2016 Symposium on Colossal Data Analysis and Networking (CDAN)

Early Phase Software Effort Estimation Model

Priya Agrawal, Shraddha Kumar
CSE Department, Sushila Devi Bansal College of Technology, Indore(M.P.), India
priya20agrawal@gmail.com,shraddhak99@gmail.com

***Abstract*— Software cost estimation is the measurement of the effort and resource required to develop a software system for a particular defined time period. From the last few years, many software cost estimation models have been proposed for the estimation of effort and development time. The cost of the software can be estimated easily in the mid of project development. But with further study, we found that the cost must be estimated before the start of the project, to satisfy customer as well as developer's needs. In earlier proposed cost estimation models, cost estimation is done with more than 20 parameters at the early conceptual phase and if input is not defined using logical approach, then the results of estimation are unpredictable. This paper presents simple approach for estimating software development effort with a minimum set of parameters yet sufficient, that can be easily identified at an early stage while considering all possible aspects. We also introduced a weight factor in the estimation of effort for improving the accuracy of the estimation and the proposed weight factor is calculated by expert learning system. Further, we developed a web based tool for the estimation of cost based on our proposed approach. Finally, we compare our result with previous works on early estimation and conclude with the points of accuracy that we observe while comparing the results with existing approaches.**

Keywords*— Project Estimation, Effort Estimation, Cost Models, KSLOC, SDLC, Early Phase Cost Estimation

I. INTRODUCTION

Cost estimation is the expert measurement of effort and development time required to develop a software system. The software cost estimation process includes determining the size of the software product to be produced, determining the effort required, developing preliminary project schedules, and finally, calculating the overall cost of the project. Software cost estimation requires the determination of the following estimates:

- Effort (usually in person-months)
- Project Duration (in calendar time)
- Number of persons required

After the large acceptance and success of COCOMO II model [4] of Boehm since 2000, there have not been much work done in this field for further improvement. An estimation model with 20-30 input parameters for cost drivers is not very helpful if we don't have a logical approach for specifying the input values for parameters like the software's complexity, database size, platform volatility, schedule compression, tool coverage, or personnel experience. After many years of COCOMO II an innovative approach came was of Wilson Rosa et al "Simple Empirical Software Effort Estimation Model" , 2014 [17] . His model overcome the problem of having too many input parameters as in COCOMO II.

In this research investigation, we proposed a software development effort estimation model on expert learning system with few input parameters. Cost estimation must be accurate because it is very helpful in determining what resources to commit to the project and how well these resources will be used. Customers expect that actual project development cost to be in line with estimated costs, this gives better customer satisfaction.

II. BACKGROUND

Various cost estimation models are available for the estimation of effort. Here we provide the details of different software cost estimation models, mainly focusing on size estimation and expert learning systems.

A. Early Phase Software Cost Estimation Model

1) *COCOMO-II*: The Early phase cost estimation model COCOMO-II uses thousand source lines of code (KSLOC) or unadjusted function points (UFP) for the estimation of size. UFPs can be changed to the equivalent SLOC and then to KSLOC to estimate the size of the software. The use of exponential scale factors is similar for Post-Architecture and the Early Design models. A reduced set of multiple cost drivers is used in the Early Phase Design model as shown in Table I. The Early Design cost drivers are accomplished by integrating the Post-Architecture model cost drivers. The value of the cost drivers is calculated and whenever it lies in the midway of the rating provided,roundup this to the nominal rating.Example: If the rating of estimated cost drivers value lies between Very High and High then select High. The effort equation is except that the number of effort multipliers is reduced to 7 (n = 7).

TABLE I
EARLY DESIGN AND POST-ARCHITECTURE EFFORT MULTIPLIERS [8]

Early Design Cost Driver	Counterpart Combined Post-Architecture Cost Drivers
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

Overall Approach: The reduced Early Design model cost driver maps all the Post-Architecture cost drivers as shown in TABLE I. The details of the cost drivers are as follows:

978-1-5090-0669-4/16/$31.00 ©2016 IEEE

- Personnel Capability (PERS) includes Analyst Capability (ACAP), Programmer's Capability (PCAP), and Personnel Continuity (PCON).
- Product Reliability and Complexity (RCPX) includes Required Software Reliability (RELY), Database size (DATA), Product Complexity (CPLX), and Documentation match to life-cycle needs (DOCU).
- Required Reuse (RUSE) is same in both the phase of software development.
- Platform Difficulty (PDIF) includes Execution Time (TIME), Main Storage Constraint (STOR), and Platform Volatility (PVOL).
- Personnel Experience (PREX) includes Application Experience (AEXP), Platform Experience (PEXP), and Language and Tool Experience (LTEX).
- Facilities (FCIL) include use of Software Tools (TOOL) and Multisite Development (SITE).
- Schedule (SCED) is same in both the phase of software development.

