Predicting software test effort in iterative development using a dynamic Bayesian network

Nasir Majeed Awan
Adnan Khadem Alvi
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 40 weeks of full time studies.

Contact Information:

Author(s):
Nasir Majeed Awan
Address: C/O Uzair Hashmi Folkparksvagen 18:03 372 40 Ronneby, Sweden.
E-mail: nasir.awn@gmail.com

Author(s):
Adnan Khadem Alvi
Address: Folkparksvagen 18:06, 372 40 Ronneby, Sweden.
E-mail: adnankhadem@gmail.com

University advisor(s):
Dr. Richard Torkar
School of Engineering

School of Computing
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
ABSTRACT

It is important to manage iterative projects in a way to maximize quality and minimize cost. To achieve high quality, accurate project estimates are of high importance. It is challenging to predict the effort that is required to perform test activities in an iterative development. If testers put extra effort in testing then schedule might be delayed, however, if testers spend less effort then quality could be affected. Currently there is no model for test effort prediction in iterative development to overcome such challenges. This paper introduces and validates a dynamic Bayesian network to predict test effort in iterative software development.

In this research work, the proposed framework is evaluated in a number of ways: First, the framework behavior is observed by considering different parameters and performing initial validation. Then secondly, the framework is validated by incorporating data from two industrial projects.

The accuracy of the results has been verified through different prediction accuracy measurements and statistical tests. The results from the verification confirmed that the framework has the ability to predict test effort in iterative projects accurately.

Keywords: Software test effort estimation, Bayesian network, dynamic Bayesian network, Iterative development, test effort estimation in iterative development.
ACKNOWLEDGEMENT

First of all, we would like to thanks to Almighty Allah for providing us an opportunity to complete the research.

Secondly, we are extremely thankful to our Supervisor Richard Torkar for his consistent support, guidance and encouragement throughout our thesis work. His assistance, knowledge, and valuable feedback enable us to produce high quality work.

Thirdly, we would like to thank all of our friends (names cannot be mentioned here) for their continue encouragement, support, motivation and interminable memories. Further, special thanks to the friends as they enable us to complete industrial part of the research. We also appreciate the industrial contacts for their valuable time, support, feedback on our proposed framework and on providing industrial data.

Finally, we would to express special gratitude to our beloved family for their continue prayers, support, and motivation. We would like to dedicate this research work to our beloved parents.
# CONTENTS

1 INTRODUCTION .................................................................................................................. 7

1.1 BACKGROUND .................................................................................................................. 7

1.1.1 Aims And Objectives ................................................................................................... 9

1.1.2 Research Questions ..................................................................................................... 9

1.1.3 Expected Outcomes .................................................................................................... 9

1.1.4 Research Methodology ............................................................................................... 10

1.1.5 Relationship Between The Objectives And Research Methodology ....................... 11

1.1.6 Thesis structure ......................................................................................................... 12

1.1.7 Terminology ............................................................................................................... 13

1.2 SOFTWARE TESTING ...................................................................................................... 14

1.2.1 Software Testing ......................................................................................................... 14

1.2.2 Verification And Validation ......................................................................................... 14

1.2.3 V-Model .................................................................................................................... 15

1.2.3.1 Unit Testing ............................................................................................................ 15

1.2.3.2 Integration Testing .................................................................................................. 15

1.2.3.3 System Testing ....................................................................................................... 15

1.2.3.4 Acceptance Testing ............................................................................................... 16

1.2.4 Testing life cycle ......................................................................................................... 16

1.3 ITERATIVE DEVELOPMENT ........................................................................................... 18

1.4 BAYESIAN NETWORK ..................................................................................................... 19

1.4.1 Definitions .................................................................................................................. 19

1.4.2 Bayesian network basics ............................................................................................ 19

1.4.2.1 Conditional Probability ........................................................................................ 19

1.4.2.2 Joint Probability Distribution (JPD) ..................................................................... 20

1.4.2.3 Bayes’ Rule ............................................................................................................ 20

1.4.2.4 Conditional independence .................................................................................... 20

1.4.2.5 D-Separation ....................................................................................................... 20

1.4.3 How to build a BN model? ......................................................................................... 21

1.4.4 Advantages of using BN ............................................................................................ 21

1.4.5 DYNAMIC BAYESIAN NETWORK ........................................................................... 22

1.5 EFFORT ESTIMATION .................................................................................................... 24

1.5.1 Software Effort Estimation ......................................................................................... 24

1.5.2 Classical Methods For Software Estimation ............................................................... 24

1.5.3 TEST EFFORT ESTIMATION ................................................................................... 26

1.6 LITERATURE REVIEW – AN OVERALL SUMMARY .................................................... 28

1.6.1 Inputs ......................................................................................................................... 28

1.6.1.1 Databases and electronic resources ...................................................................... 28

1.6.1.2 Keywords search ................................................................................................... 28

1.6.1.3 Selection criteria ................................................................................................... 29

1.6.2 Processing .................................................................................................................. 30

1.6.2.1 Apply the literature ............................................................................................... 31

1.6.2.2 Analyze and evaluate ............................................................................................ 32

1.6.3 Output ....................................................................................................................... 33

2 PROPOSED FRAMEWORK DESIGN .................................................................................. 34

2.1 FRAMEWORK TOOLS .................................................................................................. 34

2.2 FRAMEWORK REQUIREMENTS .................................................................................... 34

2.3 FRAMEWORK LIMITATIONS ........................................................................................ 34

2.4 FRAMEWORK DESIGN .................................................................................................. 34

2.5 FRAMEWORK NOTATIONS ........................................................................................... 36

2.6 FRAMEWORK PARAMETERS ......................................................................................... 36

2.7 FRAMEWORK NODES .................................................................................................... 37

2.7.1 Rework test effort ..................................................................................................... 37

2.7.2 Rework test process quality ...................................................................................... 37

2.7.3 Rework test effectiveness ......................................................................................... 37

2.7.4 Test tool quality ....................................................................................................... 37

2.7.5 Test team experience ............................................................................................... 37
2.7.6 Test process quality ................................................................. 37
2.7.7 Number of bugs found by test cases ........................................... 38
2.7.8 Total number of bugs found ........................................................ 38
2.7.9 Test case effectiveness .............................................................. 38
2.7.10 Test process effectiveness .......................................................... 38
2.7.11 Test process overall effectiveness .............................................. 38
2.7.12 Iteration length ........................................................................ 38
2.7.13 Team size .................................................................................. 38
2.7.14 Test effort .................................................................................. 39
2.8 FRAMEWORK FOR SINGLE ITERATION ........................................ 40
2.9 FRAMEWORK INTERACTION IN MULTIPLE ITERATIONS ............. 40

3 FRAMEWORK BEHAVIOR AND INITIAL VALIDATION .................... 41
3.1 FRAMEWORK EVALUATION TOOLS .......................................... 41
3.2 FRAMEWORK BEHAVIOR ............................................................ 41
3.2.1 Framework parameters learning ............................................... 42
3.3 FRAMEWORK INITIAL VALIDATION ........................................... 44
3.3.1 Prediction with and without observation ...................................... 44

4 FRAMEWORK INDUSTRIAL VALIDATION ...................................... 47
4.1 INDUSTRIAL INTERVIEWS ........................................................... 47
4.1.1 Company A ............................................................................ 47
4.1.1.1 Interviewee A .................................................................. 47
4.1.1.2 Project A ......................................................................... 47
4.1.2 Company B ............................................................................ 48
4.1.2.1 Interviewee B .................................................................. 48
4.1.2.2 Project B ......................................................................... 48
4.2 INDUSTRIAL VALIDATION .......................................................... 49
4.2.1 Project A ................................................................................ 49
4.2.2 Project B ................................................................................ 52
4.3 VALIDATION RESULTS ............................................................... 55
4.3.1 Project A ................................................................................ 56
4.3.1.1 Prediction accuracy measures ............................................. 56
4.3.1.2 Descriptive statistics ......................................................... 56
4.3.1.3 Normality test ................................................................. 56
4.3.1.4 Hypothesis testing .......................................................... 58
4.3.2 Project B ................................................................................ 60
4.3.2.1 Prediction accuracy measures ............................................. 60
4.3.2.2 Descriptive statistics ......................................................... 60
4.3.2.3 Normality test ................................................................. 60
4.3.2.4 Hypothesis testing .......................................................... 62
4.4 RESULTS ANALYSIS AND DISCUSSION .................................... 64

5 VALIDITY THREATS ...................................................................... 66
5.1 RELIABILITY OF MEASURE .......................................................... 66
5.2 HETEROGENEITY OF SUBJECTS .................................................. 66
5.3 EVALUATED APPREHENSION ....................................................... 66
5.4 MONO-OPERATION BIAS ............................................................. 66
5.5 MONO-METHOD BIAS ................................................................. 66
5.6 HYPOTHESIS GUESSING ............................................................. 67
5.7 SELECTION OF SUBJECTS ............................................................ 67
5.8 SELECTION OF TOOLS ............................................................... 67
5.9 FRAMEWORK COMPARISON ....................................................... 67
5.10 LACK OF SYSTEMATIC REVIEW ............................................... 67
5.11 EXPERT JUDGMENT ................................................................. 67
5.12 STATISTICAL VALIDATION ....................................................... 67

6 EPILOGUE ...................................................................................... 68
6.1 CONCLUSIONS .......................................................................... 68
6.2 FUTURE WORK ........................................................................... 69

7 REFERENCES .................................................................................. 70
TABLE OF TABLES

Table 1: Terminologies.................................................................13
Table 2: Estimation methods with objectives, advantages and limitations .................................................25
Table 3: List of articles selected by considering title, abstract, inclusive/exclusive criteria ..............................................30
Table 4: Selected articles with different concepts ..................................................................................31
Table 5: Node symbol, title, formula and range values ............................................................................36
Table 6: Framework behavior - Baseline scenario based on initial settings .............................................41
Table 7: Test effort values in success, average and failing scenarios.........................................................42
Table 8: Motorola project data .........................................................................................................44
Table 9: Actual values with test experience data....................................................................................44
Table 10: Test effort prediction with observation entered........................................................................45
Table 11: Project A - actual data collected from interviewee A.................................................................49
Table 12: Project A - actual vs. predicted test effort values without observations entered..................49
Table 13: Project A - actual vs. predicted test effort values with observations (I1 & I2) entered ....................................................50
Table 14: Project A - actual vs. predicted test effort values with observations (I1, I2 & I3) entered ..........................................................................................................................51
Table 15: Project B actual data collected from interviewee B.................................................................52
Table 16: Project B - actual vs. predicted test effort values without observations entered..................52
Table 17: Project B - actual vs. predicted test effort values with observations (I1, & I2) entered ..........................................................................................................................53
Table 18: Prediction accuracy for Project A............................................................................................56
Table 19: Descriptive statistics for project A ...........................................................................................56
Table 20: Shapiro-Wilk normality test result for project B data ...............................................................57
Table 21: Test Statistics - Wilcoxon signed ranks test results for project A........................................59
Table 22: Prediction accuracy for Project B .............................................................................................60
Table 23: Descriptive statistics for project B dataset ..............................................................................60
Table 24: Shapiro-Wilk normality test result for project B actual data ....................................................61
Table 25: Wilcoxon signed ranks ...........................................................................................................63
Table 26: Test Statistics - Wilcoxon signed ranks test results for project B .........................................63
Table 27: Project A & B - Differences between actual & predicted values - with three observations ..........................................................................................................................64
Table 30: Project A - Differences between actual & predicted values - with three observations..........................................................................................................................64
TABLE OF FIGURES

Figure 1: Research flow diagram .......................................................... 10
Figure 2: Relationship between the objectives and research methodology ........... 11
Figure 3: An overview of thesis structure ................................................. 12
Figure 4: V-Model (adopted from [30]) ...................................................... 15
Figure 5: System testing (adopted from [27]) ............................................. 16
Figure 6: Steps to construct a Bayesian network (adopted from [40]) ................. 21
Figure 7: Dynamic Bayesian network general structure (adopted from [39]) ....... 23
Figure 8: Estimating project duration (adopted from [43]) ............................. 24
Figure 9: Proposed Framework design ....................................................... 35
Figure 11: Framework for multiple iterations ............................................. 40
Figure 10: Framework design for a single iteration ...................................... 40
Figure 12: Test effort with baseline scenario .............................................. 42
Figure 13: Test effort parameter learning in baseline, success, average, failing scenarios ................................................................. 43
Figure 14: Test process overall effectiveness parameter learning in different scenarios ................................................................. 43
Figure 15: Test effort prediction without entering any observations ................ 45
Figure 16: Test effort prediction with observations entered ............................. 46
Figure 17: Project A - actual vs. predicted Test effort .................................. 50
Figure 18: Project A - actual vs. predicted test effort – with observations \( I_1 \) & \( I_2 \) entered .................................................. 50
Figure 19: Project A - actual vs. predicted Test effort values with observations \( I_1 \), \( I_2 \) & \( I_3 \) entered .................................................................................................................. 51
Figure 20: Project B - Actual vs. predicted Test effort - no observation entered ........ 53
Figure 21: Project B - Actual vs. predicted Test effort with observations \( I_1 \) & \( I_2 \) entered .......................................................... 53
Figure 22: Histogram - Normality test for Project A actual values .................. 57
Figure 23: Histograms - Normality test for Project A predicted values .............. 57
Figure 24: Box plot - Normality test for Project A actual values ...................... 58
Figure 25: Box plot - Normality test for Project A predicted values ................. 58
Figure 26: Statistical test results for Project A - graph .................................. 59
Figure 27: Histograms - Normality test for Project B actual values .................. 61
Figure 28: Histograms - Normality test for Project B predicted values .............. 61
Figure 29: Box Plot - Normality test for Project B actual values ...................... 62
Figure 30: Box Plot - Normality test for Project B predicted values ................. 62
Figure 31: Statistical test results for Project B - graph .................................. 63
Figure 32: A Framework model in multiple iterations – (part I), next page ............ 74
Figure 33: A Framework model in multiple iterations – (part II) ......................... 75
Figure 34: A detail proposed Framework design with probability distribution ........ 76
Figure 35: Industrial questionnaire used for project A data collection ................. 77
Figure 36: Industrial questionnaire used for project B data collection ................. 77
1 INTRODUCTION

1.1 BACKGROUND

Testing is an essential part of software development, since it is 40-60 percent of the whole software development effort [1][2]. Many times, testing teams are responsible to handle different testing activities such as estimating test effort, bug identification, test case designing, test tool selection, test team selection and etc. Hence, test managers need to plan the schedule accurately and efficiently to utilize the testing resources to meet deadlines. An accurate and efficient test effort estimation method ensures that test managers can help in completing projects successfully and on-time [1].

Inaccurate effort estimates may lead to poor quality, customer dissatisfaction, and developer’s frustration [3]. Project uncertainty, use of estimation development processes, use of estimation management processes and the estimator’s experience are a few factors that can affect effort estimation errors [4]. In 2008, Standish group concluded that reliable estimates are in the list of top ten success factors of the software development projects [5]. Therefore, it is important to have an accurate and efficient method for test effort estimation.

