U.S. Gov't

N92-22709

COSTMODL: AN AUTOMATED SOFTWARE DEVELOPMENT COST ESTIMATION TOOL

George B. Roush
NASA Johnson Space Center
Houston TX 77058

ABSTRACT

The cost of developing computer software continues to consume an increasing portion of many organizations' total budgets, both in the public and private sector. As this trend develops, the capability to produce reliable estimates of the effort and schedule required to develop a candidate software product takes on increasing importance. The COSTMODL program was developed to provide an in-house capability to perform development cost estimates for NASA software projects. COSTMODL is an automated software development cost estimation tool which incorporates five cost estimation algorithms including the latest models for the Ada language and incrementally developed products. The principal characteristic which sets COSTMODL apart from other software cost estimation programs is its capacity to be completely customized to a particular environment. The estimating equations can be recalibrated to reflect the programmer productivity characteristics demonstrated by the user's organization, and the set of significant factors which effect software development costs can be customized to reflect any unique properties of the user's development environment. Careful use of a capability such as COSTMODL can significantly reduce the risk of cost overruns and failed projects.

INTRODUCTION

Need for Formalized Software Cost Estimation

In the early days of the evolution of computer software, managers were forced to rely on the opinions of software development team leaders for estimates of the time and effort required to develop a candidate software product. This was a highly subjective process which was greatly influenced by factors such as the personality of the estimator (optimist or pessimist), pressure to underestimate to win a contract, etc. Since these estimates were based primarily on intuition and personal experience, the estimates were extremely difficult to reproduce or to refine as a project matured.

In the late 1970s, mathematical models began to emerge which attempted to quantify the parameters upon which an estimate was based, and to compute the estimates using equations which were developed based on actual data derived from completed software projects. The successful development of an estimating algorithm is dependent on the identification of the factors which affect productivity, and the careful collection of data over a sufficient period of time and number of projects to permit a valid statistical analysis. It is not surprising that the early models were hampered by the scarcity of good data.

Different Methodologies

As managers began to search for ways to improve the fidelity of their estimates, several methodologies emerged. Examples are Expert Opinion, Analogy, and Parametric Models. The expert opinion approach involves convening a group of experienced project managers to assess the known requirements for the new project, and to apply the knowledge resulting from their previous experience to collectively arrive at an estimate of the effort and schedule required to build the candidate product. This approach helps to minimize the effects of individual personalities and biases, and broadens the base of experience upon which the estimates are based. However, it is still an inherently subjective process which does not lend itself well to analysis and refinement.

When applying the analogy approach, the intent is to identify completed projects which have significant similarities to the candidate project, and to draw conclusions about the expected development cost based upon the development costs of the completed projects. This process moves one step farther away from purely subjective opinion, but still does not lend itself to refinement over time.

The parametric approach is the approach most commonly used in automated software cost estimation tools. This approach normally involves a set of basic estimating equations, usually non-linear, where the values of the coefficients and exponents are determined from the application of statistical analysis to a database of productivity data derived from completed software development projects. In addition, a set of factors which affect productivity are identified, and a set of ratings for each factor is developed such that the appropriate characteristics of the candidate project can be quantified. The desired estimates are then computed by some formula which utilizes this set of numerical data. This is a repeatable process inasmuch as providing the same inputs will produce the same outputs, and it lends itself well to refinement over time as the knowledge of the input parameters matures during the life cycle of the project.

THE COCOMO MODEL

In 1981, Dr. Barry Boehm introduced the COnstructive COst MOdel (COCOMO)[1]. The COCOMO was developed using data from a database of 63 projects which were completed between 1968 and 1979. It is a non-linear parametric model whose exponent is greater than unity, indicating that the larger a program is, the more expensive each line of code becomes.

The principal input to the model is the anticipated size of the program to be developed, expressed in thousands of delivered source instructions (KDSI). This size estimate may be divided into two parts - the new code to be developed and the existing code to be adapted for use in the new program. To provide for uncertainty in the new code size estimate, the model requires estimates of the smallest expected size, the largest expected size, and, within that range, the most likely size. A beta distribution is then used to compute the estimated size. For the adapted code, the amount of rework to be required is expressed in terms of percentages of redesign, recoding and integration. Together, the new code and adapted code combine into thousands of effective delivered source instructions (KEDSI), the parameter upon which the effort calculations are actually based.

The original COCOMO contained fifteen parameters called Cost Drivers. These parameters describe properties of the program to be developed, the development team, and the environment within which the development will be done which significantly effect the productivity of the development team. The cost drivers are intended to be mutually orthogonal, meaning that each one stands independent of the others. It is further intended that any factor which will affect program development cost can be associated with one of the cost drivers.

