Best Vision is Insight: Significance of Cost Estimations in Software Project Management for Growth and Sustainability of IT Organization.

Kartiki Bhamre, Dr. Amol Goje
Research Student, Research Guide
VIIT Baramati, Savitribai Phule Pune University, Pune India

Abstract

The key aspects of a software project management are to deliver the project on time and in budget regardless of requirement, scope and time changes. Cost estimation has always been a difficult task in software development. Software Estimation practices offer a mechanism for analyzing effort, cost and time along with easy to handle ever-changing business environment and requirements. An entire organization, especially management comes across numerous issues while executing the software projects with changing requirements, the rise in new services/demands, tight budget plans and rapid turnaround demands. This study determines the impact of 'Software Cost Estimation' techniques on software development industry by focusing on different types of methodologies being used in IT industry. The study reveals that software cost estimation facilitates IT organization to grow and sustain in today's competitive era.

Index Term

Software Project Management, Software Cost Estimation, Algorithmic Models, Non-Algorithmic Methodologies

INTRODUCTION

Precision in software cost estimation has a direct impact on the organization's growth in multiple dimensions. It also affects IT industries decision making on business expansion, research in advance sciences and social activities. Eventually, this visualization becomes the motive of IT industries growth and sustainability. The effective software cost estimation can reduce the redundant costs and amplify the productivity and efficiency of the organization. In recent years, many software cost estimation techniques have been developed to deal with the multiple challenges faced by the IT industry. Following this development, there is still a scale of enhancement in Software Cost Estimation practices. In today's scenario software cost estimation is one of the most complex areas in project management. Fundamental parameters of effective estimations are based on several resources and schedules essential for software development. The software estimation process includes estimating the size of the software product or project to be produced, the effort required for completing specified work, preparing the early project plan, and ultimately, estimating the entire cost of the project. Certainly, organizations are taking efforts to do more with limited assets, with less finance, and, in many cases with a reduced workforce. It is expected that performance and outcome should be maintained always, but most of the time project runs over budget or crosses the time limits. Hence it is essential to study, research and practice efficient software cost estimation techniques to accomplish successful project delivery and subsequent activities.
Primarily there are ranges of software cost estimation approaches which are the combinations of software cost estimation methodologies, models, techniques, and processes. These are divided into two basic types as,

1. Algorithmic methodologies/models
2. Non-algorithmic methodologies

According to the analysis, no one method or model is inevitably superior or inferior to the other. Indeed, their strengths and flaws are admiring to each other. According to experiences, it has been suggested that a combination of models and analogy or expert judgment estimation methods are beneficial to get consistent, precise cost estimation for software development.

Software Cost Estimation is a vital part of Software Engineering; it has potentials to turn out accurate estimates which have an impact on key economic processes, as budgeting, bid proposal and deciding the implementation boundaries of the project. The work in this article explores the association among diversified dimensions of software projects as, entire project scope, project size, effort, time and effort influencing factors. The study aims at Software Cost Estimations are driving the growth and helping it to move the next level.

II. SOFTWARE PROJECT MANAGEMENT

2.1 Background

Software Project Management encompasses a set of activities that are executed during the phases of a typical software development project. The basic elements of Project Management are Project itself, People working on projects, the Process used to produce the project and final Product delivered to the customer.

A Software Project is the process of software development from requirement gathering to testing and maintenance, it implemented according to the execution methodologies, in a specified period to achieve the intended software product.

The execution pattern of an IT company can be seen in two parts Software Development and Software Project Management. Software Project Management is a well-defined task, which is a compilation of numerous courses of actions done in order to attain an objective.

