Estimation of Project Size Using User Stories

Murad Ali, Zubair A Shaikh , Eaman Ali
{ali.murad, zubair.shaikh, k123421}@nu.edu.pk
National University of Computer & Emerging Sciences, FAST, Karachi, Pakistan.

Abstract—**There exist no concrete models for project size estimation in agile, contrary to traditional projects. In agile, contrasting approaches like standard component estimation, wideband-Delphi or expert consensus and function points technique do not proof much workable for early estimation of size, cost and duration due to uncertain initial requirements. Estimation in agile projects is important regardless of the immature nature of the requirements, on which they work. Next is the translation of effort estimates to project size or vice versa, which is equally important. Technique researched in this paper follows the user stories, which get us most possible accurate estimates. Number of story points with a story matter as well for correct size of story. There are certain specific metrics contained in a story which help in identification of its size and reliability. These factors, in weight age of story point are used to calculate sizing of an agile project.**

Index Terms—**Estimation, Project Size, User Stories, Agile.**

I. INTRODUCTION

Software projects, being one off in nature, contain an element of uncertainty within themselves. This element amplifies with the choice of development life cycle adopted in their production. With the traditional waterfall model, comes a basic assumption of clear and well settled requirements, before the start of the project. This attitude is not realistic as in the real world projects customer has a very less, near to no idea of the product he wants. Usually there is just a single flight of imagination which drives huge enterprise projects from the kick start. This tangibility needs to be translated in product and its development stages, hence the emergence of agile framework. Like any other SDLC projects, success of agile projects depends on its realistic estimates, whether of effort or project size. This estimation fluctuates drastically cause of the varying nature of requirements. This paper focuses on the challenge of discovering such metrics and techniques which determine as close as possible estimate of the project size, hence getting near to real project cost and effort. User stories developing in the initial phases are taken as input here and then are categorized based on their complexity and effort required to implement them. This categorization gives a fairer idea so as to which stories should come in earlier iterations and which can be left for quick sprints. After every iteration this categorization can be revised, based on the experience achieved so far, so as to calculate more accurate estimates of the remaining stories and their efforts.

II. RELATED WORK

Agile methodology refers to an approach which caters for the inevitable factor of unpredictability that comes in play when developing relatively complex software as input of stakeholders is taken at every increment.

Unlike the old and famous 'Waterfall model', which the software industry has been following for several years, agile methodology introduced by Dr. Royce provides an 'inspect and adapt' approach. Requirements are not clear in this approach, and as the project is developed in multiple phases, the team gets a chance to gather requirements on the fly and steer the project in another direction if necessary. Most of the time, teams spend so much time up front in gathering requirements, that they are hesitant to overthrow them as the needs of stakeholders evolve. Agile gives a solution in the form of 'User Stories'. These can be considered as light weight requirements, which consist of conversation (between the user & the team) and confirmation. Conversations can happen at any time during the project lifecycle, which also serves the purpose of creating test confirmations (i.e. the acceptance criteria that can be turned into tests).

Product Backlog: The product backlog is a collection of user stories, themes and epics.

User Stories: The most paramount stories are on the highest point of the item build-up booked to be actualized first in the following few emphases. Those are granular user stories, where every story will convey the unmistakable quality to the item holder.

Themes: The item accumulation stacks are the topics. Topics are accumulation of a few related user stories. Those user stories must be further prepared before they are prepared for execution.

Epics: At the bottom of the product backlog are the epics. Epic as its name suggests is a gathering of a few to over several of user stories that help us to understand the backlog.

Accurate estimate of software size is an essential element to calculate project cost and schedule. In general, size estimates are based on KSLOC or SLOC, or as function points. But this is not how this paper presents the expected estimation technique. The technique explored here is based on user stories, their complexity and time factor associated with them, explained in the later sections. Several approaches have been developed for accurately estimating the project size. This is of paramount importance because it influences cost, budgeting, resources and scheduling. Some of these techniques include: Wideband Delphi, Component, Estimating and Function Points etc.

In Wideband Delphi technique, expert opinion is used to estimate the size of the project, until a consensus has been made. In component estimating, size of previous similar components is used to estimate the size of the current component, hence the software. Function point estimation uses the correlation that greater functionality requires a larger program. This method uses common functions of software to produce a weighted count [1].

