E-ISSN: 2706-8927
P-ISSN: 2706-8919
www.allstudyjournal.com
IJAAS 2020; 2(4): 511-515
Received: 09-10-2020
Accepted: 19-11-2020

Hamayoon Ghafory
Assistant Professor,
Faculty of Computer Science,
Information System
Department, Shahid Proff.
Rbbani Education University,
Kabul, Afghanistan

Faqeed Ahmad Sahnosh
Faculty of Computer Science,
Information System
Department, Shahid Proff.
Rbbani Education University,
Kabul, Afghanistan

Corresponding Author:
Hamayoon Ghafory
Assistant Professor,
Faculty of Computer Science,
Information System
Department, Shahid Proff.
Rbbani Education University,
Kabul, Afghanistan

The review of software cost estimation model: SLIM

Hamayoon Ghafory and Faqeed Ahmad Sahnosh

Abstract
Software cost estimation is the process of predicting the amount of effort required to build a software system. Models provides one or more mathematical algorithms that compute cost as a function of a number of variables, size is primary cost factor in most models and can be measuring using lines of code (LOC). The models should be used to estimate the cost of software, SLIM is a useful model to estimate the cost and it is a good model for big projects, also this model required some parameters to generate an estimation. Software cost estimation always characterized one of the biggest challenges in Computer Science for the last decades. Because time and cost estimate at the early stages of the software development are the most difficult to obtain, and they are often the least accurate. Traditional algorithmic techniques such as regression models, Software Life Cycle Management (SLIM), require an estimation process in a long term.

Keywords: Software cost estimation, SLIM model

Introductions
Software Cost estimation is the process of predicting the amount of effort required to build a software system. Cost estimates are needed throughout the software lifecycle. Preliminary estimates are required to determine the feasibility of a project. Detailed estimates are needed to assist project planning. The actual effort for individual tasks is compared with estimated and planned values, enabling project managers to reallocate resources when necessary. Most of the work in the cost estimation field has focused on algorithmic cost modeling (Jamil, 2007) [6]. Software development effort estimation deals with the prediction of the probable amount of time and cost required to complete the specific development task. Generally, software development effort estimations are based on the prediction of size of software, which is a very difficult task in the sense that estimates obtained at the early stages of development life cycle are inaccurate because not much information of the system is available at that time (Attarzadeh and Ow, 2009) [1]. Several cost estimation techniques have been used in the last few decades. These techniques include algorithmic models, expert judgment, and estimation by analogy and machine learning. Algorithmic models are the oldest and most famous methods used in software cost estimation. Examples of algorithmic models includes SLIM. In algorithmic models, linear and non-linear regression models are used to predict software effort. The dependent variable is usually software effort, where the independent variables include software size and some non-functional requirements (Nassif, 2012) [8]. The reliable and accurate cost estimation in software engineering is an ongoing challenge due to it allows for considerable financial and strategic planning. Software cost estimation techniques can be classified as algorithmic and non- algorithmic models. Algorithmic models are based on the statistical analysis of historical data (past projects), for example, Software Life Cycle Management (SLIM). Non-algorithmic techniques are based on new approaches such as, Parkinson, Expert Judgment, Price-to Win and machine learning approaches. Machine learning is used to group together a set of techniques that represent some of the facets of human mind. For example regression trees, rule induction, fuzzy systems, genetic algorithms, artificial neural networks, Bayesian networks and evolutionary computation. The last five of these approaches are classified as soft computing group. The remaining paper is structured into the following sections: section 2 Problem issues in software cost estimation; section 3 Software cost estimation model ; section 4 over view of SLIME; Section 5 advantage and disadvantage of SLIM; section 6 SLIM Method; section 7 conclusion.

Problems in software cost estimation

Software cost estimation is the most complex activity that requires a deep understanding and knowledge of a number of key attributes about the project for which the estimate is being constructed. Software cost estimation seems to be much difficult as they are roughly estimated at the start without knowing the actual facts and figures. Moreover at the start of the project it is difficult to list down all requirements and specifications correctly and precisely, abrupt change to requirement is also a big hurdle in estimating a good software cost. There are numerous factors that affect efforts and time to develop software these factors include complexity of software, experience of development team and commitment to work and quality of requirements specifications plus changes in Information Technology (IT) and methodologies of software development cause instability of cost estimation process. Experience of an estimator (mostly the project manager) comes in to play in developing estimates especially for large projects. As he tries to bid on a certain project relatively low in order to get it accepted which is a core example of poor project cost estimation. In contrast to complex and big projects smaller or ordinary projects have more than 95% of associated effort and cost accuracy.

Principle of SCE Models

Most models found nowadays are two-stage models'. The first stage is a size and the second stage provides a productivity adjustment factor. In the first stage an estimate regarding the size of the product to be developed is obtained. In practice several sizing techniques are used. The most well-known sizes nowadays are function points" and lines of code". But other sizing techniques like 'software science!' and DeMarco's Bang method":", have been defined. The result of a sizing model is the size/volume of the software to be developed, expressed as the number of lines of source code, number of statements, or the number of functions points. In the second stage it is estimated how much time and effort it will cost to develop the software of the estimated size. First, the estimate of the size is converted into an estimate in nominal man-months of effort. As this nominal effort takes no advantage of knowledge concerning the specific characteristics of the software- product, the way the software-product will be developed and the production means, a number of cost influencing factors (cost drivers) are added to the model. The effect of these cost drivers must be estimated. This effect is often called a productivity adjustment factor. Application of this correction factor to the nominal estimation of effort provides a more realistic estimate. A phase distribution and sensitivity/risk analysis com potent are distinguished. In the phase distribution com potent the total effort and duration is split up over the phases and activities of a project. This division has to be based on empirical data of past projects. The sensitivity and risk analysis phase supports project management - especially at the start of a project when the uncertainty is great - in determining the risk factors of a project and the sensitivity of the estimates to the cost drivers settings. Again data on past projects provide an important input for this component. Before using a model for the first time validation is necessary, and it may also be necessary to calibrate the model. Mostly the environment in which the SCE model has been developed and the database of completed projects on which the model is based will differ from the project characteristics of the environment(s) in which the model is to be used. To make validation and calibration possible, data on historical projects have to be available in an organization. As already mentioned, this information is often lacking.

Most of the tools implementing SCE models do not support project management in all of these steps. The seven steps are:

1. Creation of database of completed projects.
2. Size estimation.
3. Productivity estimation.
4. Phase distribution.
5. Sensitivity and risk analysis.
6. Validation.
7. Calibration.

What makes SCE Models difficult?

The following terms make software cost estimation models difficult:

- There is a lack of data on completed software projects. This kind of data can support project management in making estimates.
- Estimates are often done hurriedly, without an appreciation for the effort required to do a credible job. In addition, too often it is the case that an estimate is needed before clear specifications of the system requirements have been produced. There- fore, a typical situation is that estimators are being pressured to write an estimate too quickly for a system that they do not fully understand.
- Clear, complete and reliable specifications are difficult to formulate, especially at the start of a project. Changes, adaptations and additions are more the rule than the exception: as a consequence plans and budgets must be adapted too.
- Characteristics of software and software development make estimating difficult. For example, the level of abstraction, complexity, measurability of product and process, innovative aspects, etc.
- A great number of factors have an influence on the effort and time to develop software. These factors are called 'cost drivers'. Examples are size and complexity of the software, commitment and participation of the user organization, experience of the development team. In general these cost drivers are difficult to determine in operation.
- Rapid changes in information technology (IT) and the methodology of software development are a problem for a stabilization of the estimation process. For example, it is difficult to predict the influence of new workbenches, fourth and fifth generation languages, proto typing strategies, and so on.
- An estimator (mostly the project manager) cannot have much experience in developing estimates, especially for large projects. How many 'large' projects can someone manage in, for example, 10 years?
- An apparent bias of software developers towards underestimation. An estimator is likely to consider how long a certain portion of the software would take and then to extrapolate this estimate to the rest of the system, ignoring the non-linear aspects of software development, for example co-ordination and management.

- The estimator estimates the time it would take to perform the task personally, ignoring the fact that a lot of work will be done by less experienced people, and junior staff with a lower productivity rate.

Software Life Cycle Model (SLIM)

Putnam's SLIM is one of the first algorithmic models. It is based on the Norden / Rayleigh function and generally known as a macro estimation model. SLIM can record analyze data from previously completed projects which are then used to calibrate then a set of questions can be answered to get values of manpower buildup existing database (Jamil, 2007) [6].

SLIM enables a software cost estimator to perform the following functions:

- Calibration fine tuning the model to represent the local software development environment by interpreting a historical database of past projects.
- Build an information model of the software system, collection software characteristics, personal attributes, computer attributes.
- Software sizing SLIM uses an automated version of the lines of code (LOC) costing technique.

$$K=(LOC / (C^* t^{4/3}))^*3$$

K is the total life cycle effort in working years, t is development and the C is the technology constant, combining the effort of using tools, languages, and methodology and quality assurance (QA) time in years. The value of technology constant varies from 610 to 57314. For easy, experienced projects technology constant is high.

Putman developed a constraint model called SLIM to be applied to project exceeding 70,000 lines of code. Putman's model assumes that effort for software projects is distributed similarly to a collection of Rayleigh curves. Putman suggests that staffing rises smoothly during the project and then drops sharply during acceptance testing.

The SLIM model is expressed as two equations describing relation between the development effort and the schedule, the first equation, called the software equation, states that development effort is proportional to the cube of the size and inversely proportional to the fourth power of the development time. The second equation, the manpower-buildup equation, states that the effort is proportional to the development time.

Proposal using SLIM Model

This section suggests using SLIM model to estimate the cost of software by applying the Putnam's SLIM, and it is based on the Norden /Rayleigh function. This is explaining in the general algorithm:

Step 1: Input Software development.

Step 2: Compute the value of C, E, t and D.

Step 3: Compute the effort by applies the Software equation is:-

Size= CE^1/3(t^4/3)

Step 4: Compute the Manpower – Buildup Equation to find the total of time is needed to development software by: - D=E/t^ 3

Step 5: Analysis the result of step 4 and step5 and show the relation between the development effort and the schedule of time.

Step 6: Display the result of software cost estimate by draw the Rayleigh Curve.

Step 7: End.

An important advantage of applying the SLIM model to estimate the constraint model of project then the result is a good guide to show the relation between the effort and schedule time.

Advantage and disadvantage of SLIM

Advantages of SLIM

- Uses linear programming to consider development constraints on both cost and effort.
- SLIM has fewer parameters needed to generate an estimate over COCOMO'81 and COCOMO'II

Drawbacks of SLIM

- Estimates are extremely sensitive to the technology factor
- Not suitable for small projects

Estimation Techniques

Generally, there are many methods for software cost estimation, which are divided into two groups: Algorithmic and Non-algorithmic. Using of the both groups is required for performing the accurate estimation. If the requirements are known better, their performance will be better. In this section, some popular estimation methods are discussed. (Khatibi, V., & Jawawi, D. N. (2011) [7]

Algorithmic Models

These models work based on the especial algorithm. They usually need data at first and make results by using the mathematical relations. Nowadays, many software estimation methods use these models. Algorithmic Models are classified into some different models. Each algorithmic model uses an equation to do the estimation:

Effort = f(x1, x2,…, xn)

Where, (x1…xn) is the vector of the cost factors. The Differences among the existing algorithmic methods are related to choosing the cost factors and function. All cost factors using in these models are:

Product factors: required reliability; product complexity; database size used; required reusability; documentation match to life-cycle needs;

Computer factors: execution time constraint; main storage constraint; computer turnaround constraints; platform volatility;

Personnel factors: analyst capability; application experience; programming capability; platform experience; language and tool experience; personnel continuity;

Project factors: multisite development; use of software tool; required development schedule.

Quantizing the mentioned factors is very difficult to do and some of them are ignored in some software projects. In this study several algorithmic methods are considered as the most popular methods. The mentioned methods have been selected based on their reputation. (Khatibi, V., & Jawawi, D. N. (2011) [7]

Slim

Larry Putnam of Quantitative Software Measurement developed the Software Life-cycle Model (SLIM) in the late

1970s. (Boehm, B. *et al*. 2007) [3] It supports most of the popular size estimating methods including ballpark techniques, source instructions, function points, etc. It makes use of a so-called Rayleigh curve to estimate project effort, schedule and defect rate. A Manpower Buildup Index (MBI) and a Technology Constant or Productivity factor (PF) are used to influence the shape of the curve. SLIM can record and analyze data from previously completed projects which are then used to calibrate the model; or if data are not available then a set of questions can be answered to get values of MBI and PF from the existing database. Software Life – Cycle Model (SLIM) is based on Putnam's analysis of the life-cycle in terms oso called Rayleigh distribution of project personal level versus time. It makes use of a so called Rayleigh curve to estimate project effort, schedule and defect rate. In Software Life Cycle Management (SLIM), productivity is used to link the basic Rayleigh manpower distribution model to the software development characteristics of size and technology factors, productivity, P, is the ratio of software product size, S, and Development effort E that is:

$$P = \frac{S}{E} \quad \ldots (1)$$

The Rayleigh curve used to define the distribution of effort is modeled by the differential equation: (Jamil, A.S 2007) [6]

$$dy = 2Kate\text{-}at\text{^}2 \quad \ldots (2)$$

Recently, Quantitative Software Management has developed a set of three tools based on Putnam‘s SLIM. These include SLIM-Estimate, SLIM-Control and SLIM-Metrics. SLIM-Estimate is a project planning tool, SLIM-Control project tracking and oversight tool, SLIM-Metrics is a software metrics repository and benchmarking tool.

SLIM-Estimate

SLIM-Estimate helps you estimate the cost, time, and effort required to satisfy a given set of system requirements and determine the best strategy for designing and implementing your software or systems project. In addition to software cost estimation, this powerful systems and software project estimation tool provides a high level of configurability to accommodate the different design processes being used by developers today: such as Agile development, package implementation, hardware, call center development, infrastructure, model-based development, engineering and architecture design, service-oriented architecture, SAP, Oracle, and more.

SLIM-Estimate gives you the ability to

- Evaluate and sanity-check project plan alternatives with industry data or your own history before any task-level planning occurs;
- Validate estimates with the QSM Database, one of the largest industry databases of its kind;
- Negotiate a reasonable schedule and budget, using SLIM's powerful reporting capability
- Customize estimates with a push of a button;
- Eliminate updating and double entry with smart components;
- Interface with MS Office and the Web.

SLIM-Control

SLIM-Control has the statistical process control techniques you need to assess the status of your software or systems project, by comparing the project plan against project actuals and generate a forecast to completion. SLIM-Control offers both built-in and user-defined metrics as well as earned value charting and reporting.

SLIM-Control gives you the ability to

- Import plans created in SLIM-Estimate or SLIM-Data Manager
- Track project actuals against the plan using SLIM-Control's core metric set
- Assess project status at a glance
- Generate forecasts based on actual performances to date
- Evaluate different staffing options using SLIM-Control's Tradeoff Forecast
- Eliminate updating and double entry with smart components
- Use current industry benchmarks
- Interface with MS Office and the Web

SLIM-Metrics

SLIM-Metrics works with the SLIM-Data Manager data repository tool to help you preserve project history, assess competitive position, identify bottlenecks, quantify the benefits of process improvement, and defend future project estimates. Benchmark your data against industry reference trends from the QSM database of over 10,000 complete software projects or create your own benchmark trends to use in SLIM-Metrics, SLIM-Estimate, or SLIM-Control.

SLIM-Metrics and SLIM-Data Manager: Powerful tools to help you quantify and collect data

SLIM-Data Manager is a robust data repository tool, included with SLIM-Metrics, that helps you create a corporate database of your complete projects. This database can then be leveraged by SLIM-Metrics to analyze your data and uncover key relationships and trends.

SLIM-Metrics helps answer the questions software professionals ask every day:

- How does our development team stack up against the industry?
- Have our investments in process improvement increased our productivity?
- What kind of cost and schedule performance can we expect from different divisions?

SLIM-Metrics gives you the ability to

- Create and compare data subsets
- Create unlimited charting/report views
- Use current industry benchmarks
- Employ statistical analysis tools
- View details for any data point
- Add custom notes and other features
- Produce superb graphs and tables
- Export data easily (James T. Heires, (no date))

Conclusion

As conclusion, Software Cost estimation is the process of predicting the amount of effort required to build a software system. Cost estimates are needed throughout the software lifecycle. Preliminary estimates are required to determine the feasibility of a project. Detailed estimates are needed to

assist project planning. The actual effort for individual tasks is compared with estimated and planned values, enabling project managers to reallocate resources when necessary. Most of the work in the cost estimation field has focused on algorithmic cost modeling. Putnam's SLIM is one of the first algorithmic models. It is based on the Norden / Rayleigh function and generally known as a macro estimation model. SLIM-Control has the statistical process control techniques you need to assess the status of your software or systems project, by comparing the project plan against project actuals and generate a forecast to completion. SLIM-Control offers both built-in and user-defined metrics as well as earned value charting and reporting. SLIM Data Manager is a robust data repository tool, included with SLIM-Metrics, that helps you create a corporate database of your complete projects. This database can then be leveraged by SLIM-Metrics to analyze your data and uncover key relationships and trends.

References

1. Attarzadeh I, Ow SH. Proposing a New High Performance Model for Software Cost Estimation. In Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on 2009;2:112-116. IEEE.
2. Basha S, Ponnurangam D. Analysis of empirical software effort estimation models. 2010;arXiv preprint arXiv:1004.1239.
3. Boehm B, Abts C, Chulani S. Software development cost estimation approaches: A survey. Annals of Software Engineering 2007;10(1-4):177-205.
4. Barry Boehm, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy. Richard Selby Cost Models for Future Software Life Cycle Processes: COCOMO 2.0, International Society of Parametric Analysts, May 1995.
5. Barrow, Dean, Susan Nilson, Dawn Timberlake. Software Estimation Technology Report, Air Force Software Technology Support Center, Hill Air Force Base, Utah 1993.
6. Jamil AS. Used SLIM Model to Estimate Software Cost 2007.
7. Khatibi V, Jawawi DN. Software cost estimation methods: A review. Journal of emerging trends in computing and information sciences 2011;2(1):21-29.
8. Nassif AB, Capretz LF, Ho D. Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model. In Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on 2012, 589-594.
9. Patel AR, Banga AS. An Analytic and Parametric Study and Comparison of Software Cost Estimation Models. International Journal 2012, 2(9).
10. James T Heires (no date), Doing More with Less: SLIM-Estimate 5.0 Product Review