

## **A Story Point based approach for effort estimation with regression testing effort for Agile projects**

Final Dissertation submitted

By

**Shivali Chopra**  
(11302040)

To

**Department of Computer Science Engineering**

In partial fulfillment of the Requirement for the Award of the Degree of

**Master of Technology in CSE**

**Under the guidance of**

**Ms. Rupinder Kaur**  
**Assistant Professor**

**(May, 2015)**

PAC Form



School of: Computer Sciences & Engg.

DISSERTATION TOPIC APPROVAL PERFORMANCE

Name of the Student: Shivali Chopra Registration No: 11302040  
Batch: 2013-15 Roll No. ....  
Session: 2014-15 Parent Section: K2206  
Details of Supervisor: Designation: Asst. Prof.  
Name: Manpreet Kaur Qualification: M.Tech  
U.ID: 15387 Research Experience: 4 Years

SPECIALIZATION AREA: SW Engg. (pick from list of provided specialization areas by DAA)

PROPOSED TOPICS

- Effort (size & cost) estimation in Agile using various traditional effort estimation techniques and comparison.
- Software fault localization to reduce the efforts for debugging
- Software effort estimation using Neural Network's learning algorithms.

Signature of Supervisor: M. Kaur  
15387

PAC Remarks:

Topic ① is approved. Publication expected.

11265

Signature: [Signature]  
Date: 26/9/14

APPROVAL OF PAC CHAIRPERSON:

Date:

- \* Supervisor should finally encircle one topic out of three proposed topics and put up for approval before Project Approval Committee (PAC)
- \* Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.
- \* One copy to be submitted to Supervisor.

## ABSTRACT

To be competitive in today's fast moving market place, organizations need to derive innovations in every part of their business. In order to accomplish the same, more and more companies are embracing Agile development as a viable development methodology that deliver customer value faster with IT and cross key business units. Effort estimation is one of the key feature for the success of the projects. As effort estimation is the biggest challenge in Agile based projects because of the volatile nature of the requirements, so there is a need for efficient techniques for better results. This work proposes a technique to estimate effort in Agile software development based projects.

Story point based approach is found to be a suitable metrics for Agile projects in spite of being relative in nature. This work has incorporated this approach to form a new technique. As in Agile projects the requirements keep on changing as per the need and the demand of the client which makes regression testing as an integral part of Agile methodology. But in existing techniques of effort estimation, it is not taken into account. This work proposes a new technique (E<sub>3</sub>RT) for effort estimation by including the effort of regression testing in Agile software development.

In addition, a comparison of the existing techniques with the new one is made which proves that the new technique gives more realistic and accurate results.

## **ACKNOWLEDGEMENT**

I would like to express my deepest sense of gratitude to my supervisor Ms. Rupinder Kaur for her valuable suggestions, extensive guidance, continuous encouragement and support throughout my dissertation work. I have benefitted a lot from her feedback. Moreover, I extend my heartiest thanks to all other faculty members of LPU.

Last but not the least I would like to thank my husband and family for their continuous encouragement and support.

**(SHIVALI CHOPRA)**

## DECLARATION

I hereby declare that the progress report on dissertation entitled, “**A Story Point based approach for effort estimation with regression testing effort for Agile projects**” submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date:

SHIVALI CHOPRA

Reg. No: 11302040

## CERTIFICATE

This is to certify that **Shivali Chopra** is pursuing M.Tech (CSE) dissertation titled, “**A Story Point based approach for effort estimation with regression testing effort for Agile projects**” under my guidance and supervision. To the best of my knowledge, the present work is the result of her original investigation and study. No part of the dissertation has ever submitted for any other degree or diploma. The dissertation is fit for the submission and the partial fulfillment of the conditions for the award of M.Tech Computer Science and Engineering.

Date:

Ms. Rupinder Kaur (Asst. Professor)

U.ID: 16835

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF FIGURES	vi
	LIST OF TABLES	viii
1.	INTRODUCTION	
1.1	Agile- Expects the Unexpected	2
1.1.1	Agile Methods	3
1.1.2	Scrum	4
1.2	Effort Estimation in software development	6
1.2.1	Project estimation techniques	7
1.2.2	Agile Effort Estimation	10
1.3	Regression Testing	13
2.	REVIEW OF LITERATURE	14
3.	SCOPE OF THE STUDY	19
4.	OBJECTIVE OF THE STUDY	20
5.	RESEARCH METHODOLOGY	21
5.1	Tools used	21
5.2	Activity diagram for the E <sub>3</sub> RT technique	22
5.3	Proposed work	23
6.	RESULTS and DISCUSSIONS	30
7.	SUMMARY and CONCLUSIONS	44
	REFERENCES	45

## LIST OF FIGURES

Figure No.	Figure Title	Page No.
1.1	Waterfall Model	1
1.2	Essence of Agile	2
1.3	Agile Manifesto	2
1.4	Agile Principles	3
1.5	Agile Umbrella	4
1.6	Scrum Process	4
1.7	Framework for Software estimation	6
1.8	Estimation techniques	7
1.9	Size estimating methods	9
5.1	Activity Diagram for E <sub>3</sub> RT technique	22
5.2	User Story Size Scale	24
5.3	User Complexity Scale	25
6.1	User Story Size Scale	30
6.2	User Complexity Scale	31
6.3	Calculation of the initial effort	32
6.4	Calculation of Initial Velocity	33
6.5	Decelerating factors of velocity	34
6.6	Ideal Case for the any Agile project	36
6.7	Calculation Based on Case1	37
6.8	Calculation Based on Case2	38
6.9	Calculation Based on Case3	39
6.10	Comparison of results for four projects with E <sub>3</sub> RT case 1	40
6.11	Comparison of results for five projects with E <sub>3</sub> RT case 2	41
6.12	Comparison of results for five projects with E <sub>3</sub> RT case 3	41
6.13	Comparison of results	42



## LIST OF TABLES

Table No.	Table Name	Page No.
1.1	Qualities of good user story	11
1.2	Agile estimation approaches and problems	12
5.1	Decelerating Factors	26
5.2	Change type and its related degree	27
5.3	TEC and its related degree	27
5.4	NSTC and its related degree	28
5.5	TTP and its related degree	28
6.1	Initial Effort and velocity	35
6.2	Comparison of results of four projects out of 21 projects	40
6.3	Calculation of error and MRE value with respect to Case1	42
6.4	Percentage comparison of previous technique and E <sub>3</sub> RT	43

# Chapter 1

## INTRODUCTION

---

In the world of industrial process control, we see two major approaches to control any process. These are the “defined” process control model and the “empirical” process control model. The defined model is what you use, when you attempt to thoroughly plan a software project. Here, every piece of work must be completely understood. In the defined process model we are given with a set of well defined inputs, the same outputs are generated every time. This states that if the requirements are given clear and stable and the technology is well understood then we can predict a software project.

Waterfall model is a defined process control model which is the first software development model. It is also known as linear sequential model which has defined start and end points and has identifiable deliveries to next phase.



Figure 1.1: Waterfall Model

The ‘V’ model is refinement of waterfall model which integrates verification and validation activities. In each phase tests are planned and addressed. But there are drawbacks of this model which includes late integration, expensive requirement changes.

The empirical process control is followed when one wants to control situations where inputs are varying, where the process is too complex to produce repeatable output, and where one need to frequently inspect and adapt to control such processes. An example of the same is agile approach. Agile overcomes the drawbacks of traditional software development methodologies.

## 1.1 Agile – Expects the Unexpected

Agile is the latest buzz word in the software development industry. It is iterative, incremental, and evolutionary. The development of the software is done in short time boxed iterations – 1 to 4 weeks.

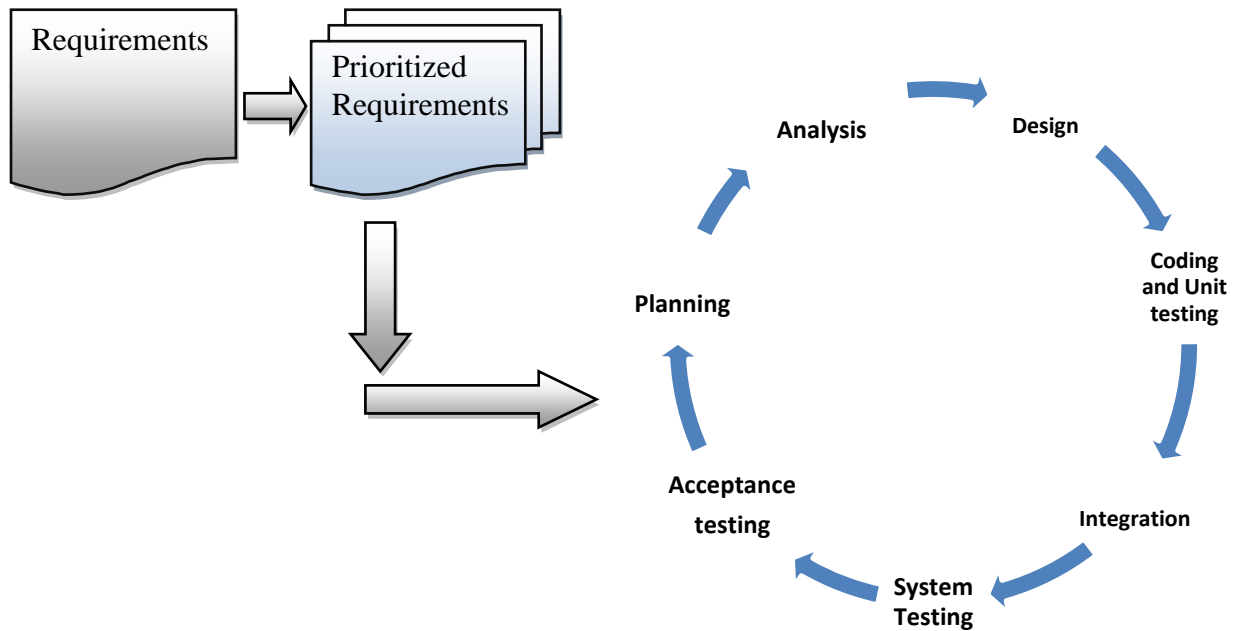


Figure 1.2: Essence of Agile

From the requirements, a prioritized set of requirement are chosen for which development is done in short time boxed iterations. At the end of every iteration one can have production ready version of the software. Thus the system grows iteratively and incrementally. So, Agile is a highly collaborative approach with emphasis on open communication. Agile Manifesto was developed by the Agile Alliance which includes four values as stated below:

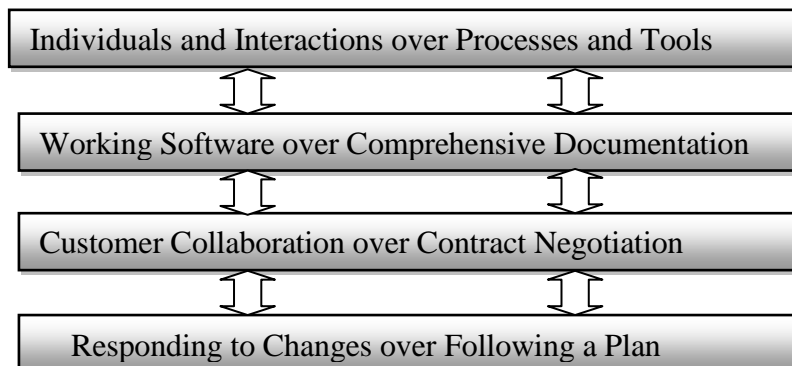


Figure 1.3: Agile Manifesto

There are 12 agile principles as shown below:

