Parametric Models for Effort Estimation for Global Software Development

S. Ramacharan and K. Venu Gopala Rao

Abstract—The main objective & aim of this paper is to identify the problems that arise due to distributed software & offshore software development. This includes numerous critical success factors which may make vulnerable to the success of development. Generally for software development perfect estimation is crucial especially in globally distributed projects & offshore projects. This article introduces a Study based on COCOMO II, SLIM, ISBSG effort estimation methods. These methods enable more precise estimation methods than conventional methods.

Index Terms—COCOMO II, effort estimation methods, SLIM, ISBSG, off shoring, risk management.

I. INTRODUCTION

Software development efforts estimation is the process of predicting the most realistic use of effort required to develop or maintain software based on incomplete, uncertain and/or noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets and investment analyses, pricing processes and bidding rounds.

Off shoring has become a key software development strategy. Due to globalization software developments lead to major bang on the whole software process. So many organizations are looking for high quality of the product with reduced time & cost. The new forms of competition & cooperation that have arisen in software engineering as a result of the globalization process have an impact on the whole software process. Generally global software development involves following issues tactical issues, cultural issues, inadequate communication, knowledge management, project & process management issues, technical issues [1].

Software researchers and practitioners have been addressing the problems of effort estimation for software development projects since 1960. Typically, effort estimates are over-optimistic and there is a strong over-confidence in their accuracy. The mean effort overrun seems to be about 30% and not decreasing over time. For a review of effort estimation error surveys are performed. The strong over-confidence in the accuracy of the effort estimates is illustrated by the finding that, on average, if a software professional is 90% confident or “almost sure” to include the actual effort in a minimum-maximum interval, the observed frequency of including the actual effort is only 60-70%.

Currently the term “effort estimate” is used to denote as different concepts as most likely use of effort (modal value), the effort that corresponds to a probability of 50% of not exceeding (median), the planned effort, the budgeted effort or the effort used to recommend a bid or price to the client. This is believed to be unfortunate, because communication problems may occur and because the concepts serve different goals.

Existing effort estimation Techniques used for co-located software projects do not have sufficient competence to estimate effort and duration exactly for distributed software projects. It fall out into overly cost and schedule, and most of the projects fail. This is why, it is essential to find some new methods or calibrate accessible methods for distributed software projects. It is required to explore the accuracy of present effort estimation methods and find the prospective improvement alternatives.

<table>
<thead>
<tr>
<th>Estimation approach</th>
<th>Category</th>
<th>Examples of support of implementation of estimation approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analogy-based estimation</td>
<td>Formal estimation model</td>
<td>ANGEL, Weighted Micro Function Points</td>
</tr>
<tr>
<td>WBS-based (bottom up) estimation</td>
<td>Expert estimation</td>
<td>Project management software, company specific activity templates</td>
</tr>
<tr>
<td>Parametric models</td>
<td>Formal estimation model</td>
<td>• COCOMO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SLIM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SEER-SEM</td>
</tr>
<tr>
<td>Size-based estimation models</td>
<td>Formal estimation model</td>
<td>Function Point Analysis, Use Case Analysis, SSU (Software Size Unit), Story points-based estimation in Agile software development</td>
</tr>
<tr>
<td>Group estimation</td>
<td>Expert estimation</td>
<td>Planning poker, Wideband Delphi</td>
</tr>
<tr>
<td>Mechanical combination</td>
<td>Combination-based estimation</td>
<td>Average of an analogy-based and a Work breakdown structure-based effort estimate</td>
</tr>
<tr>
<td>Judgmental combination</td>
<td>Combination-based estimation</td>
<td>Expert judgment based on estimates from a parametric model and group estimation</td>
</tr>
</tbody>
</table>

So in this research article focused more on the construction of formal software effort estimation models. The early models were normally based on regression analysis or mathematically derived from theories from other domains. Ever since enormous number of model building approaches have been evaluated, such as approaches founded on case-based reasoning, classification and regression trees, simulation, neural networks, Bayesian statistics, lexical analysis of requirement specifications, genetic programming, linear programming, economic

Manuscript received December 1, 2012; revised April 11, 2013.
S. Ramacharan is with JNT University, Hyderabad, India (e-mail: s.ramacharan@gmail.com).
K.Venu Gopala Rao is with Department of Computer Science at G.Narayannama Institute of Technology and Science (GNITS) Shakkpet, Hyderabad, India (e-mail: kvgraol234@gmail.com).

