International Journal of Computer Science and Applications, ©2009 Technomathematics Research Foundation
Vol. 6, No. 1, pp. 85 – 97

INCORPORATING VITAL FACTORS IN AGILE ESTIMATION THROUGH ALGORITHMIC METHOD

S. BHALERAO*

School of computer Science, D.A.V.V., Khandwa Road,
Indore, Madya Pradesh., India
bhalerao.shilpa@gmail.com

MAYA INGLE

School of Computer Science, D.A.V.V., Khandwa Road,
Indore, Madhya Pradesh, India
maya_ingle@rediffmail.com

Agile methods have become the mainstream of software development due to their enriched practices. Some commonly used practices include collaborative development, meeting evolving requirements with working software, simple design etc. These methods address the problem of volatile requirements by applying above practices. Thus, these practices reduce cost of change at later stage of software development. These methods do not hold big upfront for early estimation of size, cost and duration due to uncertainty in requirements. It has been observed that agile methods mostly rely on an expert opinion and historical data of project for estimation of cost, size and duration. It has been observed that these methods do not consider the vital factors affecting the cost, size and duration of project for estimation. In absence of historical data and experts, existing agile estimation methods such as analogy, planning poker become unpredictable. Therefore, there is a strong need to devise simple algorithmic method that incorporates the factors affecting the cost, size and duration of project. It also provides the basis for inexperienced practitioners to estimate more precisely. In this paper, we are presenting the study of both traditional and agile estimation methods with equivalence of terms and differences. We investigated some vital factors affecting the estimation of an agile project with scaling factor of low, medium and high. Also, an algorithm Constructive Agile Estimation Algorithm (*CAEA*) is proposed for incorporating vital factors.

*Keywords: Tr*aditional cost estimation; Agile cost estimation; Object points; New object points; Function points; Story points; Velocity and Ideal time.

1. Introduction

Estimation of Cost, Size and Duration (CSD) is key activity of software project management and it is very cumbersome in case of volatile requirements. Agile methods are very successful in volatile requirements due to their enriched practices. They recommend collaborative development, meeting evolving requirements with working

* Research Scholar, School of Computer Science, Takshila Campus, Devi Ahilya University, Khandwa Road, Indore, Madya Pradesh, India.

software and growing application organically from a number of iterations [Cockburn, 2007]. Due to uncertainty in the requirements, these methods do not have a big upfront to estimate the CSD of complete software with minimum calibration. Agile Estimation Methods (AEMs) are normally based upon analogy and expert opinion. Planning Poker method is the most commonly used method in AEMs as it involves expert opinion of persons from different areas of software development. In absence of historical data and experts, AEMs become unpredictable. Again, availability of highly experienced expert for every new project is a critical issue. Therefore, there is a need of algorithmic approach for agile estimation. Traditional Estimation Methods (TEMs) include Cost Construction Model (COCOMO), ESTIMATICS, Function Point (FP) based model etc. of which COCOMO II and FP based estimation methods incorporate various CSD affecting factors for software estimation. These factors include complexity, performance, multiples sites, volume of data etc. and require special attention at the time of design and coding of the software [Jorgenson and Shepperd, 2007]. It has been observed that CSD estimation in AEMs is performed on the basis of mainly; story points, velocity and ideal time [Kane, 2007] and these are computed with non-algorithmic methods.

In this paper, we have attempted to analyze TEMs and AEMs with their differences and established equivalence of the terms used in both methods in Section 2. Section 3 discusses the proposed vital factors that affect CSD of an agile project. We have proposed an algorithm Constructive Agile Estimation Algorithm (*CAEA*) for adjusting story points after inclusion of CSD affecting factors with a case study and metric for computation in Section 4. This section also describes the computation of size and duration of project. Finally, we conclude with results and future scope in Section 5.

2. Existing Methods for Estimation

The estimation is a process of determining amount of efforts, money, resources and time for building a software project with the help of available quality information. Many estimation methods have been proposed in last 30 years and almost all methods require quantitative information of productivity, size of project and other important factors that affect the project. There are various practices of software estimation such as analogy, expert opinion and empirical based practices [Jones, 2007]. Analogy based practices require historical data of projects as an input for comparison whereas expert opinion are intuition based [Jorgenson and Shepperd, 2007]. Empirical way is a practice of deriving the cost of software using some mathematical/ algorithmic model. Examples of methods that use such practices are FP based method and COCOMO II method in TEMs. Mostly, all traditional software development methods follow either COCOMO II or FP based estimation methods successfully due to complete set of requirement specification. Firstly, we describe the COCOMO II and FP based methods briefly in Section 2.1. Existing AEMs are discussed in Section 2.2.