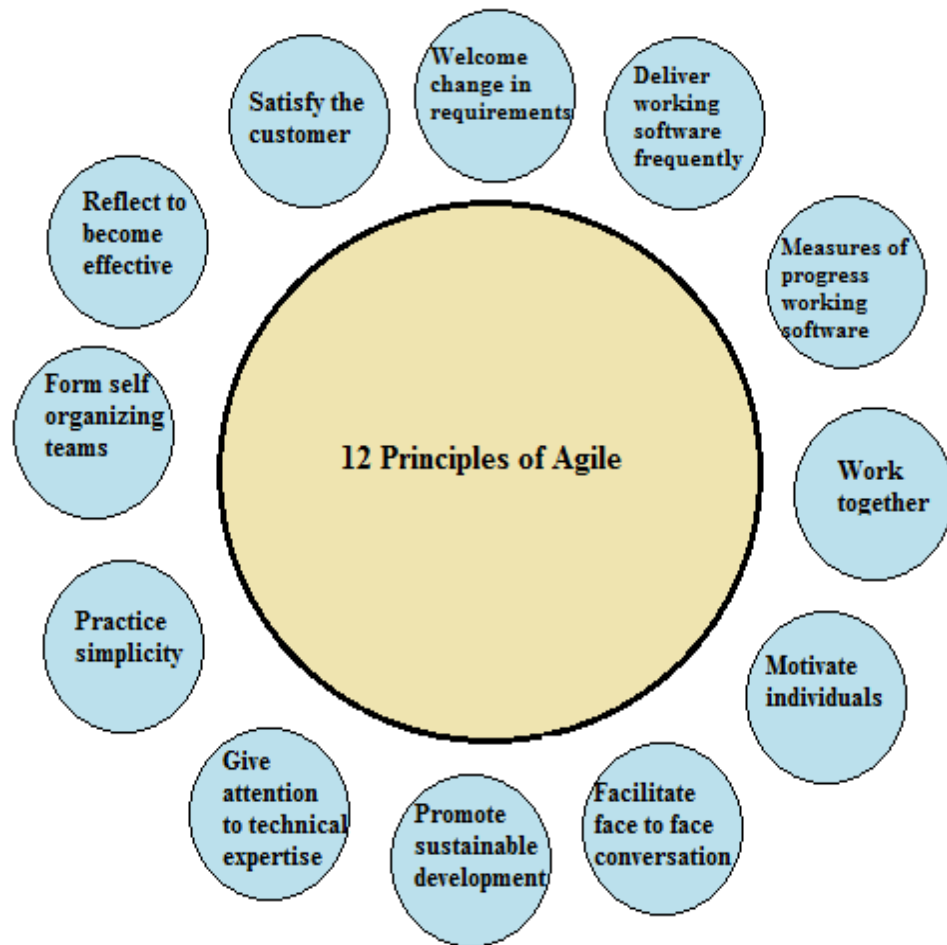


Figure 1.4: Agile Principles

The basic advantages of using Agile are – its efficiency for handling Requirement Volatility and higher Return on Investment.

### 1.1.1 Agile Methods

There are different methods that are the part of Agile Methodology. These are Scrum, eXtreme Programming, Crystal Clear, Adaptive Software Development, Dynamic Systems Development method, Feature Driven Development, Agile Unified Process etc. Out of all, Scrum methodology is widely used by most of software development industries now-a-days. Now, let's have a look of the agile umbrella in the Figure 1.5 which contains various agile development methodologies.

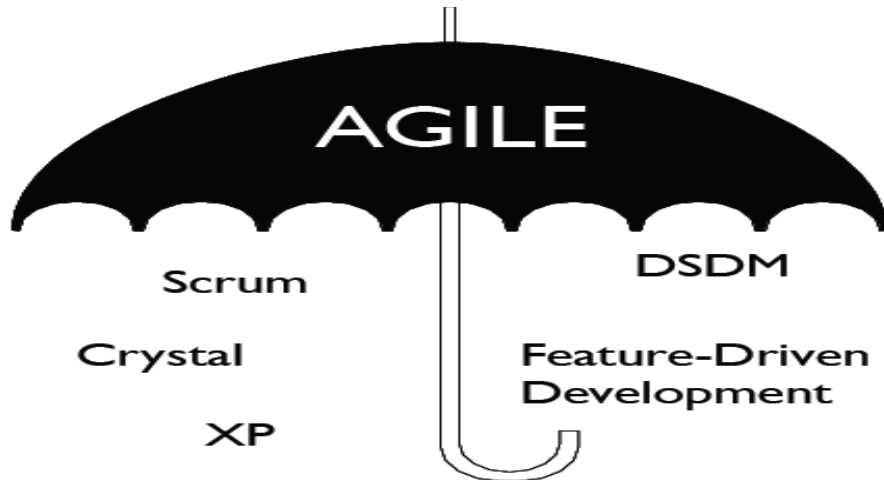


Figure 1.5: Agile Umbrella

### 1.1.2 Scrum

Scrum is an Agile software development framework in which work is structured in short cycles called sprints that are typically two to four weeks in duration. A potentially shippable product is delivered at the end of each sprint. The entire cycle from Sprint Planning to Retrospectives is called a Sprint.

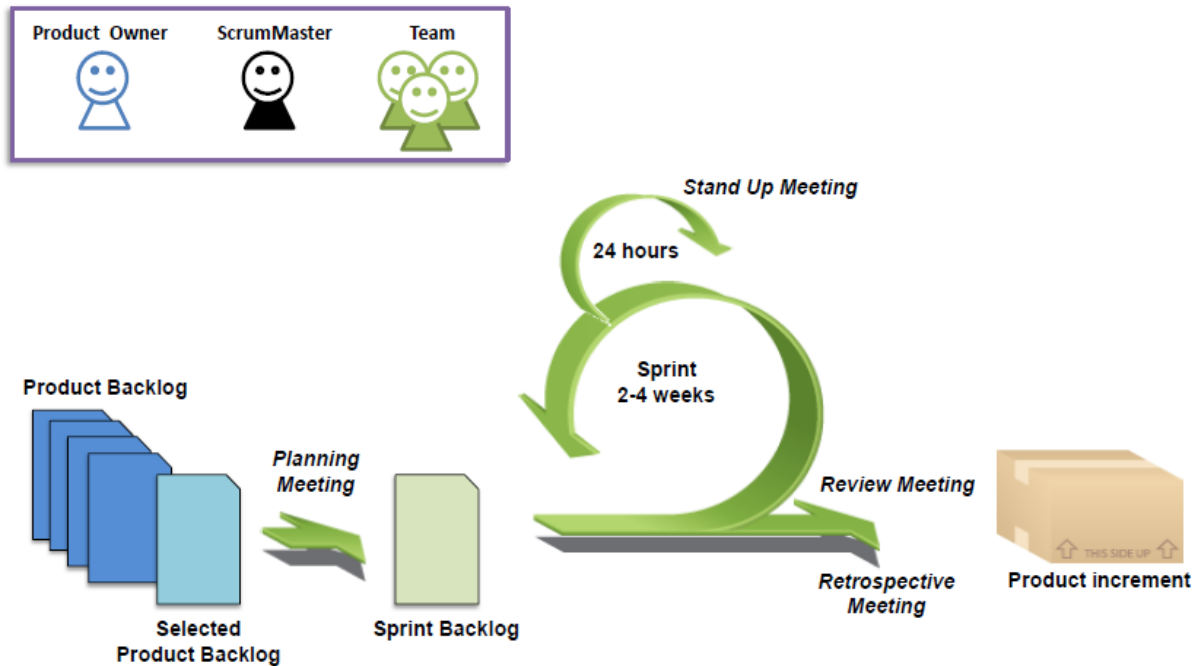


Figure 1.6: Scrum Process [1]

**a) Product Backlog:** Customer prioritizes requirements as per business value and lists them in the product backlog. Requirements are listed in the form of user stories, explained later in detail. While the customer gives the priority, the project team also called the scrum team provides a high-level estimate for each user story.

**b) Sprint Planning:** At the beginning of each time-box called sprint, team conducts a sprint planning meeting. All the stakeholders participate in the meeting. In the meeting, team picks up stories to be implemented in the sprint from the prioritized product backlog. The number of stories picked up depends on the available capacity in person-hours and the team's productivity. It is very important that the customers prioritize the product backlog. Prioritization ensures that the features developed first are of the highest value. The sprint planning meeting normally takes about half a day.

**c) Sprint Backlog:** The Scrum team also breaks down product backlog's requirements into sprint tasks. These are the specific development activities needed to implement the requirement. The output of the sprint planning meeting is the sprint backlog. The sprint backlog contains the tasks and task-level estimates of the selected stories. When the Sprint Backlog is complete, you compare the estimated total work with original high-level estimates from the Product Backlog.

**d) Implementation Cycle:** Once the team is ready with the sprint backlog, implementation of stories commences. The Implementation cycle involves the activities of design, coding and testing. The progress of the team is monitored through visual controls like Story Boards and Effort Burn-down charts.

**e) Daily Scrum:** Every day the daily scrum meeting is conducted at a pre-determined time – typically done at the beginning of the day. This is a fifteen-minute meeting designed to clarify the state of the Scrum without deviating to technical issues. It is mandatory for the team to “stand-up” during these meetings so that the stipulated time is not exceeded. Each team member shares the status of their work by answering the three questions: what did I do yesterday, what will I do today, and what impediments are getting in my way? While anyone can attend this meeting, only team members who have are working on a story are allowed to speak. The goal is to get a global snapshot of the project, discover any new dependencies, and address any personal needs of committed individuals, and finally adjust the work plan in real time. At the end of the daily scrum meeting the sprint backlog is updated – with addition,

deletion and modification to the planned tasks and the remaining efforts for the same.

**f) Sprint Review:** The output of the sprint is a potentially shippable product, which is demonstrated to all the stakeholders and their feedback is sought – this is called the sprint review meeting. All enhancements, bugs or defects identified by the customer are added to the product backlog and are addressed based on their priority.

**g) Retrospective:** A retrospective is conducted post the Sprint Review. The team assesses what went well, what did not and identifies the changes needed to make the process better. Retrospectives allow team to inspect and adapt. Metrics collected are also analyzed to identify improvement areas.

## 1.2 Effort estimation in software development:

For the successful and effective execution of the project so that the budget may not overrun or delivered late, Software development effort estimation plays an indispensable role in SDLC. It is the process of predicting the amount of effort required to develop the software at the beginning of the project. It is very much essential for Software Project Management. The estimating framework for the same is given below:

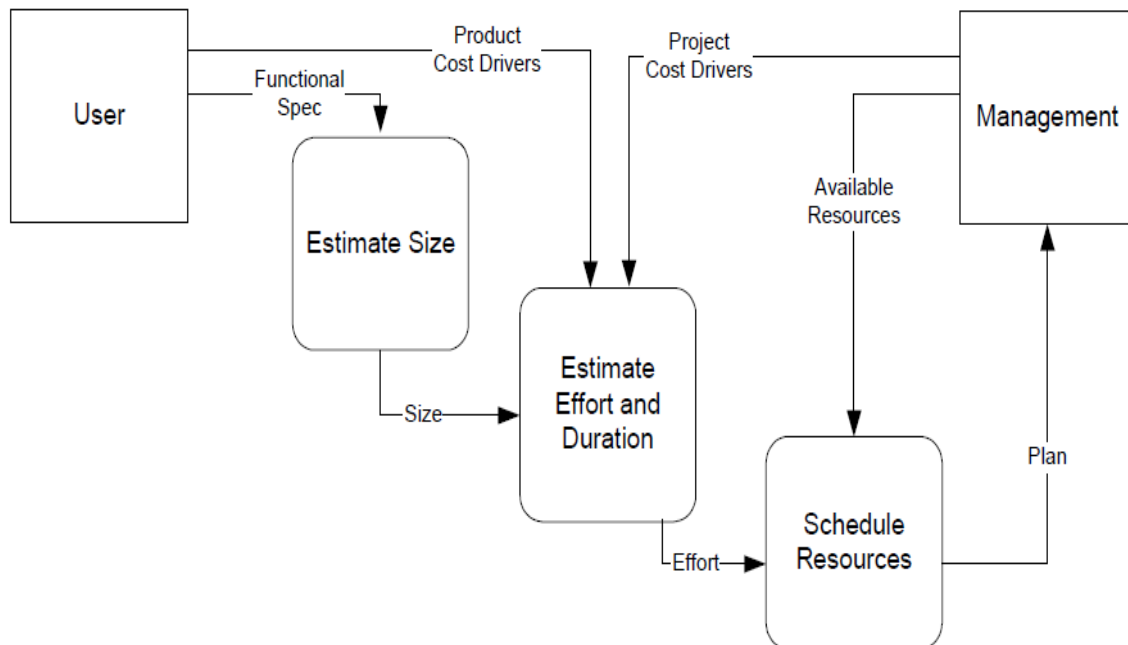


Figure 1.7 Framework for Software estimation

The following attributes are used in the above diagram:

Project size, cost, duration and effort.

### 1.2.1 Project estimation techniques

There are many approaches for project estimation. They are broadly classified into three categories:

#### a) Empirical estimation techniques:

This technique is based on making a guess of the various project parameters by the use of prior experience. However, it is subject to errors and individual bias. One of the improvements is to consider a decision by a coordinator and a group of experts. This is called as Delphi cost estimation.