DOI: 10.7763/LNSE.2013.V1.40
production models, soft computing, fuzzy logic modeling, statistical bootstrapping, and combinations of two or more of these models. The perhaps most common estimation products today, e.g., the formal estimation models COCOMO [2] and SLIM have their basis in estimation research conducted in the 1970s and 1980s. The estimation approaches based on functionality-based size measures, e.g., function points, is also based on research conducted in the 1970s and 1980s, but are re-appearing with customized size measures under diverse labels, such as “use case points” in the 1990s and COSMIC in the 2000s.

II. ESTIMATION APPROACHES

There are many ways of categorizing estimation approaches in show in the Table I. The major categories are the following:

• **Expert estimation**: The quantification step, i.e., the step where the estimate is produced based on judgmental processes.

• **Formal estimation model**: The quantification step is based on mechanical processes, e.g., the use of a formula derived from historical data.

• **Combination-based estimation**: The quantification step is based on a judgmental or mechanical combination of estimates from different sources.

• **COCOMO** (The Constructive Cost Model (COCOMO)) is an algorithmic Software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current project characteristics.

• **SLIM** (Putman’s SLIM (Software Life Cycle Management)) is an automated ‘macro estimation model’ for software estimation based on the Norden/Rayleigh function. SLIM uses linear programming, statistical simulation, program evaluation and review techniques to derive a software cost estimate.

• **SEER-SEM** (powerful decision-support tool for estimating software development and maintenance cost, labor, staffing, schedule-Software Evaluation and Estimation of Resources - Software Estimating Model [3])

III. PARAMETRIC MODELS

A. **COCOMO**

Its hypothetical basics were considered in the 70’s by Barry Boehm which was establish for better and more realistic estimations for software projects. The COCOMO-I was the first version introduced in 1981. The intention of this model was to provide an approach to deal with software engineering problems from the outlook of human economics as well as from the programming perception. Software engineering world has in the interim changed creating a need to transform the innovative COCOMO model [4] to the COCOMO II which was published in 2000. Which is widely accepted public cost model. COCOMO II is based on more than 250 projects and is calibrated with 161 actual project data. It can be calibrated from the organization’s historical data. Its factors can be taken as customary values in case there is no data available for the parameter objective impact analysis.

The basic version of COCOMO estimates the effort of a software development project in person-month. The COCOMO II enables the use of source code lines and function points as reference parameters [5] for the calculation of the projects’ Size (S). It can be used to conclude the actual size of the project by algorithmic methods as well as historical data or expert opinions.

<table>
<thead>
<tr>
<th>TABLE II: EFFORT MULTIPLIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Effort Multipliers</strong></td>
</tr>
<tr>
<td><strong>Product Attributes:</strong></td>
</tr>
<tr>
<td>RELY</td>
</tr>
<tr>
<td>DATA</td>
</tr>
<tr>
<td>CPLX</td>
</tr>
<tr>
<td>REUSE</td>
</tr>
<tr>
<td>DOCU</td>
</tr>
<tr>
<td><strong>Platform Attributes:</strong></td>
</tr>
<tr>
<td>STOR</td>
</tr>
<tr>
<td>PVOL</td>
</tr>
<tr>
<td>TIME</td>
</tr>
<tr>
<td><strong>Personnel Attributes:</strong></td>
</tr>
<tr>
<td>ACAP</td>
</tr>
<tr>
<td>PCAP</td>
</tr>
<tr>
<td>LTEX</td>
</tr>
<tr>
<td>PLEX</td>
</tr>
<tr>
<td>APEX</td>
</tr>
<tr>
<td>PCON</td>
</tr>
<tr>
<td><strong>Project Attributes:</strong></td>
</tr>
<tr>
<td>SITE</td>
</tr>
<tr>
<td>SCED</td>
</tr>
<tr>
<td>SITE</td>
</tr>
<tr>
<td>TOOL</td>
</tr>
</tbody>
</table>

• Required Reusability (RELY)  
• Database Size (DATA)  
• Product Complexity (CPLX)  
• Developed for Reusability(REUSE)  
• Documentation match to SDLC( DOCU)  
• Main storage Constraint(STOR)  
• Platform Volatity (PVOL)  
• Execution Time Constraint(TIME)  
• Analysys Capability(ACAP)  
• Programming Capability(PCAP)  
• Language & Tools Experience(LTEX)  
• Platform Experience(PLEX)  
• Application Experience(APEX)  
• Personal Continuity (PCON)  
• Multisite Development(SITE)  
• Scheduling Factors(SCED)  
• Use of Software Tools(TOOL)  

There are three sub models for COCOMO II:

• Application Composition Model;  
• Post Architecture Model;  
• Early Design Model.

