Chapter IV

A SPRINT-POINT BASED ESTIMATION FRAMEWORK IN SCRUM: PROPOSED WORK

4.1 PROBLEMS IN AGILE ESTIMATION

Based on literature study [57,58] it has been found that most of the existing effort estimation techniques have been developed to support traditional sequential software development methodologies whereas ASD is iterative and dynamic in nature. If these traditional techniques are used for effort estimation of Agile software projects, then the results will be definitely inaccurate.

Various approaches have been suggested by different researchers for estimation which are discussed below:

O. Benediktsson, Dalcher [69] performed a controlled experiment to investigate the impact of software development approach on the resulting product and its attributes by comparing V-model (VM), evolutionary model (EM), incremental model (IM), and extreme programming (XP). The conclusion was that: XP groups spent significantly less time in requirements specification than V-model evolutionary model groups, XP produced significantly more LOC in general than all other methodologies, XP produced significantly lower number of pages per Person Months (PM) than all other methodologies, XP produced significantly higher pages per PM than VM, no differences in total pages per PM between methodologies.

S. Bhalerao, Maya Ingle [98] introduced an algorithm to calculate the development cost, time and effort. The need for mathematical algorithm arises due to the limitations of the previous work. The authors have considered some of the factors that affect the estimates. The name of the algorithm is termed as Constructive Agile Estimation Algorithm
This algorithm uses the vital factors mainly; project domain, configuration, performance, complex processing, data transaction, operation ease, multiple sites and security to calculate effort, time and cost. This algorithm helps reduce the risks factors.

Buglione [11] investigated the effort estimation activity within Agile software development methodology. He briefly described the estimation approaches used in various Agile software development methods.

P. Abrahamsson, J. Koskela [72] demonstrated how to collect software metrics to measure effort estimation, productivity, quality and schedule estimation and cost estimation for a Software project using XP.

Based on the critical study of various research papers it has been found that there are several problems in existing estimation and tracking methods as discussed below.

The first problem is that if the estimates are unrealistically low, the project will be understaffed and the resulting excessive overtime or staff burnout will cause attrition and compound the problems facing the project. Overestimating a project have the problems like overstaff and over cost. Thus, an effort metric is needed that calculates the size of the Agile team.

Second, the story points and velocity are used to estimate the initial size of the project, there are magnitudes of factors which can affect the story points and decelerate the velocity and thus affect on productivity of team. Although S.Bhalerao and Maya Ingle [98] identified some vital factors that affect estimation, there are various other factors that must be considered to check the affect on velocity. So, there is a need to find out a comprehensive list of various factors which can affect the velocity and thus estimation.

Third, in Agile environment, at the initial stage of a project, there is high uncertainty about various project attributes. The estimates produced at early stages are inaccurate, as the accuracy depends highly on the amount of reliable information available to the
estimator. Agile estimation methods may lead to the errors in case of inexperienced Agile team. Therefore, there is strong need of analyzing the uncertainty that may affect the estimation of the Agile project.

Considering the characteristics of ASD methodology and all the problems discussed above, an effort estimation framework to predict development effort of Agile software project is proposed in this chapter.

4.2 PROPOSED SPRINT-POINT BASED ESTIMATION FRAMEWORK IN SCRUM

When planning about first sprint, at least 80% of the backlog items are estimated to build a reasonable project map. These backlog items consist of user-stories grouped in sprints and user-stories based estimation is done using story-points [80,81,84]. When a team member estimates that a given task can be completed within 8 hours it does not mean that he can complete the task in 8 hrs. Because no one can sit in one place for the whole day and there can be a number of factors that can affect story-points and hence decrease the velocity.

To resolve this problem the concept of Sprint-point is proposed. A Sprint-point basically calculates the effective story-points. Sprint point is an evaluation unit of the user story instead of story-point. By using Sprint points, more accurate estimates can be achieved. Thus, the unit of effort is Sprint Point (SP) which is the amount of effort, completed in a unit time.

In the proposed sprint-point based estimation framework, requirements are first gathered from client in the form of user-stories. After requirement gathering, a user-story based prioritization algorithm is applied to prioritize the user-stories. Consequently story-points in each user-story are calculated and uncertainty in story-points is removed with the help of three-types of story-points proposed. Then these story-points are converted to sprint-
points based on the proposed Agile estimation factors. Afterwards sprint-point based estimation algorithm is applied to calculate cost, effort and time in a software project.

As regression testing [94] is a necessary but expensive activity aimed at showing that code has not been adversely affected by changes. So defect data is gathered based upon the similar kinds of projects, which is used to calculate rework effort and rework cost of a project. Finally the sprint-point based estimation algorithm is applied to calculate the total cost, effort and duration of the project.

