Circulation in Computer Science
Vol.2, No.6, pp: (20-24), July 2017
https://doi.org/10.22632/ccs-2017-252-32

Hybridization of Class Responsibility Collaborators Model (HCRCM) with Function Point to enhance Project Estimation Cost in Agile Software Development

Faki Agebee Silas
Computer Science Department,
University of Ilorin,
Ilorin-Nigeria

Musa Yusuf
Computer Science Department
Bingham University
Karu, Nasarawa State

Anah Hassan Bijik
Computer Science Department
Bingham University
Karu Nasarawa State

ABSTRACT
Estimating software cost in an agile system in terms of effort is very challenging. This is because the traditional models of software cost estimation do not completely fit in the agile development process. This paper presents a methodology to enhance the cost of project estimation in agile development. The hybridization adopts Class Responsibility Collaborators models with function point thereby boosting the agile software development estimation process. The study found out that adopting the Hybridized Class Responsibility Collaborator with function point has great improvement on cost estimation in agile software development.

Keywords
Agile development, Class Responsibility Collaborators, Function points, Line of codes, Scrum, Software estimation models.

1. INTRODUCTION
Software Cost estimation is a process whereby amount of effort needed to build a software system is estimated. This process is important because it gives a blueprint of the amount of effort and cost of building software. It also determines in monetary terms the cost of software project in respect to the time, effort and resources that are required to build it. Though, software estimation in most cases focus on size measure, there exist other useful measurable metrics like estimating progress made, estimation changes in development, estimating the level of risk due to changes and the value earned at a particular developmental stage [1].

Estimation of software in terms of size is widely done by using either line of code (LOC) or function point (FP). Both LOC and FP measure size of software but not actually the same measurement. LOC focus on size of system built which has much emphasis on technology employ in the building, design process and coding style. FP measures deliverable function independent of technology being used in the development process with focus on counting input, outputs, external interface, files and inquiries.

There are several models and techniques of software development like water fall, Spiral, incremental and agile model. Among all, agile model has gain ground recently due to its flexibility, evolutionary and highly collaborative, and balanced it create in effort and development process [2] and reduction in project failures because of faster development cycles that deliver functioning software sooner than other models. Agile software model is like a toolbox, a lazy worker with the best tools remains a lazy man, because of the assertion, not all projects that adopt agile model do succeed [3]. Among the advantages agile models offers, there still no standard acceptable method on how cost should be estimated even to agile purist [4]. With the growing popularity of object oriented programming languages in software development, it is becoming more difficult to estimate software cost in nontraditional development models like agile development [5]. This is due to the fact to there exist variation of coding syntax, semantics and style involved in object oriented languages. One language may correctly code a function in fewer lines while another does same task in more lines of code.

This study therefor, improves on cost estimation of software by using class collaboration estimation techniques with function point. Moreover, this method works better because, most programming languages now are object oriented in nature and function point builds on functionality which is independents of programming languages.

2. REVIEW OF RELATED MODELS

2.1 General Overview of Estimation Models
The successful completion of any software project depends on proper estimation of cost required in development. Unfortunately, there is no single standard accurate way to exactly estimate software cost in an agile system [6].

Traditionally, there are approaches that help estimate software cost like line of codes and function points in addition to other point's estimation techniques such as use case point, object point, class point. At the initial stage of software development, the expertise and skills of developers and requirements are not clearly known, also due to constant change in technology and the duration a software process takes, it become unfamiliar to say which computers the software will be implemented on. All these and many others add to the fundamental issues that makes software cost estimation very elusive [7].

Base on the fact that many authors classified cost estimation method differently, cost estimation remain a top issues as it was several years back [8]. Due to recent technology advancement and improve programming languages, many software estimation models abound. Irrespective of software estimation model, they are broadly classified as algorithmic and non-algorithmic [9]. Making use of either algorithmic or non-algorithmic or both is the prerogative of the developing team which is hinge on expertise, problem at hand and requirement understanding.

2.2 Algorithmic Software Estimation Models

Algorithmic cost estimation basically makes use of mathematical formulas [10], which leads to some form of equation(s) which are used for estimations.

