COCOMO II Based Project Cost Estimation and Control

Aihua Ren[1, a], Yun Chen[1, b]

[1] School of Computer Science and Technology, Beihang University, Beijing 100191, China;

[a]renah@buaa.edu.cn, [b]chenclouding@163.com

Keywords: COCOMO II; process performance model; software cost control; cost estimation

Abstract. COCOMO II is a constructive cost model that is widely used in the world. With function points or source lines of code and adjustment factors as input, it can predict software cost. Software process performance model is mainly used to indicate the relationship between the attributes and product in the software development process. Based on COCOMO II and process performance model theory, the viability that COCOMO II serves as a software process performance model is introduced in this paper. Controllable factors, the way to adjust them and benefits we can get are proposed.

Introduction

With the expanding application scope, increasing size and complexity of software, software development problems occur frequently, for example, development schedule is unpredictable, cost is often out of control, users are unsatisfied with product functions, product quality cannot be guaranteed, software is difficult to maintain, etc. CHAOS report from Standish group shows that the percentage of projects that overweighed cost is as much as 59% [1]. Thus, cost control is a necessary condition for software success.

To reduce the risks that are caused by inaccurate cost estimation, organizations should establish project cost management framework based on process performance model. On one hand, software cost estimation should be done well. On the other hand, cost control should be done well.

At present, there are many problems when organizations establish process performance model. Organizations usually rely on personal experiences. Collected project data are not enough. The quality of historical data is not high. Modeling method is not correct, established model is useless. Thus, organizations expect to use an existing, general process performance model as a foundation, and make a few adjustments according to the actual situations, and then establish the organization's process performance model as soon as possible.

COCOMO II is one of the most widely used constructive cost estimation model. Not only it can be used in software cost estimation, but also it can control software cost by adjusting the scale factors or cost drivers.

This paper first briefly introduces COCOMO II and process performance model, then elaborates what are controllable factors, how to use COCOMO II as process performance model.

COCOMO II [2]

Overview.

COCOMO II uses object points, function points or source lines of codes as input. It provides scale factors and cost drivers to adjust the effort and cost estimation result. The basic formula is shown in formula (1) and formula (2).

$$\begin{aligned} \mathrm{PM} &= \mathrm{A} \times Size^{E} \times \textstyle\prod_{i=1}^{n} EM_i \\ \mathrm{E} &= \mathrm{B} + 0.01 \times \textstyle\sum_{j=0}^{5} SF_j \end{aligned} \tag{1}$$

$$\mathrm{COST} = \mathrm{PM} \times \text{Average Monthly Wage} \tag{2}$$

PM represents effort in Person-Month. A, B are constants, the values are 2.94 and 0.91 respectively. Size represents project size in thousand lines of source code. E is calculated with 5 scale factors. EM represents effort multiplier for cost driver. COST represents human cost, which is the result of multiplying PM and monthly average salary. Effort directly determines human cost, so these two concepts will not be distinguished in this paper. The values of scale factors and cost drivers can be found in [3].

Sub-models.

There are three sub-models in COCOMO II: application composition model, early design model and post-architecture model. They are suitable for different phases of project.

1) Application Composition Model

In early prototype development stage, many potential high risk problems need to be solved, such as user interface, interaction, performance or technical maturity. Application Composition Model is suitable for estimating this kind of work. This sub-model uses new object points [4] as size input. The effort calculation formula is shown in formula (3).

$$PM = \frac{\mathrm{NOP}}{\mathrm{PROD}} \qquad \mathrm{NOP} = \mathrm{Object\ Points} \times (1 - \mathrm{Reuse\ Percentage}) \tag{3}$$

NOP represents new object points. PROD represents developer's productivity, that is, the object points that one person can develop each month. The value of PROD is determined by table 1.

Table 1 The value of PROD

Factor			**Level**		
developers' experience/ability	very low	low	normal	high	very high
environment maturity experience/ ability	very low	low	normal	high	very high
PROD	4	7	13	25	50

2) Early Design Model

In the early design stage, alternative architectures or incremental development strategies are explored. Information in this stage is not enough to support fine-grained cost estimation. Early design model is most suitable. This sub-model uses thousand lines of source code or function points as size input, 7 coarse-grained cost drivers to adjust. The effort calculation formula is shown in formula (4).

$$\mathrm{PM} = \mathrm{A} \times Size^{E} \times \prod_{i=1}^{7} EM_i \tag{4}$$

7 cost drivers is a simplified set of 17 cost drivers in table 2. The values of these 7 cost drivers are determined by the sum of corresponding post-architecture factor values. The specific name of these 7 cost drivers and calculation method can be found in [3].