The focus of our paper is not on these techniques, as requirements are not established at the beginning. User stories have emerged as an important factor to estimate the size of the project in Agile Software Development. In another work [2], a survey was conducted and questionnaires were sent to experts all over the world. It was observed that the whole process of agile development starting from User Stories to small increments was not followed in the IT industry. Some steps were followed, but only conditionally. Pair programming could not be observed as the costs related to it were unacceptable. In an article "User Stories: An Agile Introduction" [31] an anonymous author states that there are two areas where user stories affect the planning process on agile projects. One of them is 'Estimating' where he states that Developers are responsible for estimating the effort required to implement the things which they will work on, including stories. Although you may fear that developers don't have the requisite estimating skills, and this is often true at first, the fact is that it doesn't take long for people to get pretty good at estimating when they know that they're going to have to live up to those estimates.] Large stories, sometimes called epics, would need to be broken up into smaller stories to meet these criteria. In [3], a new technique was established using user stories for project estimation. A new terminology, Story Point, is introduced. It is a unit to measure the size of a user Story or feature. These points are relative in nature. A story that is assigned 2 points is considered to take twice the effort compared to a story which is assigned a single point. In addition to this, expert opinion may be taken, or estimates of an experienced developer can also be taken into account.

Another term 'Velocity' is used, which is a measure of team's rate of progress per iteration. The estimated velocity may not be 100% accurate. There may be hidden factors that trigger delay in the deployment of the software. In [29], the idea of Story Point is further clarified. Story Point is an abstract measure based on the size of functionality. Some questions should be answered, for e.g. what is involved in the story? , how big is the story relative to other stories that the team has already developed? These story points have to be re-estimated if the team finds significant flaws in its understanding. [4] Explains how a study was conducted aimed at finding out how teams have incorporated agile techniques like *scrum and XP, and have improved upon them.* According to the results, most teams started giving accurate estimates after 3-4 sprints. But the criteria upon which teams were estimating is not explained. Silent grouping is another technique devised by Jean Tabaka [5]. When discussing user stories, and having meetings over them, sometimes developers and managers engage into endless conversations. Silent Grouping eliminates this and helps in processing large amounts of information. There are mainly 4 levels. Preparation, individual placement, group placement, and discussion. In preparation phase, expectation levels are set. In individual placement, each member of the team puts up a user story on the board. Once all user stories are put up, we move on to the next phase. In group placement, each team member silently gives points to each user story. Once all members have given points, user stories are prioritized. A discussion is held to have all the team members on the same page and project is continued. It has also been observed that there can be a tendency to focus too early on the few user stories where there is disagreement, consequently it takes longer to put all the user stories on the board. Though, overall, time is used efficiently, hence this method gives a better cost scenario. Another terminology used in Agile Development is T-Shirt sizing [6]. The concept is easy to grasp. Just like T-Shirts come in different sizes such as small, medium, large and XL, user stories can also be categorized into further sections. Carrying out estimation on these sub categories is more important than estimates of absolute time or effort. The question is, each team will have its own standard of small, medium and large stories, then how to converge to a unanimous solution? For example, team X will consider Medium story to be 50% more complicated than a Small story, but team Y may consider it to be 25% more complicated. If a team manages to standardize this size estimation of a user story over a considerable period of time, then the team has to determine its velocity. I.e. how many large/medium/small stories has a team completed in one iteration? Once it has been calculated, the team can be in a better position to predict that in how much time the other remaining iterations will be delivered.

[28] Says that a user story shortly summarizes the functional requests by customers. It has three characteristics, defined by 3 C's, card, conversation and confirmation. Mike Cohn proposed an I.N.V.E.S.T criteria, which determines whether the user story is Independent, negotiable, has value for business, and is estimable, small and testable. However in professional practice, it was observed that these practices were ignored at large. The USP value cannot be considered as a measure. It is a number only meaningful to the team that have assigned it. From team to team, this User Story Point will vary. Agile teams measure velocity, defined by the number of user story points completed per iteration over time. When teams have historical data, they can use it in estimation. However, not all teams cumulate or have access to historical data and often rely on an estimate of their upcoming velocity to perform their preliminary project estimate. Also, teams assume that team composition, chosen technology and development process will not change which changes nevertheless.

