Editor: Donald J. Reifer ■ Reifer Consultants ■ d.reifer@ieee.org

Safe and Simple Software Cost Analysis

Barry Boehm

"Everything should be as simple as possible, but no simpler." – Albert Einstein

Simple software cost-analysis methods are readily available, but they aren't always safe. The simplest method is to base your cost estimate on the typical costs or productivity rates of your previous projects. That approach will work well if your new project doesn't have any cost-critical differences from those previous projects. But it won't be safe if some critical cost driver has degraded.

Simple history-based software cost-analysis methods would be safer if you could identify which cost driver factors were likely to cause critical cost differences and estimate how much cost difference would result if a critical cost driver changed by a given degree. In this column, I'll provide a safe and simple method for doing both of these by using some recently published cost estimating relationships (*Software Cost Estimation with COCOMO II*, by Barry Boehm et al., Prentice Hall, 2000). COCOMO II is an updated and recalibrated version of the Constructive Cost Model (COCOMO) originally published in *Software Engineering Economics* (by Barry Boehm, Prentice Hall, 1981). I'll also show how the COCOMO II cost drivers let you perform cost sensitivity and trade-off analyses, and discuss how you can use similar methods with other software cost-estimation models.

COCOMO II Productivity Ranges

Figure 1 shows the relative productivity ranges (PRs) of the major COCOMO II cost-driver factors, as compared to those in the original COCOMO 81 model. A factor's productivity range is the ratio of the project's productivity for the best possible factor rating to its worst possible factor rating, assuming that the ratings for all other factors remain constant. Here, we define relative "productivity" in either source lines of code (SLOCs) or function points per person-month. The term "part" in Figure 1 reflects the fact that the COCOMO 81 development mode involved a combination of development flexibility and precedentedness.

For example, suppose your business software group has been developing similar applications for 10 years on mainframe computers and their next application is to run on a micro-based client-server platform. In this case, the only cost-driver factor likely to cause significant cost differences is the group's platform experience (assuming that such factors as platform volatility and use of software tools do not change much). Figure 1 shows that the productivity range for the platform experience factor is 1.40. Thus, changing from the best level of platform experience (over 6 years) to the worst level (less than 2 months) will increase the amount of effort required for the project by a factor of 1.40, or by 40%.

If we compare the productivity ranges for platform experience between COCOMO II and COCOMO 81, we see that they are fairly close (1.40 versus 1.34). That's true for most of the cost-driver factors. In particular, personnel and team capability remains the single strongest influence on a software project's productivity. Its productivity range in COCOMO II is somewhat smaller than in COCOMO 81 (3.53 versus 4.18), but that might be because COCOMO II has added two more people-related factors (team cohesion and personnel continuity), which might account for some of the variance previously associated with personnel capability.

COCOMO II has some additional new cost drivers besides team cohesion and personnel continuity:

- software developed for reuse;
- architecture and risk resolution;
- software process maturity (somewhat replacing use of modern programming practices);
- documentation match to life cycle needs; and
- multisite development (somewhat replacing turnaround time).

Each of these new factors has a statistically significant greater-than-1.0 productivity range in the multiple regression analysis of the 161 projects in the COCOMO II database. The one exception was the "developed for reuse" factor, which had insufficient dispersion in its rating levels to produce a statistically significant result. We determined its productivity range of 1.31 (and the productivity ranges of the other COCOMO II factors) using a Bayesian weighted average of data-determined multiple regression results and expert-determined Delphi results. When the data-determined values are weakly determined, the weighted average will give a higher weight to the expert-determined values. (See Chapter 4 of the COCOMO II book for a detailed discussion.)

Some factors in Figure 1 have an asterisk, indicating that their productivity ranges vary by size, because they enter the COCOMO II estimation formula as exponential functions of size rather than as effort multipliers. The productivity ranges shown in Figure 1 for these size-sensitive variables are for a product whose size is 100,000 source lines of code (100 KSLOC). Table 1 shows how the productivity ranges for these factors vary by size.

Thus, for example, the effect of varying a project's process maturity from a low Level 1 to a Level 5 on the SEI Capability Maturity Model scale will typically improve productivity by only 20% on a 10-KSLOC project, but it will improve productivity by 71% on a 1,000-KSLOC project, mostly by reducing rework. These ranges probably under-estimate the effect of high maturity levels, as they normalize out the effects of other productivity factors such as the use of software tools and reusable software components. Details are available in Bradford Clark's USC PhD dissertation, "The Effects of Software Process Maturity on Software Development Effort," USC-CSE-TR-97-510, available via the USC-CSE COCOMO II Web site (see the "Leading Software Cost-Estimation Tools" box), under "Related Research."

