Springer AISC SERIES).

[41] Darbari, M., & Yagyasen, D., Application of Granulized OWL framework for modelling Urban Traffic System, Parsec Multi Displinary Journal, Vol. 75, Issue 9, 2013.

[42] Ahmed, S.S., Purohit, H., Ashaikhly, F., & Darbari, M., Information Granular for Medical Infonomics, International Journal of Information and Operations Management Education (IJIOME), Vol. 5, No.3, 2013 Inderscience.

[43] Shukla, P.K., & Tripathi, S.P., A Survey on Interpretability Accuracy Trade-Off in Evolutionary Fuzzy Systems, IEEE International Conference on Genetic and Evolutionary Computation (ICGEC 2011), Japan, 29 August-01 September, 2011. (IEEE Xplore).

[44] Angelov, P., An approach for fuzzy rule-base adaptation using on-line clustering, Volume 35 Issue 3, March 2004, Pages 275–289, Elsevier.

[45] Cordón, O. et al., Ten years of genetic fuzzy systems: current framework and new trends, IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th. Vol. 3. IEEE, 2001.

[46] Cordón, O. et al., Genetic fuzzy systems. Singapore: World Scientific Publishing Company, 2001.

[47] Hoffmann, F., Combining boosting and evolutionary algorithms for learning of fuzzy classification rules, Fuzzy Sets and Systems 141.1 (2004): 47-58.

[48] Alcalá, R. et al., Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation, International Journal of Approximate Reasoning 44.1 (2007): 45-64.

[49] Fernández, A., et al., A study of the behavior of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets, Fuzzy Sets and Systems 159.18 (2008): 2378-2398.

[50] Esfahanipour, A., & Aghamiri, W., Adapted Neuro-fuzzy inference system on indirect approach TSK fuzzy rule base for stock market analysis, Expert Systems with Applications 37.7 (2010): 4742-4748.

[51] Alcalá, Rafael, et al. A multi-objective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems. Fuzzy Systems, IEEE Transactions on 17.5 (2009): 1106-1122.

[52] Cordón, O., Herrera, F., & Villar, P., Generating the Knowledge Base of a Fuzzy Rule-Based System by the Genetic Learning of the Data Base, IEEE Transactions on fuzzy systems, Vol. 9, No. 4, August 2010.

[53] Ishibuchi, H., Evolutionary Multi-objective Design of fuzzy Rule-Based Systems, Foundations of Computational Intelligence, 2007. FOCI 2007, IEEE.

[54] Yu-Chuan, C., Fuzzy Interpolative Reasoning for Sparse Fuzzy-Rule-Based Systems Based on the Areas of Fuzzy Sets, IEEE TRANSACTIONS 2008.

[55] Angelov, P., & Yager, R., A Simple Fuzzy Rule Based System through vector membership and kernel based granulation, 5th International Conference IEEE, 2010.

[56] Singh, A., & Wesson J., Evaluation Criteria for assessing the usability of ERP Systems, SAICSIT' 09-12-14 October 2009, Riverside, Vanderbilt Park, South Africa. (ACM).

[57] Parks, N. E., Testing & Quantifying ERP Usability, RIIT'12, October 11-13, 2012, Calgary, Alberta, Canada (ACM).

[58] Ince, F. I., Salman, B. Y., & Yidrim, E. M., A User Study: The Effects of Mobile Phone Prototypes and Task Complexities on Usability, ICIS 2009, November 24-26, 2009 Seoul Korea. (ACM).

[59] King, P., the Promise and Performance of Enterprise Systems in Higher Education, ESAR, 2002.

[60] Retna, E.J., Varghese, G., Soosaiya, M., & Joseph, S., A Study on Quality Parameters of Software and the Metrics for Evaluation, IJCET Volume 1, 05-06, 2010, pp 235-249.

[61] Amid, A., Moalagh, M., & Ravasan, Z.A., Identification and Classification of ERP Critical Failure Factors in Industries, Information Systems 37 (2012) 227- 237. Elsevier.

[62] Sobiesiak, R., & Tim O’Keefe, T., Complexity Analysis: A Quantitative Approach to Usability Engineering, IBM Canada Laboratory & IBM Rochester Laboratory, 2006.

[63] White paper, Usability in software design, Microsoft Corporation, 2007.

[64] Uflacker, M., & Busse, D., Complexity in Enterprise Applications vs. Simplicity in user experience, (SAP Labs), 2009.