This **Sprint-point Based Estimation Framework** as shown in Figure 4.1 performs estimation in scrum using below steps:

**Step 1:** User-stories are prioritized by using User-story Based Prioritization Algorithm (will be discussed in section 4.3).

**Step 2:** As Agile is highly dynamic so uncertainty cannot be removed completely. Some steps can be taken to reduce uncertainty (will be discussed in section 4.4).

**Step 3:** Story-points are converted into sprint-points by considering Agile estimation factors like project-related, people-related and resistance factors. These factors affect the user-stories and thus affect the cost, effort and duration of a software project. (will be discussed in section 4.5).

**Step 4:** Regression testing is done in Agile to make sure that the new incorporated changes should not have side effects on the existing functionalities and thereby finds the other related bugs. Thus regression testing may consume much time, cost and effort. So there is a need to calculate regression testing efforts in Agile.

**Step 5:** Sprint-point based estimation is done by using Proposed Sprint-point Based Estimation Algorithm including Agile estimation factors and regression testing (will be discussed in section 4.6)
Figure 4.1: Sprint-Point Based Estimation Framework
4.3 PRIORITIZATION OF USER-STORIES

In Agile a client always gives the requirements in the form of user-stories. A user story is autonomous, unfixed, precious, estimable, small and testable requirement as discussed in chapter 2. User Stories are great for development teams and product managers as they are easy to understand, discuss and prioritize. The user-stories are more commonly used at sprint-level. For requirements elicitation and prioritization these user-stories must be prioritized. However there is no algorithm in Agile environment for prioritization of user-stories as per importance of the client. This research work suggests a method of prioritization that helps in choosing the optimal order of user stories.

4.3.1 Problems in Existing Prioritization Methods

If the requirements are well identified at early stages of software then the prioritization can be done according to importance of client. But based on the critical study of various research papers, it has been found that the existing techniques of prioritization have various problems due to dynamic nature of Agile. The problem in MOSCOW method was that the managers are worried that their requirements will fall into "should" or "could", and won't get done, so they make up reasons why their requirement is a "must". This ends up delaying business-critical functionality. In MOSCOW method, a lot of time is wasted in discussing things that "should", "could" or "would" happen, delaying progress on the things which are absolutely essential.

By considering the problems and based on the characteristics of ASD methodology a prioritization rule has been proposed to prioritize user-stories in Agile environment.

4.3.2 Proposed Prioritization Rule

As Agile is people-centred, so considering the importance of user-stories for client and effort for each user-story of developers, prioritization rule has been proposed [83].
The proposed prioritization rule is “To prioritize the user stories such that the user-stories with the highest ratio of importance to actual effort will be prioritized first and skipping user stories that are “too big” for current release”.

Consider the ratio of importance as desired by client to actual effort done by project team (I/E) as in Formula 4.1.

\[
\text{Prioritization of user stories} = \frac{\text{Importance of user stories}}{\text{Effort per user stories}} \quad (4.1)
\]

The various steps involved in prioritization of user stories in the Agile environment are as below:

- Gather requirements in the form of user-stories.
- Find out importance and effort related factors as in section 4.3.3.
- Calculate I/E per user story to decide the priority.

### 4.3.3 Proposed Importance and Effort Related Factors

In this work importance related factors such as timely-delivery, dependencies in user-stories etc. and effort related factors such as project domain, technical ability etc. are proposed that impact the prioritization of user-stories.

a) **Timely delivery:** The time constraint is a big issue for client as well as Agile environment. The product release date is given to client according to user story i.e. the client will tell which story is to be done earlier so to start his work as soon as possible. The product which is released early or periodically is best way to satisfy the client needs. If the team is uncertain about the implementation of feature, then early release is the best solution.
b) **Dependencies**: Dependencies of user-stories in the product backlog is always having a vital role in Agile environment. Dependencies between user stories affect the prioritization in Agile environment. Combining several dependent items into large one and splitting the items differently are two common techniques for dealing with dependent user stories.

c) **Business value**: The business value of a user story can be assessed as a combination of user value, revenue, validated knowledge and future return on investment[110].

d) **Risk minimization**: User stories with high risks and high business priority are implemented at early stages of project so that changes in requirement can be detected in early iteration and product can be thoroughly tested in various rounds of testing.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Importance Related Factors</th>
<th>Effort Related Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Timely Delivery</td>
<td>Project domain and ease of coding</td>
</tr>
<tr>
<td>2.</td>
<td>Dependencies</td>
<td>Technical Ability</td>
</tr>
<tr>
<td>3.</td>
<td>Business value</td>
<td>Usability</td>
</tr>
<tr>
<td>4.</td>
<td>Risk Minimization</td>
<td>Complexity</td>
</tr>
<tr>
<td>5.</td>
<td>Cost minimization</td>
<td>Security</td>
</tr>
<tr>
<td>6.</td>
<td>Quality</td>
<td>Pre-requisite availability of resources</td>
</tr>
</tbody>
</table>

e) **Cost minimization**: The cost per user-story is also a big concern for a client. The total cost should be in budget of the client.

