A Hybrid Effort Estimation Technique for Agile Software Development (HEETAD)

N.A.Bhaskaran, V.Jayaraj

Abstract: ***Software development becomes a complex process when the software grows in size or complexity making it difficult to estimate usage of resources or development costs. Software effort estimation is that part of development which helps in assessing resource prior to development. An estimate is a quantified evaluation of the effort necessary to carry out a given development task and most often expressed in terms of durations. Effort estimation is done with an intent to aggregate individual estimates and obtain the overall duration, effort or cost of a software project. The workforce is measured as effort and the total time required is defined for a task in effort estimations which is usually expressed in units (Man-day, Man-month, and Man-year). Most other factors like cost or total time required to developed software are dependent on these estimations. Further, Algorithms used for estimating software developments efforts, may also be imprecise. Thus, Effort estimations plays an important part of software development in planning and monitoring projects. Agile methodology is relatively a new set of practices in software development. Agile estimations are based on many factors. Improperly recorded information from Agile methods can result in erratic estimations thus creating an impending need for precise effort estimations. It is difficult to find a single technique which can suit all conditions. Hence, this paper attempts to estimate agile development efforts by using a hybrid technique based on function points and user stories. Results of the proposed technique demonstrate that the arrived effort estimations based on user stories are efficient.***

Keywords: ***Effort Estimation, Function Point Analysis (FPA), Story Points(SP), Agile Development Process(ASD., use case points***

I. INTRODUCTION

Software metrics are used to measure process and product characteristics in software development [1], where effort estimation plays an important role. Software estimation helps in calculating software size, development efforts and schedules for any software project approximately. Effort estimation is a critical activity in the planning of software projects and focuses on time. Many survey studies have shown that nearly one-third of software projects overrun their budgets, while the remaining overshoot original estimates. Accurately determination of efforts for a project is critical to stakeholders also. Project estimation is one of the most important steps in project management irrespective of the project size. Estimations gain their importance as software development schedules are directly dependent on them. Many further factors like development time and cost are based on effort estimations. Good project estimations can make project execution easier as a project's success or failure depends on proper estimations.

Revised Manuscript Received on October 15, 2019

N.A.Bhaskaran, Research Scholar, School of Computer Science Engineering & Applications, Bharathidasan University, Tiruchirappalli, India, nabhaskaran@gmail.com

Dr. V.Jayaraj, Professor, School of Computer Science Engineering & Applications, Bharathidasan University Tiruchirappalli, India, jaya_v2000@yahoo.com

Project have an objective with defined requirements, but may also have a huge set of unclear requirements.
hus, uncertainties on project requirements at the beginning of the project can make their estimation a difficult and challenging task. Amongst estimation methods, Agile methodology is a relatively new concept and is used in software development to fulfill the primary goal of customer satisfaction. Companies using agile methods are inclined to be more flexible [2]. Further, Agile practices emphasize on proper communications between teams for improving project outputs [3], faster delivery and adaptability to customer changes [4]. Figure 1 depicts Agile processes.

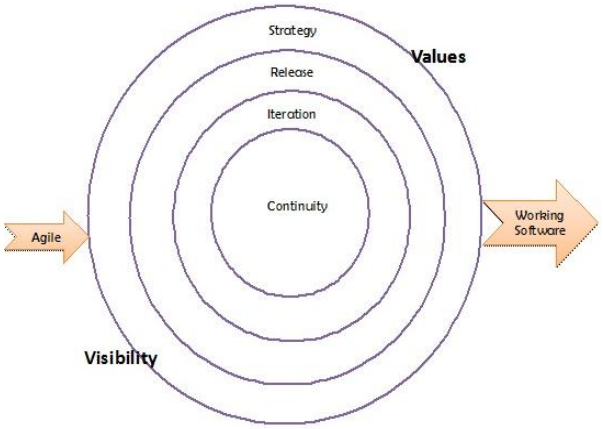

Fig. 1 – Conceptual Agile Development

Agile Software Development (ASD) processes promote modular and stage-wise software development. Complex projects in Agile are managed by breaking the overall process into small activities. Agile iterative processes focus on completed activities. Agile processes are effective and maneuverable as they result in minimal documentations [5]. One main advantage of the agile process is its convergence to reality. They are being adopted in software projects for time bound deliveries and where traditional approaches overshoot delivery times [6]. One main challenge in project estimation stems from uncertainties involved at the beginning of the project. When the client is not clear on the complete requirements, it is not possible to estimate the project correctly in term of efforts and time. Studies and development teams have been attempting to simplify effort estimation methods [7]. One main pitfall has been the reduced applicability of proposed estimations methods, where technological growth has induced indirections in effort estimations. Thus, there is scope for newer models of software effort estimation. Moreover, estimations based on tasks identified by non-developers may lead to mismatches in the estimation process. This paper attempts to overcome general challenges in agile software effort estimation by proposing a hybrid technique called Hybrid Effort Estimation Technique for Agile Software Development (HEETAD). Estimations are based on task estimations and