3) Post-Architecture Model

Once the project is ready to develop, it should have life cycle architecture and can provide more accurate information for determining cost drivers. Post-architecture model can be used at this time. This sub-model uses thousand lines of source code or function points as size input, 17 cost drivers to adjust the effort. The effort calculation formula is shown in formula (5).

$$\mathrm{PM} = \mathrm{A} \times Size^{E} \times \prod_{i=1}^{17} EM_i \tag{5}$$

Process Performance Model

The Concept of Software Process Performance Model.

Software process performance model (PPM) is mainly used to describe the relationship between attributes and process product. It uses historical project data to establish the mathematical model, and then this model can be used to predict the target value. If estimated value can't meet the target, controllable factors in PPM need to be adjusted to ensure the achievement of organizational targets. Therefore, PPM can predict and control target value. The most typical usage of PPM is to collect process and product measurement data throughout life cycle to predict the target value that can only be measured until late in the project [5]. Software organizations use PPM to predict and monitor software development process more accurately.

The Application of PPM[6].

When PPM is applied in project level, it is used to estimate, analyze and predict the performance of project's defined process and the feasibility of target. When choosing the processes of project, it is used to determine whether each process meet the defined targets and help to choose the optimal sub-processes. When the project is implemented, it is used to predict the feasibility of target. If the target cannot be achieved, you need to adjust the controllable factors of PPM.

Predict and Control Cost Using COCOMO II

Analyze the Feasibility of Target in the Early Stage.

When the organization establishes the project cost target, it may only depend on subjective judgment by experiences. It has no scientific basis and difficult to achieve quantitative project management. The project personnel have no reliable basis to persuade managers, customers or salesmen that the budget they have put forward is unrealistic. So, it is necessary to estimate project cost in order to analyze the feasibility of cost target in the early stage.

At early stage, the project information is not enough. Application composition model can be used to estimate the effort and cost. New object points, the developers' experience/ability, environment maturity experience/ability are needed. Cost estimation in COCOMO II only includes the human cost. The cost of hardware facilities or services needs extra estimation. If estimation cost is much higher than the target cost, cost target should be modified, or low priority function modules should be removed. If estimation cost is much lower than target cost, requirements analysts should check with customers that whether some requirements are left out.

Predict Target Feasibility and Adjust Controllable Factors.

In the project progress, three milestones should be set up. They are when prototype design is finished, outline design is finished and coding phase begins. Cost percentages of each stage are determined according to organization's historical experience. When the project reach the milestone, actual value of controllable factor will be used to estimate the cost. Target cost minus current cost and get the remaining cost, named C1. Estimation cost multiplies by the remaining percentage and get C2. If C1 is bigger than C2, it means there is cost saving in the early stage. It should be paid attention that if prophase work is done insufficiently, if that happens, it could lead to more work in later development phase. If C1 is less than C2, it means cost in early stage is beyond expected. Controllable factors need to be adjusted in order to achieve cost target.

After prototype design is finished, the developers' experience/ability and environment maturity experience/ ability need to be collected. Application composition model can predict cost with these two parameters. When outline design is finished, 5 scale factors and 7 coarse-grained cost drivers need to be collected. Early design model can predict cost with these parameters. When Coding phase begins, 5 scale factors and 17 cost drivers need to be collected. Post-architecture model can predict cost with these parameters.

Controllable factors in COCOMO II are scale factors or cost drivers, but not all scale factors and cost drivers are controllable. User should choose controllable factors from scale factors and cost drivers, and measure whether it is valuable to adjust the factors to ensure the achievement of cost targets. For example, if user improves PCAP (Programming Capability) by increasing the number of developers, it will reduce cost, but it may actually increase cost since organization should pay for more employees. So the difference determines whether this adjustment makes sense.

1) Scale Factors

Effort is reduced exponentially by adjusting scale factors. PREC and FLEX is inherent and uncontrollable. The other three scale factors are controllable in project management.

RESL represents the design thoroughness and risk elimination after review, review frequency and effectiveness determine the value.

TEAM shows that the main reason for project confusion is the difficulty to coordinate stakeholders, including users, customers, developers, maintenance personnel and coordinator, etc. The differences of stakeholders' targets and culture, the difficulty to balance the target, experience and familiarity of stakeholders in the team are main influence factors to TEAM. Through team building activities or absorbing employees who have cooperation experiences for many years can improve the level of TEAM.