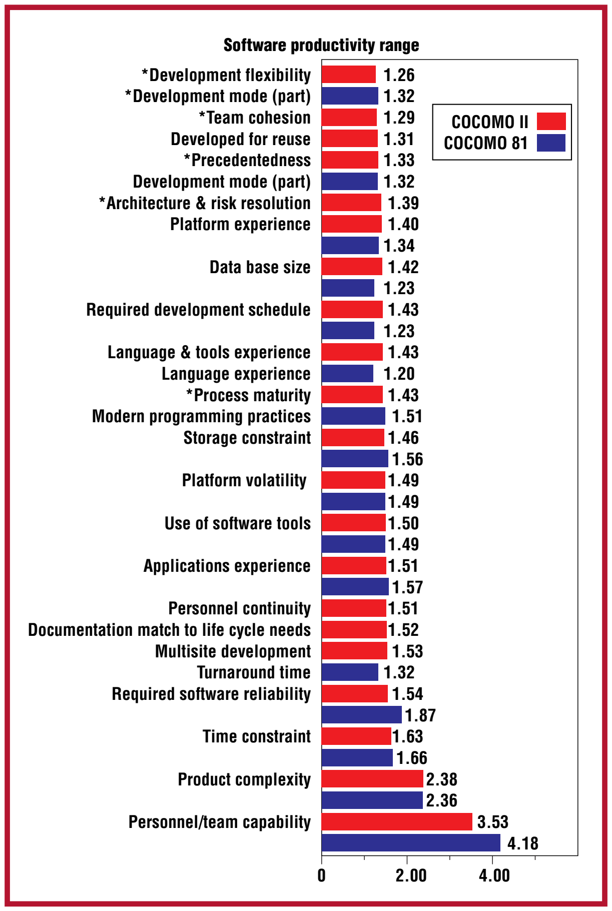

Figure 1. Comparing COCOMO 81 and COCOMO II. (*Varies by size; see Table 1. Values for 100 KSLOC products.)

Safe and Simple Software Cost Analysis

To return to our platform experience example, you might find that you can do better than a cost increase of 40% if you can find a few people with some microprocessor-based client-server platform experience for your project. For each cost-driver factor, COCOMO II provides rating scales and tables showing how productivity will vary for each rating level. Table 2 shows the resulting effort multipliers by rating scale for each of the personnel-experience cost factors in COCOMO II.

These tables will let you perform

Table 1

Size-Dependent Productivity Ranges

	Size (KSLOC)		
Factor	**10**	**100**	**1,000**
Development flexibility	1.12	1.26	1.42
Team cohesion	1.13	1.29	1.46
Precedentedness	1.15	1.33	1.53
Architecture and risk resolution	1.18	1.39	1.63
Process maturity	1.20	1.43	1.71

Table 2

COCOMO II Effort Multipliers Versus Length of Experience

	Average length of experience					
Type of experience	< 2 mo	6 mo	1 year	3 years	> 6 years	PR
With platform	1.19	1.09	1.00	0.91	0.85	1.40
With languages and tools	1.20	1.09	1.00	0.91	0.84	1.43
With application area	1.22	1.10	1.00	0.88	0.81	1.51

Table 3

The Effect of Adding Experienced Client-Server Developers

	Effort multipliers platform x application = product			
Options	Platform	Application	Product	Person-months
A	1.09	0.81	0.88	51
B	1.00	0.88	0.88	51

Table 4

The Effect of Changing Programming Languages and Tools

	Effort multipliers Platform x application x language and tools = product				
Options	Platform	Application	Lang. & tools	Product	Person-months
A	1.09	0.81	1.09	0.96	56
B	1.00	0.88	1.00	0.88	51

the following safe and simple three-step cost-estimation method, illustrated by the personnel-experience cost factors I've described.

1) *Estimate the new project's productivity and effort based on previous experience*. The new project is sized at 20,000 SLOC. Productivity for previous mainframe applications was 500 LOC/person month (PM), yielding an effort estimate of 40 PM. Multiply by your labor rate/PM to estimate the cost.
2) *Determine which COCOMO II cost-driver factors will change for the new project and by how much*. With the current staff, platform experience will go from less than 6 years to more than 2 months. By going to an Option A, with a couple of fairly experienced client-server developers added to the team, the average length of platform experience on the new project would be 6 months.
3) *Use the COCOMO II effort multipliers to revise the original estimate*. Going from less than 6 years of platform experience to more than 2 months changes the effort multiplier from 0.85 to 1.19—an increase of a factor of 1.19/0.85 = 1.40. This leads to a revised effort estimate of 40 PM (1.40) = 56 PM. Going to a platform experience of 6 months (Option A) changes the effort multiplier from 0.85 to 1.09, an increase of a factor of 1.09/0.85 = 1.28. The revised effort estimate is then (40 PM) (1.28) = 51 PM.