In iterative development, short-term iterations are used to deal with customer feedback and changes [6][7]. A product requirement frequently changes from one iteration to the next, owing to external factors such as new technologies, new competitors, or new features demanded by the customer. Scope, quality, time, and cost are the most important variables that shape iterative projects. It is important to manage iterative development projects in a way to maximize quality, minimize project cost and time to market [8]. To attain this, an accurate project estimate of effort and duration are of high importance [9]. It is challenging to determine the amount of effort required to perform test and defect fixing activities, because if testers spend extra effort in testing, then schedule may be delayed; however, if they spend less testing effort then quality can be affected [10]. Incomplete artifacts also make it difficult to estimate effort and size [11]. Furthermore, little work has been done on modeling, planning and controlling incremental development [12], particularly in terms of planning and controlling the effort.

Bayesian networks (BN) may provide a way to plan, control and estimate effort. Bayesian networks are cause-effect graphs that are capable of modeling uncertainty. It can combine sparse data, prior assumptions and expert judgment into one single model [13].

In recent years, many researchers [14][15][16] have used Bayesian network to model uncertainties for different kinds of software problems. BN has been used successfully to support the managerial decision-making [17] [18], allowing the project manager to trade-off the resources used against the output that is functionality and quality [17].

Fenton and Neil [19] [13] explained that BN has many benefits over classical or regression-based models. Bayesian network approach does not rely on single point value; instead a complete distribution of a variable is involved. Instead of predicting a single value of the variable it provides a complete probability distribution.

In 2006, Wang et al. presented a project level estimation model framework using Bayesian belief network (BBN) [20]. It uses four sub-models: component estimation, test effectiveness estimation, residual defect estimation and test estimation. By using this framework, estimation of quality, effort, schedule and scope can be computed at both project level and specific phase level. It also provides support for managerial decision-making. However, the
problem with this framework was that it has not been validated in any real project and is not attached with any development life cycle. Bayesian network is also used to handle uncertainties in software testing. Rees et al. [14] used Bayesian graphical models (BGM) to model the uncertainties involved in software testing, quality process and provide support to test managers to use the model.

In 2009, Hearty et al. [13], presented a Bayesian network causal model for the extreme programming (XP) methodology. They showed how, without using any additional metrics collection programs, their model could learn from project data in order to make quantitative effort predictions and risk assessments. They validated their model from a real industry project.

Wooff et al. presented an approach for software testing using a Bayesian graphical model [15]. They have conducted a case study and tried to solve software testing problems with the help of formal mechanisms for logical structuring of software testing, test design and analysis process, incorporation and implication of test results, probabilistic and statistical treatment of uncertainties. The model used software tester’s knowledge for further use.

Abrahamsson et al. [21], proposed an incremental, iteration-based effort prediction model for agile or iterative development methodology. They have validated the model with two semi-industrial projects by conducting a case study. The results show, that their model can perform better as compared to traditional approaches. However, their model is not used to estimate test effort, but estimates development effort. Further, they have used traditional incremental approaches for prediction. In addition, most of the participants in their case study were master students and new to XP approach, thus they do not represent true sample selection.

As far as we know, no model has been developed for test effort estimation in an iterative development methodology. Thus, we propose a framework for iterative development using a dynamic Bayesian network that will be helpful for the test manager to predict and analyze effort required in iterative development.

In software development planning, effort estimation is the main challenge [22]. Briefly due to challenges and complexities involved in test processes [10][12][11], many problems can occur such as budget overruns, delays, contract loss, suboptimal resource allocation and poor software quality [23], so there is a need to handle these issues. An accurate effort estimate and duration are of high importance [9], particularly for test processes. Our research work is a small contribution towards the improvement of test effort estimation processes in iterative development. We have proposed a detailed framework for test effort estimation in iterative development using a dynamic Bayesian network. In this way the practitioner, test manager, technical lead or project manager can easily learn and execute the model to produce accurate test effort prediction results.
1.1.1 AIMS AND OBJECTIVES

The overall aim of this study is to improve test effort estimation processes in iterative development and propose a framework to predict test effort in iterative development using a dynamic Bayesian network (DBN). To attain this aim the following objectives must be met:

- Propose a detailed framework design for test effort estimation in iterative development using a DBN.
- Framework should be able to learn from projects data
- Framework should be able to predict test effort in iterative development
- Framework should be statistically valid to predict test effort in iterative development

1.1.2 RESEARCH QUESTIONS

The following research questions will be answered:

*RQ1*: How do we can construct a test effort estimation framework in iterative development using a dynamic Bayesian network?

*RQ2*: Does the framework have learning and predicting behaviour?

*RQ3*: Is such a framework statistically valid to predict test effort in iterative development?

The above research questions aim to provide a detailed framework using a dynamic Bayesian network. It helps to predict test effort required for the future iterations in test processes. The proposed framework is analyzed and validated using two projects from industry.

1.1.3 EXPECTED OUTCOMES

The expected outcomes include:-

- Framework design to predict test effort in iterative projects
- Framework learning and predicting behavior
- Framework statistical validation
1.1.4 RESEARCH METHODOLOGY

The mixed research method [24] was used to conduct this research work. Figure 1 shows the detailed research flow that we have used during the research work. The study comprised of two stages.

In the first stage of this study, a literature review and qualitative analysis was performed. It facilitates to recognize important estimation models, key factors, their logical relationships, and nature of relationships. It provides basis to design a framework.

During the framework design, possible types or states, symbols and initial values were defined for each variable. A graphical structure was built on the basis of factors and their logical relationships. The framework design was also reviewed by an external researcher to validate the design. Further, different scenarios were built to evaluate the framework behavior and perform initial validation.

![Research flow diagram](image)

In the second stage, framework industrial validation was performed. In order to validate the model, several interview sessions were arranged to collect data from industry. The collected data was incorporated with the framework to validate, analyze and test. Further, several statistical and accuracy measurements were performed to check the accuracy of the proposed framework.
1.1.5 RELATIONSHIP BETWEEN THE OBJECTIVES AND RESEARCH METHODOLOGY

The relationship between the objectives and research methodology is shown in Figure 2. In this study, there were several objectives:

One of main objective was to propose a Bayesian framework design that was accomplished by conducting a literature review and qualitative analysis.

Second, the framework should learn from projects data. This objective was achieved in the framework behavior and initial validation step, where the framework learning and predicting behavior was evaluated.

Third, the framework should be able to predict the test effort in iterative projects; it was one of the main objectives of this study. This objective was achieved by incorporating industrial projects data with the model and framework validation is performed.

However, the framework statistical validation was another important objective. It aims to validate the framework prediction results. This was achieved by performing several prediction accuracy measurements and statistical tests.

![Figure 2: Relationship between the objectives and research methodology](image)

Figure 2 shows the relationship between the objectives and research methodology, however, the research questions are answered as follow:

The literature review and qualitative analysis was helpful in answering the RQ1 regarding the framework design (Section 2).

While the RQ2 is answered through the framework behavior and initial validation step (Section 3). In this step, we have evaluated the framework by observing the framework parameters behavior in different scenarios, initial validation is also performed to evaluate the predictions behavior.

However, the RQ3 is answered through framework validation and qualitative analysis step (Section 4). In this step, we have validated the framework by incorporating data from two industrial projects. The accuracy of the results has been verified through different prediction accuracy measurements and statistical tests. The results show that the model predictions are statistically valid.
1.1.6 Thesis Structure

In this section we have provided an overview of the thesis report structure as shown in Figure 3.

1. Introduction
   - Background
   - Software Testing
   - Iterative Development
   - Bayesian Network
   - Effort Estimation
   - Literature review – an overall summary

2. Propose Framework Design
   - Framework tools
   - Framework design
   - Framework notations
   - Framework parameters
   - Framework Nodes
   -...

3. Framework Behaviors and initial Validation
   - Framework evaluation tools
   - Framework behavior
   - Framework initial validation

4. Framework industrial validation
   - Industrial interviews
   - Industrial validation
   - Validation results

5. Validity threats
   - Reliability of measure
   - Heterogeneity of subjects
   - Evaluated apprehension
   - Mono-operation bias
   - Mono-method bias
   -...

6. Epilogue
   - Discussion and conclusions
   - Future work

Figure 3: An overview of thesis structure

The remainder of the report structure is organized as follows: The rest of Section 1 provides an overview of software testing, iterative development, Bayesian network, effort estimation, and literature review – an overall summary. Section 2 describes the design of the proposed framework and discusses notations, parameters, nodes, framework design for single and multiple iterations. In Section 3, framework behavior and initial validation is discussed. Section 4 validates the framework through the data from two real projects, while validation results are also discussed by performing prediction accuracy measurements and statistical tests. In Section 5, validity threats are discussed, while Section 6 provides conclusions including possible future work.
1.1.7 TERMINOLOGY

Following table provides the list of terms or abbreviations and their definitions that are used in the report.

Table 1: Terminologies

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BN</td>
<td>Bayesian network</td>
</tr>
<tr>
<td>BBN</td>
<td>Bayesian belief network</td>
</tr>
<tr>
<td>XP</td>
<td>Extreme programming</td>
</tr>
<tr>
<td>BGM</td>
<td>Bayesian graphical model</td>
</tr>
<tr>
<td>DAG</td>
<td>Directed acyclic graph</td>
</tr>
<tr>
<td>CP</td>
<td>Conditional Probability</td>
</tr>
<tr>
<td>JPD</td>
<td>Joint probability distribution</td>
</tr>
<tr>
<td>DBN</td>
<td>Dynamic Bayesian network</td>
</tr>
<tr>
<td>COCOMO</td>
<td>COnstructive COst MOdel</td>
</tr>
<tr>
<td>SLOC</td>
<td>Source line of code</td>
</tr>
<tr>
<td>FPA</td>
<td>Function point analysis</td>
</tr>
<tr>
<td>FP</td>
<td>Function point</td>
</tr>
<tr>
<td>TPA</td>
<td>Test point analysis</td>
</tr>
<tr>
<td>UCP</td>
<td>Use case point</td>
</tr>
<tr>
<td>Framework or model</td>
<td>It is assumed that the term model and framework are same, therefore it is interchangeability used.</td>
</tr>
<tr>
<td>MRE</td>
<td>Magnitude of Relative Error</td>
</tr>
<tr>
<td>MMRE</td>
<td>Mean of Magnitude of Relative Error</td>
</tr>
<tr>
<td>EMRE</td>
<td>Estimation Magnitude of Relative Error</td>
</tr>
<tr>
<td>MEMRE</td>
<td>Mean of Estimation Magnitude of Relative Error</td>
</tr>
<tr>
<td>Asymp. Sig. (2-tailed)</td>
<td>Asymptotic significance 2-tailed</td>
</tr>
<tr>
<td>SLIM</td>
<td>Software Lifecycle Management</td>
</tr>
<tr>
<td>OLS</td>
<td>Ordinary Least Squares</td>
</tr>
<tr>
<td>SEER</td>
<td>Software Evaluation and Estimation of Resources</td>
</tr>
<tr>
<td>Initial validation</td>
<td>It aims to test the model learning and predicting behavior before going to industry.</td>
</tr>
<tr>
<td>Industrial validation</td>
<td>It is performed by incorporating a complete dataset from two industrial projects.</td>
</tr>
</tbody>
</table>
1.2 SOFTWARE TESTING
This section describes the software testing, verification and validation, V-model and software testing life cycle.

In the current era of software engineering, testing activities are getting popular day by day. Testing activities play an important role to give confidence that companies are meeting requirements. Software testing processes help companies to improve quality of software by finding and fixing errors in program. The main objective of software testing is to uncover errors.

1.2.1 SOFTWARE TESTING

Software testing is the process of using a system or component under given circumstances of observing or noting results from a given perspective [25]. It is the process of executing a piece of software with the intention of finding failures in software [26]. Software testing is a corrective action approach in which defects are identified.

There are some testing principles that have been addressed in [27].
1. Testing is a process of exercising a software component using a selected set of test cases with the intent of (i) revealing defects, and (ii) evaluating quality.
2. The testing objective is to detect defects, a good test case is one that has high probability of revealing a defect that is yet undetected.
3. Testing should be carried out by a group that is independent of the development group.
4. A test must be repeatable.
5. Testing should be planned.

1.2.2 VERIFICATION AND VALIDATION

Software verification and validation (V&V) is the process to assess the software for defects and faults during software development life cycle, it ensures software quality and functionality. Software verification is the defect preventive approach: Inspections, walkthroughs and reviews come under verification [28].

Software validation is a corrective approach, consisting of a collection of different activities that help to ensure that software is traceable according to the customer requirements [29]. Software validation tries to answer the question “are we building the right product?”. System testing, integration testing, functional testing and acceptance testing are involved in validation process.

There are a number of benefits to conduct V&V activities e.g. it assists in deciding whether to proceed to the next phase or not, highlight problems at early stage of development life cycle, provides early statistics of software quality and bugs prevention. More detailed information on software verification and validation can be found in [26][27][28][29].

Software testing is performed using different techniques at different levels that help to improve software quality and reliability. These methodologies are also performed at different levels in the V-model [30]. Software verification activities are performed during documentation and development of software while validation activities are performed during unit testing, integration testing, system testing, and acceptance testing in the V-model.
1.2.3 V-Model

Different testing methodologies are performed at different levels in the V-model. Each level can have more sublevels. Different projects have different levels of testing. Simple projects use only one level of testing while complex projects use more than one level of testing. Testing is performed in parallel with all other activities of the software development processes.

We can also divide software testing into two different levels e.g. low level and high level testing. Component level testing is performed at low level while system level testing is performed at high level.

![V-Model](image)

**Figure 4: V-Model (adopted from [30])**

1.2.3.1 Unit Testing

A unit is a small component or module of testable software. Unit level testing has a high importance in different testing techniques. The basic purpose of unit testing is to ensure that the functionality is according to the specification. During the unit testing different kind of defect can be found such as functional description defects, algorithmic defects, data defects, control logic and sequence defects [27]. Unit testing should be performed by independent testers [27]. Each unit should be reviewed by a separate team of reviewers. Defects found during unit testing are less expensive to fix as compared to other levels of testing.

1.2.3.2 Integration Testing

Integration testing is a systematic technique for constructing and executing tests to uncover errors associated with interfacing [29]. The purpose of the integration testing is to: [27]

1) Detect defects on unit interfaces.
2) Integrate units into working subsystem or system.

There are two different types of integration testing i.e. top-down integration testing and bottom-up integration testing.

1.2.3.3 System Testing

Software assembled into system after completing unit and integration testing. System testing is performed on the basis of a requirement specification to test functional and non-functional behavior of the system. There are many types of system testing e.g. GUI testing, stress testing, load testing, performance testing and usability testing. System testing is always performed according to a test plan. It includes test cases, test procedures and schedule. All newly developed software components and modules are tested under system testing.
1.2.3.4 Acceptance Testing
Acceptance testing is performed by the customer. It ensures that the system is built according to customer requirements.

1.2.4 TESTING LIFE CYCLE

The basic purpose of the testing life cycle is to focus on test planning, test designing, test execution and test analysis. All these activities are carefully planned and managed.

