Palestine Polytechnic University
Deanship of Graduate Studies and Scientific Research
Master of Informatics

Reliability Prediction from Early Software Development Life Cycle Phases

Submitted by:
Mohammad AL-Jundi

Supervisor:
Dr. Mahmoud Saheb

Thesis submitted in partial fulfillment of requirements of the degree "Master of Science in Informatics"

DECLARATION

I declare that the Master Thesis entitled "Reliability Prediction from Early Software Development Life Cycle Phases" is my original work, and Herby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Mohammad AL-Jundi

Date:_____ Signature:_____

STATEMENT OF

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for the master's degree in Informatics at Palestine Polytechnic University, I agree that the library shall make it available to borrowers under the rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from, reproduction or publication of this thesis may be granted by my main supervisor, or in his absence, by the Dean of Graduate Studies and Scientific Research when, in the opinion of either, the proposed use of the material is for scholarly purposes.

Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Mohammad AL-Jundi

Date:_____ Signature:_____

DEDICATIONS

This thesis is dedicated to my greatest parents, my great brothers and sisters,my dearest wife, and my beloved kids: Mahmoud, Joury, and Taim.

ACKNOWLEDGMENTS

Many people have directly or indirectly contributed to the successful completion ofthis thesis. They will all be remembered in my heart. First, I would like to takethis opportunity to highly appreciate my thesis supervisor Dr. Mahmoud Al-Saheb for hisexemplary guidance and encouragement throughout my thesis research and writing.

I appreciate my family's encouragement, especially my greatest parents, wonderful wife, great brothers, and sisters. Their support and encouragement were the reason for all the success which Ihave made.

I express my deep sense of reverence and gratitude to all of my respected teachers and colleagues for their valuable knowledge they imparted to me. And finally, my special thanksshould be brought to Dr.Ali Zein for his constant inspiration and motivation.

Also, I would like to thank the project managers of Newsoft company for their assistance in analyzing some factors which are used in this thesis.

الملخّص

في هذه الأطروحة، تم اقتراح نموذج تنبؤ لاحتساب الأخطاء في المرحلتين الأوليين من دورة حياة تطوير البرمجيات، للسماح لمديري المشاريع باتخاذ القرار المناسب في الوقت المناسب. تستخدم هذه الأطروحة المقاييس الأكثر ارتباطاً بموثوقية البرمجيات في المراحل الأولى من دورة حياة تطوير البرامج. تم احتسابالأخطاء باستخدام نظام الضبابية،ثم تمت مقارنة النتائج بنماذج أخرى لاحتساب الموثوقية، والتي تم نشرها بعد عام ٢٠٠٥ وتعتمد في عملها على مجموعة بيانات لفحص النموذج. تم إجراء المقارنة من خلال حساب متوسط حجم الخطأ النسبي (MMRE) ومتوسط توازن حجم الخطأ النسبي (BMMRE)لكل نموذج لمقارنة العيوب المتوقعة من النموذج المقترح والنماذج الأخرى. أظهرت نتائج MMRE & BMMRE أن النموذج المقترح له القيم الدنيا ، مما يعني أن دقته هي الأعلى مقارنة بالنماذجالأخرى. من خلال النتائج والمقارنات أظهر هذا النموذج أنه الأكثر موثوقية وفعالية مقارنة بالنماذج الأخرى. فيما يتعلقبمجموعة البيانات المستخدمة للتحقق من صحة النموذج، تم استخدام مجموعة بيانات تُعنى بالبرمجيات المضمنة في المعدات الإلكترونية ، لذلك يجب التحقق من صحة النموذج باستخداممجموعة بيانات لنوع آخر من الأنظمة ليتسنى تعميم النموذج على الأنواع الأخرى.

Abstract

In this thesis, a prediction model has been proposed to calculate the defect density number within the first two stages of the software development life cycle to let the projects managers take the right decision at the right time. This thesis uses the most related metrics to software reliability within the early stages of the software development life cycle. The residual defects were calculated using the fuzzification and defuzzification system. Then the results have been compared with other quantitative reliability models, which are published after 2005 and havea dataset. The comparison has been madeby calculating the Mean Magnitude of Relative Error and Balance Mean Magnitude of Relative Error factors for each model to compare the predicted defectsfrom the proposed model and the other models. The results of MMRE & BMMRE showed that the proposed model has the minimum values, which means its accuracy is the highest compared with the other models that can be used within the early stages of the software development life cycle. This model showed that it is reliable and effective.Depending on the used dataset to validate the model, the results are considered to the software embedded into the electronic equipment, so the model should be validated by another type of software dataset to be used more generally.

Keywords

Reliability Prediction, Software Reliability, Software Development Life Cycle, Fault Density Indicators, Reliability Indicators.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

No	Abbreviation	Definition
1	CC	Complexity of functionality
2	DPDDI	Design Phase Defect Density Indicator
3	DRE	Design Review Effectiveness
4	H	High
5	L	Low
6	M	Medium
7	QDI	Quality of Documentation Inspected
8	RFD	Requirement Fault Density
9	RIW	Regularity, Inspection, and Walk-through
10	RPDDI	Requirement Phase Defect Density Indicator
11	RS	Requirement Stability
12	SDD	Software Design Document
13	SDLC	Software Development Life Cycle
14	SNF	Scale of New functionality
15	SRS	Software Requirement Specifications
16	VH	Very High
17	VL	Very Low

Chapter 1: Introduction

At present, most people depend on software directly orindirectly.Governments and human's dependence on software have been increased during the last thirty years [1]. Software reliability and qualitymodeling become essential because the software is used in various areas of applications. Many historical events show theeffect of software failures and defectsthat existed in [1].Hence, software developers have a great challenge, which is to build reliable software thatcontains fewer defects. So, the reliabilityprediction of software is of great importance. An accurate estimation of the reliability can be obtained using reliability models only in the later phases of softwarebuilding.The existing reliability models can be applicable only in the later stages of development and helping developers either by the end of coding or in the testing phase. This is too late for the developers to take corrective measure to improve software reliability. However, with the objectives of cost-effectiveness and management of time and effort of resources, the reliability prediction of thesoftware in the early phases of the Software Development Life Cycle (SDLC) is one of the significant areas of concern.

Software Reliability is the probability that a system can deliver its purposed functionality and quality for a specified period, and under specified conditions, and at the start of this period [2]. The main method to measure the software quality is to detect the defects in the software, and usually, the metric used for this issoftware defect density. The software defect density is the total number of defects divided by the size of the software [3].

Software defect density prediction has an important role inproducing reliable software. In order to achieve the aim of defect estimate,it is required to predict the defect density indicator in each phase of the SDLC from the early phases, mainly at the end of each phase. Many models have been proposedfor the estimation and prediction of software reliability in thepast three decades. It is observed that traditional models, which predict the defect density at the end of the testing phase of SDLC phases,for software reliability prediction are not universally successful in predictingthe reliability of the software and not generally tractable tousers [5].

So, the failure of information during the early phasesof the SDLC is available in the form of expert knowledge, whichmay be reflected in terms of software metrics [6].

This chapter outlines the problem statement (section 1.1) and research objectives (section 1.2) of the research, and its purposes (section 1.3). Section 1.4describes the significance and scope of this research. Finally, section 1.5 includes an outline of the remaining chapters of the thesis.

1.1 PROBLEM STATEMENT

Many approaches have been developed to predict the fault density of the application, but most of them calculate the fault density depending on metrics related to coding and testing phases of SDLC [4][21][22], which means they can get an indicator about software reliability at the end of coding or testing phase of SDLC, which is too late because it will be costly in time and budget to reanalyze or redesign some features of the software.

Other approaches have been developed depending on metrics in the first two phases of SDLC [16][18], but these approaches are not widely used and cannot predict accurate reliability of software because of some factors; such as:

1- Depending on metrics in the first phase of SDLC only, which causes a lack of information about the software, so they get an inaccurate result about software reliability.

2- Depending on metrics that are not highly related to software reliability, or they have noeffects on reliability directly.

3- Limitation of software scope, some of the approaches are limited to onlyobject-orientedapplications or limited procedural applications.

Furthermore, we have no approach that can be used to predict the software reliability within the early stages of SDLC, to give the project managers theability to go ahead in software development or to solve some issues in analysis or design phases before starting the implementation phase to save time and budget, because of

getting back and solving the issues has less effort and budget than completing the software development and discovering them lately.

1.2 RESEARCH OBJECTIVES

This research aims to develop a solution for software reliability prediction within the early stages of the software development life cycle, which should use the most highly related metrics to the reliability to get the most accurate results and can be used for deciding to go ahead or not through the SDLC.This research intends to address the following objectives:

- Developing a model that can show the project manager an indicator about the faults which may appear in the software.
- The validity of using the proposed model.
- Compare the proposed model with other models to ensure that the results are more accurate than the other models that have been developed to predict reliability in the early stages of SDLC.

1.3 RESEARCH QUESTIONS

The research questions section represents some sub-objectives of the research. We achieve ourobjectives of the research by implementing every sub-objective.Experiments wereconducted to seek answers to the following research questions:

Research Question 1: How could we choose some metrics of early stages of SDLC to enhance the calculation of software defect density?

Research Question 2: What is the effect of chosen metrics on software defect density accuracy?

Research Question 3: How can the use of the proposed model improve software development?

1.4 MOTIVATIONS

Nowadays, as software development has many methodologies, which improve the creation of more reliable software, despite that, human needs to get reliable software has increased; because software playsa major role in human life, and it is used in most critical aspects of his life. The failure of software development somehow exists after applying the new methodologies, and some companies or developers waste time and budget in building software, but the faults will be discovered after implementation and testing, then sometimes it is hard or impossible to reanalyze or redesign the requirements, which means the software development has failed.

Thus, there is an essential need to find an effective, feasible, and accurate solution for finding defects in the early stages of SDLC, to allow the developers to avoid and solve these defects before starting the implementation and testing the software. Detecting and solving the defects in software during the early stages of SDLC is an active research topic in academic corporations andindustries. In this work, the authors use the four most metrics highly related to reliability in the requirement phase and three metrics in the design phase to predict the defect density more accurately.

1.5 THESIS OUTLINE

The rest of this thesis is organized as follows. Chapter 2: introduces software reliability background.Chapter 3: illustrates the literature review.Chapter 4: explains the proposed model development methodology.Chapter 5:illustrates the case study, which is used to validate the model,Chapter 6:shows the model results and its validation.Chapter 7:concludes the thesis along with the future work,followed by the references.

Chapter 2: Background

This chapter provides a background forthe first two main SDLC phasesand some of their metricsthatare related to software reliability and their importance. Next, this chapter also gives an overviewof software reliability in SDLC, especially in the earlystages.