The projects that use incorporated Computer Aided Software Engineering tools are mostly estimated for effort and schedule by using Application Composition Model. Rapid application development uses these tools. Furthermore, these tools also support prototype activities occurring later in spiral model.
The Early Design and Post Architecture models are used to estimate effort and schedule for the projects like software infrastructure, embedded systems and large applications. When there is incomplete project analysis, early design model is used for rough estimation. Whereas post-architecture model is used when analysis and top-level design of project is completed and know detailed information about project. Post Architecture and Early Design models use the same approach to find the size of project and scale factors as well. Reusing code and other data from previous projects is also included in product sizing.

COCOMO II improved and added extra cost drivers when compared with COCOMO 81. These cost drivers are categorized into four categories for “The Post-Architecture Model”.

Scale Factors like Precedentedness (PREC), Team Cohesion (TEAM) and Process Maturity (PMAT) are affected.
- Precedentedness portrays the acquaintance of a software developer to the present project state. This factor considerably varies between the offshore provider and the onshore buyer.
- The Team Cohesion indicates the potential of the team to work as a team. This factor is reasonably affected by offshore outsourcing development.
- The Process Maturity quantifies the company’s process maturity.
- Development Flexibility (FLEX) how far the product development is flexible in the given development environment.
- Architecture/ Risk Resolution (RESL) how far the architecture is confirmed & risk involved & risk resolution during development.

COCOMO II Equation

The basic equation for COCOMO II[2] is shown as

\[ PM = A \times SIZE^E \times \Pi_{i=1}^{17} EM_i \]

PM: Person Month
A: Constant (2.94 for COCOMOII.2000)
Size: KSLOC(SLOC, Function Points)
E: Scale Factors
EM: Effort Multipliers shown in the Table II.

New Effort Equation suggested by Barry Boehm for Outsourcing Projects which includes 11 effort Multipliers along with existing 17 effort Multipliers.

\[ PM = A \times SIZE^E \times \Pi_{i=1}^{17} EM_i \times \Pi_{j=1}^{11} EMO_j \]

Outsourcing software development requires 11 Effort Multipliers Outsourcing factors as shown in the Table III.
- Buyer’s outsourcing Experience (BOXP)
- Contract Design(CODS)
- Buyer’s Project Manager [6](BUPM)
- Barrier of Language ( BALA )
- Cultural Distance (CULT).
- Different Time Zones (TMZN)
- Outsourcer’s Fit (OFIT)
- Project Management (PMGM)

So there is a need of COCOMO II amplification as show in the Fig. 1 and Fig. 2 due to distributed & outsourcing projects [7] which is carried out in 3 steps:

B. SLIM

SLIM is an algorithmic technique that is used to calculate approximately effort and schedule for projects. The fundamental reason for developing SLIM is to measure the on the whole size of a project based on its estimated SLOC [3]. This method was customized for effort estimation using Rayleigh curve model.

The SLIM tool is the product of SLIM (for the proprietary of Putnam’s model) which is a metrics-based estimation tool [4], developed by Quantitative Software Management (QSM), using validated data of over 2600 projects. These projects were classified into nine different application categories. This tool helps the management to estimate the effort and time required to build medium and large software projects. Most significantly, this tool can be customized according to a specific organization. The following equation is used to allocate the Effort

\[ E = (SLOC \times B^{0.333} / P^3) \times (1/t^4) \]

where
E- Effort
P- Productivity Parameter which depends on the type of project
B- Special Skill factor
t- Duration

There are two main management indicators, Productivity Index (PI) and Manpower Buildup Index (MBI).

PI could be taken as process productivity. PI values were observed from 0.1 to 34, whereas its values were given 0.1 to 40.0 in SLIM tool by QSM to overcome future contingencies. A high PI value means that project’s productivity is high and is low complex. In case of PI having values below average implies 10% more development time and 30 % more cost.

Second important indicator MBI [5] is measure of staff buildup rate. Some factors, by which MBI is influenced, are schedule pressure, task concurrency and resource constraints. Its values are observed in the range of -3 to 10. A low MBI value implies

- Longer time
- Fewer defects
- Less effort
- Fewer people
- Fewer LOC/month
- Higher LOC/PM

The following are steps involved in effort estimation by SLIM tool:
- First of all estimate product size
- Select/analyze the PI and MBI values for this project
- Get minimum time for solution
- Determine alternative solution for time-effort-cost
- Chose your desired solution
- At the end, generate project plan from the chosen solution.