the basic unit of story points in Agile tasks is used by HEETAD for calculating absolute work. HEETAD converts story points into function points and can be applied to different phases of the project like design, sprint and project ends. HEETAD estimations are based on function Points from User Stories, thus helping estimations in early project phases. Thus, this paper attempts to reduce complexity of Agile estimations by fusing agile and traditional software development methods for arriving at a better estimation technique in agile software development. The paper is organized as follows. Introduction is followed by the problem definition, related studies, a section describing user stories, the proposed HEETAD techniques, a section of use case points in agile development which is followed by results and conclusion. The results show a comparative evaluation of HEETAD against use case points.

HEETAD provides a decently accurate estimation of agile development by automating its calculations from User Stories. HEETAD refers to internationally accepted development times of 2017. Irrespective of the complexity of a story point, HEETAD converts them into functional points which can be used for estimating agile project efforts for different programming languages.

II. DEFINITION OF THE PROBLEM

Agile methods help in flexible planning of software development, but Agile estimations have their own set of issues. Estimations are typically done by development teams. The main issue starts with estimations when accurate story point estimations are expected from fairly new Agile teams. In an Agile estimation process, stories are selected for a Sprint and estimated by taking into consideration complexity and efforts. Teams also estimate stories immediately after a discussion in clearing backlogs. These estimates may not be exact and depend on the estimation experience of a team. Teams falter when they are not experienced or coached in Agile Methodologies. Another general issue found in Agile estimations is that only a few members participate actively while others become observers and agree to the given estimates. Even the observers who do not accept changes or estimations, proceed with unresolved queries creating an uncomfortable position as either backlogs or more development time. In software development, pre-defined function points can estimate efforts much better, but are seldom used in Agile estimations.

III. ASD EFFORT ESTIMATION

Agile effort estimation methods vary widely and are adapted based on project conveniences. Agile estimation methods on a project differ completely from conventional estimation methods which are based on manager estimates. Agile estimation methods are namely story points, ordering protocol, planning poker and T-shirt sizing. In ASD, work is assigned to a team instead of a member, thus placing emphasis on collective effort. ASD does not prescribe a definite way to estimate work in terms of development time and use an abstracted metric called scales. Any agile team's scale is shared by its team members. For example Scrum development in ASD, uses effort and degree of difficulty to estimate team work. Scrum's abstract scheduling does not reflect on the true complexity of projects. Stories are formulated in one or two sentences from customer's own language and written on an index card [8]. Effort estimation takes place at the beginning of any new iteration of a planned release or Sprint. Agile sprint used in Scrum methodology is an incremental piece of work with a fixed length of days (30 days), where Story point is the sizing unit of effort estimations. Story points express the overall size and associated features in a user story. Estimates based on story points are given more importance as they form the basis for upcoming releases. Smallest story is selected based on a team's opinion and team effort is estimated. Another approach may select a medium size story for estimating team work [9][10]. Thus, the overall time required for developing each of the stories is estimated by developers. The customers then define the direction of the project by prioritizing stories based on their business value and initial developer estimates [11]. Figure 2 depicts a generic flow of an ASD Estimation.