In this technique, coordinator will provide a copy of SRS document to its estimators and with no discussion with each other they will give their individual estimates. This process is iterated for several rounds. [2]

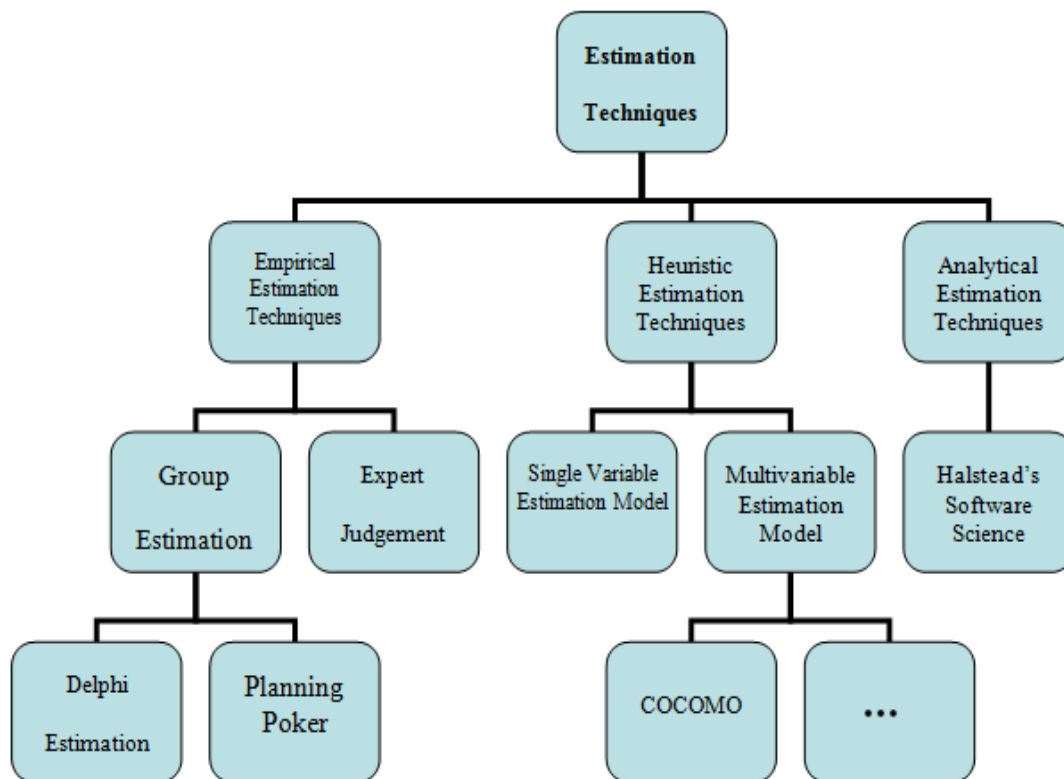


Figure 1.8: Estimation techniques



## b) Heuristic estimation:

This technique is based on the assumption that the relationship among the various project attributes can be made using mathematical expressions. It contains the single and multivariable estimation models which differ from each other in terms of parameters.

In single variable model, the estimated parameter is:

$$\text{Estimated parameter} = a_1 * e^{b_1}$$

Where  $a_1$  and  $b_1$  are constants from historical data and  $e$  is an independent variable or a characteristic of a software project, already estimated. An example can be 'size'.

In multivariable model, the estimated resource is  $= a_1 * (e_1)^{b_1} + a_2 * (e_2)^{b_2} + \dots$

An example is COCOMO. COCOMO is a constructive cost model developed by Barry Boehm. It is a statistical model of software effort and development. It uses the basic regression formula. There are three basic forms of COCOMO:

(i) Basic COCOMO: To get a rough order of software cost estimates. It computes effort as a function of program size. COCOMO applies to three classes of software projects viz.,

- Organic projects – In these projects the teams are small with good experience working with well understood domain.
- Semi-detached projects – Here the teams are medium and working with a mix of rigid and less than rigid requirements.
- Embedded projects - developed with lot of constraints. It contains projects that are hard. It is also combination of organic and semi-detached projects.

The basic COCOMO project model gives an estimate of project parameters:

$$\text{Effort Applied (E)} = a_i(\text{KLOC})^b_i \text{ in person-months.}$$

$$\text{Development Time (TD)} = c_i(\text{E})^d_i \text{ in months.}$$

(ii) Intermediate COCOMO: It computes effort as a function of program size and set of cost drivers.

$$E = a_i(\text{KLOC})^b_i * \text{Effort Adjustment Factor (EAF)}$$

(iii) Detailed COCOMO: It is an extension of intermediate COCOMO. It incorporates the assessment of the impacts of all the cost drivers on each step of the process.

COCOMO II: It is the major extension to the COCOMO. It estimates for system integration, application generator and infrastructure for two life cycles- Early Design and Post Design

CORADMO: CORADMO is the derivative of revised COCOMO II. It stands for Constructive Rapid Development Model. This method is meant for Rapid Application Development Projects. [3]

c) **Analytical techniques:** This technique derived the required result with basic assumptions of the project.

d) **Size estimating methods:**

There are various size based estimation models. The diagrammatic representation for the same is given below:

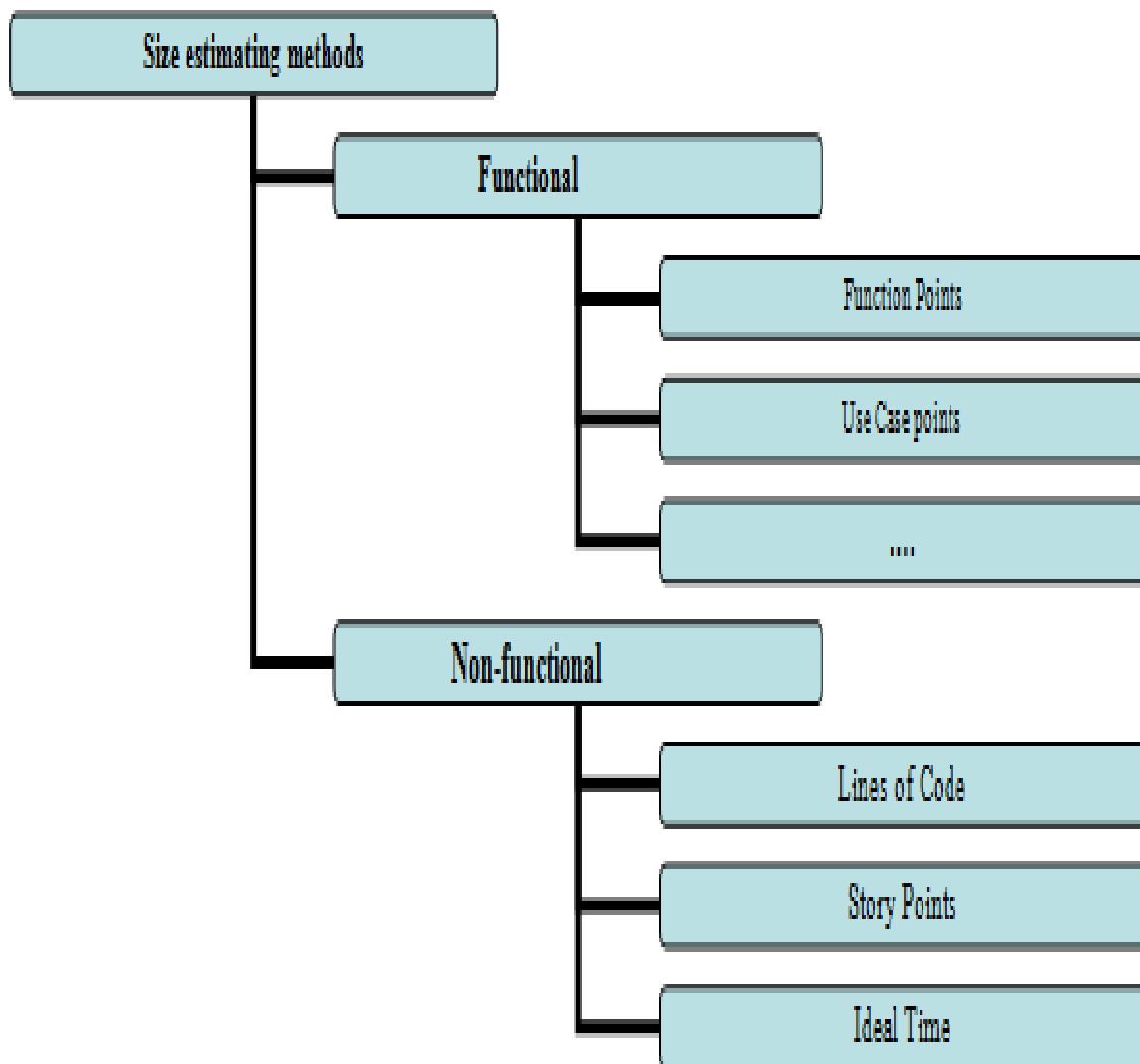


Figure 1.9: Size estimating methods

The various metrics for project size estimation includes LOC, Function point, Feature point, Use Case point, Story points. LOC refers to simply estimate the project by counting the number of lines in a source code. As per function point metric the size of the software depends upon the number of functions. Function points can be expressed as weighted sum of five characteristics and can be calculated in two parts. First by finding unadjusted Function Point (UFP) followed by technical complexity factor (TCF).  $UFP = (\text{Number of inputs}) * 4 + (\text{Number of outputs}) * 5 + (\text{Number of inquiries}) * 4 + (\text{Number of files}) * 10 + (\text{Number of interfaces}) * 10$  and TCF is computed as  $(0.65 + 0.01 * DI)$  where DI is Degree of influence. Finally,  $FP = UFP * TCF$ .

A use case point is an extension of function points. It is based on use case analysis. It has three types of actors – Simple, Average and Complex and weighing factor is assigned to them as 1, 2 and 3 respectively. Then UAW i.e. Unadjusted Actor Weights is calculated by adding the product of actors with their weighing factors. Then based on the number of transactions, unadjusted use case weights (UUCW) is calculated. Then unadjusted use case points (UUCP) is calculated by adding UUCW and UAW. Then Technical Complexity factor is calculated by applying the following formula:

$TCF = 0.6 + (0.01 * TFACTOR)$ . Then Environmental factors are calculated by the following formula:

$EF = 1.4 + (-0.03 * EfACTOR)$ . At last, Adjusted use case points are calculated as per the following formula:  $UCP = UUCP * TCF * EF$ . [4]

### **1.2.2 Agile Effort estimation**

Agile is flexible, so it is bit difficult to carry out effort estimation in it. We have already discussed that in agile, customer requirements are listed in the form of user stories. A user story is a very high-level definition of a requirement that contains just enough information to estimate, develop and test it. However, a user story must be implementable in a single iteration otherwise if not, the requirement is broken down to smaller stories. The collection of stories is known as Product Backlog. The units of work can be categorized in two ways viz., Real time units (hrs, days, etc) and abstract units (Story point, Ideal days). A good user story must adhere to INVEST [5].

The customer prioritizes the stories. The priorities could be in terms like "high, medium, low",

"Definitely needed, needed, and nice to have" or just numbers with higher numbers indicating higher priority. The responsibility of estimating the effort to implement the stories rests on the project team. Once prioritization and estimation are done, the customer's need to go to time-to-market is decided upon, and a tentative release date is arrived at.

Table 1.1 Qualities of Good user story

Letter	Meaning	Description
I	Independent	The user story should not have inherent dependency on another user story.
N	Negotiable	User stories can be changed and rewritten.
V	Valuable	A user story must deliver value to the end user.
E	Estimation	The size of the story should be able to get estimated.
S	Scalable	User stories should be of small size.
T	Testable	The user story must be testable in nature.

**Story Point:** A story point is a metric used in agile software development. It is an absolute unit to estimate the difficulty of implementing the user story. Many people use the Fibonacci sequence for estimating. Planning poker can also be used to estimate story points. It is a variation in Delphi.

**Agile Velocity:** In iteration, the number of points accomplished is agile velocity i.e. how fast a work can be completed. How long it will take to complete the project by reviewing previous sprints work. Agile methods are volatile in nature.

Some of the agile estimation methods are given below:

Table 1.2 Agile estimation approaches and problems [6]

<b>Technique</b>	<b>Approach</b>	<b>Problem</b>
<b>1.Analogy</b>	It is based on the knowledge and experience of several Project managers drawn from the results of many specific projects.	It gives unrealistic estimates.
<b>2.Top-down</b>	An overall cost estimate for the system is derived from global properties, using either algorithmic or non-algorithmic methods.	It is not considered as good software engineering practices.
<b>3.Bottom-up</b>	In this method each component of the software system is separately estimated and the results aggregated to produce an estimate for the overall system.	It is difficult to understand how the system is decomposed into different components
<b>4. Expert Opinion</b>	In this technique, an expert is asked how long a task will take to complete.	It is difficult to collect the opinion of experts.