f) **Quality delivery**: Project quality may be characterized as functionality, reliability, usability, efficiency, maintainability and portability. The outcome product should develop in such manner that it meets the customer’s requirements. These features of project increase the cost and duration of the project. For
example, Fault tolerance and recoverability are of primary concern for the interfaces but not for the other parts of the system.

g) **Project-domain and ease of coding:** The type of the project affects the cost, effort and duration of the project. The project can be of web based application, construction, new project development, information system, military project etc. Projects can categorize based on unique characteristics of different types of projects. The ease of code depends upon the type of project. Some projects use the template which is available and designed quickly. This can be called as auto generated code. But in some task, the teams have to develop their own code which requires more time and effort.

h) **Technical ability:** The technical ability of a team member is his expertise in some particular activity like any process or tool related with the ASD. It involves specialized knowledge, analytical ability within that specialty, and facility in the use of the tools and techniques of the specific discipline. Technical skills involve process or technique knowledge and proficiency in a certain specialized field such as engineering, computer and accounting.

i) **Usability:** Usability means how is it easy for user to use or operate the product. It may be in the form of a better GUI, to understand the functionality of the product, minimum user input etc. It increases the quality of the software as well as customer satisfaction. System design and architecture include numerous activities which increase the cost of a software project.

j) **Complexity:** The complexity of task in Agile environment depends upon whether the task is composite or stands alone in nature. Composite task are the task which have many dependencies. It has high cross department effect and extensive consulting activity which requires much effort. Standalone or simplex task is the task which has few or no dependencies. It has intermediate total cost tested technology.
k) **Security**: It is considered as network security, functional security, code security, documentation security etc. based on customer requirements. At the time of project development life cycle security must be taken as higher priority factor.

l) **Pre-requisite availability of resources**: Resources are money, people, material, technology, space and other asset which are necessary for effective operation. The pre-requisite availability has vital role in Agile environment because the task get delayed if resource is not available at that time.

### 4.3.4 Proposed User Story Based Prioritization Algorithm

The proposed algorithm explains the various steps involved in prioritizing the backlog in Agile environment as discussed in Figure 4.2.

1. Identify the importance of the user story based on the importance related factor. Suppose I is importance of user stories i.e. I₁, I₂, I₃, ......Iₙ. Importance is reciprocal of rank of user story i.e. the user story with rank 1 has least important value.
2. Identify the effort per user story based on effort related factor. Suppose E is effort of each user story i.e. E₁, E₂, E₃, ......Eₙ. Effort is calculated in hours or days based on size of project.
3. Compute the I/E for all the user stories and takes this value prioritization.
   \[ P_i = \frac{I_i}{E_i} \]
4. Order the data such that \( I_i/E_i > I_{i+1}/E_{i+1} \).
5. Draw a graph between the user stories and I/E user stories and choose user stories from top to down.

Figure 4.2: Proposed User Story Based Prioritization Algorithm

### 4.4 MANAGING UNCERTAINTY IN STORY-POINTS

Due to dynamic nature of Agile, size of user-story is not certain. New user-stories can be added or existing user-stories can be removed at any time. This creates uncertainty in Agile project that leads to poor estimation of time and thus cost. Due to this, all estimates
of project schedule in Agile are subject to uncertainty. This research work focuses on the reasons of existing uncertainty and proposes a new technique to reduce this. Various factors for existence of uncertainty are as below.

- There may be incomplete understanding of scope or incomplete understanding of work per scope.

- Sometimes team has imperfect understanding of known work or the team is not able forecast the unexpected work.

Consider an example of the story-point estimation in Agile by comparing it with a physical artifact like ‘Remodeling a House’. Then the work will be started by breaking down the above project into a few smaller steps as below: Remodel Kitchen, Remodel Bedroom, Remodel Living Room.

Then the total work required for each of these steps is estimated such as “Remodel living room.” Unfortunately, estimate will not be exact because of uncertainty. Suppose an estimate of “Paint living room” is 6 days, with an uncertainty of 2. The best case is 3 days, it is 3 days below the estimate, the most likely case is 6 days and the worst case is 12 days which is 6 days greater than the estimate.