2.2 Software Project Management Process Groups

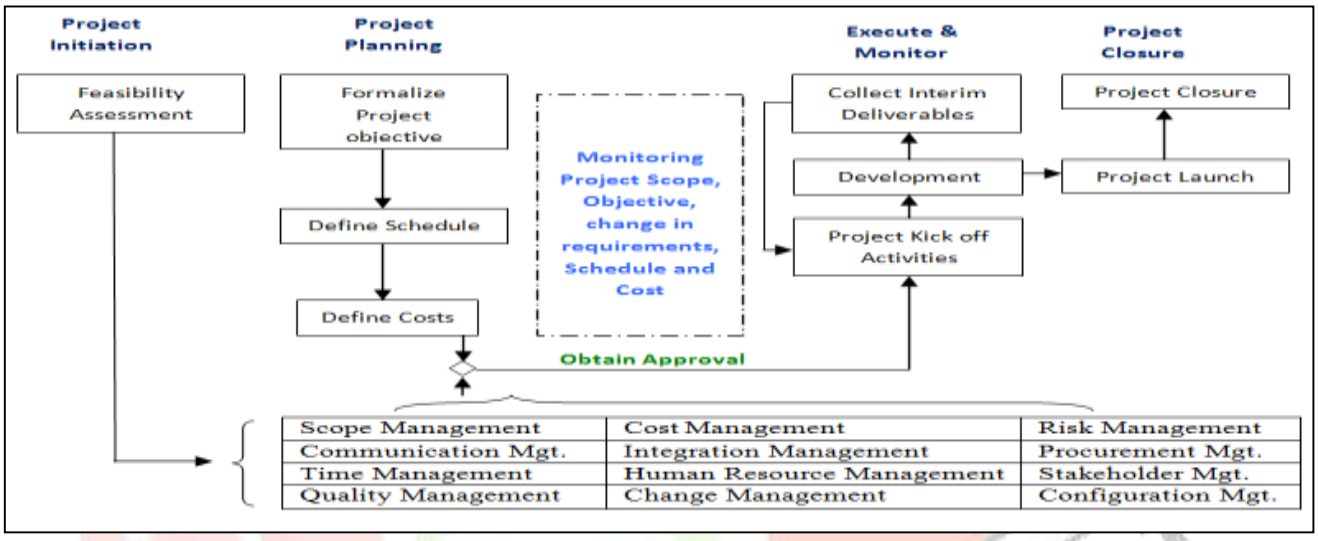

Figure 1: Software Project Management Process Group

The researcher provides a view on each of these areas which helps to accomplish successful Software project management. Figure 1 explains these four process groups are connected with the outcome they produce. An outcome of one process becomes an input to another. Project Initiation helps to assess requirements. It sets the vision of what is to be accomplished. At this stage the project is formally authorized by the sponsor, initial scope is defined, and stakeholders are identified. Stakeholder identification is very important here because correct identification of stakeholders can factually make or break the project. This process group performs the tasks so that projects and programs are approved by a sponsoring entity, and hence projects are aligned with the strategic objectives of the organization.

The researcher explores the various aspects of Software Project Planning. The aim of this process group is to ensure that various project tasks are well coordinated, and they meet the project objectives including timely completion of the project. Generally, project planning is considered to be a process of estimating, scheduling and assigning the tasks to the resources in order to deliver an end product of suitable quality. Project planning process group includes scope definition and scope planning, project activity definition and activity sequencing, time, effort and resource estimation, risk factor identification, cost estimation and budgeting, schedule development, selecting technology, databases and coding development plan, quality planning, risk management planning, project plan development and execution, performance reporting, planning change management, project rollout planning.

The purpose of Project Monitoring, Execution and Control is to develop the product or service that the project was commissioned to deliver. Typically, execution and control are the longest phase of the project management lifecycle, where large numbers of resources are applied. There are three main stages as; kicking activities off, development of a project as working on programming language, databases, securities, reporting and various technologies as per project requirement and collecting the earlier activities output and implementing work in iterations if required. Project Execution and monitor involves tracking, reviewing, regulating project progress, monitoring status report, progress measurement and forecasting. Monitoring helps to generate reports on scope, schedule, cost, resources, quality, and risks. It formalizes acceptance of deliverables and records quality control results. It implements risk treatment plans and related actions. The end of the phase arrives when the product of the project is developed, tested, accepted, implemented in iteration and ready for a transition to the performing organization.

Project closure is the last and important phase of the project life cycle and like any other aspect, it requires a process group. Project closing phase is the combination of upcoming tasks, as take assurance that work is complete. The assurance of all stakeholder agreed upon project management processes have been executed. Formal recognition, sign off, and approval of completion from the customer is the final stage.

2.2 Project Failure

Even after utilizing the software development experience of hundreds of software companies and software specialist across the globe, a large chunk of software projects still fails. Usually, the deadline date is decided before the project start which later is non-negotiable. This deadline turnout in a headlong rush to get started on the assumption, the sooner developer begins coding, the sooner he'll finish. An urgency to start coding is almost always the incorrect approach. It is important to spend time to create a good algorithm, block diagrams, and design. The absence of a good design leads to continuing changes throughout the development phase. When this happens, time and budget are consumed at a rapid rate and resources get overburdened with work. Dividing and distributing task between more than one resource results in unfocused completion of work. Projects fail while initiating with a certain amount of money and making the project fit in that fixed cost. Project hampers if stakeholder fails to put up or consider the cost of an unforeseen event. The project fails if there is no relevant relation to estimates and changes in scope. Even preparing and presenting estimates in project meetings under pressure turn to project failure.

2.3 Need of Estimation in Successful Project Management

Frequently, project managers resort to estimating schedules skipping to estimate size. This may be because of the timelines set by the top management, marketing team or some time by the customers. For any reason, if this is done, at a later stage it would be difficult to estimate the schedules to accommodate the scope changes. While estimating, certain assumptions may be made. It is important to note all these assumptions in the estimation sheet, as some still does not document assumptions in estimation sheets. Even good estimates have inherent assumptions, risks, and uncertainty, and yet they are often treated as though they are accurate. To overcome certain issue Software Project Management requires efficient estimations. For the successful completion of project following points are essential to execute as,

1. Prepare effective development approach for the project
2. Develop a time schedule for the project
3. Select or prepare an appropriate template for estimating project

III. SOFTWARE COST ESTIMATION

3.1 Significance of Software Cost Estimation

The researcher observes that estimation is the process of finding guesstimate, or approximation, which is a value that can be used for some purpose. even if input data may be incomplete, uncertain, or unstable. Estimation determines how much money, effort, resources, and time it will take to build a specific system or product. Estimation is based on past data and experience, available documents/knowledge, assumptions, and identified risks. The four basic steps in software estimation are essential i.e. estimate the size of the development product; estimate the effort in person-months or person-hours; estimate schedule in calendar months, estimation include the cost of associated factors which will affect the entire life cycle of software development. Ultimately, intelligent cost estimation will deliver a product on decided time with expected quality.