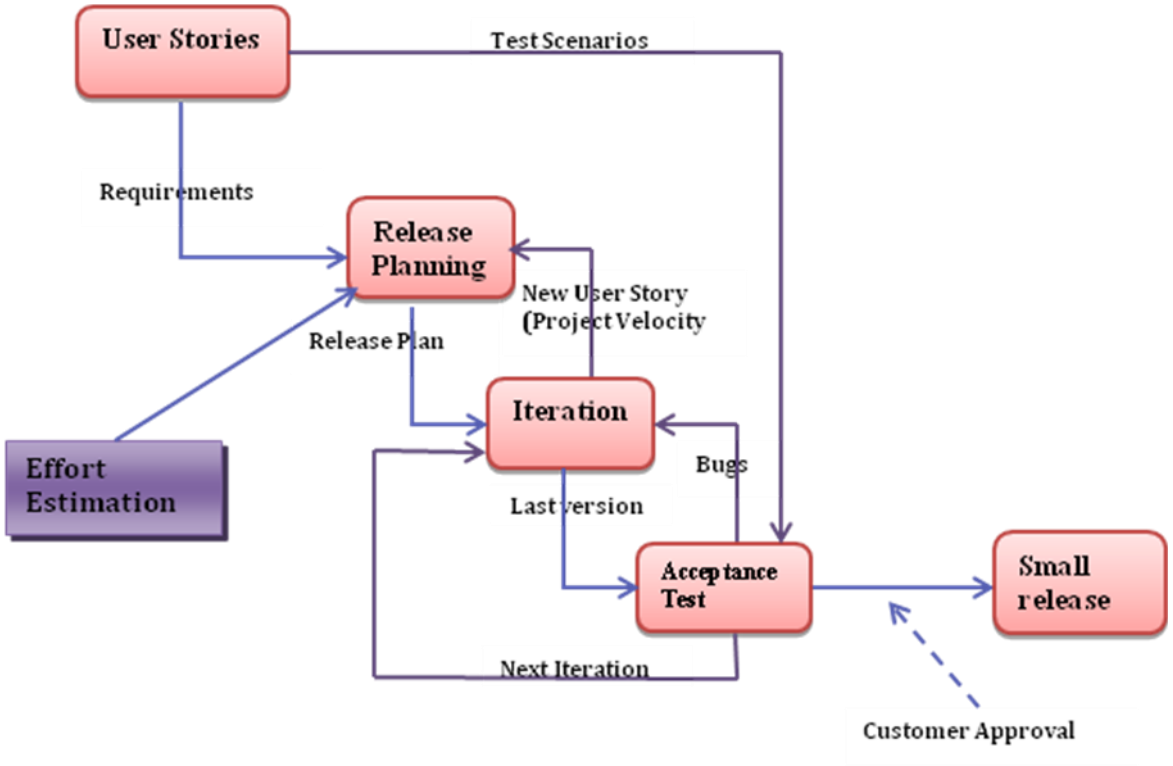

Fig. 2 – ASD Effort Estimation

IV. RELATED STUDIES

Many studies related to this work are presented in this section. User story points in Agile estimations and function points are two disparate areas in effort estimations and normally do not converge. Many studies have investigated the use of story points in agile projects [12]. The study in [13] presented a research of software estimation methods. It evaluated papers published between 1996 and 2006, but did not present specific findings on the use of agile methods. Evidences that effort estimation is a critical task in agile software for project planning was presented in [14], [15]. Though these studies focused on Agile estimation methods, they did not explore the levels of accuracy of the approaches and how cost drivers could be used in estimations. Computing techniques to solve effort estimation in Agile projects was studied in [16], where techniques like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Neural Network (ANN), and Fuzzy Inference Systems (FIS) were focused. Neither evidence of benefits of these techniques was presented nor were the methods validated. [17], but the study was a guide to effort estimates in Agile methodology. A study on XP estimation concluded that scrum and XP were similar, but differed in their estimated functionalities [18]. Effort estimation within XP-projects was identified by a study as a part of project plans [19].

V. USER STORIES

User Stories in Agile development are important customer conversations which are the starting points of a new project. It is used both by development teams and managers. Customer conversations get converted into requirements (Stories). Each statement that is easy to comprehend, discuss and prioritize is represented as a user story. Agile sprints (Plan for completion of unfinished work) use these user stories. They are also a part of iteration planning, where programming tasks are listed and scheduled. Each user story is independent without dependencies on other stories or requirements. In agile developments, large stories or epics are disintegrated into smaller user stories for ease of management. Many user stories are also joined together to create an Epic in the development roadmaps [20]. Bigger User stories are difficult to prioritize or plan as they create complexity in effort estimations. On the other hand, smaller stories help in accurate estimates. Story point is the evaluation unit of a user story. Story Points have their origin in Xtreme Programming (XP) and measure complexity/ size of stories [21]. In general, estimating user stories is based on story points that are the basic unit of estimation. Estimation based on story points can help narrow down efforts to days or hours. If a user story size's effort cannot be judged, it may left out of sprint planning, thus leaving estimations incomplete. Teams judging efforts arrive at their estimates by comparing them previous task history estimates and in terms of story points [22]. Agile method estimations use **"the power of two"** for estimating work, where the simplest user story (least effort) to implement in a work backlog is assigned a value of 1. Other stories are given values of 2, 4, 8, 16, 32 etc. in comparison to simple ones. User stories with values of 32 or more are split into smaller stories in planning. User stories are depicted in Figure 3.

- Students can purchase monthly parking passes online
- Parking passes can be paid by credit cards, Online
- Teaching Faculty can input their student internal marks
- Students can obtain their current seminar schedules
- Students can only enroll in seminars for which they have perquisites
- Students can order for official transcripts
- Transcripts will be available online

Fig. 3 – User Story Examples

VI. HYBRID EFFORT ESTIMATION TECHNIQUE FOR AGILE SOFTWARE DEVELOPMENT (HEETAD)

Each user story has to be estimated independently in Agile Estimations as dependencies between stories can make planning, prioritization, and estimation of work a complex task. Agile developers are used in agile estimations. Though at the outset, it may look like developer estimates can be inaccurate, they actually estimate work precisely. Main hindrances in developer estimations stem from lack of domain knowledge or in estimates of epics. Moreover, estimating a user story's difficulty requires repeated rounds of estimation when Agile projects begin their iterations with incomplete requirements. Agile methodology can thus have unpredictability in its Sprint estimates [23]. Thus, a framework which converts story points to function points can help reduce problems in agile estimations and improve it from abstractness to absoluteness. The proposed technique called Hybrid Effort Estimation Technique for Agile Software Development (HEETAD) attempts to overcome such shortages in agile estimations. It is based on the simple and standardized method of functional size. The proposed technique uses function points for dependability. Figure 4 depicts the architecture of HEETAD.