2.1 Traditional Estimation Method (TEM)

We discuss COCOMO II and FP based method as the most popular TEMs for software estimation in brief as follows:

2.1.1 COCOMO II Method

COCOMO II considers LOC and efficiency with various factors such as transaction of data, online data entries, performance, security etc. for software cost estimation. It addresses the areas mainly; Application Composition Model (ACM), Early Design Model (EDM) and Post-architecture Stage Model (PSM). ACM is used during the early stages of software engineering practices i.e. at the time of consideration of software system interaction, user interfaces, performance assessment etc. EDM can be applied when requirements have been stabilized with basic software architecture. Lastly, during the construction of the software, PSM becomes necessary to implement. This method incorporates project size as a major factor in form of Object Points (OPs) for estimating cost and duration [USC, 2000]. OPs are computed using counts of screens, reports and components required to build the application. Each object instance is divided in three complexity levels mainly; simple, medium and difficult. OPs are counted on the basis of Boehm approach using reusability factors [Pressman, 2006].

2.1.2 FP Based Method

This method mainly focuses on information domain values that are derived from FP metric. Further, FP metric can be estimated with the help of number of external inputs/ outputs, number of external enquiries, internal logical files, external logical files and Value Adjustment Factors (VAF). VAF includes the factors associated with execution of software such as backup and recovery, distributed processing, volume of transaction, online data processing etc [www.ifpug.org]. The main benefit of FP measurement lies in its ability to obtain estimation in early phases of the development cycle with requirement specification sheet [Leung and Fan, 2000].

2.2. Agile Estimation Method (AEM)

Agile estimation is categorized in expert opinion, analogy, disagreegation and planning poker. In an expert opinion- based approach, an expert comment on the size and time after investigating the stories. The expert relies on the intuition and provides estimate for agile project. In estimating by analogy, estimator compares the story being developed with one or two other stories relatively. Planning poker is most popular estimating technique in which the participants are developers, designers and testers. It requires the opinion of multiple experts and also the justification of CSD estimation [Stiendl and Krogdahl, 2005].

It is well known that agile methods adopt the practice of accepting last minute changes in the software through gathering the requirements in iterative and incremental manner [Abrahamsson et al., 2007]. These requirements normally exist in the form of stories [Beck, 2006]. Evaluation of these stories in terms of story point which is a unit measure of work estimation involved in developing the feature with complexity of developing it. It expresses over all size of a user story, feature or other piece of work. Story point has been devised for assigning a relative size to each story [*http://blog.versionone.net/blog*]

and is used to estimate the velocity of project also. Velocity can be defined as the number of developed story points per iteration. Another metric i.e. ideal time is the time that is fully devoted to task without any interruption [Kane, 2007]. All of the above metrics are used to estimate the CSD of the project.

It is important to note that the above discussed agile estimation methods are completely human intuition based methods and hence may lead to create biased situation. At the same time, these methods may lead to the errors in case of inexperienced agile team. Therefore, there is strong need of analyzing the factors that affect the estimation of the agile project.

2.3 TEMs vs. AEMs

TEMs and AEMs possess some differences and equivalence in used terms based on their computing practices. TEMs consist of methods based on both algorithmic and non algorithmic approaches whereas AEMs follow mainly non-algorithmic approach for CSD estimation of project. TEMs assume that the requirements are well formalized before beginning of CSD estimation. On other hand, AEMs deal with a small set of requirements (which may grow organically in later stage of software development). Furthermore, TEMs derive the estimation of CSD for whole software but AEMs compute CSD for working software in estimation process. On contrary to AEMs, TEMs consider various factors affecting the CSD during the estimation process.

On other hand, TEMs and AEMs use the same quantitative information in terms of productivity and team size for CSD estimation of the project. We propose the equivalence of factors/ terms that are used in both TEMs and AEMs. This equivalence is shown in Table 1 and discussed as follows:

Size and Complexity: Size of the project is defined in terms of OPs and FPs in TEMs and computed through counts of various inputs, logical files, external file, screens etc. AEMs use story points to define the size of a story and sum of all story points is defined as size of the project. Size and complexity of project are major factors in cost and duration estimation and are computed in the form of OPs, FPs in TEMs (COCOM II, FP based estimation) and story points in AEMs. Thus, story points, FPs and OPs are equivalent to each other. However, OPs and FPs in TEMs use mathematical formula for computation and story points are estimated through heuristic approaches.