## **1.3 Regression Testing**

Regression testing is the type of testing which is used to find out the side effects of the incorporated changes in the project. The main motive of this testing is to ensure that no faults are induced with the introduction of the new changes. It retests the software after the modification of the code so that other parts of the software don't get affected.

### **1.3.1 Need and purpose of Regression Testing**

The cost of regression testing is major in the cost of the project. It is required to calculate effort in man hours for effort estimation for regression testing. It is more important in Agile as agile is an iterative and incremental approach where user stories are implemented in each iteration. As in Agile development software undergoes lot of changes which is good as the software so developed is in accordance with the client requirements and recent market trends. The software development process needs to be of very flexible nature. Because of these rapid changes it is very obvious that any new update, new features addition and/or performance issues fix may affect the existing developed modules in terms of design, requirements and code.

### **1.3.2 Partial Regression testing**

It is not required to execute all the test cases. Only a fraction of test cases can be executed. There are number of techniques that are used to decide for this fraction of test cases.

- Regression Test Selection- It refers to the selection of subset of test cases from the test suite based on the modifications.
- Test Suite Minimization- Test suite is minimized such as it still maintains the same coverage as the original test suite was covering.
- Test Case Prioritization- The ordering of test cases is done in accordance with the priority. High priority test cases are executed earlier compared to less priority.

# REVIEW OF LITERATURE

---

**Ziauddin, Shahid Kamal Tipu, Shahrukh Zia (2012)** Effort Estimation Model for Agile Software Development. This paper presents a model for the effort estimation for the agile projects. The author first discusses about some cost estimation techniques and explains agile software development and its characteristics. Then techniques for effort estimation in agile which can be used are discussed. These estimating techniques include numeric sizing, t-shirt sizes, the Fibonacci sequence etc. It clearly states that the estimation is done by the team members in the sprint planning meeting for the stories of product backlog.

Story size scale is built which is basically an estimate of the relative scale of the work in terms of actual effort of development. Complexity of the project which may be because of user stories or technical complexity is measured on the complexity scale. These two values, the effort can be calculated for a particular user story which in turn can be summed up for total effort. Then concept of velocity is used for the calculation of finding out that how many units of effort the team can accomplish in one sprint. Then this velocity is optimized by taking into account the friction forces and the dynamic forces which reduces the project velocity. Then completion time is calculated followed by the calculation of development cost based on the data collected from the 14 CMMI level 3 companies. Then experimental analysis is made from the empirical data which was collected from 21 software projects. The results shows the estimated results are near to the actual results.

This paper opens number of research problems for the future investigation like the use of the number of scales for the estimation of effort like ranking scale or use of Fibonacci sequence. Moreover, the other factors that are affecting the velocity other than that are mentioned in the paper can be analyzed and more optimized results can be obtained. An improvement can be made on the estimation method by analyzing the major factors that seem missing in this approach. [7]

**Evita Coelho, Anirban Basu (2012)** Effort Estimation in Agile Software Development using Story Points. This paper discusses about the most acceptable approach in agile

methodology – Story Points. Story points are the unit of measurement of user stories which expresses its overall size. The effort and duration that is required for the delivery of features to the customer are estimated by the team member. The traditional methods of effort estimation are not appropriate for effort estimation. For story points approach, the estimation of the schedule and effort starts by understanding the customer's conditions of success and failure for the product backlog followed by the estimation of user stories and selection of iteration length, then estimation of velocity, prioritization of user stories and then estimation of delivery date.[8]

**Ratnesh Litoriya, Narendra Sharma, Abhay Kothari (2012)** Incorporating Cost Driver substitution to improve the effort using Agile Cocomo 2. In this paper, the author has analyzed the behavior of different cost drivers that are responsible for the prediction of cost of any project and then substitute it with its near values which will result in the decrease of cost of any project. The investigation is done on the 60 NASA past project data whose actual efforts are already given. And this data is then put into WEKA tool and K-mean clustering is applied on the data set which results in the formation of clusters. Then the value of these clusters is used to analyse the values of the cost drivers or in other words, the value of cost drivers get optimized with the formation of clusters. So the effect of the reduced values of cost drivers has direct impact on the cost of the project. This reduction of values of cost drivers which results in the reduction of cost of the project is calculated by the use of the online freely available web based tool AGILE COCOMO-II which was developed by the University of Southern California.

The future work of the paper says that the other data mining algorithms such as apriori etc. can be used to determine the better optimization of the cost drivers. And these optimized values can be applied on the web based AGILE COCOMO-II. This paper must had served a great problem definition by incorporating data mining techniques in Cost estimation and new combinations of these fields have come out. [9]

**Ritesh Tanmrakar, Magne Jorgensen (2012)** Does the use of Fibonacci Numbers in Planning Poker affect effort estimates. This paper gives the study of the affect of the use of Fibonacci numbers with respect to the linear numbers for effort estimation. Two case studies



have been performed for analyzing the same. In first case study, a group of students were divided into two groups – one for linear and the other for Fibonacci scale for effort estimation. The result showed a large difference in the values of effort estimations whereas linear scale shows higher value of the estimation. The second case study was performed with the set of experienced developers. The difference between the values of Fibonacci and linear scale estimations differ with smaller value. So, the use of Fibonacci is considered to be better than for the use of linear scale. [10]

**Zhamri Che Ani, Shuib Basri (2013)** A Case study of effort estimation in Agile Software Development using Use Case Points. In this paper, the author has investigated on how to estimate the effort for the software development in Agile Environment using Use Case Points. As calculating the initial efforts in Agile Projects is a challenge because of the volatile requirements in these projects. And implementing UCP is difficult in agile projects due to two reasons. First, the product backlog contains short descriptions of user stories which don't fit into the documentation standards of use case points. Secondly, none of the studies have clearly describes how to use Agile Product Backlog with this approach. So, the authors have successfully implemented this method in spite of its limitations.

For the implementation of this method, KOINS i.e. Kobena Information System's data was taken for analysis. The steps for UCP are followed which involves determining and computing of unadjusted use case points, technical complexity factors, environmental complexity factors, productivity factors and estimated number of hours. The estimated result was near to the actual result stating that UCP approach is suitable for estimating the efforts for software development at the early stages.

The future work states that other estimation models are needed to be compared with this method like COCOMO on Agile projects. The major challenge in this paper was the relation of the agility with the calculation is not clearly explained. The concept of user stories which is the baseline of agile projects seems to be disappeared in this paper. So, the problem definition can be formed by merging the concept of story points with use case points. The major challenge against this statement is availability of the data set for the analysis. [4]

**Abhilasha, Ashish Sharma (2013)** Test Effort Estimation in Regression Testing. Test Effort Estimation in Regression Testing. This paper explains the concept of regression testing and test effort estimation for the regression testing. Test effort estimation turns out to be costly if all test cases needs to be executed. So, there are various techniques used for the selection of test cases that minimally needs to be executed. An approach for the calculation of the Test effort estimation is proposed.

**Rashmi Popli (2014)** An Agile Software Estimation Technique based on Regression Testing Effort. This paper states that there is a major need for the inclusion of regression testing effort in the agile methodology. As Regression testing means to test if the incorporated changes are affecting the existing features or its side effects, it becomes mandatory to include the regression testing effort for the more accurate estimation of completion date, effort, cost and duration of the project. Further, this paper discusses 14 people and project related factors are taken into account and then regression testing time and effort is calculated which in-turn affects the total cost and time for the completion of the project. The future work of this paper states that other factors which affect the estimation to make it more efficient. [11]

**Govind Singh Rajput, Ratnesh Litoriya (2014)** CORAD Agile Method for Agile Software Cost Estimation. This paper presents the new method for software cost estimation for web based agile projects. CORADMO is basically a derivative of revised COCOMO 2. It is Constructive Rapid Development Model for Agile. CORADMO is an effort estimation technique used for RAD projects which can be used in agile projects which is named as CORAD\_AGILE. It explains three new cost drivers which are substituted with three old cost drivers. The three new cost drivers are degree of collaboration support, multisite development and daily basis customer interaction with vendor team. These new cost drivers are replaced with personnel, collaboration support and prepositioning assets. Effort, schedule and person productivity are calculated based on this model. [3]

**Rashmi Popli, Naresh Chauhan (2014)** Cost and Effort Estimation in Agile Software development. In this research paper, the author proposed an algorithm for Agile Effort and Cost estimation. It is a related work to the previous paper which takes into account the

concept of story points. This paper explains a mathematical estimation technique. The author also explained the life cycle of agile and explains the reason for the necessity of effort estimation in any project. The major causes which are responsible for inaccurate estimation in agile development are also discussed which includes the methodology adopted, the political forces like managerial pressure, improper communication between client and customer, management control problems like management reviews inaccuracy, uncertainty and self-knowledge. Then the existing agile estimation techniques which are available are given along with their problems.

Then the author proposed their own method for the estimation using story point approach. Total story points are calculated followed by the calculation of velocity which basically is the value computed by the story point completed in one iteration divided by story point in one user story. Then, decelerated velocity is calculated by considering the dynamic changes in agile environment. Then estimated development time, effort and cost are calculated. Then a case study is done using hypothetical values of the various factors. The future work of the paper states that other factors which affect the estimation can be added and estimation can be made more correct and efficient. [12]

**Rashmi Popli, Naresh Chauhan (2014)** Agile Estimation Using People and Project Related Factors. This paper presents the algorithmic estimation method based on the effect of various people and project factors. The author explains why it is necessary to include these factors and what problems peeps in if we don't include these factors in estimation. The author discussed 14 factors(both for project related factors and people related factors)which includes types of project, quality requirement, hardware and software requirements, ease of operation, complexity, data transaction, multiple site, communication skill, familiarity in team, managerial skill, security, working time, experience of previous projects and technical ability. The algorithm begins with the calculation of unadjusted values, quality factors and time factor. Based on these calculations, estimated story points and estimated time for the project. The future work states the inculcation of other factors which affect the estimation process to make the process of estimation more accurate and efficient. [6]

# SCOPE OF THE STUDY

---

Effort estimation is the most challenging task in the agile software development due to the volatility of the requirements. This study proposes the new estimation technique for Agile Software development. The traditional methods of estimation didn't work well so there was the need to improve them. This study covers the efficiency of already existing method by inculcating the missing factor which is also an important part of the Agile Software Development. This study covers the existing improved methods and the formation of new algorithms by merging the existing techniques.

- Story point based approach seems to work well in Agile. So there is a need to improvise it more by uncovering the hidden factors which are not quoted while doing the effort estimation.
- Regression Testing is an important part when incorporating the new changes. So, it has an important role in the agile projects.
- The merging of regression testing effort with the story points serves as a good problem definition and is of utmost importance.
- Regression testing will be the part of all the sprints except the first one. So, there is a need to find the way for the calculation of the test effort estimation for the each sprint. This study proposes the method for its calculation by considering the factors that are required for the calculation.

This study states that this technique is better in comparison to the simple story based effort estimation approach. This leads to the better MMRE (Mean Magnitude of Relative Error) value of effort in terms of completion time.

# OBJECTIVES OF THE STUDY

---

As effort estimation is a biggest challenge in agile projects due to two reasons. First, agile developers don't believe in detailed planning that helps in estimations. Second the requirements are volatile in nature. So, the estimation becomes difficult. These reasons are not the problems of agile, but it is the design of agile. This requires any such estimation technique which can accommodate this design. The main purpose of this work is to find out the method which is appropriate to the structure of agile methodology. The objectives are listed below:

1. To propose a new technique for effort estimation for agile projects that includes the effort of regression testing in addition to the already existing story point based approach.
2. To estimate effort using the new approach- E<sub>3</sub>RT (Effort Estimation including the Effort of Regression Testing).
3. To estimate test effort of the regression testing for all the iterations and added in the project development effort.
4. To compare the results of the E<sub>3</sub>RT, previous technique and the actual results to find out how near we are to the actual effort.
5. To calculate the MMRE value based on the completion time.

# RESEARCH METHODOLOGY

---

The research methodology includes the following three steps:

1. Formation of new technique  $E_3RT$  for effort estimation in order to get the improved value of the effort and completion time.
2. Implementation of the same on the dataset.
3. Comparison of the actual results, previous results and obtained results in terms of MMRE for completion time.

### 5.1 Tool Used:

MATLAB is Matrix Laboratory which is a multi paradigm computing environment. It allows easy implementation of algorithms with the easiness to use mathematical functions. MATLAB GUI is used for the implementation purpose so as to demonstrate the work easily and in an efficient way.