The software testing life cycle involves following activities [31]:
- Test planning
- Test designing
- Test execution
- Test review

A software testing process consists of following activities [29]:
- Test planning
- Test case design
- Test execution
- Data collection and evaluation result

The following activities are performed during software testing life cycle [32]:
- Test plan generation
- Test design generation
- Test case generation
- Test procedure generation
- Test execution

According to [33] a test strategy is composed of following activities:
- State your assumptions
- Build your test inventory
- Perform analysis
- Estimate the test effort
- Negotiate for the resources to conduct the test effort
- Build the test scripts
- Conduct testing and track test progress
- Measure the test performance
After carefully observing the different stages of software testing life cycle, we found [34] that the software testing life cycle can be divided in four categories that are

- Test planning
- Test designing
- Test execution
- Test analysis
1.3 **Iterative Development**

This section briefly describes iterative development methodology and its related issues.

The iterative development methodologies help to develop software in a way to minimize risks and cost, increase productivity, manage change and make improvements based on quick customer feedback [35][36]. These approaches are self-organizing, incremental, and iterative in nature [37] which facilitates to build software in multiple iterations in a particular sequence, where iterations are short-term, varying from two to three weeks [7]. A single iteration consists of requirements analysis, design, implementation, test, and release phases. Thus, iteration is a self contained mini project with a well defined outcome that is a stable and executable release [7][36].

In iterative development, a product requirements frequently changes from one iteration to the next, owing to external factors such as new technologies, new competitors, or new features demanded by the customer. The scope, quality, time, and cost are the most important variables that shape iterative projects [8][36].

As the iterative development cycles are short term, sometimes it is said that there is no need for planning and estimating. This argument is not true; it is also likely saying that during the iteration we make continuous decisions about where to go next, where to end. Customer goals needed to be met on time. Therefore, planning and estimating provides a mechanism to keep track on project progress that helps to achieve project goals [36]. More detailed information on iterative software development can be found in [7][36].

Although iterative development provides a quick delivery, one of the common drawbacks is incomplete artifacts between iterations that cause many problems to estimate effort and size [11]. An accurate project estimates and their durations are of high importance [9]. However, it is challenging to determine the amount of effort required to perform test and defect fixing activities [10]. Further, little work has been done on modeling, planning and controlling incremental development [12], particularly in terms of planning and controlling the effort.
1.4 Bayesian Network

Before proceeding further, it is very important to have an overview of Bayesian network. Therefore, in this section, we have described briefly Bayesian network definitions, basics, advantages and dynamic Bayesian network.

1.4.1 Definitions

The term Bayesian Network is defined in different ways. Some of these definitions are explained below:

According to [38], a Bayesian network consists of
- A set of variables and a set of directed edges between different variables
- Each variable has a limited number of mutually exclusive states
- Variables and directed edges construct an acyclic directed graph (DAG)
- If there is no directed path $A_1 \rightarrow \ldots \rightarrow A_n$, so that $A_1 = A_n$, then a directed graph is acyclic
- Each variable has its own probability table

A Bayesian network is a graphical structure that represents and reasons about an uncertain domain. It contains nodes that represent a set of random variables from domain. These nodes are connected by a set of directed arcs that represent direct dependencies between the variables [39].

A Bayesian network is composed of nodes and directed arrows. Nodes represent variables while arrows (directed edges) represent casual relationship between the nodes. Nodes without parents (root node) are defined by their prior probability distribution but nodes with parents (leaf nodes) are defined through conditional probability distribution (CPD). A Bayesian network is a high level representation of a joint probability distribution for variables used to build a model for specific problem [40][41][13].

Bayesian network are used to model knowledge in engineering, medicine, text analysis, image processing, data fusion, and decision support systems [40]. Bayesian network has the ability to combine sparse data, prior assumptions and expert judgment into a single casual model [13].

1.4.2 Bayesian Network Basics

In this section, we have explained briefly Bayesian network basics that includes conditional probability, joint probability distribution, conditional independence and d-separation, more detailed information can be found in [38][39][13].

1.4.2.1 Conditional Probability

Conditional probability (CP) [38] can be defined as the probability of an event A is p, given an event B. For an event A and B, as $P(B) > 0$, then the conditional probability for A given B can be written as

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

For example: Let suppose $P(A) = \{4\}$, $P(B) = \{2, 4, 6\}$ and $P(A \cap B) = \{4\}$ then the Conditional probability for a die that will turn up the digit 4 given that an even number can be shown as $P(A|B) = \frac{1}{6} = \frac{1}{3}$. 

3/6
1.4.2.2 Joint Probability Distribution (JPD)

Bayesian network is a compact representation of JPD. Chain rule allows to factorize joint probability distribution. Let Bayesian network over $u = \{A_1, \ldots, A_n\}$ then joint probability distribution (JPD) $P(u)$ given all conditional probabilities [38][39].

$$P(u) = \prod_{i=1}^{n} P(A_i | \text{pa}(A_i))$$

Where $\text{pa}(A_i)$ is the parents of $A_i$

OR

$$P(x_1, x_2, \ldots, x_n) = P(x_1) \times P(x_2 | x_1) \times \cdots \times P(x_n | x_1, \ldots, x_{n-1})$$

$$= \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$$

1.4.2.3 Bayes’ Rule

Bayes’ rule is used to calculate the probability of effect node given the probability of cause node, it update our belief about node A given that we get information about node B. It can be stated as follow [38].

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where,

$P(A)$ is called as prior probability.

$P(A|B)$ posterior probability

$P(B|A)$ is known as likelihood of A given B.

1.4.2.4 Conditional independence

According to [38] , Conditional independence can be defined as

*The events A and B are independent if*

$$P(A|B) = P(A) \text{ and } P(A) \neq 0, P(B) \neq 0$$

Independence is symmetric, if A is independent of B then B is independent of A, that is, if $P(A) \neq 0$ and $P(B) \neq 0$, then $P(A|B) = P(A)$ if and only if $P(B|A) = P(B)$.

1.4.2.5 D-Separation

D-separation can be defined as [38], two variables A and B are d-separated, if there an intermediate variable V in all paths between A and B, such that

- V is instantiated and connection is serial or diverging.

Or

- Neither V or any V’s descendents receive observation, connection is converging

D-separation is also known as direction directed separation. D-separation can be of three types or connections such as serial, diverging or converging connections. A full detailed discussion on Bayesian network is beyond the scope of this research work, but more detailed information can be found in [38][39][13].
1.4.3 **HOW TO BUILD A BN MODEL?**

Following are the steps involved in the construction of BN model [40]

1. Identify variables
2. Determine graphical structure
3. Elicit and compute initial probabilities
4. Build scenarios to update model
5. Conduct sensitivity analysis
6. Evaluate model
7. Computational model

In the construction of BN, first step involves the identification of problem domain and relevant variables to the problem domain. Possible states and initial probabilities are assigned to each variable. Probability values can be gathered from the sources, for instance, expert judgment, literature review or empirical data [40]. In the second step, relationship among the different variables will be determined and graphical model is established. Third step is to determine the conditional probability for each variable. The fourth step is to create different scenarios to update and train the model. In the fifth step, sensitivity analysis is performed against the data or parameters entered to check the model performance. In the last step of the validation, it ensures that the designed model is accurate, valid and useful to resolve the domain problems.

![Figure 6: Steps to construct a Bayesian network (adopted from [40])](image)

1.4.4 **ADVANTAGES OF USING BN**

Bayesian network offers potential solution to the problems in various domains; it provides the following advantages over other approaches [40]

- In expert systems, BN models are the powerful tool used to represent the relationship between different variables and handles uncertainties.
- Graphical structure of BN provides a simple and easiest way to create relationship among different variables or nodes.
- Graphs in probabilistic modeling are suitable
• Its modeling approach allows qualitative inferences without any computational inefficiencies of Joint Probability Determinations (JPD).
• Bayesian network can be easily improve, it allows to include additional variables and mapped them from mathematics to common understanding easily.
• It provides a way to enter evidence and update the network to propagate probabilities in each node.
• It proposes an interactive graphical modeling mechanism so that the researchers and experts can use to understand the system behavior.
• It has a qualitative and quantitative part provides more advantages over any other methods.
• BN structural representation can be used for any natural occurring and complex problem domains.
• It is easy to understand direct dependencies and local distributions than complete joint distribution [40].
• Bayesian network encoded information facilitates the analysis of actions, sequences of events, consequences, observations, and expected utility.
• It has an important power to support the use of probabilistic inference to update and revise belief values.

1.4.5 DYNAMIC BAYESIAN NETWORK

Dynamic Bayesian networks are the temporal extensions of the Bayesian networks [42][13], it extends the Bayesian networks by adding temporal dimensions to the model [38][13]. DBNs explicitly model change over time, BN do not have any explicitly temporal relationship between the nodes or variables [38].

DBN consist of a sequence of identical BNs, Z, where (t=1, 2, 3 …), each Z represents a process that is modeled at time t and called a time slice [13]. While modeling DBN, we can introduce a discrete time stamp, it has a model for each unit of time, such local model is called time slice [38].

Time slices are connected by temporal links; if two time slices structures are identical and temporal links are same then the model is a repetitive temporal model. Furthermore, if conditional probabilities are also identical, then we called it as dynamic Bayesian network [38].

Suppose a domain consists of a set of n random variables X = \{X_1, X_2, \ldots, X_n\}, where each represents a node. While modeling changes over time, one node for each X represents a time slice. If current time slice is represented by t, then the previous time slice is represented by t-1 and next time slice by t+1 [39]. Such as

- Current: \{X_1^t, X_2^t, \ldots, X_n^t\}
- Previous: \{X_1^{t-1}, X_2^{t-2}, \ldots, X_n^{t-1}\}
- Next: \{X_1^{t+1}, X_2^{t+2}, \ldots, X_n^{t+1}\}
Figure 7 shows a general structure of dynamic Bayesian network, with a sequence of static BNs connected with inter-slice arcs. Nodes that have the link between the two time slices are known as link nodes [13].

Figure 7: Dynamic Bayesian network general structure (adopted from [39])
1.5 EFFORT ESTIMATION

In this section, we have discussed software effort estimation and its classical methods. In addition, test effort estimation methods are also discussed.

1.5.1 SOFTWARE EFFORT ESTIMATION

A good estimation is a combination of better understanding, range of tools, techniques and expert judgment [43]. Software effort can be measured in terms of working hours and number of workers.

Software effort can be defined by an equation [44][45].

\[ \text{Effort} = \text{people} \times \text{time} \]

One study showed that 30 to 70 percent of effort and cost estimates were incorrect [46]. It was observed that the 60 to 80 percent projects run into effort and schedule overruns [47]. Estimates can be failed due to [46]:

- Lack of historical data
- Overoptimistic leadership, fail to estimates on a solid foundation.
- Fail to implement the estimates

1.5.2 CLASSICAL METHODS FOR SOFTWARE ESTIMATION

There are a number of classical methods or techniques introduced by different authors. Few of them are discussed below:

Fenton and Pfleeger [48] classified the estimation techniques into four different categories:

- Expert opinion
- Analogy
- Decomposition
- Models

Top-down and bottom-up are the two ways to implement above techniques.

Boehm et al. [49] has divided the cost and estimation techniques in six different categories:

- Model based
  This technique is useful in budgeting, planning, controlling, tradeoff analysis, and investment analysis. It includes the SLIM, COCOMO, Check point, SEER. However, these approaches do not worth to deal with project effort estimation in recent technologies for example agile.
- Expertise-based techniques
  This technique is helpful when the quantified empirical data is not present. Techniques use the knowledge and experience of practitioner from their interested
domain. It involves Delphi, Rule-based. However, in these approaches only expert judgments are involved that can be biased or wrong.

- **Learning-oriented techniques**
  It is a collection of both new and old techniques that can be applied to estimation activities. The old techniques are known as *case studies* while later techniques are known as *neural networks*. Thus it involves case-based and neural techniques. In 1995, Witting has reported that this kind of techniques provide only ten percent results accuracy. These techniques also require large amount of historical data to make estimation or prediction.

- **Dynamics-based models**
  This technique acknowledged that the software project effort and cost factors changes during the software development. These factors may include deadline, staffing level, design requirements, training needs and budget. All of these factors are dynamic over time. *Abdul Hamid Madnick* is one of the technique involved in this approach. Sometime these techniques are sometime good for planning and controlling, however, difficult to calibrate.

- **Regression-based models**
  Regression-based models are the popular way to build estimation models. These techniques can be used in combination with model based techniques. It involves OLS, Robust techniques. However, it requires large amount of data, no data items should be missing.

- **Composite-Bayesian techniques**
  The composite techniques provide a way to combine two or more techniques. For example, Bayesian and COCOMO-II can be used together. Although the above discussed models, methods or techniques can be used to estimate effort. However their application is limited and not suitable when iterative development is considered [11].

Galorath and Evans [46] has defined the estimation models as

- Analogy
- Expert judgment
- Top-down
- Bottom-up
- Design to cost
- Parametric model

Table 2 shows the estimation methods with their objectives, advantages and limitations [46].

<table>
<thead>
<tr>
<th>Methods</th>
<th>Objectives</th>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analogy</td>
<td>Make comparison with similar past projects</td>
<td>It is based on experience</td>
<td>Projects should be similar</td>
</tr>
<tr>
<td>Expert judgment</td>
<td>Expert opinion can be taken from one or more experts</td>
<td>Little or no historical data required. Good for new and different projects</td>
<td>Experts knowledge level can be questionable, experts opinion can be biased</td>
</tr>
<tr>
<td>Top-down</td>
<td>Decomposition of a system into a sub-systems or component used to estimate the size</td>
<td>Requirements can be linked with estimates</td>
<td>It required valid requirements, architectural tracing issue</td>
</tr>
</tbody>
</table>

Table 2: Estimation methods with objectives, advantages and limitations
During the past years, few techniques or methods were introduced for size, complexity, effort and cost estimation. Most commonly used approaches for estimation are discussed here.

**Constructive Cost Model (COCOMO)**, is one of the most famous estimation model. It converts Function Point (FP) and Source Line of Code (SLOC) into effort estimation to develop a system. Its formula uses effort multipliers and scale factors, it is calculated according to the project characteristics such as environment, teams and procedures [50][49].

**Function Point Analysis (FPA)** is a model based estimation technique. It is used to measure size of the system by measuring system complexity and counting function points (FP). FP count is used to determine effort to develop a system. Function point counts can be compared at the end with the requirements, analysis, design, code and testing. It requires detailed system information, experts for calculations that is relatively high cost [50][49]. FP is subjective approach and depends on estimator [51].

**Test Point Analysis (TPA)** is similar to FPA used to estimate effort to define, develop and execute functional tests [52].

**Use Case Point Analysis (UCP)**, is based on use case specification; it is an extension of FPA. UCP also consider the development system complexity [49]. UCP based techniques takes testing activities as a whole such as test planning, design, execution and review, it does not consider the separate estimates for each test activity or phase [50].