For easy reference, the COCOMO II book has the full set of these cost-driver effort multiplier tables on its inside cover.

Simple Sensitivity and Trade-off Analysis

You can extend this approach to cover sensitivity and trade-off analysis among several cost-driver factors. For example, suppose you also had an Option B to add a couple of extremely experienced client-server developers, increasing your average platform experience to 1 year. However, they have very little business applications experience, decreasing your average applications experience from ≥ 6 years to 3 years. In this case, Table 3 shows the comparison to the previous Option A.

This tables shows that Option A and B are roughly equivalent in effort and cost (unless there are major differences in salary levels), in which case the project can use other criteria to choose between A and B (such as the opportunity for client-server expert-mentoring and risk reduction with Option B).

As a further example, the comparison between Options A and B would be different if the new project involved a change in programming languages and tools (such as from Cobol to Java), causing the average experience for this factor also to be 6 months for Option A and 1 year for Option B. Table 4 shows the new comparison.

In this case, the added 9% benefit in Option B makes it look more attractive from a cost and effort standpoint.

Thus, we can see that the COCOMO II rating scales and effort multipliers provide a rich quantitative framework for exploring software project and organizational tradeoff and sensitivity analysis. The framework lets the project manager explore alternative staffing options involving various mixes of application, platform, and language and tool experience. Or, an organization-level manager could explore various options for transitioning a portfolio of applications from their current application/platform/language configuration to a desired new configuration (for example, by using pilot projects to build up experience levels).

You can perform these analyses either by running the COCOMO II model or by doing calculator- or spreadsheet-based analysis using the published multiplier values in Table 2 or in the COCOMO II book. You can perform similar tradeoff analyses by running commercial COCOMO II tools or alternate proprietary cost models (see the box for Web sites of major software cost models).

It is generally a good practice to compare analysis results between two or perhaps more software cost models, as each reflects a somewhat different experience base. Some particular advantages of having one of these models be COCOMO II are that all of its detailed definitions and internals are available for examination; the Bayesian methods by which it combines expert judgment and data analysis are well defined and the evidence of statistical significance of each cost driver factor is provided. From an expectations-management standpoint, however, there is no guarantee that COCOMO II can fit every organization's style of development, data definitions, and set of operating assumptions. For example, it is only calibrated to organizations and projects that collect carefully defined data that maps to COCOMO II's definitions and assumptions. If your organization operates in a different mode, the COCOMO II-based analyses will likely provide useful relative guidance, but less precise cost estimates and payoff factors. And if your organization does collect carefully defined data, the analyses based on COCOMO II or other models will be stronger if you use the data to calibrate the models to your experience.

Barry Boehm is the TRW Professor of Software Engineering in the Computer Science Department at the University of Southern California and the director of the USC Center for Software Engineering. His research interests include software process modeling, requirements engineering, architectures, metrics and cost models, and engineering environments, and knowledge-based software engineering. He has a BA from Harvard and an MS and PhD from UCLA, all in mathematics. He is an AIAA Fellow, ACM Fellow, and IEEE Fellow, and a member of the National Academy of Engineering. Contact him at boehm@sunset.usc.edu.

Leading Software Cost-Estimation Tools

These URLs provide descriptions of leading software cost-estimation tools. Each tool cited has been backed by a lot of effort to relate the tool to a wide variety of software project experiences. My apologies if the list fails to include your favorite tool.

- Estimacs (Computer Associates Int'l): www.cai.com/products/estimacs.htm
- Knowledge PLAN (Software Productivity Research/Artemis): www. spr.com
- PRICE S (Price Systems): www.pricesystems.com
- SEER (Galorath, Inc.): www.galorath.com
- SLIM (Quantitative Software Management): www.qsm.com

COCOMO II-based tools:

- COSTAR (Softstar Systems): www.softstarsystems.com
- CostXpert (Marotz, Inc.): www.costxpert.com
- Estimate Professional (Software Productivity Center): spc.ca/products/estimate
- USC COCOMO II.2000 (USC Center for Software Engineering): http://sunset.usc.edu/research/COCOMOII