National Conference on Recent Research in Engineering and Technology (NCRRET-2015)
International Journal of Advance Engineering and Research Development (IJAERD)
e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

An Approach for Improving the Efficiency of Software Team by Continuous Evaluation of Software Development

Arpit Mehta[1], Yoothika Patel[2]

[1]*Asst.Professor in Computer Science Department at DJMIT Mogar*
[2] *Asst.Professor in Computer Science Department at DJMIT Mogar*

Research Scholar CIIT Indore, Prof.CSE Dept CIIT INDORE ,Asst. Prof. & Head, CSE Dept

ABSTRACT

Accurate software cost and schedule estimations are essential specially for large software projects. However, once the required efforts have been estimated, little is done to recalibrate and reducethe uncertainty of the initial estimates. To address this problem,we have developed and used a framework to continuously monitor the software project progress and readjust the estimated effort utilizing the Constructive Cost Model II (COCOMO II) and the Unified Code Count Tool. As a software project progresses, we gain more information about the project itself, which can then be used to assess and re estimate the effort required to complete the project. With more accurate estimations and less uncertainties, the quality and goal of project outcome can be assured within the available resources. The paper thus also provides and analyzes empirical data on how projects evolve within the familiar software “cone of uncertainty.”

Categories and Subject Descriptors

[**Management**]: Cost estimation, Life cycle, Time estimation

General Terms Management, Measurement, Economics

Keywords Cost Estimation, Uncertainty

1. INTRODUCTION

Having accurate estimations of the effort and resources required to develop a software project is essential in determining the quality and timely delivery of the final product. For highly precedented project and experienced teams, one can often use“yesterday’s weather” estimates of comparable size andproductivity to produce fairly accurate estimates of project effort.More generally, though, the range of uncertainty in effortestimation decreases with accumulated problem and solutionknowledge within a “cone of uncertainty” defined in [1] and calibrated to completed projects in [2]. To date, however, there have been no tools or data that monitor the evolution of a project’s progression within the cone of uncertainty.

Our goal is to develop a routine, semi-automated assessment framework that helps reduce uncertainties of the software project estimation as the project progresses through its life cycle. The assessment framework integrates the Unified Code Count tool (UCC) with the COCOMO II estimation model to quickly generate information to analyze the team's performance and estimations. This is similar to the concepts of [10], which shows that frequent assessment of the project status help improve the team as well as the final product of the project. We apply this concept to assess the efforts spent on the project and compare with the current progress to predict the effort required to complete the project. This information is then used to evaluate the current project estimations and adjust the estimation parameters as necessary. This will eventually enable the actual and estimated effort to converge. The assessment framework allows the team to validate the direction of the project, while increasing the project understanding as well.

The key benefits of achieving a convergence between actual andestimated efforts are as follows:

- It allows the development team to improve planning andmanagement of project resources and goals.
- It enables the product's quality to be controlled closely.
- It helps the stakeholders to better understand the actualproject's progress and status.