Most of above discussed traditional approaches are not suitable for more recent software development methodologies such as agile or incremental approaches[11][21]. As COCOMO is heavily depends on many project specific settings and adjustments, also its impact is difficult to collect, asses, and quantify [21]. Therefore, an affective collection of metrics is difficult and unrealistic in iterative development.

### 1.5.3 TEST EFFORT ESTIMATION

There are number of estimation models [44][53] exist but still no one gained universal acceptance. Many of them uses the traditional approaches such as use cases [54], test specification [52], Test point analysis (TPA)[52], test suite [45] to estimate test effort.

In 2007, Aranha and Borba presented test execution effort model based on test specification, test cases measure of size and execution complexity is defined and validated. They evaluated the model through an empirical study on mobile application domain [49].

DeMarco defines an estimation is a prediction based on a probabilistic assessment, he identified that lack of experience is the reason for poor estimation, he proposed metrics groups for data collection and estimation [55].
Chulani et al. [56] compared the Bayesian and multiple regression approaches. They concluded that the Bayesian approach was better and more robust than the multiple regression. A good decision making was difficult and challenging through incomplete and scarce data, they concluded that predictive performance of Bayesian was significantly better than the multiple regression method. However, they did not proposed a Bayesian model suitable to estimate test effort.

Bayesian network (BN)[21][20][13] has been used to model the uncertainties and software effort estimation. Wang et al. developed a project level estimation framework using Bayesian network [20]. It uses component estimation, test effectiveness estimation, residual defect estimation and test estimation as four sub models. The presented test estimation model is not detailed and not validated with any real project.
1.6 LITERATURE REVIEW – AN OVERALL SUMMARY

In this section, we have provided an overall summary of the literature review including the steps such as inputs, processing and outputs.

In order to conduct an effective literature review we have partially followed an approach suggested by Ellis and Levy [57]. They defined three steps to conduct an effective literature review such as inputs, processing and outputs.

1.6.1 INPUTS
This step involves the selection of database and electronic resources, keywords, inclusive and exclusive criteria.

1.6.1.1 Databases and electronic resources
During the literature review, following online resources were used:

- IEEE Xplore Digital Library
- ACM Digital Library
- Direct Science
- Springer Link
- Compendex
- Inspec

1.6.1.2 Keywords search
The following keywords or search terms were used to identify primary studies. However, we have not described all keywords and their possible combinations here.

1. Software effort estimation
2. Software testing / Testing
3. Iterative development
4. Bayesian network
5. 1 AND challenges
6. 1 AND problems
7. 1 AND issues
8. 2 AND challenges
9. 2 AND problems
10. 2 AND issues
11. 3 AND challenges
12. 3 AND problems
13. 3 AND issues
14. 3 AND ( 5 OR 6 OR 7)
15. 3 AND ( 8 OR 9 OR 10)
16. 1 AND 3
17. 16 AND 4
18. 17 AND 2
19. 1 AND ( Model OR Techniques OR Methods)
20. 19 AND 2
21. 20 AND 3
22. 1 AND 4
23. 2 AND 3
1.6.1.3 Selection criteria
The study selection criteria involve the inclusive and exclusive criteria.

1.6.1.3.1 Inclusive criteria
Following are the inclusive criteria that helped to identify right studies or articles. To check the existing effort estimation models or frameworks is one of the basic criteria.

- The article should be available as full text.
- The article should be related to somehow software testing, effort estimation, iterative development, and Bayesian network.
- The article may contain a literature review, systematic review, case study, survey, experiment, or comparative study.
- The article will be included if it contains models or techniques or methods for effort estimation.
- The article will be included if it contains test effort estimation models or techniques.
- The article will be included if it identifies the challenges or problems in software testing.
- The article will be included if it identifies the challenges in iterative projects testing processes.
- The article will be included if it identifies the challenges in test effort estimation.
- The article will be included if it identifies the challenges in test effort estimation in iterative development.

1.6.1.3.2 Exclusive criteria satisfy
The articles are not included that do not fulfill the inclusive criteria.
1.6.2 PROCESSING

In this section, we have identified, applied, analyzed and evaluated the existing literature on effort estimation models or techniques, iterative development, and Bayesian network.

In order to search and explore the existing effort estimation models, we have used different combinations of the above mentioned keywords and searched in different databases such as IEEE, ACM, etc. Many articles or studies were found during this process. After reading the title, abstract, conclusion and applying inclusive/exclusive criteria many articles were excluded.

Table 3 provides a list of estimation models that were selected by reading the title, abstract, conclusion and considering the inclusive/exclusive criteria.

Table 3: List of articles selected by considering title, abstract, inclusive/exclusive criteria

<table>
<thead>
<tr>
<th>Sr.#</th>
<th>Ref</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[10]</td>
<td>Quantitative defects management in iterative development with BiDefect</td>
<td>2009</td>
</tr>
<tr>
<td>5</td>
<td>[13]</td>
<td>Predicting project velocity in XP using a learning dynamic Bayesian network model</td>
<td>2009</td>
</tr>
<tr>
<td>6</td>
<td>[14]</td>
<td>Managing the uncertainties of software testing: a Bayesian approach</td>
<td>2001</td>
</tr>
<tr>
<td>7</td>
<td>[15]</td>
<td>Bayesian graphical models for software testing</td>
<td>2002</td>
</tr>
<tr>
<td>8</td>
<td>[16]</td>
<td>Software measurement: uncertainty and causal modeling</td>
<td>2002</td>
</tr>
<tr>
<td>9</td>
<td>[58]</td>
<td>A critique of software defect prediction models</td>
<td>1999</td>
</tr>
<tr>
<td>10</td>
<td>[20]</td>
<td>Software project level estimation model framework based on Bayesian belief networks</td>
<td>2006</td>
</tr>
<tr>
<td>11</td>
<td>[21]</td>
<td>Effort prediction in iterative software development processes -- incremental versus global prediction models</td>
<td>2007</td>
</tr>
<tr>
<td>12</td>
<td>[22]</td>
<td>Software effort estimation: planning XP guidelines compared to research on traditional software development</td>
<td>2003</td>
</tr>
<tr>
<td>13</td>
<td>[35]</td>
<td>COCOMO-based effort estimation for iterative and incremental software development</td>
<td>2003</td>
</tr>
<tr>
<td>14</td>
<td>[44]</td>
<td>Test effort estimation models based on test specifications</td>
<td>2007</td>
</tr>
<tr>
<td>15</td>
<td>[45]</td>
<td>Estimate test execution effort at an early stage: an empirical study</td>
<td>2008</td>
</tr>
<tr>
<td>16</td>
<td>[52]</td>
<td>An experience-based approach for test execution effort estimation</td>
<td>2008</td>
</tr>
<tr>
<td>17</td>
<td>[53]</td>
<td>A multiple-case study of software effort estimation based on use case points</td>
<td>2005</td>
</tr>
<tr>
<td>18</td>
<td>[56]</td>
<td>Bayesian analysis of empirical software engineering cost models</td>
<td>1999</td>
</tr>
<tr>
<td>19</td>
<td>[59]</td>
<td>Predicting software defects in varying development lifecycles using Bayesian nets</td>
<td>2006</td>
</tr>
<tr>
<td>20</td>
<td>[60]</td>
<td>Bayesian statistical effort prediction models for data-centred 4GL software development</td>
<td>2006</td>
</tr>
</tbody>
</table>
1.6.2.1 Apply the literature

This step involves the activities to classify, solve, and relate the studies with different concepts. We have classified the studies depending on different concepts such as project level effort estimation, testing level estimation, iterative, and etc. Table 4 provides a complete matrix between the concepts and studies, where the most left column (Sr. #) shows the article number from Table 3 while the concepts are shown vertically (top rows).

<table>
<thead>
<tr>
<th>Sr. #</th>
<th>Project level effort estimation</th>
<th>Test effort estimation</th>
<th>Iterative</th>
<th>Bayesian network</th>
<th>Industrial validation</th>
<th>Statistical validation</th>
<th>Any prediction accuracy measures</th>
<th>Proposed a model(s)</th>
<th>Defect prediction or management</th>
<th>Size estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>15</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From Table 4, we can see that in most of the studies, a new model or framework is presented. All of these models focused on either project or testing level effort estimation. Few of them have used Bayesian and iterative approaches.

Further, few of them have used industrial, statistical and prediction accuracy measures. However, no model has been found that can be used to predict test effort in iterative development.

Most of the above mentioned studies are discussed and analyzed in prior sections such as Section 1.1 and 1.5. The majority of these studies discussed the software development cost
and effort at project level using traditional approaches such as COCOMO, FPA, TPA, UCP, etc.

1.6.2.2 Analyze and evaluate

In analyze and evaluate step, different articles or studies were selected to compare, explain and evaluate. After reading and understanding the articles, we have selected three most relevant articles that are related to effort estimation, iterative development and Bayesian network.

In 2006, Wang et al. [20] presented a project level estimation model for Siemens Company. They included the following factors in their sub-model.

- Test duration
- Average number of people
- Test effort
- Test case number
- Requirement number
- Test case density
- Total test case effort
- Test execution effort
- Test design effort
- Testing effectiveness
- Test team experience

However, most of the above factors such as test execution, test design effort are not estimated in iterative projects. The interviewees also verified that they do not estimate the effort for test designing and execution separately. Further, their proposed model was not validated through any project data. Moreover, no statistical and prediction accuracy measures were performed.

In 2006, Fenton et al. [59] have identified the following factors in their testing and rework sub-model.

- Rework process overall effectiveness
- Rework process quality
- Rework effort
- Test process quality
- Quality of documented test cases
- Testing process well defined
- Testing staff experience
- Test process overall effectiveness

In 2009, Hearty et al. [13] have presented another framework that aims to predict project velocity in iterative projects (such as XP). Their presented model contains the following factors.

- Process effectiveness
- Process improvement
- Effectiveness limit
- Actual effort
- Iteration effort
- Predicted project velocity
However, the model was validated with a single project data, only two XP practices were introduced into the model. They focused on project level predictions rather than specific testing processes.

Following are the most common factors that are used in the above three models:

- Process effectiveness
- Rework process overall effectiveness
- Test team experience
- Test process quality
- Effort

In order to estimate effort, people and time are of high importance [20][44][45], since effort is the multiple of these factors. Further, process effectiveness can also be an important factor that can affect the predicted development effort of the iteration [13]. Therefore, we have decided to include the test team size, iteration length and process effectiveness as sub-parts of our propose framework. However, after the deep analysis and understanding of the above discussed models and factors[13][20][59], we have compiled a list of factors that are logically related to each other, and make sense to predict test effort. These factors include the following:

- Iteration length
- Test team size
- Test process overall effectiveness
- Rework test effectiveness
- Rework test effort
- Rework test process quality
- Test process effectiveness
- Test process quality
- Test tool quality
- Test team experience
- Test case effectiveness
- Number of bugs found by test cases
- Total number of bugs found

These factors provide basis to structure the framework design. The details of each factor or variable are provided in Section 2.

1.6.3 OUTPUT

This section aims to write and discuss the literature findings. However, we have discussed the literature in prior sections such as Section 1.1, 1.3, 1.4 and 1.5.

In the next section, we have presented a detailed design of the proposed framework.
2 PROPOSED FRAMEWORK DESIGN

In this section, we have explained the proposed framework design including notations, parameters and nodes.

2.1 FRAMEWORK TOOLS

There are a number of tools available that can be used for Bayesian network modeling, designing and implementation. We have used a powerful software package *AgenaRisk v5.0* toolset to design the proposed framework. It helps the decision makers in resolving risky and complex problems by providing features such as predictive analysis, risk monitoring and assessment. The AgenaRisk toolset is recommended and used by many researchers in different areas of software engineering [13][20].

2.2 FRAMEWORK REQUIREMENTS

The following requirements must be met by the framework

- To minimize the framework complexity, the framework should be small and simple because it will be replicated in all iterations.
- The model should be able to learn from projects data and expert judgment.
- The model should be able to respond to any observation when entered into the model. For example, it will affects the adjacent nodes or variables
- The model should be able to learn in different scenarios such as project failing, average, or success.
- The model should be able to predict test effort by considering impact of different factors such as test process effectiveness, test team, test tools, etc
- The model should be able to handle different kind of data such as integer, rank values.

2.3 FRAMEWORK LIMITATIONS

The framework has the following limitations

- Framework only deals with the iterative projects
- Framework is unable to predict test effort for the first iteration
- Framework requires Nth iteration data to predict for N+1, N+2, … N+n iterations
- Predictions can be valid for the same project whose data is incorporated into the model.

2.4 FRAMEWORK DESIGN

Fenton and Pfleeger [48] has described the software measurements as follow

- **Processes**
  It is related to software activities such as development and support.
- **Products**
  It involves the deliverables such as requirements, design, code and etc.
- **Resources**
  It involves the assets such as people, tools, equipments and etc.

---

According to Fenton and Pfleeger [48], all predictive and estimation models fall within these three classes. In our case, we have used processes and resources measurement. To minimize the framework complexity, we have not used products measurements.

The proposed Bayesian network model is composed of two main sections i.e. test process overall effectiveness \((e)\) and test effort \((E)\), where the Test effort \((E)\) node is controlled by iteration length \((l)\), team size \((s)\) and test process overall effectiveness \((e)\).

**Test process overall effectiveness** contains two fragments that is test process rework effectiveness and test process effectiveness, these fragments control the test process overall effectiveness. Further, rework test effectiveness is controlled by two nodes such as rework test effort and rework test process quality nodes. In fragment 2, test process effectiveness is controlled by test process quality which is further controlled by test tool quality, test team experience and TC effectiveness nodes. However, TC effectiveness is controlled by number of bugs found by TC and total number of bugs found nodes.

The previous existing research work has been used to build the proposed framework, more detailed information can be found in [13][20][59]. Our propose model is different from the other existing models as discussed in Section 1.1 and 1.6. Rather than creating a complex and large model, we have constructed a simple and small model with few important nodes or variables. It helps to make the computations easy and flexible.

Figure 9 shows how different nodes are linked with each other in the model.

![Figure 9: Proposed Framework design](image-url)
2.5 FRAMEWORK NOTATIONS

We have used different notations and symbols to represent model nodes, such as capital $E$ for Test effort, small $e$ for test process overall effectiveness, and so on, details are given in Table 5. Subscripts are used with node symbols to represent a specific iteration number. For example $E_3$, $t_4$ represent test effort and team size for iteration number 3 and 4 respectively.