3.2 Uses of Software Cost Estimation

Estimation has many purposes like a justification of the project; if this applied at early stages of the project, where stakeholder would anticipate the benefits as technical, functional knowledge gain in team members, opportunities to team members to handle different responsibilities, monetary benefits to the team and organization.

Estimation always gives directions for sharing the resources required to complete the project deliverables successfully. Generally, respective stakeholder applies some cost estimation methods to estimate the cost of software development. But the important factor is continuously re-estimating the cost and compare the targets against the actual expenditures at each key milestone. This maintains the status of project transparent and visible; this identifies required corrections in work schedule, resources, technology, and budgets.

Always at the estimation and re-estimation anchor point, iteration is a significant tool to improve estimation quality. The respective stakeholder and estimator can use several estimation techniques and check whether their estimate converges.

Different estimation methodologies and models may use different data. This results in improved knowledge-based reports for the estimation process. This helps to identify cost components that may be not available or overlooked in one of the estimating methods. It is also very important to compare the actual cost and time to the estimates even if only one or two techniques are used. It will also provide the necessary feedback to improve the estimation quality in the future.

The journey of software cost estimation began in 1960 by Frank Freiman, who build up the concept of parametric estimation models. This further goes ahead to the development of the PRICE model for hardware, in 1970. The researchers from this area analyze multiple projects using different techniques attempting to identify the factors which influence the cost of software development using correlation and regression techniques. After that, by the end of 1980, the Constructive Cost Model (COCOMO) was being formulated by Barry W. Boehm and PRICE software cost estimation parametric model was developed by the end of the same decade. Allan Albrecht and John Gaffney of IBM developed function point analysis FPA to estimate the size and effort of information systems. Further, in 1990 the key person of software cost estimation research, Barry Boehm reformulates his estimation model into COCOMO II, which consists three major sub models as Application Composition, Early Design, and Post architecture models where he used many software sizing models like Object Points, Function Points and a source line of code. While in last two decades, Software Cost Estimation moved to rapid exponential improvement in software and information technology industry. Subsequently, professionals and researchers who handle the new phenomena in software engineering such as a reusable component, object oriented environment, agile projects, component-based programming, real-time systems, etc. via new sizing techniques are inventing some new measures to measure the new existed items and improving and standardize the previous models and techniques to be applicable to the new projects and overall in Information technology and related industry.

3.2 Software Cost Estimation Methods, Models, Techniques

Eventually, from the literature review and practices observed in the industry researcher come to know that there are ranges of software cost estimation approaches. Those approaches are a combination of methodologies, models, techniques, and processes. Methodologies are merely the concept of thought based on theories, industrial observations and interpretations. Models are constructed upon those identified perceptions in a structured way. There are two types of methodologies, Algorithmic Methodologies/Models and Non-Algorithmic Methodologies.