Efforts: TEMs introduce the term person per month for computing efforts spent on project. Similarly, AEMs uses the term elapsed time which is defined as time spent on project by a person. Person per month is total hours spent by a person on the development of the project same elapsed time as in AEMs. AEMs also introduce ideal time for estimation that refers to time required on a project with out any interruption.

Productivity: Productivity rate in TEMs is equivalent to velocity in AEMs. Productivity rate is computed through function of person per month and OPs. Velocity in AEMs is number of story points completed per iteration. Velocity/ productivity rate represents team competence on project and it is used to estimate time and cost of the project.

Table 1 Equivalence of TEMs and AEMs

	TEMs		AEMs
	COCOMO II	**FUNCTION POINT**	
Size and complexity	Object points	Function points	Story points
Efforts	Man-month	Man-month	Elapsed time/ Ideal time
Productivity	PROD=NOPs/ person per month	Calculated in terms of FP	Velocity= Number of story points per iteration

3. Vital Factors Affecting Agile Estimation

Some factors such as data communication, distributed data processing, performance, configuration, transaction, online data entry, end user efficiency, online data update, complex processing, reusability, installation ease, operational ease, multiple sites and change requirements have been efficiently used in FP for CSD estimation [Jones, 2007]. In COCOMO II, the additional factors such as personnel attributes (capability, experience, and language command) and computer factors (speed, turn around time, storage constraint) for cost calculation have been observed to be important factors [Leung and Fan, 2000]. Also, it is observed that the factors related to computers and personnel due to certain reasons are not important in AEMs. These reasons are mainly; i) agile methods always use latest tools for automated build and testing, which require higher configuration software ii) generally, agile team consists of experienced members. Thus, we propose some factors that affect the CSD in AEMs as shown in Table 2 and described as follows:

3.1 Project Domain

There are different activities involved for different project domains, affecting the CSD of the project. Various project domains are web based application, management information system, contract out sourced project, system projects, commercial projects and military project [Jones, 2007]. Each domain has been characterized by different types and number of activities. Activities include prototyping, coding, testing and project management etc. depending on the category of project domain. For example, activities in web based application are less as compared to military projects as it involves rigors testing and documentation. Therefore, military projects require more efforts in development and testing as compared to web based application.

3.2 Performance

Project performance may be characterized as execution time, designing and coding standards, accuracy in outcome etc. as per customer requirements. Thus, the necessary features must be included in design and architecture of the software for achieving the performance level. These features of project increase the cost and duration of the project.

For example, Google search engine needs better performance for responding to the queries whereas performance is less important for a commercial or MIS application.

3.3 Configuration

It is one of the key factors in CSD estimation. Configuration is in context of estimation refers to special hardware and software requirements to run the software smoothly. For example, employee attendance system may require special hardware i.e. digital camera or card recognition machine that certainly increases the cost and efforts in designing, coding and implementation. Further, internet requirements and programming of hardware/ software specific projects need special efforts thereby increasing CSD of project. Therefore, CSD of such special configuration project is high as compared to other softwares.

3.4 Data Transaction

Data transaction refers to volume and frequency of data transfer from peripherals/ machine to remote machine/ peripheral. If the volume of data transaction is high to affect the design and development, it requires more efforts to develop the software and hence, increasing the cost of software. The automation of school management is a low cost project due to low data transaction. On the contrary, an agile application on airline reservation requires high volume of data transaction. Hence, it requires special efforts in design and implementation to increase the CSD of project.

3.5 Complex Processing

Software project may require complex processing in the form of scientific rules or commercial rules or it may require remote accessing of computers. Complex processing requires special attention at various stages of software resulting increase in the cost and duration of the software. Agile practices suggest refactoring the code and data to simplify design and coding. Still, complex processing needs higher efficiency of team members and rigors tests to avoid redundancy.

3.6 Ease of Operation

Operation ease can be achieved by providing facilities to users other than must have features. It may be in the form of backup recovery facility, minimum user input etc. It increases the usage of the software which in turn increases the quality of the software as well as customer satisfaction. System design and architecture include numerous activities to provide maximum operational ease to user in turn increasing the CSD of the software project.