Table 5: Node symbol, title, formula and range values

<table>
<thead>
<tr>
<th>Node symbol</th>
<th>Node title</th>
<th>Formula</th>
<th>Node type</th>
<th>Range / values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$rte_i$</td>
<td>Rework test effort</td>
<td></td>
<td>Ranked</td>
<td>enough, not enough, more than enough</td>
</tr>
<tr>
<td>$rtq_i$</td>
<td>Rework test process quality</td>
<td></td>
<td>Ranked</td>
<td>very low to very high*</td>
</tr>
<tr>
<td>$re_i$</td>
<td>Rework test effectiveness</td>
<td>$re_i = \frac{(rte_i + rtq_i)}{2}$</td>
<td>Ranked</td>
<td>As above</td>
</tr>
<tr>
<td>$ttq_i$</td>
<td>Test tool quality</td>
<td></td>
<td>Ranked</td>
<td>As above</td>
</tr>
<tr>
<td>$tte_i$</td>
<td>Test team experience</td>
<td></td>
<td>Ranked</td>
<td>As above</td>
</tr>
<tr>
<td>$tpe_i$</td>
<td>Test process effectiveness</td>
<td></td>
<td>Ranked</td>
<td>As above</td>
</tr>
<tr>
<td>$tpq_i$</td>
<td>Test Process quality</td>
<td>$tpq_i = wmean(1,ttq_i,5.0,tte_i,1.0,tpe_i)$</td>
<td>Ranked</td>
<td>As above</td>
</tr>
<tr>
<td>$b_i$</td>
<td>Number of bugs found by TC</td>
<td></td>
<td>Numeric</td>
<td>[0-100]</td>
</tr>
<tr>
<td>$tb_i$</td>
<td>Total number of bugs</td>
<td></td>
<td>Numeric</td>
<td>[0-100]</td>
</tr>
<tr>
<td>$tece_i$</td>
<td>Test case effectiveness</td>
<td>$tece_i = \frac{b_i}{tb_i} \times 100$</td>
<td>Numeric</td>
<td>[0-1]</td>
</tr>
<tr>
<td>$te_i$</td>
<td>Test process overall</td>
<td>$te_i = \frac{(re_i + tpe_i)}{2}$</td>
<td>Numeric</td>
<td>[0-1]</td>
</tr>
<tr>
<td>$l_i$</td>
<td>Iteration length</td>
<td></td>
<td>Numeric</td>
<td>[1-30]</td>
</tr>
<tr>
<td>$s_i$</td>
<td>Test team size</td>
<td></td>
<td>Numeric</td>
<td>[1-15]</td>
</tr>
<tr>
<td>$E_i$</td>
<td>Test effort</td>
<td>$E_i = l_i \times s_i \times e_i$</td>
<td>Numeric</td>
<td>[0-500]</td>
</tr>
</tbody>
</table>

Two types of nodes are used in the framework design such as numeric and ranked. The numeric node can have different numeric values such as 0, 1, 2, and etc. While the ranked type represent discrete variables where the states are expressed on an ordinal scale such as an interval [0-1]. For example, we have five-point scale such as {very low, low, average, high, very high}, each state has an interval of 0.2. Therefore, "very low" value lies within the interval [0 - 0.2] or 0 to 20 percent, "low" value lies within the interval [0.2 -0.4) or 20 to 40 percent and so on. AgenaRisk tool facilitates to compute the probabilities of ranked nodes. More detailed information on Raked nodes can be found in [61].

The details of the above mentioned nodes and formulas can be found in Section 2.7.

2.6 FRAMEWORK PARAMETERS

There is a number of key input parameters used to enter project data, observation or expert judgment into the model. These parameters involve the nodes such as rework test effort ($rte_i$), rework test process quality ($rtq_i$), test tool quality ($ttq_i$), test team experience ($tte_i$), number of bugs found by TC ($b_i$), total number of bugs found ($tb_i$), iteration length ($l_i$) and team size ($s_i$). In this way the model can learn from project data.
2.7 FRAMEWORK NODES

This section describes the nodes that are used in the framework design, each node details are as follows.

2.7.1 REWORK TEST EFFORT

Rework test effort is repetitive, caused by many factors such as changed requirement, environmental factors, design constraints, etc [62]. Rework effort can be involved in fixing and regenerating bugs, setting test suites, writing test specification, executing test cases etc.

2.7.2 REWORK TEST PROCESS QUALITY

Rework test process quality concerns the quality of the activities performed during test rework process. It considers quality of test suite, test tools, test team etc.

2.7.3 REWORK TEST EFFECTIVENESS

Rework test effectiveness concerns effectiveness of rework test process. It is controlled by two nodes or parameters such as rework test effort and rework test process quality. It is an average of rework test effort and rework test process quality, it is formulated as

\[ r_{te_i} = \frac{rte_i + rtq_i}{2} \]

Where
- \( rte_i \) represents rework test effort for \( i^{th} \) iteration
- \( rtq_i \) represents rework test process quality for \( i^{th} \) iteration.

2.7.4 TEST TOOL QUALITY

Test tool quality node concerns quality of tools that are used for test case design, test case execution, test specification, marking bugs, etc. It provides quality level of tools used in test process. Test tool quality can affect the test process quality. During the interviews, we found that companies are using different testing tools e.g. N-unit and selenium.

2.7.5 TEST TEAM EXPERIENCE

It is important to consider test team experience (staff skills level) in estimation process [46]. Test team experience node can affect the test process quality; it determines how much the test team is capable to perform test activities. High team experience can improve the test process quality.

2.7.6 TEST PROCESS QUALITY

Test process quality concerns the quality of test tools, test team, test environment, etc. Test process quality is controlled by test team experience, test tool quality and TC effectiveness. It is the weighted mean of test tool quality, test team experience, and test process effectiveness. It is formulated as follows

\[ tpeq_i = \text{wmean}(1, ttq_i, 5.0, tte_i, 1.0, tpe_i) \]

Where
- \( ttq_i \) represents test tool quality
- \( tte_i \) represents test team experience, and
- \( tpe_i \) represents test process effectiveness.
2.7.7 **NUMBER OF BUGS FOUND BY TEST CASES**

This node concerns the number of bugs found by test cases. It controls the test case effectiveness node.

2.7.8 **TOTAL NUMBER OF BUGS FOUND**

This node determines the total number of bugs found during the function test cycle. It is the sum of bugs found by TC and bugs found as side effect or mistake [63]. It also controls the test case effectiveness.

2.7.9 **TEST CASE EFFECTIVENESS**

Test case effectiveness is an important factor to determine the overall test process effectiveness. It also determines how effective is the test case design, execution phases. Its range is from 0 to 1, where 0 represents ineffectiveness while 1 represents effectiveness of test case. We have calculated test case effectiveness by a metric provided by Chernak [63].

The test case effectiveness is formulated as

\[ tce_i = \frac{b_i}{tb_i} \times 100 \]

Where
- \( b_i \) represents number of bugs found by test cases for \( i^{th} \) iteration
- \( tb_i \) represents total number of bugs for \( i^{th} \) iteration.

2.7.10 **TEST PROCESS EFFECTIVENESS**

It provides information about the test process effectiveness. It is controlled by the test process quality node. Test process effectiveness is estimated on the basis on five different values: very low, low, medium, high and very high.

2.7.11 **TEST PROCESS OVERALL EFFECTIVENESS**

It gives the overall effectiveness of the test process including rework effectiveness. It is an average of rework test effectiveness and test process effectiveness, and formulated as

\[ e_i = \frac{re_i + tpe_i}{2} \]

Where
- \( re_i \) represents rework test effectiveness for \( i^{th} \) iteration
- \( tpe_i \) represents test process effectiveness for \( i^{th} \) iteration.

2.7.12 **ITERATION LENGTH**

It is the length of an iteration determined in number of working days such as 5 days, 10 days, etc. It can affect the test effort node.

2.7.13 **TEAM SIZE**

Time size is determined by the number of persons involved in each iteration, which may affect test effort node. For example, first and second iteration may have three and six persons respectively.
2.7.14 Test Effort

It determines the total test effort required for overall test process by considering the test process effectiveness, iteration length and team size. It is multiple of iteration length, test team size, and test process overall effectiveness. It can be formulated as

\[ E_i = l_i \times s_i \times e_i \]

Where
- \( l_i \) represents iteration length for \( i^{th} \) iteration
- \( s_i \) represents test team size for \( i^{th} \) iteration.
- \( e_i \) represents test process overall effectiveness for \( i^{th} \) iteration.

Test effort is measured in terms of person-days. To predict test effort, we have used median values throughout the report.
2.8  FRAMEWORK FOR SINGLE ITERATION

The proposed framework can also be represented in symbolic form where each node is represented by a unique symbol. Figure 10 shows Framework symbolic representation for a single iteration.

![Figure 10: Framework design for a single iteration](image)

2.9  FRAMEWORK INTERACTION IN MULTIPLE ITERATIONS

In iterative development, tasks are split into number of discrete iterations where the iteration is completed at time t. To model such iterations that changes over time, dynamic Bayesian network has the capability to handle such a situation.

Therefore, we have used dynamic Bayesian network to construct the framework, DBN is a collection of Bayesian network as discussed in prior sections. The term "dynamic" means to model a dynamic system, where the model or network design will not change over time [64]. Each model is modeled at time t, and known as time slice. The proposed DBN Framework is shown in Figure 11, it shows link between two iterations, each iteration is known as time slice. The nodes that create links between two iterations are known as link nodes and they are linked by temporal links [38]. For example, \( rte_1 \) and \( rqe_1 \) are the two link nodes between iteration 1 and 2.

![Figure 11: Framework for multiple iterations](image)
3 FRAMEWORK BEHAVIOR AND INITIAL VALIDATION

After the completion of the framework design, it is important to test the framework by performing a pilot test. Therefore, we have evaluated the framework by observing the framework behavior through parameters learning in different scenarios. Further, to observe and validate the framework prediction abilities, initial validation was performed by incorporating data from Motorola project into the framework. In this section, we have provided details on framework evaluation tools, framework behavior, and initial validation.

3.1 FRAMEWORK EVALUATION TOOLS

As we have used a software package AgenaRisk v5.0 toolset for framework designing purposes, similarly, we have also used the same software package for evaluating the framework behavior, parameters learning and performing initial validation.

3.2 FRAMEWORK BEHAVIOR

To test the framework behavior, the authors have assumed that we have a project X consisting of total eight iterations with 120 hours of available effort for a single iteration. The framework was solidly based on initial settings, where no data or observations were entered into the framework. The model was executed and the Test effort (E) output parameter mean, median and standard deviation (SD) values were observed to evaluate the model predicting behavior. Table 6 shows the mean, median, standard deviation (SD) values for all iterations (I1 to I8). These values show our baseline scenario, where we have not entered any observation into the model.

Table 6: Framework behavior - Baseline scenario based on initial settings

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Mean</th>
<th>Median</th>
<th>SD</th>
<th>Mean - SD</th>
<th>Mean + SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>50.92</td>
<td>35.65</td>
<td>48.57</td>
<td>2.35</td>
<td>99.49</td>
</tr>
<tr>
<td>I2</td>
<td>53.06</td>
<td>37.22</td>
<td>50.5</td>
<td>2.56</td>
<td>103.56</td>
</tr>
<tr>
<td>I3</td>
<td>54.35</td>
<td>38.16</td>
<td>51.68</td>
<td>2.67</td>
<td>106.03</td>
</tr>
<tr>
<td>I4</td>
<td>55.23</td>
<td>38.78</td>
<td>52.51</td>
<td>2.72</td>
<td>107.74</td>
</tr>
<tr>
<td>I5</td>
<td>55.89</td>
<td>39.24</td>
<td>53.13</td>
<td>2.76</td>
<td>109.02</td>
</tr>
<tr>
<td>I6</td>
<td>56.42</td>
<td>39.62</td>
<td>53.65</td>
<td>2.77</td>
<td>110.07</td>
</tr>
<tr>
<td>I7</td>
<td>58.89</td>
<td>39.94</td>
<td>54.9</td>
<td>3.99</td>
<td>113.79</td>
</tr>
<tr>
<td>I8</td>
<td>57.3</td>
<td>40.23</td>
<td>54.47</td>
<td>2.83</td>
<td>111.77</td>
</tr>
</tbody>
</table>

SD = Standard deviation

With the help of Table 6, a graph was generated as shown in Figure 12. It shows mean, median, mean – SD, and mean + SD as baseline scenarios. The key parameters of the model such as iteration length (l), test team size (s) and test process overall effectiveness (e) allows the test effort graph to increase gradually with each iteration. To model such kind of behavior was one of the main objectives of the model.
A number of different scenarios can be generated by entering different observations into the Test effort \((E)\) parameter of the model. In this way, we can evaluate the predicted and learned values of Test effort future iterations. This kind of parameter learning is discussed in the next section.

### 3.2.1 Framework Parameters Learning

To evaluate the framework behavior, it is important to consider framework parameters under different scenarios. Therefore, we have created three different scenarios in the model such as success, average and failing. The success scenario represents that the project is making progress, while the average scenario represents that the project is going on as likely, and the failing scenario represents project failing.

The baseline scenarios predicted values for iteration one to eight \((I_1 \text{ to } I_8)\) lies in the range of 35.65 to 40.23 as shown in Table 6. However, we have assumed different values for test effort \((E)\) under success, average and failing scenarios for iterations \((I_1 \text{ to } I_6)\) in the range of 42 to 50, 34 to 41 and 2 to 6 respectively. Table 7 shows the test effort values in success, average and failing scenarios.

<table>
<thead>
<tr>
<th>Iterations ((I))</th>
<th>Test effort ((E))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>(I_1) (E_1)</td>
<td>42</td>
</tr>
<tr>
<td>(I_2) (E_2)</td>
<td>44</td>
</tr>
<tr>
<td>(I_3) (E_3)</td>
<td>44</td>
</tr>
<tr>
<td>(I_4) (E_4)</td>
<td>47</td>
</tr>
<tr>
<td>(I_5) (E_5)</td>
<td>48</td>
</tr>
<tr>
<td>(I_6) (E_6)</td>
<td>50</td>
</tr>
<tr>
<td>(I_7) (E_7)</td>
<td>-</td>
</tr>
<tr>
<td>(I_8) (E_8)</td>
<td>-</td>
</tr>
</tbody>
</table>

To observe model parameter learning, all values from Table 7 were entered into the model, no values for iterations 7 and 8 \((I_7 \text{ & } I_8)\) were entered, allowing the model to predict Test effort \((E)\) for these iterations. The model was executed and parameter learning was observed in different scenarios, as a result graph of baseline, success, average and failing scenarios was generated (Figure 13).
From the graph shown in Figure 13, we can see that *average* scenario graph is very close to the baseline scenario and begins to stabilize by iteration $I_5$. The *failing* scenario shows worst case, the graph is far away from baseline scenario. The low values of test effort for failing scenario can be a reason, however the model tries to learn from data and improve the graph of *Test effort* ($E$) gradually. While in *success* scenario, Test effort ($E$) parameter learned from the observations, predicted values for $I_7$ and $I_8$ are very stable that are $E_7=39.07$ and $E_8=39.51$ as compared to average and failing scenarios. Therefore, we can say that the framework is able to learn from data.

![Test effort (E) with different scenarios](image)

Figure 13: Test effort parameter learning in baseline, success, average, failing scenarios.

*Test process overall effectiveness* node was also observed against the parameter learning. In Figure 14, different scenarios for test process overall effectiveness nodes are shown, failing scenario shows no improvement until fifth iteration ($I_5$), it means no effectiveness improvement, after $I_5$ graph gradually improves. Success scenario shows high effectiveness as compared to average and failing scenarios, also its predicted values for $I_7$ and $I_8$ are stable.

