Future Trends, Implications in Cost Estimation Models

The rapid pace of change in software technology requires everybody in the software business to continually rethink and update their practices just to stay relevant and effective. This article discusses this challenge first with respect to the USC COCOMO II software cost modeling project, and then for software-intensive organizations in general. It then presents a series of adaptive feedback loops by which organizations can use COCOMO II-type models to help cope with the challenges of change.

A major team effort was recently completed to re-engineer the original Constructive Cost Model (COCOMO) for software cost and schedule estimation into a new model, COCOMO II. The overall COCOMO framework remained about the same, but significant changes were found to be necessary to keep pace with the changing nature of software development and evolution. These trends have included a move away from the Waterfall process model toward evolutionary, incremental, and spiral models; product line management approaches to software reuse; applications composition capabilities; and graphic user interface builder tools that made traditional size metrics such as source lines of code (SLOC) inappropriate.

We have replaced the COCOMO development modes (organic, semidetached, embedded) by a set of scale factors (precedentedness, development flexibility, architecture and risk resolution, team cohesiveness, and process maturity). These enable project managers to control that which affects their project's economies and diseconomies of scale. We added some multiplicative cost drivers (development for reuse, degree of documentation, multisite development); dropped the Turnaround Time cost driver; merged the Modern Programming Practices cost driver into the process maturity scale factor, and changed the requirements volatility cost driver into a size factor.

We changed the main size parameter from Delivered Source Instructions to a user-determined mix of SLOC and function points; changed to a more detailed nonlinear model of software reuse effects; and provided a family of models (Applications Composition, Early Design, and Post-Architecture) tuned to the information available at different stages of the development process. We developed a Bayesian approach to calibration of COCOMO II to 161 projects from 18 organizations, resulting in a model that estimates within 30 percent of the actual effort 75 percent of the time (80 percent of the time if calibrated to the individual organizations' data). A book describing the model is to be released in June; it will include a CD with a USC COCOMO II tool and demo versions of three commercial implementations [1]. Further information about COCOMO II is available at http://sunset.usc.edu/COCOMOII/suite.html

Rather than leaving the model as is for the next 18 years as with the original COCOMO, we are determining extensions to COCOMO II to address emerging trends such as Rapid Application Development (RAD) and commercial off-the-shelf (COTS) integration. This need to continually update your software estimation capabilities also holds for most organizations. This paper explores the reasons for this and some of the implications.

Trends in Software Productivity, Estimating Accuracy

In principle, your organization should be able to continuously measure, recalibrate, and refine models such as COCOMO II to converge uniformly toward perfection in understanding your software applications and in accurately estimating the costs and schedules.

In practice, convergence toward perfection in estimation is not likely to be uniform. Two major phenomena are likely to interrupt your progress in estimation accuracy:

1. As your understanding increases about the nature of your applications domain, you will also be able to improve your software productivity and quality by using larger solution components and more powerful applications definition languages. Changing to these construction methods will require you to revise your estimation techniques, and will cause your estimation error to increase.

2. The overall pace of change via new technologies and paradigm shifts in the nature of software products, processes, organizations, and people will cause the inputs and outputs of software estimation models to change. Again, these changes are likely to improve software productivity and quality, but cause your estimation error to increase.

Effects of Increasing Domain Understanding

Suppose you are entering a new applications domain, (e.g., control of distributed, heterogeneous, real-time automated agents for robotics devices). Your initial software productivity in this domain is likely to be low, largely due to the effects of such COCOMO II variables as precendentedness, architecture and risk resolution, complexity, and applications experience. In particular, your understanding of the architecture for such systems and your ability to reuse components will be low. And your unfamiliarity with the domain will cause your cost and schedule estimation errors to be relatively high.

As you increase your understanding of how to build such systems and their components, both your productivity and your estimation accuracy will improve. However, at some point you will understand enough about the domain to begin developing a product line architecture and reusable components to be used in future products. At this point (point A in Figure 1), your productivity will go up faster, as you will be reusing rather than developing more and more of the software (in COCOMO II terms, your equivalent SLOC will decrease for the same type of project). However, at point A, your estimation error will go up, as your previous cost driver ratings will be less relevant, and you will be starting on the learning curve in rating your reuse parameters. You will also find that reuse and product line management cause significant changes in your processes [2, 3].

As you improve your understanding of how to increase productivity and reduce estimation error in using component-based development, you will often find that other organizations in the domain are doing so as well. Soon, some of the more general
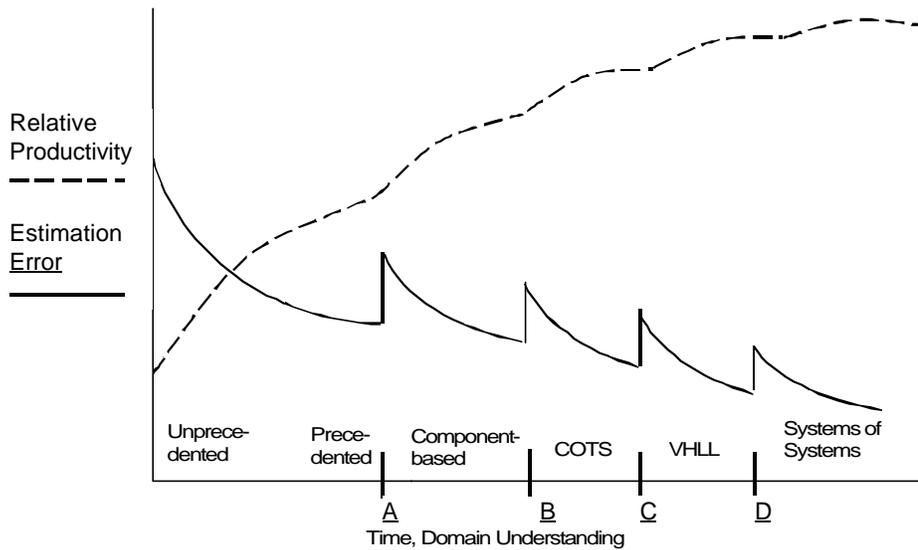
Figure 1. *Productivity and Estimation Accuracy Trends*