2. Problem and Motivation

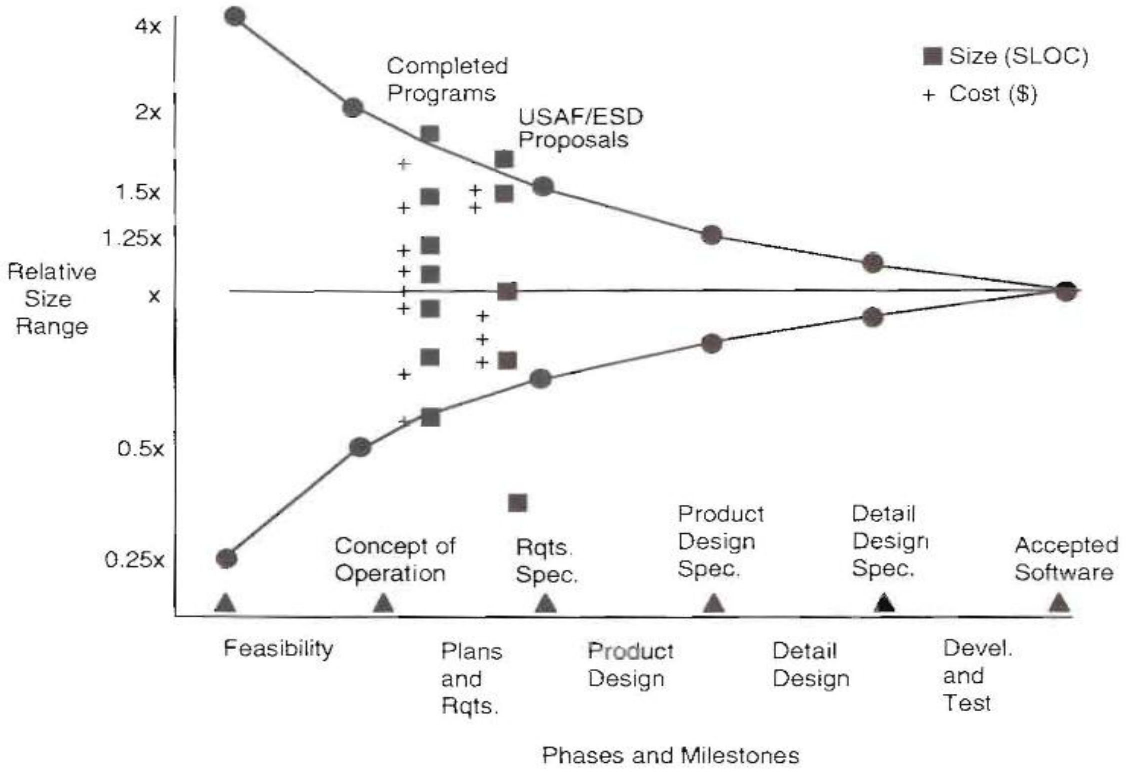

Figure 1: The Cone of Uncertainty [2]

The main motivation behind the development of the assessment framework is derived from the well-know software "cone of uncertainty" problem. Figure 1 shows the accuracy of software sizing and estimation by phases. The level of estimation uncertainties is high during the initial estimations due to lack of data and experience. As long as the projects are not reassessed or the estimations not revisited, the cones of uncertainty are not effectively reduced [1].

2.1 Imprecise Project Scoping

When the projects begin with the initial overestimations, teams are required to re-negotiate with the clients to either reduce the size of the projects or adjust the time frame. On the other hand, when a project underestimates the resources, it tends to overshoot the goals that the project can achieve. Thus, the project's quality suffers significantly or the project itself becomes undeliverable due to insufficient resources.

2.2 Project Estimations Not Revisited

During the initial estimation for the software project to be developed, the teams often do not have sufficient data to carefully analyze and perform the necessary predictions. This missing information includes aspects that are specified in the COCOMO II cost drivers [2]. In most cases, the project estimation turns into a constant value once the project enters the development phase regardless of how well the project progresses or how capable the programmers actually are. There is a significant number of uncertainties at the beginning of the project as there are instability in requirements and there are many directions that the project can proceed on.

2.3 Manual Assessments are Tedious

The tasks of manually assessing the project progress are tediousand discouraging to the team due to their complexities and theamount of effort required. In order to collect enough informationto have a useful assessment data, the teams often need to performvarious surveys and reviews to determine how well the teamperformed in previous iterations [10].

2.4 Limitations in Software Cost Estimation

Regardless of what software cost estimation technique is used, there is little that the technique can compensate for the lack of information and understanding of the software to be developed. As clearly shown in [1], until the software is delivered, there exists a wide range of software products and costs that can turn into the final outcome of the software project. In addition to the fact that the initial estimations lack the necessary information to achieve accurate estimates as mentioned in section 2.2, the software design and specifications are prone to changes throughout the project life cycle as well, especially in an agile software engineering environment.