2.1 SDLC

SDLC is a term that refers to the Software Development Life Cycle, which means: "The period that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirement phase, design phase, implementation phase, test phase, installation and check out phase, operation, and maintenance phase, and sometimes retirement phase." [39] As shown in Figure 2.1.Or in other words: it is the progress that is used to produce software with the highest quality and lowest effort and budget in the planned time. SDLC usually includes a detailed plan for how to build the software, maintain, and install it with the replacement of the old software system.It is used to facilitate the production of large software, and it contains many phases starting from analyzing the requirements and ending with the testing and delivering the software. It contains all of the information related to software from conception to implementation.

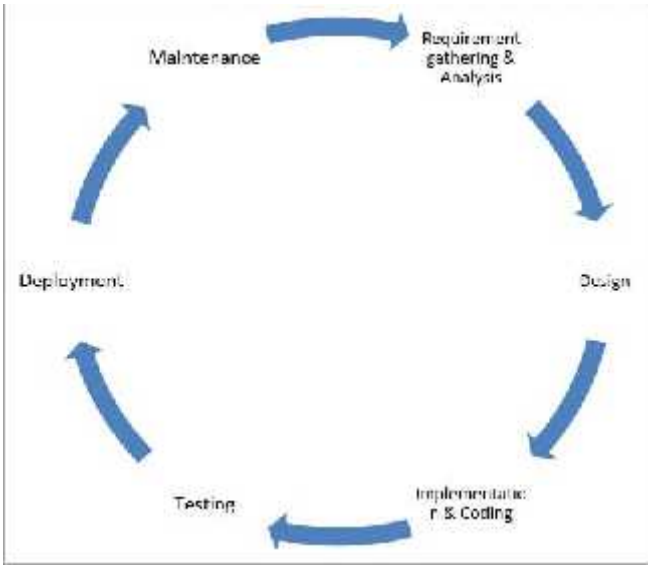

Figure 2.1: SDLC Phases

There are many reasons that force developers to use SDLC within software development, which are [24]:

- Understanding all of the processes of software.
- Having a clear and structured approach to build the software.
- Enabling the developers to plan the resources that will be used.
- Enabling the developers to track the progress of the system.

2.1.1 Requirement Phase

It is considered as the first main step for software developers; its output is a document called SRS, which refers to Software Requirement Specification; it contains the description of what the system will do, without containing how it will be done.

It is divided into two main parts[24]:

- **Functional requirements**: Describes what the software has to do; they are called product features.
- **Non-Functional requirements**: They are often related to the quality of software and how the software will work, such as reliability, availability, usability, flexibility.

The objective of this phase is to provide solutions for stated problems in the form of specifications to meet users' and clients' needs.

There are many metrics related to this phase which the developer can depend on to verify the output of the phase[25]:

- **Relevant experience of specification and documentation staff**: It is related to the experience and skills of the team members for executing this project during the requirements and specification phase.
- **Quality of documentation inspected**: It is related to the quality of the requirements collected from clients.
- **The regularity of specification and documentation reviewers**: It is related to whether all the requirements documentation have been reviewed.

- **Standard procedures followed**: It is related to the effectiveness of the review procedure.
- **Specification defects discovered in the review**: It is related to whether the defects density of specification reviews is on the high side.
- **Requirements stability**: It is related to how stable the requirements in the projectare.

2.1.2 Design Phase

It is considered as the second main step for software developers; its output is a document called SDD, which refers to a Software DesignDocument, which contains the description of how the system requirements will be implemented.

The design describes the final system and process by which it is developed. During this phase, the software is designed to meet the requirements collected and identified in the previous phase. It is the most creative and challenging phase; it organizes the software module in a pattern that is easy to be developed and allows developers to deal with the size and complexity of programs.

It is divided into two main parts[24]:

- **Conceptual design**: Describes the data source and destination in the system and how the system will look to users.
- **Technical design**: It mainly translates the customer's problem and requirements into solutions, and it describes the hardware configuration, software needs, communication interfaces.

Some of the metrics which can be extracted from the design phase and the developers can depend on to improve the software design documents, and which are important to be used in the measurement of software defect density[25]:

- **Relevant development staff experience**: It is related to the experience and skill set of the team members for executing this project during the design phase.
- **Defined processes followed**: It is related to the review effectiveness in the project for this phase.

- **Development staff motivation**: It is related to the motivation levels of team members who execute this project during the design phase.

2.2 SOFTWARE RELIABILITY

The production of reliable software is a crucial operation, especially those projects which have a vital role in the business and publicsectors. Human is affected in certain waysby the presence of software. All of the life domains, suchas medical, education, transportation, andentertainment, directly or indirectly depend onsoftware. Reliability is an attribute of quality thatmust be considered in most safety-critical systems.

The reliability is defined by IEEE as:"The ability of a system or a component to perform its required functions under stated conditions for a specified period of time."For project managers and software developers, reliability means that software performs its functions correctly without failures, so they consider the fixed bugs in the testing phase. During that, it is necessary to assure the reliability and develop robust, high-quality software with free defects reliability through all of the stages of the software lifecycle. That means the reliability of the delivered product is related to the quality of all of the processes and software development, the requirements documentation, the code, test plans, and testing[26].

It is important to show that software failure rate differs from hardware failure rate as shown inFigure 2.2; when the component is tested for the first time, the initial fault count is high, but then it decreases as the faulty components are identified and removed or edited to be stable. Thecomponent will be in the useful life phase when few if any faults are found. As the component physically wears out, the fault rate starts to increase.

To get the highest reliability, the developers have to focus on comprehensive requirements and a comprehensive testing plan and ensuring all requirements are tested. The focusing also must be on the maintainability of the software since there will be a "useful life" phase where sustaining engineering will be needed. There are some steps that must be done to avoid software error, which are[26]:

1. Ensure the requirements are clearly and accurately defined and meet the final product functionality.

2. Ensure the implementation way can easily support sustaining engineering without arising additional errors.

3. An overall testing must be done to verify that all functionalities stated in the requirements are included.

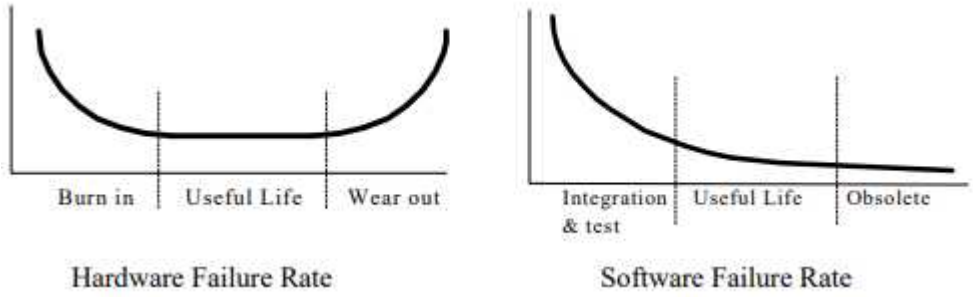

Figure 2.2: Failure Rates

2.2.1 Reliability importance

Computerization is playing a very important role in our life. Most of the tools which human use daily that hadanalog and mechanical parts were replaced with digital devices, CPU's and software such as Dishwashers, TV's, Microwave Ovens, AC's, and others. So,the human cannot dispense with these tools, which depends on all of their life aspects.

In the last two decades, digitization is used widely, and it comes instead of analog methodology. So, the use of software increased related to the use of digitization, which depends on software directly; therefore, the software must be more reliable and can perform its job as it is required without failures. So, software developers consider this issue as the main part ofthe software development life cycle

Increasing development costs have putpressure to quantify software quality and to measure and control the level of quality delivered. There are many software quality factors, but Software Reliability is the most important and most measurable aspect of software quality[27].

2.2.2 Software Reliability as a Quality Metric

There are many various models that were created to find the software quality, but in most models, reliability is oneof the criteria attributes that is incorporated. ISO defines some qualitycharacteristics; one of them is reliability. IEEE states,"A software reliabilitymanagement program requires the establishment of a balanced set of user quality objectives, andidentification of intermediate quality objectives that will assist in achieving the user qualityobjectives." Since reliability is an attribute of quality, it can be concluded that softwarereliability depends on high-quality software[30].

To build high-reliability software;developers should depend on the applying ofthe quality attributes at each phaseof the development life cycle with ensuring there are no errors, especially in the early lifecycle phases, and they should use themetrics related to reliability at each development phase to measure applicable qualityattributes, [26], which are shown in Figure 2.3.

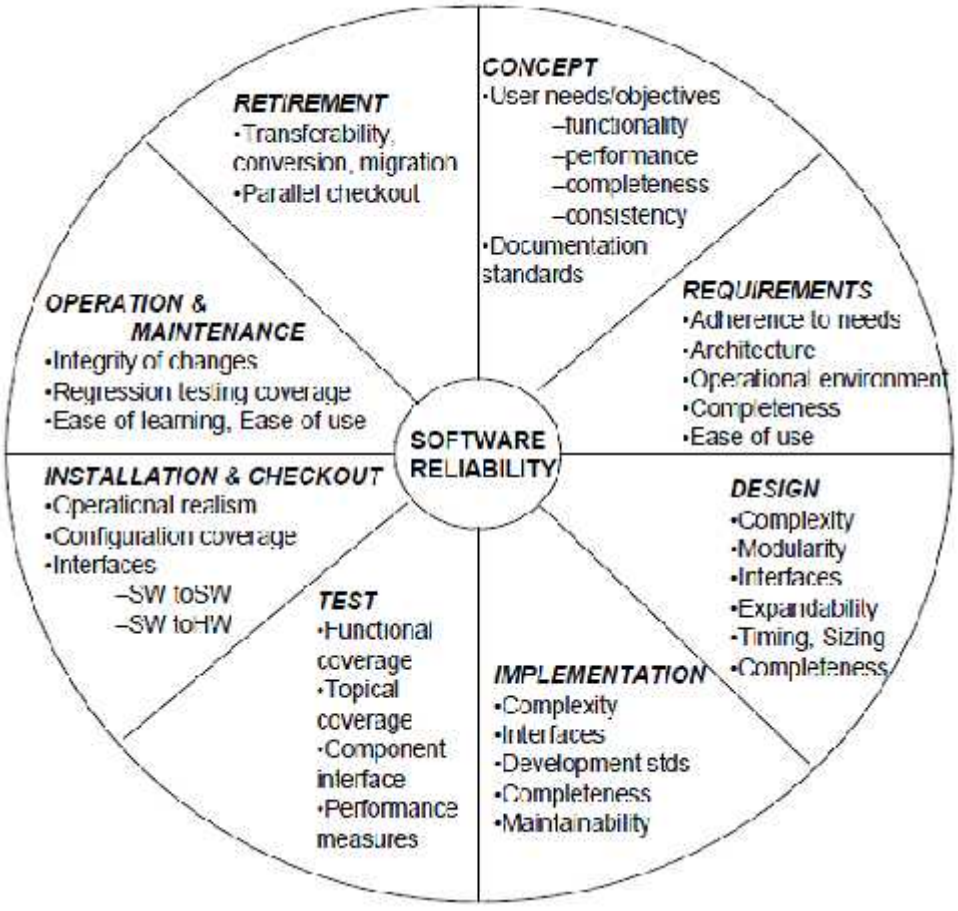

Figure 2.3: Quality Metrics Impacting Reliability

Chapter 3: Literature Review

Researchers have proposed many different models to address the problem of software reliability and to predict the defects count that might be arisen in the software.