The reduced cost drivers include the combination and use of numerical equivalent rating level values of complete cost drivers. Numerical values of Post-Architecture cost driver rating is 1 for Very Low, 2 for Low, 3 for Nominal, 4 for High, 5 for Very High and 6 for Extra High. The Early Design model cost drivers rating scale is from Extra Low to Extra High and the values of each cost driver is computed by the summation of values of combined Post-Architecture cost drivers value . If the contributing Post-Architecture cost driver has Nominal scale than the corresponding Early Design model rating is also Nominal. Effort will be calculated after analysis of each cost drivers value with the following given equation:

$$PM_{Estimated} = 3.67 \times (Size)^{(SF)_i} \times [\prod EM_i]$$

Where PM gives effort in persons-month, size in terms of KSLOC of the software, SF is the scaling factor and EM is the effort multipliers.

Mainstream parametric cost models use source lines of code (SLOC) as a measurement for predicting software effort. The main reason of using SLOC is that it allows practitioners to determine or at least estimate what parts of the system they will actually be developing. In contrast, function-point or use-case based size estimates are made without determining which functions are going to be provided by Commercial-Off-The-Shelf products, cloud services, or other non-developed items, causing serious overestimates. However controversy exists over whether or not SLOC is a good indicator, continuous use of this metric generates meaningful statistical results.

2) *Empirical Software Effort Estimation*

Estimation can be done basically in three stages of software development , at the start of the project, at early design phase and just before completion of the project. Among three, early design cost estimation is done with a popular model COCOMO-II and with other proposed model such as Simple Empirical Software Effort Estimation Model.

In the Early design phase of the software development, the software project manager knows very basic details about the project like the size of the project to be produced, the type of manpower involved in developing the product, the type of methodology used for the development of the product, the nature of the development platform used, etc. The early phase COCOMO-II model is used at this stage for the estimation of cost and can be employed either in System Integration, Application Generator, or the Infrastructure development sectors. The model uses 7 cost drivers, 5 scaling factors and product size for the estimation of effort and development time required for the development of software products.

Another model for early effort estimation is *Simple Empirical Software Effort Estimation Model* [17]. This model uses the size of the product to be produced and its application type to forecast effort. Product size is measured in terms of the equivalent source lines of code. Application type is defined in terms of different type of applications that can be developed shown in Table II.

$$PM = (2.047 \times KESLOC0.9288) \times (2.209^{D1}) \times (1.917^{D2}) \times (3.068^{D3}) \times (3.072^{D4}) \times (3.434^{D5}) \times (4.521^{D6}) \times (4.801^{D7}) \times (4.935^{D8}) \times (5.903^{D9}) \times (7.434^{D10}) \times (10.72^{D11})$$

Where:

PM = Engineering Effort for application type in Person Months

KESLOC = Product size in thousand Equivalent Source Lines of Code

D1 to D11 defines the application type. The values of each from D1 to D11 are 0 or 1 depending on the type of application to be developed. Table 2.2 shows the application type for all D1 to D11.

B. Size Estimation

The definition of the size can be defined as "the amount of work done into program development". It is very important for good estimation model to have an accurate size estimation. Normally, applications will be developed by writing new code segment, from the reused code taken from various sources with or without modification, or by automatic code generator tools. Adjustment factors capture the quantity of design, code and testing that are altered. It also takes into account the code understandability and the programmer familiarity with the code. It can be done in two ways: (i) Using function point analysis and (ii) Using a source line of code. Both the ways of estimating software size are explained in the later section of this work.

