ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 1, January 2012

Software effort estimation and risk analysis –A Survey

Poonam kaushal
Poonamkaushal14@gmail.com

Abstract— ***Software effort estimation and risk analysis are the two key components of a good software project. Quantitative approaches to software schedule and cost estimation have been developed to help project managers to estimate schedule and cost for a software project and do some sensitivity analyses based on various criteria. Such models are useful and complement the manager's judgment and intuition for decision-making. Large deviations from the estimates can cause partial/ total failure of a software project. Since risk and estimation go hand in hand, the project manager must identify and analyze risk factors to take corrective actions. In this paper a systematic approach to risk analysis is illustrated in addition a methodology for software effort estimation of component based software development is also studied.***

***Index Terms*— CBSD, Effort Estimation, Risk.**

I. INTRODUCTION

Software is measured to indicate its quality, to assess the productivity of people producing the product, to assess the benefits, to assess requirements of new tools and training and to form a baseline for estimation. The metric of the business software is different from that of engineering and scientific software. Estimations are subject to assumptions and approximations. There will be exist. Estimation of resource, cost and schedule for a software project requires experience, access to good historical information. provide an insight into various aspects of software: namely, software process, software products and so on. The metrics data collected over a period on several software projects are useful to build standards for planning and eliminating high-risk problems by careful process design. Risk items Efforts, software size and time for software development. The IEEE standard glossary of software engineering terms defines 'metrics' as "Quantitative measure of the degree to which a system, component or process possesses a given attribute". Software Metrics have two categories: "direct measure" and "Indirect Measure". Metrics based on direct measures are easy to establish, as these are more tangible and quantifiable, whereas metrics based on indirect measures are difficult to establish, as they are evolved through measures that are based on subjective judgments of the software engineer. Function Point estimation, COCOMO, Putnam, etc,are some the estimation methods discussed in the literature for effort estimation. The estimations always go hand in hand with risks. Therefore, risks are to be identified and the probabilities of occurrences are to be estimated to study the consequence of risks. Risk analysis comprises of Identifying, Analyzing, Planning, Monitoring and resolving risks. A software project is expected to produce reliable product certain degree of uncertainty in the estimates. Many techniques for estimation of time and effort required for software development do within a limited time using limited resources. Any project runs risk of not producing the Software metrics are built out of several measures to provide an insight into various aspects of software: namely, software process, software products and so on. The metrics data collected over a period on several software projects are useful to build standards for planning and eliminating high-risk problems by careful process design. Risk items include personnel estimation of resources; cost, process design risk items include personal shortfalls, unreliable schedule and budgets, developing wrong software functions, wrong interfaces, continuous changes in requirements etc. Barry Boehm a pioneer in software engineering believes that "Risk Management helps people avoid disasters, avoid rework, avoid overkill, and stimulate win-win situations on software projects". Software engineering is an approach for developing software under given project aspects such as schedule, effort and with the desired quality. This process begins with estimating the size, effort and time required for the development of the software and ends with the product and other work products built in different phases of development. Defining the project estimated cost, duration and maintenance effort early in the development life cycle is a valuable goal to be achieved for software projects. Model based technique is one of the best techniques used for estimation. The technique uses different parameters for estimation. Many model structures evolved in the literature. These model structures consider modeling software effort as a function of the Source Line of Code (SLOC) or Function Point Analysis. Building such a function helps project managers to accurately allocate the available resources for the project. "The most interesting difference between estimation models is between models that use SLOC as the primary input versus models that do not." The goal of the thesis is to develop a web based tool for Software project estimation by integrating the Function Points (FP) approach with a more traditional Source Lines of Code (SLOC) -based approach for cost estimation. A detailed description of the software requirements is all that is needed to conduct a complete FP analysis. Along with this a combination of estimation models: System Evaluation and Estimation of Resources and Constructive Cost Model (SEER-SEM and COCOMO II) are also evolved.