PMAT is measured by capability maturity model (CMMI) from SEI. It is determined by the organization's software process capability. It costs a lot to upgrade this value. The timing to determine the level is at the beginning of the project. Once the project starts, this factor is not controllable.

The differences between levels can be found in [3]. The changes in effort after changing scale factor's level can be calculated with formula (1). Supposing the project size is 10 KSLOC, table 2 lists the changes in effort after changing the level of scale factor from the "standard" to "high".

Table 2 Changes in effort after changing level of scale factor

Scale Factor	**Level Changes**	**Difference Between Levels**	**Percentage of Effort Reduction**	**Change of Effort**
RESL	Nominal to High	1.41	$1-1/Size^{0.0141}$	3.2% reduction
TEAM	Nominal to High	1.1	$1-1/Size^{0.011}$	2.5% reduction
PMAT	Nominal to High	1.56	$1-1/Size^{0.0156}$	3.5% reduction

2) Cost Drivers

a) Product

RELY represents the degree that software must act normally in a period of time. This cost driver is basically not controllable since no organization will pay extra effort to provide higher reliability than expected. Blindly reducing reliability to cut down the effort will lead to higher cost.

DATA represents the effect that test requirements have on product development. It is calculated by dividing D by P. D means the bytes of the test database and P means bytes of source code. The factor is not controllable, test data are usually necessary.

CPLX is related with five aspects of the operations, including control operations, calculation operations, equipment related operations, data management operations and user interface operations. User can control this factor in two ways: on the one hand, user can simplify the functions. Unnecessary functions will only increase the ineffective effort. On the other hand, user can simplify the processing logic. Unnecessary complexity is not the characteristic of a good product.

RUSE shows the extra works which are needed to construct reusable components. These extra works are mainly for making a more general software design, writing more detail documents and doing more extensive tests. Users need to balance between reusability level and cost reduction. If the priority of cost reduction is higher, then user can decrease the reusability level.

DOCU represents the degree that project documentation match life cycle requirements. If user tries to save cost by setting the document level as "low" or "very low", it will usually result in extra cost in maintenance stage.

b)Platform

TIME represents the ratio between the execution time that software system consumes and available execution time. By speeding up computation, you don't have to pay more effort to satisfy the execution time constraints, which will eventually reduce the effort. The cost for speeding up the computation also needs to be considered.

STOR represents the ratio between the main memory resources that software system consumes and available main memory resources. By enlarging storage capacity, you don't have to pay more effort to satisfy the storage constraints, which will eventually reduce the effort. The cost for enlarging storage capacity also needs to be considered.

PVOL represents the Volatility of all used hardware and software products in the development. There are three ways to Reduce Volatility level. One is packaging the changes in the phased delivery rather than constantly adding changes. The second is to choose more mature and stable foundation support system. The third is using information hiding technology, that is, listing possible changes in the early stage, and splitting these changes into different modules when partitioning modules [7].

c) Personnel

Analysts in ACAP include people who are in charge of requirements analysis, outline design and detail design. Analysis and design ability, efficiency, thoroughness, communication ability and cooperation ability are considered. By raising the level of analyst, the effort can be reduced.

PCAP represents the programmers' ability, efficiency, thoroughness, communication ability and cooperation ability. By raising the level of programmers, the effort can be reduced.

PCON is determined by the project staff turnover.

The level of above three factors can be changed by personnel placement, motivating and management measures.

1. Personnel placement

The five principles of personnel placement are as following.

1) Top talent principle. With fewer but more productive staffs, communication costs are reduced.
2) Task matching principle. Assign appropriate tasks according to the capability of each person.
3) Career development principle. Improving their software professional skills can help employees realize their self-value.
4) Team balance principle. Choose people who can make up for each other and coordinate with each other in both technical skills and character.
5) Phasing out principle. Only keep qualified members, which is good to the whole team.

2. Personnel incentive

Organization can find the appropriate incentives according to the actual situation of the personnel, which includes individual achievements, the opportunity of personal development, appreciation, promotion, responsibility, interpersonal relationship, salary and etc.

3. Personnel management

Efficient management can not only promote the efficiency, but also coordinate the software development process. Improving the management level or using more reasonable and efficient management methods can benefit the cost control.

