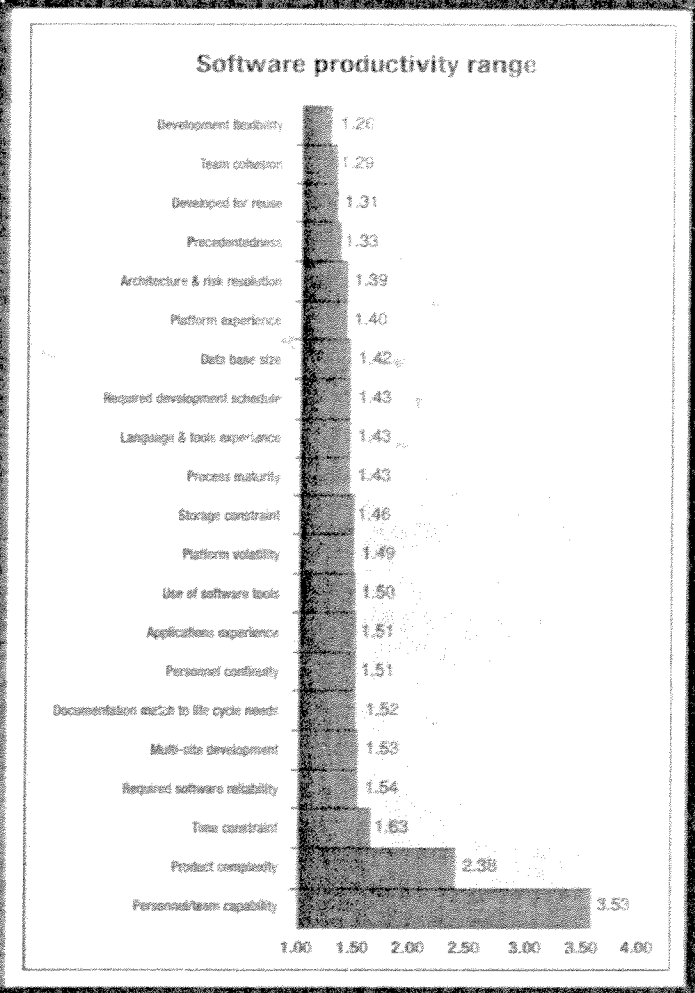
EXHIBIT 11



SOFTWARE COST ESTIMATION WITH COCOMO II

Barry W. Boehm · Chris Abts · A. Winsor Brown
Sunita Chulani · Bradford K. Clark · Ellis Horowitz
Ray Madachy · Donald Reifer · Bert Steece

Library of Congress Cataloging-in-Publication Data

Software cost estimation with Cocomo II / by Barry W. Boehm . . . [et al.].
p. cm.
Includes bibliographical references and index.
ISBN 0-13-026692-2
1. Computer software--Costs. I. Boehm, Barry W.
QA76.76.C73 S64 2000
005.3'068'1--dc21 00-032644

Acquisitions editor: *Paul Petralia*
Cover designer: *Talar Agasyan*
Cover design director: *Jerry Votta*
Manufacturing manager: *Maura Goldstaub*
Marketing manager: *Bryan Gambrel*
Editorial assistant: *Justin Somma*
Project coordinator: *Anne Trowbridge*
Compositor/Production services: *Pine Tree Composition, Inc.*

Prentice-Hall, Inc.
Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.

The publisher offers discounts on this book when ordered in bulk quantities. For more information contact:

Corporate Sales Department
Phone: 800-382-3419
Fax: 201-236-7141
E-mail: corpsales@prenhall.com

Or write:

Prentice Hall PTR
Corp. Sales Dept.
One Lake Street
Upper Saddle River, New Jersey 07458

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN: 0-13-702576-9

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte Ltd.
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Code size is expressed in thousands of source lines of code (KSLOC). A source line of code is generally meant to exclude nondelivered support software such as test drivers. However, if these are developed with the same care as delivered software, with their own reviews, test plans, documentation, etc., then they should be counted [Boehm 1981, pp. 58–59]. The goal is to measure the amount of intellectual work put into program development.

Defining a line of code is difficult because of conceptual differences involved in accounting for executable statements and data declarations in different languages. Difficulties arise when trying to define consistent measures across different programming languages. In COCOMO II, the logical source statement has been chosen as the standard line of code. The Software Engineering Institute (SEI) definition checklist for a logical source statement is used in defining the line of code measure. The SEI has developed this checklist as part of a system of definition checklists, report forms and supplemental forms to support measurement definitions [Park 1992, Goethert et al. 1992].

Figure 2.1 shows the SLOC definition checklist as it is being applied to support the development of the COCOMO II model. Each checkmark in the "Includes" column identifies a particular statement type or attribute included in the definition, and vice versa for the excludes. Other sections in the definition clarify statement attributes for usage, delivery, functionality, replications, and development status. The full checklist is provided at the end of this chapter in Table 2.53.

Some changes were made to the line-of-code definition that depart from the default definition provided in [Park 1992]. These changes eliminate categories of software, which are generally small sources of project effort. For example, not included in the definition are commercial-off-the-shelf software (COTS), government-furnished software (GFS), other products, language support libraries and operating systems, or other commercial libraries. Code generated with source code generators is handled by counting separate operator directives as lines of source code. It is admittedly difficult to count "directives" in a highly visual programming system. As this approach becomes better understood, we hope to provide more specific counting rules. For general source code sizing approaches, such as PERT sizing, expert consensus, analogy, top-down, and bottom-up, see Section 21.4 and Chapter 22 of [Boehm 1981].