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 1, January 2012

II. Effort Estimation for Database, Reports and Documentation

Database is one of the important component of the software project. The database is created using MySQL. A database update module has been included in the software model to help the user to edit entries from the software itself so that user need not open the database tables. Validation checks have been introduced to check for correctness of the values entered into the tables. The effort required to create a Database is estimated on the basis of number of tables. The effort required to create a database includes design of tables along with the effort for training/ entering the values for the tables. It is assumed that based on experience, one effort unit will be required to create a data table including design and entries. "Reports are Mirrors" to users through which the quality of the software will be visible. Results must be tabulated in such way that analyses can be made easily. In addition to reports, facility to view bar/ pie charts or x-y graphs must be provided. It is difficult to analyze the results from large reports, but a graph/ chart is sufficient to draw a number of conclusions. The effort required to generate a report is nothing but the effort required to design a report and interlink with the database and integrate with appropriate forms associated with the report. Based on experience, it is assumed that each report generation requires about two effort units. Software is basically an invisible product. Though the code can be the studied, in most cases it cannot be easily read and understood. The software design is often not apparent from the eliminating high-risk problems by careful. This becomes a major difficulty in the maintenance of software. Therefore, considerable documentation is required for all software products. Documentation requires considerable effort and skilled manpower. Knowledge of software and working capabilities are essential to estimate the total documentation effort and to find its cost. The documents commonly produced for most projects are Software project definition Software project plan, Feasibility study, Software requirement specifications including minutes of review meetings and discussions held, Risk management plan, Software quality assurance plans, Test plan and specifications, User manuals, Installation manual, Instructors manual, Software design documents, Software maintenance manuals, Change/Version control, Acceptance report. Documentation effort is usually based number of pages of documents produced. Based on experience, on an average a person can produce 10 pages of document per day.

III. Risk Analysis Methodology

In order to analyze the impact of risk involved in the development of software, the project manager has to identify the risk drivers similar to cost drivers in COCOMO model. Software risk components can be classified as "Performance", "Support", "Cost" and\ "Schedule" risks. The degree of uncertainty that the product will meet its requirements and be fit for its intended use is the performance risk. The degree of uncertainty that the project budget will be maintained is the cost risk. The degree of uncertainty that the resultant software will be easy to correct, adapt and enhance is the support risk. The degree of uncertainty that the project schedule will be maintained and that the product will be delivered in time is the schedule risk. The performance drivers associated with "Requirements" are complexity, size, stability, post deployment support environment, repair maintenance. The performance drivers associated with" Constraints" are computer resources, personnel, standards, and environment and performance envelopes. The performance drivers associated with "Technology" are language, hardware, tools, data rights, experience. The performance drivers associated with "Development approach" are prototypes and reuse, documentation, environment, management approach, integration. The cost drivers associated with "Requirements" are size, resource constraints, application, and technology and requirements stability. The cost drivers associated with "Personnel" are availability, mix, experience, and management environment. The cost drivers associated with "Reusable software" are availability, modifications language, rights, and certification. The cost drivers associated with "Tools and environment" are facilities, availability, rights, and configuration management. The support drivers associated with "Design" are complexity, documentation, completeness, data rights, configuration management and stability. The support drivers associated with "Responsibilities are management (hardware, software), configuration management, software identification, and technical management and change implementation. The support drivers for "Tools and environment" are facilities, software tools, computer hardware, production and distribution. The support drivers for "Supportability" are changes, operational interfaces, personnel, release cycle and procedures. Schedule drivers for "Resources" are personnel, facilities and financial. The schedule drivers for "Need dates" are threat, economic, political, tools. The schedule drivers for "Technology" are availability, maturity and experience. The schedule drivers for "Requirements" are definition, stability, and complexity. The impact of each risk driver on the risk components is divided into four categories negligible, marginal, critical and catastrophic. The probability of occurrence for each driver for each risk component is then determined. The risk projection activities that are to be performed are:

- Establish the scale that reflects the perceived likelihood of a risk.
- Delineate the consequences of the risk
- Estimate impact of risk on the project.
- Note the overall accuracy on the risk projection so that there will be no misunderstandings. One of the approaches is to assess risk drivers on a qualitative probability (p) scale – impossible, improbable, probable and frequent. One of the approach [5] is to specify value = 0 or 1 (if $p < 0.4$) for impossible or improbable risk event, value =2 (if $0.4 < p < 0.7$) for a probable risk event and value = 3 (if $0.7 < p < 1.0$) for frequently occurring risk event. Risk analysis approach

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 1, January 2012

proposed by US Air Force recommends determining the over all effect of risk as per the following steps:

1. Determine the average probability of occurrence value for each risk component
2. Determine the impact assessment for each component based on the criteria decided
3. The probability and impact have been determined in above steps. Determine the overall risk (high, moderate, low or none) associated with the risk component. Risk drivers can be supplemented based on the local conditions and environment. Specific project and local conditions also decide importance (values) to be given for each risk drivers. An average value for each of the four risk components (performance, cost, support and schedule) can then be computed.