Features of MATLAB:

- Easy implementation of GUI (Graphical User Interface in MATLAB)
- Mathematical functions like Exponential functions, logarithmic functions, trigonometric functions etc
- Easy coding in high level language
- Easy use of global data and its manipulations
- Faster results

5.2 Activity Diagram for the E<sub>3</sub>RT technique:

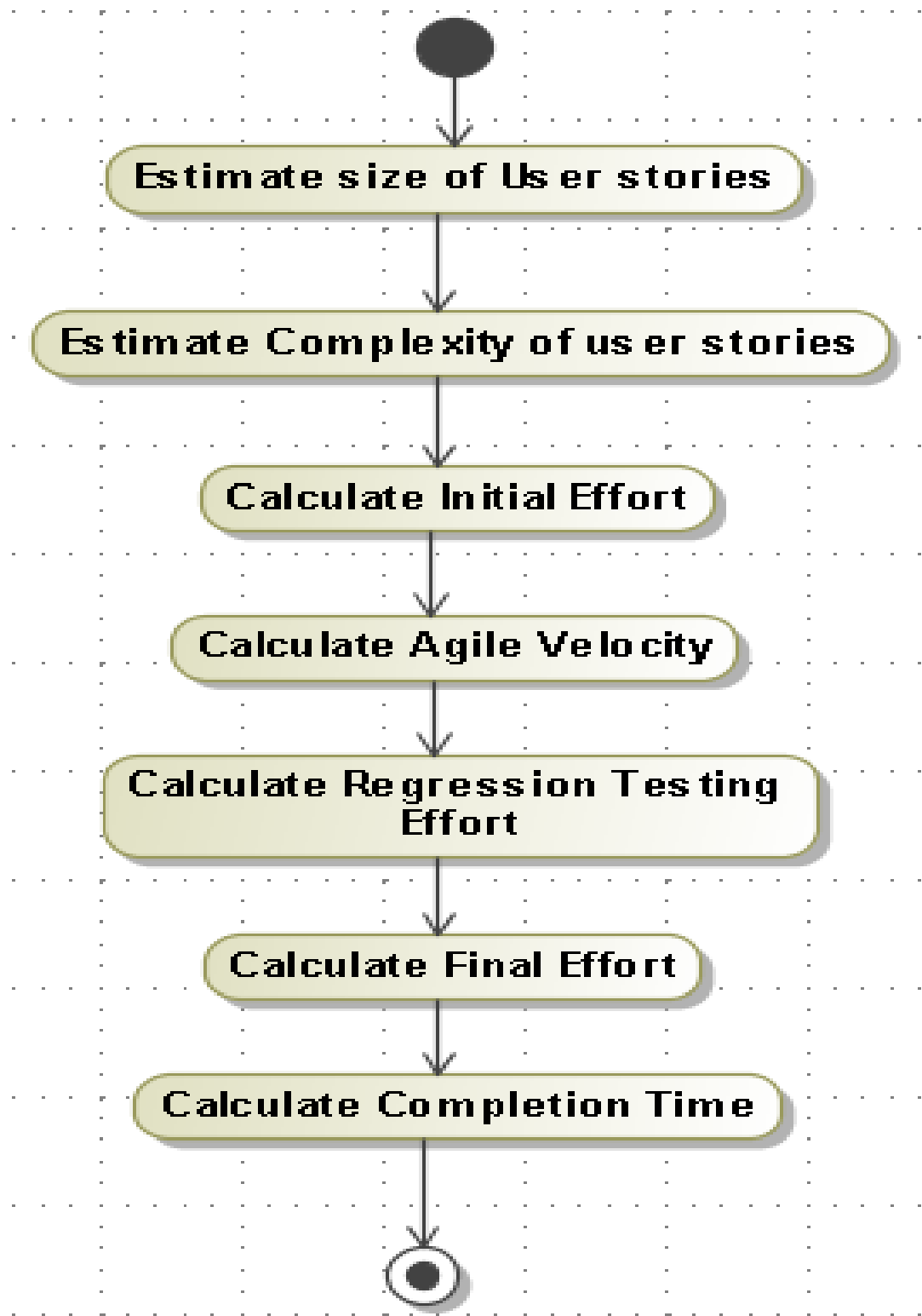


Figure 5.1: Activity Diagram for E<sub>3</sub>RT technique

### 5.3 Proposed Work:

The proposed technique E<sub>3</sub>RT involves the following steps:

1. **Selection of Sprint Size:** The duration of a single iteration is known as Sprint size. In Agile, sprint size is decided earlier unlike in traditional approaches where completion time was estimated according to the requirements. Here the number of stories to be implemented is based on the sprint size. Sprint size may be usually from one week to four weeks.
2. **Estimation for the user story size:** There are numerous ways of estimation of the user story which are proposed by the agile practitioners. Some of the proposed scales are as follows:
  - The selected user stories are rated according to the scale from 1 to 5 where 1 shows smallest and 5 shows epic which means too large story and should be broken down into smaller stories.
  - Scale based on Fibonacci series for the size of the story. For example: 3,5,8,13,21. This method is considered good as it gives fair differentiation among the size of the story and easier for estimator to tag in the suitable range.
  - Another method is T-Shirt size approach- XL, L, M, S, and XS. If the story size is bigger, it is again broken down into number of smaller stories. This is similar to the first scale as mentioned.
  - Another similar scale is dog breeds which is an innovative scale used by agilists. For e.g. the user story is Great Dane and the other is Chihuahua.

The team agrees upon one scale which they want to use in the project which should be followed for the whole estimation process.

In our approach, we have chosen the first scale which has rating from 1 to 5 as shown below:



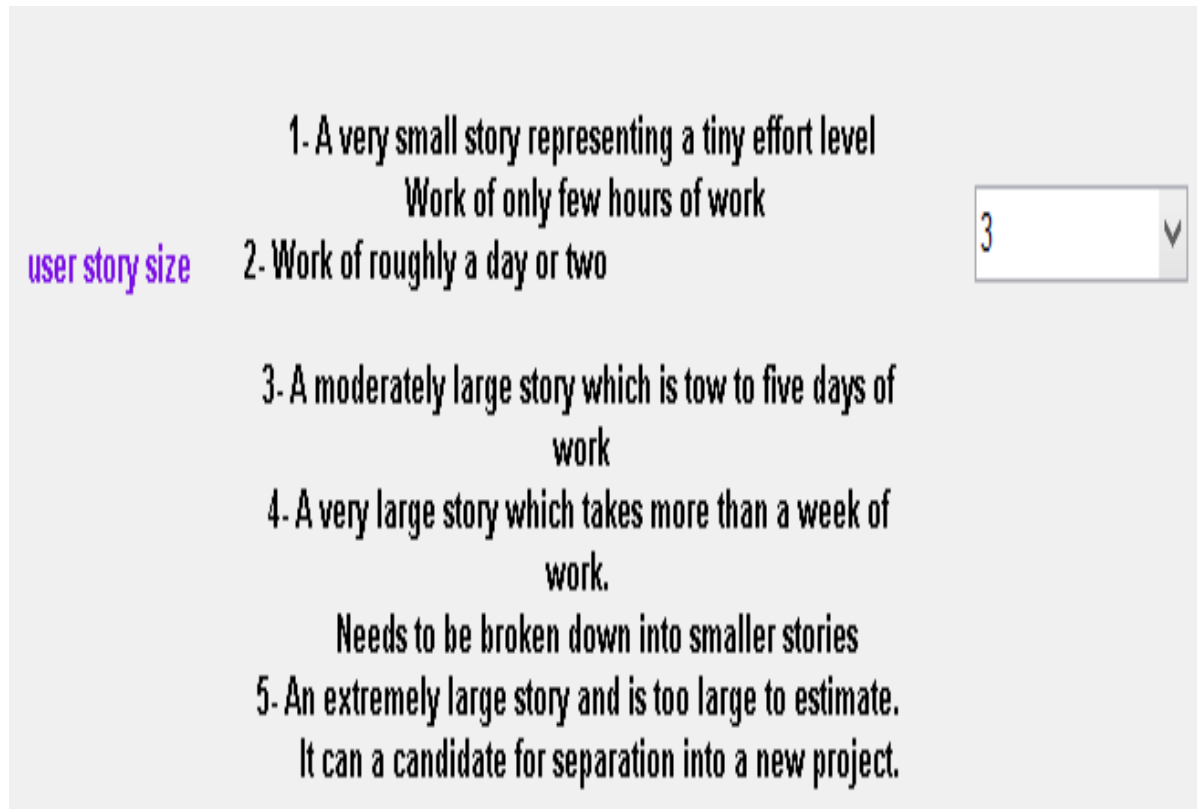


Figure 5.2: User Story Size Scale

**3. Estimation of user story complexity:** Complexity means the measure of uncertainty in terms of technicality or the requirements. It is again rated on the similar relative scale as used earlier for the size measurement.

**4. Calculation of Initial Effort based on size and complexity:** Initial effort is calculated in terms of story point which is the determined by the product of story size and the complexity for a particular user story. The total initial effort is calculated by the sum of the initial effort calculated per user story. So, the formulas are as follows:

$$E_i = S * C \quad \text{-Eq. 5.1}$$

where  $E_i$  is the effort per user story and  $S$  is user story size and  $C$  is user story complexity

$$E_{in} = \sum_{i=1}^n (SC) \quad \text{-Eq. 5.2}$$

where  $n$  is the total no. of user stories and  $E_{in}$  is the total initial effort.

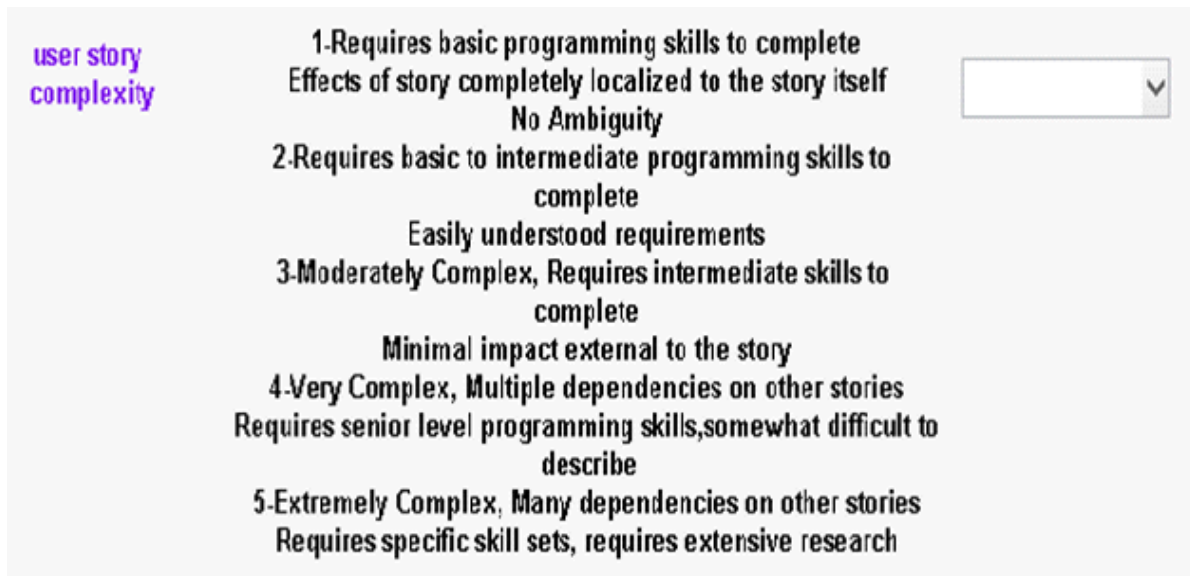


Figure 5.3: User Complexity Scale

**5. Calculation of Project Velocity:** Velocity is the term used to define ‘how fast the work is being done’. Project Velocity is calculated in two steps:

Step 1: Initial Velocity: It is equal to the initial effort divided by the initial total completion time which is the sum of the completion time assumed from the size of the individual user story.

Step 2: Calculation of final velocity based on decelerating factors: There are various factors that affect the velocity of the project. These factors that hinder the velocity are known as decelerating factors and the consideration of these factors in the final calculation of the velocity is known as optimization of velocity. There are 13 factors discussed by Ziauddin which are taken into consideration. These factors are shown in the Table No.

These values are taken based on the analogy. And the value of each factor is considered as per the severity of the situation. The severity is here categorized as stable, volatile, highly volatile and very highly volatile or can be stated as normal, high, very high or extra high.

The formula for calculating the deceleration is

$$D = \prod_{i=1}^{13} DF \quad \text{-Eq. 5.3}$$

Where D is the deceleration and DF is the value of Decelerating Factors.