Challenges about sizing Agile projects lies in the purpose of the measurement outcome and in the moments at which measurement should be done to fulfill this purpose, namely:
• Benchmarking;
• Upfront project estimation and budgeting;
• Iteration planning and project re-estimation;
• Process improvement monitoring.
Also, the author has suggested a COSMIC method, which can be accessed from www.cosmicon.com. Here, users are sought to help and improve the method and the dataset.

[26] Emphasizes that user stories should not be confused with 'user requirements'. User stories are short, whilst user requirements are much detailed. User stories help to bridge the gap between the developer and customer. The author has described the alliteration Card, Conversation, and Confirmation, which defines the key elements of a user story. Recommendations for developing good user stories in accordance with the INVEST model and specifically described how small stories increase throughput and quality are provided. A set of patterns for splitting large stories into small stories has also been described, so that each resultant story can independently deliver value in iteration. Also, guidelines for creating spikes as story like backlog items for understanding and managing development risk are given. In [25], a pre-project perspective is taken where Function Points (FP) are difficult, if not impossible to apply since they require a thorough understanding of the requirements and hence typically also initial analysis results such as the system operations that need to be implemented in the upcoming project. This information is usually only discovered within the project and thus Function Points are not well suited where merely a set of rudimentary requirements is available. The author has proposed combining COCOMO II with the relatively novel Use Case Points [Karner 1993] approach that will allow deriving comparatively more precise estimates based on the requirements of a project. Moreover, they have analyzed the required minimum information content of preliminary use case briefs and sketched how Use Case Points can even be applied as a pre-project estimation method. The problem highlighted in [23] is while working with user stories. One major problem, is that if you have like 300 activities to implement and you then are going to discuss them all in detail in the group, it takes a very long time. The developers also noticed some problems when using user stories. User stories have become fuzzier and a bit more general. One problem is how to document and store these user stories. Sometimes, the feeling is that you're doing the same things twice. Story points can increase the efficiency of the team because if the whole team takes part in the estimation all team members will have a better understanding of the tasks that have to be carried out and the amount of unnecessary work done will be reduced.

Literature shows that the more people that take part in the estimation process, the better the estimate will be. Also, the estimates will also be better if done by the same people that will do the implementation later. However, including the entire team in performing estimates may cause increased project costs. This has not been considered in this study. When working with user stories and, to some extent, story points, the discussions are held at a non-technical level. This makes it possible for non-technical people to take part and to contribute.
According to an article, 'Writing a Great User Story' by Agile Expert Ronica Roth [30], a user story represents a small piece of business value that a team can deliver in an iteration. While traditional requirements like use cases try to be as detailed as possible, a user story is defined in 3 stages:

- The brief description of the need
- The conversations that happen during backlog grooming and iteration planning to solidify the details.
- The tests that confirm the story's satisfactory completion.

Some mistakes are also highlighted in this article. These include:

- Too formal or too much detail. Product owners with good intentions often try to write extremely detailed user stories. If a team sees a story at iteration planning that looks like an IEEE requirements document, they often assume that all the details are there and will skip the detailed conversation.

- Technical tasks masquerading as stories. Much of the power of Agile comes from having a working increment of software at the end of each iteration

- Skipping the conversation. Stories are intentionally vague before iteration planning. If you skip the acceptance criteria conversation, you risk moving in the wrong direction, missing edge cases or overlooking customer needs.

[22] Says that story point is a relative measure heavily used for agile estimation of size. The team decides how big a point is, and based on that size, determines how many points each work item is. But this technique demands a degree of consistency across teams for a more streamlined approach to solution delivery. This generates a challenge for CMMI organizations to adopt Agile in software estimation and planning. The proposed process and methodology are applied in a CMMI company level three on different projects. By that, the story point is used on the level of the organization, not the project. Then, the performance of sizing process is measured to show a significant improvement in sizing accuracy after adopting the agile story point in CMMI organizations. To complete the estimation cycle, an improvement in effort estimation dependent on story point is also introduced, and its performance effect is measured. [21] Recognizes that in the case of agile projects, story points are used to measure the effort required to implement a user story. Hence in this paper, total number of story points are used along with project velocity to calculate the effort required for agile software development. In order to achieve better