C. Reason for Electing COCOMO II & SLIM

There are many methods for effort estimation but many reasons to select this two estimation techniques due to the information provided in Table IV.

<table>
<thead>
<tr>
<th>Reasons</th>
<th>Availability Of Literature</th>
<th>Availability of Tools</th>
<th>Coverage of Parameters</th>
<th>Used for early Estimation</th>
<th>Algorithmic to widely used</th>
<th>Possible to calibrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCOMO II</td>
<td>Many</td>
<td>Free tools are many</td>
<td>More than 17 cost drivers &amp; 5 very important scale factors</td>
<td>YES</td>
<td>YES</td>
<td>Yes, Different factors are available according to situations</td>
</tr>
<tr>
<td>SLIM</td>
<td>Sufficient but some are free</td>
<td>Most of the tools are not for free</td>
<td>2 main parameters MBI &amp; PP</td>
<td>YES</td>
<td>YES</td>
<td>Yes, according to QSM</td>
</tr>
</tbody>
</table>

IV. FUTURE WORK

There are many factors that require analysis for the sake of effort estimation methods in GSD. Exploring obscured factors for both of the models and adjusting their values for them would be captivating future work. There is a need for hypotheses for the accuracy of methods that required testing in this manner “accuracy of effort estimation methods for distributed and co-located projects is the same”. Furthermore, to explore which scheme provides more correctness, it can be tested like “SLIM provides more accurate estimation than COCOMO II for GSD projects”. Testing these hypotheses might be favorable for the improvement of effort estimation methods. More research work needed in this area in the context of GSD.

This paper established an approach of effort estimation of offshore outsourcing software development based on COCOMO II & SLIM Approach. Cost drivers were added as Effort Multipliers. It should be noted that this is still work in progress. The models eventually need more calibration and validation. An accurate estimation on the effort remains complex and hardly realizable. Traditional methods can also not live up to that. The proposed model provides an estimation of the range but not particular outline. It nonetheless helps predict the outcome of a global software development project by reducing the overall risk.

Multi-sourcing is indeed a motivating research topic. It would be much more interesting to carry out a differentiated assessment to the consequence of software development processes on the effort. More calibration based on expertise opinion needs to be done. Further interaction between Experts is needed so as to unveil any unidentified or overlapping cost drivers.

REFERENCES

S. Ramacharan was born in Guntur, Andhra Pradesh, in 1975. He received the B.E degree from Gulbarga University in 1997, the M.Tech CSE from the JNT University, Kakinada in 2007. Presently he is pursuing Ph.D degree in the area of Software development Effort Estimation using empirical methods from JNT University, Hyderabad. Started Teaching profession from 1998 and worked in Vasavi Engineering College, Hyderabad for two academic years. From 2000 started working in G. Narayanamma Institute of Technology and Science, Autonomous, Hyderabad. Apart from this also guest faculty for Wipro and Satyam companies for MS Programme through BITS Pilani.

K. Venu Gopala Rao was born in Vijaywada, Andhra Pradesh, in 1963. He received the B.Tech ECE degree from JNT University, Hyderabad in 1985, the M.Tech CSE from the JNT University, Hyderabad in 1997 and Ph.D degree in the area of Computer Science & Engineering from Osmania University, Hyderabad in 2008. As a Professor, he was in Department of computer science at G.Narayanamma institute of technology and science (GNITS) Shaikpet, Hyderabad (2006 onwards). As an Assoc. Professor, he was in Department of computer science & Engineering at Koneru Lakshmaiah College of Engineering (KLCE), Vijayawada,(1999-2002) Greenfields, Vadeesvaram, Guntur (dist). As an Asst. Professor, he was in Department of computer science & Engineering at VR Siddhartha Engineering college (VRSEC), Vijayawada (1997-1999). As an Asst. Professor, he was in Department of computer science & Engineering at Jntuce, Hyderabad (1992-1995). As a Technical Officer, he was in CSG GROUP AT ECIL Hyderabad (1989-1992). As a Quality control engineer, he was at Ashok Leyland, Hyderabad (1988-1989). As an Engineer (Maintanance), he was at Radiant Cables Ltd, Hyderabad (1985-1988).