3. Related Work

The most thorough and balanced coverage of **software estimation methods** is "Estimating Software-Intensive Systems"[14]. More recent updates, including discussions of expert judgment vs. parametric-model estimation strengths and weaknesses, are [8] and [9]. A good treatment of agile estimation is [4].Early treatments of **software estimation uncertainty** include the PERT sizing method in [12] and the wideband Delphi estimate distributions in [2] and the accuracy-vs.-phase chart in [1],calibrated in [2], and termed the "cone of uncertainty" in [11].Most commercial estimation models now include capabilities to enter input uncertainties, run a number of random-sample Monte Carlo estimates, and produce a cumulative probability distribution estimate of the probability that the actual cost will exceed a given budget [7].In the aspect of **software project tracking methods**, a good early treatment is "Controlling Software Projects" [5]. Tracking progress vs. estimated budget and schedules via Earned Value Management (EVM) systems is covered well in [6].

4. Model

The framework that we developed introduces a semi-automated method to help rapidly assess the project status and progress based on the effort spent and the number of SLOC. Figure 2provides an overview of the assessment framework.

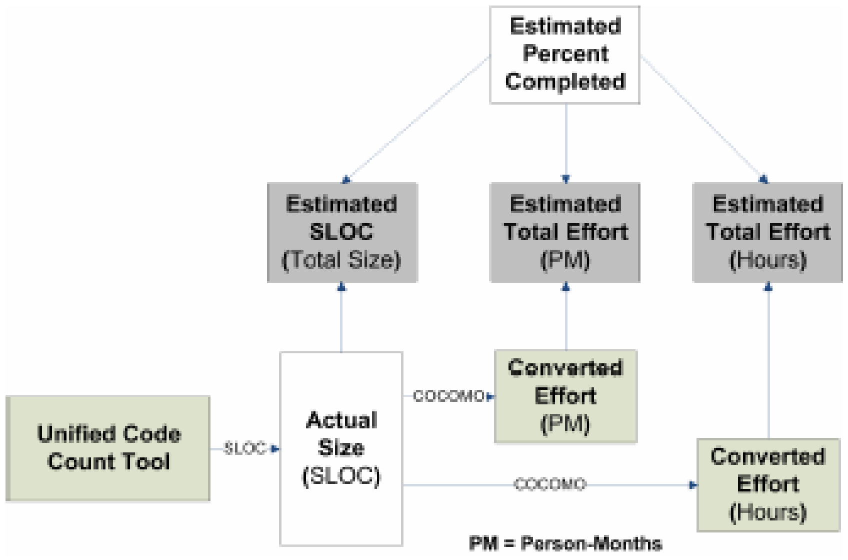

Figure 2: Assessment Framework Model

4.1 Effort Estimation

The assessment framework utilizes the COCOMO II estimation model to estimate the resources required to complete a software development project. It takes the adjusted SLOC of each module along with the necessary effort multiplier parameters and applies them to the COCOMO II estimation model to generate actual efforts in PM, which can then be converted to number of hours.

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

$$E = 0.91 + 0.01 \times \sum_{j=1}^{5} SF_j$$

where:
- *A* = 2.94 (a constant derived from historical project data)
- *Size* is in KSLOC

- *EM* is the effort multiplier for the **ith** cost driver. The geometric product results in an overall effort adjustment factor to the nominal effort.
- *SF* is the scale factor used to compensate for the economies or diseconomies of scale.
- *NS* stands for "nominal schedule"

4.2 Size Counting

The sizes of the projects are obtained using the Unified CodeCount tool (UCC) of which the counting standards are based on [12]. The UCC tool provides a fully automated process to obtain the number of SLOC. The tool takes a list of source code files as input and generates the number of physical and logical SLOC as outputs, which are then fed to the COCOMO II formula. We only take the number of logical SLOC as these are the lines of code that require real effort to develop.

The main motivation behind the development of the assessment framework is derived from the well-know software "cone of uncertainty" problem. Figure 1 shows the accuracy of software sizing and estimation by phases. The level of estimation uncertainties is high during the initial estimations due to lack of data and experience. As long as the projects are not re-assessed or the estimations not re-visited, the cones of uncertainty are not effectively reduced [1],[12]. The UCC tool provides a fully automated process to obtain the number of SLOC. The tool takes a list of source code files as input and generates the number of physical and logical SLOC as outputs, which are then fed to the COCOMO II formula. We only take the number of logical SLOC as these are the lines of code that require real effort to develop.