**4.4.1 Proposed Technique of Reducing Uncertainty in Story-points**

Agile developers have to face the problem of release planning because of the dynamic nature of Agile. At any time new user-stories can be added or changes according to the requirements of client. Once the stories are defined, the development team will define the size of story. The size of user story is defined in term of number of days i.e. time needed to complete a user story but this time estimation contains uncertainty [87,88,89]. In this proposed technique of uncertainty management, three types of story-points are defined for reducing uncertainty as below:
**Fastest story-points (FSP):** is the minimum number of story-points required for an activity to be completed. For minimum number of story-points, supposition is made that all predecessor activities are completed as planning is done for them and also all the essential resources whether software or hardware are available when desired.

**Practicable story-points (PSP):** Most of the times, project executives are asked to suggest only one estimate. This is the estimate that goes to the upper management.

**Fatalistic (Maximum) story-points (Max FSP):** The fatalistic is the maximum number of story-points required to complete an activity. In this case, assumption is made that resources are not available when needed. Also the predecessor activities are not completed as planned.

In Agile uncertainty cannot be eliminated completely but when estimating work, some steps can be taken to reduce it. The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. While producing estimate, if number of items are less, then results will be more reliable. The proposed strategy of decreasing the size or “granularity” of items which are to be estimated improves accuracy and reduces uncertainty.

Consider previous example of remodeling a bedroom. It contains a number of steps. If the estimate of “Remodel Bedroom” is taken directly without breaking it into smaller steps like paint room, remove old carpet etc., then estimates will be uncertain but if the various smaller steps are taken into account then uncertainty can be removed to some extent.

The proposed strategy for reducing uncertainty is to break large specifications or work items into smaller pieces. Thus the work of “Remodel Bedroom” is divided into four smaller tasks: paint room, remove old carpet, interior decoration and install new carpet. An estimate for each of the smaller specification or task is produced. When these story-point estimates are added together, then the uncertainty of estimates will be reduced.
In proposed technique to reduce uncertainty in an Agile project in a better way, the stories must be divided into tasks as called as sub-stories by the development team. The task is smallest parts in which a story can be divided into. Next thing is to determine the fastest, practicable and fatalistic story-points by development team. Then the average story-points will be calculated by proposed Formula 4.2. Time of each task is combined together to estimate the size of the user story.

Then the average story-points are calculated by proposal formula

\[
\text{Estimated Story-points} = \text{FSP} + 4 \times \text{PSP} + \text{MaxFSP}
\]

4.4.2 Proposed Rules for Breaking Stories into Sub-stories

If the sub-stories are very small then it will become difficult to analyze the stories, and delay the project completion. There is no use of reducing the size of stories less than a certain level where the relative uncertainty does not improve.

The proposed rule is to pick a granularity that

- Enable an acceptable level of uncertainty.
- Produce a set of specifications or tasks to estimate that are small enough to be sensible and practical.

4.4.3 Proposed Algorithm of Managing Uncertainty

The proposed algorithm as shown in Figure 4.3 describes the various steps involved in managing uncertainty with release planning in Agile environment.
1. Identify user-stories.

2. Then divide the user-stories into sub-stories until they overlap with each other according to the proposed rule of granularity.

3. Then estimate fastest, practicable and fatalistic number of story-points in each sub-story.

4. Calculate estimate number of story-points for each sub-story by using proposed formula

\[
\text{Estimated Story-points} = \frac{FSP + 4 \times PSP + \text{MaxFSP}}{6}
\]

(4.2)

Figure 4.3: Proposed Algorithm for Reducing Uncertainty

- Identify user-stories.

- Then divide the user-stories into sub-stories until they overlap with each other according to the proposed rule of granularity.

- Then estimate fastest, practicable and fatalistic number of story-points in each sub-story.

- Calculate estimate number of story-points for each sub-story by using proposed formula

\[
\text{Estimated Story-points} = \frac{FSP + 4 \times PSP + \text{MaxFSP}}{6}
\]

4.5 PROPOSED SPREEINT-POINT BASED ESTIMATION ALGORITHM USING AGILE ESTIMATION FACTORS

The requirements are taken in the form of user-stories which are grouped in sprints and user-stories based estimation is done using story-points. When a team member estimates that a given task can be completed within 8 hours it does not mean that he can complete the task in 8 hrs. Because no one can sit in one place for the whole day and there can be a number of factors that can effect story-points and hence decrease the velocity.
To resolve this problem the concept of Sprint-point is proposed. A Sprint-point basically calculates the effective story-points. It is an evaluation unit of the user story instead of story-point. By using Sprint points, more accurate estimates can be achieved. The unit of effort is Sprint Point (SP). A Sprint Point is the amount of effort, completed in a unit time. The Sprint-Point Based Estimation Algorithm is as shown in Figure 4.4.