Table 5.1: Decelerating Factors

Decelerating Factors	Very Volatile	Highly Volatile	Highly Volatile	Volatile	Stable
Team Dynamics	0.85	0.91	0.91	0.98	1
Process	0.89	0.94	0.94	0.98	1
Team Composition	0.91	0.95	0.95	0.98	1
Environmental Factors	0.96	0.98	0.98	0.99	1
Relocation	0.98	0.99	0.99	0.99	1
Team Changes	0.91	0.95	0.95	0.98	1
New Tools	0.96	0.97	0.97	0.99	1
Environmental Changes	0.97	0.98	0.98	0.99	1
Vendor Defects	0.90	0.94	0.94	0.98	1
Ambiguity	0.95	0.97	0.97	0.98	1
Responsibility of Team Member outside the project	0.98	0.98	0.98	0.99	1
Delay in response of stakeholders	0.96	0.98	0.98	0.99	1
Personal Issues	0.98	0.99	0.99	0.99	1

**6. Calculation of Regression Testing Effort:** Regression testing is the necessary part of the agile development methodology. It is being performed after all the sprints except the first one. To calculate the test effort estimation for the regression testing, we have

**i. Change Type (CT):** Change type basically means what kinds of changes are expected in the project – is it in the code, design or the requirement. Changes are always welcome in the agile methodology based on the feedback of the customer. The change can fall in the following three categories:

Table 5.2: CT and its related degree

Change Type	Degree
Requirement	3
Design	2
Code	1

**ii. Test Execution Complexity (TEC):** It is the measure of the complexity of the test in execution. The degree of test execution complexity is based on some factors shown below:

Table 5.3: TEC and its related degree

TEC scale	Characteristics	Degree
Low	<ul style="list-style-type: none"> <li>• Static item display</li> <li>• Requires Simulators</li> <li>• Special procedures due to company policy</li> </ul>	1
Medium	<ul style="list-style-type: none"> <li>• Dependency between Steps</li> <li>• Integration with external applications</li> <li>• Requires support team due to Access authorization</li> </ul>	2
High	<ul style="list-style-type: none"> <li>• Logical dependency between data</li> <li>• UI actions/feedback</li> <li>• Business Process</li> <li>• Data Volume</li> <li>• Network Conditions</li> </ul>	3

**iii. No. of Selected Test Cases (NSTC):** By the term NSTC, it means that how many number of test cases needs to be executed. It is very expensive to retest all the test cases, so some set of test cases are executed which are expected to have some impact of the changes. The degree is assigned based on the number of test cases.

Table 5.4: NSTC and its related degree

NSTC	Degree
Few	1
Some	2
All	3

The term All means all the test cases are needed to be executed but it is rarely used. As the retest all is very expensive so test selection is made. It not only saves cost but also the time of execution. The term Few and Some vary from project to project which is based on the number of test cases having higher risk factor or coverage techniques.

**iv. Testing Team Productivity (TTP):**

The test effort largely depends on the productivity of the testing team. If the team is experienced, then the effort made by the team will be less and vice versa.

Table 5.5: TTP and its related degree

Category	Degree
Experienced	3
Mediocre	2
Inexperienced	1

With degree 1, it defines the tester is inexperienced or a fresher who is having lesser knowledge about the application. With degree 2, it defines the tester as mediocre having experience in the testing but may not have experience with the testing of similar applications. With degree 2, it defines the tester as experienced professional having familiarity with the similar applications.

In other words, productivity is inversely proportional to the effort.

The formula for the calculation of test effort estimation for regression testing per iteration ( $RTE_{PI}$ ) is:

$$RTE_{PI} = ((CT * NSTC) + TEC) / TTP \quad \text{-Eq. 5.4}$$

Where CT is change type, NSTC is no. of selected test cases, TEC is test execution complexity and TTP is testing team productivity.

Similarly the total regression testing effort is calculated by:

$$RTE = (n-1) * RTE_{PI} \quad \text{-Eq. 5.5}$$

Where n is the no of iterations and n is calculated by the formula:

$$n = E_{in} / SS \quad \text{-Eq. 5.6}$$

where  $E_{in}$  is initial effort and SS is sprint size

**9. Calculation of Final Effort:** Final Effort will be equal to the sum of the initial effort and total regression testing effort.

$$E = E_{in} + RTE \quad \text{-Eq. 5.7}$$

Where  $E_{in}$  is the initial effort and RTE is the total regression testing effort.

**10. Calculation of Development Time:** Development time T is calculated as :

$$T = E / V \quad \text{-Eq. 5.8}$$

where E is the total effort and velocity.

# RESULTS AND DISCUSSIONS

---

The results and discussions is an important part for any research. Let's have a look on the results:

1. **Estimation for the user story size:** Assume there is a large story which seems to be the work of two to five days. So, it is rated as 3 according to user story size scale:

**user story size**

- 1- A very small story representing a tiny effort level  
Work of only few hours of work
- 2- Work of roughly a day or two
- 3- A moderately large story which is tow to five days of work
- 4 A very large story which takes more than a week of work.  
Needs to be broken down into smaller stories
- 5- An extremely large story and is too large to estimate.  
It can a candidate for separation into a new project.

3

The diagram shows a vertical list of five user story size levels. To the right of the list is a dropdown menu with the number '3' selected. The background is a light gray gradient.

Fig 6.1: User Story Size Scale

2. **Estimation of user story complexity:** The same user story requires intermediate programming skills and has easily understood requirements, then it falls under the category of degree 2 in the scale.

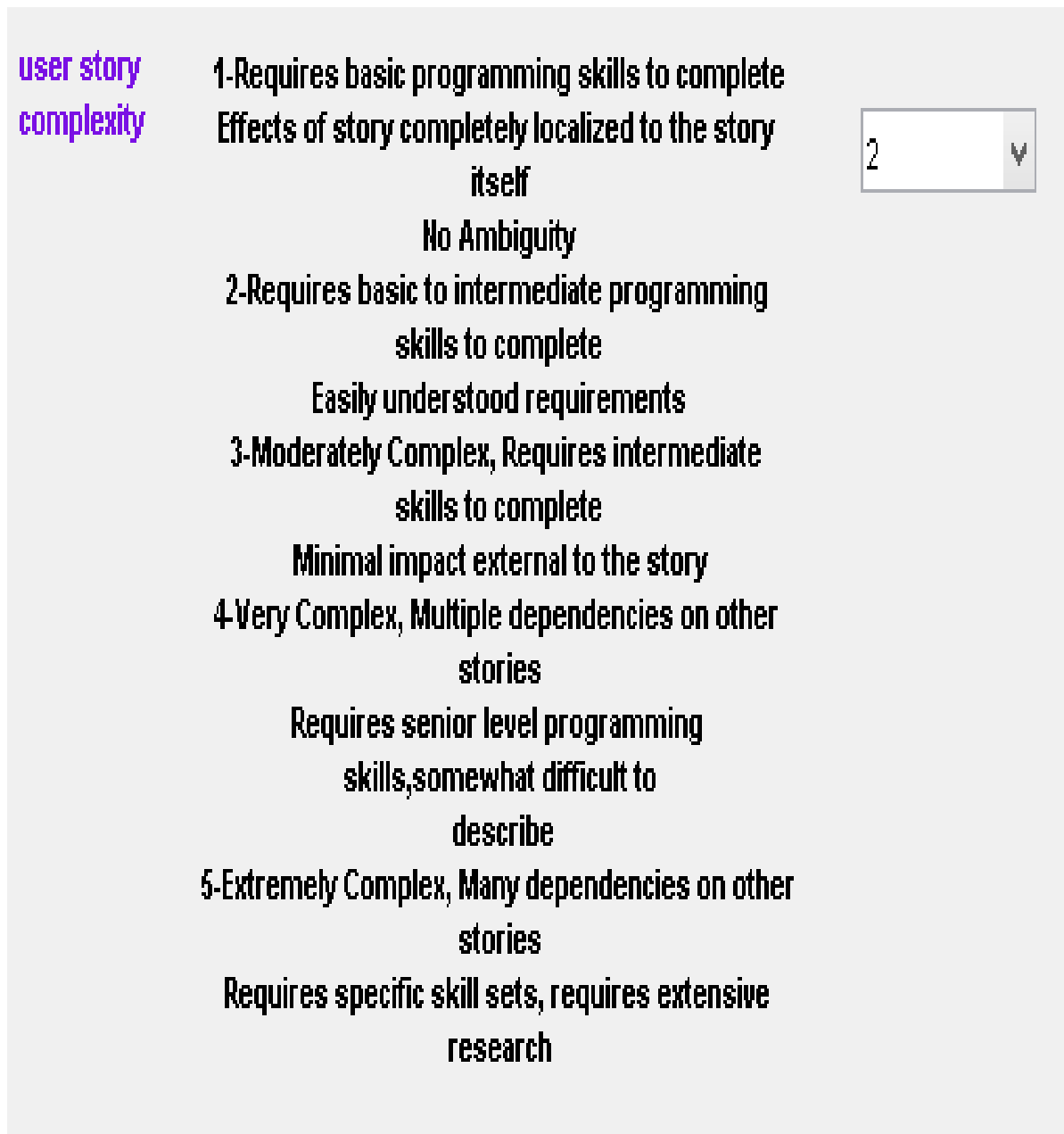


Figure 6.2: User Complexity Scale

3. **Calculation of Initial Effort based on size and complexity:** Initial effort is the product of story size and the story complexity for a particular user story.



If we assume we have 40 user stories and each user story carries the same size 3 and complexity as 2, then the total initial effort would be the sum of all the efforts of each user story which comes out to be 240 SP. The calculation for the same is shown Figure 6.3.

The screenshot shows a window titled "form1\_thesis" with the following content:

- user story size** (dropdown menu): 3
  - 1- A very small story representing a tiny effort level  
Work of only few hours of work
  - 2- Work of roughly a day or two
  - 3- A moderately large story which is tow to five days of work
  - 4- A very large story which takes more than a week of work.  
Needs to be broken down into smaller stories
  - 5- An extremely large story and is too large to estimate.  
It can a candidate for separation into a new project.
- user story complexity** (dropdown menu): 2
  - 1-Requires basic programming skills to complete  
Effects of story completely localized to the story itself  
No Ambiguity
  - 2-Requires basic to intermediate programming skills to complete  
Easily understood requirements
  - 3-Moderately Complex, Requires intermediate skills to complete  
Minimal impact external to the story
  - 4-Very Complex, Multiple dependencies on other stories  
Requires senior level programming skills,somewhat difficult to describe
  - 5-Extremely Complex, Many dependencies on other stories  
Requires specific skill sets, requires extensive research
- Calculate Initial Effort per user story** (button)
- Enter total user stories** (input field): 40
- Total Initial Effort** (input field): 240
- Initial Velocity** (input field): [empty]
- Next** (button)

Figure 6.3: Calculation of the initial effort

**4. Calculation of Project Velocity:** The Initial velocity for the corresponding project will be 2 for the values of the user story.

The screenshot shows a web form titled "form1\_thesis" with the following elements:

- user story size** (dropdown menu): 3
  - 1- A very small story representing a tiny effort level  
Work of only few hours of work
  - 2- Work of roughly a day or two
  - 3- A moderately large story which is tow to five days of work
  - 4- A very large story which takes more than a week of work.  
Needs to be broken down into smaller stories
  - 5- An extremely large story and is too large to estimate.  
It can a candidate for separation into a new project.
- user story complexity** (dropdown menu): 2
  - 1-Requires basic programming skills to complete  
Effects of story completely localized to the story itself  
No Ambiguity
  - 2-Requires basic to intermediate programming skills to complete  
Easily understood requirements
  - 3-Moderately Complex, Requires intermediate skills to complete  
Minimal impact external to the story
  - 4-Very Complex, Multiple dependencies on other stories  
Requires senior level programming skills,somewhat difficult to describe
  - 5-Extremely Complex, Many dependencies on other stories  
Requires specific skill sets, requires extensive research
- Calculate Initial Effort per user story** (button): 6
- Enter total user stories** (input field): 40
- Total Initial Effort** (input field): 240
- Initial Velocity** (input field): 2
- Next** (button)

Figure 6.4 Calculation of Initial Velocity

For the optimization of velocity, the decelerating factors are considered as shown below:

The screenshot shows a software application window titled "form2\_thesis" with a pink header bar. The main content area is titled "Optimization of Velocity" and contains a form with the following elements:

- Team Composition:** Dropdown menu set to "Highly Volat...".
- Team member's responsibilit'es outside project:** Dropdown menu set to "Very Highly...".
- Process:** Dropdown menu set to "Very Highly...".
- Expected Change in Environment:** Dropdown menu set to "Highly Volat...".
- Environmental Factors:** Dropdown menu set to "Volatile".
- Expected Relocation:** Dropdown menu set to "Very Highly...".
- Team Dynamics:** Dropdown menu set to "Very Highly...".
- Expected Team Changes:** Dropdown menu set to "Highly Volat...".
- Calculate Dynamic Forces:** Button with a corresponding output box containing the value "0.55152".
- Intro of New tools:** Dropdown menu set to "Very Highly...".
- Vendor's Defect:** Dropdown menu set to "Volatile".
- Final Velocity:** Button with a corresponding output box containing the value "1.46562".
- Personal Issues:** Dropdown menu set to "Highly Volat...".
- Expected Delay in Stakeholder response:** Dropdown menu set to "Very Highly...".
- Expected Ambiguity in Details:** Dropdown menu set to "Very Highly...".
- Next:** Button located at the bottom right of the form.