TABLE II
APPLICATION TYPE TAXONOMY [17]

Application Type	Symbol	SEER-SEM Application Domain(s)
Test	TST (D6)	Diagnostics, Testing Software
Software Tools	TUL (D1)	Business Analysis Tool, CAD, Software Development Tools
Intelligence & Information Systems	IIS (D2)	Database, Data Mining, Data Warehousing, Financial Transactions, GUI, MIS, Multimedia, Relational/Object-Oriented Database, Transaction Processing, Internet Server Applet, *Report Generation,* Office Automation
Mission Planning	PLN (D1)	Mission Planning & Analysis
Mission Processing	MP (D8)	Command/Control
Real Time Embedded	RTE (D7)	Embedded Electronics/Appliance, GUI (cockpit displays), Robotics
Scientific Systems	SCI (D3)	Expert System, Math & Complex Algorithms, Simulation, Graphics
Sensor Control and Signal Processing	SCP (D11)	Radar, Signal Processing
System Software	SYS (D4)	Device Driver, System & Device Utilities, Operating System
Telecommunications	TEL (D5)	Communications, Message Switching
Vehicle Control	VC (D9)	Flight Systems (Controls), Executive
Vehicle Payload	VP (D10)	Flight Systems (Payload)

C. *Expert Learning System*

Computers require intelligence to solve problems, intelligence requires knowledge for the accurate prediction, and finally knowledge requires learning throughout time for updated information. So we can say that learning is the central part of providing intelligence to the system. Expert Machine learning performs this purpose well. Machine learning considers being a system capable of gaining, grasping and integrating the knowledge automatically. The ability of the system to learn from expert person's knowledge in terms of feedback systems, through experience, analytical observation, training, and other means of learning results in intelligent system consistently self-improve through learning and thereby acquire efficiency and effectiveness. A machine learning system usually begins with very little knowledge in hand so that it can analyze, understand and then test the knowledge that it acquires with the time duration and further improve the system accuracy and performance.

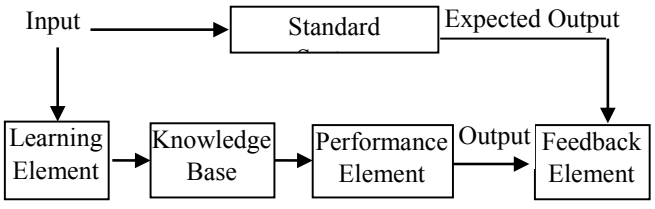

Fig. 1 Expert Learning System Model

The Fig.1 shown above is a typical expert learning system model. It basically consists of five components with input and expected output. The description of each component is as follows:

Learning element: It receives input given by an expert person or through other mediums like books, journals, electronic media processing, etc., and processes it for providing more accuracy to the standard system.

Knowledge base: It is the collection of data from the learning element output. Initially, it contains very basic knowledge and once learning starts it upgrades the database. As the collection of data increases, system efficiency and accuracy also increases.

Performance element: It uses the data contained in the knowledge database and performs functions on it like calculations, predictions, etc. and gives the desired resultant output for further processing.

Feedback element: It is receiving two inputs, first from learning element and second from standard system. The purpose is to find out the differences between the two input values. This process is used to find out what shall be done to produce the desired output.

Standard system: It is the main system from which input, output operations are performed. It is an experienced, trained person or an intelligent computer program that produces the desired output.

The series of operations described above can be repeated till the time the system gains desired efficiency and accuracy.

III. LITERATURE SURVEY

Software Cost Estimation (SCE) is a process of forecasting of efforts and cost in terms of cost, schedule and personnel for any software system. Software cost estimation is a method which is as old as the computer industry itself and it has been developed many times until function points were formulated by Albrecht in 1979. Nowadays software cost estimation is becoming a complex branch of computer science, therefore many sizing techniques, sizing metrics, cost and effort models appeared which may not exist in this volume in the rest of computer sciences, the author Abedallah Zaid et.al, [12] shows the common techniques used in SCE and along with it, it highlights the very important trends in this field also. He described the most urgent topics to be investigated and the challenges in SCE.

Software project planning is one of the most important tasks in software project development. Poor planning of software development often leads to many problems in the long term of project use. Project errors & unwanted and unrealistic outputs are some common problems often occur in front of the project team. Nowadays software project managers should be aware of the increasing cases of project failures. The reason behind it is imprecision of the cost estimation. The author Vahid Khatibi et.al, [13] shows several existing methods for software cost estimation and demonstrated their aspects. Comparing the characteristics of the methods it can be applied for ability based clustering; it is also helpful in selecting the special method specific for each

project. The author also gives a complete case study of estimation in an actual software project.