IV. EFFORT ESTIMATION OF CBSD

Owing to the extensive use of existing components, the CBSD process is quite different from traditional software development. As opposed to the development-centric activities of traditional software development, CBSD requires focus on integration-centric activities namely, searching and identifying candidate components, assessing and selecting components based on system requirements and architectural and project constraints, tailoring and integrating the selected components into a seamless software system and upgrading the system as components evolve over time with newer versions . Therefore when predicting the cost and schedule of CBSD, the cost drivers associated with these new development activities must be taken into the account. This is supplemented by various other factors to be considered including, but not limited to, the costs of licensing and redistribution rights for components, component certification costs for compliance with critical requirements and indemnification costs for faults or damages caused by third-party components. Moreover, the complications in sizing component-based systems (CBS) because of the black-box nature of components and the lack of CBS metrics make it an even more challenging task. Thus, the introduction of CBSD has also brought a host of unique challenges to software effort estimation that are quite different from those associated with traditional software development. Despite its brief history, a considerable amount of research has already been directed towards CBSD. This includes a wealth of research on defining fundamental concepts, producing component evaluation and integration techniques, developing methodologies and processes for component development and formulating metrics for assessing traits of components and CBSSAIC However, only a handful of effort models have been proposed to address the problem predicting the cost and schedule of CBSD. In fact, a majority of such approaches are algorithmic models. Fig.1shows a classification of these models into three major categories on the basis of their modeling technique used. Here we have followed the same classification that was used by Boehm et al. to classify the software effort models in traditional software development [1] [4].

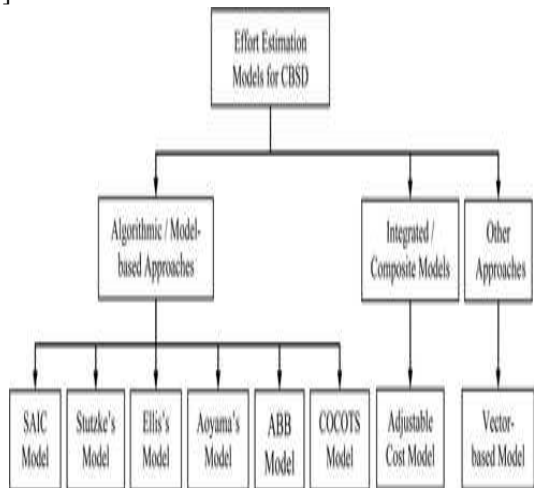

Fig.1. Classification of Models

V. ALGORITHMIC/MODEL-BASED APPROACHES

The algorithmic/model-based approaches for estimating the effort of software development use some sort of mathematical relationship between dependent and independent parameters to predict the estimates. The independent parameters normally characterize the software to be developed and the environmental conditions under which the development will be performed and the system will be operated whereas the dependent parameter corresponds to the effort estimated. The mathematical relationship is usually determined by the statistical analysis of data from past projects. During our survey of research literature on effort estimation approaches for CBSD, we found six effort models that can be classified into this category.

A. SAIC model:

The first model found in research literature on estimating the effort of CBSD is the SAIC model developed at the Science Applications International Corporation (SAIC), California in the early 1990s .The model focuses mainly on the end user cost of adopting a particular component into a larger system.

$$EC = LC \times N + TC + GC$$

Where EC is the estimated cost and LC, N, TC and GC stand for component licensing cost, number of licenses required, component training cost and glue code development cost, respectively. The SAIC model underscores some of the important cost factors of using components such as licensing costs and training costs. However, it does not take the effort of searching and selecting components into account. Moreover, the model does not provide details of determining the effort of glue code development.

B. Stutzke's model:

Stutzke of SAIC proposed another effort model for CBSD. This model is mainly frequency with which a vendor releases new versions of its components. The model estimates the additional cost of using a given component with a significant volatility

$$EAC = CV \times AC \times IS\ (CS + CC)$$

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 1, January 2012

where EAC is the estimated additional cost of using a given component and CV, AC, IS, CS and CC stand for the component's volatility over the system's life, architectural coupling of the component, interface size of the component, cost of screening the component and all other components with which it interfaces and the cost of making changes to the impacted components, respectively. Component volatility is one of the two primary determinants in the cost of using software components whereas the other is the actual size of the glue code written to integrate them. However, component volatility is only one of the factors to be considered when predicting the effort of CBSD. Moreover, there has been no attempt made to implement this model.

C. Ellis's model:

Ellis of the then Loral Federal Systems, New York (now Loral Space and Communications), proposed some 17 cost drivers and described the construction of an effort model for predicting the effort of component integration. This model takes the following form

$$WU = fn(\text{size, drivers})$$
$$P = LM/WU$$
$$\text{Estimated Cost} = WU \times P$$

Where P and LM denote productivity and labour months, respectively, WU stands for work units, a measure internal to Loral and fn is a function that relates the size of glue code and ratings of cost drivers to work units. Ellis used function point (FP) analysis to estimate the glue code size. Unlike other models, Ellis's model is an actual database application with a graphical user interface. It has been calibrated to a number of CBSD projects and claims a focused on the volatility cost of components. The component volatility is defined as the general description found in, the deep details its modeling function and calibration dataset remain proprietary and are not available in the public domain.

D. Aoyama's model:

Aoyama of the Software Continuous Acquisition and Lifecycle Support consortium, Japan, reported four major differences between conventional software development and CBSD process models. They include newly introduced component acquisition, compositional design and component integration processes and wiped out unit testing process. Taking these differences between two process models into account, Aoyama proposed an economic model for CBSD.

E. ABB model:

Dagnino et al. of ABB Corporate Research Laboratories, Raleigh, proposed another economic model for CBSD. They used the goal-question-metrics approach for the construction of this model. First, they identified two goals to evaluate the benefits of CBSD. The two goals are: Goal 1: evaluate whether there is a reduction in cost as a result of using CBSD; and Goal 2: evaluate whether there is a reduction in effort as a result of using CBSD. Then they defined questions that have to be answered to satisfy the two goals. The answer to each question could be obtained by asking several sub-questions. A question that does not have any further sub-questions corresponds to a derived metric. The meaning of each metric is finally explained. To define questions and metrics, they proposed a measure called measurable unit (MU) and formulated the following relations

TENC = size of any fraction of system Developed by custom code in MU

F. COCOTS model:

The most comprehensive approach published accuracy of +15% against an internal dataset of Loral projects. However, apart from some of to date on estimating the effort of CBSD is the constructive commercial off-the-shelf (COCOTS) integration cost model developed at the University of Southern California as an extension of the well-known COCOMO II model. The model is based on two defining characteristics of components: (i) the source codes of components are not available to the application developer; and (ii) the future evolutions of components are beyond the control of the application developer. The COCOTS model has been composed with three sub-models that estimate the efforts of component assessment, tailoring and integration activities of CBSD.

G. Integrated/composite models:

Integrated/composite models for predicting the software development effort incorporate a combination of two or more modeling techniques and formulate the most appropriate functional form for estimation. The details of one integrated effort model for CBSD can be found in the research literature available in the public domain.

H. Adjustable cost model:

Naunchan and Sutivong proposed an adjustable cost model for estimating the effort and duration of the component integration. The model integrates three existing approaches, namely effort multipliers of the COCOTS model to identify and determine productivity factors, system dynamics to simulate the software process and communication overhead assumptions to adjust the development team's productivity.

I. Other approaches:

Other software effort estimation models where the modeling technique cannot be classified elsewhere such as vector based, lexical analysis and linear programming approaches are discussed in this section. During our survey, we found details of one vector-based effort estimation approach proposed for CBSD. Effort in terms of any measurable unit such as man hours.

J. Vector-based approach:

Yakimovich et al.presented a somewhat different approach for estimating the effort of component integration. The rationale of this approach is to account for the increase of effort for writing wrappers or adapters by means of glue code to make the interaction assumptions of components compatible with those of the system's architecture to which components are integrated. In this approach, the interaction assumptions of individual components and system's architecture are presented as interaction vectors V = (P, C, I,

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 1, January 2012

S, B) where variables P, C, I, S and B represent the inter-component interaction assumptions for packaging, control, information flow, synchronization and binding, respectively. The component integration effort is then estimated by comparing VS and VP, the interaction vectors for the system's architecture and component, respectively. If VS = VP for all elements, the integration is economical because effort is required only to match the syntax and semantics of their interactions. If VS ≥ VP, the integration effort is still acceptable as the majority of assumptions of components are still compatible with those of the architecture.