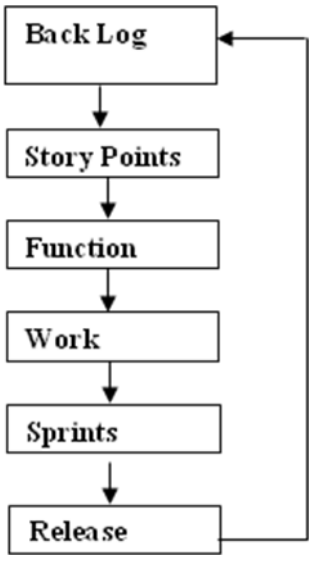

Fig. 4 – HEETAD Architecture

The Sprints required to complete the project can be estimated by linking velocity and the number of story points to be developed from Product Backlogs. Moreover, during the design phase of agile development, it is difficult to estimate project efforts based on project functionalities in product backlogs. Story point measures are also relative and cannot be used to estimate the effort or cost of a project in agile development. Difficulties in functional size estimation are not exclusive to agile development and have been present in traditional methods like Waterfall methods which experience problems in functional size estimation. Functional size calculations are based on elementary processes and target smaller software functionalities. HEETAD first finds the relationship between functional size and story points in estimation. It estimates and measures the size and efforts of agile projects based on the principle that the functional size of software is directly proportional to the number of data movements. Thus, HEETAD delivers the final effort required in terms of development time for ach sprint.

VII. IMPLEMENTATION

- Different attributes of the library system under the function point and use case point methods were tested. The main attributes selected for analysis are Effort, Lines of Code, Languages and Time for development.
- **Effort**: Effort is the software development effort calculated from function point as well as use case point which is expressed in Person Hours (PH).
- **Lines of Code**: It is the number of lines in the source code of the particular software. We have taken the programming language as Java for developing software.
- **Languages**: Development time with repect to different languages.
- **Time:** It is the time taken to develop the project which is expressed in Hours.

Story points generated from users are a powerful sizing technique with many advantages. Agile Scrum planning is usually held in two sessions, where session one has user stories and team estimates in sizes for user stories. In the next session the stories are broken into tasks or story points. The team capacity is matched against its goal with estimates in terms of hours. Story points estimates can also be based on complexity. HEETAD was applied to a part of a new Library system for elucidating function points from story points. This implies all functions are new to the application and function point analysis can be used to measure the size of the project. The application requirements define addition of the following functions namely Maintenance transactions to add, change, and delete data in the Library file and a library report. HEETAD story point estimates are based on the complexity of a story point. While estimating story points, HEETAD assigns relative values since they are more important than the raw values. Effort Time Based on Complexity used by HEETAD is given in Equation (1).

$$EFT(C) = SP_{Count} * K \quad (1)$$

where SP_{Count} = Story Point Complexity and K is a constant of 6 Hours. HEETAD sizing is using Table 1 in its estimations.

S.No	Story Point Complexity	Story Point Count	Relative Values in Hours
1	Elementary	1	6 Hours
2	Medium	2	12 Hours
3	Complex	3	18 Hours

Table I – HEETAD Story Point Classification Table

User Stories in the Library

I may need a book or journal
I will search for a list of book
I will need to store a new list of book
I will have to take a report of books, journals and magazines

VIII. HEETAD SP-FP ESTIMATION

Story Points can be implemented faster than Function Points as it is a collaborative approach involving project teams for comprehending each user story. On the other hand, Function Points measure software by quantifying the features mainly based on a logical conception of needs. Function Points are a standard replicable measuring unit irrespective of who measures them. Both the methods aim at efficiently managing a software development project. HEETAD combines the aforesaid methods for effective estimation of agile projects. INITIAL EFFORT is calculated using Equation (2)

$$E_i = a * SIZE\ b \qquad (2)$$

Where a and b are constants depending on the project type.

Story points are used as a base by HEETAD for efficiently managing the workflow of an agile project. Table 2 lists the sizing based on the selected example's user stories, while Table 3 lists the function point complexity of the project.