Software cost estimation is a critical factor in project management. If we fail to use right software cost estimation method, it might become a reason for the project failures. According to Report found by author Ali Bou Nassif et.al. [14], approximate 65% of projects are delivered to client over budget or post-delivery deadline. Performing SCE in the early phases of the SDLC is crucial and this would be helpful to project development managers to bid on projects. The authors proposed a novel method to determine a software effort estimation using a cascade correlation neural network approach on Use Case Diagrams. They evaluated results based on the criteria of the MMER (Mean Magnitude of Error Relative to the estimate) and PRED (Prediction Level). They use 214 industrial projects and 26 educational projects to test the results on Use Case Point model and multiple linear regression model. The author concludes that the proposed model can be positively used with acceptable results as an alternative approach to calculate software effort as an early design phase of software development.

Most of the estimation models require details of different cost drivers that will be available at the later stage of the development process. The author Tharwon Arnuphaptrairong [15] proposes to use Function Point Analysis in application with a dataflow diagram to solve the timing critical problem. The proposed methodology by the author was validated through the graduate students' software. Although the results got by author were disappointed, but some interesting insights are worth looking at the model.

The author Wilson Rosa et.al. [17] explained that an SCE method with so many parameters that cannot be defined logically for input is not useful at an early conceptual stage. Author gives a simple approach for forecasting software development effort at an early stage of project development. The regression model is used along with product size and application types to calculate effort in this approach. Product size is calculated in terms of the equivalent source lines of code. The author gathered and then analyzed empirical data from 317 very recent projects implemented within the US Department of Defense over the tenure of 9 years started in 2004. The equation is easier and more relevant to use for early cost estimation than traditional parametric cost models. The Statistical results explained that source lines of code and application type, both are important contributors to the development effort.

IV. Problem Identification

The estimation of effort can be done through COCOMO-II, but it uses many cost drivers for effort estimation and at early design phase it is difficult to define them logically. Another model, i.e., Simple Empirical Software Effort Estimation Model uses only application type for the estimation of effort hence it lacks with other important cost drivers that are necessary to consider for the calculation of effort. There is no such method exist that can improve the system's efficiency and accuracy and keep it updated with the current scenario. Hence a system is needed with a feature of learning capability.

Generalize Issues in Cost Estimation

- Defensible estimates are needed at the early conceptual phase of a software-intensive system's definition and development.
- A model with 20-30 parameters is not very helpful if you don't have a defensible approach for specifying the inputs.
- As with the changing technology and increasing variety of application development, such a system is needed that can cope up with the changing environment and also improves itself with time through learning.

Problem Definition

A model is required for the estimation of effort that will take minimum cost drivers and able to predict more accurate estimation and also the minimum cost drivers used by the proposed model should be logically defined at early design phase of software development.

V. Proposed Solution

Early stage cost estimation is an important step of the software development. Currently, the approaches available for early cost estimation take 10-20 inputs that are not possible to find and to define logically at an early stage. Our proposed work divided into two main sections. The first section identifies the issues related to early phase effort estimation and gives approach for more accurate estimation. Second section uses an expert machine learning approach that enhances the accuracy of the estimation with time without being explicitly programmed. The key features of our proposed approach are:

- To develop an estimation model for early phase of software development that can estimate cost with minimum cost drivers or parameters.
- Provide more accurate estimation, than the available models currently for software cost estimation.
- The size of the software project is measured in terms of Equivalent Source Lines of Code. Also, in the estimation of SLOC, the concept of reusability is also considered.
- Effort is represented in Person Months.
- Use of expert machine learning for calculating weight factor for achieving results that are more practical.

The proposed architecture shown in Fig. 2 consists of two different modules i.e. effort estimation and machine learning

using previous cost estimation performed on projects. It estimates the software cost with minimum inputs that a software manager selects or enters. Estimation of software starts with the selection of the input size in terms of KLOC in three different parameters i.e. expected, worst and best. Also, consider the reusability of the source code and calculate the EKLOC based on that. After that project manager has to select an application type that describes which type of application is being developed. Also he has to enter experience related to the domain of the application and lastly has to enter experience