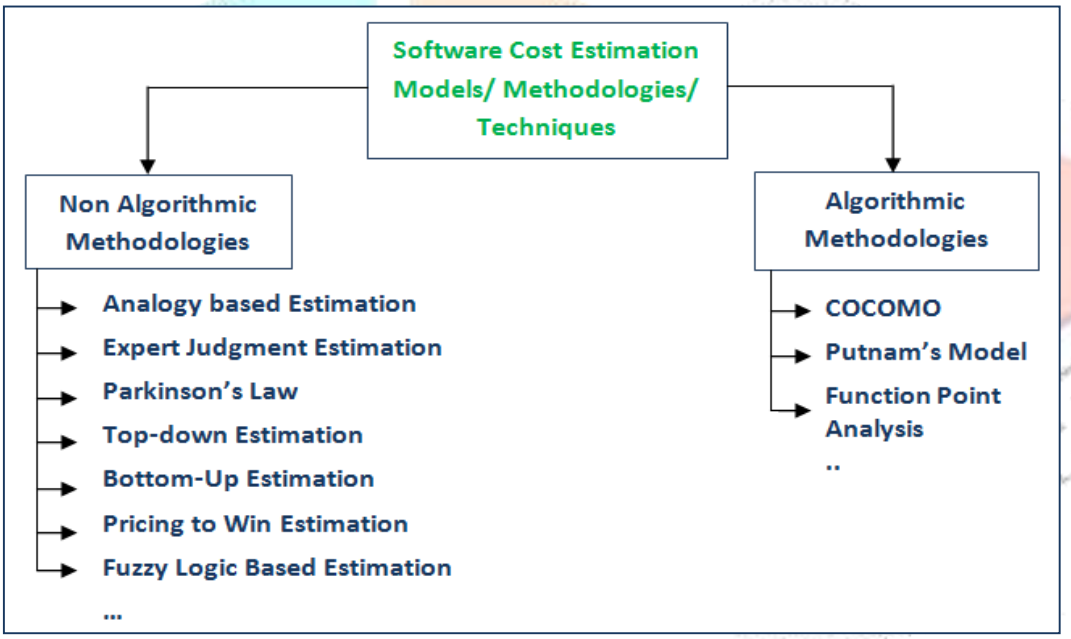

Figure 2: Types of Software Cost Estimation Methodologies

Figure 2 Segregate Software Cost Estimation Methodologies. Analogy based, Expert Judgment, Parkinson's Law, Top-down Estimation, Bottom-Up Estimation, Pricing to Win, Fuzzy Logic Based estimations comes into Non-Algorithmic methodologies. And COCOMO, Putnam's Model, Function Point analysis and COSYSMO are a type of Algorithmic Methodologies. Apart from these, there is a variety of estimation methods are available.

Eventually, after understanding the two fundamental categories of software cost estimation, researcher focuses on a range of Software Cost Estimation methodologies and models, which shows efficiency, reliability and accuracy without applying the exhaustive procedures.

The researcher has focused on various software cost estimation practices. These help to improve the correctness of software development effort estimation, size estimation, and resource estimation and ultimately cost estimation techniques. These approaches include Algorithmic and Non-Algorithmic estimation techniques, analogy based estimation, data mining techniques, rule initiation, soft computing techniques, and function point based analysis model, neural network based approaches and expert judgment based techniques.

An evolution of estimation echoes the reality of project's progress. It achieves cost or budget & controls overruns. Not an individual method is essentially superior or inferior to the other, strengths and weaknesses of each method are often admired to one other. So, bringing in and focusing on the estimation methods seems essential thing for achieving accurate and reliable estimations. In the present study majority of the early and present estimation approaches have been illustrated systematically. In Figure 2 researcher focuses on few of above-mentioned methodologies.

3.2.1 Non-Algorithmic Methodologies

Non-Algorithmic methodologies are based on analytical comparisons and conclusions. Non-algorithmic methods usually do not use any formula to calculate the software cost estimation. These methods make evaluation purely on the basis of comparison between earlier dataset and existing dataset.

3.2.1.1 Analogy-Based Estimation: The groundwork of a cost estimation practices is based on reasoning by analogy. This method requires one or more completed projects that are similar to the imminent project and derives the estimation through reasoning by analogy using the actual costs of previous projects. Estimation by analogy can be performed on the basis of either on the entire project level or at the subsystem level. The entire project level has the advantage that all cost components of the system will be considered while the subsystem level has the advantage of providing a more detailed evaluation of the resemblance and dissimilarities between the forthcoming project and the completed projects. The potency of this methodology is that the estimate is based on actual project experience. Still, it is not apparent that up to what extent the earlier project is representative of the constraints, environment and functions to be performed by the new system.

3.2.1.2 Expert Judgment Estimation: Expert judgment techniques employ discuss and take advice with software estimation experts to use their experience and understanding of the proposed project to arrive at an estimate of its size, effort, schedule, and cost. This method is accomplished by the group consensus technique. The Expert judgment method is performed by subsequent steps as coordinator presents each expert with a specification and an estimation form. Coordinating stakeholders arrange group discussions in which the experts discuss estimation concerns with the each other. Experts fill out forms secretly. Coordinators prepare and give out a review of the estimation. The coordinator calls a group meeting, with a focus on the experts discussing points where their estimates diverse extensively. Experts write out the forms, again namelessly, earlier tasks in iterations. Experts can put up project impacts caused by the advanced technology, applications, languages, architecture involved in future projects. It is not easy to quantify this method and difficult to document estimation parameters of experts.

3.2.1.3 Parkinson's Law: According to the Parkinson's principle "work expands to fill the available volume" [25] the cost is resolute or considered by the existing resources rather than based on an objective evaluation. If the software must be delivered in 10 months and 6 people are available, the effort is estimated to be 60 person-months. Even though it gives a good estimation, this method is not suggested as it may provide very unrealistic estimates. The disadvantage of this method is a system usually remains unfinished.

3.2.1.4 Pricing to Win: The software cost is estimated to be the best price to win the project. The price-to-win model estimates costs based on the customer's budget rather than internal resources or capabilities, software functionality and expansion depend on the product pricing, which in turn depends on market forces. This is a realistic approach, at the end, the net cost is what the customer pays, minus profits and any surplus scope or size perpetually get toned down to match customer budgets. If a software project was estimated at 700 person-months, it would invariably be toned down to 500 person-months if the customer could afford only that much. On the other side, this approach does not judge the options of projects incurring a loss due to scope creep or because of any other reason. This is fishy practice since it is very likely to cause a delay of delivery or force the entire team to work for extra time.