4.3 Model Calculations

The framework's inputs can be categorized into two types: static and dynamic inputs. The static inputs are not frequently changed until the project meets the major milestones. These include the SLOC sizes of each module, the COCOMO II parameters, and the requirements evolution and volatility (REVL) for each module. The dynamic input needs to be updated for each assessment, which is the estimated percent completed of each module. When the raw SLOCs are obtained from the UCC tool, the SLOCs are readjusted with REVL to reflect the cost from requirements evolution. The estimated total size and effort for each software module are calculated using these formulas:

$$Estimated\ SLOC = \frac{Adjusted\ SLOC}{Estimated\%complete} \times 100$$

$$Estimated\ Total\ effort = \frac{Converted\ Effort(PM)}{Estimated\ \%\ Complete} \times 100$$

$$Hours = PM \times 180\ Hours/PM$$

5. Analysis

We performed simulations of our assessment framework on two software projects from USC's software engineering course with24-week development timeframe. The versions of the source code files submitted to the Subversion server at the end of each week were used as inputs to the UCC tool to provide us with the data. The two projects were chosen for their similarities in project

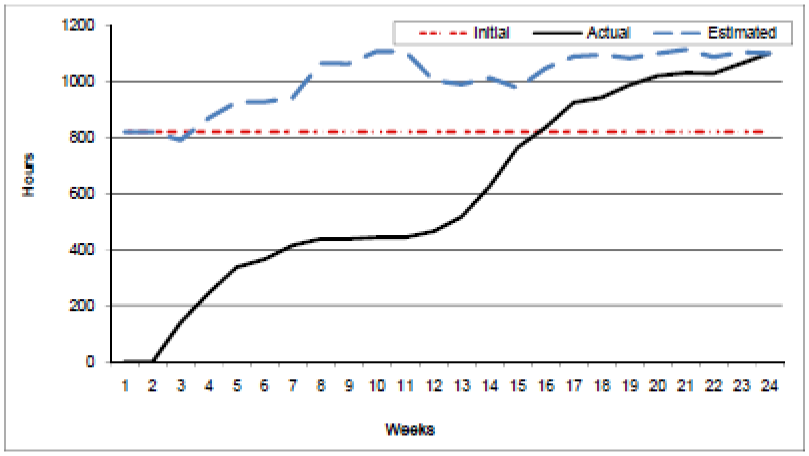

Figure 4: Simulation Result of Team B

types, sizes, and complexities, which are e-service projects to develop web-based database management systems using JSP technology. Both teams were closely involved in this process for the simulation to reflect the reality as much as possible.

5.1 Overview of Results

The results of the assessment simulation on both projects show that the estimated and actual efforts converge as the projects progress through their lifecycles.