Most of those different models are proposed to predict the software reliability at the implementation and testing stages of the software development life cycle, which is considered to be late and cost prediction for project managers to regather the requirements or redesign them because the source of defects might be from requirements or design.

The authors have reviewed many papers related to the software reliability that had been proposed to get defect density of software. The authors classifiedthe reviewed papers based on papers proposed work or the goal of the research.The authors first made a category for the papers that provided the relations between software metrics and software reliability.Then the authors made another category for the papers that proposed models to address the software reliability and got results in this field.

After research and deep inspection, the authors found that the collected models from the second category can be classified into three approaches depending on their methodologiesto quantify the errors density shown inTable 3-1[4][11][12][13][14][15][16][18][21][22].

1. **Fuzzification models**: these models use Fuzzy logic, which has logical values and real numbers between 0 and 1 to determine the value of its variables.
2. **Network algorithms models**: these models use networking algorithms such as Bayesian and Neural algorithms and apply principles of these algorithms using Component Dependency Graph (CDG) as an input to the model to predict the defect density of software.

3. **Object-Oriented models**: these models depend on the structure and analysis of object-oriented unified modeling language (UML) and the design classes to predict the defect density of software.

Table 3-1: Models classification based on the methodology

No	Model classifications	Model examples
1	FuzzificationModels.	Phase-wise model [4], SRQF model [15], Yadav [16], Pandey and Goyal model [22]
2	Network algorithms Models.	ERSA model [18], Fenton model [21]
3	Object-oriented models	SBRA model [20]

3.1 METRICS AFFECT THE SOFTWARE RELIABILITY

During the review of the literature[10][11][12][13][14][15]; it is observed that the software metrics play animportant role in fault prediction, through that the metrics for which the SDLC phase belong to, and the number of SDLC stages used in the model.

Many papers and researches have aimed to define the relations between software metrics and software reliability. Software reliability is highly related to the quality of all of the processes and products of software development, starting from the requirementsdocumentation, the coding, test plans, and testing, so; it depends on many metrics are extracted from all stages of SDLC [26].

The authors of [26] had identified the most highly related metrics to the software reliability in the whole process of the SDLC, as shown inTable 3-2, and they illustrate how each metric affects the reliability.

Table 3-2: Metrics affect software reliability

#	SDLC Stage	Metric	Description
1	Requirements Reliability Metrics	Lines of Text	Physical lines of text as a measure of the size
2		Imperatives	Words and phrases that command that something must be done or provided

3		Continuances	Phrases that follow an imperative and introduce the specification of requirements at a lower level
4		Directives	References provided to figures, tables, or notes
5		Weak Phrases	Clauses that are apt to cause uncertainty and leave room for multipleinterpretation measure of ambiguity
6		Incomplete	Statements within the document that have TBD (To be Determined) or TBS(To Be Supplied)
7		Options	Words that seem to give the developer latitude in satisfying the specifications but can beambiguous.
8	Design and Code Reliability Metrics	Cyclomatic Complexity	Computed as the number of linearly independent test paths
9		Code Size	Total lines of code, counting all lines; non-comment non-blank
10		(WMC)	Weighted methods per class – related to Object-oriented
11		RFC	Response for a Class – related to Object-oriented
12		CBO	Coupling Between Objects – related to Object-oriented
13		DIT	Depth in Tree – related to Object-oriented
14		NOC	Number of Children – related to Object-oriented
15	Testing Reliability Metrics	The total set of test cases	Each requirement must be tested at least once

3.2 FUZZIFICATION MODELS

Fuzzificationmodels use Fuzzy logic, which has logical values (VH, H, M, L, VL), and real numbers between 0 and 1 to determine the value of its variables using fuzzification algorithms and equations.

3.2.1 Phase-wise Model

This model is proposed to predictsoftware defects density indicator at each phase of SDLC, requirement, design, implementation, and testing, which is considered as a later stage reliability model. The defect density indicator in the requirement analysis, design, coding, and testingstage is predicted using nine software metrics of these four stages.The defect densityindicator metric,which ispredicted at the end of each stage, is considered as an input to the next stage. Software metrics areassessed in linguistic terms (VH: Very High, H: High, M:

Medium, L: Low, VL: Very Low), and a fuzzification algorithm was usedto develop the model, which is shown inFigure 3.1. The proposed model has been applied totwenty software projects[4].

The steps of this model are:

1- Select software metrics.

2- Define the membership function of input and output metrics.

3- Design Fuzzy rules.

4- Perform fuzzy inference and defuzzification.

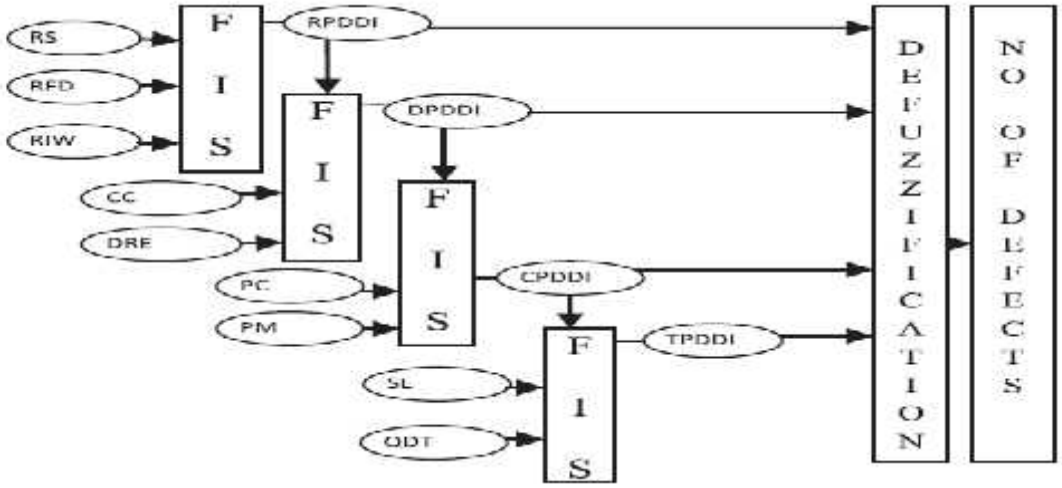

Figure 3.1: Phase-wise model

The input metrics which this model depends on are shown inTable 3-3.

The output of each phase is considered depending on fuzzy rules, which have linguistic terms and fuzzy ranges between 0 and 1.

The fuzzy rule is defined in the form of an IF-THEN conditionalstatement. IF part of the rule is known as an antecedent, andTHEN part is consequent.

Table 3-3: Models classification based on the methodology

No	Requirement Phase	Design Phase	Coding Phase	Testing Phase
1	Requirement Stability (RS)	Cyclomatic Complexity (CC)	Programmer Capability (PC)	Staff Experience (SE)

2	Requirement Fault Density (RFD)	Design Review Effectiveness (DRE)	Process Maturity (PM)	Quality of Documented Test Cases (QDT)
3	Regularity-Inspection, and Walk-through the specification & documentation (RIW)			

3.2.2 Yadav Model

This model is proposed to predict the number of residual defects before the designingstage, which is considered as an early stage reliability model. Software metrics are available in the requirements analysis stage, in which the top four metrics are:requirement defect density, requirement stability, error distribution, and reviews, inspections, and walkthroughs. The third and fourth metrics depend on the experience of the Requirement stage team members. Therefore, these three metrics of the requirements analysis stage, i.e., Experience of Requirement Team (ERT), Requirement Defect Density (RDD), and RequirementStability (RS), are considered as input to the proposed Fuzzy model. ERT metric measures the relevant experience and skill set of team members for executing the project during the requirements analysis stage of SDLC. RDD metric measures the percentage of defective requirements in the requirements specification documents, which are obtained through regular reviews. Requirement stability (RS) metric measures the stability of the requirements in the projects.

Three metrics of the requirements analysis stage are used as inputs to the fuzzy rule-based inference system, which gives an aggregated fuzzy set of the total number of residual defects as output. The total number of residual defects obtained through defuzzification further helps to decide the testing strategy during the testing stage of the software. The model is shown in Figure 3.2.

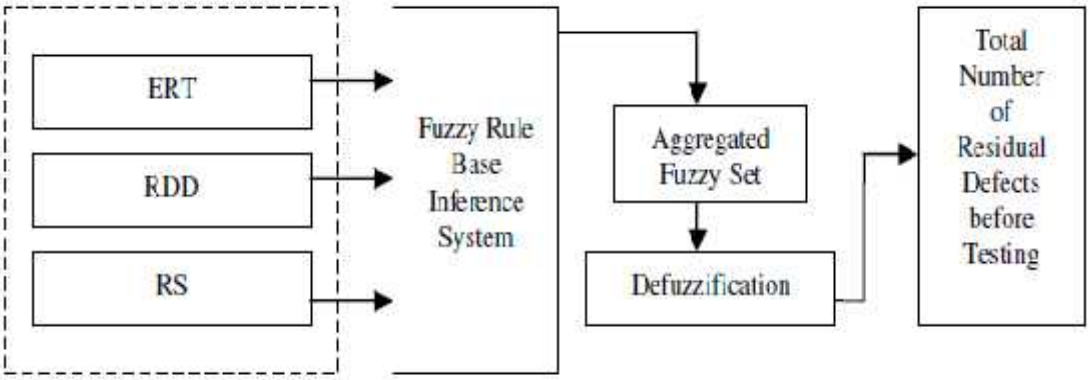

Figure 3.2: Yadav model

3.2.3 Pandey and Goyal Model

This model predicts the number of faults at the end of each SDLCstage using related software metrics and the level of the developer's Capability Maturity Model (CMM) with reliability. The model uses 11 metrics within three SDLC stages, so; it is considered as a later stage reliability model.