![Test process overall effectiveness (e)](image)

Figure 14: Test process overall effectiveness parameter learning in different scenarios

As we have seen that *Test effort* ($E$) and *Test process overall effectiveness* ($e$) propagates as observations were entered, they updated their prediction accordingly. Similarly different scenarios for other nodes such as rework test effort effectiveness ($re$), test process effectiveness ($tpe$), etc can also be generated.
3.3 FRAMEWORK INITIAL VALIDATION

This section provides details on the initial validation of the framework, in order to check the model for predicting abilities we have conducted an initial validation by incorporated data from Motorola project.

We have used the data from Motorola project found in [13]. This project consists of total eight iterations varying from two to three weeks. Three to nine people were involved in the project, a complete project dataset is given in Table 8 below.

<table>
<thead>
<tr>
<th>Iterations (I)</th>
<th>Iteration length (l)</th>
<th>Team size (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>I₂</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>I₃</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>I₄</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>I₅</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>I₆</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>I₇</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>I₈</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

3.3.1 PREDICTION WITH AND WITHOUT OBSERVATION

Initially, we have created two scenarios that are actual and predicted. We have assumed different values for the Test team experience parameter as shown in Table 9. From Table 9, Iteration length (l), team size (s) and test team experience (tte) values were entered into the model. The model was executed without entering any additional observations, and results were observed (Table 10).

<table>
<thead>
<tr>
<th>Iterations(I)</th>
<th>Iteration length (l)</th>
<th>Team size (s)</th>
<th>Test team experience (tte)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>15</td>
<td>3</td>
<td>High</td>
</tr>
<tr>
<td>I₂</td>
<td>15</td>
<td>3</td>
<td>High</td>
</tr>
<tr>
<td>I₃</td>
<td>15</td>
<td>6</td>
<td>Very high</td>
</tr>
<tr>
<td>I₄</td>
<td>16</td>
<td>6</td>
<td>Very high</td>
</tr>
<tr>
<td>I₅</td>
<td>12</td>
<td>7</td>
<td>Very high</td>
</tr>
<tr>
<td>I₆</td>
<td>10</td>
<td>7</td>
<td>Very high</td>
</tr>
<tr>
<td>I₇</td>
<td>8</td>
<td>9</td>
<td>Very high</td>
</tr>
<tr>
<td>I₈</td>
<td>10</td>
<td>4</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actual vs. predicted values without observation entered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
</tr>
<tr>
<td>I₁</td>
</tr>
<tr>
<td>I₂</td>
</tr>
<tr>
<td>I₃</td>
</tr>
<tr>
<td>I₄</td>
</tr>
<tr>
<td>I₅</td>
</tr>
<tr>
<td>I₆</td>
</tr>
<tr>
<td>I₇</td>
</tr>
<tr>
<td>I₈</td>
</tr>
</tbody>
</table>
In Figure 15, a graph of predicted and actual test effort values was generated using Table 10. As we can see the predicted graph is too far away from actual graph except the iterations $I_1$ and $I_2$. Therefore, we can say the predicted graph values are too high persistently as compared to the actual values.

![Graph showing prediction without any observation](image)

**Figure 15: Test effort prediction without entering any observations**

The objective of the proposed framework is to learn from the environment and project data in order to improve the prediction accuracy. To observe model learning effects, we took predicted scenario and entered previous iterations ($I_1$ and $I_2$) values. When a new piece of data or information is entered into the model, it updates probability distributions and these distributions influence the predictions for future iterations.

When we have entered *Test team experience* ($tte_1$, $tte_2$, $tte_3$) values into the model, predicted graph move lower closer to the actual values (Table 11 & Figure 16). However, the prediction for all remaining iterations ($I_3$, $I_4$, $I_5$, $I_6$, $I_7$, and $I_8$) improved after the insertion of observations. Therefore, we can say that the model can learn from observations or data and improves the predictions as a result.

<table>
<thead>
<tr>
<th>Actual vs. predicted values without observation entered</th>
<th>Actual median</th>
<th>Predicted median</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>15.8</td>
<td>15.8</td>
</tr>
<tr>
<td>$I_2$</td>
<td>16.36</td>
<td>16.36</td>
</tr>
<tr>
<td>$I_3$</td>
<td>31.02</td>
<td>31.02</td>
</tr>
<tr>
<td>$I_4$</td>
<td>32.46</td>
<td>34.57</td>
</tr>
<tr>
<td>$I_5$</td>
<td>28.63</td>
<td>31.22</td>
</tr>
<tr>
<td>$I_6$</td>
<td>24.06</td>
<td>26.48</td>
</tr>
<tr>
<td>$I_7$</td>
<td>25.05</td>
<td>27.85</td>
</tr>
<tr>
<td>$I_8$</td>
<td>15.39</td>
<td>14.81</td>
</tr>
</tbody>
</table>
We have evaluated the framework behavior by considering the different parameters learning in different scenarios. It is shown that the model is able to predict in different scenarios, even in worst scenarios such as project failing, the prediction improves. Further, the results show that the model can learn from prior iterations data or information and improve the predictions for proceeding iterations. Therefore, we can say that the framework has the learning and predicting behavior.

In the next section, we have validated the framework by incorporating data from two industrial projects. However, the prediction results are evaluated and analyzed by performing prediction accuracy measurements and statistical tests.
4 FRAMEWORK INDUSTRIAL VALIDATION

In this section, the authors have provided details on the industrial validation of the framework. First, interviews were conducted from industry for data collection purpose and the proposed framework was validated through the collected real projects data. Second, validation results and analysis are discussed.

There are a number of ways that can be used to collect data such as observations, interviews, focus groups, surveys, etc [65]. In our case, interview approach was selected because it is much flexible to explore the subjects in greater details. It also opens new areas of inquiry that are not anticipated in the interview instrument (questionnaire), in this way we can have clear picture.

4.1 INDUSTRIAL INTERVIEWS

This section provides details on industrial interviews including companies, interviewees and projects information. The main objective of the interviews was to collect iterative projects data having minimum six iterations. To achieve this, a number of online and face-to-face interviews were conducted in two different companies.

The companies, interviewees and project names must be anonymous, therefore dummy names are used.

4.1.1 COMPANY A

Company A is a Swedish based company that deals with telecom corporations, embedded systems, mobile multimedia, and network solutions. Its origin is in 1996, more than ten years of experience in software development, SCRUM, continuous integration and test driven development. They provide agile development services in several domains such as mobile applications, multimedia, telecom networks, software tools and enterprise systems. Company A follows different practices for software development such as SCRUM, XP, TDD, and etc, most of them are iterative.

4.1.1.1 Interviewee A

Interviewee A is working in Company A as a project manager for last three years. He has six years of working experience in total. He worked as a SCRUM master in the investigated project.

Initially an online interview was conducted with interviewee A to investigate the completed project. The questionnaire design was sent to the interviewee via email before the commencement of the interview, several questions were asked during the interview session. It took about an hour to complete the first interview session. We had collected three iterations data in the first interview.

Another interview was arranged to remove ambiguities and collect more iterations data. During the second session, information about six iterations of the investigated project was gathered. However, the gathered data was analyzed and sorted, while several emails were exchanged during this process.

4.1.1.2 Project A

Project A was related to the company A, six people took part in this project. The project duration was around six months with thirteen iterations of different lengths. Iterative
development methodology was followed during the project execution. There were two products released at the end of project A.

4.1.2 COMPANY B
Company B is also a Swedish based Telecom Company, spread all over the world, leading provider of telecommunication equipments, mobile and network services. Its origin is back in 1876 and currently they have more than seventy thousand employees. The company plays an important role in communication world, also invested a lot in research and development. They provide low-cost communication services to their customers.

4.1.2.1 Interviewee B
Interviewee B belongs to company B. He has about ten years of total industrial experience; he worked as a test lead in the investigated Project B.

4.1.2.2 Project B
Project B belongs to company B, consisting of about fifteen iterations of different lengths. Two to three persons were involved in the test team during the project execution. Iterative development methodology was used during the project B.

After the analysis of the data gathered from company A and B regarding the project A and B, data was entered into the model to validate the proposed framework. The section below provides details on the framework validation through real projects data.
4.2 **INDUSTRIAL VALIDATION**

In this section, we have provided details on the validation of framework. The proposed framework was validated through two industrial projects data that is project A and B.

4.2.1 **PROJECT A**

This section describes how the project A data was incorporated with the framework to predict test effort for later iterations.

The information gathered from interviewee A regarding project A is shown in Table 12 below. This detailed information is compiled after completing several interviews with interviewee A.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration length</td>
<td>I₁  I₂  I₃  I₄  I₅  I₆</td>
</tr>
<tr>
<td>Test team size</td>
<td>10  10  10  10  10  10</td>
</tr>
<tr>
<td>Test tool quality</td>
<td>Low  Medium  Medium  High  High  High</td>
</tr>
<tr>
<td>Test team experience</td>
<td>Medium  Medium  High  High  Very high  Very high</td>
</tr>
<tr>
<td>Bugs found by TC</td>
<td>0  ~5  ~10  ~10  ~10  ~10</td>
</tr>
<tr>
<td>Total bugs found</td>
<td>0  5  10  10  10  10</td>
</tr>
<tr>
<td>Rework test effort</td>
<td>Enough  Enough  Enough  Enough  Not Enough  More than enough</td>
</tr>
<tr>
<td>Rework test process quality</td>
<td>Medium  High  High  High  Very high  Very high</td>
</tr>
</tbody>
</table>

There were two scenarios developed in the model that are actual and predicted. Initially from Table 12, only iteration length (I) and test team size (s) values of all six iterations were entered into both scenarios. However, other remaining variables from Table 12 that includes Test tool quality, test team experience, bugs found by TC, Total bugs found, rework test effort and rework test process quality values from iteration I₁ to I₆ were entered only into actual scenario.

After entering the information from Table 12 into the actual and predicted scenarios, model was executed and Test effort (E) node median values were observed for actual and predicted scenarios as shown in Table 13.

<table>
<thead>
<tr>
<th>Test effort actual vs. predicted values without observation entered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>I₁</td>
</tr>
<tr>
<td>I₂</td>
</tr>
<tr>
<td>I₃</td>
</tr>
<tr>
<td>I₄</td>
</tr>
<tr>
<td>I₅</td>
</tr>
<tr>
<td>I₆</td>
</tr>
</tbody>
</table>

With the help of Table 13, without entering observations into the predicted scenario, a graph for test effort (E) node was generated for both actual and predicted scenarios as shown in Figure 17. It shows that the first iteration I₁ the predicted values are too low and for the fifth iteration I₅ the predicted values are too high as compared to actual values. Further the actual graph seems to be unstable hence it is difficult to predict it. Frequent changes in test tool quality and test team experience values might be a reason for instability.
Now we need to look how the model can learn from real projects data and improve its predictions. To see model learning process, we took only predicted scenario and entered the first two completed iterations ($I_1$ and $I_2$) information from Table 12 into the model. As a new piece of data was entered, propagation takes place and allowing the distributions of key parameter to be updated. These key parameter distributions effect the prediction of future iterations (Table 14 & Figure 18).

**Table 14: Project A - actual vs. predicted test effort values with observations ($I_1$ & $I_2$) entered**

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Actual median</th>
<th>Predicted median</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>14.41</td>
<td>14.41</td>
</tr>
<tr>
<td>$I_2$</td>
<td>12.19</td>
<td>12.19</td>
</tr>
<tr>
<td>$I_3$</td>
<td>10.42</td>
<td>12.47</td>
</tr>
<tr>
<td>$I_4$</td>
<td>10.42</td>
<td>12.61</td>
</tr>
<tr>
<td>$I_5$</td>
<td>8.79</td>
<td>12.67</td>
</tr>
<tr>
<td>$I_6$</td>
<td>9.81</td>
<td>12.7</td>
</tr>
</tbody>
</table>

From Figure 18, we can see that predicted graph has changed, when we entered the iteration ($I_1$ and $I_2$) into the model. Further, the predicted graph values are high as compared to prior predicted values given in Table 13, instability of actual data and information provided by the interviewee A can be a reason for this. Moreover, we entered new completed iteration $I_3$ information into the model and executed it again. The parameter probabilities were updated and predictions were improved as a result (Table 15 & Figure 19).
Table 15: Project A - actual vs. predicted test effort values with observations ($I_3$, $I_2$ & $I_1$) entered

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Actual median</th>
<th>Predicted median</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>14.41</td>
<td>14.41</td>
</tr>
<tr>
<td>$I_2$</td>
<td>12.19</td>
<td>12.19</td>
</tr>
<tr>
<td>$I_3$</td>
<td>10.42</td>
<td>10.42</td>
</tr>
<tr>
<td>$I_4$</td>
<td>10.42</td>
<td>10.81</td>
</tr>
<tr>
<td>$I_5$</td>
<td>8.79</td>
<td>11.05</td>
</tr>
<tr>
<td>$I_6$</td>
<td>9.81</td>
<td>11.21</td>
</tr>
</tbody>
</table>

Figure 19: Project A - actual vs. predicted Test effort values with observations ($I_3$, $I_2$ & $I_1$) entered

The graphs in Figure 19 show the change in the graph for the predicted test effort when we have entered iteration $I_3$ information. The whole predicted graph moves lower and closer to the actual graph, the predictions for future iterations ($I_4$, $I_5$ & $I_6$) improve as compared to prior predictions (Figure 18). Therefore, we can say that the model is capable to learn from the projects data, as new information is entered, probabilities propagates and model predictions improve as a result.
4.2.2 PROJECT B

In this section, the framework is validated by using data from project B. The data of project B was incorporated with the framework to check its learning and prediction abilities. Table 16 shows the actual data of project B that was gathered from the interviewee B.

Table 16: Project B actual data collected from interviewee B

<table>
<thead>
<tr>
<th>Variables</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I₁</td>
</tr>
<tr>
<td>Iteration length</td>
<td>30</td>
</tr>
<tr>
<td>Test team size</td>
<td>2</td>
</tr>
<tr>
<td>Test tool quality</td>
<td>High</td>
</tr>
<tr>
<td>Test team experience</td>
<td>V. high</td>
</tr>
<tr>
<td>Bugs found by TC</td>
<td>0</td>
</tr>
<tr>
<td>Total bugs found</td>
<td>15</td>
</tr>
<tr>
<td>Rework test effort</td>
<td>Enough</td>
</tr>
<tr>
<td>Rework test process quality</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Similar to project A, two scenarios were created in the model for project B that are actual and predicted. Initially, using Table 16, iteration length (I) and test team size (s) values were entered into both scenarios (actual and predicted). However, all other remaining variables from iteration I₁ to I₆ were entered into the actual scenario only. These variables include Test tool quality, Test team experience, Bugs found by TC, Total bugs found, rework test effort and rework test process quality. Further, the model was executed and Test effort (E) probability distribution median values were observed for both actual and predicted scenarios. These actual and predicted Test effort median values are shown in Table 17.