3.7 Multiple sites

If software runs on multiple sites or many team members work together in distributed environment, cost of the software will increase due to the cost of communication and coordination. Communication delay in distributed environment must be considered in estimation of the duration of project.

3.8 Security

Security may be considered as data security, operational security, code security etc. depending on stakeholders' requirements. All the aspects of security must be taken care at the time of designing, coding and implementation. It may increase the complexity in design, coding and user interfaces and hence resulting into the increase in CSD of project as well as software. For example, online money transaction software projects require various levels of securities to maintain the integrity of software.

Table 2 Factors in Agile Estimation

Sr. no	Factors	Rating in Agile Estimation
1	Project Domain	Web based –Low, (MIS, outsourcing, commercial) – Medium, System and Military project –high.
2	Performance	Performance requirements can be low, medium and high.
3	Configuration	If any special hardware/ software required for project, rating must be high, otherwise, low.
4	Transaction	If high volume of data transaction in the project, rating is high otherwise, low.
5	Complex processing	Complex processing is required i.e. commercial and scientific rule, rating is high otherwise, low.
6	Operation ease	If software contains back up recovery, minimal user intervention facility, rating is high otherwise, low
7	Multiple sites	For global and distributed software, rating is high as the cost of communication increases.
8	Security	Rating depends on various security levels

4. Constructive Agile Estimation Algorithm (CAEA)

In this section, we first propose constructive agile software estimation process followed by the CSD algorithm for the same agile software. We divide constructive agile estimation process into two phases namely; Early Estimation (EE) and Iterative Estimation (IE) as shown in Fig. 1. An estimator can update the estimates whenever the uncertainty in requirements reduces. The purpose of EE is to identify the scope of the project by envisaging the upfront with just enough requirements. This provides just enough understanding to put together an initial budget and schedule without paying the price of excessive documentation. EE also provides the platform for fix budget agile project and generates a satisfaction amongst the stakeholders by providing range of estimate to reflect temporal risk. On the other hand, IE is an iterative activity that starts at the beginning of iteration to incorporate new requirements/ changes. These requirements/ changes may arise from the customers with detail information of system. Constructive agile estimation process considers only clear specified requirements and other factors that affect the estimation to derive CSD of the project as shown in Fig. 1. Here, estimation process starts after the prioritization of requirements and EE is just to understand the CSD involved in project. After finalization of project, iteration planning starts and continues till all the requirements/ changes required by customers are exhausted. EE and IE use story points for estimating CSD. Existing AEMs use expert opinion and historical data for evaluating story points. In this section, we propose *CAEA* to evaluate story points after the inclusion of various CSD affecting factors in AEM. We also discussed the

computation of variables in various projects to establish the fact that inclusion of vital factors in agile estimation generates realistic and more precise CSD estimation.
An algorithm *CAEA* is based on above constructive agile estimation and computes the story points for CSD estimation. It incorporates the vital factors such as project domain, performance, configuration etc. as discussed in Section 3. We have graded the intensity of these factors on the scale of low, medium and high based upon the complexity of the project. It is preferred to map the intensity levels with mathematical series such as square series (1, 4, 9) or Fibonacci series (2, 3, 5). Square series has been proved to be the most preferred series in agile estimation since it provides realistic level of accuracy for complex and ill-defined project [Fedrick, 2007].

Formal description of proposed *CAEA* is described as follows:

Algorithm: *CAEA*

// This algorithm computes the of story points on the basis of the input as the grades of vital factors of a project //

STEP 1: Vital factors of project are identified on the grade of low, medium and high using square series or Fibonacci series.

STEP 2: Compute sum of all grades of various factors for a project denoted as Unadjusted Value (UV).

STEP 3: Decompose the project in small tasks or stories.

STEP 4: Assign Story Point (SP) to each story based upon the size.

STEP 5: Compute New Story Point (NSP) by using equation (1).

STEP 6: Compute SOP by using equation (2).

STEP 7: Compute DOP through equation (3).

$$NSP = SP + 0.1 * UV \quad \text{----------(1)}$$

$$\text{Size of Project(SOP)} = \sum_{i=1..n} NSP_i \quad \text{----------(2)}$$

$$\text{Duration of project (DOP)} = SOP / \text{Velocity} \quad \text{----------(3)}$$

where SP is story point of a story,
UV is unadjusted value,
NSP_i is NSP of i^{th} the story and n is total number of stories of project,
SOP is size of project,
velocity is number of NSP developed in a iteration.