This model used a fuzzy inference system - shown in Figure 3.3- to get the number of faults in the system.The used fuzzy rules depending on the expert's

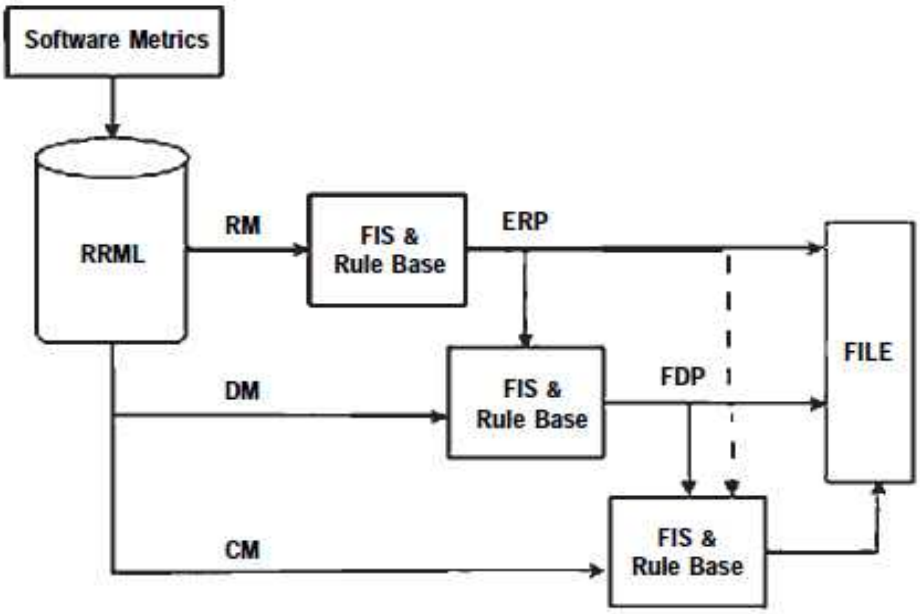

Figure 3.3: Pandey and Goyal Model

opinion.

The metrics used by this model are: Requirements Change Request (RCR), Regularity-Inspection, and Walk-through the specification &documentation(RIW), and Process Maturity (PM) as input to the requirements stage, design defect density (DDD), fault days number (FDN), and data flow complexity (DC) have been considered as inputs to the design stage.Code defect density (CDD), and Cyclomatic Complexity (CC) as inputstothe coding stage. The outputs of the model will be the number of faults at the end of the Requirements Phase (FRP), the number of Faults at the end of the Design Phase (FDP), and the number of Faults at the end of the Coding Phase (FCP). This model uses a fuzzy logic approach to predict the number of faults in the software [22].

3.3 NETWORK ALGORITHM MODELS

Network Models use some network algorithms such as Bayesian Belief Network, Artificial Neural Network, and others to predict software reliability.

3.3.1 ESRA

This model is proposed depending on Bayesian Belief Network to predict the reliability of object-oriented software; it works in three phases during the first two stages of SDLC, so; it is considered as an early stage reliability model.

In the first phase, the reliabilities of the classes are predicted from the design metrics obtained from the UML model of the software, whichis known as class reliability prediction, which is shown inFigure 3.4.The second phase, which is known as use case reliability prediction,contains a prediction of use case reliabilities depending on the operational profile and predicted values of the classes in phase 1. In the last phase, which is known as system reliability prediction, the system reliability is predicted from reliability values of the use cases and the operational profile.

The first phase uses 14 metrics to calculate the reliability value of classes, which are illustrated inTable 3-4. This model uses multiple equations to calculate the reliability of each phase, then uses these values to calculate the reliability value[18].

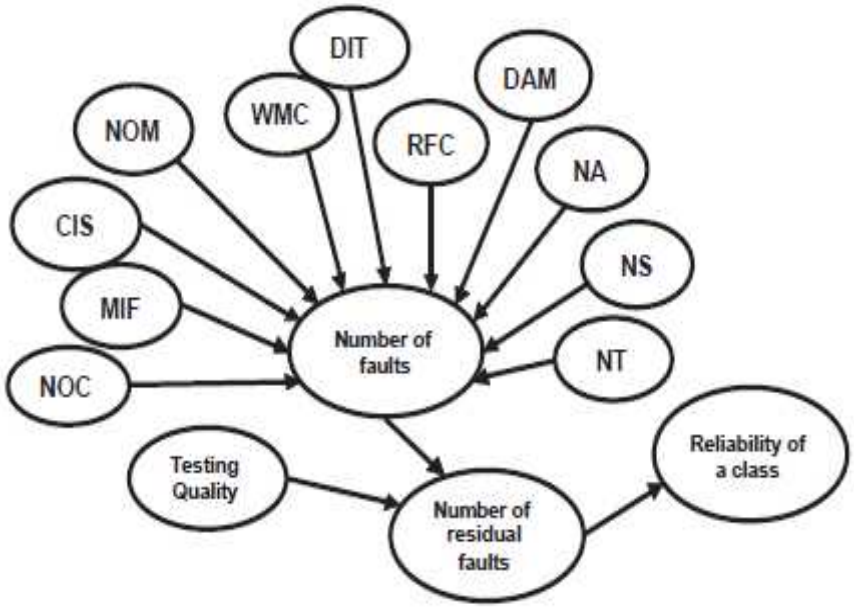

Figure 3.4: A BBN model for predicting the reliability of a class

Table 3-4: Description of the object-oriented metrics

Name of the metric	Description
Weighted Methods per Class (WMC)	This metric is defined as the sum of the complexities of all methods in a given class
Coupling Between Objects (CBO)	It counts the number of classes whose attributes or methods are used by the given class and the number of classes which use the attributes or the methods of the given class
Depth of Inheritance Tree (DIT)	It is defined as the length of the longest path from a given class to the root class in the inheritance hierarchy
Lack of Cohesion of Methods (LCOM)	It counts the sets of methods in a class that are not related through the sharing of some local variables of the class
Number of Children (NOC)	It is the count of the total number of immediate child classes of a given class
Response for Class (RFC)	It measures the number of methods (NOM) and constructors that can be invoked as a result of a message sent to an object of the class

Method Inheritance Factor (MIF)	It represents the ratio of inherited methods to the total NOM in a class
Attribute Inheritance Factor (AIF)	It represents the ratio of inherited attributes to the total number of attributes in a class
Data Access Metric (DAM)	This is represented by the ratio of private and protected attributes to the total number of attributes declared in a class
Class Interface Size (CIS)	This is a count of the total number of public methods of a class
Direct Class Coupling (DCC)	A count of classes that accepts instances of a given class as a parameter and the classes that include attributes of the given class type
Number of Transitions (NT)	This metric measures the total NT in the state diagram of a given class
Number of States (NS)	This metric computes the total NS in the UML state diagram of a given class
Number of Activities (NA)	This metric counts the total number of events in the UML state diagram
Number of Entry Actions (NEntryA)	The total number of the actions performed each time a state is entered

The problem with this model is that; the model could be used with the software that analyzed using UML classes and use-case methodology. So, it is special to deal with object-oriented software.

3.3.2 Fenton Model

This model is proposed to develop a causal model (Bayesian net) for predicting the number of defects that are likely to be found during independent testing or operational usage. Its method does notrequire detailed domain knowledge. The model incorporates a set of quantitative and qualitative factors describing a project

and its development process, which are inputs to the model. The researchers of the Fenton model [21] have performed various sensitivity analyses. They found that the most influential qualitative factors are project complexity and scale of distributed communication. Although none of the individual process factors appear to be highly influential on their own, the aggregation of such factors as process effectivenessis highly influential[21].

The proposed model used many metrics collected through the whole process of the software development life cycle, which are shown inTable 3-5, so it is considered as a later stage reliability model.

The proposed model results are more accurate in medium and large software than the small projects, as shown inFigure 3.5.

Table 3-5: Metrics used in Fenton Model

#	Stage	Metric	Description
1	Requirement stage	Relevant experience of spec and doc staff	How would you rate the experience and skill set of your team members for executing this project during the requirements and specifications phase?
2		Quality of documentation inspected	How would you rate the quality of the requirements given by the client or other groups?
3		The regularity of spec and doc reviews	Have all the Requirements, Design Documents, and Test Specifications been reviewed in the project?
4		Standard procedures followed	In your opinion, how effective was the review procedure?
5		Review process effectiveness	What was the review effectiveness in the project for the requirements phase?
6		Spec defects discovered in the review	In your opinion, is the defect density of spec reviews on the high side?

7		Requirements stability	How stable were the requirements in your project?
8	New functionality	The complexity of new functionality	What was the complexity of the new development or new features that happened in your project?
9		The scale of new functionality implemented	How large was the extent of working on new functionality rather than just enhancing the older functionalities in your project?
10		Total no. of inputs and outputs	For your product domain, would you rate the total no of outputs/inputs (newly developed/ enhanced) as high?
11	Design and development	Relevant development staff experience	How would you rate the experience and skill set of your team members for executing this project during the design and development phase?
12		Programmer capability	On average, how would you assess the quality of code produced by the team members?
13		Defined processes followed	What was the review effectiveness in the project for the Design and Development phase?
14		Development of staff motivation	What is your opinion about the motivation levels of your team members?
15	Testing and rework	Testing process well defined	How effective was the testing process adopted by your project?
16		Staff experience unit test	What was the level of software test competence of those performing the unit test?
17		Staff experience independent test	What was the level of software test competence of those performing the unit test?
18		Quality of documented test cases	What was the extent of the defects that were found using formal testing against the intuitive/random testing?

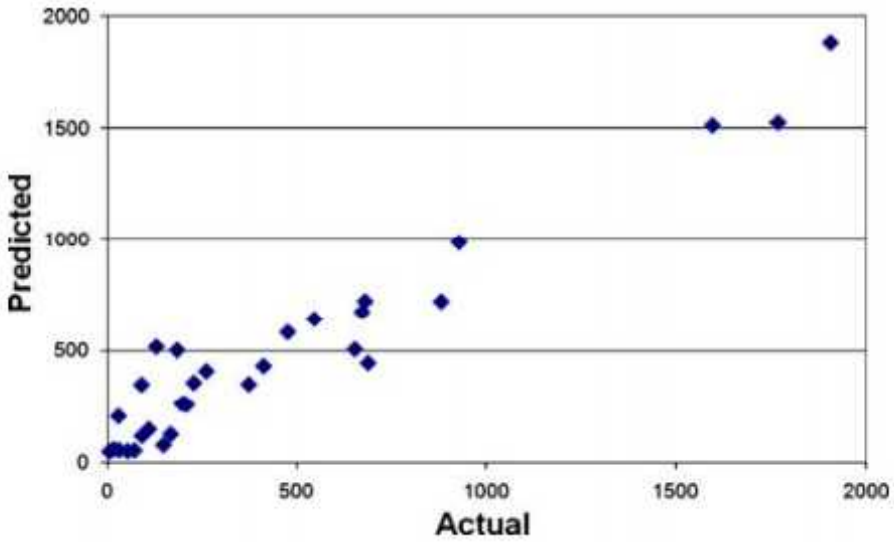

Figure 3.5: Predicted and actual values

Chapter 4: Methodology

The proposed work aims to make a new model that predicts the software reliability within the first two stages of the software development life cycle, which are requirements and design stages. This work will add value for predicting the software reliability and assist project managers in making the right decisions at the right time, and allow them to refine the requirements or design if they consist of some troubles.

4.1 MODELDESIGN

The architecture of the proposed model is shown inFigure 4.1. The proposed model depends on seven metrics, which are collected from the first two stages of SDLC, which affect the software reliability directly. The metrics collected from the requirements stageare Quality of Documentation Inspected (QDI), Regularity-Inspection, and Walk-through the specification &documentation (RIW), Spec Defects Discovered in Review (RFD), Requirements Stability (RS) as explained inTable 4-1. The metrics collected from the design stage are Complexity of new functionality (CC), Scale of New Functionality implemented (SNF), Defined processes followed (DRE) as explained in Table 4-2.