3.2.1.5 Top- Down Estimation: In Top-Down estimation overall cost estimate for the system is calculated from global properties. This is performed by using either algorithmic or non-algorithmic methods. The entire cost can be split up among the various components. This approach is appropriate for cost estimation at the early stage of project life cycle. It becomes more accurate because it gets formulated early in the project's planning stage, the budget estimate is generally based on comparable estimating, taking budget lessons learned from a similar and earlier project and applying them to the imminent project.

With this estimate, estimator starts at the top and work towards down into the project details. This estimate would include situations, a range of variation, and any assumptions that went into your computations. A Top-Down estimate is comparatively fast, but not certainly accurate. The range of variance in Top Down estimate is from -10 percent to +25 percent.

3.2.1.6 Bottom-Up Estimation: In Bottom-Up approach, every factor of the software system is independently estimated, and overall outcome combined to produce an estimate of the overall system. The requirement for this approach is that an early design of the system must be at a place which would point to how the system is decomposed into different components. This method is very accurate but takes a lot of time to create. The perfect estimate needs a work break down structure (WBS). A WBS is a deliverables-oriented decomposition of project scope. This is a very detailed method because it estimates all the components. This

method is stable because the estimation errors in the various components have a chance to balance out. Sometimes this method may be inaccurate because the essential information may not be presented at an early stage of the project.

3.2.1.7 Fuzzy Logic Based Estimation: Fuzzy Logic is used to estimate a project's size in lines in lines of code [2]. This is competent to categorize features as very small, small, medium, large, and very large. The practitioner can use historical data about how many lines of code the average very small feature need, small feature needs and so on to calculate total line of code. The estimated line of code is calculated as the product of Average Lines of code per feature and Number of features for every feature size. This method is hard to use, estimation cost of complex features is much tedious.

3.2.2 Algorithmic Methodologies/Models

Algorithmic models are based on simple arithmetic formulas using summary statistics as means and standard deviations. Few of them are based on regression models and differential equations. Algorithmic models also called parametric models. In this method, costs are analyzed using the mathematical formulas, inputs with metrics to produce an estimated output.. These methods use the mathematical equations to accomplish the application estimation. The exact equations used in historical equations or theory. To improve the accuracy of algorithmic models, there is a need to adjust or calibrate the model to the ongoing situation. These models cannot be used off-the-shelf. Even with calibration the accuracy can be quite mixed.

3.2.2.1 COCOMO

The Barry Boehm's COCOMO model is one of the frequently used estimation models. The primary version of the model delivered in 1981 and COCOMO II is currently available. The first estimation model COCOMO'81 is derived in the year 1981 from the extensive study and analysis of 63 software projects. Bohem proposed three levels of COCOMO as Basic, Intermediate, and Detailed. The basic COCOMO'81 model is a static and single-valued model which measure software development effort and cost. Program size is expressed in estimated lines of code. Next is the intermediate COCOMO'81 model which calculates software development effort as a function of program size and a set of "cost drivers" which include subjective evaluation of a product, hardware, personnel, and project attributes. Eventually, the detailed COCOMO'81 model integrates all characters of the intermediate version with an assessment of the cost driver's impact on every step of the software engineering process as analysis, design, and so on. COCOMO 81 depends on two primary equations as development effort (based on MM - man-month / Person-month / staff-month is one month of effort by one person) for the Basic Model.

$$MM=aKDSI^{b}$$

Second is effort and development time (TDEV)

$$TDEV=cMM^{d}$$

KDSI is the number of thousand delivered source instructions and it is a measure of size. The coefficients a, b, c, and d depends on the mode of the development.

There are three modes of development as Organic, Semi-Detached, and Embedded. Basic mode uses the only size in estimation. The intermediate mode also uses 15 cost drivers as well as size. In intermediate mode development effort equation becomes:

$$MM=aKDSI^{b}\mathbf{C}$$

C (effort adjustment factor) is calculated simply multiplying the values of cost drivers. So, the intermediate model is more accurate than the basic model.

The intermediate model COCOMO'81 uses the following steps to compute estimation:

a. Recognize the mode as organic, semi-detached, and embedded of development for the new product.
b. Estimate the size of the project in KDSI for obtaining a nominal effort calculation.
c. Adjust 15 cost drivers to reflect your project.
d. Calculate the project effort using first equation and the effort adjustment factor (C)
e. Calculate the project duration using second equation.

Drivers are chiefly helpful to the estimator to realize the impact of dissimilar factors that affect project costs. Ultimately, it is hard to accurately estimate KDSI early in the project, when most effort estimates are required and is not a size it's a length.

Furthermore, COCOMO 81 was not enough efficient to apply to advance technology-based and to newer software development practices. Hence it is redeveloped in the late 90s. COCOMO'II offers different effort estimating models based on the stage of development of the software project. COCOMO'II offers ranges of estimates which represent one standard deviation around the

most likely estimate. COCOMO'II offers the software reuse and reengineering where automated tools are used for translation of existing software. COCOMO'II considers for requirements instability in its estimates.