S.No	User Story Tasks	Complexity	SP_{Count}	EFT(C)
1	Library Information File Creation	Complex	3	18
2	Library Information Manipulation	Complex	3	18
3	Information Search	Complex	3	18
4	Library Reports	Complex	3	18
			Total Hours	**72 Hrs**

Table II – HEETAD Agile Sizing Table

S.No	Function Description	Function Point	Complexity	SP_{Count}	EFT(C)
1	Library File Creation	Design	M	2	12
2.	Library Information	Add	M	2	12
3	Library Information	Modify	E	1	6
4	Menu Creations	Design	E	1	6
5	Library Information	Delete	E	1	6
6	Library Information Search	Query	C	2	18
7	Library Information Report	Report	M	2	12

Table III – Library System Example's Function Points Complexity Table

HEETAD technique differs from other methods in calculating effort based on the Lines of Code (LOC) that is universal in character. The estimations of HEETAD are based on Table 4's Language complexity and line of code table [24] [25]. The effective KLOC is calculated using Equation(3)

$$KLOC_{Eff} = \frac{KLOC_{Est}}{SP_{Weight}} \qquad (3)$$

Where $KLOC_{Est}$ = KLOC of the Programming Language

SP_{Weight} = Agile Fibonacci Weight

Complexity	Language	Kilo Lines of Code	Man Hours	Estimated Time / KLOC	Weight	Effective KLOC/Hr
C	Java	53.33	12.697	4.20020477	8	0.5250256
C	C	128	26.273	4.87192174	8	0.60899022
M	Java	58.33	12.697	4.59399858	5	0.91879972
M	C	132	26.273	5.0241693	5	1.00483386
E	Java	62.33	12.697	4.90903363	3	1.63634454
E	C	134	26.273	5.10029308	3	1.70009769

Table IV – Language-Complexity and LOC metrics

HEETAD based on Table 3 and Table arrives at the final output of estimation. The weight assigned to each function namely complexity with weights of (8,5,3) are used along with the average Kilo LOC required for the function point and multiplied by 1.5 to accommodate testing time required for the project. HEETAD uses the weights of story points, EFT(C) to arrive at the Man days Calculation given by Equation (4)

$$\text{Agile man Days } \mathbf{MN} = \frac{ETC(C) \big/ UP_{Weight}}{KLOC_{Eff}} \quad \mathbf{(4)}$$

S.No	Function Description	Function Point	UP_{Weight}	Effective KLOC/Hr In "C" from Table 4	SP_{Count}	EFT(C)	$\frac{EFT(C)}{UP_{Weight}}$	$\frac{ETC(C)/UP_{Weight}}{KLOC_{Eff}}$
1	Library File Creation	Design	8	0.60899	2	12	1.5	2.463095
2	Library Information	Add	8	0.60899	2	12	1.5	2.463095
3	Library Information	Modify	8	0.60899	1	6	.75	1.231547
4	Menu Creations	Design	8	0.60899	1	6	.75	1.231547
5	Library Information	Delete	3	0.60899	1	6	2	3.284126
6	Library Information Search	Query	8	0.60899	2	12	1.5	2.463095
7	Library Information Report	Report	8	0.60899	2	12	1.5	2.463095
** Totals **			51 FPs	15.5996 (Man Days) or 124.7968 Hrs.				

Table V – HEETAD Library System Example - Output

It can be seen Table 5 arrives at a lesser estimation in hours based on HEETAD. Thus, HEETAD's Table 5 can be used for accurate prediction of estimates. Further, the factor of 1.5 taken into account by HEETAD is not considered in Table 2 where only the story point estimates are used.

IX. PROJECTED HEETAD AGILE PROJECT SPRINT ESTIMATION

Agile Projects are estimated by using Quick Function Point Analysis (QFPA) i.e. calculating the functional size of an application. Story points assigned to user stories are used plan releases, number of user stories and sprints. Hours and task estimations at the sprint level are obtained from user stories within a sprint. Thus, tasks can be estimated in man hours for any user story.

Agile projects are also estimated by breaking down business requirements into lower levels called the Bottom-Up Approach. The purposes hours for tasks in Bottom-Up Approach is based on the user stories taken in a sprint. Once estimated it is entirely dependent on the developers and teams to deliver their commitments on time. The tasks need to be small and a maximum of six hours. Agile estimation of the library system is listed in Table 6 where the estimation done using Fibonacci Sequence 1, 2, 3, 5, 8, 13, 21, 34… Table 7 lists the no of sprints and effort in hours based on different team velocities

.

Id	User Story	Estimated Story Points
1	I may need a book	1
2	I will search for a book	21
3	I will need to store a book	34
4	I will have to take a report of books	21
Total Story Points		77

Table VI – Agile SP Estimation

Project Story Points	Team Velocity SP/Sprints	Estimated Sprints	Estimated Weeks (Sprint=2 weeks)	Estimated Hours @ 40 hours a week
77	25	3.1	6.2	248
77	30	2.6	5.2	208
77	35	2.2	4.4	176
77	40	1.9	3.8	152
77	45	1.7	3.4	136
77	50	1.5	3	120
77	55	1.4	2.8	112
77	60	1.3	2.6	104
77	65	1.2	2.4	96
77	70	1.1	2.2	88

Table VIII – Agile estimation in Hours based on different velocities

Table 8 lists the comparison between HEETAD estimation and agile estimations with sprints, while Figure5 depicts the same as a graph.