The authors have chosen these metrics because these metrics are the most influencing the software reliability depending on Li and Smidts [34]software reliability metrics ranking.

Table 4-1: Requirement metrics used in the proposed model

#	Metric	Description	
1	QDI	Quality of Documentation Inspected	How would you rate the quality of the requirements documentation?
2	RIW	Regularity, Inspection, and Walk-through the specification	Have all the Requirements, Design Documents, and Test Specifications

		&documentation	been reviewed in the project?
3	RFD	Requirement Fault Density or Spec Defects Discovered in Review	In your opinion, is the defect density of spec reviews on the high side?
4	RS	Requirements Stability	How stable were the requirements in your project?

Table 4-2: Design metrics used in the proposed model

#	Metric	Description	
1	CC	The complexity of new functionality	What was the complexity of the new development or new features that happened in your project?
2	SNF	Scale of New functionality implemented	How large was the extent of working on new functionality rather than just enhancing the older functionalities in your project?
3	DRE	Design Review Effectiveness	What was the review effectiveness in the project for the Design phase?

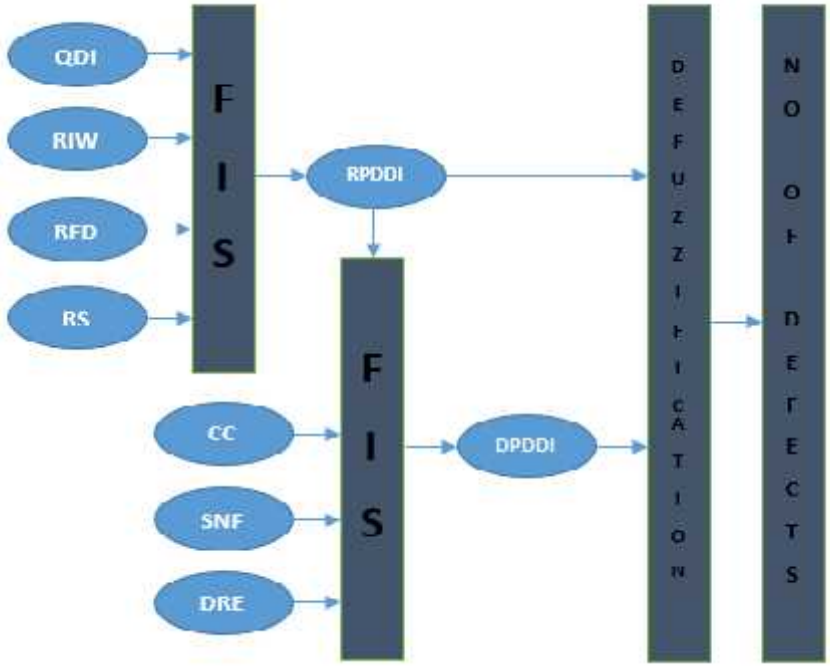

Figure 4.1: The proposed model architecture

The steps of the proposed model:

I. Selection of software metrics.

II. Define membership function of input and output metrics.

III. Design fuzzy rules.

IV. Perform fuzzy inference and defuzzification.

Here, the authors will explain the steps which are involved in the proposed model:

4.1.1 Selection of software metrics

Many software reliability prediction models have beenproposed since 2000 using softwaremetrics. These models helped developers and software managers to develop reliable software.But, someof them have used a lot of software metrics to predict the software reliability, which made the process very complex and more expensive processing cost; these models have many useless metrics which doesn't affect the software reliability and less important to be used there[4].

But instead, there are software reliability prediction models have been proposed using a reasonable number of software metrics, but these models have been

built to be used within the whole software development life cycle, including the coding stage, which is considered as a late-stage to allow the developers and software managers to make the right decisions which will be time and effort costly[32][33][34].

Also, there are some models that work in the first stage of SDLC to predict the software reliability, these models assist the decisions makers in making the decisions at the early stageof the software life cycle, but these predicting models' results are not accurate as needed, because in the first stage there is no enough information to be used.

So, the decision-makers need a predicting model, which can work through the early stages of SDLC and give accurate results to be considered in their decisions.

According to that, the authors investigate how the software metrics affect the software reliability to select the right metrics,which could improve the predictionaccuracy depending on Li and Smidts [34], who had ranked the top thirty metrics with respect to their ability to affect the software reliability. The selected metrics are shown inTable 4-1 and Table 4-2 and explained as follows:

4.1.1.1. Requirement phase software metrics

1- Quality of Documentation Inspected (QDI): The purpose of documentation inspection is that to detect detailed errors in the requirements; if the quality of documentation is high, this leads to low defects, which means high reliability.

2- Regularity, Inspection, and Walk-through the specification &documentation (RIW):The purpose of this metric is to carry out a technical analysis of product components or documentation to find mismatches between the specification and client requirements; high RIW leads to low defects.

3- Requirement Fault Density (RFD): The purpose of this metric is to measure the faulty of requirements and specification documents; high RFD leads to high defects.

4- Requirement Stability (RS): Requirement stability is proportional to theinverse of requirements changing. The main reason forvarying requirements is that the stakeholders have no clearrequirements.Requirement changescould be made at any timeduring the software development. Requirementschanges adversely affect the cost, quality, and reliability of the software, which is under development; high RS leads to low defects.

4.1.1.2. Design phase software metrics

1- The complexity of new functionality (CC): This metric measures the complexity of software depending on the decision points. It is usually to estimate the number of remaining software defects[36]. High CC leads to high defects.

2- The scale of New functionality implemented (SNF): This metric measures the size of the expansion of working on new functionality rather than just enhancing the older functionalities in the project; high SNF leads to low defects.

3- Design Review Effectiveness (DRE): The purpose of this metric is to check whether the design documents meet the stakeholder's requirements or to find whether design documents need to be modified; high DRE leads to low defects.

Table 4-3:Requirement analysis phase software metrics

Requirement analysis phase software metrics		Fuzzy range	Linguistic terms
Input metrics	Quality of Documentation Inspected (QDI)	[0 - 1]	[L, M, H]
	Regularity, Inspection, and Walk-through (RIW)	[0 - 1]	[L, M, H]
	Requirement Fault Density (RFD)	[0 - 1]	[L, M, H]
	Requirement Stability (RS)	[0 - 1]	[L, M, H]
Output metric	Requirement Phase Defect Density Indicator (RPDDI)	[0 - 1]	[VL, L, M, H, VH]

Table 4-4: Design analysis phase software metrics

Design analysis phase software metrics		Fuzzy range	Linguistic terms
Input metrics	Complexity of functionality (CC)	[0 - 1]	[L, M, H]
	Scale of New functionality (SNF)	[0 - 1]	[L, M, H]
	Design Review Effectiveness (DRE)	[0 - 1]	[L, M, H]
	Requirement Phase Defect Density Indicator (RPDDI)	[0 - 1]	[VL, L, M, H, VH]
Output metric	Design Phase Defect Density Indicator (DPDDI)	[0 - 1]	[VL, L, M, H, VH]

4.1.2 Define membership function of input and output metrics

In the fuzzy logic world, we have many ways to perform membership functions.

This operation can be intuitive, or it can be based on some mathematical or logical process. In an intuitive way, membership function is derived from the ability of humans to develop membership functions through their own innate intelligence and understanding. So, there are no standard guidelines or rules that can be considered to create membership functions. Another problem that makes building membership functionsessentialis the lack of consensus regarding the definition and interpretationof membership functions. Membership functions for all input and output metrics are taken into account in the proposed modelshould be determined by domain experts. Membership development job with the help of expert knowledge is one of the essentials steps in designing a problem to be solved by a fuzzy settheory[16].

Membership functions can have a variety of types, such as triangular, trapezoidal, polygonal, and others. Triangular andtrapezoidal types providea suitable representation of domainexpert knowledge as it simplifies the calculation process.Triangular and trapezoidal membership functions are lineartypes, and they are more suitable when the membership functions of a fuzzy set are not known.

Therefore, in the proposed model, triangular and trapezoidal memberships are used to represent the linguistic states, depending on experts opinions – who are three experts having more than 5 years of experience in project management - who took

part in the development of these functions,Figure 4.2 to Figure 4.9, which have been taken as snapshots from the MATLAB tool during the model building.