![Sprint-Point Based Estimation Algorithm](image)

**Figure 4.4: Sprint-Point Based Estimation Algorithm**

### 4.5.1 Proposed Agile Estimation Factors

The project and people-related factors can increase or decelerate the velocity of project [80,84]. But the resistance factors always decelerate the velocity and affect on productivity, thus increases the duration of the project. If the duration of the project increases then the costs of the project also get affected. The various Agile estimation factors are shown in Figure 4.5.

93
4.5.1.1 **Project-Related Factors**

These factors are related to project like complexity of project, type of project and quality requirements etc. The various project related factors are shown in Table 4.2.

Table 4.2: Project-Related Factors

<table>
<thead>
<tr>
<th>S.No</th>
<th>Project-Related Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Project-domain</td>
</tr>
<tr>
<td>2.</td>
<td>Quality requirement</td>
</tr>
<tr>
<td>3.</td>
<td>Hardware and software requirements</td>
</tr>
<tr>
<td>4.</td>
<td>Operational ease</td>
</tr>
<tr>
<td>5.</td>
<td>Complexity</td>
</tr>
<tr>
<td>6.</td>
<td>Data transaction</td>
</tr>
<tr>
<td>7.</td>
<td>Multiple sites</td>
</tr>
</tbody>
</table>

**A. Project domain:** The project domain affects the cost, effort and duration of the project. The project can be of web based application, construction, new project development, information system, military project etc. Projects can be categorized based on unique characteristics.
B. Quality requirements: Project quality may be characterized as functionality, usability, effectiveness, maintainability and reliability and portability. The outcome product should develop in such manner that it meets the customer’s requirements. These features of project increase the cost and duration of the project. For example, fault tolerance and recoverability are of primary concern if quality is considered.

C. Hardware and software requirements: Hardware and software requirements are basic need for the development of the project. All projects need certain hardware components or other software resources to be present on a computer. These prerequisites are known as system requirements. With increasing demand of new functionalities in newer versions of software, system requirements tend to enhance over time. System requirements depends on the project, the requirement can vary according to the project. These requirements affect the cost of the project. For example to run applet java virtual machine is necessary.

D. Complexity: It refers to how complex is to develop project. Technical complexity includes a number of aspects such as numbers of technologies are involved, number of technical interface. Management complexity includes project staffing and management etc. Complexity is major aspect in estimation of a project, as the complexity of the project increase Cost, Size and duration of project also increase. Military project are more complex as compared to information based system.

E. Operation ease: Operation ease refers to that how it is easy for a user to use and operate the product. It may be in the form of a better GUI to understand the functionality of the project. It increases the usage of the product which increases the quality and thus customer satisfaction.

F. Data transaction: Data transaction refers to the transfer of the data from one machine to another. Data can access from other machine as per requirement. Data transaction affects the estimation of the project, for example if high data transaction required it necessitate high security. That means high cost of the project.
G. **Multiple sites**: Multiple sites means software or project is developed on one workstation or it is developed on different workstations or sites and integrated later. Big size projects are broken in parts and developed at different sites according to the availability of requirement to develop project. If software runs on multiple sites or many team members work together in distributed environment, cost of the software will increase. Communication delay in distributed environment must be considered in estimation of the duration of project. High communication delay will increase the effort and duration of the project.

4.5.1.2 **People Related Factors**

People related factors are the factors which are people or team oriented. These factors affect the duration of the project and ultimately the cost of the project. People related factors are described in Table 4.3.

<table>
<thead>
<tr>
<th>S.No</th>
<th>People Related Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Communication skills</td>
</tr>
<tr>
<td>2.</td>
<td>Familiarity in team</td>
</tr>
<tr>
<td>3.</td>
<td>Managerial skills</td>
</tr>
<tr>
<td>4.</td>
<td>Security</td>
</tr>
<tr>
<td>5.</td>
<td>Working time</td>
</tr>
<tr>
<td>6.</td>
<td>Past- project experience</td>
</tr>
<tr>
<td>7.</td>
<td>Technical ability</td>
</tr>
</tbody>
</table>

A. **Communication-skills**: Communication is an important part of life. Communication skills are essential in all areas of life. People in organizations usually spend 75 percent of their daily time on other activities like documenting, meetings, listening, e-mail checking and speaking etc. Communication skills also have the importance like technical skills in a Agile team. Communication skills are important in reducing the duration and cost of the
project, since if there is good communication within the team it will take less time to understand each other’s work and worked efficiently and effectively.