In addition to producing estimates of the total effort and schedule required to complete a project, the COCOMO computes the distribution of the effort and schedule among the project development life cycle phases. The life cycle phases included in the COCOMO are Product Design, Programming, and Integration & Testing. The apportionment of effort and schedule among these phases is based on the contents of the Phase Distribution Tables, a set of tables which contain the percentages of the total effort and schedule to be allocated to each life cycle phase.

Included Models

The original COCOMO model actually consisted of three estimating algorithms. These three algorithms represented three levels of fidelity and were called the Basic, Intermediate, and Detailed models. In the Basic COCOMO, the estimation is performed at the total project level, and no cost drivers are included. The Intermediate COCOMO permits the total project to be decomposed into a set of components, and the cost drivers are included in the equation. The Detailed COCOMO provides for the decompositon of the program down to the subroutine/procedure level, and also includes the cost drivers.

In 1987, a fourth model was defined. This model, called the Ada COCOMO, was developed to accommodate changes in programmer productivity which can result from careful application of the software engineering practices which are supported by the Ada language. However, its applicability is not restricted to Ada program development. It can be the model of choice in any situation where modern software engineering practices are being followed, regardless of the language being used in the implementation.

In addition to these basic estimating models, an extension is included called the Incremental Development Model. This model permits the development of separate estimates for a set of independent intermediate deliveries, and combines these into a total estimated for the complete project.

Development Modes

The COCOMO model identifies three distinct software development modes. These are called the Organic mode, Semi-Detached mode, and Embedded mode.

In the organic mode, relatively small software teams develop software in a highly familiar, in-house environment. The Semi-Detached mode represents an intermediate stage between the organic and embedded modes. This can mean either an intermediate level of the project characteristics or a mixture of the organic and embedded mode characteristics. The major distinguishing factor of an embedded-mode software project is a need to operate within tight constraints. The product must operate within (is embedded within) a stringently coupled complex of hardware, software, regulations, and operational procedures. It is a characteristic of embedded mode projects that changes to one part severely affect other parts. This results in the development typically being more expensive (less productive) than the more independent organic mode projects.

The three modes are represented in the model by different sets of coefficients and exponents in the estimating equations. The different coefficients reflect the differences in productivity associated with the different modes, and the different exponents reflect the different effect that changes in program size have on programmer productivity.

The Estimating Equations

The COCOMO effort estimating equation is

$$MM = \alpha \times KDSI^{\beta} \times \Pi F_i$$

where MM = Man Months
α = Productivity Coefficient
KDSI = Thousands of Delivered Source Instructions
β = Exponent relating productivity to program size
ΠF_i = Product of the Cost Drivers

The schedule estimating equation is

$$TDEV = \delta \times MM^{\varepsilon}$$

where TDEV = Development schedule in months
δ = Schedule Coefficient
MM = Man Months
ε = Exponent relating productivity to program size

Note that the Basic COCOMO effort equation does not include the cost driver term, and that the coefficients and exponents are different for each development mode.

The COSTMODL Program

The COSTMODL is an automated software development cost estimation program. It runs on IBM PCs and compatible computers. It implements all of the COCOMO models except for the detailed model. In addition, it includes a simplified linear model which was developed by NASA at the Johnson Space Center using productivity data from completed NASA projects. This model is called the "Keep It Simple, Stupid" (KISS) model.

Major Features

Major features of the COSTMODL program are its ease of use, its incorporation of multiple estimating models, its ability to support multiple projects and multiple model configurations, and the capability it provides for a user to completely customize the estimating equations to a particular software development environment.

Ease of Use

The most immediately apparent major feature of COSTMODL is its ease of use. It is delivered with an automated installation utility (CINSTALL) which includes a default set of destination subdirectories for the installation. The user is given the opportunity to modify the directory specifications and then CINSTALL automatically configures the destination hard disk, uncompresses the delivery files into the destination directories, checks for any errors during installation, and, if there were none, transfers to the directory into which COSTMODL was installed and executes COSTMODL.

COSTMODL was designed to be immediately useful without requiring that the user read any printed documentation or attend any user training. The program is completely menu-driven, can be controlled from either the keyboard or a mouse, and includes an extensive context-sensitive help system. Immediately upon entry into the program for the first time, the new user is given instructions for entering the help system, and then is led through a brief tutorial on the COCOMO model, usage of the menu system, the function keys, and the help system itself. Complete user's documentation is included with the program, but the user has the option of going directly into the program and using the user's guide only for reference.