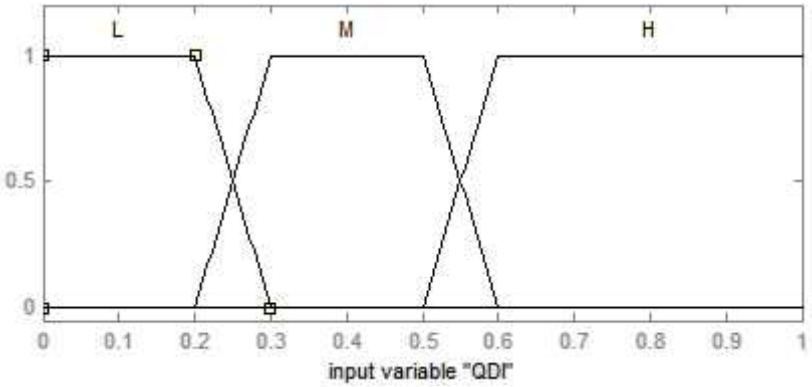

Figure 4.2: Quality of Documentation Inspected

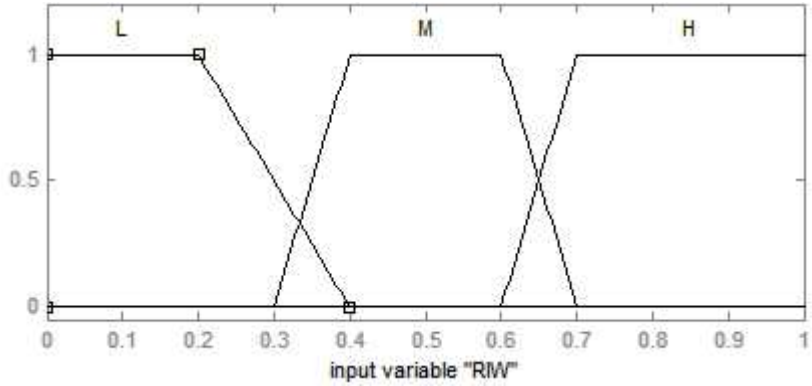

Figure 4.3: Regularity, Inspection and Walk-through

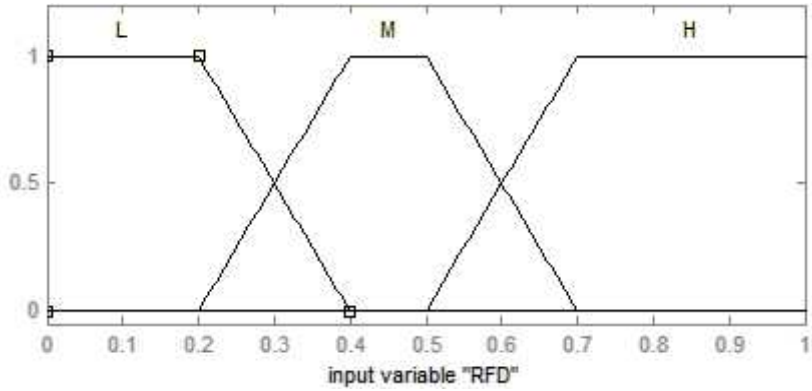

Figure 4.4: Requirement Fault Density

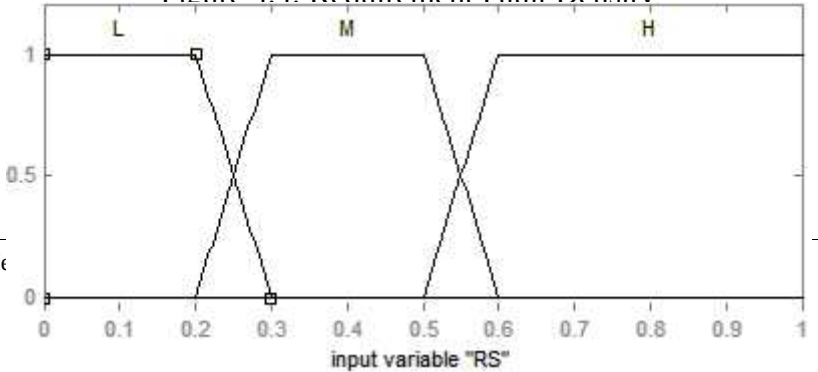

Figure 4.5: Requirement Stability

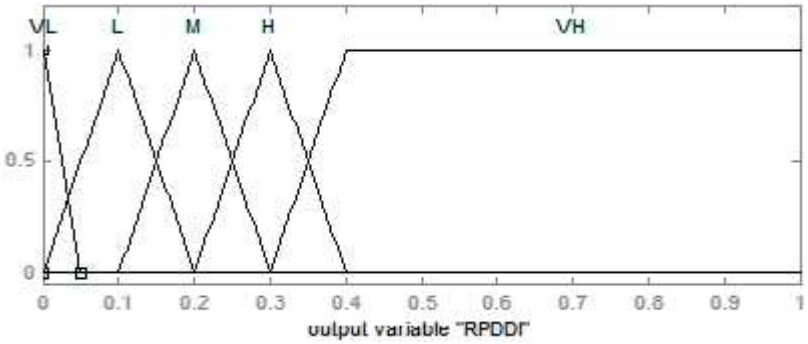

Figure 4.6: Requirement Phase Defect Density Indicator

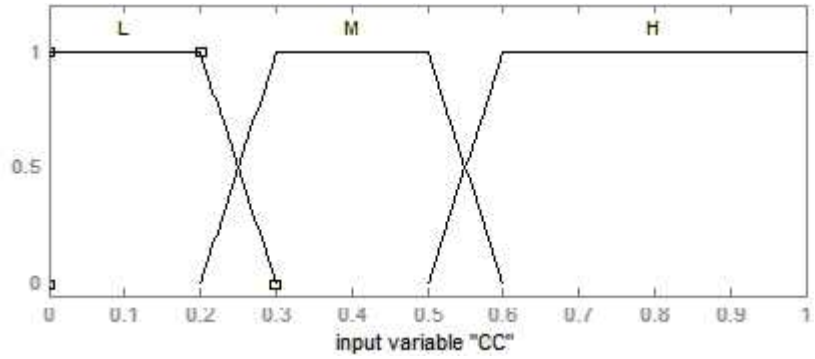

Figure 4.7: Complexity of functionality

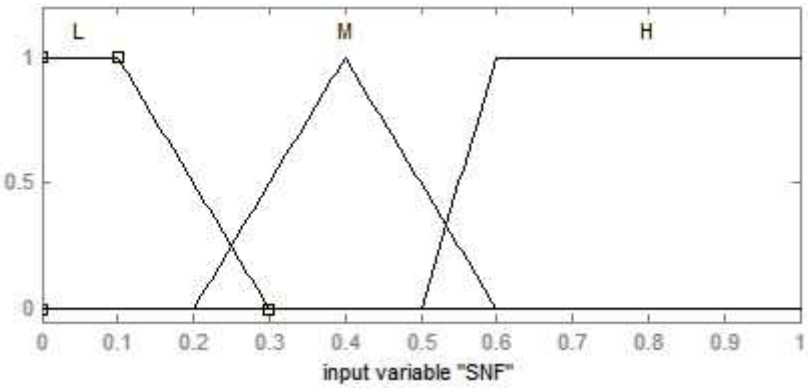

Figure 4.8: Scale of New functionality

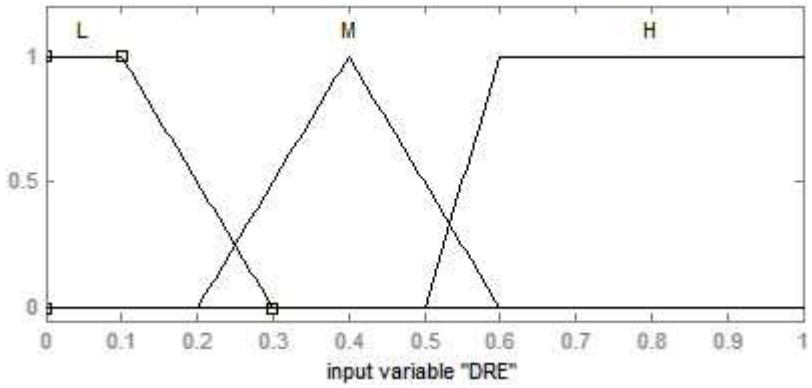

Figure 4.9: Design Review Effectiveness

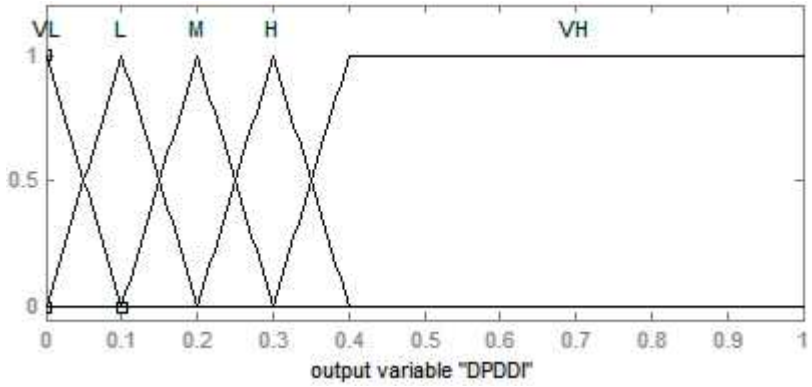

Figure 4.10: Design Phase Defect Density Indicator

4.1.3 Design fuzzy rules

Fuzzy rules design is not an easy work, especially for real-world problems. The fuzzy rule base can be decided using different approaches such as domain experts, automated generation, historical data analysis, and knowledge of existing literature. The most efficient approach mostly used is the expert's domain to enable the design of such a system with more human-like reasoning, especially with the fuzzy IF-THEN rules. IF part of the rule is called Antecedents, and the THEN part is called Consequent[38].

In the proposed model, the fuzzy rules set, which are used to predict the number of defects in the software, are defined with the help of domain experts who are three experts having more than five years in software management. The proposed

model has two phases; the first phase has four input metrics; each metriccontains three linguistic states Low (L), Medium (M), High (H)Table 4-3, so, the number of rules in the first phaseis: 3^4 = 81 rules, Eq. (4.1),Table 4-5, and the second phase has four input metrics, one of them has five linguistic states Very Low(VL), Low (L), Medium (M), High(H), Very High (VH) and the other metrics have three linguistic states,Table 4-4, so, the number of rules in the second phase is: $3^3 * 5$ = 135 rules,Eq. (4.1),Table 4-6.

$$Number\ of\ Rules = (Number\ of\ Linguistic\ states)^{(Number\ of\ Metrics)} [38] \qquad (4.1)$$

Table 4-5: Requirement phase fuzzy rules

Rule No	Fuzzy rule
1	If QDI is L and RIW is L and RFD is L, and RS is L, Then RPDDI is H
2	If QDI is L and RIW is L and RFD is L, and RS is H, Then RPDDI is H
……...	……
80	If QDI is H and RIW is H and RFD is H, and RS is M, Then RPDDI is L
81	If QDI is H and RIW is H and RFD is H, and RS is H, Then RPDDI is L

Table 4-6: Design phase fuzzy rules

Rule No	Fuzzy rule
1	If RPDDI is VL and CC is L and SNF is L, and DRE is L, Then DPDDI is M
2	If RPDDI is VL and CC is L and SNF is L, and DRE is M, Then DPDDI is M
……...	……
134	If RPDDI is VH and CC is H and SNF is H, and DRE is M, Then DPDDI is H
135	If RPDDI is VH and CC is H and SNF is H, and DRE is M, Then DPDDI is M

4.2 METHODOLOGY EXPLANATION

The proposed model aims to find the number of defects in the software within the early stages of the software development life cycle against the models that find the number of defects at the end of the testing stage. This enhancement will add a positive value in this field to develop more reliable software and help the project managers to make the right decisions at the right time.

In order to validate the proposed model, the authors will present a case study using project number 1 from the used NASA dataset[21] to explain the steps of the model and how it works

Then, the authors will compare the proposed model withother reliability models that quantify the software defect density at the early stages of SDLC by showing the final results of each model.

The evaluation measures have been considered[37]to get the percentage of error of each model compared with real data of the NASA projects to validate the prediction accuracy of the proposed model, which they are:

1- Mean Magnitude of Relative Error (MMRE): MMRE is the mean of absolute percentageerrors. It is a measure of the spread of a variable such as Z, where Z=estimate/actual.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \qquad (4.2)$$

2- Balanced Mean Magnitude of Relative Error (BMMRE).

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{Min\,(y_i, \hat{y}_i)} \qquad (4.3)$$

Where*n* is the number of projects,y_i is the actual value, and $\hat{y}_i$ is the estimated value of a variable of the compared model.

The model which has fewer values of MMRE and BMMRE indicates better accuracy of prediction, so the real project's data has the minimum values which are zeros of MMRE and BMMRE.

Chapter 5: Case Study and Analysis

5.1 INTRODUCTION

This chapter shows the steps of the proposed model implementation by applying the case study using project # 1 from the NASA dataset[25].

The rest of this chapter is organized as follows. Section 5.2 introduces the used dataset, section 5.3 explains how the model has been tested, section 5.4 illustrates the proposed model using the case study.