**B. Familiarity-in-team:** Familiarity in team affects the team performance, namely team errors, i.e. errors that occur in the interactions of team members. It has been observed that there is a U-shaped relationship between team familiarity and team errors. Initially when familiarity in team member increases it reduces team errors; but they increase if team members become too familiar. Familiarity in team reduces the duration and effort up to a point but if team familiarity increased so much then the reverse impact can be on the performance. So familiarity in team is an important factor that affects the estimation of project.

**C. Managerial-skills:** Management is a tough job. Managerial skill is the ability to communicate with other persons in the department or organizations and the ability to understand their desire and persuade them to work as a team.

**D. Security:** Security may be considered as network security, functional security, code security, documentation security etc. based on customer requirements. At the time of project development life cycle security must be taken as higher priority factor. But it may increase the complexity of the project development and hence resulting into the increase in cost, size, effort and duration of project. For example, online money transaction software projects require various levels of securities to maintain the integrity of software. Banking system and Military projects also requires higher security; hence cost of these projects is high as compared to others.

**E. Working-Time:** Working time is the period of time that an individual spends at paid occupational labour. The working Time is defined as the period during which the worker is doing his work, at the employer's disposal and carrying out his or her duties, in accordance with national laws or practice.
**F. Technical-ability:** The technical skill means implies expertise of a team member in a specific kind of activity like any process or any tool related with the project. It involves knowledge in a specific area, critical ability within that area, and capability of using the tools and techniques of specific discipline. Technical skills involve process or technique knowledge and proficiency in a certain specialized field such as engineering, computer and accounting. It refers to a person’s proficiency in any type of process or technique.

**G. Experience of previous project:** It basically involves specialized knowledge of managers with the previous projects. It refers to a person’s past knowledge and proficiency of previous projects.

4.5.1.3 Resistance Factors

The resistance factors always decelerate the velocity of the project [86]. These factors have long-term affect. In Agile environment some resistance factors are shown in Table 4.4.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Resistance Factors in Agile Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Perfect team composition</td>
</tr>
<tr>
<td>2.</td>
<td>Working place uncomfort</td>
</tr>
<tr>
<td>3.</td>
<td>Drifting to Agile</td>
</tr>
<tr>
<td>4.</td>
<td>Team dynamics</td>
</tr>
<tr>
<td>5.</td>
<td>Expected team changes, other project responsibilities</td>
</tr>
<tr>
<td>6.</td>
<td>Introduction to new technology</td>
</tr>
<tr>
<td>7.</td>
<td>Usability</td>
</tr>
<tr>
<td>8.</td>
<td>Defects in third-party tools</td>
</tr>
<tr>
<td>9.</td>
<td>Stakeholder response</td>
</tr>
<tr>
<td>10.</td>
<td>Lack of clarity in requirements</td>
</tr>
<tr>
<td>11.</td>
<td>Volatility of requirements</td>
</tr>
<tr>
<td>12.</td>
<td>Change in working environment</td>
</tr>
<tr>
<td>13.</td>
<td>Prerequisite availability of resources</td>
</tr>
</tbody>
</table>

Table 4.4:Resistance Factors
A. **Perfect team composition:** The most important feature of Agile teams is that the teams are small and must have the skills like design skills, database skills, testing skills and user interface skills. To compose such a team is a difficult task and it takes lots of time and effort.

B. **Working place uncomfort:** These factors affect the working place. These include interruptions, noise, poor ventilation, poor lighting, uncomfortable seating and desks, inadequate hardware and software etc.

C. **Drifting to Agile:** With the introduction of Agile in an organization it is needed to change the complete process of organization which is again a resistance factor.

D. **Team dynamics:** Team members need to interact frequently with the entire team during meetings, pair programming, or discussions throughout the project. All team members participate in ‘daily stand-up meetings’ using technology-mediated communication. Since Agile primarily encourages open communication and emphasizes effective exchange of thoughts within the team and this takes more time.

E. **Expected team change and outside project responsibilities:** This factor describes the change in team. Some team members may be added and some team members may leave the project and sometimes the responsibilities of the team members may also change. This factor affects the duration as well as the effort of the project. Switching between the projects may be done and due to this the duration of the project is affected.

F. **Introduction of new technology:** With the introduction to new technologies in software industry, the duration of the project is affected. The members are required to learn these technology which takes more time.

G. **Usability:** Usability means how is it easy for user to use or operate the product. It may be in the form of a better GUI, to understand the functionality of the product, minimum
user input etc. It increases the quality of the software as well as customer approval and satisfaction. Better GUI include numerous activities which increase the cost of a software project.

**H. Defects in third party tools:** Various projects require third party tools and software for the implementation as well as design. Some defects may also arise in these tools and software and thus they affect the duration and cost of the project.