prediction accuracy, various kernel methods-based support vector regression techniques are introduced.
The performance of the various models generated using SVR kernel methods can be evaluated by using the following evaluation criteria.
* Mean Magnitude of Relative Error (MMRE)
* Prediction Accuracy (PRED)
The results obtained are optimized using four different support vector regression kernel methods. At the end of the study, the results generated are compared in order to access their accuracy. While comparing the results obtained using various SVR kernel methods, it can be concluded that RBF kernel-based support vector regression technique outperformed other three kernel methods. Poor effort estimation is one of the main problems, as highlighted in [18], which impact the success of the software projects. Underestimation results in schedule and budget overruns, on the other hand overestimation can result in inefficiency and waste of resources. Several size measures, including source lines of code (SLOC) and function points, have been defined but SLOC can only be measured at the end of the project. Functional size measurement methods are much more suitable for early size measurement. However, many of these methods are suitable for procedural business information systems and effort estimation based on these measures does not usually consider the software development methodology used. The solution proposed is an effort estimation method for projects employing object oriented software development methodology. Use cases are broadly applied in object oriented software development and use cases are usually key requirements inputs to object oriented analysis and design activities. Therefore, use cases are valuable resources for software size measurement and effort estimation. The authors have considered problem domain models rather than design class diagrams for size and effort estimation. The simplest technique used to measure the size of a program is Source Lines of Code (SLOC), as identified in [17] but SLOC required to develop the same application in two different platforms may not be the same. Also, LOC is dependent on the programming language. A better technique proposed was Function Point Analysis (FPA). Function Points have to be counted manually (drawback is that it is based on human decisions). The counting process cannot be automated. An extension to FPA is the Use Case Points method for sizing and estimating projects developed using object oriented methods. The main drawback of this approach is that use case based estimation method based on UML cannot be done during the early project phase as the use case document is usually prepared after project sign off and requires detailed analysis.

Story Point is a unit to measure the size of a user Story or a feature. A point is assigned to each user Story. These Points are relative in nature, i.e. a Story that is assigned a two point value is assumed to take twice the effort than a Story that is assigned a single point value. A Story Point may be assigned based on the effort involved, the complexity and the inherent risk in developing a feature. An estimate of the effort of developing a user Story requires the developer to have some experience of estimating, to have access to historical data and have the freedom to use a trial based estimation approach.[16] Highlights that user stories are a widespread instrument for representing requirements. Often however, user stories are too coarse, so that misunderstandings or dependencies remain unforeseeable. Granularity of user stories needs to be investigated more, but at the same time is a hard-to-grasp concept. This paper investigates Expected Implementation Duration (EID) of a user story as a characteristic of granularity. We want to find out, whether it is suitable as a quality aspect and can help software teams improve their user stories. Here, a hint of use case point estimation technique is adopted for estimating a user story. It will be like combining two different concepts, one from the scenario where requirements are clear, so that the use cases, or to be specific key elements used in them are derived, and other from uncertain requirement scenario, i.e. story points.

III. RESEARCH FRAMEWORK

The idea is to utilize user stories in deriving a model where related metrics are identified. For this interviewing technique is used to evaluate weight-age of user stories in agile life cycle. It was observed that story development is just an initial step; the major time a development team spends is on investing in group discussions within teams, where they discuss the high level implementation details as well.

FLOW DIAGRAM OF THE RESEARCH FRAMEWORK

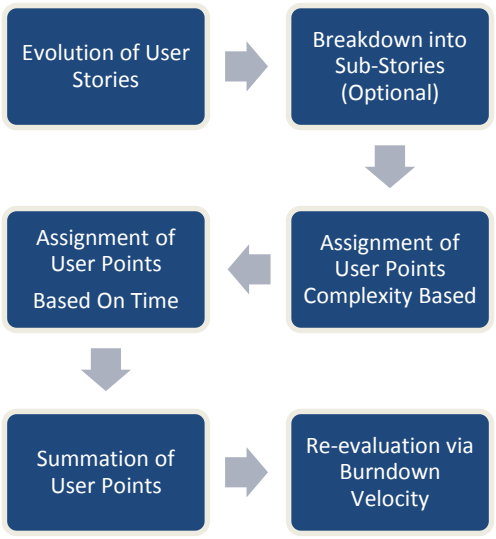

Figure 1: Research Framework

The projects are divided in to user stories. These are user driven so they are short and generic in nature. The success an implementation is gauged in reference to its user story. Then the estimation team may optionally breakdown the user stories into sub-stories, if needed, which would basically be the high level activities a development person perform to implement it completely. This is optional stage as in some cases user stories are self-explanatory and simple to understand, like providing the functionality of

login feature is a well understood task already. But breakdown of activities becomes important for newer, not yet fashioned features.