A general form of algorithmic model could be

$$Effort\ (Cost) = f(X_1,\ X_2,\ X_3,\ X_4\ ,\ldots,\ X_n) \text{ --- (1)}$$

Where x_1, x_2, x_3 . . . x_n is the vector of cost factors.

Some of the cost factors could be computer, personal, project or product factors. In practice, setting quantities for the factors is a herculean task to a point that some developers decide to ignore them in some project [11].

Most Algorithmic models takes care of cost based on types of software, size of project, software team, software attributes and the process followed to attain estimation. Line of codes (LOC) estimation model comes in to play when estimation in size of software is of interest [12]. With the fact that LOC are difficult to implement in the early life of a project before design is made and is heavily influence by choice of programming language, it still remains a foundation and simplest estimation metrics. It has the ability of decent applications if the lines counted are define to mean logical lines rather than irrelevant lines [13]. According to [2], line of Codes is the simplest method of estimating size and effort in software projects. This model measures software by number of relevant line. Due to the physical nature of lines of codes, it is possible for manual counting to be eliminated by automating the counting system. Because lines of code can be seen and the effect easily visualized, it serve as metric for measuring software size.

Function point model was first presented by [11], [14] with intent of covering wider range of businesses and applications. Function points model uses estimates based on size of requirements which is the functionality of the project. The estimation is carried out using factors as user inputs, user outputs, logic files, inquiries and interfaces. A complexity degree of simple, medium and complex are attached with weights values as 1, 2 and 3 respectively.

Constructive Cost Model (COCOMO) was proposed by Boehm in 1981 [6]. This model uses equations and other parameters which are data collected from previous project. The latest version COCOMO II uses Bayesian statistical analysis of empirical data on completed projects and expert opinions. It has three estimation models (Application suite, early design and final design architecture) to estimate effort and cost. COCCMO models are comprehensive with large numbers of parameter that can take a range of values [15].

Other algorithm models of software estimation are linear, multiplicative, software evaluation and estimation of resources-software estimation (SEER-REM), and Putman model.

2.3 Non-Algorithmic Software Estimation Models

Non-algorithmic models estimates software projects using inferences, previous experience and analytical comparisons [16]. In estimation by analogy model, completed project of similar are used and estimation is done based on their cost. This proceeds in steps as choosing an analogy, investigating similarity and differences for previous projects, examining the analogy and then provision of an estimation.

According to [17] expert estimation model involved getting information from experts who have intensive experiences from similar projects. This model involved consultation and is applicable in situation where data and information are scanty of past similar projects. Delphi is example of expert model. In recent times, difficulty of estimation techniques and availability of standard accurate result is leading many developers to harnessed artificial intelligence prow in software estimation hence the use of machine leaning model [18]. Machine leaning model is best suited in projects where high accuracy is required. This is achieved by training some rules and running them over and over again in different software cycles. Machine leaning model could be Artificial Neural Network (ANN) which composed of layers called neurons which are used in effort estimation, fuzzy model which is applicable in situation where decision making is very difficult and conditions are vague or feed forward neural network model which disallowed self-loop or backward feeds.

2.4 Estimation of Software Cost in Agile Environment

Software development is becoming highly a complex process with too many variables and requirements having impact on the system [19]. This leads to conflict when traditional software estimations models are solely used in cost estimations due to changing requirement as software process progresses. Many authors are recently applying combinations of models to estimates software cost. This is evident in [20] as a total number of story points in combination with ANN is implemented to estimate effort in an agile software product.

The accuracy of the estimation model is enhance by using fireworks and Levenberq-Marquardt algorithms for weight optimization and biased of the ANN. The uses of statistical method are evident as [21] applied support vector regression (SVR) model in software effort estimation. The model implemented on NASA dataset software projects is found to outperform radial basis neural networks and linear regression models.

Agile development methodology could be Srcum, extreme programming, crystal methods, adaptive software development, agile modeling or feature driven developments. Srcum in addition to possessing agile concepts and methodologies has project management capabilities which help developers find the next task in the next iteration. The basic objective of scrum is its project simplification, easy update documentation and higher team iteration over exhaustive document [22].