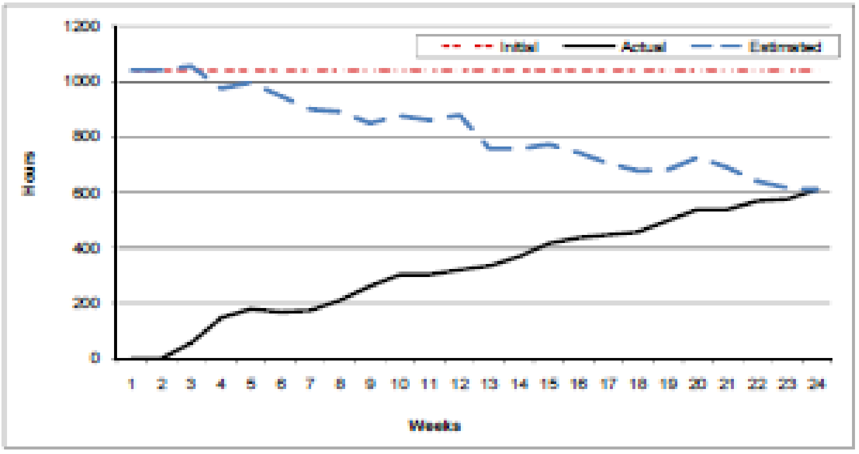

Figure 3: Simulation Result of Team A

project (shown in the coarse-dotted line), on the other hand, converges to the required effort as the estimations are revisited and adjusted during each assessment. Finally, the fine-dotted line represents the effort estimation performed by the team at the beginning of the project. It is interesting to observe the difference in the behavior of the “cone of uncertainty” between the two teams. Team A overestimated the effort required to complete the project by over50%. Based on our discussion with the team members, the main reason for their estimation error was due to the fact that they were pessimistic about the developers’ capabilities and assumed the project to be more complicated than it actually was. On the other hand, Team B underestimated their required effort by over 18%due to the lack of experience in identifying the actual effort that would be required to develop certain modules. Moreover, the developers were not experienced with the development language, JSP, so they were not aware of the complexities that could potentially occur during the project. Based on the simulation, both teams demonstrated the same phenomenon where the gaps in the "cone of uncertainty" in effort estimation decreases throughout the project lifecycle and converges at the end of the project.

5.2 Percentage of Estimation Errors

Figure 5 shows the rates of estimation errors for both teams throughout the 24 weeks of development. Although we had hoped that the error rate would be smoother and more linear, the end result clearly shows the improvement week by week. The reason that the error rates in estimation error fluctuate as such is due to the fact that there are still discrepancies and lack ofexperience in identifying the percent completeness of each module and of the project as a whole. However, the reductions in error rates are significant compared to the initial estimates done by the developers, thus, showing a much more accurate estimation when utilizing our assessment framework.

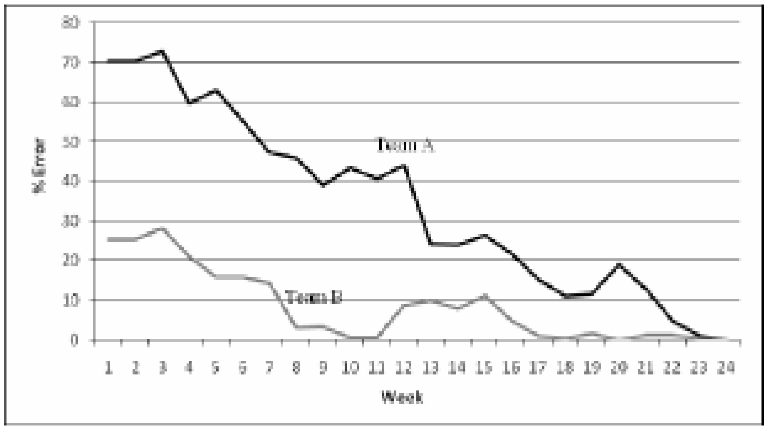

Figure 5: Estimation Error Percentage

5.3 Estimated Overall Project Progress

Currently, a project's overall progress is generally reported based on the initial estimates of the project. Since the initial estimates are often inaccurate with either an overestimation or underestimation, the actual project progress cannot be determined accurately.

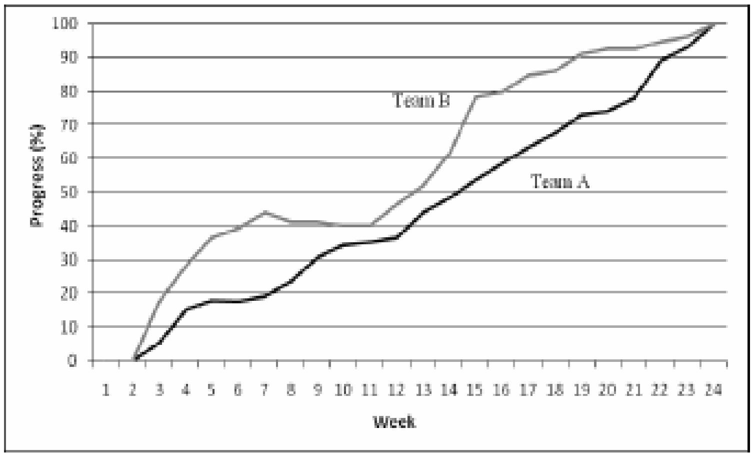

Figure 6: Project Progress Percentage

Figure 6 shows the estimated overall project progress for both teamsthroughout the 24 weeks of development. The assessmentframework's output can be represented as the overall projectprogress which is useful to all critical stakeholders in order to adjustproject plan. The project progress is calculated by using the effortconverted from SLOC developed and comparing it against theadjusted estimated effort. As the assessment allows the estimationsto become more and more accurate as the project moves forward,the project progress becomes more realistic as well. This allows allthe success critical stakeholders to observe the actual progress of theproject and monitor to see whether the project can be delivered on time or not.