After this stage, each story is supposed to have a limit on its sizing i.e. the number of user points in it. Now the interesting part is calculation of these user points. Two parameters are explored here, complexity and man hours. Computation of both of these parameters may be facilitated with breakdown stage. The activities estimated will give a fair idea as to how much user interfacing the story will provide, how much database transactions it may involve, how much class implementations it may exercise and most importantly the success criteria of the user story. These are scaled on a range for assignment as below:

Classification	Criteria	Scale
Easy	Simple UI Single database entity hit <= 3 steps of success scenario <= 5 classes are involved Single user type involved	1
Moderate	More UI >= 2 database entity hit 4 to 7 steps of success scenario 5 to 10 classes are involved <= 3 user types involved	2
Complex	Complex UI processing >= 3 database entity hit 7+ steps of success scenario 10+ classes are involved > 3 user types involved	3

Table 1: Classification of User Story

Then the velocity of a resource for each activity is calculated. This is limited on the factors like expertise and experience of resources assigned on the story. This is then summed up, adding buffers as well, providing the man days a project need. The time value gained here is the time based user point associated with the story. This multiplied with the other complexity based user point calculated above will give the number, which ultimately is the size of user story. This activity is to be done at the beginning of the project, the time when you don't have all the details, and at the end of the project, where you exactly know the number of user points that are implemented. The comparison of these would give a reasonable idea how correct and concrete the user story technique may be considered.

The idea is to measure the size of a project in an agile environment using user stories. A project can be considered as a huge story and stories are complex. So determining complexity of a story will eventually lead in identification of variables associated with it, henceforth affecting its size. The size may be translated in to units of time, efforts (man hours), cost, human resource etc.

Complexity of a story may depend on various factors associated with it. One could be the user types. If multiple types of users are involved in a story, then its complexity is expected to increase. A simple story would involve a single type of user.

Another component of a story's complexity could be the action steps involved in it. Action steps could be understood as the steps required for the successful completion of a story. Having greater number of steps entails involvement of more transactions and hence more complexity in the story.

Having said this, we may deduce the below relations:

$$PS \propto US_A$$

$$PS \propto US_B$$

Where,

PS refers to the size of the project (i.e. its complexity),
US_A is the number of user types associated with a story
US_B is the number of action steps involved in a story

In gaming segment, it has been observed that the games that have stories associated with them are less popular in comparison to the games which do not have any known story plots as their base. One example could be of the famous Tom & Jerry cartoon series, which do not have any known games based on it, but the cartoon is very famous on its own. Another example could be of Angry Birds game which is very popular as a game but it is not based on any famous story or cartoon series. This strengthens the idea that when stories are user driven and not repetitive, successful development is a challenge. This is the case when you have scenarios where standards operating procedures (SOPs) are not known or available. By availability of SOP, it is meant the scenarios with repetitive work, like applications of banking systems, ATMs, as evident from the survey, mostly agile work is being carried out in repetitious manner where experience gained in development of one module or sprint is utilized in the next upcoming sprints. People provide their estimates based on their experience and in comparison to their early development work. This cannot be applied to the stories which are user driven and hence have no relevant experience applicable to them. Example scenarios are

taken, and categorization of each with respect to the relations and relevant parameters explored.
Major Story Scenarios are:

1. User Types
2. Action Steps
3. Technology
4. Class Implementations Involved
5. Database Entity Hits
6. User Interface Complexity
7. Time Cost
8. Human Resources
9. Code Size

A month long 'Hospital' project is taken here to test the discussed concept. The project involves CRM implementation of MS dynamics in a hospital where paper less environment is desired and everything is to be kept online for example patients will just bring a card and showing that all their past record, reports, receipts & appointments are retrieved & used from the system.