Extreme programming places its focus on project development rather than management [23]. Extreme programming earlier was an adopted method for small high-tech product companies but has now been used for companies of different sizes [24]. It provides a simple and seemingly naïve principles that are specific and guided values that work at all phases of a software development cycle [25] It suffice to say that different problem domain in software development requires different agile methodologies. Some agile methodologies focus on business problems why others are for development. In the interest of the majority, achieving success in project development within shortest possible time is of utmost interest [26]

3. METHODOLOGY

Class responsibility collaboration (CRC) is a great technique that is used in designing software especially when used in group/workshop environment.

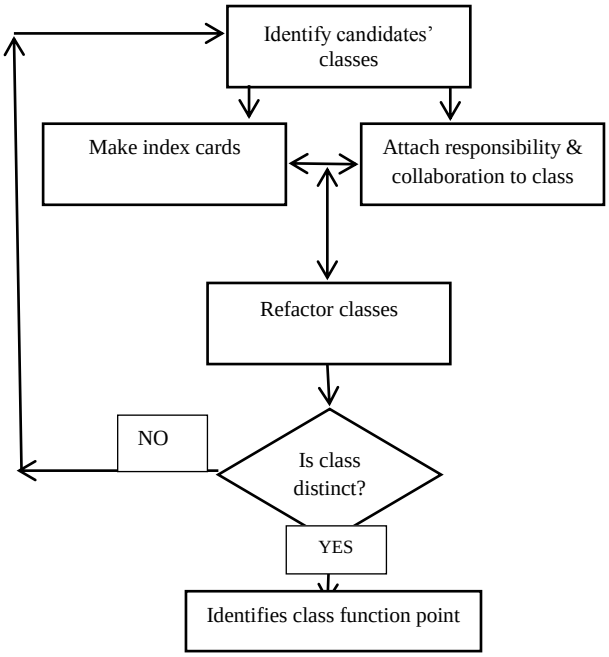

Fig 1: Flowchart of class responsibility collaboration model

After designing stage in software development is achieved, the next stage is implementation which is most cases is achieved by coding in a suitable choice language. In this study, a simple walk through earlier design diagram like use stories and use cases diagrams are used to identify the candidates classes which are written on index cards. The classes are further furnished with responsibilities and collaborators. In order to improve on re-usability, some of the classes are refactored by combining or splitting some classes depending on whether they have same responsibilities or too many responsibilities. This process is done iteratively and incrementally with the software team evaluating and improving on each iteration. By doing this, all services (massages and functions) rendered by each class and its collaboration are clearly visible leaving us with a clear and distinct CRC card with functions for the software project.

Also this introduced and reorganized new classes, as the iteration continues, existing classes will disappear, and new ones emerges until user requirement are achieved. This methodology is summarized in flowchart (see Fig 1). On completion of the CRC, function point analysis model is then used to estimate the cost based on functions in the CRC diagrams. This model has the advantages of edging out similar function(s) or function(s) that performs same or similar activities among classes due to the fact that during factoring of CRC classes, all association, extension and inclusion are all taken care of. The cost of a function is seen here as an independent entity with no same behavior as others found in same or different classes. If the behavior is found to be same, such function assumes cost only once to avoid duplication of cost. The cost involve evaluating all individual functions f(x) noting the same functions in same of other classes as S(n).

$$Effort\ Cost = \sum_{i=1}^{n} f(x) - \sum_{i=1}^{n} (Sn) \quad (2)$$

In eqn. (2), the cost of effort is defined to be the different in cost of all functions and that of similar functions. The notion of taking the difference here it to avoid double counting of functions, This is evident in association, friendship, extension, and reusable nature of object oriented programming languages which are the most programming language in use recently.

4. FINDINGS

On application of this model to small in-house project, the study found out that cost estimation is achieved closed to target value. The cost value of the project keeps depreciating to its true value after every iteration. This is because at the end of every iteration, similar class collaborators (similar or same functions) cancelled out there by reducing the number of functions to be estimated. This is evident as shown in Figure 2 and 3. In Figure 2, a CRC SaleClerk began with eight responsibilities and six collaborations.