Data Management

Once an estimate has been done for a project, that project can be saved in COSTMODL's project library for future use. Many projects can be saved, and many project data files can be saved for each project. The first required input when a new project is being defined is the project name. This name will appear in the project menu on subsequent runs until it is deleted from the library. At any point in the development of a project estimate, the current status of the project can be saved to a file. Whenever that file is recalled on a later run, control will return to the point in the program at which the file was saved, and any desired changes to the project description can be entered at that time.

When a project data file is saved, it is associated with the model which was being used at the time it was saved. However, the file can be imported into a different model. For example, suppose a new project is being considered, and very little is known about its detailed requirements. A rough estimate can be obtained by using the Basic COCOMO or the KISS model, and the resulting estimate saved to a file. Then, as additional information becomes available about the program to be developed, that file can be imported into the Intermediate COCOMO model for refinement of the estimates. The data in the project file will automatically be promoted to the new model, and, once the new information is entered, the new project file will be associated with the model currently in use. This provides the capability to progress to higher fidelity models as one's knowledge of the program to be developed matures. In addition, the retention of a series of project data files throughout the development life cycle of a project makes available a valuable resource for evaluating the validity of the inputs at various points in the development process. These data files can also be useful for identifying those input parameters which are most likely to have adverse effects on the accuracy of the resulting estimates.

In addition to the project data files, additional data files are included which contain the detailed definitions of the models. The coefficients and exponents for the estimating equations, the definitions of the cost drivers and their associated multiplicative values, and the phase distribution tables are all located in data files.

Customization

The most significant feature of COSTMODL which sets it apart from other automated cost estimation tools is its ability to be completely customized by the user. All of the parameters which define the models are available to the user for modification. The most likely parameters to be modified are the coefficients and exponents on the estimating equations. Let us look at why one would want to do this.

The original COCOMO was defined using a database of 63 projects. This set of projects included a mix of business data systems, military real-time control systems, aerospace programs, operating systems, etc, all of which were completed in the 1968-1979 time frame. It is reasonable to expect that, if you are developing transaction processing software for the banking industry in a highly interactive software development environment, the accuracy of the estimates produced by a model would probably be more reliable if the model had been tuned to that specific environment. If a user can obtain good productivity data either from within his own organization or from one which has similar attributes, this data can be used to recalibrate the estimating equations to that environment. This recalibration normally involves only the coefficients on the estimating equations, and can have a significant effect on the accuracy of the estimates produced. COSTMODL provides the capability for the user to perform the recalibration externally and to enter the resulting coefficients (and possibly exponents) directly into the proper input forms, or he may provide the productivity database directly to COSTMODL and it will perform the desired calibration.

The set of cost drivers can also be modified or totally replaced. The number of cost drivers, their names, attributes, descriptions, and multiplicative values are also under the control of the user. It is not unlikely that a particular development environment contains a factor which significantly affects productivity but which is not included in the original set. For example, suppose a project is being developed in a very large organization and different parts of the program are being developed at widely separated locations. This can affect the total productivity, but is not included in the default cost driver set. It is simple in COSTMODL to add a cost driver for that factor.

It is possible that different organizations may include different activities in the various development life cycle phase definitions, resulting in percentage distributions of effort and schedule which do not agree with the original COCOMO definitions. In this case, the user can modify the contents of the phase distribution tables to reflect those differences, resulting in estimated phase distributions which more accurately reflect those expected within his organization. In addition, the next release of COSTMODL will provide the capability for a user to change the number and names of the life cycle phases in addition to the percentage distributions.

Since the behavior of the program is completely dependent upon the contents of the various data files, it is important that the integrity of these files be carefully protected. To provide for the situations where multiple users may use COSTMODL on the same machine, or when COSTMODL is installed on a network server for use by many people, that portion of the program which permits modification of the configuration files is password protected. This makes it possible for the system administrator to control the contents of these files, and all of the users to make use of the configurations that have been standardized for use within that organization.

Incremental Development Model

The COSTMODL implementation of the incremental development model provides the user with complete flexibility in the definition and modification of the increments, ordering them, and relating one to another. Each increment is created as a total project and saved to a file. Once all of the increments have been defined, the user identifies them in sequence to the incremental development model. As each increment is added to the project, the user specifies the time relationship between it and the previous increment. For example, the Critical Design

Review (CDR) for the fourth increment might be scheduled to be held two weeks after the beginning of coding on the third increment.

In addition to the time relationships between the increments, the user specifies the expected amount of rework to be done on the previous increments to accommodate the next increment. This adaptation effort is frequently called the "breakage factor".