Project Story Points	Team Velocity SP/Sprints	Estimated Hours @ 40 hours a week	HEETAD Hours
77	25	248	96
77	30	208	96
77	35	176	96
77	40	152	96
77	45	136	96
77	50	120	96
77	55	112	96
77	60	104	96
77	65	96	96
77	70	88	96

Table VIII - HEETAD estimation Vs Agile Estimations

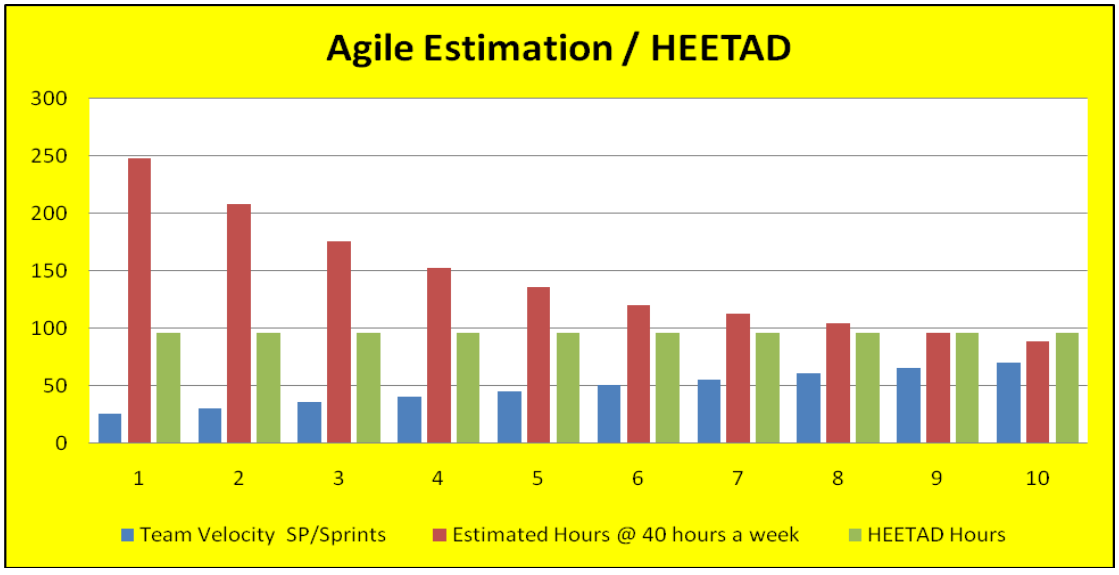

Fig. 5 – Agile Estimation Vs HEETAD

It is evident from Table 9 and Figure 5 that HEETAD performs weel is agile estimation and converting them into hours based on tasks.

X. USE CASE POINTS

Use Case Points was first proposed in 1993 by Gustav Karner[26][27]. It uses a case diagram with descriptions. It is a well-documented approach and reliable approach for estimating software development which is similar to Function Point estimation. It provides the ability to estimate the man- hours from its use cases. This research used Use case points in the same library system. Table 10 lists the UCP weight counts based on Complexity, while Table 11 lists the UCP Matrix for the Library System of the implementation

Use Case Type	No of Transactions	Weight	Use Case Count	Weight Count
Simple	1 – 3	5	3	15
Average	4 – 7	10	2	20
Complex	> 7	15	2	30

Table IX – UCP Weightages

Transaction	Use Case Type	Weight	Use Case Count	Weight Count
Library File Creation	Average	10	2	20
Library Information Add	Average	10	2	20
Library Information Modify	Simple	5	3	15
Menu Creations	Simple	5	3	15
Library Information delete	Simple	5	3	15
Library Information Search	Simple	5	3	15
Library Information Report	Simple	5	3	15
Total Use Case Points				**115**

Table X- Use Case point matrix

The Table below list comparative assessments of function points and hours based on story points and the same is depicted in Figure 6.

S. No	Function	FP		Use Case	
		Size count	Effort Hours	Size Count	Effort Hours
1	Library File Creation	8	2.49593499	20	19
2	Library Information	8	14.97561	20	19
3	Library Information	8	14.97561	15	14.25
4	Menu Creations	8	4.99186998	15	14.25
5	Library Information	3	2.49593499	15	14.25
6	Library Information Search	8	19.9674799	15	14.25
7	Library Information Report	8	19.9674799	15	14.25
			79.8699198		109.25

