Challenges of Software Metrics Estimation

Sanjana Shashi Kadapa

Abstract- One of the problems in software development is to estimate the efforts in person hours required to develop the software application. Software developers quote their rates in dollars/ hour. However, the customer must have an idea of the cost of development. If the rate quoted is high, then the customer may turn to another developer. If the rate quoted is too low, then the software developer incurs losses. This paper examines the challenges of software estimation and suggests metrics used to size the software.

Index Terms- software metrics, function-point, coding efforts, effort estimation

1. INTRODUCTION

Software Estimation is the science of closely estimating the time and efforts required by developers and programmers to complete a software project. The modern software development process has provides detailed methodologies and practices that allow estimation and forecasting the time and efforts required for design, coding and testing activities. With correct estimation, the project manager is able to plan the project, arrange for resources and ensure on time deliveries. Software estimation when done properly stops overruns on project schedules and costs, and helps to keep the development costs within the specified parameters and helps to increase the profitability of the company (Moses, 2002). This paper discusses various aspects and challenges of software estimation.

2. LITERATURE REVIEW

There are many important aspects that should be understood before discussing the finer aspects of software estimation. A few basic concepts are presented in this section.

2.1 Need and Challenges for Software Estimation

Grimstad (2006) has cited a recent report by the Standish report which indicates that only about 16% of software projects are completed on time with the allotted resources. About 41% of the projects are cancelled and 53% exceeded the allotted budget, required more resources and incurred extra expenses and were considered a failure.

Current software applications are highly specialized and use a wide variety of technologies, domains, tools and languages. A complex banking solution may use Java, JSP, PHP as well as C#, Oracle databases running on Windows or Linux platforms. Software estimation ensures that all phases of the software delivery cycle are calculated and analyzed properly so that the project manager is able to say with certainty the amount of time required for writing so many lines of code, development of the GUI, development and deployment of the database, testing and the final delivery time. Proper software estimation ensures that client is satisfied with on time delivery without using extra resources, resources are not under or overestimated, and that the projected profit margins are retained (John, 2002).

2.2 Challenges for Software Estimation

Agarwal (2001) of Infosys has pointed out that the automotive industry and other automated industries are very precisely able to predict and measure their manufacturing activity because of the nature of repetitive work, use of sophisticated machines that optimize loading and machining time and the cycles that is built into the system. The software industry on the other hand works on diversity and each project is unique since the requirements are different, software functionality are different, a wide variety of platforms are used, and developers may need a high level of learning and drive to make them complete on time deliveries. As such, the capital investment in terms of machinery and bought out software is far less when compared to the costs of buying an expensive machine for the automobile industry. Software coding is a manual process and developers need to type the code and the whole process depends on the skill of the developer. The more experience a developer has, the faster and more accurate the coding can be done. The authors have reported that the amount of reusable code in an existing application is very little and at a practical level, it is not possible to copy/

paste code from an existing application and use it for different applications since the amount of recoding and associated human error would be significant.

John (2002) on the other hand argues that the main challenges in software estimation are: understanding the factors to be analyzed, defining terminology and measures, deciding the strategy for isolation, measuring the estimation error and collecting data required for isolation of factors to be analyzed, and interpretation of the measured estimation error. In addition there are a number of other challenges are estimation ability factors, estimation complexity factors and the measurement process factors. Certain benchmarks and measures are usually set in an organization with regards to the estimation of efforts and subsequently cost. The estimation ability factors depend on expert judgment skill and an expert in the field does this. There are also a number of estimation models that actually transfer the skills of the expert to a layperson by using structured methodology.

2.3 Concepts of Metrics

Qin (2005) speaks of the use of metrics in software estimation. Metrics are measures and estimates used to specify the resources used in the software development process. Items that can be measured directly include the lines of code, programming and building the database, creating the GUI elements, integrating all components of the system, deploying the build, testing, rectifying the errors and the final delivery. Metrics cover four functions of the software development cycle and they are planning, organizing, controlling and improvement. In the planning function, Metrics form the basis of estimating costs, planning for training and resource development, budgeting and scheduling and this function sets the overall scope, cost and time to deliver the project. The organizing function uses Metrics to schedule and size the metrics influence the organization. In controlling phase, Metrics serve to find the status and keep track of the different activities and help to determine if the project in on track. In the improvement function, Metrics are used to improve the process and identify areas where the improvement efforts have to be used. Appendix Table 1 provides an overview of different Metrics associated with a project.

2.4 Formulas for Metrics Calculation

The Software Productivity Centre Inc (SPC 2007) has provided a number of formulas that can be used to calculate different metrics. The common formulas are:

<u>Code Growth</u>: The Code Growth – GC metrics is calculated as
CG = (latest_size_estimate)-(Initial_Size_Estimate)/ (Initial_Size_Estimate)

<u>Earned Value</u>: Earned value - EV is a very key tracking metric that measures the actual amount of work done irrespective of the effort or the time used. It is based on the Overall Proportion Complete - OPC. The inputs required are the labor wages for each task and the proportion of software that has been done in each activity.
EV = units_through_activity/ total_number_of_units_required

<u>Estimate at Completion</u>: EAC is an estimate of the completed cost of the project. Initially the EAC is equal to the initial estimated cost IEC. After the project has begun, EAC is equal to the cost to date CTD and the estimated cost to complete ETC.

$$EAC = CTD + ETC$$

<u>Labor Rate</u>: It is the cost measured as effort to put one unit of product size through an activity. Each activity has its own labor rate. Common ratios used are PH/SLOC or person hours per line of source code or PM/KSLOC person months per thousand lines of source code.

<u>Overall Proportion of Work Complete</u>: OPC gives the degree of amount of completeness of a project. It requires inputs such as the labor rate for each activity and the proportion of software that as passed through each activity.

<u>Productivity</u>: This is a measure of how efficient the labor is. It is specified as the inverse of the labor rate and is specified by the number of size units that can be put through an activity with a given effort. It is measured as SLOC/PH or lines of source code per person hour

2.5 Function Point Analysis

Function Point Analysis is methodology prescribed by ISO and used to estimate the functional size of IT systems. It determines the actual amount of functionality that is required by a user of the software application and to a great extent, this depends on the technology used. This technique is used to determine the budgets and cost, estimation of maintenance costs of an existing project, to find the productivity of the project after it is completed, and also to find the software size that is used for estimating costs. FPA is a method to break systems into smaller components, so they can be better understood and analyzed. It also provides a structured technique for problem solving and helps to define two abstract levels of data - data at rest and data in motion. Function points are a unit measure for software much like an hour is to measuring time, miles are to measuring distance or Celsius is to measuring temperature. Function Points are ordinal measures much like other measures such as kilometers, Fahrenheit, hours, so on and so forth. (Longstreet 2004)

.

3. EVALUATION AND ANALYSIS

The actual application of software estimation is presented in this section with examples. There are essentially two types of software projects: development software projects where a new application is developed from scratch and maintenance software projects, where an existing application is maintained, upgraded or migrated. The estimation and methodology is different in both cases (Jørgensen, 2007).

3.1 Estimation Models

Fairley (1992) and Fujiwara (1998) have suggested some modes that can be used in the proper estimation of different types of software projects. Some assumptions are made during estimation and they are source code lines are written by the developers and any source code lines that the application generates are not considered, one instruction is one life of code; declarations are regarded as instructions and code comments are not taken as instructions. The authors have also suggested that projects can be categorized into different modes such as Development Mode; Organic Mode and Embedded Mode. In the development mode, the project is considered to be developed from unknown and base level, in Organic Mode, the project is developed in a familiar and stable environment and similar products have been already developed such as financial or accounting systems. In the Embedded Mode, the project has tight and inflexible constraints and interface requirements and is heavily dependent ton innovation and creativity.

According to Agarwal (2001) and Bhat (2004), there are three types of models used in estimation and they are the Basic Model, the Intermediate Model and the Detailed Model. The basic model makes the estimates of required effort that is measured in person months, based on the estimate of the project size measured in thousands of source lines of code or KSLOC.

The following table gives the Basic Effort equation for the three modes.

Development Mode	Basic Effort Equation
Organic	Effort = 2.4 * $(KSLOC)^{1.05}$
Semidetached	Effort = 3.0 * $(KSLOC)^{1.12}$
Embedded	Effort = 3.6 * $(KSLOC)^{1.20}$

In the basic model, the estimates for the time to develop - TDEV is given as:

Development Mode	Basic Effort Equation
Organic	TDEV = 2.5 * $(KSLOC)^{0.38}$
Semidetached	TDEV = 2.5 * $(KSLOC)^{0.35}$
Embedded	TDEV = 2.5 * $(KSLOC)^{0.32}$

To illustrate an example in the Organic Mode for estimating a project of 3000 source lines of code, the estimation for effort and TDEV is:
Effort = 2.4 x 31.05 = 7.6 person months
TDEV = 2.5 c 7.60 .38 = 5.4 months
Average staffing = 7.6/ 5.4 = 1.4 persons

3.2 Estimation in Development Software Projects

Agarwal (2001), Bhat (2004) have suggested that a structured methodology should be used when estimating a new software development project. There are many unknowns in this process and if the organization is attempting to use new technologies then the chances for under or over estimation can increase. In addition, there can be changes in the requirements that the customer may specify from time to time. A flexible approach should be used in designing the components so that they can be scaled to accept the changes without any major restructuring. The authors have also suggested that the specifications as of date should be specified in the sign off agreement contract and that both parties should have the option of asking for a re-estimate when the requirements are changed. Figure 3.1 shows the activities in software estimation.

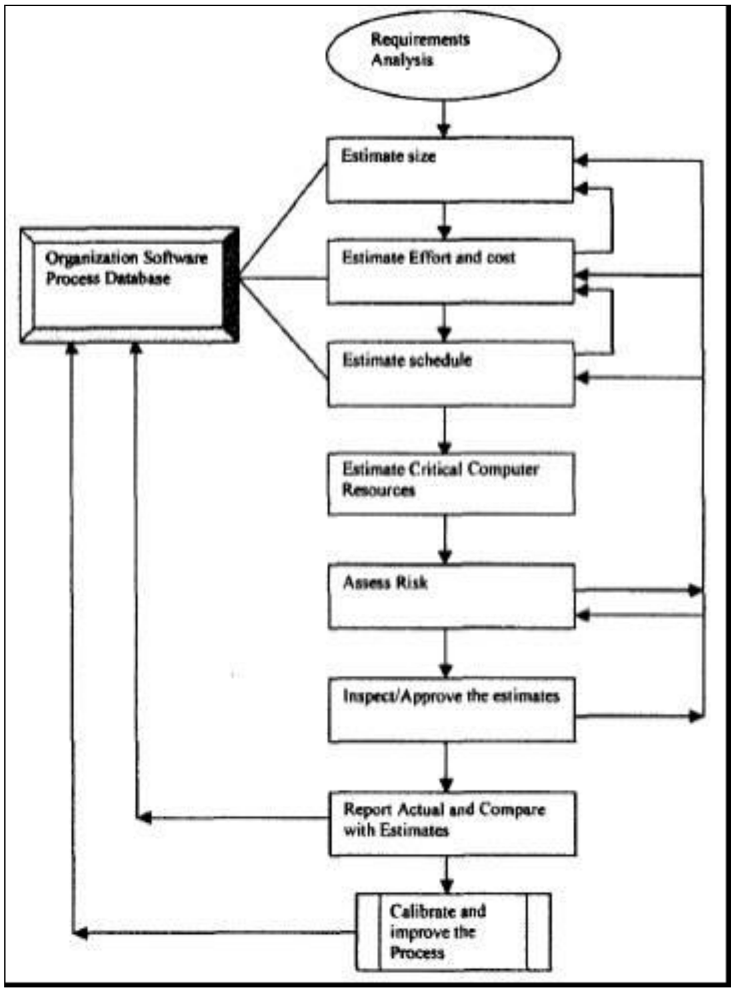
Requirements Analysis
Estimate size
Estimate Effort and cost
Estimate schedule
Organization Software Process Database
Estimate Critical Computer Resources
Assess Risk
Inspect/Approve the estimates
Report Actual and Compare with Estimates
Calibrate and improve the Process

Figure 3.1 Activities in Software Estimation (Agarwal et al., 2001)

The estimation activities start the requirement analysis where the functional requirements are studied in great detail. There are a number of nested activities in this stage and they include estimation of size, efforts and costs; schedule estimation and estimation of critical computer resources. Along with these activities, risk assessment also needs to be done. After the estimates are given, they are inspected and approved. The approved estimates should be compared with actual estimates and if required, the estimates should be calibrated and improved.

3.3 Estimation in Maintenance Software Projects

Bhat (2004) has suggested that with the increase in computerized systems, more and more applications have stabilized and need to be maintained, and this is an activity that is increasingly outsourced. Estimation of the maintenance efforts become complicated since the software is ready and there are frequent issues of minor development and coding activity and the actual maintenance activity. Other factors such as management focus, attitude of the client, need for multi location teams and so on make the maintenance efforts almost intractable. If the efforts are estimated high, the outsourcing company loses money since it pays too much for too little and if the estimated efforts are low, then the vendor loses money as they would do too much for too little. When mutual agreement is not convincingly reached, the results are poor and sloppy maintenance support with wide spread client dissatisfaction. IEEE has defined software maintenance work as 'the modification of a software product after delivery to correct faults, improve performance and other attributes or to adopt the product to a modified environment’. Figure 3.2 illustrates a model for the maintenance of software projects.

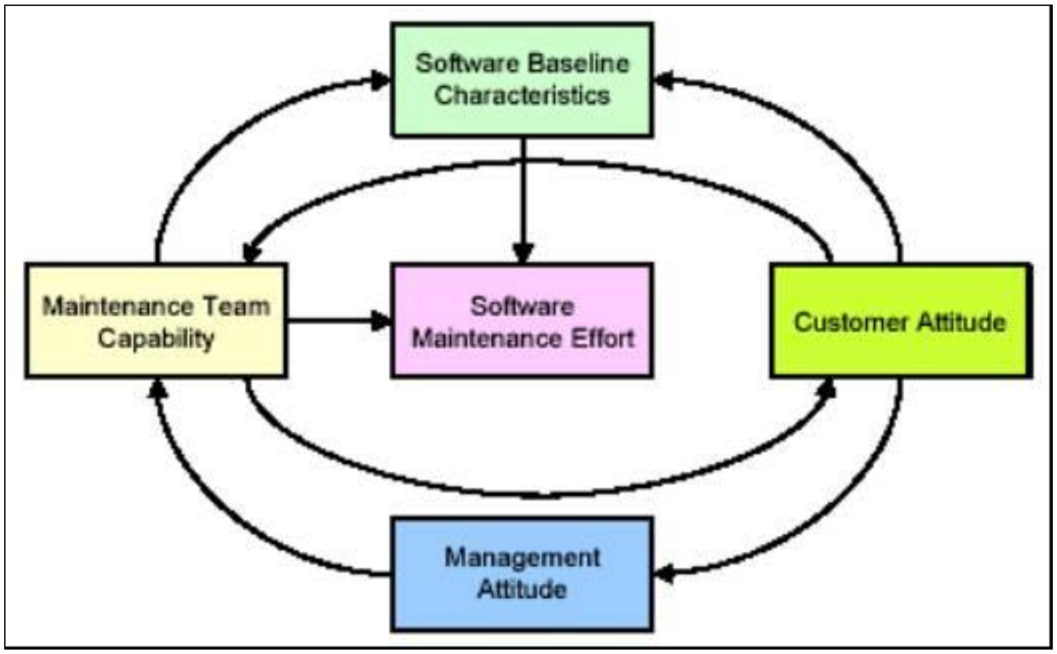
Software Baseline Characteristics
Maintenance Team Capability
Software Maintenance Effort
Customer Attitude
Management Attitude

Figure 3.2 Estimation Model for Maintenance Projects (Bhat, 2004)

The systemic dynamics model is based on the interactions of four parameters: maintenance team capability, software baseline characteristics, customer attitude and the management attitude. Scope Creep is a very active component that serves to spoil the relations between vendors and organizations. It is the attitude to include additional items in the scope of the maintenance work by exploiting competitive market conditions.

3.4 Limitations of Estimation

Laird (2006) speaks of certain limitations that are inherent in software estimation process. The author states “Estimates are typically the 50-percent view, meaning the probability for being under or over budget is the same.” There are certain dependencies that make the actual estimation a guess work and they are: Specifying unclear and incomplete requirements; inability to change and adjust work schedule when the scope is changes suddenly; self setting very aggressive development schedules and less number of people allotted.

King (2006) has analyzed the main reasons why estimations are inaccurate and the reasons for limitations. The main reasons are: lack of training in estimation; disagreement and confusion over the required schedule and effort target with the estimate; resorting to hop based planning; inability on the part of the software personal to properly communicate and support the estimates; incomplete change in requirements and scope creep and the worst reason - improperly coding and testing without the proper application if use cases that makes the software unusable.

4. CONCLUSION

Software Estimation is an important process as it allows projects to maintain their resource and cost allocation within the budget. A properly estimated project helps in increasing customer satisfaction by providing on time deliveries of the required quality of products. The main aspects for estimation rely on calculation of the resources specified by the estimate of the project size measured in thousands of source lines of code or KSLOC and the time required in person months or person hours. There are different models used for software estimation and they are basic, intermediate and the detailed models and they have different modes such as development, organic, semidetached and embedded. The paper has also examined the estimation model for new project development and maintenance projects and has also discussed the limitations of software estimation. The paper can serve as a knowledge base for students who wish to take up software estimation as a career.

APPENDIX

The following table presents the suggested set of metrics.

Indicator Category	**Management Insight**	**Indicators**
Progress	Provides information on how well the project is performing with respect to its schedule.	Actual vs. planned task completions Actual vs. planned durations
Effort	Provides visibility into the contributions of staffing on project costs, schedule adherence, and product quality.	Actual vs. planned staffing profiles
Cost	Provides tracking of actual costs against estimated costs and predicts future costs.	Actual vs. planned costs Cost and schedule variances
Review Results	Provides status of action items from life-cycle review.	Status of action items
Trouble Reports	Provides insight into product and process quality and the effectiveness of the testing.	Status of trouble reports Number of trouble reports opened, closed, etc. during reporting period
Requirements Stability	Provides visibility into the magnitude and impact of requirements changes.	Number of requirements changes/clarifications Distribution of requirements over releases
Size Stability	Provides insight into the completeness and stability of the requirements and into the ability of the staff to complete the project within the current budget and schedule.	Size growth Distribution of size over releases
Computer Resource Utilization	Provides information on how well the project is meeting its computer resource utilization goals/requirements.	Actual vs. planned profiles of computer resource utilization
Training	Provides information on the training program and staff skills.	Actual vs. planned number of personnel attending classes

REFERENCES

1) Agarwal R, Manish Kumar T, Yogesh, S (2001), 'Estimating software projects', Journal of ACM SIGSOFT Software Engineering Notes, Volume 26, Issue 4, pp: 60-68
2) Bhatt Pankaj, Shroff Gautam, Misra Arun K (2004), 'Dynamics of Software Maintenance', Journal of ACM SIGSOFT Software Engineering Notes, Volume 29, Issue 5, pp: 1-5
3) Fairley Richard E. (1992), 'Recent Advances in Software Estimation Techniques', Software Engineering Management Associates, US, pp: 1-10
4) Fujiwara Fumiko, Goto Takushi, Araki Sadao (1998), 'Examples of Applying Software Estimate Tool', IEEE Publication, O-8186-8368-6/98
5) Grimstad Stein, Jørgensen Magne (2006), 'A Framework for the Analysis of Software Cost Estimation Accuracy', ISESE Publications 06, September 21–22, 2006, Rio de Janeiro, Brazil
6) Jørgensen Magne, Shepperd Martin (2007), 'A Systematic Review of Software Development Cost Estimation Studies', IEEE Transactions on Software Engineering, Volume 33, Issue 1, pp: 33-55
7) King Julia (2006), 'Demystifying Software Estimation', Computerworld, Volume 40, Issue 21, p: 58-67.
8) Laird Linda M. (2006), 'The Limitations of Estimation', IT Pro: IEEE Computer Society, pp: 40-47
9) Longstreet David (2004), 'Function Point Training and Analysis Manual: Longstreet Consulting Inc', Retrieved 4 August 2016 from http://www.softwaremetrics.com/Function%20Point%20Training%20Booklet%20New.pdf
10) Metrics (2007), 'Software Metrics Guide', Retrieved 5 August 2016 from http://sunset.usc.edu/classes/cs577b_2001/metricsguide/metrics.html
11) Moses John (2002), 'Measuring Effort Estimation Uncertainty to Improve Client Confidence', Software Quality Journal, Volume 10, pp: 135-151
12) Qin Liu, Mintram Robert C (2005), 'Preliminary Data Analysis Methods in Software
13) Estimation', Software Quality Journal, Volume 13, pp: 91-115
14) SPC (2007), '8-Step Metrics Program', Retrieved 5 August 2016 from http://www.spc.ca/resources/metrics/metrics_formulas.pdf

AUTHORS

First Author – Sanjana Shashi Kadapa, Bachelor of Engineering (Computer Science), Cummins College of Engineering for Women, sanjanakadapa@gmail.com.

	After completing her Bachelor of Engineering in Computer Engineering from Cummins College of Engineering for Women, Pune University, Sanjana took up community service and taught several underprivileged children about computers and web technology. Since August 2015, she works as a software developer with HSBC - Global Technology Center with the Data warehousing practice. She is planning for a MS in computer science in USA.

Correspondence Author – Sanjana Shashi Kadapa, sanjanakadapa@gmail.com.