The incremental development model computes the total effort and schedule required to complete the total project, and, for comparison, computes the effort which would have been required had the total project been delivered as a single delivery. In addition, the cumulative staffing level required as a function of time is also computed. The overall schedule can be displayed graphically showing each increment and each life cycle milestone within each increment. The cumulative staffing level can be similarly displayed in the form of a histogram. To facilitate relating the staffing changes to the beginning and ending of the various increments, the two plots can be superimposed.

Each of the plots can be output in the form of a printed report. The total time duration and the number of increments is examined to determine the page orientation and number of pages to be used for the printed displays. It is therefore possible to print complete schedule and staffing plots for any size project.

INDUSTRY ACCEPTANCE

COSTMODL has enjoyed broad acceptance within the software cost estimation community. It is currently being used at more than 300 installations throughout the aerospace community, government agencies, DOD, and academia in the U.S., Europe and Canada. It is a principal estimating tool in use at NATO headquarters in Brussels. It is also being used as a teaching tool at several universities, and was recently selected as the best overall implementation of the COCOMO model in a competitive evaluation conducted by MIT's Sloan School of Management.

COSTMODL is also being distributed to the students and staff at the Defense Management Systems College, and is being distributed by the Air Force Cost Center through their software and data distribution channels.

TYPICAL USES

Typical uses of COSTMODL include feasibility assessment, bid preparation, bid evaluation, budget/manpower planning, and project scheduling. In a time of severe budget constraints, a defensible early estimate of the cost to develop a candidate software product can help avoid spending resources on a project that may not be cost effective. It can also be quite valuable in the control of cost overruns. What is seen as a cost overrun is frequently the result of an underestimate of the effort which can be expected to be required.

The use of a formal cost estimation tool in bid preparation is obvious. It can also be quite valuable to the team evaluating a proposal. It is at this point that possible underestimates can be identified, resulting in the possible avoidance of either perceived cost overruns or even failed projects.

A less obvious benefit resulting from the use of a parametric model can be the thorough documentation of all of the assumptions that went into the development of the estimate. Suppose funds are being sought for a new program development, and senior management is challenging the estimated cost to complete the project. The program size estimate and each of the cost driver inputs can be discussed to see if there is disagreement with any of the inputs, and, if changes are agreed upon, a revised estimate can be produced. Further credibility can be attached to the estimates by referencing the productivity data in the database which was used to calibrate the model, particularly if in-house data was used.

APPLICABILITY

A tool such as COSTMODL can be an important asset for any organization which has a significant financial interest in software development cost management. This includes organizations which develop software products for the commercial marketplace, organizations which develop custom software, the organizations which contract for the development of custom software, and organizations which develop software in-house for their own use.

FUTURE PLANS

COSTMODL will continue to incorporate the latest advances in software cost estimation technology as they become available. The next version will include the capability for the user to redefine the software development life cycle by adding or deleting life cycle phases, or modifying the definitions of existing phases. The classifications of personnel to be included in the development cost computations can be defined, along with their dollar cost per man month and the percentages of their effort to be applied to the current project. An expanded adapted code model will be included which will better accommodate the situation where a new product will consist mainly of adapted code, with a relatively small amount of new code to be written. This situation is quite common in a mature organization where a large pool of software exists which is specific to that organization's function, much of which is adaptable for use in new software products.

Software sizing models are being investigated for possible incorporation. A sizing model allows a user to describe the product to be built in functional terms instead of thousands of lines of code. Function Points are proving to be useful in business applications, but have not been as successful in the scientific software domain. Examples of function points are display screens, input forms, number of input fields, number and size of external databases, etc. COSTMODL will incorporate a Function Point sizing algorithm for use on appropriate applications. The evaluation of candidate sizing algorithms for scientific applications will continue.

The COSTMODL developers are currently involved in studies intended to expand the estimating models to include Fourth Generation Language development, Knowledge-Based Systems, and Totally Reusable Software Libraries. Each of these areas are increasing in importance, and are not well modeled by current algorithms.

Another effort currently underway is the integration of COSTMODL into a total project management system. This integrated capability would combine the effort and schedule estimating features of COSTMODL with the real-time collection of actual expenditures of resources during the development life cycle of a project to continuously refine cost-to-completion estimates.

AVAILABILITY

COSTMODL has been submitted to NASA's Computer Software Management and Information Center (COSMIC) for distribution into the private sector. COSMIC can be contacted at:

The University of Georgia
Computer Services Annex
Athens, GA 30602
(404) 542-3265

In addition, copies can be requested through the Software Technology Branch Help Desk at (713) 280-2233

1. Boehm, B. W., *Software Engineering Economics*, Englewood Cliffs, NJ, Prentice-Hall, 1981.