6. Conclusions and Future Work

We have presented a novel framework for performing continuousassessments on the project progress in order to produce betterestimates. The assessment framework utilizes an automated codecount tool, UCC, to generate inputs to our framework which can beconverted into effort using the COCOMO II model. As theassessments are performed, the COCOMO II parameters areevaluated and updated in order to yield better predictions based onthe current situation.
We performed a simulation of our assessment framework on datafrom two software development projects taken from USC's softwareengineering course. As shown in our analysis, the results of thesimulation have shown significant improvements in estimatingproject resources with significant reduction in estimation errors asthe project progresses through its life cycle. It can thus be concludedthat the continuous assessment can help predict the efforts which arerequired to achieve similar projects with fixed schedules. Again, thisconclusion is only suggestive vs. definitive for other classes of applications. It is interesting to note that, relative to skeptical statements that onlythe optimistic lower part of the Cone of Uncertainty is ever visited,
in this case, one of the projects underestimated and had to increaseeffort, while the other project found ways to satisfy the client usingless effort. This is a not a large sample size, but shows that the upperpart of the Cone of Uncertainty does exist.Our primary target for future work is to develop a tool to fullysupport the framework by integrating both the UCC tool and theCOCOMO II calculation model. We will then observe the effects on project performance as well as determine the frequencies of the assessment that will yield the most effective results, or the sweet spot of our assessment framework. Furthermore, we will experiment our assessment framework on projects of large scale and of different types in order to observe the economies of scale and the prediction accuracy of the framework as the nature of the projects changes.
Finally, we will apply the concepts of value-based software engineering practice into our assessment model by taking the priority of the requirements. As each software module has different levels of importance and criticality, they should not be treated as equal. Weights should be applied to each module with respect to the priority of the software requirements. This will affect the estimation and percent completion as software modules with higher priority and criticality should yield higher percentage of completion than those with lower priorities.

7. References

[1] Boehm, B. "Software Engineering Economics". Prentice-Hall,1981.
[2] Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B. K.,Horowitz, E., Madachy, R., Reifer, D. J., and Steece, B.*Software Cost Estimation with COCOMO II*, Prentice-Hall,2000.
[3] Boehm, B., Egyed, A., Port, D., Shah, A., Kwan, J., andMadachy, R. "Using the WinWin Spiral Model: A CaseStudy," *IEEE Computer*, Volume 31, Number 7, July 1998, pp.33-44 (usc-csse-98-512)
[4] Cohn, M. *Agile Estimating and Planning*, Prentice-Hall, 2005

[5] DeMarco, T. *Controlling Software Projects: Management,Measurement, and Estimation*, Yourdon Press, 1982

[6] Fleming, Q. W. and Koppelman, J. M. *Earned Value ProjectManagement, 2nd edition*, Project Management Institute, 2000

[7] Galorath, D. and Evans, M. *Software Sizing, Estimation, andRisk Management*, Auer-bach, 2006

[8] Jorgensen, M. and Boehm, B. "Software Development EffortEstimation: Formal Models or Expert Judgment?" *IEEESoftware*, March-April 2009, pp. 14-19

[9] Jorgensen, M. and Shepperd, M. "A Systematic Review ofSoftware Development Cost Estimation Studies," *IEEE Trans.Software Eng.*, vol. 33, no. 1, 2007, pp. 33-53

[10] Krebs, W., Kroll, P., and Richard, E. Un-assessments reflections by the team, for the team. Agile 2008 Conference

[11] McConnell, S. *Software Project Survival Guide*, MicrosoftPress, 1998

[12] Nguyen, V., Deeds-Rubin, S., Tan, T., and Boehm, B. "ASLOC Counting Standard," COCOMO II Forum 2007

[13] Putnam L. and Fitzsimmons, A. "Estimating Software Costs,Parts 1,2 and 3," *Datamation*, September through December1979

[14] Stutzke, R. D. *Estimating Software-Intensive Systems*, PearsonEducation, Inc, 2005.