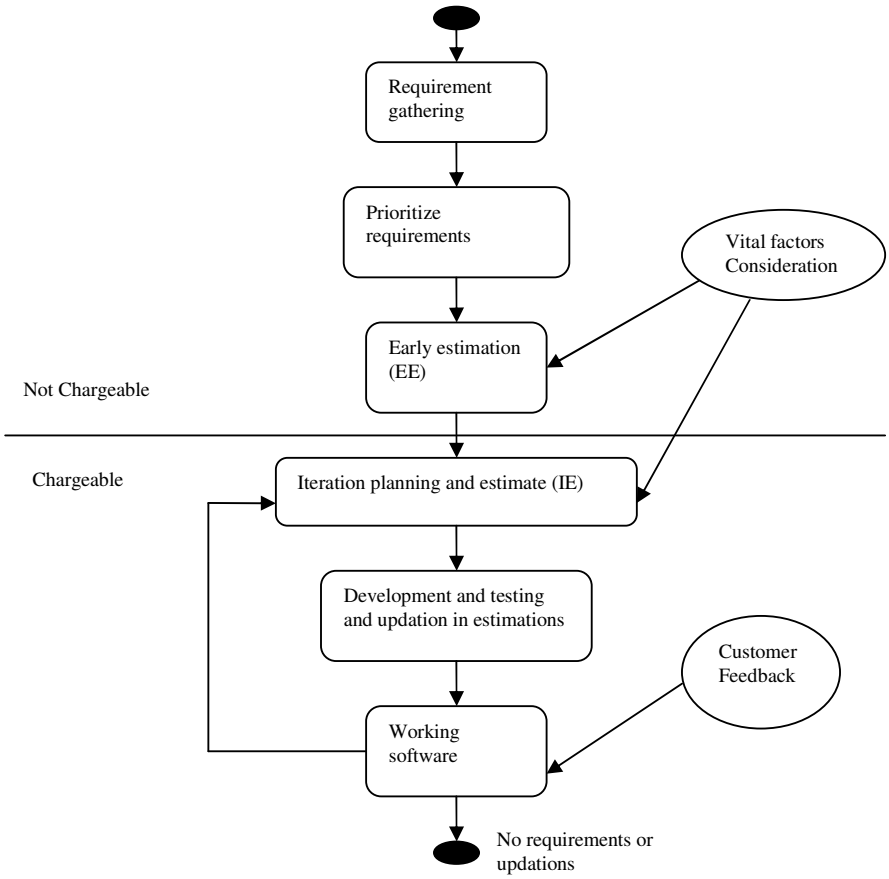

Fig. 1 Estimation Activity Diagram

4.1 Case studies

In this section, we consider the various applications of small project category with SP as 25. These application include web based application, contract outsourced and critical or military projects. The performance of the algorithm has been investigated for the various types or levels of vital factors such as performance, configuration, complex processing and data transaction. We show this performance in terms of different case studies as follows:

Case I:

The proposed algorithm takes the input as the high levels of vital factors mainly; performance, configuration, complex processing and volume of data transactions. Also,

various intensity levels of other vital factors such as multiple sites, operation ease and security have been considered as another set of input. The values of UV, NSP1, NSP2 and NSP3 have been computed using the proposed algorithm at different intensity levels for various applications of small project category. Table 3 shows the NSP computations for the small projects with SP as 25. For example, for a small military project with specified input of vital factors (specially the combination of multiple sites, high operation ease and higher security level), NSP has been computed as 44.3. It has been observed that inclusion of vital factors in such projects, changed the estimated SP to approximately twice of its previous SP and estimated more realistic values with lower deviations in later stages of IE estimation.

Case II:

In this case, *CAEA* uses the higher levels of performance, complex processing, configuration but low data transaction as input with various intensity levels of remaining vital factors such as multiple sites, operation ease and security levels. It has been noticed that change in single vital factor (i.e. data transaction from high to low) make a lot of difference in NSP computation as shown in Table 4. For example, the small military project with high configuration, high performance and complex processing with low data transaction, single site and low security factors, *CAEA* computes NSP value 34.9. It has been analyzed that any change in the intensity level of vital factor generate different the NSP value thereby changing the CSD estimation.