Figure 6.5 Decelerating factors of velocity

Based on the above three steps, the following table for effort and velocity is generated for 21 previously developed projects which used agile methodology:

Table 6.1 Initial Effort and velocity

Initial Effort	Velocity
156	2.7
202	2.5
173	3.3
331	3.8
124	4.2
339	3.6
97	3.4
257	3
84	2.4
211	3.2
131	3.2
112	2.9
101	2.9
74	2.9
62	2.9
289	2.8
113	2.8
141	2.8
213	2.8
137	2.7
91	2.7

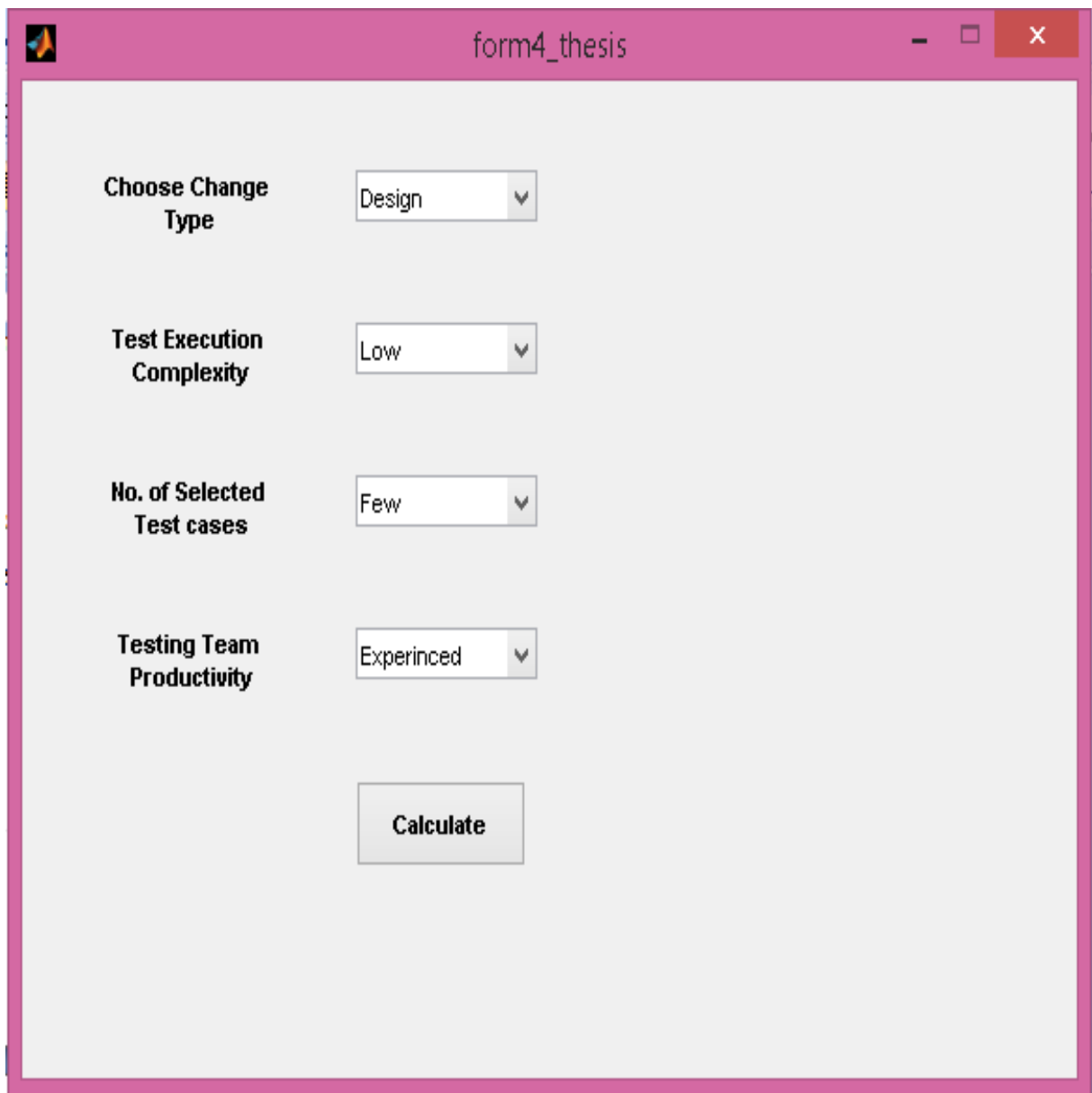
Now is the time for the calculation of the regression testing effort.

**5. Calculation of the Regression testing Effort:** For each project, the parameters for the effort estimation of regression testing differ. So, we have used to do calculation using

different combinations. The better approach to do it is to eradicate the least practiced situations and calculate for degrees which are assumed to have higher probability.

**Ideal Case:** The ideal case for any agile project will be when the test execution complexity is low, no. of selected test cases will be few and the testing team productivity is high i.e. experienced. Design, code, and requirement states the nature of change which has nothing to do with the ideal case so we get three ideal cases have been taken with the different change types-

- (i) When CT is Design:



The image shows a screenshot of a software application window titled "form4\_thesis". The window has a pink header bar with standard window controls (minimize, maximize, close) on the right. The main content area is light gray and contains four dropdown menus, each with a label to its left and a downward arrow on its right. The labels and selected values are: "Choose Change Type" (Design), "Test Execution Complexity" (Low), "No. of Selected Test cases" (Few), and "Testing Team Productivity" (Experinced). Below these dropdowns is a single "Calculate" button.

Figure 6.6: Ideal Case for the any agile project

form3\_thesis

<b>Initial Effort</b>	156		
<b>Velocity</b>	2.7		
<b>Regression Testing per iteration</b>	0.66667	Calculate	Get Value
<b>Total Regression testing effort</b>	9.7333	Calculate	
<b>Final Effort</b>	165.7333	Calculate	
<b>Estimated Completion Time</b>	61.3827	Calculate	
<b>Actual Completion time</b>	63		
<b>Time MRE</b>	0.025671	Calculate	

Figure 6.7 Calculation Based on Case1

(ii) When CT is Code :

Field	Value	Action
Initial Effort	156	
Velocity	2.7	
Regression Testing per iteration	1	Calculate, Get Value
Total Regression testing effort	14.6	Calculate
Final Effort	170.6	Calculate
Estimated Completion Time	63.1852	Calculate
Actual Completion time	63	
Time MRE	-0.0029394	Calculate

Figure 6.8 Calculation based on Case 2

(iii) When CT is requirement

<b>Initial Effort</b>	156		
<b>Velocity</b>	2.7		
<b>Regression Testing per iteration</b>	1.3333	Calculate	Get Value
<b>Total Regression testing effort</b>	19.4667	Calculate	
<b>Final Effort</b>	175.4667	Calculate	
<b>Estimated Completion Time</b>	64.9877	Calculate	
<b>Actual Completion time</b>	63		
<b>Time MRE</b>	-0.03155	Calculate	

Figure 6.9 Calculation based on Case 3



If we compare this value with previous approach and actual results, we can conclude that we are more near to the actual results than the previous approach.

Table 6.2: Comparison of results of four projects out of 21 projects

Project No.	Initial Effort	Actual Completion time	Previous technique's Completion time	E <sub>3</sub> RT Case I	E <sub>3</sub> RT Case II	E <sub>3</sub> RT Case III
1	156	63	58	61.3	63.1	64.9
2	202	92	81	85.9	88.4	91.0
3	173	56	52	55.7	57.3	59.0
4	124	32	29	31.33	32.23	33.14

However, the figure for the comparison of four projects with E<sub>3</sub>RT case 1 is given below:

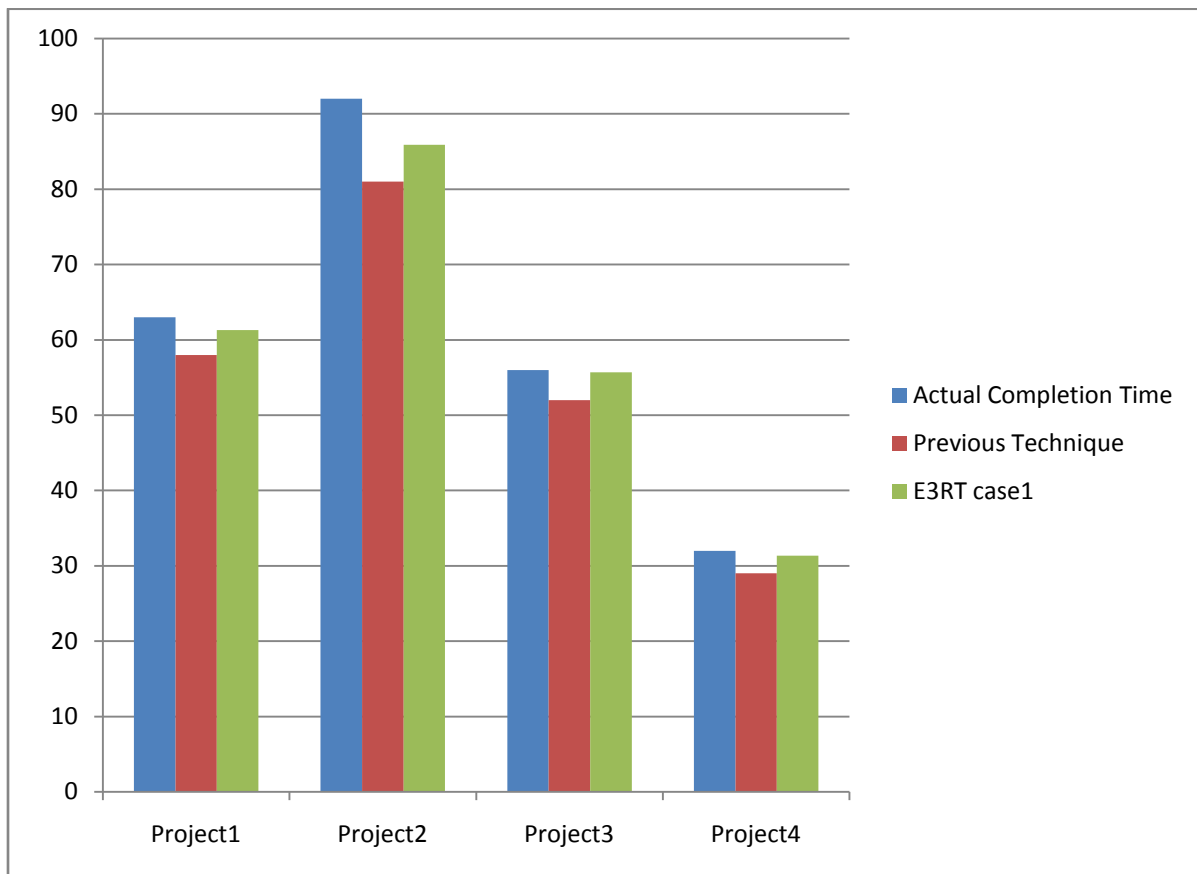


Figure 6.10: Comparison of results for four projects with E<sub>3</sub>RT case 1

The figure for the comparison of four projects with E<sub>3</sub>RT case 2 is given below:

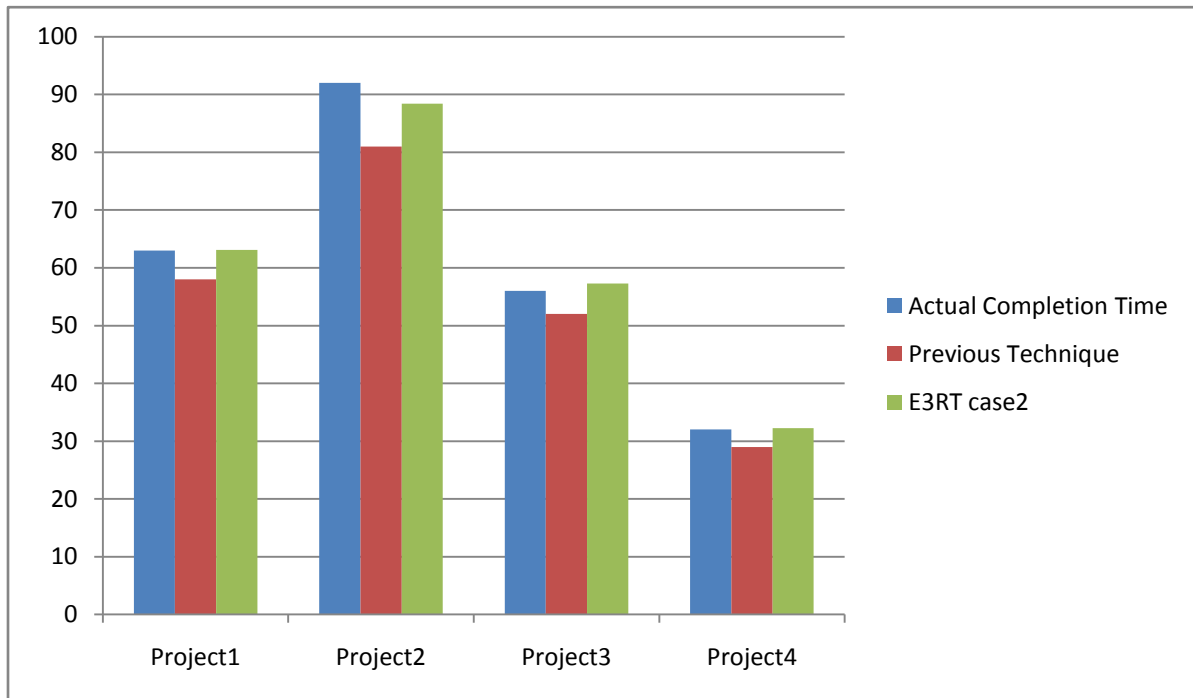


Figure 6.11: Comparison of results for four projects with E<sub>3</sub>RT case 2

The figure for the comparison of four projects with E<sub>3</sub>RT case 3 is given below:

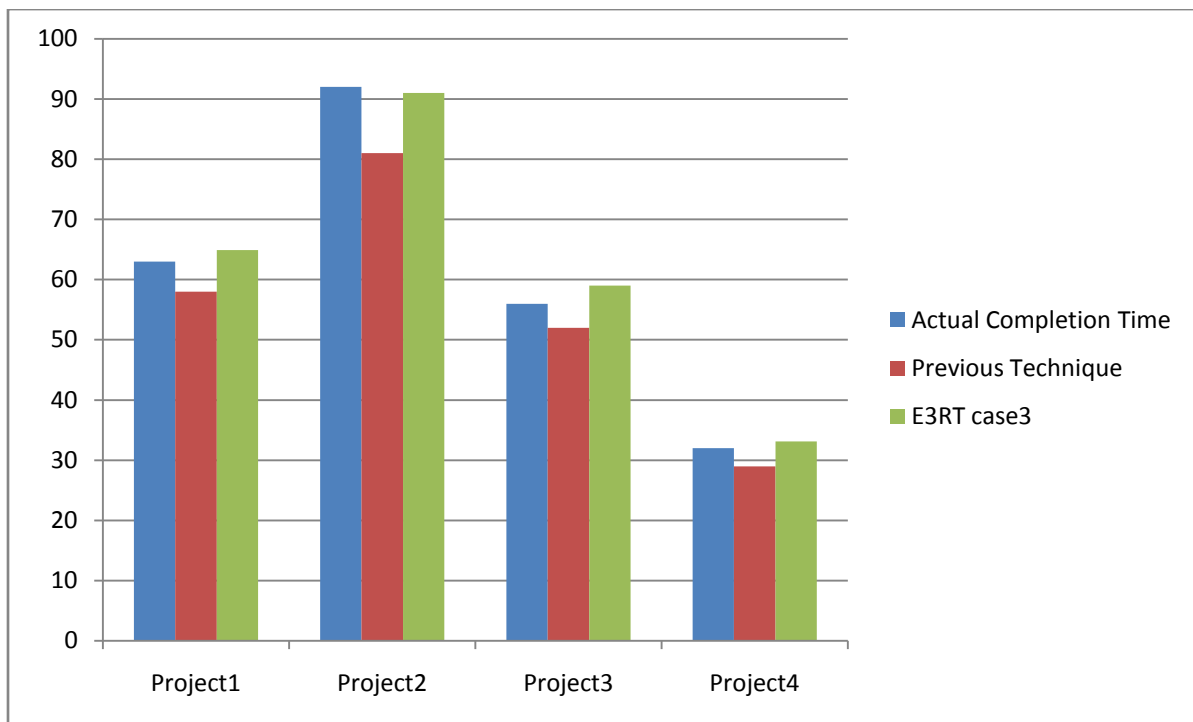


Figure 6.12: Comparison of results for four projects with E<sub>3</sub>RT case 3

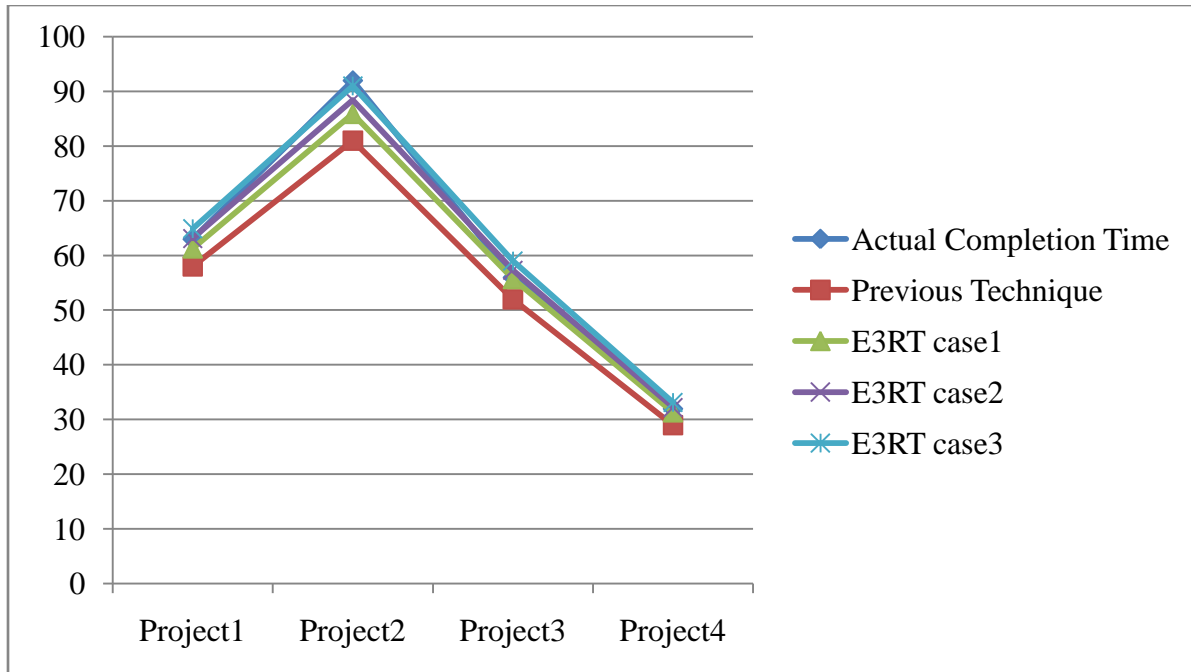


Figure 6.13 Comparison of results

Case II: Worst Case: Worst Case here means that the TEC complexity is high or medium, and the NSTC are many and the testing professionals are mediocre or inexperienced. This case may hold true only in certain cases. But initially no team would assume such conditions in the initial stage of the estimation due to some reasons. First, TEC and NSTC both indirectly depend on the nature of user stories. And user stories should be made as independent as possible. But the dependency can't be completely eradicated so we take minimum consideration of the dependency. Moreover, if we consider about the testing team productivity, Agilest prefer experienced professionals.

For instance if we consider the worst case, it can double the effort and increase the completion time to a great extend.

MMRE value: The effort estimation for all the algorithms is compared based on low MMRE value to get the best suitable algorithm. MMRE is expected to be near to zero.. For calculating the MMRE, following process is used.

$$\text{Error} = \text{Actual Effort} - \text{Estimated Efforts}$$

Relative Error (RE) = (Actual Effort – Estimated Efforts)/ Actual Efforts

MRE = abs(RE)

MMRE is the summation of MRE for number of projects.

Table 6.3: Calculation of error and MRE value with respect to Case1

Project	Error	MRE value
1	1.7	0.0269
2	6.1	0.0663
3	0.3	0.0053
4	0.67	0.0209

MMRE = 0.02985

Now, let's have a comparison of the previous technique and E<sub>3</sub>RT technique. The percentage of accuracy is given in Table 6.4.

Table 6.4: Percentage comparison of previous technique and E<sub>3</sub>RT

Actual Completion time	Previous technique's Completion time	E <sub>3</sub> RT Case I	Percentage of accuracy of previous technique	Percentage of accuracy of E <sub>3</sub> RT Technique
63	58	61.3	92.063%	97.301%
92	81	85.9	88.043%	93.369%
56	52	55.7	92.857%	99.464%
32	29	31.33	90.625%	97.906%

# Summary and Conclusions

---

This work proposes an effective technique E<sub>3</sub>RT for the effort estimation of the Agile Based projects. This technique uses story points and regression testing effort as the base of this estimation. Calculation of regression testing effort was a challenge in the agile projects. This work has eliminated this limitation to some extent. As regression testing is must in Agile projects, its effort needs to be added. This is the reason that this new technique is more appropriate as compared to the previous technique. Also, the results given by this new technique are 5 to 10% closer to the actual effort than the previous technique.

This work opens a number of directions for which work can be carried out in the future. NSTC factors can be studied in more detail. Neural Networks is a powerful branch to carry out estimations for traditional methods. They can be implemented along with these techniques. This work will serve as an opening for further discussion and investigation.

## References

---

- [1] Laurie W., Gabe B. (2009) “Scrum + Engineering Practices: Experiences of Three Microsoft Teams” , ESEM11
- [2] Sinhal A. , Verma B. “Software Development Effort Estimation: A Review” ,IJARCSSE, ISSN: 2277 128X
- [3] Rajput, G.S., Litoriya,R. (2014) “CORAD Agile Method for Agile Software Cost Estimation”, Open Access Journal,1:e579
- [4] Ani,Z.C., Basri,S. (2013), “A Case study of effort estimation in Agile Software Development using Use Case Points” Agile Symposium, Malaysia ISSN 1013-5316
- [5] [http://en.wikipedia.org/wiki/INVEST\\_\(mnemonic\)](http://en.wikipedia.org/wiki/INVEST_(mnemonic))
- [6] Popli,R., Chauhan,N.(2014) “ Agile Estimation Using People and Project Related Factors” Published in IEEE,978-93-80544-12-0/14
- [7] Ziauddin, Tipu, S.K., Zia, S. (2012), “An Effort Estimation Model for Agile Software Development”, ACSA, ISSN 2166-2924
- [8] Coelho, E., Basu, A. (2012), “Effort Estimation in Agile Software Development using Story Points”, IJAIS, ISSN 2249-0868
- [9] Litoriya, R., Sharma, N., Kothari, A. (2012), “Incorporating Cost Driver substitution to improve the effort using Agile Cocomo 2”, Published in IEEE, CONSEG, ISBN 978-1-4673-2174-7
- [10] Tanmrakar,R. , Jorgensen,M. (2012), “Does the use of Fibonacci Numbers in Planning Poker affect effort estimates” , Published by IET, Proceedings of EASE, ISBN 978-1-84919-541-6
- [11] Popli,R., Chauhan,N. (2013), “An Agile Software Estimation Technique based on Regression Testing Effort”, Annual International Software Testing Conference in India”
- [12] Popli,R., Chauhan,N. (2014) “Cost and Effort Estimation in Agile Software development” Published in IEEE, ICROIT 978-1-4799-2995-5/14
- [13] R. Martin(2003), “Agile software development: principles, patterns, and practices”. Conference New York : Prentice Hall

[14] Nassif A., Danny H.(2011) “Regression Model for Software Effort Estimation Based on the Use Case Point Method” Conference IPCSIT vol.14 IACSIT Press

[15] M. Fowler and J. Highsmith(2001), “The agile manifesto,” Software Development, vol. 9, no. 8, pp. 28–35