Table XI – HEETAD- UCP Hours Assessment

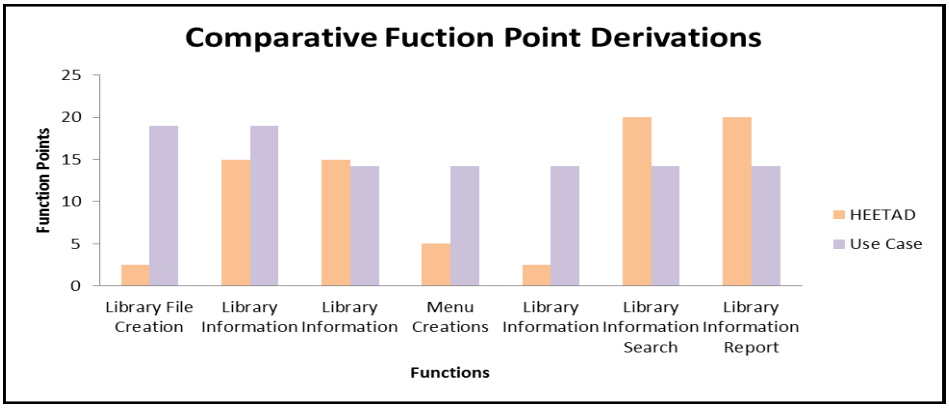

Fig. 6 – Comparative Function Point Derivations of HEETAD and UCP

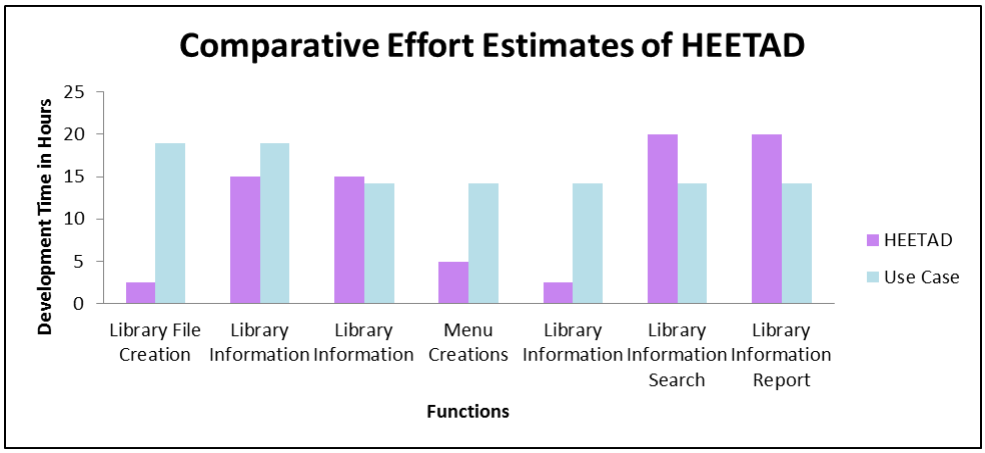

Fig. 7– Comparative Effort Estimates of HEETAD and UCP

A fundamental problem with estimating with use case points is that the estimates happen towards the end and after all use cases in the project are written. About 20% of the use case effort estimations represent user goals, which also result in delaying the release plan. Moreover, Use cases are large planning units in a system and also cannot be used for any further learning even when all use cases are written. Hence, this research work chose the HEETAD estimation for estimating efforts in AGILE project estimations.

XI. CONCLUSION

Software projects need appropriate software estimation methodologies that can help plan their development and maintenance projects. Some organizations implement in-house tools or method for the estimation which need to include well defined software development processes and standard estimation methods. Hence, there must be suitable estimation methods for various types of projects. There were many reasons for project failures in ISO software organizations when compared to CMM level software organizations. Many organizations use model-based methods like COCOMO, Use-case-based estimation, Agile and FPA. For such software organizations with no past completed projects, initially suggestive effort estimation tool is required. Agile development is becoming a reality in many organizations. However, Agile metrics have gaps from the perspective of an organization's needs. One major limitation of agile methods concerns effort estimation and performance monitoring of agile projects which is difficult. An estimate of effort/duration unlike traditional projects is difficult in Agile due to many factors like Technology, domain Knowledge or technical expertise of developers and complexity of work. This paper has proposed and demonstrated an easy and viable technique for estimating work to be done in agile development. Further, since it uses function points and LOC in its estimations, a measured level accuracy can be expected in estimations using this technique. It can be concluded that HEETAD is a viable technique for effort estimation of Agile projects.