Table 17: Project B - actual vs. predicted test effort values without observations entered

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Actual median</th>
<th>Predicted median</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>17.64</td>
<td>19.17</td>
</tr>
<tr>
<td>I₂</td>
<td>29.75</td>
<td>33.39</td>
</tr>
<tr>
<td>I₃</td>
<td>29.75</td>
<td>34.17</td>
</tr>
<tr>
<td>I₄</td>
<td>14.5</td>
<td>14.5</td>
</tr>
<tr>
<td>I₅</td>
<td>14.64</td>
<td>17.89</td>
</tr>
<tr>
<td>I₆</td>
<td>14.64</td>
<td>17.84</td>
</tr>
</tbody>
</table>

Figure 20 shows the graph of actual and predicted test effort scenarios generated with the help of data given in Table 17. The graph was generated without entering any observation into the model. In Figure 20, we can see that the predicted graph values for all iterations (I₂, I₃, I₅ and I₆) are too high except for iteration I₄ as compared to the actual graph.
To validate the framework with the data from project B, we took only predicted scenario and entered two completed iterations ($I_1$ and $I_2$) data from Table 16 into the model. As we entered the new piece of information into the model, propagation takes place and parameter distributions were updated. These distributions effect the prediction of all future iterations, as shown in Table 18.

### Table 18: Project B - actual vs. predicted test effort values with observations ($I_1$, $I_2$) entered

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Actual median</th>
<th>Predicted median</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>17.64</td>
<td>17.64</td>
</tr>
<tr>
<td>$I_2$</td>
<td>29.75</td>
<td>29.75</td>
</tr>
<tr>
<td>$I_3$</td>
<td>29.75</td>
<td>30.78</td>
</tr>
<tr>
<td>$I_4$</td>
<td>14.5</td>
<td>14.5</td>
</tr>
<tr>
<td>$I_5$</td>
<td>14.64</td>
<td>16.6</td>
</tr>
<tr>
<td>$I_6$</td>
<td>14.64</td>
<td>16.56</td>
</tr>
</tbody>
</table>

Using Table 18, a graph for actual and predicted values was generated as shown in Figure 21. We can see that the whole predicted graph moves lower and closer to the actual values as the model learns from the observations entered into the model. Therefore, the prediction of all future iterations $I_3$, $I_4$, $I_5$ and $I_6$ improves as a result, when iterations $I_1$, $I_2$ were entered into the model. Thus, we can say that the model is able to learn from the projects data, as new information is entered, probabilities propagates and model predictions improve as a result.
The prediction results for project B are much improved and stable as compared to the predictions of project A.

The *Test effort (E)* prediction graphs (Figure 20 and Figure 21) showed improvement in the prediction graph values, but we need to verify the accuracy of the results, therefore we have performed prediction accuracy measures and statistical test to verify these prediction improvement, details are given in next section.
4.3 VALIDATION RESULTS

This section describes the model accuracy results by performing prediction accuracy measures and statistical test. We have used statistical software package SPSS 18.0\(^2\) to perform all statistical calculations.

In order to validate the model results, we have performed the following prediction accuracy measures suggested by [66][67][60].

- Magnitude of Relative Error (MRE)
- Mean of Magnitude of Relative Error (MMRE)

MRE is a normalized measure of discrepancy between actual and predicted values, it provides the basis for MMRE calculations, and it can be defined as

\[
MRE = \frac{|E_{ai} - E_{pi}|}{E_{ai}}
\]

Where \(E_{ai}\) represents actual and \(E_{pi}\) represents predicted effort for \(i^{th}\) iteration.

We are interested in the deviation of relative to the predicted values not to the actual values. As Kitchenham et al. [67] suggested the use of MMRE is relative to the prediction or estimation is MEMRE, where absolute residuals \(|E_{ai} - E_{pi}|\) are divided by estimate \((E_{pi})\), therefore the EMRE is different from MRE and it can be formulated as

\[
EMRE = \frac{|E_{ai} - E_{pi}|}{E_{pi}}
\]

The Mean of EMRE can be defined as

\[
MEMRE = \frac{1}{n} \sum_{i=1}^{n} EMRE_i
\]

With the help of above definitions and measures, we have calculated MRE and EMRE for both projects A and B.

4.3.1 Project A

In this section, we have performed prediction accuracy measures, descriptive statistics, normality test and hypothesis testing for project A.

4.3.1.1 Prediction accuracy measures

In order to check prediction accuracy for Project A (Table 15 and Figure 19). We have calculated MRE and MEMRE measures. The results are shown in Table 19 below.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>$E_{at}$</th>
<th>$E_{pt}$</th>
<th>$\text{MRE}_1$</th>
<th>$\text{EMRE}_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>14.41</td>
<td>14.41</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_2$</td>
<td>12.19</td>
<td>12.19</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_3$</td>
<td>10.42</td>
<td>10.42</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$I_4$</td>
<td>10.42</td>
<td>10.81</td>
<td>0.037</td>
<td>0.036</td>
</tr>
<tr>
<td>$I_5$</td>
<td>8.79</td>
<td>11.05</td>
<td>0.257</td>
<td>0.205</td>
</tr>
<tr>
<td>$I_6$</td>
<td>9.81</td>
<td>11.21</td>
<td>0.143</td>
<td>0.125</td>
</tr>
<tr>
<td>MEMRE</td>
<td></td>
<td></td>
<td>0.073</td>
<td>0.061</td>
</tr>
</tbody>
</table>

It is suggested in the literature that $MEMRE \leq 0.25$ is an indicator of good prediction model [67][66]. From Table 19, we can see that the Mean EMRE (MEMRE) value is 0.061 which is smaller than 0.25. It shows that our prediction results are very good indeed for Project A. Furthermore, we have also performed a statistical test to verify the prediction results. The section below provides the descriptive statistics for project A dataset.

4.3.1.2 Descriptive statistics

Descriptive statistics are used to collect, organize, summarize and present data related to any sample or population that is under study [68]. It deals with measures of data from different aspects [69]. Normally the distribution, central tendency and the dispersion of the data is considered as descriptive statistics. Mean, median, mode, standard deviation, Skewness, kurtosis are few examples of descriptive measures. Table 20 shows the summary of descriptive statistics of the project A actual and predicted dataset. Since the mean and median values are not equal, therefore the actual and predicted data is not normally distributed.

<table>
<thead>
<tr>
<th>Statistic - Project A</th>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>11.0067</td>
<td>11.6817</td>
</tr>
<tr>
<td>Median</td>
<td>10.42</td>
<td>11.1300</td>
</tr>
<tr>
<td>Variance</td>
<td>4.003</td>
<td>2.135</td>
</tr>
<tr>
<td>S.D.</td>
<td>2.00085</td>
<td>1.46118</td>
</tr>
<tr>
<td>Minimum</td>
<td>8.79</td>
<td>10.42</td>
</tr>
<tr>
<td>Maximum</td>
<td>14.41</td>
<td>14.41</td>
</tr>
<tr>
<td>Skewness</td>
<td>1.051</td>
<td>1.674</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>0.850</td>
<td>2.779</td>
</tr>
</tbody>
</table>

4.3.1.3 Normality test

This section provides details on normality test performed on the project A dataset. One way to check data for normality is to use the Shapiro-Wilk normality test, it is powerful test in many situations as suggested in [70][71]. In our case we have used Shapiro-Wilk normality test, histogram and box plot to see either the project A data is normally distributed or not. All
normality tests were performed on project A actual and predicted dataset separately as discussed below.

4.3.1.3.1  Shapiro-Wilk normality test
In the Table 21, we can see the Shapiro-Wilk normality test results for project B.

<table>
<thead>
<tr>
<th>Table 21: Shapiro-Wilk normality test result for project B data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapiro-Wilk tests</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Actual data</td>
</tr>
<tr>
<td>Predicted data</td>
</tr>
</tbody>
</table>

From the Table 26, we can see that the actual data Sig. value is 0.476 which is greater than 0.01 (level of significance). Therefore the actual data is not normalized. Similarly, the predicted data Sig. value (.097) is also greater than 0.01, therefore we can say predicted data is also not normalized.

4.3.1.3.2  Histogram
Histogram also provides a way to test the data for normality [69]. In the Figure 22, project A actual distribution is shown, where the distribution has long right tail. Therefore the distribution is positively skewed and represents a non-normal distribution [68].

Figure 22: Histogram - Normality test for Project A actual values

The Figure 23 shows the histogram for the predicted values of project A, we can see that the distribution has also long right tail. Therefore it also represents a positively skewed and non-normal distribution.

Figure 23: Histograms - Normality test for Project A predicted values

4.3.1.3.3  Box plot
Box plot provides a graphical representation of the dispersion and Skewness. It presents the minimum, lower quartile, median, upper quartile and maximum statistics values visually. A line drawn across the box represents median value, more detail information can be found in [69].
The Figure 24 shows the box plot for project A actual values, we can see that the top whisker is longer than the bottom whisker and line in the box is fall towards the bottom of the box. Therefore we can say that the data is skewed to the right and it is not normalized.

![Figure 24: Box plot - Normality test for Project A actual values](image)

In the Figure 25, a box plot for the predicted values of project A is shown. We can see that line in the box fall towards the bottom of the box. Therefore the data is right skewed and hence non-normalized.

![Figure 25: Box plot - Normality test for Project A predicted values](image)

4.3.1.4 Hypothesis testing

The main objective of hypothesis testing is to see if it is possible to reject a null hypothesis with high significance, based on the data collected from industry [69]. There are a number of tests available to perform hypothesis testing such as \( t \)-test, Mann-Whitney, \( F \)-test, Paired \( t \)-test, Wilcoxon, Sign test, ANOVA (Analysis Of Variance), etc. In our case, we have two factors (actual and predicted) and one treatment (test effort).

We have seen that the project A data is not normally distributed. Therefore we have selected a non-parametric test that is Wilcoxon signed ranks test to perform a statistical test, as suggested by [69][72]. Wilcoxon signed ranks test requires a null and alternate hypothesis, and its acceptance and rejection criteria.

In our case, we have defined null hypothesis is the median of differences between actual and predicted not equals to 0 and alternate hypothesis is the median of differences between actual and predicted equals to 0, with significance level of 0.01. The null and alternate hypothesis can also be formulated as

\[
H_0: \text{Median}_{\text{ai}} - \text{Median}_{\text{pi}} \neq 0 \\
H_1: \text{Median}_{\text{ai}} - \text{Median}_{\text{pi}} = 0
\]

We have established the level of significance to be equal to 0.01, because if we reject the null hypothesis then there is 99 percent certainty that it is a correct decision.
After applying Wilcoxon signed ranks test we got few positive and negative ranks. Figure 26 shows that we have three positive ranks, zero negative ranks and three ties values after subtracting the actual and predicted values of the dataset.

![Related-Samples Wilcoxon Signed Ranks Test](image)

**Figure 26: Statistical test results for Project A - graph**

From Table 22, we can see that we got three negative rank values when we subtracted actual and predicted values. Two and six is the mean and sum of negative rank values, where there is no positive rank value. Three ties between actual and predicted values exist, these ties values are neglected therefore we have three values left.

<table>
<thead>
<tr>
<th>Actual - Predicted</th>
<th>N</th>
<th>Mean rank</th>
<th>Sum of ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative ranks</td>
<td>3^a</td>
<td>2.00</td>
<td>6.00</td>
</tr>
<tr>
<td>Positive ranks</td>
<td>0^b</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>Ties</td>
<td>3^c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Actual < Predicted  
b. Actual > Predicted  
c. Actual = Predicted

In Table 23, we can see the p-value that is Asymp. Sig. 2-tailed (asymptotic significance, 2-tailed) equals to 0.109. However, we have used 0.01 as level of significance. Since the p-value (0.109) is greater than 0.01, we reject null hypothesis.

<table>
<thead>
<tr>
<th>Actual - Predicted</th>
<th>Z</th>
<th>Asymp. Sig. (2-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.604^a</td>
<td>.109</td>
</tr>
</tbody>
</table>

a. Based on positive ranks.

Wilcoxon signed ranks test results confirmed that there is no significant difference between the actual and predicted Test effort ($E$) values with confidence interval of 99.9% at 0.01, therefore we reject the null hypothesis $H_0$ and accept the alternative hypothesis.
4.3.2 PROJECT B

In this section, we have calculated prediction accuracy measures, descriptive statistics, normality test and hypothesis testing for project B dataset.

4.3.2.1 Prediction accuracy measures

We have also performed MRE and MEMRE calculations for project B prediction results (Table 18 and Figure 21), MRE and MEMRE accuracy results are shown in Table 24.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>(E_{at})</th>
<th>(E_{pt})</th>
<th>(MRE_{I})</th>
<th>(EMRE_{I})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I_1)</td>
<td>17.64</td>
<td>17.64</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>(I_2)</td>
<td>29.75</td>
<td>29.75</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>(I_3)</td>
<td>29.75</td>
<td>30.78</td>
<td>0.0346</td>
<td>0.0335</td>
</tr>
<tr>
<td>(I_4)</td>
<td>14.5</td>
<td>14.5</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>(I_5)</td>
<td>14.64</td>
<td>16.6</td>
<td>0.1339</td>
<td>0.1181</td>
</tr>
<tr>
<td>(I_6)</td>
<td>14.64</td>
<td>16.56</td>
<td>0.1311</td>
<td>0.1159</td>
</tr>
<tr>
<td>MEMRE</td>
<td></td>
<td></td>
<td><strong>0.2996</strong></td>
<td><strong>0.0446</strong></td>
</tr>
</tbody>
</table>

Table 24 shows that Mean EMRE (MEMRE) is equal to 0.0446 that is much less than 0.25. Therefore, we can say our predicted Test effort (\(E\)) results for project B are accurate and good enough. Furthermore, we have also performed a statistical test to verify predicted Test effort (\(E\)) results. The descriptive statistics of the dataset of project B (Table 18) are given in Table 25.

4.3.2.2 Descriptive statistics

The descriptive statistics for the dataset of project B is shown in Table 25, we can see that the mean, median values are not equal in both cases (actual and predicted data). However, the skewness is positive in both cases. Therefore we can say the actual and predicted data is non-normalized.

<table>
<thead>
<tr>
<th>Statistic - Project B</th>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>20.1533</td>
<td>20.9717</td>
</tr>
<tr>
<td>Median</td>
<td>16.1400</td>
<td>17.1200</td>
</tr>
<tr>
<td>Variance</td>
<td>56.653</td>
<td>52.964</td>
</tr>
<tr>
<td>S.D.</td>
<td>7.52679</td>
<td>7.27763</td>
</tr>
<tr>
<td>Minimum</td>
<td>14.5</td>
<td>14.50</td>
</tr>
<tr>
<td>Maximum</td>
<td>29.75</td>
<td>30.78</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.870</td>
<td>0.889</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>-1.916</td>
<td>-1.805</td>
</tr>
</tbody>
</table>

4.3.2.3 Normality test

In this section, we have provided details on the different normality tests that are performed on the dataset of project B to see if the data is normalized or not. Shapiro-Wilk normality test, histogram and box plot normality techniques were used for both project B actual and predicted datasets as discussed below.
4.3.2.3.1  Shapiro-Wilk normality test

Shapiro-Wilk normality test provides the following results as shown in Table 26.