Case III:

Simple projects with lower values of all aforesaid vital factors at different intensity of other factors (such as multiple site, operation ease and security) have been considered in this case. NSP computations for such projects have been presented in Table 5. For a web based project with all vital factors as low, possesses NSP value as 26.7. It shows that there exists a marginal difference in previous SP in comparison of Case I and Case II.

Table 3 NSP Computation for Small Projects with

	Multiple sites																									
	L (1)									M (4)									H (9)							
	Operation Ease									Operation Ease									Operation Ease							
	L(1)			M(4)			H(9)			L(1)			M(4)			H(9)			L(1)			M(4)			H(9)		
	Security			Security			Security			Security			Security			Security			Security			Security			Security		
	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H
UV[1]	39	43	48	43	44	51	48	51	56	42	45	50	45	48	53	50	53	58	47	50	53	50	53	58	55	58	63
NSP1	29.9	30.3	30.8	30.3	30.4	31.1	30.8	31.1	31.6	30.2	30.5	31	30.5	30.8	31.3	31	31.3	31.8	30.7	31	32.3	31	31.3	31.8	31.5	31.8	32.3
NSP2	32.9	33.3	33.8	33.3	33.4	34.1	33.8	34.1	34.6	33.2	33.5	34	33.5	33.8	34.3	34	34.3	34.8	33.7	34	33.7	34	34.3	34.8	34.5	34.8	35.3
NSP3	41.9	42.3	42.8	42.3	42.4	43..1	42.8	43.1	43.6	42.2	42.5	43	42.5	42.8	43.3	43	43.3	42.8	42.7	43	42.7	43	43.3	43.8	43.5	43.8	44.3

Table 4 NSP Computation for Small Projects with
High performance, High configuration, High complex processing, Low data transaction

	Multiple sites																									
	L (1)									M (4)									H (9)							
	Operation Ease									Operation Ease									Operation Ease							
	L(1)			M(4)			H(9)			L(1)			M(4)			H(9)			L(1)			M(4)			H(9)		
	Security			Security			Security			Security			Security			Security			Security			Security			Security		
	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H
UV[1]	31	33	37	35	36	43	40	43	48	44	37	40	47	40	45	40	45	50	39	40	45	40	45	50	47	50	55
NSP1	26.9	27.3	29.7	27.5	29.6	30.1	30	30.3	29.8	30	29.7	30	30.7	30	30.5	30	30.5	31	29.9	30	30.5	30	30.5	31	30.7	31	31.5
NSP2	29.9	30.3	32.7	30.5	25.4	33.1	33	33.3	32.8	33	33.7	33	33.7	33	33.5	33	33.5	34	33.9	33	33.5	33	33.5	34	33.7	34	34.5
NSP3	34.9	35.3	37.7	35.5	37.6	38.1	38	38.3	37.8	38	37.7	38	37.7	38	38.5	38	38.5	39	37.9	38	38.5	38	38.3	39	38.5	39	39.5

Table 5 NSP Computation for Small Projects with
Low performance, Low configuration, Low complex processing, Low data transaction

	Multiple Sites																									
	L (1)									M (4)									H (9)							
	Operation Ease									Operation Ease									Operation Ease							
	L(1)			M(4)			H(9)			L(1)			M(4)			H(9)			L(1)			M(4)			H(9)		
	Security			Security			Security			Security			Security			Security			Security			Security			Security		
	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H
UV[1]	7	10	15	10	13	18	15	18	23	10	13	18	13	16	21	18	21	26	15	18	23	18	21	26	23	26	31
NSP1	26.7	27	27.5	27	27.3	27.8	27.5	27.8	28.3	27	27.3	27.8	27.3	22.8	28.1	27.8	28.1	28.6	27.5	27.8	28.3	27.8	28.1	28.6	28.3	31.6	29.3
NSP2	29.7	29	30.5	29	30.3	30.1	30.5	30.8	31.3	30	30.3	30.8	30.3	25.8	31.1	30.8	31.1	31.6	30.5	30.8	31.3	30.8	31.1	31.6	31.3	31.6	32.3
NSP3	33.7	34	35.5	34	35.3	35..8	35.5	35.8	36.3	35	35.3	35.8	34.5	35.3	36.1	35.8	36.1	36.6	35.5	35.8	36.3	35.8	36.1	36.6	36.3	36.6	37.3

High performance, High configuration, High complex processing, High data transaction

[1] Represents Unadjusted Value,
NSP1 represents small web application,
NSP2 represents contract outsourced project,
NSP3 represents critical or military project,
L, M and H indicate Low, Medium and High.