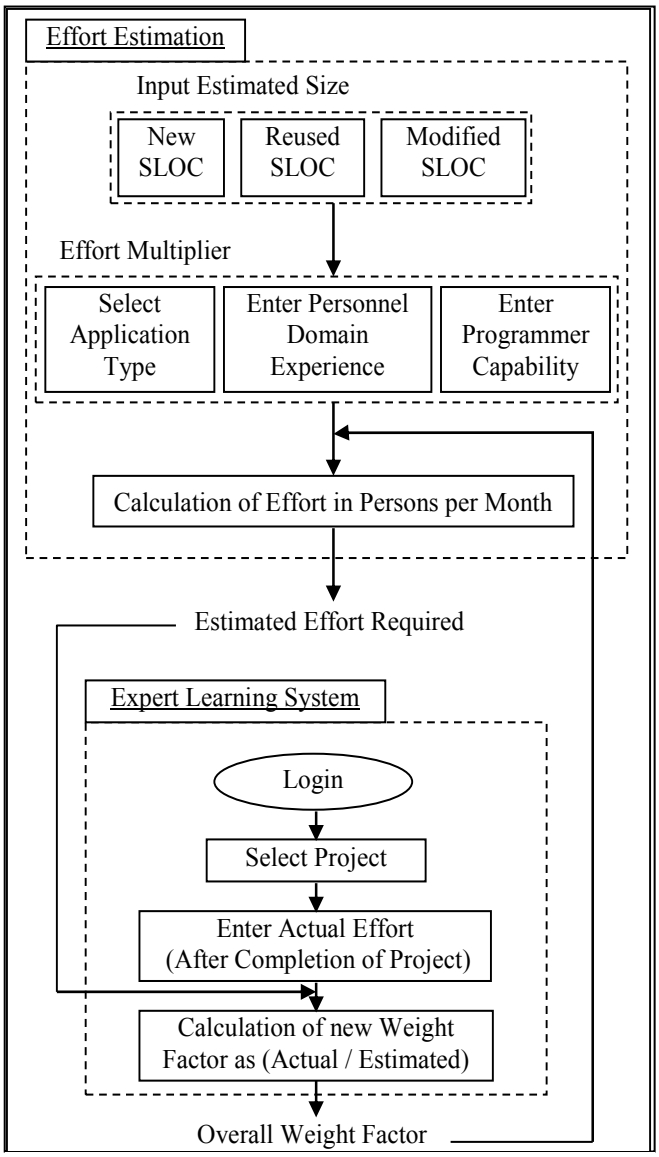

Fig.2 Proposed Architecture of Early Phase Software Effort Estimation Model

related to capability of doing the same type of project with consideration of related hardware and software used. With all these input given by the project manager, calculation of effort is done through this software and get stored in the knowledge base (database).

As the above process is repeated using this software, many effort values will be calculated and stored for many different types of software projects. Afterwards, when the actual cost of these projects were found after the completion of the project, that actual cost and estimated cost, which is stored in the database, are used to calculate the weight adjustment factor (w). This weight adjustment factor is then stored in the knowledge base and extracted for use when similar kind of project arrives in for effort calculation. When all the parameters, i.e. Programmers capability, personnel experience, application type and KSLOC are matched for two projects, then the project, which has previously calculated effort and adjustment factor is used in a new project to improve the accuracy of cost calculation.

Component Description: Our proposed architecture contains two major components by which software estimation can be done. They are Effort Estimation and Expert Learning System. The explanation of both the components are as follows:

A. Effort Estimation

Effort estimation can be further divided into three main subparts size estimation, effort multiplier selection and effort calculation.

1) *Size Estimation (KLOC)*: The actual size of estimated KSLOC breaks into estimation of three subparts as new KSLOC, Reused KSLOC and Modified KSLOC. Factors that affect the source line of code are breakage percentage (Breakage reflects the requirements volatility in a project. It is the percentage of code thrown away due to requirements volatility), reusability (design modified, code modified, and integration required for modified software), adaptability, Programmer unfamiliarity, understandability. The formula for the calculation of each type of KSLOC is given below:

$$Size = (1+ (REVL / 100)) \times (KSLOC_{New} + KSLOC_{Equivalent}) \quad \ldots \text{Eq. (1)}$$

Where,

$$KSLOC_{Equivalent} = KSLOC_{Adapted} \times (1 - (AT / 100)) \times AAM$$

$$AAM = \begin{cases} [AA + AAF (1 + (0.02 \times SU \times UNFM))] / 100, & \text{if } AAF \leq 50\% \\ [AA + AAF + (SU \times UNFM)]/100, & \text{if } AAF > 50\% \end{cases}$$

$$AAF = (0.4 \times DM) + (0.3 \times CM) + (0.3 \times IM)$$

Where:

Size = Thousands of Source Lines of Code for Software
REVL = Percentage of code modified during breakage
$KSLOC_{New}$ = Thousands of New Source Lines of Code
$KSLOC_{Equivalent}$ = Thousands of Equivalent Source Lines of Code
$KSLOC_{Adapted}$ = Thousands of Adapted Source Lines of Code
AT = Percentage of code that automatically translated
AAM = Adaptation Adjustment Multiplier
AA = Percentage of reuse effort due to assessment and assimilation
AAF = Adaptation Adjustment Factor

SU = Percentage of reuse effort due to software understanding
UNFM = Programmer Unfamiliarity
DM = Percentage of design modified during reuse
CM = Percentage of code modified during reuse
IM = Percentage of integration and test modified during reuse

2) *Selection of Effort Multipliers (Cost Drivers):* For the early cost estimation with minimum parameters, we have selected three cost drivers as:

a. *Application Type:* Based on application type the value of effort adjustment factor for application type is calculated using the given equation below:

$$EAF_{AppType} = (2.209^{D1}) \times (1.917^{D2}) \times (3.068^{D3}) \times (3.072^{D4}) \times (3.434^{D5}) \times (4.521^{D6}) \times (4.801^{D7}) \times (4.935^{D8}) \times (5.903^{D9}) \times (7.434^{D10}) \times (10.72^{D11}) \quad \ldots \text{Eq. (2)}$$

Where:
EAFAppType = Engineering Effort for application type in Person Months. Application type can be referenced from TABLE II.
KESLOC = Product size in thousand Equivalent Source Lines of Code
D1 = 1 if PLN, 0 if TUL or other application type
D2 = 1 if IIS, 0 if other application type
D3 = 1 if SCI, 0 if other application type
D4 = 1 if SYS, 0 if other application type
D5 = 1 if TEL, 0 if other application type
D6 = 1 if TST, 0 if other application type
D7 = 1 if RTE, 0 if other application type
D8 = 1 if MP, 0 if other application type
D9 = 1 if VC, 0 if other application type
D10 = 1 if VP, 0 if other application type
D11 = 1 if SCP, 0 if other application type

b. *Personnel Experience (PREX):* The Early Design phase PREX cost driver combines three cost drivers of Post-Architecture phase that are Application Experience (AEXP), Platform Experience (PEXP) and Language and Tool Experience (LTEX). Each of these cost drivers has a rating from Very Low (=1) to Very High (=5). To get the rating scale for PREX, values of all the cost drivers are added that gives a range between 3 to 15. These values are put on a scale and rating is assigned to the PREX cost driver with an associated effort multiplier as shown in Table III below.

TABLE III
EAF FOR PERSONNEL EXPERIENCE

	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Sum of AEXP, PEXP, and LTEX ratings	3,4	5,6	7,8	9	10, 11	12, 13	14,15
Applications, Platform, Language and Tool Experience	≤ 3 Months	5 Months	9 Months	1 Years	2 Years	4 Years	6 Years
Effort Multipliers	1.59	1.33	1.22	1.00	0.87	0.74	0.62

c. *Programmers Capability (PERS):* The Early Design phase PERS cost driver combines three cost drivers of Post-Architecture phase i.e., Analyst Capability (ACAP), Programmer's Capability (PCAP) and Personnel Continuity (PCON). Each of these cost drivers have a rating from Very Low (=1) to Very High (=5). To get the rating scale for PERS, values of all the cost drivers are added that gives a range between 3 to 15. These values are put on a scale and rating is assigned to the PERS cost driver with an associated effort multiplier as shown in Table IV below.

TABLE IV
EAF FOR PROGRAMMERS CAPABILITY

	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Sum of ACAP, PCAP, PCON Ratings	3,4	5,6	7,8	9	10, 11	12, 13	14,15
Combined ACAP and PCAP Percentile	20%	35%	45%	55%	65%	75%	85%
Annual Personnel Turnover	45%	30%	20%	12%	9%	5%	4%
Effort Multipliers	2.12	1.62	1.26	1.00	0.83	0.63	0.50

Table Interpretation: If the rating of ACAP, PCAP and PCON is Nominal i.e., The sum of their corresponding values is 9 (3+3+3) then the rating of PERS also Nominal and its respective effort multiplier is 1.0

3) Calculation of Effort: Final effort estimation can be made from the equation (3). This equation gives values of effort in person month. From the PM we can calculate the development time for the software being developed using COCOMO-II early phase equation.

$$PM = (2.047 \times KESLOC^{0.9288}) \times EAF \quad \ldots \text{Eq. (3)}$$

Where,
$EAF = EAF_{AppType} \times EAF_{PerExp} \times EAF_{ProCap}$

A. Expert Learning System

This system has four main sub components defined as login credentials, project selection, actual effort and calculation of the new weight factor. They are listed below:

1) Login Credentials: It authenticates the user login credentials to avoid unwanted use of the learning system. Only the authenticated user of the system can use it for improving accuracy of the estimation.