SaleClerk	
Responsibilities	**Collaborations**
empID:	manager
empSal:	Supervisor
embRank:	Customer
clockInTime()	Record
recordSale()	time_Record
issueRecipts()	CEO
recordComplints()	
clockOutTime()	

Fig 2: SaleClerk CRC at first iteration

At the last iteration/increment stage which is preceded by creating new classes, deleting unwanted or duplicate classes, factoring classes, adding and subtracting similar responsibilities and collaboration, the SaleClerk CRC ended up with five responsibilities and three collaborations as shown in Figure 3.

SaleClerk	
Responsibilities	**Collaborations**
empID:	Supervisor
clockInTime()	Customer
recordSale()	time_Record
issueRecipts()	
clockOutTime()	

Fig 3: Saleclerk CRC at 4th iteration

The summery of the CRC diagrams modeled in the project from first to the fourth iteration which was the last is shown in Figure 4.

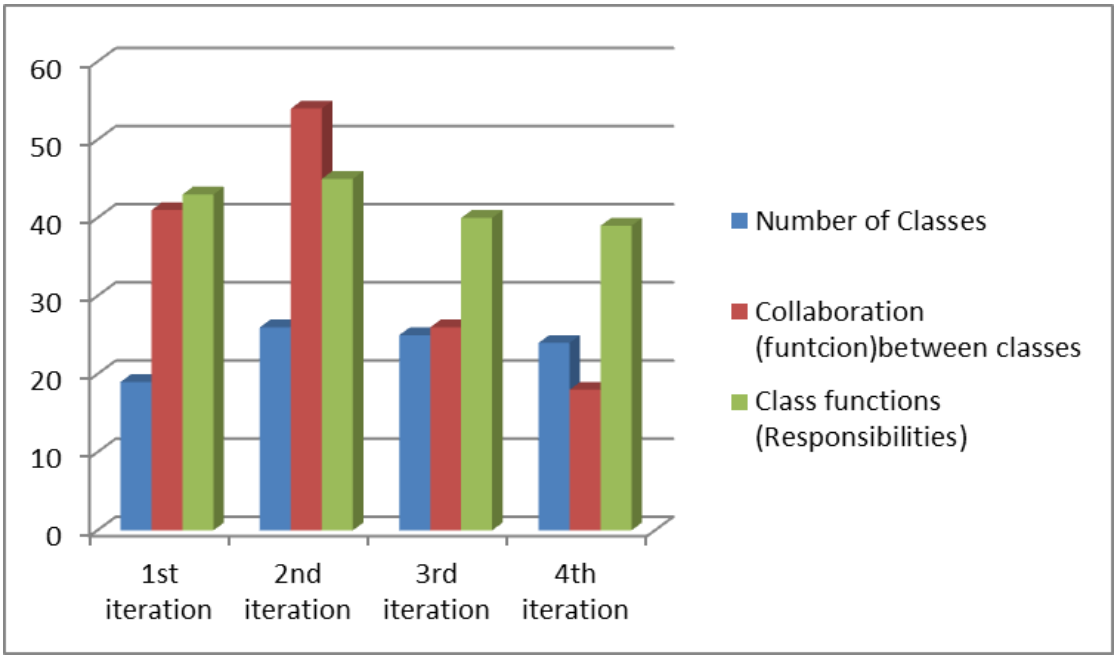

Fig 2: diagram showing objects behavio at different iterations.

It can be observed from Figure 4 that at first iteration, 19 classes were identified with 43 class responsibility functions and 41 collaboration functions. After series of class factoring, removal of similar function and addition of new ones, the last iteration which was the fourth produces 24 distinct important classes with 18 collaboration functions and 39 class functions. The reductions in function from first to last iteration causes a downward shift in cost per function (because the function became fewer) as priced by programmers thus lowering the price of the software cost close to the projected value from initial stage.

5. CONCLUSION

Agile development models in software have come to stay but standardization is the issue leaving an open system that allows different developers on ways to improve on cost estimation using it. This study hybridized CRC with function point to perform better cost estimation. The benefits here are that since CRC is applicable on all software developing stages, it is easier to applied coding from it and also, since association, extension, reusability is prominent in CRC, double calculation are eliminated.

This model also enables developers to evaluate the requirement gatherings techniques use because of its ability to track scope creep (growth or decline) of the project.

6. FURTHER STUDIES

The model discussed in this method has been implemented in projects of not too large a size. It is the intension of the researcher to implement this model on larger projects and compare it with other estimation models to evaluate its strength of estimation.