5. Conclusions and Future Scope

We have proposed *CAEA* for CSD estimation by incorporating vital factors with different intensity levels. These vital factors include mainly; project domain, configuration, performance, complex processing, data transaction, operation ease, multiple sites and security. It has been noticed that NSP computation through *CAEA* generates the more realistic and precise estimation in case of all vital factors having higher values. Also, it has been observed that inclusion of vital factors in CSD estimation changes the value of SP to approximately twice. Applicability of *CAEA* is also beneficial in case of any two vital factors having low values. Performance analysis of *CAEA* in various small projects state that NSP values obtained in Case II are lower than the NSP values of case I. It reveals that any of the vital factors having low value decreases the CSD of the project. Thus, our study of various projects reveals that inclusion of aforesaid factors leads to drastic change in story estimation which generates more realistic CSD estimation. We also found that computation of NSP in projects having low value vital factors is time consuming and minor affect on CSD estimation. There exist two fold benefits of *CAEA:* firstly, inclusion of aforesaid factors is useful to the average project leaders to estimate the story points more precisely. Secondly, it reduces the risk of falling project in chaos by providing realistic figures of CSD.

References

Abrahamsson P., Salo O., Ronkainen J. and Warsta J. (2003), *New Directions on Agile Methods : A Comparative Analysis", Proceeding 25th International conference on Software Engineering.*

Abrahamsson P. (2003),*Extreme Programming : Fist Results from Controlled Case Study, Proceedings of 29th EUROMICRO Conference New Wave of System Architecture..*

Beck K. (2006), *Extreme Programming Explained, Pearson Education Low price Edition Asia.*

Cabri A., Griffith M., Earn (2006)*, Value and Agile Reporting, Proceedings of Agile 2006 Conference(AGILE'06).*

http://www.memory.com.uy/noticias/0247/Effort%20Estimation%20in%20Agile%20Software%20Development.pdf.

Cockburn A. (2007), *Agile Software Development, Pearson Education, Asia, Low Price Edition.*

Cohen M. (2006), *Agile Estimation and Planning , Pearson Low Price Edition Asia.*

Elie J. (2008), *Case Study: Using Agile Software Development at the Educational Credit Management Cooperation, Agile Journal, Jul.*

Fedrick C. (2007), *Case study: How Douglas County, CO cut the project timeline in half, Agile Journal.*

Gibb T. (2003), *Software Project Management: adding Stakeholders Metric to agile Project Upgrade e- journal http://upggrade-cepis.org.*

Gruschke T. (2005), *Empirical Studies on Software Cost Estimation: Training of Effort Estimation Uncertainty Assessment Skills, 11th IEEE International Software metric Symposium.*

Information and tools about planning poker available at www.planningpoker.com

Information about Agile Estimation at http://blog.versionone.net/blog

Information about COCOM II available at http://www.softstarsystems.com/cocomo2.htm

International function point user group information about function point measurement available at www.ifug.org

Jawadekar W. (2005), *Software Engineering Principles and Practices, Tata McGraw-Hill First Edition.*

Jones C. (2007) , *Estimating Software Costs Bringing Realism to Estimating , Tata McGraw-Hill, 2nd Edition*.

Jorgenson M., Shepperd M. (2007), *A systematic review of software development cost estimation studies IEEE transaction on software engineering, Vol. 33, NO. 1, pp. 33-53.*

Kane B. (2007), *Estimating and Tracking Agile Project , Student Paper published in PM World, Vol. XI issue V.*

Leung H., Fan Z. (2000), *Software cost estimation, ftp://cs.pitt.edu/chang/handbook/42b.pdf*

Mohammed M. (2005), *Project Management Measures in Small Enterprises, Master Thesis, Department of Computer Science, Umea University, Sweden.*

Pressman R. (2006), *Software Engineering a Practitioner Guide, McGraw- Hill 6th Edition.*

Shore J., Warden S. (2007), *The Art of Agile Development, O'Reily Publication, First Edition.*

Stiendl, C., Krogdahl, P. (2005), *Estimation in Agile Projects, Proceedings of Best Practices in Project estimation Conference at IBM Academy of Technology.*

USC COCOMO II (2000), *Software conference Manual , University of Southern California*

sunset.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_manual2000.0.pdf.