REFERENCES

1. Bilgaiyan, S., Mishra, S., Das, M.: A Review of Software Cost Estimation in Agile Software Development Using Soft Computing Techniques. International Conference on Computational Intelligence and Networks (CINE), 112–117, 2016.
2. Highsmith. J, "Agile Project Management: Principles and Tools: Cutter Consortium, 2003.
3. Miller G.G. (2001) The Characteristics of Agile Software Processes. Proceedings of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'01)
4. Schmietendorf A., Kunz M., Dumke R. (2008) Effort estimation for Agile Software Development Projects, Proceedings 5th Software Measurement European Forum, Milan
5. H. Andrat and S. Jaswal. "An alternative approach for Effort and Cost assessment in scrum". In Computing and Network Communications (CoCoNet-2015), 2015 International Conference on, pages 535–539. IEEE, 2015.
6. Capers Jones. Applied software measurement: global analysis of productivity and quality. McGraw-Hill Education Group, 2008.
7. ZIA, Z.; Rashid, A.; UZ Zaman, K. (2011) Software cost estimation for component based fourth-generation-language software applications, IET Software , 5, Page(s): 103-110.
8. Usman, M., Mendes, E., Weidt, F., Britto, R.: Effort estimation in Agile Software Development: A systematic literature review. ACM International Conference Proceeding, 82–91, 2014.

9. Schweighofer, T., Kline, A., Pavlic, L., Hericko, M.: How is Effort Estimated in Agile Software Development Projects? Sqamia, 2016.
10. C. Santana, F. Leoneo, A. Vasconcelos and C. Gusmão, "Using Function Points in Agile Projects". (Eds.): XP 2011, Springer-Verlag Berlin Heidelberg.
11. Lovaasen, G, “Brokering with eXtreme Programming. In XPUniverse 2001. Raleigh, North Carolina.
12. M Cohn,"user stories applied: For Agile Software Development", 2004, AddisonRXesley, Doston.
13. Wake Bill. INVEST in Good Stories, and SMART Tasks. Available on – line: http://xp123.com/ articles/ invest-in-good-stories-and-smart-tasks /. Date of publication: August 17, 2003. Date retrieved: January 18, 2012.
14. Morris Rob: Agile Estimation and Planning. CDL Systems 2006.
15. Lant Michael. Estimate Story Size by Playing Agile Planning Poker. Available on – line: http://michaellant.com/2010/07/13/agile-planning-poker/. Date of publication: July 13, 2010. Date retrieved: March 18, 2012.
16. Manfred Bundschuh, Carol Dekkers, “The IT Measurement Compendium Estimating and Benchmarking Success with Functional Size Measurement, 2008.
17. M Cohn," Agile Estimating and Planning", Addison-Wesley, (2005).] [S. Kang, O. Choi, J.Baik, "Model-based Dynamic Cost Estimation and Tracking Method for Agile Software Development", 9th IEEE/ACIS.2010, IEEE.
18. Schwaber, Ken and Mike Beedle, “Agile Software Development with Scrum”, 2002, Prentice Hall.
19. Fuqua, A., "Using Function Points in XP –Considerations", In: Proceedings of Extreme Programming and Agile Processes in Software Engineering, Springerlink.
20. Sehra, S.K., Brar, Y.S., Kaur, N., Sehra, S.S.: Research patterns and trends in software effort estimation, 2017.
21. Munialo, S.W., Muketha, G.M.: A Review of Agile Software Effort Estimation Methods, 612–618, 2016.
22. Jones, C.: Applied Software Measurement. McGraw Hill, New York (2008).
23. M. Yadav, N. Goyal, and J. Yadav. “Agile methodology over iterative approach of software development a review”. International Conference in Computing for Sustainable Global Development (INDIACom-2015), pages 542–547. IEEE, 2015.
24. Software Economics and Function Point Metrics, Version 10.0 April 14, 2017, 30 Yrs of IFPUG Progress
25. Bhaskaran, N.A, and Jayaraj, V. (2019). Software Effort Estimation using Function Point based Clustering Technique (FPBCT). Indian Journal Of Science And Technology, 11(48). doi:10.17485/ ijst/ 2018/ v11i48/ 138343
26. Karner, G. (1993). Resource Estimation for Objectory Projects. Objective Systems SF AB.
27. Ibarra, G. I., & Vilain, P. (2010). Software Estimation Based on Use Case Size. Brazilian Symposium on Software Engineering, (pp. 178-187)

AUTHORS PROFILE

Mr. N.A. Bhaskaran, Ph.D. Scholar, Bharathidasan University, Department of Computer Science Engineering, Tiruchirapalli. He has completed MCA and M.Tech, and also IBM – DB2 certified professional. He is currently an Associate Professor of CSE at Bharathiyar College of Engineering and Technology, Karaikal. He has 19 year of Teaching experience and 3 years of research experience. He has written a monograph and a Book for MBA Department. He has published more than 10 research articles in different international journals and conference. His main research interests are Empirical Software Engineering, Software Estimation, Agile Development Process, Cloud Computing.

Dr.V.Jayaraj, Professor, Department of Computer Science and Engineering, Bharathidasan University, Tiruchirappalli, Tamilnadu, India. He has published more than 35 International Papers. He has guided 12 Ph.D candidates and 60 M.Phil candidates so far. He is a member various academic and Research Committees in various Universities. He has published five books. He visited many Universities in various Countries. He has organized various National and International level programs.