7. REFERENCES

[1] Abdulbasit, S.B. (2011). Proceeding of National Conference, Computing for National Development, New Dehlhi, India

[2] Evita, C. and Anirban, B. (2012). Efforts Estimation in Agile Software Development Using Story Point. International Journal of Applied Information System. Vol 3, No. 7, Foundation of Computer Science, New York, USA

[3] Dipendra, G., Shirley, G,. and Stuart, C. (2016). Software development team views of success factor in agile project.Annula conference of the National Advisory Committee on Computing, New Zealand.

[4] Mercin, N. (2010). Agile team Meet a Fixed Price Contract

[5] ESI international an informa business (2010). Successful Solutions through agile project management, An ESI international white paper. USA

[6] Waman, S. (2004). Software Engineering Principles and Practices, Tata McGraw.Hill Publishing Company Limited, New Delhi.

[7] Alexia, A and Jeffrey, W. (2009). Challengers reporting projects cost and risk to owner decision makers, International Women projects control, AACE international Transaction, USA.

[8] Jovan, Z,. Zorica, M,. Dragan, M,. Alexsandra, D,. and Sladana, V. (2011). Methods of effort estimation in software engineering International Symposium Engineering Management And Competitiveness, Zrenjanin, Serbia

[9] Nerker, L,. and Yawalkar, P. (2014). Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review. International Journal of Computer Applications, Innovations and Trends in Computer and Communication Engineering,

[10] Mathias, K. (2011). Software Test Effort Estimation Methods, Available on https://www.kerstner.at, retrieved on 9th, June, 2017

[11] Valid, K. and Dayang, N. (2010). Software Cost Estimation Methods: A Review, Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1

[12] Adolfo, V. (2014), Introduction to software project management, Auerbach publications, USA

[13] Mathias, K.. (2011). Software Test Effort Estimation Methods, Available on https://www.kerstner.at, retrieved on 9th, June, 2017

[14] Geetika,B,. and Kuntal, B. (2013). A Review on Cost and Effort Estimation Approach for Software Development, International Journal of Engineering and Innovative Technology, Volume 3, Issue 4,

[15] Doban, O,. and Andras, P. (2001). Cost Estimation Driven Software Development Process, 27th Conference of Euromicro, Warsaw, Poland

[16] Josephne, M,. and Rajeshwari, M. (2013). A Study On Software Cost Estimation, International Journal of Emerging Trend & Technology in Computer Science, Special Edition.

[17] Omprakash, T,. Jyoti, S,. and Poonam, R. (2014). Comparative Analysis of Software Cost And Effort Estimation Methods: A Review, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.

[18] Tirimula, R,. Satchidananda, D,. and Rajib, M. (2012). Computational intelligence in software cost estimation: an emerging paradigm, ACM SIGSOFT Software Engineering Notes, Volume 37 Issue 3, Pages 1-7 doi>10.1145/180921.2180932

[19] Victor, S. (2004). An introduction to agile software development. Danube technologies Inc, Belevue.

[20] Tung, K. and Li Thi M.H. (2016). An Effort Estimation Approach for Agile Software Development using Fireworks Algorithm Optimized Neural Network, International Journal of Computer Science and Information Security, Vol. 14, No. 7.

[21] Adriano, L. (2006). Estimation of software project effort work with support vector regression, Journal of Neurocomputing, Volume 69 Issue 13-15, *pages* 1749-1753.

[22] M. Cristal, D. Wildt and R. Prikladnicki, Usage of SCRUM Practices within a Global Company. Global

[23] Arun, K,. and Tejaswani, N.(2016), Agile Methodologies in Software Engineering and Web Engineering, International .Journal of. Education and Management Engineering, 5, 1-1

[24] Schwaber, C. and Fichera, R, Corporate IT leads the second wave of agile adoption. Forrester Research, Inc, 2005.

[25] Malik, H. and Siew, H. (2009), Review of agile methodologies in software development, International Journal of Research and Reviews in Applied Sciences, Volume 1, Issue 1

[26] Sriram, R,. and Saji, k. (2016), Choice of Agile Methodologies in Software Development: A Vendor Perspective,Journal of international technology and management, volume 25, issue 2.