COCOMO'II has three different models as Application Composition Model, Early Design Model, and Post Architectural Model.

The early design model is an elevated model and can be used in the architectural design stage of software development to explore architectural alternatives or incremental development strategies. This model is near to the original COCOMO.

The post-architectural model is a detailed model that can be used for the actual development stage and maintenance stage. It is the most detailed version of COCOMO II. Both, the early design model and the post-architecture model uses the similar formula to estimate the effort required to develop a software project.

The application composition model used as sizing metric for application composition; application composition model is the estimation based on the number of screens, reports, and 3GL components.

$$PM = A \cdot Size^{E} \cdot \prod_{i=1}^{n} EM_i$$

where

$$E = B + 0.01 \cdot \sum_{j=1}^{5} SF_j$$

A=2.94, B=0.91, C=3.67, D=0.28

Sometimes this model is not suitable for many complex and outsized projects as a large amount of data is required.

3.2.2.2 Putnam's Model:

Putnam's Model is basically depending on two variables as time and size. Putnam's model examines software projects and estimations are calculated on the distribution of manpower. Equation (1) is used in this model where S is the size of software in LOC. Environment factor is E which represents the development capability. The effort is denoted by person per year. Apart from this one more relation was found by Putnam as mentioned in step (2). D0 represents manpower built up factor range from 8 to 27 for new and rebuilt software.

$$S = E * Effort^{1/3} * t_d^{4/3} \quad \text{(1)}$$

$$Effort = D_0 * t_d^{3}. \quad \text{(2)}$$

t_d denotes Software delivery time

D_0 represents man-power built up factor from 8 to 27 for new and rebuilt software.

Putnam's Model does not consider other aspects of software development life cycle apart from time and size.

3.2.2.3 Function Point Analysis:

For Information Technology or Software industry software design related work packages will require a function point Analysis method for performing estimations. The precondition is that estimator needs to have a lot of knowledge about the effort of the work packages of similar scope and degree of complexity, based on observation. After that, the experts accumulate this knowledge into "function curves" which can be used to estimate effort for new work packages. Function Point Analysis was developed by Albrecht [20] to measure the functionality of software projects. This method measures the size of the software and considers internal logical files [21], external interface file, and external input-output and external inquiries from a functional viewpoint metric. With the help of this method development costs can be estimated in early phases of software development. Function Point Analysis is not dependent on the language, tools, and methods of implementation. But it requires manual work which is more time-consuming. It's always difficult for a new developer to estimate the size of the software [22] as function point usage requires experience.

3.4 Selection of Cost Estimation Practices

It will be a cautious decision to finalize software cost estimation methodology for the specific project. In case of known projects and sub projects estimator can use the expert judgment method or analogy method if the resemblance of them is present, since it is fast and trustworthy of this situation;

In case of subsequently large and comparatively lesser known projects, it is always preferable to use algorithmic model. In this case, software estimation researcher suggests the estimation models which do not require SLOC as an input.

Here the researcher believes that COCOMOII is the first applicant because COCOMOII model does not use Source Lines of Code (SLOC) but also uses object points, unadjusted function points as metrics for sizing a project.

Eventually, if software estimation practitioner's uses cost estimation by parts, they may use the expert judgment methodology up to a certain extent for some known project parts. With the help of this process, practitioners can take improvement of both: the thoroughness of models and the speed of expert judgment or analogy. Since after analyzing the advantages and disadvantages of each technique are complementary, a perfect mixture will help to decrease the negative effect of any one technique and expand their individual potency and help to cross-check one method against another.

3.5 Difficulties in Using Software Cost Estimation Practices

The researcher has focused on certain areas of difficulties faced by practitioners while estimating Software Projects. Each software project is unique, and it is always challenging to estimate software exactly. Even if the requirements are given, it is hard to estimate a complex system that is going to be built with the given requirements. So, estimator must make sure those requirements are clear to respective stakeholder and never estimates of unknown requirements.

Software programmers spend the designated time on a task. If the task is estimated to take three days, a programmer should make sure that it takes three days. Even task finishes early; he will refine solution or slack off until the designated time gets over.
Many a times, estimated tasks depend on other estimated things; it creates a series of delays. After estimations if there is a change in requirements it should be notified immediately, otherwise it may hamper entire estimation of that software project. Every stakeholder from the software development team should raise any problem or issue on time as it may slow down the project work if problem raised by him is too late.

Software project development team members should never spend a lot of time on non-estimated tasks. If it is required, the estimation must be refined.

Return on Investments concept used in economics to calculate the profits of software projects and its income after introducing the project/product on the client side. In the software industry, the estimation process should calculate and predict the ROI as a metric for any software before the development takes place. This helps the management and the customer to decide on how much it will cost as a management and as a customer how much they will bid for this software.

Mounting the recommended metric of ROI helps to measure not only cost, but analyses the criteria which affect in an indirect way as customer satisfaction parameters, software quality, maintenance, operation cost, and software improvement on the organization, reliability which measures this value according to the experience of estimator for both side customer and the software development company.