The input for estimation is the request for proposal document. Variation comes in resource planning from client because at times technical consultants from client side are not available and in-house team has to study the internal systems at client side. For UAT machine is prepared and applied at client side in production and on approval is sent for Go Live. The in-house PM co-ordinates with client side PM as functional consultant & technical consultants are the main source of information for the estimations.

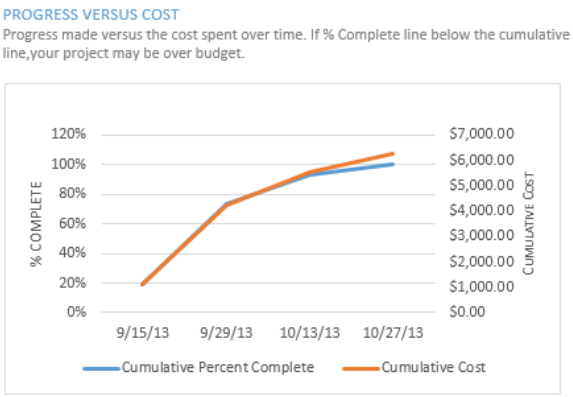

Figure 2

The project size was estimated using past experience in

User Types	Action Steps	Class Implementations Involved	Database Entity Hits	Time Cost	H R
3	5	4	3	28 hrs.	2

original model and an effort of 254 man hours was planned. In actual, the project took 322 man hours, where interestingly most variations were among activities planned against 'DEV2' resources.

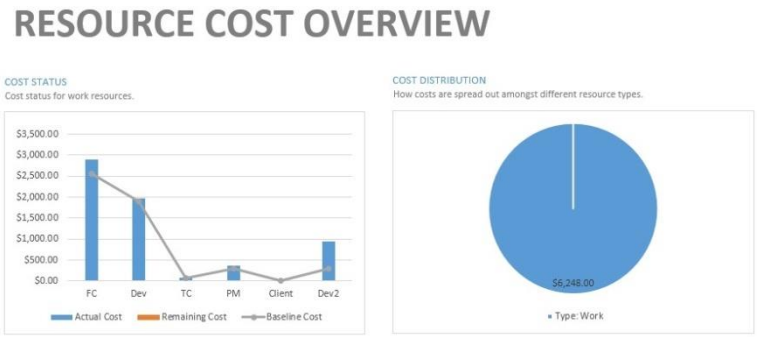

Figure 3

IV. Result

Applying our proposed technique, a story associated with 'DEV2' work is re-estimated as below: The story, as highlighted in below snapshot, involves a scenario where already in placed CRM solution needs to be integrated with IVR new customer lead. Discussion with analysis team identified that 3 user types would be associated with the story i.e. doctors, hospital data entry management team & hospital administrative staff. The success criteria involves 5 steps from login to logout, with 4 class implementations.

Breakdown of story in use case points identified following granular activities:

Integration with IVR
Integrate IVR new customer/lead with CRM
Data input from client (5 hrs.)
Data verification by FC & TC (10 hrs.)
Creation of new customer/lead module unit (10 hrs.)
Testing the module integration (3 hrs.)

This is a scale 2, moderate story per our classification as discussed in table 1.

Moderate	More UI >= 2 database entity hit 4 to 7 steps of success scenario 5 to 10 classes are involved <= 3 user types involved	2

Table 2

Scale 2 story details, using the Dev2 costs as identified in the project, $18/hr.

Table 3

The baseline estimated using experience was 15 hrs. work. Breaking down in use case points the activity estimated as 28 hrs. work which is much closer to the actual 30 hrs. work.

V. Conclusion

Future work involves identification of patterns in the case scenarios under study and then testing the effect of complexity for them. This comparison of the patterns where standard operating procedures are not known will help to analyze the effect of identified factors, especially variation

of user types & action steps of a user story on a project's size. It will involve the modus operandi of picking up an agile project, irrespective of the framework of development it follows (which could be scrum, extreme programming etc., as discussed in the above sections) and then following it through all its iterations. Doing this for all iterations in a project will provide a relatively larger data set and hopefully a bunch of scenario patterns. Collectively, data will be analyzed once before the beginning of the project and then towards its end. There could be many user stories that are not implemented towards project end, and many such stories would have been implemented which were not present at the project start, but rose up in the course of project.