BOOKS:

- J. Nielsen and R. L. Mack, “Usability Inspection Methods”, John Wiley & Sons Inc 1994, ISBN: 0-471-14965-9.
- Neilson Jacob, “Usability Engineering”, Boston AP Professional, 1994, ISBN: 0-12-518400-X.
- Dumas, D., S. Joseph, & Janice C. “A practical guide to the Usability Testing of Software”, London: Intellect Books, 1999. ISBN: 1841500208.

- L. Zadeh, “Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems”, Selected Papers by Lotfi A Zadeh edited by: George J Klir (SUNY, Binghamton) edited by: Bo Yuan (SUNY, Binghamton). ISBN: 978-981-02-2421-9.

ANNEXURE A

DATA COLLECTED USING THE QUESTIONNAIRE 1 & 2

BANK-A		BANK-B		BANK-C	
CS-1	**DL-1**	**CS-2**	**DL-2**	**CS-3**	**DL-3**
31	81	62	42	84	28
34	75	71	36	91	25
41	73	76	33	92	24
47	68	87	37	108	27
42	61	91	26	104	19
39	80	56	33	90	15
43	72	40	35	98	19
50	58	30	25	110	35
53	60	35	28	93	45
44	49	45	38	106	55
59	69	55	48	96	68
46	57	65	49	82	20
55	72	75	66	93	45
47	63	85	75	102	55
52	66	95	85	97	43
48	67	84	62	102	46
37	54	53	31	99	30
41	49	64	48	101	52
60	77	61	31	41	18
65	87	71	41	51	20

70	90	81	51	61	30
75	98	91	61	71	40
80	97	62	32	81	50
85	99	72	42	91	60
90	82	82	52	42	19
95	105	92	62	52	22
61	74	63	33	62	32
71	83	73	43	72	42
81	94	83	53	82	52
91	101	93	63	92	62

ANNEXURE B

QUESTIONNAIRE 1

Name: ....

Designation: ....

Years of Experience: .... (Years)............. (Months)

<u>Rate the following tasks on the scale of 0(least) to 20(Highest) :</u>

1- Difficulty level to locate/identify the login page?

2- Difficulty level to enter the parameters (Login_Id/Password)?

3- Ease of use in entering the correct parameters?

4- Support level provided by the system in guiding the procedure of entering login details?

5- Difficulty level to locate the desired operation in the menu bar?

6- Difficulty level to operate the task?

7- Support level provided by the system to navigate from login page to the desired operation's menu?

8- Significance level of Information display on alert box?

9- Understanding level of the language of message in the alert box?

10- Difficulty to locate the cause of error if occur?

11- Difficulty level to go to return menu?

12- Difficulty level in aborting the operation?

13- Difficulty level in fetching the data from the database?

14- Satisfaction level on the amount of data entered to fetch the record?

15- Difficulty to select the exact data in case similar records have been found?

16- Level of ease in locating the exact data?

17- Level of ease in extracting the data?

18- Level of ease in completing the task?

19- Difficulty level in storing the data in database?
20- Difficulty in accessing the printer in case printing is required?
21- Level of ease in returning to the main menu?
22- Ease of use with graphical user interface provided?
23- Difficulty level to report any error if occurs?
24- Difficulty level in finding the logging out option?
25- Satisfaction level on the information provided regarding the completion of task?

(SIGNATURE)

ANY OTHER COMMENTS:-

ANNEXURE C

QUESTIONNAIRE 2

Name: ....

Designation: ....

Years of Experience: .... (Years)............. (Months)

a. **Rate your users experience with the software?**
    1. Excellent
    2. Very Good
    3. Good
    4. Average
    5. Poor

b. **Rate your comfort in shifting between the software module according to ease of use.**
    1. Excellent
    2. Very Good
    3. Good
    4. Average
    5. Poor

c. **How do you find the navigational guidance effective during the context shift?**
    1. Excellent
    2. Very Good
    3. Good
    4. Average
    5. Poor

d. **Rate the software data input features in the package.**

   1. Excellent
   2. Very Good
   3. Good
   4. Average
   5. Poor

e. **How often do you that the software hangs up during operation?**

   1. Excellent
   2. Very Good
   3. Good
   4. Average
   5. Poor

f. **How often do you detect errors and give feedback?**

   1. Excellent
   2. Very Good
   3. Good
   4. Average
   5. Poor

SIGNATURE:

ANY OTHER COMMENTS: