A Review of Parametric Effort Estimation Models for the Software Project Planning Process

Pablo Rodríguez-Soria[a], J.J. Cuadrado-Gallego[a,b], J.A. Gutiérrez de Mesa[a], Borja Martín-Herrera[a]

[a] *Universidad de Alcalá, Departamento de Ciencias de la Computación, 28805 Alcalá de Henares, Madrid, Spain*
[b] *Ecole de Téchnologie Superieure (ETS) – Université du Québec à Montreal, Canada*

{pablo.rsoria, jjcg, borja.martinh}@uah.es

Abstract

The aim of this research paper is to present a review of thee of the main software effort estimation methods, focused on Parametric models, that have been developed and then commercialized across the software engineering history. These models, among others, have been considered as the basis of the recent software project effort estimation and today conforms the nucleus of some of the most important companies of software effort estimation.

For each model is shown its main features, publications and equations that allow us to see as a whole the operation and implementation of each of these effort estimation methods.

1. Introduction to Software Cost Estimation

The more software becomes important in almost every human activity, the more it becomes complex and difficult to implement. Even if modern software technologies render easier the development of certain types of software products, increased user demands and new application domains produce additional problems. It is not surprising that software project management activities are becoming increasingly important.

One of the most critical activities during the software life cycle is that of estimating the effort and time involved in the development of the software product under consideration. This task is known as Software Cost Estimation (see *Figure 1*).

Estimations may be performed before, during and after the development of software. The cost and time estimates are necessary during the first phases of the software life cycle, in order to decide whether to proceed or not (feasibility study). Accurate estimates are obtained with great difficulty since, at this point, available data may not be precise, wrong assumptions may be made, etc. During the development process, the cost and time estimates are useful for the initial rough validation and the monitoring of the project's progress. After completion, these estimates may be useful for project productivity assessment.

Estimation methods fall in three main categories, namely expert judgment, machine learning and algorithmic cost estimation. Expert judgment [13] relies purely on the experience of one or more experts. Machine Learning estimation [19, 4], compares the software project under consideration with few (e.g. two or three) similar historical projects (i.e. projects with known characteristics, effort and schedule) using different automated or iterative rules. Algorithmic cost estimation involves the application of a cost model, i.e. one or more mathematical formulas which, typically, have been derived through statistical data analysis.

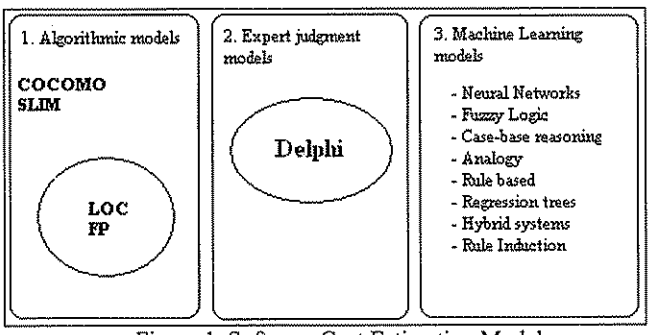


Figure 1. Software Cost Estimation Models

All of the three approaches have known advantages and disadvantages. Expert judgment is easy to apply and produces fast evaluation but suffers from the difficulty to find real experts and is exposed to wrong subjective assessment. Machine Learning models concentrate on a concrete, well-defined estimation framework provided that, suitable projects of the past may be easily found and the mechanism applied is correct. Parametric models are very useful when they are used correctly after they have been calibrated with historical data reflecting the characteristics of the estimated project.

It is important to remark that no single technique is best for all situations, and that a careful comparison of the results of several approaches is most likely to produce realistic estimates. In this survey, we are going to focus onto Algorithmic Models, also known as Parametric Models.

The rest of this paper is structured as follow: Section 2 introduces the reader to Parametric Models and its features. Section 3 presents three Parametric Commercial Models that we have chosen due to their historical relevance. Finally, Section 4 outlines the Conclusions and the main objectives raised in this research.

2. Cost Estimation focused on Parametric Models

Since the 1970's, a considerable amount of the software cost estimation research has been focused on the development of new and improved cost estimation models. New models have been proposed and existing models have been compared and validated.

Parametric Models were the most used techniques in the beginning of Software Cost Estimation. These models correspond to the box 1 in *Figure 1*. Then we are going to describe some issues about how these models work in Section 2.1; and how these models have been commercialized in section 2.2.

2.1. Parametric Models

The parametric, or statistical, method uses regression analysis of a database of two or more similar systems to develop cost estimating relationships (CERs) which estimate cost based on one or more system performance or design characteristics (e.g., speed, range, weight, thrust). The parametric method is most commonly performed in the initial phases of product description and development. Although during this phase an acquisition program is unable to provide detailed information (like drawings and standards), the program can specify top-level system requirements and design characteristics. In other words, estimating by parametric is a method to show how parameters influence cost.

Parametric estimating is used widely in government and industry because it can yield a multitude of quantifiable measures of merit and quality (i.e., probability of success, levels of risk, etc.). Additionally, CERs developed using the parametric method can easily be used to evaluate the cost effects of changes in design, performance, and program characteristics.

The fast changing nature of software development has made it very difficult to develop parametric models that yield high accuracy for software development in all domains. Software development costs continue to increase and practitioners continually express their concerns over their inability to accurately predict the costs involved. Accurate software cost estimates are critical to both developers and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time. Overestimating may result in too many resources committed to the project, or, during contract bidding, result in not winning the contract, which can lead to loss of jobs.

Accurate cost estimation is important because:

- It can help to classify and prioritize development projects with respect to an overall business plan.
- It can be used to determine what resources to commit to the project and how well these resources will be used.
- It can be used to assess the impact of changes and support re-planning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs.

Software cost estimation involves the determination of one or more of the following estimates:

- effort (usually in man-months)
- project duration (in calendar time)
- cost (in dollars)

Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost. Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in man/months (MM) of the programmers, analysts and project managers. This effort estimate can be converted into a dollar cost figure by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required.

Most cost models are based on the size measure, such as Lines of Code (LOC) [2] and Function Points (FP) [1], obtained from size estimation. The accuracy of size estimation directly impacts the accuracy of cost estimation.

2.2. Commercial Tools

Since the mid 1990's there have been about 50 commercial software cost estimation tools marketed in the United States and another 25 in Europe, although not all at the same time. Many of these tools are "black boxes" and their methods of operation are proprietary and regarded as trade secrets by their owners [12]. However, while these estimating tools were developed by different companies and

are not identical, they do tend to provide a nucleus of common functions and public equations.

The software cost estimation market was created by researchers who were employed by large enterprises that built large and complex software systems: IBM, RCA, TRW, and the U.S. Air Force were the organizations whose research which led to the development of commercial cost estimating tools.

Commercially available cost estimation tools try to offer the user greater utility by packaging the parametric model with a user interface, database of completed projects, some way of estimating the size of the project, and/or context-sensitive help.

Whatever features any tool may have, most parametric models are likely to employ one or more of three methodologies; Putnam methodology [15] is based on the insight that efficiently run software projects follow well-defined patterns that can be modeled with a set of exponential equations. COCOMO II [5] is a continuation of work begun by Dr. Barry Boehm at USC. Monte Carlo simulation models complex interactions in the face of uncertain estimating assumptions.

As of 2009, some of these estimating tools include COCOMO II, CoStar, CostModeler, CostXpert, KnowledgePlan, PRICE S, SEER, SLIM, and SoftCost. Some older automated cost estimating are no longer being actively marketed but are still in use, such as CheckPoint, COCOMO, ESTIMACS, REVIC, and SPQR/20. Since these tools are not supported by vendors, usage is in decline. The major features of commercial software estimation tools include these attributes:

- Sizing logic for specifications, source code, and test cases
- Phase-level, activity level, and task-level estimation
- Adjustments for specific work periods, holidays, vacations, and overtime
- Adjustments for local salaries and burden rates
- Adjustments for various software projects such as military, systems, commercial, etc.
- Support for function point metrics, lines of code metrics, or both.
- Support for "backfiring" or conversion between lines of code and function points
- Support for both new projects and maintenance and enhancement projects

Some estimating tools also include more advanced functions such as:

- Quality and reliability estimation
- Risk and value analysis
- Return on investment (ROI)
- Sharing of data with project management tools
- Measurement modes for collecting historical data
- Cost and time to complete estimates mixing historical data with projected data
- Support for software process assessments
- Statistical analysis of multiple projects and portfolio analysis
- Currency conversion for dealing with overseas projects

3. Review of 3 Parametric Commercial Models

In this section, three of the most relevant models in software cost estimation history will be treated. We have reviewed these tools trying to show their main features, publication dates and central equations. The models in review are: SLIM, SEER-SEM and SPR-Knowledge Plan.

3.1. SLIM – Putnam – 1979

SLIM; Software LIfecycle Management.
First Publication: Putnam, 1978 [15]
Patent: Quantitative Software Management (QSM).
Tools:

1. SLIM-Estimate. It is a project planning tool.
2. SLIM Control. It is a project tracking and control tool.
3. SLIM Metrics. It is a software benchmarking tool.

Larry Putnam and Ann Fitzsimmons founded Quantitative Software Management (QSM) and build the first version of SLIM in 1979. It became the second commercial software cost estimation tool on the market.

This model is based on the software lifecycle analysis of Putnam in terms of the size distribution of the development team of a software product against the time that follows a distribution of Rayleigh and it is based on the work of Norden [14] and Aron [3]. Norden observed through the graphical representation of the personnel distribution frequencies during the development and maintenance phases of many projects implemented in IBM, that the curves resembled quite to the distribution curves of Rayleigh since 90% of the project was completed in two-thirds of the total time, while the remaining 10% needed a third of the total time remaining to be completed. Although this distribution was purely empirical, Norden found no theoretical basis for it.

SLIM supports the widespread methods of size estimating, including the source lines of code and function points. It can predict the size of the project, the effort, the development time and the proportion of defects.

Equations: The equations of the model have not been edited for the public domain, although the central algorithms of the model were published by Putnam [16]. These are the ones collected here:

1. Size: $e = c.\ (E_d)^{1/3}\ .\ (t_d)^{4/3}$

   Where e is the size in SLOC, E_d is the total effort needed to complete the project, selected form a database of previous projects, c is a constant of the project called by Putnam *Technological Factor*, this factor reflects the effect of numerous costs drivers as the constraints of hardware, the complexity of the program, the personnel levels of experience and the programming environment. t_d is the total development time of the project.

2. Effort: $E(t) = Ed\left(1 - e^{-at^2}\right);\ a = \frac{1}{2\,(td)^2}$

   Where $E(t)$ is the effort that has been consumed in MM in order to develop the project during t months, a is a constant of the project that determines the curve, it is also obtained form previous projects.

3.2. SEER-SEM – 1989

SEER-SEM; System Evaluation and Estimation Resources – Software Estimation Model.

First Publication: Jensen, 1983 [10]

Patent: Galorath Associates Inc.

Tools:

- SEER-SEM.

It is based on the Model of Jensen of 1979 [9], which is based on the Model of Putnam of 1977 [15].

The scope of the model is broad and covers all phases of the lifecycle, from the first specifications, until design, development, delivery and maintenance. It manages a wide variety of development environment configurations and types of applications, such as client – server, distributed, graphics, etc. It manages the development methods and languages more used. The development methods include objects oriented, reuse, development in spiral, cascade, of prototypes and incremental. Languages include both the 3rd and 4th generation (C++, FORTRAN, COBOL, Ada, etc.) as well as application generators. It allows taking as constraints the capacity of the development team, the design standards and process required, and the levels of an acceptable development risk. Among the characteristics of the model, the following are included:

- It allows that the level of the estimation probabilities, the development team and the development time are inputs as independent variables.
- It allows an extended sensitive analysis and a monitoring of the input parameters of the model.
- It shows the project cost drivers.
- It allows and interactive adjustment of the schedule of the project elements through Diagrams of Gantt.
- It builds the estimates by a knowledge base of existing projects.

The model specifications include:

1. *Parameter:* Size, personnel, complexity, development environment, method of development and acquisition, applicable standards.
2. *Predictions:* Effort, development time, development team, defects, costs. The estimates may be based in development time or effort. The constraints can be specified at the development time or development team.
3. *Risk Analysis*
4. *Methods for Size Estimating:* Function Points, approved by IFPUG (International Function Points User Group) [7] in addition to an increased set and lines of code, both new and existing ones.

Figure 2 is adapted from a Galorath illustration and shows gross categories of model inputs and outputs, but each of these represents dozens of specific input and output possibilities and parameters. The reports available from the model cover all aspects of input and output summaries and analyses numbers in the hundreds.

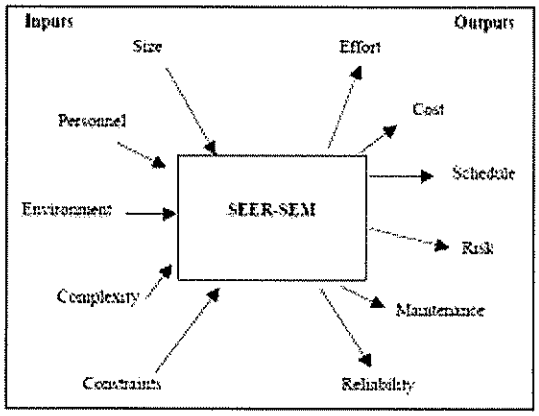


Figure 2. SEER-SEM Inputs and Outputs

As part of this effort, Galorath maintains a software project repository of approximately 6,000 projects (and growing). About 3,500 projects containing effort and duration outcomes are stored in a unified repository that can be readily accessed for studies. SEER is also available with data repositories from The International Software Benchmarking Standards Group (ISBSG) [8]. ISBSG provides the largest open repository of software project history and includes standardized, verifiable data from over 4,000 software projects.

These projects are from both defense and commercial sources representing many development organizations, permitting calibration of the model to a wide array of potential projects. Additional project outcomes, in the hundreds, are also available to the company, which has also collected sizing and other information on thousands of additional projects. Analysis involves running project data through SEER-SEM using a special calibration mode. The

model is essentially run backwards to find calibration factors.

Productivity factors are evaluated across different data attributes (e.g. platform, application, etc.) to detect trends. A variety of methods are used to mitigate outlier data points and control for variation. The variance in the data set is also used to establish default parameter ranges; nearly all settings accommodate risk. Model settings are updated as new trends are established.

SEER technology provides project results by generating a virtual project based on:

- *The SEER Modeling Engine:* SEER mathematical models are derived from extensive software project histories, behavioral models, and metrics. SEER for Software (SEER-SEM) employs a multi-faceted approach to project estimating, leveraging industry and/or company project histories and proven formulaic cost relationships.
- *SEER Knowledge Bases:* Serve as a virtual "in-house expert," providing default values, ranges, and calibrations based on comparable software project histories.

Together, these capabilities enable users to develop first-look estimates when very little information is known, and to those estimates as details become available over time.

Equations: The equations of the model have not been edited for the public domain, although a few of the central algorithms of the model were published by Jensen [9] and collected here:

1. Size: $s = c\,(td)\,(Ed)1/2$

   Where s is the size in SLOC, E_d is the total effort needed to complete the project, selected from a database of previous projects, c is a constant of the of the project called Technological Factor of Jensen, this factor reflects the effect of numerous costs drivers such as the hardware constraints, complexity of the program, the levels of the team experience and the programming environment. t_d is the total development time of the project.

2. Effort: $e = 0.4\left(\frac{s}{c}\right)^2\left(\frac{1}{t^2}\right)$

   Where e represents the Effort measured in MM and t is the time consumed since it began development. K is the total effort of the lifecycle.

3.3. SPR-Knowledge Plan – 1997

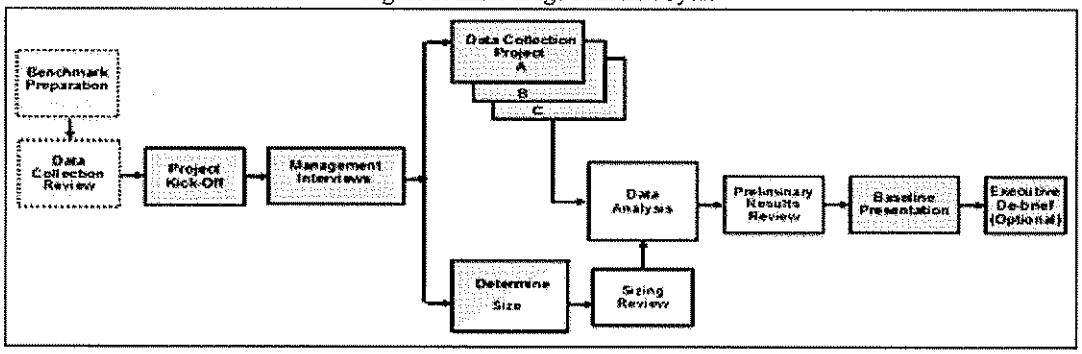
Patent: SPR – Software Productivity Research

SPR Knowledge PLAN is a commercial Windows-based software estimating tool from Software Productivity Research (SPR).

Launched in 1997, SPR Knowledge PLAN was the first powerful, knowledge-based software estimation tool to combine project estimation and scheduling in a flexible environment. This tool quickly creates and refines detailed project plans for seamless export to Microsoft Project or other enterprise project management systems.

SPR Knowledge PLAN is a software tool designed to help in planning software projects. With this tool, the user can effectively size each project and then estimate work, resources, schedule, and defects. User can even evaluate project strengths and weaknesses to determine their impact on quality and productivity.

Figure 3. Knowledge Plan LifeCycle



SPR Knowledge PLAN provides a complete and rational view of all tradeoffs among features, schedules, quality and costs. User can explore the cost/value implications of additional resources, more powerful languages, development tools, improved methods and other technical changes. User can also track milestones, schedules, resources, actual work effort, and defects found.

This estimation tool provides a bi-directional interface with project management applications to create an integrated, full life-cycle solution. For convenience, some project management functions such as critical path scheduling are offered.

Knowledge Plan collects information at the project level, using a "representative sample approach." The

sample projects selected reflect the work patterns of user's organization itself, e.g., a mix of new projects, enhancements, and those with special factors such as high usage of contractors or integration of packages. In the analysis, the project data is summarized to create a complete and accurate picture of the organizational.

SPR uses its expertise in measurement to assist clients in establishing meaningful portfolio and project baselines. Then, SPR draws on its extensive industry knowledge base, derived from more than 14,531 completed software projects of all types, as a reference point against which to compare (benchmark against) client baseline data. It is the actual organizational data that forms the baseline and the comparative analysis against industry data that constitutes the benchmark.

4. Conclusions

Software estimating is simple in concept, but difficult and complex in reality. The difficulty and complexity required for successful estimates exceeds the capabilities of most software project managers to produce effective manual estimates. The commercial software estimating tools can often outperform human estimates in terms of accuracy, and always in terms of speed and cost effectiveness.

However, no method of estimation is totally error-free. As mentioned before, the current "best practice" for software cost estimation is to use a combination of software cost estimating tools coupled with software project management tools, under the careful guidance of experienced software project managers and estimating specialists.

The fundamental objective raised in this research was the study and the analysis of three of the main software effort estimation methods, focused on Parametric models, that have been developed and then commercialized across the software engineering history.

With this review, we have tried to show how these estimation models work and which their main features are. Taking a clear view of these estimation models as a whole, we will be able then to understand how actual software project estimation companies has gained an important "piece" of the today software industry market.

Acknowledgement

We would like to thank the University of Alcalá for supporting this research (Ph.DC researchers support programme).

References

[1] Albrech, A. "Measuring Application Development Productivity". Proceedings of the IBM Application Development Symposium, GUIDE/SHARE, California, USA, pp. 83-92, 1979.

[2] Albrecht, A.J., and Gaffney, J.E. "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Transactions on Software Engineering (SE-9:6), pp 639-648, 1983.

[3] Aron, J. "Estimating Resources for Large Systems", In NATO Conference Report on Software Engineering Techniques, Eds. J.N. Buxton y B.Randel, Rome (Italy), 1969.

[4] Bisio, R. and F. Malabocchia. 'Cost estimation of software projects through case base reasoning', in Proc. 1st Intl. Conf. on Case-Based Reasoning Research & Development . Springer-Verlag, 1995.

[5] Boehm, B., Clark, B., Horowitz, E., Madachy, R., Selby, R. and Westland, C. "Cost Model for Future Software Life Cycle Processes: COCOMO 2.0". Annals of Software Engineering Special Volume on Software Process and Product Measurement, Eds. J.D. Arthur, S.M. Henry and J.C. Baltzer, Ed. AG Science Publishers, Amsterdam (Netherlands), Vol. 1, 1995.

[6] Freiman, F.R., and Park, R.E. "The PRICE software cost model," Proceedings of the IEEE National Aerospace and Electronics Conference NAECON, p. 500. New York, USA, 1979.

[7] IFPUG, International Function Points Users Group, "Function points counting practices manual 4.1.1". Ohio, USA, 1999.

[8] ISBSG, International Software Benchmarking Standards Group repository, Release 10. http://www.isbsg.org

[9] Jensen, R.W. "A macro-level software development cost estimation methodology". Conference Record of the Fourteenth Asilomar Conference on Circuits Systems & Computers, p. viii+520, 1979.

[10] Jensen R. "An improved Macrolevel Software Development Resource Estimation Model". Proceedings 5th ISPA Conference, pp. 88-92, 1983.

[11] Jones, C. "Programming Quality and Programmer Productivity", IBM Technical Report TR-02-764, pp. 39-63, 1977.

[12] Jones, C. "How software estimation tools work". SPR Technical Report, Version 5 – February 27, 2005.

[13] Jørgensen, M. "A Review of Studies on Expert Estimation of Software Development Effort". Journal of Systems and Software 70 (1-2): pp. 37-60, 2004.

[14] Norden, P.V. "Curve fitting for a model of applied research and development scheduling," A-IBM Systems Journal (2:3), 1958.

[15] Putnam, L.H. "A general empirical solution to the macro software sizing and estimating problem," IEEE Transactions on Software Engineering (SE-4:4), pp 345-361, 1978.

[16] Putnam, Lawrence H., and Ware Myers. "Measures for Excellence: Reliable Software on Time" Within Budget, Englewood Cliffs, NJ: Yourdon Press, 1992.

[17] Shepperd, M.J., C. Schofield, and B.A. Kitchenham. 'Effort estimation using analogy', in Proc. 18th Intl. Conf. on Softw. Eng. Berlin: IEEE Computer Press, 1996.