<table>
<thead>
<tr>
<th></th>
<th>Shapiro-Wilk tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual data</td>
<td>Statistic</td>
</tr>
<tr>
<td></td>
<td>.723</td>
</tr>
<tr>
<td>Predicted data</td>
<td>.768</td>
</tr>
</tbody>
</table>

From Table 26, we can see that the actual data Sig. value is 0.011 which is equal to the level of significance 0.01; therefore we can say the actual data is normalized at 0.01. While the predicted data Sig. value is 0.30 which is greater than 0.01, therefore we can say predicted data is not normalized.

4.3.2.3.2  Histogram

Figure 27 shows the histogram for project B actual values, where the distribution has a bit longer right tail. Therefore we can say project B actual data is not normalized.

The histogram shown in Figure 28, represents the project B predicted data distribution, it is clear from the graph that the distribution has a longer right tail. Therefore, we can say it is positively skewed and a non-normal distribution.

4.3.2.3.3  Box plot

Figure 29 shows the box plot for project B actual values, we can see that the top whisker is longer than the bottom whisker and line in the box is fall towards the bottom of the box. Therefore we can say that the data is skewed to the right and it is not normalized.
Similarly, Figure 30 shows a box plot for project A predicted values, since the line in the box fall towards the bottom of the box. Therefore we can say project B predicted data is right skewed and hence non-normalized.

4.3.2.4 Hypothesis testing

To perform a statistical test, first of all, we have checked project B dataset (Table 18) for normality check as we did before for project A dataset. Boxplots, histograms, Shapiro-Wilk normality test results confirmed that the project B dataset is not normally distributed.

Since project B data is not normalized, we have selected Wilcoxon signed ranks test non-parametric test, similarly as we did for project A. We have defined null and alternate hypothesis for Wilcoxon signed ranks test. Null hypothesis is the median of differences between actual and predicted not equals to 0 and alternate hypothesis is the median of differences between actual and predicted equals to 0, with significance level of 0.01. The null and alternate hypothesis can also be written as

\[ H_0: \text{Median}_{al} - \text{Median}_{pl} \neq 0 \]
\[ H_1: \text{Median}_{al} - \text{Median}_{pl} = 0 \]

For project B, we have also established the level of significance to be equal to 0.01.

The Wilcoxon signed ranks test shows that we have few positive and negatives ranks, in Figure 31 we can see that there are three positive, zero negative and three ties exist for actual and predicted values of project B.
When the actual and predicted values are subtracted we got three negative ranks with 2 and 6 as mean rank and sum of ranks respectively. Similarly, we got zero positive ranks and three ties, these ties values are neglected for any calculation. Table 27 provides the details of Wilcoxon signed ranks for project.

Table 27: Wilcoxon signed ranks

<table>
<thead>
<tr>
<th>Actual - Predicted</th>
<th>N</th>
<th>Mean rank</th>
<th>Sum of ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative ranks</td>
<td>3a</td>
<td>2.00</td>
<td>6.00</td>
</tr>
<tr>
<td>Positive ranks</td>
<td>0b</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>Ties</td>
<td>3c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Actual < Predicted  
b. Actual > Predicted  
c. Actual = Predicted

Table 28 shows the Wilcoxon signed ranks test statistics for project B, where calculated Z and Asymp. Sig. 2-tailed values are -1.604 and .109 respectively. The asymptotic significance 2-tailed value is also known as p-value. We have used 0.01 as level of significance, since the p-value (0.109) is greater than 0.01, therefore we can reject null hypothesis.

Table 28: Test Statistics - Wilcoxon signed ranks test results for project B

<table>
<thead>
<tr>
<th>Actual - Predicted</th>
<th>Z</th>
<th>Asymp. Sig. (2-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.604a</td>
<td>.109</td>
</tr>
</tbody>
</table>

a. Based on positive ranks.

Further Wilcoxon signed ranks test results confirmed that there is no significant difference between the actual and predicted Test effort (E) values with confidence interval of 99.9% at 0.01, therefore we can reject the null hypothesis $H_0$ and accept the alternative hypothesis.

Therefore, we can say that the model prediction results are quite accurate and good enough as it is proved by prediction accuracy measures. Further, the statistical test also confirmed that there is no significant difference between the medians of actual and predicted graph values.
4.4 RESULTS ANALYSIS AND DISCUSSION

In this section, we have discussed and analyzed the prediction results for project A and B.

Framework prediction results can also be analyzed by taking the difference between the actual and predicted values. If there is no difference between the predicted and actual values then predictions will be accurate. Therefore, we can say that the accuracy can be measured by how close a predicted value to the actual value is.

The project A and B prediction results (Table 14 & Table 18) can be analyzed by taking the difference between the actual and predicted values as shown in Table 29. In both projects, two observations were entered into model and results were observed. Table 29 shows that the project B prediction results are more accurate and close to actual values as compare to project A.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>-2.05</td>
<td>-1.03</td>
</tr>
<tr>
<td>4</td>
<td>-2.19</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>-3.88</td>
<td>-1.96</td>
</tr>
<tr>
<td>6</td>
<td>-2.89</td>
<td>-1.92</td>
</tr>
</tbody>
</table>

Furthermore, it was observed that project A predictions were not good enough, therefore third observation (I₃) was entered into the model. The result show that the fourth, fifth and sixth iteration improved by 82.2, 42.0 and 51.56 percent respectively (Table 30). New predicted values are more accurate and closer to actual values. Therefore we can say that the model can learn from the data.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Project A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New predictions</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-0.39</td>
</tr>
<tr>
<td>5</td>
<td>-2.26</td>
</tr>
<tr>
<td>6</td>
<td>-1.4</td>
</tr>
</tbody>
</table>

In the previous section, we have already evaluated the prediction results by performing the MRE and MEMRE measures. It is found that the MEMRE values less than 0.25 is an indicator of good prediction model. In our case, the result shows MEMRE equal to 0.061 and 0.0446 for project A and B respectively, which is quite smaller than 0.25. Therefore we can say our prediction model is quite accurate.

Furthermore, the statistical test results for project A and B also confirmed that there is no significant difference between the actual and predicted values at 99 percent level of significance.
Moreover, the prediction results analysis and discussion also provides evidence that the proposed model has learning and predicting ability from projects data. As new piece of information is entered into the model, it learns and makes the prediction more accurate. Therefore, we can conclude that the proposed framework is statistically valid and accurate.

There were a number of threats involved in our study; we have discussed those threats in the next section.
5  VALIDITY THREATS

There are mainly four types of threats such as conclusion validity (Reliability), Construct validity, internal validity (Causality), External validity (Generalization) [69] [73][74]. In our study, a number of threats were involved as discussed below:

5.1  RELIABILITY OF MEASURE
In our study, there were a number of threats related to reliability of measures.

First, there was a threat that the framework design was not reliable. To minimize such threat, the framework design was also reviewed by an external researcher having rich experience in designing and implementing Bayesian networks, he has many publications in Bayesian network research. The framework design was improved based on his suggestions.

Second, an inappropriate industrial questionnaire design was also a threat to validate the model. To minimize this, questionnaire design was reviewed and validated by one researcher to remove the ambiguities and unnecessary questions.

Third, there was a threat that the framework behavior is not reliable. Therefore to minimize it, framework behavior was evaluated by observing the model parameters learning and performing an initial validation.

5.2  HETEROGENEITY OF SUBJECTS
Homogeneous subjects could affect the external validity, since the subjects did not belong to general population.

In our study, the subjects are heterogeneous since the framework was validated through two different companies. Further, Motorola project data was also used to validate the model. Therefore, the proposed framework can be generalized to any company that uses iterative development methodology.

5.3  EVALUATED APPREHENSION
In this study, the evaluated apprehension was a main threat to construct validity. Evaluated apprehension means that humans have the tendency to perform better when they are evaluated [69]. Sometimes people get afraid to do this. In order to minimize this threat, we ensured the interviewees that their company and personal information should be kept confidential. Therefore, we have not used their names in the report.

5.4  MONO-OPERATION BIAS
In this study, mono-operation bias was a threat that the study may include a single independent variable, case, subject or treatment. It does not provide the full picture of the theory. To minimize such a threat framework was validated with different kind of datasets collected from industry.

5.5  MONO-METHOD BIAS
In our study, there was a risk that prediction results could be inaccurate. Therefore, to reduce such risk, prediction accuracy measures and a statistical test was performed.
5.6 **HYPOTHESIS GUESSING**
There was a threat that the interviewee might try to guess the purpose of the study and react accordingly without considering other factors. To minimize such threat, the purpose of the study was explained to the interviewees. Further, the framework design and questionnaire were discussed with the interviewees during the interviews.

5.7 **SELECTION OF SUBJECTS**
The wrong selection of subjects can also affect the results of an experiment. In our case, the interviewees were selected based on their testing experience.

5.8 **SELECTION OF TOOLS**
There was a risk to select inappropriate tools for Bayesian network modeling and implementation. However, we had compiled a list of existing Bayesian network designing software. Further, a detailed analysis and comparison was performed based on features, pros and cons. We found *AgenaRisk* toolset one of the most appropriate for our study.

5.9 **FRAMEWORK COMPARISON**
Due to unavailability of appropriate model that can be used for test effort prediction in iterative development, the proposed framework cannot be compared to any model. To minimize such risk, we have validated the model through prediction accuracy measures and statistical tests.

5.10 **LACK OF SYSTEMATIC REVIEW**
There was a risk that we missed any important key paper because we have not performed a systematic review. To minimize such a risk, we have performed an effective literature review recommended by Levy and Ellis [57]. It helped to search key articles (based on BN) from different online databases. These articles were recently published and up-to-dated, we have compiled the list of their references. It helped to minimize the risk of missing any key paper.

Furthermore, most of models or frameworks are proposed based on literature review; however they did not performed a systematic literature review. For example, the studies [13][11][10] are based on literature review rather than systematic review. Moreover, it is the researcher who defines about the review (literature or systematic) depending upon the research area, aims and objectives.

5.11 **EXPERT JUDGMENT**
There was a risk that an expert decision can be biased. In industry knowledge is distributed among number of experts [75]. Therefore, it is recommended to make coordination with other experts before taking any decision. For example, Test manager and Test team lead can have some discussion before taking any decision.

5.12 **STATISTICAL VALIDATION**
There was a risk that the selection of prediction measurements and statistical tests can be wrong or it can produce false results. We have minimized this threat by selecting proper prediction measurements and statistical tests that are suggested by several researchers.

In order to minimize the risk that the statistical test can produce false results. First, we have applied several normality tests (Shapiro-Wilk, histogram, and box plot) on the original dataset without applying any data transformation techniques such as natural log, reciprocal, square root. Second, a non-parametric test (Wilcoxon signed ranks test) was selected as it was right choice depending on the distribution of nature of original dataset.
6 EPILOGUE

This section presents the conclusions and possible future work.

6.1 CONCLUSIONS

It is important to manage iterative projects in a way to maximize quality and minimize cost. To achieve high quality, accurate project estimates are of high importance. However, it is challenging to predict the amount of effort required to perform test activities in an iterative development. Therefore, to overcome such a challenge we present here one alternative model.

In this research, the main contribution was to introduce and validate a dynamic Bayesian network for predicting test effort in iterative software development. Predicting test effort enables the test manager to plan and control the test activities that involve test team, test tools, etc. Thus, delayed test activities and schedule overrun could be avoided.

In our research work, the framework predictions were evaluated in a number of ways.

First, the framework was evaluated by considering parameter learning and initial validation. The framework parameters learning were observed under different scenarios such as success, average and failing. The results show that the framework parameters learn from prior iterations data and make prediction for next iterations. Further, initial validation was performed by incorporating Motorola project data with the model. In that case, we had two scenarios in the model such as actual and predicted. When two iterations data entered into the model, the prediction results did not improve much. However, when third iteration data was incorporated into the model, the prediction results improved a lot. The results show that the framework can learn from previous iteration data and improve its prediction for future iterations.

Second, the framework was validated by incorporating data from two industrial projects. To evaluate the model predictions, two scenarios were constructed such as actual and predicted. In the case of project A, initially data from two iterations were entered into the model and test effort prediction results were observed. The result indicates worst predictions for project A. However, when additional iteration data was incorporated with the model, it learned quickly and the prediction results were more accurate and improved a lot. In the case of project B, two iterations data were incorporated with the model and results were observed. The results show that the model performed well, predictions were more accurate as compared to actual for all future iterations.

Thus, we can conclude that the proposed model can learn from projects data or information, as new information entered into the model, prediction are more improve and accurate.

Further, the results of the industrial framework validation were validated through prediction accuracy measures (MRE and MEMRE) and statistical tests. The prediction accuracy measures indicate that the prediction results for both projects (A and B) are very good indeed. Moreover, a statistical test was performed using Wilcoxon signed rank test. The results confirmed that there was no significant difference between the medians of actual and predicted values for the test effort.

Thus, we recommend following the proposed framework to predict test effort in iterative projects. It is different from other models; it has a very simple structure with small number
of nodes. Therefore, we believe that the proposed framework is simple, flexible and easy to adopt.

6.2 FUTURE WORK

This section describes the possible future work.

Specific agile practices can be introduced into the proposed model such as SCRUM, XP, TDD, and etc. Framework behavior and prediction accuracy can also be further evaluated by implementing the model into a real industry environment. In this way, managers can incorporate data from real projects into the model and evaluate prediction results.

Further, the results of the model can also be validated through more data from different kind of industrial projects to check the model prediction accuracy. Moreover, similar kind of models for predicting effort can also be constructed for each phase of software development life cycle such as requirements, development and so on.
7 REFERENCES


[45] D.D. Galorath and M.W. Evans, Software sizing, estimation, and risk management:


APPENDIX A: FRAMEWORK INTERACTION WITH MULTIPLE ITERATIONS

The following diagram show how the framework is repeats and interacts with multiple iterations. A copy of the model is replicated in all iterations, where each prior model provides an input to the next iteration model and vice versa.

The green color indicates the nodes that provide input to the next iteration nodes (indicated by blue color). For example, in iteration #1 the “Rework test process quality POST” is an input to the “Rework test process quality PRE” node to the second iteration and so on. This indicates a dynamic Bayesian network where each prior node is linked with the next iteration.

Figure 32: A Framework model in multiple iterations – (part I), next page
Figure 33: A Framework model in multiple iterations – (part II)
APPENDIX B: A DETAILED FRAMEWORK DIAGRAM WITH PROBABILITY DISTRIBUTION

The following diagram shows the framework execution view, where the probability distributions for each node is shown.

There are different kinds of probability distribution are shown in the model. For example, Test process quality node has five scale point values such as {very low, low, medium, high, very high}. For each value, there is a different probability value such as very low is 3.605%, low is 41.917%, medium is 47.769% and so on.

Figure 34: A detail proposed Framework design with probability distribution
APPENDIX C: INDUSTRIAL QUESTIONNAIRES

Two questionnaires were used to collect projects data from industry. Figure 35 shows a questionnaire used to collect project A data. Whereas in Figure 36, an improved version of project A questionnaire design is shown where few startup questions are included, it was used to collect project B data. There is no major difference between the both questionnaires.

Figure 35: Industrial questionnaire used for project A data collection

Figure 36: Industrial questionnaire used for project B data collection