The before and after activity will deduce the preciseness and accuracy of this adaptation for the closest possible estimation, as we think of the traditional estimation models in classical software development life cycle.

ACKNOWLEDGMENT

We would like to thanks to Mr. Huzafah Noor from e-BizSoft for providing information test project data. We are also thankful to Dr. Zubair A Shaikh and his research center CRUC as they also facilitated this research.

REFERENCES

[1] "Estimating Software Size", software engineering online lectures, The University of Edinburgh, last modified autumn 2004
[2] Andreas Schmietendorf, Martin Kunz, Reiner Dumke, "Effort estimation for Agile Software Development Projects"
[3] Evita Coelho and Anirban Basu, "Effort Estimation in Agile Software Development using Story Points", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA, Volume 3– No.7, August 2012.
[4] V. Mahnic, "A Case Study on Agile Estimating and Planning using Scrum" (University of Ljubljana, Faculty of Computer and Information Science, Electronics and electrical engineering, ISSN 1392 – 1215 2011. No. 5(111)).
[5] Ken Power, May 22, 2011, "Using Silent Grouping to Size User Stories", System Agility Blog,
[6] Mike Cohn, Apr22, 2013, "Estimating with Tee Shirt Sizes", Succeeding with Agile - Mike Cohn's Blog,
[7] Popli, R. & Chauhan, N., "Agile estimation using people and project related factors"
[8] Siobhan Keaveney & Kieran Conboy, "Cost Estimation in Agile Development Projects".
[9] Muhammad Usman, Emilia Mendes, Francila Weidt & Ricardo Britto, "Effort estimation in agile software development: a systematic literature review"
[10] S. Bhalerao & Maya Ingle, "Incorporating Vital Factors in Agile Estimation through Algorithmic Method".
[11] Abrahamsson P., Fronza, I. Moser, R., Vlasenko, J. & Pedrycz W., "Predicting Development Effort from User Stories"
[12] Eduardo Miranda & Pierre Bourque, "Sizing User Stories Using Paired Comparisons".
[13] Ziauddin, Shahid Kamal Tipu & Shahrukh Zia, "An Effort Estimation Model for Agile Software Development"
[14] C. Sathish Kumar, A. Anitha Kumari & R. Srinivasa Perumal, "An Optimized Agile Estimation Plan Using Harmony Search Algorithm", Vol 6
[15] Jean-Marc Desharnais, Luigi Buglione & Bura Kocatürk, "Using the COSMIC method to estimate Agile user stories"
[16] Olga Liskin, Raphael Pham, Stephan Kiesling & Kurt Schneider, Why We Need a Granularity Concept for User Stories (Book Title» Agile Processes in Software Engineering and Extreme Programming), Volume 179, 2014
[17] Evita Coelho, Anirban Basu,"Effort Estimation in Agile Software Development using Story Points"
[18] Tülin Erçelebi, Altan Koçyiit,"An Early Software Effort Estimation Method Based on Use Cases and Conceptual Classes", VOL. 9, NO. 8, AUGUST 2014
[19] Richard Dick Carlson, Story Point Estimating
[20] Marcelo Schenone, "Using Story Points to Estimate Software Development Projects in the Commercial Phase"
[21] Shashank Mouli Satapathy, Aditi Panda & Santanu Kumar Rath, "Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods"
[22] El Deen Hamouda, "Using Agile Story Points as an Estimation Technique in CMMI Organizations"
[23] Anna Georgsson,"Introducing Story Points and User Stories to Perform Estimations in a Software Development Organisation"
[24] Carl Friedrich Kreß, Oliver Hummel, Mahmudul Huq, "A Practical Approach for Reliable Pre-Project Effort Estimation"
[25] Joe Schofield, Alan W. Armentrout & Regina M. Trujillo,"Function Points, Use Case Points, Story Points: Observations From a Case Study"
[26] Dean Leffingwell, Pete Behrens, A User Story Primer
[27] Viljan Mahni & Toma Hovelj, On using planning poker for estimating user stories
[28] Sylvie Trudel & Luigi Buglione, Guideline for Sizing Agile Projects with COSMIC
[29] Henry Selvaraj, Dawid Zydek, Grzegorz Chmaj. Progess in Systems Engineering.
[30] Ronica Roth, Write a Great User Story
[31] User Stories: An agile introduction. Anonymous.