APEX is determined by the experience of project team. Platform in PLEX includes graphical user interface, database, network and distributed middleware and etc. LTEX represents the experience that project team use programming language and software tools. These three factors are related to experience. Application experience can be promoted by introducing people who have similar project experience. Platform experience, language and tool experience can be improved by increasing relevant professional skill training.

d) Project

TOOL represents the integration level and maturity of used tools. If current tool support is insufficient, buying integrated software tools would be a good choice. When buying tool, it must be considered that if this tool can improve the level of TOOL. A thorough cost-benefit analysis should be made, the cost of introducing new tool should be considered, and the cost benefit that the new tool can bring should also be considered.

SITE is determined by site distribution and exchange methods. Gathering project team members together is good for face-to-face communication. Improving the means of communication will greatly reduce the communication cost.

SCED represents the schedule constraint that is added to project development. Compressing schedule means more effort in the early stage to eliminate risks and refine the architecture, and more effort in the late stage to complete more testing and documentation work in parallel. So compressing schedule is useless for cost control.

The difference between levels can be found in [3], assuming the difference caused by higher or lower cost driver level is α. By using formula (1), the changes of effort can be calculated. Table 3 shows some examples.

Table 3 Changes in effort after changing the level of cost drivers

Cost Driver	**Level Changes**	**Difference Between Levels**	**Percentage of Effort Reduction**	**Change of Effort**
CPLX	High to Nominal	0.17	$100\alpha/EM_{CPLX}$	14.5% reduction
RUSE	High to Nominal	0.07	$100\alpha/EM_{RUSE}$	6.5% reduction
PMAT	High to Nominal	0.11	$100\alpha/EM_{DOCU}$	10% reduction
TIME	High to Nominal	0.11	$100\alpha/EM_{TIME}$	10% reduction
STOR	High to Nominal	0.05	$100\alpha/EM_{STOR}$	4.8% reduction
PVOL	High to Nominal	0.15	$100\alpha/EM_{PVOL}$	13% reduction
ACAP	Nominal to High	0.15	$100\alpha/EM_{ACAP}$	15% reduction
PCAP	Nominal to High	0.12	$100\alpha/EM_{PCAP}$	12% reduction
PCON	Nominal to High	0.1	$100\alpha/EM_{PCON}$	10% reduction
APEX	Nominal to High	0.12	$100\alpha/EM_{APEX}$	12% reduction
PLEX	Nominal to High	0.09	$100\alpha/EM_{PLEX}$	9% reduction
LTEX	Nominal to High	0.09	$100\alpha/EM_{LTEX}$	9% reduction
TOOL	Nominal to High	0.1	$100\alpha/EM_{TOOL}$	10% reduction
SITE	Nominal to High	0.07	$100\alpha/EM_{SITE}$	7% reduction
SCED	Nominal to High	0	$100\alpha/EM_{SCED}$	0% reduction

Conclusion

This paper shows how to use COCOMO II as a PPM to estimate and control software cost. It also explains what factors are controllable and the effectiveness of adjusting each controllable factor. The next step in this research is to track multiple projects, collect the value of controllable factors, estimate and control cost and verify the accuracy and reliability of COCOMO II.

References

[1]. New Standish Group report shows more Project failing and less successful projects: http://www.standishgroup.com/chaos_news/newsletter.php?id=54, 2013.

[2]. COCOMO II Model Definition Manual: http://wenku. baidu.com/link?url=r6DpwDbFeUG93Y

X_gpLTlegBirLxu0gst50c50yJ6oYYItSqDn7P1DRtKY5JrYTI4Z5O0VdFMmV4O-yAfaJN37Od9j0ppTmKeNwzPKw_u1S.

[3]. Barry W.Boehm. Software cost Estimation with COCOMO II. Translated by Li Shixian. Beijing: China Machine Press, 2005, p. 22-29.

[4]. Software Engineering: Theory and Practice: http://wenku.baidu.com/view/18ec987f16888868762d6f9.html.

[5]. GAO Li-e, KANG Feng-ju, LIU Wei-dong, et al. Research on Real-time Multitask Scheduling Based on Timed Petri Nets. Journal of System Simulation, Vol. 18(2006) No. 11, p. 3075-3077, 3147.

[6]. Lv Yingjie. Engineering Methods Research for Software Process Performance Mode (Master, Shanghai Jiao Tong University, China 2011). p. 25-28.

[7]. Barry W.Boehm. Software Engineering Economics. Translated by Li Shixian. Beijing: China Machine Press, 2004, p. 518-551.