2.2.2 Counting Unadjusted Function Points (UFP)

The function point cost estimation approach is based on the amount of functionality in a software project and a set of individual project factors [Behrens 1983; Kunkler 1983; IFPUG 1994]. Function points are useful estimators since they are based on information that is available early in the project life cycle. A brief summary of function points and their calculation in support of COCOMO II follows.

Function points measure a software project by quantifying the information processing functionality associated with major external data or control input, out-

clearly believe that Model E captures some software phenomena better than Model D, and may be a better estimator for future projects. But others don't.

Now, suppose you have a new project to estimate. Model D estimates it will take 122 person-months (PM) to complete; Model E estimates 236 PM. Which value should you choose? If you want to average the estimates, do you just go halfway, at 179 PM? How do you justify your estimate to your management, your venture capitalist, or to the source selection authority for your competitive proposal?

4.1.1 Bayesian Calibration

What you would like is a technique for creating a Model B (Balanced), that favors the experts for cost drivers where they are in strong agreement and the data fit is weak, and favors the data fit for cost drivers where it is strong and the experts disagree. You would also like a technique that has a strong theoretical foundation to justify its results.

This technique is provided by the Bayesian approach to determining model parameters [Box-Tiao 1973; Gelman et al. 1995]. In the COCOMO II version of the Bayesian approach, the Model E parameter values and their variances are taken as the *a priori* knowledge about the parameter values. The Model D parameter values and their variances are then taken as new information which can be used to determine an *a posteriori* update of the parameter values. The Bayesian approach basically produces a weighted average of the Model D and E values, which gives higher weights to parameter values with smaller variances. The detailed approach and formulas are provided in this chapter.

We encountered the Model E–Model D problem when calibrating the 1997 version of COCOMO II to eighty-three project data points. Because of data imprecision and lack of dispersion of rating values, one of the cost-driver parameters (for Develop for Reusability) emerged from the Model D analysis with a negative sign (indicating a trend opposite to expert judgment) and a large variance. Model D's estimates for the eighty-three 1997 data points came within 30 percent of the actuals 64 percent of the time [PRED(.30) = 64%]. But when it was used on the calibration year 2000 sample of 161 data points, it achieved only a PRED(.30) of 44 percent.

When we applied the Bayesian approach to the Model E and 1997 Model D parameter values and their variances, the Develop for Reusability parameter emerged with a positive sign. The resulting Bayesian model achieved only a PRED(.30) of 58 percent on the 1997 data, but it achieved a much stronger PRED (.30) of 64 percent on the 2000 sample of 161 data points.

Using the Bayesian approach to combine Model E with a 2000 Model D calibrated to the full 2000 set of 161 data points produced even stronger results: a PRED(.30) of 75 percent for the general calibration and a PRED(.30) of 80 percent when each organization's data was separately calibrated to its own coefficient value. Thus, we believe that the Bayesian calibration approach has provided a ro-

bust estimation model (now called COCOMO II.2000) with a sound underlying rationale for its results. Again, further specifics are provided in this chapter and also in further detail in [Chulani 1999 and Chulani et al. 1999].

4.1.2 COCOMO II Modeling Methodology

Before one can poll experts and collect data on the parameters of an estimation model, one needs to carefully define the parameters. Also, one needs to have a good idea of how the parameters are going to be used to produce effective estimates.

This means that one needs to do a lot of early homework to determine an appropriate functional form for the estimation model, and to determine which parameters are most likely to significantly influence the quantity being estimated. One needs to ensure that the parameters are sufficiently well defined to obtain consistent values for parameter estimates and project data.

We have developed a seven-step modeling methodology to minimize the risk that we consume a lot of people's time and effort supplying data that produces poor estimates. We began by using it for COCOMO II and have continued to use it for COCOTS, COQUALMO, COPSEMO, and CORADMO.

4.1.2.1 Modeling Steps

The seven steps are summarized in Figure 4.1.

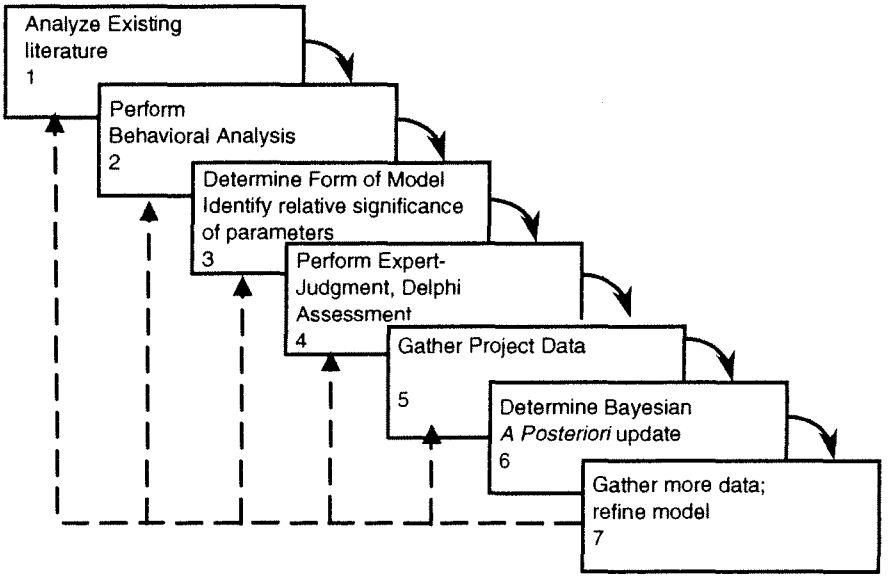


Figure 4.1 COCOMO II Modeling Methodology