IV. IMPACT OF SOFTWARE COST ESTIMATIONS ON ORGANIZATION'S GROWTH AND SUSTAINABILITY

Significant Software Cost Estimation practices help IT industry to offer cost-effective, great quality, high reliability, speedy deliveries to the market. In reality, this is possible because of effective planning, execution, and closure of a software project. Apart from planning, execution, and launch there are multiple functionalities associated to run IT Industry. These functionalities are human resource management, technical and soft skill training of resources, development, training, and support to customers, quality assurance, infrastructure development and management, safety and health care resources, marketing and advertisement, research and development, social responsibilities. The core business of IT industry is a software development and providing solutions to markets. With effective estimations organizations can perform smart work; it results in quality delivery of products, stress-free work environment, innovations in work culture and software project management. Ultimately, these are the reason for organization's economic and social growth. Emerging technologies such as Social media, Mobility, Analytics and Cloud are driving the growth and helping industry to move to the next level.
Subsequently, sustainability matters to keep up organizations existence in the market. It is the ability to be maintained at a certain rate or level. It is an organization's ability to express its relevancy through a significant solution that has a quantifiable impact. Researcher identifies three kinds of resources needed by software life cycle processes: human resources, economic resources, and energy resources. Organizations that have achieved sustainability are self-contained, dedicated to their mission, and engage in continuous planning.

Sustainable software organizations are not essentially economically self-sufficient entities. Majority organizations never perform their assignments with their own income. Though sustainable organizations are financially self-reliant, and these organizations require resources other than its own to complete their work, but at organization compromise their targets or programming by attempting to align with a potential funder's priorities.

Keeping software organization sustainable, it needs constant effort and unity. This is a continuous process which requires short, medium, and long-term planning, knowledgeable, experienced, energetic and sufficient management and staff, visionary guidance, and ongoing strategic planning process. Through this process, organizations know what resources they need, what resources are available, or potentially so, and how they will pursue them.

Organizations are increasingly getting a tipping point at which they make a decision to incorporate sustainability into their structures. However, this tipping point looks special for each organization, depending on their exclusive strategies and existing structures. Coordination is a key role to maintain sustainability. Sustainability programs promote strategic and long-term thinking, and sustainability staff implements initiatives across a broad spectrum of environmental issues, relying on the work of others. And all this is achievable with three key factors as talent pool in the organization, efficient software project management, and stress-free work culture in an organization. A principle reason to achieve these three key factors is matured and effective Software Cost Estimation.

Additionally, eco-friendly business models help to set up trust with customers and provide strong values for a company. A sustainable business means providing products and services that gratify the customers without jeopardizing the environment. Organizations that build sustainability into their business models are the ones that are long-lived. If organizations are running their prime business efficiently they become economically stable and these organizations can serve in society.

V. FUTURE OF SOFTWARE COST ESTIMATION

Further research would be conducted in a different context to see if larger projects or continuous product development projects suffer from traditional problems. The research of more accurate size and effort estimation methods led to revising former models and approaches to this problem. New estimation approaches include many alternatives to Putnam's model, function point analysis, application of fuzzy logic, neural network, test execution effort, and Bayesian belief network.
There are a few more Software Cost Estimation methods are available in the industry as Bayesian Belief Networks which is the process of fore - casting the software effort to estimate software costs of both development and maintenance. An important fact to realize about Bayesian Belief Networks is that they are not dependent on knowing exact historical information or current evidence.

VI. CONCLUSIONS

In this paper, the researcher focuses on the existing software estimation methods. Software cost estimation is really a vital activity that requires knowledge of a number of key attributes. This paper provides a background of Software Project Management Process Group, Need of estimations in successful project management. Information about the past and/or the current situation is vague, incomplete, conflicting, and uncertain. Estimation techniques focus only on the actual development effort furthermore. For accomplishing successful project Software Cost estimation is mandatory. In absence of the estimated project cannot move ahead. It actually affects the business relation with customers. Hence it directly hit the financial growth of the organization. There is a high impact of software cost estimation on organizations growth and sustainability.