5.2 DATA SET

In order to validate the proposed model, the authors use the twenty real software projects dataset [25] which were produced by NASA, which are shown inTable 5-1 where the qualitative values of selected software metrics are represented in terms of Very Low (VL), Low (L), Medium (M), High (H), Very High (VH). The dataset contains 31 projects; the authors have used the projects that provide enough information about the used metrics in the proposed model, so it is called twenty real software dataset.

These data set projects developed software embedded in electronics products. Each developed software was integrated with another software; so, it is not stand-alone software. The project managers followed a waterfall life cycle to develop the projects. The overall development of these projects was distributed into many locations in a global organization. The software specification and independent testing were in a different location to the software location[25].

The authors have divided the dataset projects into two parts, the first part, which contains 9 projects, has been used to build the proposed model, and the other, which contains 11 projects,has been used to test the proposed model, as shown in Table 5-2.

Table 5-1: The considered metrics of software projects

Case study #	Project # [25]	Size (KLOC)	QDI	RIW	RFD	RS	CC	SNF	DRE
1	1	6	M	VH	H	L	M	L	M
2	2	0.9	H	VH	H	H	L	VL	M
3	3	53.9	H	VH	VH	H	H	H	VH
4	7	21	M	VH	L	M	L	VL	M
5	8	5.8	M	H	L	H	M	L	M
6	9	2.5	VH	VH	M	VH	L	L	M
7	10	4.8	H	H	M	H	M	L	M
8	11	4.4	M	H	H	H	H	H	H
9	12	19	M	H	M	L	H	H	H
10	13	49.1	M	M	H	L	H	H	H
11	15	154	H	H	VH	VL	H	H	M
12	16	26.7	H	H	H	M	L	VL	M
13	17	33	H	M	H	M	L	VL	M
14	19	87	M	H	H	M	H	H	H
15	20	50	L	M	M	VL	VH	H	VH
16	21	22	H	H	M	M	L	M	VH
17	22	44	L	M	M	L	M	M	VH
18	24	99	M	M	H	L	M	M	H
19	29	11	M	VH	M	VH	M	M	H
20	30	1	H	VH	M	VH	L	L	M

5.3 MODEL TESTING

The authors have tested the proposed model by dividing the dataset projects into two sets, depending on the size of the projects, the first set, which contains nine projects, was used to build the model, and the other set, which contains eleven projects, was used to test the model, as shown in Table 5-2.

Table 5-2: Dataset projects splitting for model building and testing.

#	Project number	
	Used in model building	Used in model testing
1	2	1
2	9	3
3	15	7
4	19	8
5	20	10
6	21	11
7	22	12
8	29	13
9	30	16
10		17
11		24

5.4 MODEL ILLUSTRATION: CASE STUDY

To explain the proposed model, project number #1 from the dataset is considered as a case study. Following are the steps to find the defect density indicator for each software development life cycle stage and the total number of defects for case study 1.

5.4.1 Selection of software metrics

Usually, the fuzzy range of software metrics differs from project to project depending on the lowest and highest values. So, in the proposed model, the considered metrics have various fuzzy ranges; because of that, the authors have represented the fuzzy ranges in the normalized form, which is [0, 1].

The software metrics which are used in the proposed model are shown in Table 4-3andTable 4-4 for both SDLC stages (Requirement and Design), respectively.

5.4.2 Define membership function of input and output variables

The membership functions are usually used to describe the fuzziness of software metrics. To assign membership values to fuzzy variables, we have many approaches, such as rank-ordering, intuitive, inference.

In the proposed model, the intuitive approach is used to assign membership values, which depend on human intelligence and domain expert. The membership functions for all of the input and output metrics are developed using triangular and trapezoidal functions, as explained in Figure 4.2 to Figure 4.10.

5.4.3 Design fuzzy rules

The fuzzy rules which are used in the proposed model are explained in section4.1.3 and shown in Table 4-5 and Table 4-6. The authors briefly explain how the metrics affect defect density:

5.3.3.1. Requirement phase fuzzy rules:

- If QDI is High, the defectdensity will be Low.
- If RIW is High, the defectdensity will be Low.
- If RFD is High, the defectdensity will be High.

- If RS is High, the defectdensity will be Low.

5.3.3.2. Design phase fuzzy rules:

- If CC is High, the defect density will be High.
- If SNF is High, the defect density will be Low.
- If DRE is High, the defect density will be Low.

5.4.4 Perform fuzzification and defuzzification

To get the defect density indicator value at the end of the requirement and design stages, the MATLAB tool is used for fuzzification and defuzzification input metrics values, by applying the proposed fuzzy rules using the membership functions.

Chapter 6: Results and Discussion

6.1 INTRODUCTION

This chapter shows the proposed model results. In order to show the effectiveness of the proposed model, the authors will compare the model results with other models, and to show the accuracy of the proposed model, the MMRE and BMMRE will be calculated for the proposed model and the other compared models. Results are discussed at the end of this chapter.

The rest of this chapter is organized as follows. (section 6.2) explains the prediction results, the model validation is described in (section6.3), and (section 6.4) concludes the model results.

6.2 PREDICTION RESULTS

The metrics used in the proposed model are the most related metrics to the software reliability within the early two stages of SDLC (Requirement and Design), depending on Li and Smidts [34], who had ranked the top thirty metrics which have improved the software reliability.

The proposed model predicts the defect density indicator of the first two stages of SDLC using the software metrics and fuzzy inferencesystem. The prediction results of the twenty real software projects of NASA [25] areshowninTable 6-1; which also contains the actual defects, requirement and design stages defect density indicators, Yadav model[16], Fenton model[21], and the Pandey and Goyal[22] predicted defects which all of them used the same dataset to predict the defect density. The total number of defects is calculated at the end of the design stage using the two stages output metrics, which are RPDDI and DPDDI multiplied with the software code size which will be explained next.

Many researchers considered that there is an approximate relationship between the defect density count and the size of software [4][16][21]. So, we can use the software size to calculate the residual defects.

Table 6-1shows the RPDDI, DPDDI, results of the proposed model, and the results of other models that predict at the early stages of SDLC.

Depending on that; the total number of software defect density is calculated as:

Case study number 1:

RPDDI (Requirement Phase Defect Density Indicator): 0.2, from Table 6-1.

DPDDI (Design Phase Defect Density Indicator): 0.16, from Table 6-1.

LOC: 6000 line, from Table 5-1

Total Number of defects = RPDDI * DPDDI * LOC = 0.2 * 0.16 * 6000 = 192.

Table 6-1: Actual and predicted defects number

Case study #	RPDDI	DPDDI	Actual defects	Defects Predicted by			
				Proposed model	Yadav model	Fenton model	Pandey and Goyal
1	0.2	0.16	148	192	88	75	56
3	0.0867	0.09	209	421	261	254	211
4	0.0752	0.13	204	205	204	262	113
5	0.15	0.06	53	52	56	48	54
7	0.1	0.06	29	29	70	203	26
8	0.186	0.09	71	74	64	51	41
9	0.119	0.05	90	113	92	347	176
10	0.1	0.04	129	196	476	516	337
12	0.1	0.04	109	107	130	145	128
13	0.15	0.14	688	693	589	444	136
18	0.224	0.07	1597	1552	1440	1514	-

Table 6-2: Actual and predicted defects with differences

		Proposed Model		Yadav model		Fenton model		Pandey and Goyal	
Project size	Actual defect	Predicted	Diff	Predicted	Diff	Predicted	Diff	Predicted	Diff
4.4	71	74	3	64	-7	51	-20	41	-30
4.8	29	29	0	70	41	203	174	26	-3
5.8	53	52	-1	56	3	48	-5	54	1
6	148	192	44	88	-60	75	-73	56	-92
19	90	113	23	92	2	347	257	176	86
21	204	205	1	204	0	262	58	113	-91
26.7	109	107	-2	130	21	145	36	128	19
33	688	693	5	589	-99	444	-244	136	-552
49.1	129	196	67	476	347	516	387	337	208
53.9	209	421	212	261	52	254	45	211	2
99	1597	1552	-45	1440	-157	1514	-83	-	-

The proposed model is used in the early stages of SDLC, where the software size is not known, so; the project managers should use one of the estimation tools to get a rough estimation of the software size to be able to use this model.

6.3 MODEL VALIDATION

To evaluate the proposed model, the authors use the most suggested and common measures, which are MMRE and BMMRE, to measure the spread of the error, then the authors discuss and validate the results with the other compared models. We tried to validate the proposed model using one of the statistical tests such as T-test or ANOVA test, but in this case, there is no need to validate with them, because we need to compare these models to get the accuracy of the models and measure the spread of the error which could be measured by MMRE & BMMRE and not applicable using statistical tests.

6.3.1 Evaluation measures

The authors use the following measures:

1- Mean Magnitude of Relative Error (MMRE) using its equation, which is explained in equation 4.2.

2- The Balanced Mean Magnitude of Relative Error (BMMRE) using its equation, which is explained in equation 4.3.

The result of applying these two equations (4.2, 4.3) is shown inTable 6-3.

The model which has fewer values of MMRE and BMMRE is the closest to the actual, as explained inTable 6-4.

Table 6-3: Applying MMRE & BMMRE on the evaluation dataset

Case study #	Actual defects (y_i)	Predicted$(\hat{y}_i)$	$\lvert y_i - \hat{y}_i \rvert$	$\frac{\lvert y_i - \hat{y}_i \rvert}{y_i}$	$\frac{\lvert y_i - \hat{y}_i \rvert}{Min\,(y_i, \hat{y}_i)}$
1	148	192	44	0.2972973	0.29729729
3	209	421	212	1.01435407	1.01435406
4	204	205	1	0.00490196	0.00490196
5	53	52	1	0.01886792	0.01923076
7	29	29	0	0	0
8	71	74	3	0.04225352	0.04225352
9	90	113	23	0.25555556	0.25555555

10	129	196	67	0.51937984	0.51937984
12	109	107	2	0.01834862	0.01869158
13	688	693	5	0.00726744	0.00726744
18	1597	1552	45	0.02817783	0.02899484
		Sum		2.20640407	2.20792689
		MMRE & BMMRE respectively		0.1226	0.1227

The MMRE and BMMRE for the other models have been calculated using the same projects that have been used to evaluate the proposed model.