**I. Stakeholders response:** The involvement of a stakeholder is required at almost every stage of the development life cycle. But sometimes they do not respond to the requests for information from the developers and sometimes they are not present at the time of meeting. So certain decisions get delayed due to the absence of the stakeholders. Thus directly or indirectly they affect the duration of the project.

**J. Lack of clarity in requirements:** Sometimes lack of clarity in the requirements causes the change in duration as well as cost of the project. Requirements are gathered in the beginning of the project and on that basis the task is performed, but if the requirements are not clear no task can be started.

**K. Volatility of requirements:** Agile promotes volatility, which means that the requirements can be changed at any point of time in the project. Due to change in requirement there may be a situation where more tools are required which affect the cost and duration of the project.

**L. Change in working environment:** Change in the working environment affects the duration of the project as at new place to install proper hardware or software takes more time thus affects cost of the project.

**M. Prerequisite availability of resources:** Resources are money, people, material, technology, space and other asset which are necessary for effective operation. The
availability of resource means that the resource should be available when required. Prerequisite availability has vital role in Agile environment because the task get delay if resource is not available at that time.

4.5.2 Velocity Factor and Complexity Factor

Agile estimation factor decides the velocity factor and complexity factor of the project.

**Velocity Factor:** It refers to the velocity of the project. If the factors are taken at low level then it means that the velocity is not very much affected, but if the level of factors is high then velocity will be affected more. The rating of factors is shown in Table 4.5.

**Complexity Factor:** It refers to how complex is to develop project. Technical complexity includes a number of aspects such as numbers of technologies are involved and number of technical interfaces. To accommodate all characteristics of Agile software development methodology, the complexity relating to each project is rated; if factors are

<table>
<thead>
<tr>
<th>S.No</th>
<th>Level of Factor</th>
<th>Rating</th>
<th>Type of project</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Low</td>
<td>0.94—0.98</td>
<td>Project is simple. For example requirements are very straightforward, no volatility of requirements, All business and technical requirements are very clear to the team with no uncertainty, No research required in the project and it requires basic programming skills to complete.</td>
</tr>
<tr>
<td>2.</td>
<td>Medium</td>
<td>0.90—0.94</td>
<td>Project is Moderately complex. For example it requires little or no research and team has strong expertise in allotted work.</td>
</tr>
<tr>
<td>3.</td>
<td>High</td>
<td>0.85—0.89</td>
<td>Project is extremely complex and demands accurate estimates by consideration of all the factors at a high level. For example the project requires specific expertise or skill set that is important, but missing in the team .The project requires extensive research.</td>
</tr>
</tbody>
</table>
at low level then complexity is less, if the level of factor increases then complexity becomes high. The rating of complexity factor is shown in Table 4.6.

Table 4.6: Rating of Complexity Factor

<table>
<thead>
<tr>
<th>S.No</th>
<th>Level of factor</th>
<th>Rating</th>
<th>Type of Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Low</td>
<td>1</td>
<td>Project is simple. For example requirements are very straightforward no volatility of requirements, No research required in the project and it requires basic programming skills to complete. All business and technical requirements are very clear to the team with no uncertainty. There is no product uncertainty, process uncertainty and resource uncertainty. Team of right ability and experience is available.</td>
</tr>
<tr>
<td>2.</td>
<td>Medium</td>
<td>3</td>
<td>Project is moderately complex. For example it requires little or no research and team has strong expertise in allotted work.</td>
</tr>
<tr>
<td>3.</td>
<td>High</td>
<td>5</td>
<td>Project is extremely complex and demands accurate estimates by consideration of all the factors at a high level. For example the project requires specific expertise or skill set that is important, but missing in the team. The project requires extensive research.</td>
</tr>
</tbody>
</table>

4.5.3 PROPOSED REGRESSION TESTING EFFORTS IN SPRINT-POINT BASED ESTIMATION ALGORITHM

Regression testing is done in Agile to make sure that the new incorporated changes should not have side effects on the existing functionalities and thereby finds the other related bugs. Regression testing is applied to code immediately after changes are made.

The main goal of regression testing is to assure that the changes have no unintended effects on the behavior of the software. These effects may be either in the software being tested, or in another related software component. It is often necessary to ensure software quality. Thus regression testing may consume much time, cost and effort in the project and therefore it is considered as an expensive process. So there is a need for a technique to calculate regression testing efforts in Agile.
The proposed scenario of regression testing in Agile shows that the iterations in Agile are of short duration [82,94]. The very first day of sprint is fixed for sprint planning, decision making and for meetings with the developer and tester. Thereafter the actual work starts, the development takes place for one sprint in first iteration and in second iteration it is sent to the tester for testing work. The tester tests the first sprint meanwhile the developer starts his work on the second sprint, when the second sprint is ready for the testing then the regression testing of first sprint is started. When the third sprint is under development then regression testing of first and second sprint is done. This process continues till all the sprints are developed and are completely tested.