REFERENCES

[1]] M. Lefley and M. J. Shepperd, Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets , LNCS, Genetic and Evolutionary Computation — ISBN: 978-3-540-40603-7, page-208, GECCO 200.
[2] H. Agahi, S. Malhotra and J. Quirk, Estimating Software Productivity and Cost for NASA Projects, Journal of Parametrics, pp. 59-71, 1998.
[3] . Chulani, B. Boehm and B. Steece, From Multiple Regression to Bayesian Analysis for Calibratuing COCOMO, Journal of Parametrics, vol. 15(2), pp. 175-188, 1999
[4] Boehm, 1981 "Software Engineering Economics", Prentice Hall.
[5] Boehm, B. W. and P. N. Papaccio (1988). ―Understanding and controlling softwarecosts" IEEE Transactions on Software Engineering 14(10): 1462-1477.
[6] Benton, B.: Model-Based Time and Cost Estimation for Software Testing Environment. In: Sixth International Conference on Information Technology: New Generations, pp. 801-806. IEEE (2009)
[7] Ms. Aruna Shrivastava, Asst. Professor Leena Sahu - Software Cost Estimation: A Survey of Current Practices for International Journal of Engineering and Technical Research (IJETR) ISSN: 2321-0869, Volume-3, Issue-5, May 2015.
[8] Ricardo Valerdi, Yue Chen, Ye Yang,Center - System Level Metrics for Software Development Estimation for Software Engineering, University of Southern California.
[9] Abedallah Zaid, Mohd Hasan Selamat, Abdual Azim Abd Ghani, Rodziah Atan and Tieng Wei Koh, Issues in Software Cost Estimation , Faculty of Computer Science and Information Technology, University Putra Malaysia.

[10] Caper Jones, "Estimating Software Cost", Tata Mc–Graw Hill Edition,2007
[11] M.V.Deshpande, S.G.Bhirud, "Analysis of Combining Software Estimation Techniques" for International Journal of Computer Application (0975-8887)", August,2010
[12] Jurka Rahikkala, Sami Hyrynsalmi, Ville Leppänen Vaadin Ltd, Turku, Accounting Testing in Software Cost Estimation: A Case Study of the Current Practice and Impacts, Finland
[13] Shivangi Shekhar, Umesh Kumar Govt.Women Engg.College,Ajmer,Rajasthan - Review of Various Software Cost Estimation Techniques for International Journal of Computer Applications (0975 – 8887) Volume 141 – No.11, May 2016.
[14] Chetan Nagar, Anurag Dixit, "Software efforts and cost estima- tion with sy stematic approach", IJETCIS, ISSN:2079 -8407, V ol.2 No.7, July 2011. Pressman, Roger S., "Software Engineering: A Practioner' s Approach", 6th Edn., McGraw-Hill New York, USA., ISBN: 13: 9780073019338,2005.
[15] Pichai Jodpimai, Peraphon Sophatsathit, and Chidchanok Lursin- sap, "Analysis of Effort Estimation based on Software Project Models", IEEE, 2009.
[16] C. S. Reddy, K. Raju, An Improved Fuzzy Approach for COCOMO„s Effort Estimation using Gaussian Membership Function, Journal of Software, VOL. 4, NO. 5, pp. 452-459, 2009.
[17] Andriano L.I. Oliveira, Estimation of Software Project Effort with Support Vector Regression, www.journals.elsevier.com/neurocomputing, Vol.- 69, Issues 13–15, pp. 1749–1753, August 2006.
[18] Abedallah Zaid, Mohd Hasan Selamat, Abdual Azim Abd Ghani, Rodziah Atan and Tieng Wei Koh, Issues in Software Cost Estimation, IJCSNS, VOL.8 No.11, November 2008.
[19] A. S. Andreou, E. Papatheocharous, Software Cost Estimation using Fuzzy Decision Trees, 23rd IEEE/ACM International Conference on Automated Software Engineering, pp. 371 - 374, 2008.
[20] Yinhuan,Z., W.Beizhan,et al. "Estimation of software projects efforts based on function point",Computer Science & Education. ICCSE ,4th International Conference,2009.
[21] Mustafa,K.K Gowthaman, and R.A.Khan,"Measuring the Function Points for Migration Project: A Case Study", American Journal of Applied Sciences,2005.
[22] "FAHSCEP: Fuzzy and Analogy based Hybrid Software Cost Estimation Process", International Review on Computers and Software,2013.
[23] Duncan Haughey, Work Breakdown Structure 101 The Foundation of Project Planning for www.projectsmart.co.uk
[24] Francis S. Patrick, Turning Many Projects into Few Priorities with TOC for http://www.focusedperformance.com
[25] Narendra Sharma, Aman Bajpai, Mr. Ratnesh Litoria, A Comparison of software cost estimation methods: A Survey for TIJCSA Publication
[26] Steve McConell,"Software Estimation Demistifying the Black Art ", Microsoft Press, Dreamtech Press 2013
[27] T.N.Sharma, Anil Bhardwaj, Anita Sharma, Cost Estimation Tool for Commercial Software Development Industries for International Journal of Computer Trends and Technology- volume3Issue2- 2012
[28] M. A. Parthsarathy, "Practical Software Estimation" for PEARSON Education, 2007
[29] Khalid Mahmood , M. Manzoor Ilahi, Shakeel Ahmad , Bashir Ahmad, Integration Efforts Estimation in Service Oriented Architecture (SOA) Applications for Information and Knowledge Management www.iiste.org
ISSN 2224-5758 (Paper) ISSN 2224-896X (Online) Vol 1, No.2, 2011
[30] Manisha Arora, Richa Arya, Dinesh Tagra, Anil Saroliya, Varun Sharma, Cost Estimation Tool for Commercial Software Development Industries for International Journal of Computer Trends and Technology- volume3Issue2- 2012