2) Project Selection: In the expert learning process, the user has to select projects for which learning has to be done. By the selection of the project, its pre-estimated effort value is selected by the tool and that value is used in the further learning process.

3) Actual Effort: In this step, the project manager first calculates the actual effort for the project selected above after completion of the project. Once calculated, the actual effort is entered in the tool for the estimation of new weight factor.

4) Calculation of new Weight Factor: The last step of the learning system calculates the new weight factor for the system using actual effort and estimated effort value. Finally, the average weighted mean of all the previously estimated weights is taken to finalize the new weight factor. This average weight value is used for further estimation of effort for all upcoming new projects.

VI. Implementation and Result Analysis

To verify our proposed work, we have identified some case studies. For each case study, we collected the project requirements. Given below are the project requirements for five case studies among them:

i. *Early Cost Estimation (Id=1)*: This project aimed to develop a web based tool for the estimation of software development effort based on size estimation, effort multiplier estimation and calculation of effort based on predefined equation. This project also adds the feature of expert learning system for further improvement in the accuracy of the estimation.

ii. *Human Disease Diagnosis System (Id=2)*: This project aimed to develop web based software for the diagnosis of human diseases. It takes input as several symptoms and based fuzzy logic, it identifies the disease and shows the percentage of disease a person can have.

iii. *Matrimonial Website (Id=3)*: This project aimed to develop a web based application for the community based marriage portal. It has features like registration, matching likes qualities, send messages and chatting for registered users.

iv. *Payroll Management (Id=4)*: This project aimed to develop an HRMS system handling salary of the employee of a particular organization. This project has features like calculation of attendance, holidays and TA, DA, etc. and calculate the salary of the employee and generate report based on certain criteria.

v. *Inventory Management System (Id=5)*: This project has listed requirements to develop a desktop based application for Shopping Malls or General Stores with features like order management, purchase item, sales of packed and unpacked items, account management and reporting based on different criteria.

For the experimental purpose, we estimated the effort for the above five case studies based on existing early software effort estimation approach and also using our proposed approach for early phase software estimation. For both types of estimation, our tool estimates KLOC based on the requirements collected for each case study.

TABLE V
ESTIMATED EAF AND EFFORT VALUES FOR EACH PROJECT

Pro. Id	Project Name	KLOC	EAF		Effort	
			Base	Proposed	Base	Proposed
1	Early Cost Estimation	19	1.91	2.33	62.43	75.73
2	Human Disease Diagnosis System	39	1.91	1.59	121.77	101.73
3	Matrimonial Website	78	1.91	1.38	241.80	176.18
4	Payroll Management	98	1.91	1.38	289.84	212.74
5	Inventory Management	108	1.91	1.59	313.96	262.34

Table V shows the estimated value of KLOC, estimated EAF and Effort value for the existing base approach and for our proposed approach. In the table, first column shows the project Ids of all five case studies from 1 to 5, the second column shows the names of five projects respectively. Based on the requirements gathered third column shows the value of KLOC required developing the software project. The fourth column shows the estimated EAF values for the existing and proposed work and based on the estimated KLOC and EAF value obtained, the fifth column shows the estimated Effort values required for the development of the respective project for each case study taken from project Ids 1 to 5 for both the existing and our proposed work.