However, if VS ≤ VP or VS _ VP, a considerable effort is required to overcome the architectural mismatches. In this case, another vector called VC, such that VC ≥ VS and VC ≥ VP is defined. Then the minimum integration effort is estimated by finding the minimal common upper element VC for VS and VP in the interactions vector space. A remarkable feature of this approach is that the information required to employ it can be easily obtained from the system's architectural description and components' specification. Furthermore, this approach can be used to decide whether or not to go for a CBS solution, to select the best components and to determine the amount and type of the glue code to be written. However, this vector-based approach only provides comparative results for the component integration but not the required integration.

VI. CONCLUSION

In this paper, we have presented a literature survey of the most up-to-date research work published on the effort estimation of CBSD. The effort invested in a software project is probably one of the most important and most analyzed variables in recent years in the process of project management. It highlights the strengths and weaknesses of the discipline and analyses research work in terms of the modeling technique used, kind of data required for their use, type of estimation provided, lifecycle activities covered and the level of their acceptability with regard to any validation. The paper has also attempted to identify the remaining challenges to be addressed and promising directions for future research.

Effort estimation is a vital input for any economic analysis. Therefore constructing a sound business case either for or against CBSD requires economic models that can deal with the peculiarities of CBSD. In this sense, much work still remains to be done to support practitioners make many crucial decisions in relation to CBSD. More research on new metrics, effort models, techniques and tools dedicated to CBSD will only help in this endeavor. There is no quick fix that will immediately make us better estimators and users of estimates. Effective estimates come about as a result of process definition and improvement, education and training, good project management, use of proper tools and techniques, measurement, sufficient resources, and sheer hard work.

REFERENCES

[1] B.Boehm, Software engineering, Economics Prentice Hall, (1981).

[2] J.Verner and G Tate, Estimating size and effort in fourth generation development-IEEE Software 5(4), (15 July 1988).

[3] Roger S. Pressman, Software engineering – a practitioner's approach, McGraw- Hill, (1992).

[4] Waman.S.Jawadekar, Software engineering, principles and practices, McGraw-Hill, (2004).

[5] Roger S Pressman, A Manager's guide to software engineering, TataMcGraw-Hill Edition, (2005).

[6] Ravichandran, T., Rothenberger, M.A.: 'Software reuse strategies and Component markets', Commun. ACM, 2003, 46, (8), pp. 109–114.

[7] Mahmood, S., Lai, R., Kim, Y.S.: 'Survey of component-based software development', IET Softw., 2007, 1, (2), pp. 57–66.

[8] Yang, Y., Boehm, B.W., Clark, and B.: 'Assessing COTS integration risk using cost estimation inputs'. Proc. 28th Int. Conf. on Software Engineering, (ICSE 2006), Shanghai, China, May 2006, pp. 431–438.

[9] Yang, Y., Boehm, B.W., Wu, D.: 'COCOTS Risk analyzer'. Proc. Fifth Int. Conf. on Commercial-off-the-Shelf (COTS)-Based Software Systems, (ICCBSS 2006), February 2006, pp. 144–151.

[10] Software effort estimation and risk analysis – A case study, IET-UK International Conference on Information and Communication. Technology in Electrical Sciences (ICTES 2007), Dr. M.G.R. University, Chennai, Tamil Nadu, India. Dec. 20-22, 2007. pp. 1002-1007.

[11] Effort estimation of component-based software development – a survey T. Wijayasiriwardhane1 R. Lai1 K.C. Kang2 Department of Computer Science and Computer Engineering, La Trobe University, Victoria 3086, Pohang University of Science and Technology (POSTECH), Pohang, Korea Published in IET Software Received on 26th June 2009 Revised on 4th February 2010 pp 2-6..

Poonam Kaushal received her B.E in Computer Science & Engineering from Truba college of Engineering and Technology Indore, India in 2009. She is pursuing M.Tech from LNCT, RGPV, and Indore (M.P.) in Computer Science & Engineering. Her areas of interest are Software Engineering, Soft Computing and computer network. She has published and presented a research paper on Effort estimation and brain operational architecture in a national conference.