Table 6-4: Model evaluation measures

MMRE				BMMRE			
Proposed model	Yadav model	Fenton model	Pandey and Goyal	Proposed model	Yadav model	Fenton model	Pandey and Goyal
0.1226	0.2685	0.2846	0.2870	0.1227	0.4664	0.7380	0.5625

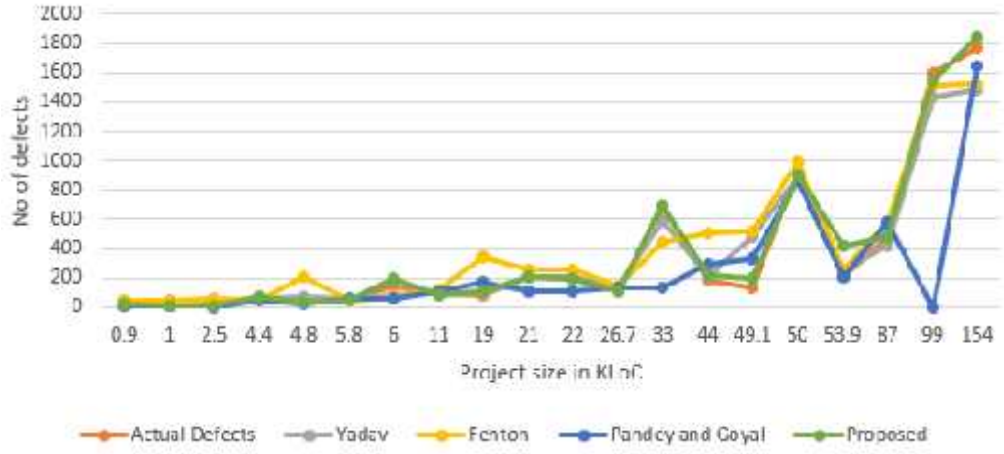

Figure 6.1: Project size vs number of defects

6.3.2 Model results validation

FromTable 6-4, it can be noticed that the MMRE and BMMRE for the proposed model are: 0.1226, 0.1227,respectively, which are the least compared with the other models, so it is clear that the predicted accuracy of the proposed model is the closest to the actual compared with Yadav [16], Fenton [21], Pandey and Goyal [22] models for both measures.

Also, it is observed in Table 6-2 that the differences between the actual and proposed defects sometimes increase when the size of the project increases in a nonlinear relation.

From Table 6-4Table 6-4 andFigure 6.1, it is clear that the prediction of the proposed model is the closest to the actual, so the proposed model accuracy is the highest compared with the three models.

6.4 RESULTS CONCLUSION

After explaining the case study, which illustrated how the proposed model works step by step, the resultsshowed that the proposed model accuracy is the highest compared with the other models that can be used within the early stages of SDLC. This model shows that it is reliable, effective, and can be used to detect the defect density during the early stages of SDLC by the project managers to take the right decision at the right time to develop more reliable software. In the proposed model, the authors used the most related metrics to the reliability during the first two stages of SDLC, which are the requirement and design stages.

The proposed model was validated using part of the twenty software projects[25], which were embedded in electronic products so that the model results can be considered with electronic software types.

To generalize these results with other project types, the model should be validated using another dataset that has sufficient data to be used, which will be as future work.

Chapter 7: Conclusion and Future Work

This thesis proposed a model that predicts software reliability at the early stages of the software development life cycle, which are the requirement and design stages. The proposed model uses some of the most related software metrics to software reliability in both stages. The proposed model uses Quality of Documentation Inspected (QDI), Regularity, Inspection, and Walk-through the specification &documentation (RIW), Requirement Fault Density (RFD), Requirement Stability (RS) metrics as the inputsto the requirement stage in fuzzification system, and uses Complexity of functionality (CC), Scale of New functionality (SNF), Design Review Effectiveness (DRE) metrics as the inputsto the design stage. The output of the first stage is Requirement Phase Defect Density Indicator (RPDDI), which is used as an input metric to the second stage. The design stage output metric is the Design Phase Defect Density Indicator (DPDDI). The outputmetrics are used as variables multiplied with the software size to get the total number of defects at the end of the defuzzification.

The proposed model used the fuzzy inference system to predict software defect density. The fuzzy system membership functions and rules were developed using human intelligence and domain expert who have a long experience period in software analysis and development.

This model is proposed to be used within the first two stages of SDLC, and it uses the software size, which is not available in these stages, so; the managers should roughly estimate the software size by using any ofthe estimation tools.

From the proposed model results, the defect numbers can be calculated at the end of the design stage of SDLCwith a high accuracy percentage compared with the other models that predict the defects during the early stages of SDLC.These results will enhance the software development by giving the ability to the project managers to discover the software defaults in early stages, so they can make the right decision at the right time to develop reliable softwarewith saving time and budget.

Furthermore, the proposed model can be used easily by the concerned people because it uses some of the software metrics, which can be evaluated through the first two stages of SDLC, which can be easily calculated.

The proposed model is evaluated using data set of software that is embedded into the electronic equipment, so to make it more general and valid for other types of software, the model should be validated using another dataset thatconsists of other types of software.

As future work, an evaluation for the proposed model using different dataset types is needed to be used generally.

Bibliography

[1] M.R. Lyu, *Handbook of Software Reliability Engineering*, vol. 222, IEEE Computer Society Press, CA, 1996.

[2] Mladen A. Vouk, *Software Reliability Engineering*, Box 8206 North Carolina State University, Raleigh, NC 27695, 2000.

[3] Norman E. Fenton and Martin Neil, *A Critique of Software Defect Prediction Models*, Member, IEEE Computer Society, VOL. 25, NO. 3, MAY/JUNE 1999.

[4] Harikesh Bahadur Yadav, Dilip Kumar Yadav, *A fuzzy logic based approach for phase-wise software defects prediction using software metrics*, Department of Computer Applications, National Institute of Technology, Jamshedpur 831014, India, 2015.

[5] K.Y. Cai, C.Y. Wem, M.L. Zhang, *A critical review on software reliability modeling, Reliable.* Eng. Syst. safety 32 (3) (1991) 357–371.

[6] C. Kaner, *Software engineering metrics: what do they measure and how do we know?* In: 10th International Software Metrics Symposium, vol. 6, 2004.

[7] William Farr,*software reliability modeling survey*, Naval Surface Warfare Center.

[8] Latha Shanmugam, Dr. Lilly Florence, *An Overview of Software Reliability Models*, Volume 2, Issue 10, October 2012.

[9] Y. Jiang, B. Cukic and T. Menzies, *Fault Prediction using Early Lifecycle Data*, In (ISSRE-07) Proceeding of 18th IEEE International Symposium on Software Reliability Engineering (ISSRE), pp. 237–246, (2007).

[10] Ramamoorthy, FB Bastani, *Software Reliability—Status and Perspectives*, IEEE Transactions on Software Engineering, 1982.

[11] C. Smidts, M. Stutzke, R.W. Stoddard, *Software reliability modeling: an approach to early reliability prediction*, IEEE Trans. Reliab. 47 (3) (1998).

[12] V. Cortellesa, H. Singh, B. Cukic, *Early reliability assessment of UML based software models*, Proceedings of the 3rd International Workshop on Software and Performance, 2002, pp. 302–309.

[13] Y. Maa, S. Zhua, K. Qin and G. Luo, Combining the Requirement Information for Software Defect Estimation in Design Time, *Information Processing Letters*, vol. 114(9), pp. 469–474, (2014).

[14] Jelinski. Z, Moranda P.B. —*Software Reliability Research Statistical Computer Performance Evaluation*, W. Frebierger, New York, Academic Press.

[15] Syed Rizvia, Vivek Singhb and Raees Khanc, *Fuzzy Logic based Software Reliability Quantification Framework: Early Stage Perspective (FLSRQF)*, Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016).

[16] Dilip Yadav, Chaturvedi, Ravindra Misra, *Early Software Defects Prediction Using Fuzzy Logic*, International Journal of Performability Engineering Vol. 8, No. 4, July 2012, pp. 399-408.

[17] Nasa iv&v facility. Metric data program. Available from http://MDP.ivv.nasa.gov/, visited on 12/02/2017.

[18] Sirsendu Mohanta, Gopika Vinod, Rajib Mall,*A technique for early prediction of software reliability based on design metrics*, Int J Syst Assur Eng Manag (Oct-Dec 2011) 2(4):261–281.

[19] IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, 1990.

[20] Sherif Yacoub, Bojan Cukic, Hany H. Ammar, *A Scenario-Based Reliability Analysis Approach for Component-Based Software*, 2004.
[21] Norman Fenton, Martin Neil, William Marsh, Peter Hearty, Łukasz Radliński, Paul Krause, *On the effectiveness of early life cycle defect prediction with Bayesian Nets*, 2008.
[22] Ajeet Kumar Pandey, N. K. Goyal, *A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics*, International Journal of Electronics Engineering, 1(2), 2009.
[23] American Psychological Association (APA). (2010). *Publication Manual of the American Psychological Association* (6th Ed.). Washington, DC: Author.
[24] B.B. Agrawal, S. P. Tayal, M. Gupta, *Software Engineering & Testing*, Library of Congress cataloging-in-publication Data, 2010.
[25] N. Fenton, M. Neil, W. Marsh, P. Hearty, L. Radlinski, *Project data incorporating qualitative factors for improved software defect prediction*, Third international workshop on predictor models in software engineering, 2007.
[26] Linda Rosenberg, Ted Hammer, Jack Shaw, SOFTWARE METRICS AND RELIABILITY, 1998.
[27] Meena, Vipin Aroda, *Software Reliability-Most Important Aspect of Software Quality*, IJIRST –International Journal for Innovative Research in Science & Technology| Volume 2 | Issue 1 | June 2015.
[28] *SDLC (Software Development Life Cycle) Phases, Methodologies, Process, And Models.*https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/, visited on 04/04/2020.
[29] Gillies, A.C.,*Software Quality, Theory and management*, Chapman Hall Computing Series, London, UK, 1992.
[30] IEEE Standard 982.2,*Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software*, 1987.
[31] C. Catal, B. Diri, *A systematic review of software fault predictions studies*, Expert Syst. Appl. 36 (4) (2009) 7346–7354.
[32] C. Catal, *Software fault prediction: a literature review and current trends*, Expert Syst. Appl. 38 (4) (2011) 4626–4636.
[33] D. Radjenovic et al., *Software fault prediction metrics: a systematic literature review*, Inf. Softw. Technol. 55 (8) (2013) 1397–1418.
[34] M. Li, C. Smidts, *A ranking of software engineering measures based on expert opinion*, IEEE Trans. Softw. Eng. 29 (9) (2003) 811–824.
[35] *What is software complexity and how you can manage it*? https://carlalexander.ca/what-is-software-complexity, visited on 26/09/2020.
[36] S.V. Wong, A.M.S. Hamouda, *Optimization of fuzzy rules design using genetic algorithm*, Elsevier Science. Advances in Engineering Software 31 (2000) 251–262.
[37] Dan Port, Marcel Korte. *Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research, ESEM'08*, October 9–10, 2008, Kaiserslautern, Germany, ACM 978-1-59593-971-5/08/10.
[38] LOTFI A. ZADEH, *Knowledge Representation in Fuzzy Logic*, IEEE Trans. Knowl. Data Eng. 1 (1) (1989) 89–100.
[39] Software development life cycle, https://practice.geeksforgeeks.org/problems/software-development-life-cycle, visited on 04/04/2020.
[40] Mohammad Ibraigheet, Sayed Abdullah Fadzli, *Software Reliability Prediction In Various Software Development Stages*, Journal of Theoretical and Applied Information Technology, ISSN: 1992-8645, 15th April 2018. Vol.96. No 7.