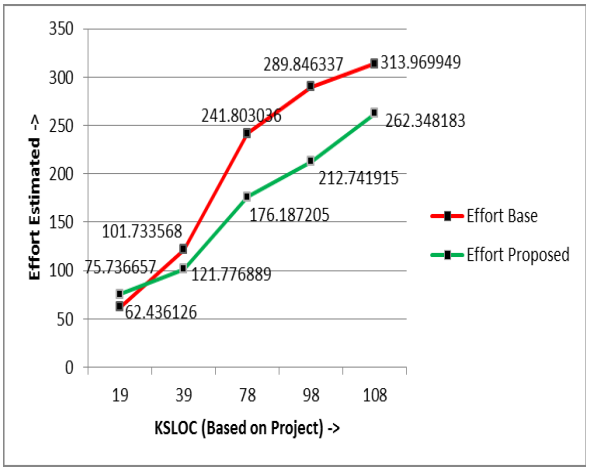

Fig. 3 Effort comparison based on KLOC

The Fig. 3 shows the comparison of estimated effort for all case studies covered in Table V using both existing and proposed approach. In the comparative graph, KLOC shown on X-axis and estimated effort for each project shown on the Y-axis.

The above graph analyzed and based on the results observed, we identified that effort estimation for software development using our proposed strategy for estimating effort gives more accurate results as compare to existing early phase effort estimation approach.

VII. CONCLUSIONS

This study introduced a simple effort estimation model for predicting early phase software development projects with self-improving feature using machine learning mechanism. Our study shows that SLOC and application type are two important factors in predicting effort. For more accurate estimation we include other cost drivers like personnel capability and experience with related domain while machine learning capability will improve efficiency further. As a result, we observe that with our proposed work more accurate estimation of a software product can be achieved with minimum parameters as input in terms of effort. Although the proposed model for early cost estimation is not highly precise, it has the advantage of providing information on its relative accuracy. The model may also be applicable to all kinds of software projects in any sector. Future work will also examine the impact of other cost drivers such as process maturity and requirements volatility for further improvement of this model.

REFERENCES

[1] Boehm, Barry W., "A Technical Book on Software Engineering Economics." in Prentice Hall, 1981.

[2] Boehm B., "Software Engineering Economics" in Englewood Cliffs, NJ, Prentice---Hall, 1981

[3] Boehm B., Abts C., Brown W., Chulani S., Clark B., Horowitz E., Madachy R., Reifer D., Steece B., "Software Cost Estimation with COCOMO II", in Prentice---Hall, 2000

[4] Boehm, B., "Safe and simple software cost analysis," in Software, IEEE, 17(5), pp. 14–17, 2000.

[5] Clark, B., Devnani-Chulani, S., and Boehm, B., "Calibrating the COCOMO II Post-Architecture model," in Proc. Int'l Conf. Software Eng. (ICSE '98), pp. 477–480, 1998.

[6] Boehm, B., 2001 COCOMO Website: http://sunset.usc.edu/research/COCOMOII/cocomo_main.html

[7] Boehm, B., 2001 COCOMO Website: http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/help.html

[8] Boehm, B., "COCOMO II Model Definition Manual" in University of Southern California, 1999.

[9] Hareton Leung and Zhang Fan, "Software Cost Estimation", in The Hong Kong Polytechnic University, 1999

[10] Dr. N. Balaji, N. Shivakumar & V. Vignaraj Ananth, "Software Cost Estimation using Function Point with Non-Algorithmic Approach", in Global Journal of Computer Science and Technology Software & Data Engineering, Vol. 13 Issue 8, 2013.

[11] Ian Sommerville, "Software Cost Estimation", in Software Engineering Book, 6th edition, 2000

[12] Abedallah Zaid, Mohd Hasan Selamat, Abdual Azim Abd Ghani, Rodziah Atan and Tieng Wei Koh, "Issues in Software Cost Estimation", in International Journal of Computer Science and Network Security, 350 VOL.8 No.11, 2008.

[13] Vahid Khatibi, Dayang N. A. Jawawi, "Software Cost Estimation Methods: A Review", in Journal of Emerging Trends in Computing and Information Sciences, Vol 2, 2011.

[14] Ali Bou Nassif, Luiz Fernando Capretz and Danny Ho, "Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade Correlation Neural Network Model", in International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2012

[15] Tharwon Arnuphaptrairong, "Early Stage Software Effort Estimation Using Function Point Analysis: Empirical Evidence", in IMECS Vol-2, 2013

[16] Brad Clark, Wilson Rosa, Barry Boehm and Ray Madachy, "Simple Empirical Software Effort Estimation Models", in 29th International Forum on COCOMO and Systems/Software Cost Modeling, 2014

[17] Wilson Rosa, Ray Madachy, Barry Boehm and Brad Clark, "Simple Empirical Software Effort Estimation Model", in ESEM'14, ACM, 2014.