In proposed scenario of regression testing as shown in Figure 4.6 the rework effort and cost is calculated by finding out defects in each sprint as given below.

- **Rework effort** = Total number of defects / Defect fixing effort (DFE)

- **Defect Fixing Effort (DFE)** = Number of defects fixed per hour * No of working hours per day * No of persons working

- **Rework Cost** = Rework effort * Cost of one employee * Number of employees.

- **Defect Density of one sprint** = Defect in that sprint / Total sprint points that iteration is covering.

For regression testing in a particular sprint total number of defects fixed per hour are need to be calculated. On the basis of defect fixing effort rework effort is calculated. This rework effort is called as regression effort of the system which is then used to find out the rework cost of the sprint.
4.5.4 Proposed Sprint-point based Estimation Framework Algorithm

The proposed algorithm explains the various steps involved in estimating a project in Agile environment as shown in Figure 4.7. This algorithm calculates total estimated cost, effort and time of the project by using proposed Agile estimation factors like people-related, project-related and resistance factors. Also in this algorithm regression testing effort is calculated.
• Identify the number of user-stories.
• Find out estimated story-points (ESP) by using three types of story-points.
• Identify the People-related, Project-related and resistance factors which affects the story points in Agile Scrum environment where \( P = \{p_1, p_2, ..., p_{i_1}\} \), where \( 1 < i <= 27 \).
• Identify the level set \( L \) for all factors where \( L = \{1, 2, 3\} \), If \( L=1 \) then level of factors is low, If \( L=2 \) then level of factors is medium, If \( L=3 \) then level of factors is high.
• Assign the Unadjusted value of sprint point (UVSP) corresponding to each level, If \( L=1 \) then \( UVSP = 1 \), If \( L=2 \) then \( UVSP = 3 \), If \( L=3 \) then \( UVSP = 5 \).
• Compute the Sprint Points (SP) as
  ▪ \( SP = ESP + 0.1 \times UVSP \)
• Compute the Velocity from first iteration as
  ▪ \( V = \frac{\text{Sprint point completed in one iteration}}{\text{Sprint point in one user story}} \)
• Assign Velocity factor (VF) depending upon the velocity to perform the task and complexity factor depending upon the complexity of factor (CF) in all the cases.
• Compute the Decelerated Velocity by considering various factors to optimize the velocity
  ▪ \( DV = V \times VF \)
• Compute the Estimated development time required for the Scrum project
  ▪ \( \text{Estimated Development Time (EDT)} = \frac{SP}{Velocity} \) (in Days)
• Compute Total Estimated Effort (TEE)
  ▪ \( TEE = SP + \text{Complexity factor (CF)} \)
• Compute Total Estimated Cost (TEC)
  ▪ \( TEC = TEE \times \text{Cost per person} \)
• Compute Defect Fixing Effort (DFE)
  ▪ \( DFE = \text{Number of defects fixed per hour} \times \text{Number of working hours per day} \times \text{Number of persons working} \)
• Compute Rework Effort (RE)
  ▪ \( RE = \frac{\text{Total number of defects}}{DFE} \)
• Compute Total Estimated Effort using Regression testing (TEERT)
  ▪ \( TEERT = TEE + RE \)
• Compute Rework Cost (RC)
  ▪ \( RC = RE \times \text{cost of one employee} \times \text{number of employees} \)
• Compute Total Estimated Cost using Regression testing (TECRT)
  ▪ \( TECRT = TEC + RC \)

Figure 4.7: Proposed Sprint-Point Based Estimation Framework Algorithm
4.6 CONCLUSION

The purpose of this research work is to develop an algorithm for estimation in scrum which can calculate accurate cost, effort and duration of the project. Due to dynamic nature of Agile there exists uncertainty which cannot be eliminated completely but when estimating work, some steps can be taken to reduce it. The proposed technique reduces uncertainty by reducing the size of the user-story to be estimated. The fewer the elements or specifications that are to be considered while producing an estimate, the more reliable will be the result. The proposed strategy of decreasing the size or granularity of items to be estimated improves accuracy and reduces uncertainty.

In this work people-related, project-related and resistance factors in Agile environment are proposed that impact the estimation of the project. Sprint-point highly depends on the value of these factors. The approach developed is really simple and easy to understand and can be effectively used for estimation in Agile environment.

Using this sprint-point based estimation framework the estimation of small and medium size project can be calculated efficiently. Further as in Agile projects regression testing is very important. So, the estimation is done by using regression testing cost and effort.