components will be shared across organizations or offered as COTS products. With their development and maintenance costs amortized over more and more user organizations, they become cheaper to employ than some of your reusable components. Again, using these COTS products or shared components will increase your productivity rate (point B in Figure 1). Initially, you will find it harder to predict the cost and schedule of integrating heterogeneous COTS components with your components and with each other, and your estimation error at point B will also go up.

VHLL's and System of Systems

This scenario will generally repeat itself at points C and D in Figure 1. At point C, you and/or others will know enough about how to compose domain components to be able to automate their composition, and to provide a domain-specific Very High Level Language (VHLL) with user-oriented terminology to specify the particular application desired. Productivity rates will increase (in COCOMO II terms, via the need for much fewer source lines of code), but estimation errors also initially go up.

At point D, you will find that there is a demand to closely integrate your VHLL-driven robotic device systems for subassembly manufacturing, for example, with other VHLL's and application generators for factory control systems and electronic commerce systems, into a total factory system of systems. Integrating the systems will be more productive than building a whole new factory system, but

your error in estimating cost and schedule will be higher than for an individual system. This is because of uncertainties you will have in estimating the effort required to reconcile the unpredictable incompatibilities in interfaces, priorities, assumptions, and usage conventions among the subassembly manufacturing, factory control, and electronic commerce VHLL's and systems [4].

Effects of Innovation, Change

Other sources of innovation and change may cause changes in the nature of your software projects' product, process, organization, and people. These may improve your organization's overall productivity, but their effect on your projects' practice may again increase your estimation error.

In the area of product technology, such changes have included changes from batch-processing to interactive systems, and from single mainframes to distributed and networked systems. Other product technologies such as graphic user interface builders will also increase productivity, but estimation error increases because of new challenges in determining what to count as product size.

In the area of process technology, the change from waterfall to evolutionary or spiral development requires rethinking the project's endpoints and phases. Incremental development, RAD, cost-as-independent-variable (CAIV), or schedule-as-independent-variable (SAIV) cause further rethinking of process strategies, endpoints, and phases. With CAIV or

SAIV, for example, you may specify and design more product than you deliver when you run out of budget or schedule. Collaborative processes (Joint Application Development, Integrated Product Team, etc.) require involving users, operators, and others in product definition. Should their effort be included in the estimate? To what extent will virtual-reality distributed collaboration technology improve software costs and schedules?

With organizations and people, changes in organizational objectives affect products, processes, and estimation accuracy. One example is the increasing emphasis on reducing schedule (time to market) in order to remain competitive, rather than minimizing cost. Another example is the effect of increasing emphasis on software quality as a competitive discriminator.

The effects of having tens of millions of computer-literate people will also change the nature of software products and processes. Also, the increasingly critical nature of software to an organization's competitive success creates stronger needs for integrating software estimates into business-case and financial-performance models. Trends toward human economics will affect both software products' required functions and user interfaces.

Estimation Accuracy: The Bottom Line

If only our software engineering domain understanding, product and process technology, and organization and people factors stayed constant, we could get uniformly better and better at estimating. But they do not stay constant, and their changes are generally good for people and organizations. The need to continually rethink and re-engineer our software estimation models is a necessary price to pay for the ability to incorporate software engineering improvements.

Coping with Change: COCOMO II

We are trying to ensure that COCOMO II will be adaptive to change by trying to anticipate trends in software engineering practice, as discussed in the Introduction. The resulting three-stage set of COCOMO II models (application composition, early design, post-architecture) anticipates some dimensions of

future change. Other dimensions are addressed by the new or extended cost drivers such as process maturity, architecture and risk resolution, team cohesion, multisite development, use of tools, and the various reuse parameters.

We are also attempting to anticipate future trends via our overall Model-Based (System) Architecting and Software Engineering (MBASE) project. MBASE's key objective is to avoid harmful model clashes by integrating a project's product, process, property, and success models [5]. The COCOMO II suite of models is our main effort in the property model area. Concurrently, we are integrating complementary research into product models (domain, requirements, and architecture models); process models (WinWin spiral model, process anchor points); and success models (stakeholder win-win, business case analysis, *I will know it when I see it* prototyping).

We have been trying to understand and anticipate trends in software engineering product, process, property, and success models via workshops with our affiliates, via model research, and via model experimentation with our annual series of digital library applications projects using MBASE [6]. For example, our initial formulation of the Constructive COTS Integration Cost Model (COCOTS) was based on an affiliates' workshop on COTS integration, and our efforts to incorporate COTS assessment and integration into MBASE extensions of spiral process models and object-oriented product models. Our major refinement of COCOTS into a family of four models was based on analysis of COTS integration experience data from the MBASE digital library projects.

Similarly, our formulation of the Constructive Rapid Application Development Estimation Model (CORADMO) has been based on an affiliates' RAD workshop, and on integrating RAD process models such as schedule-as-independent-variable (SAIV) into MBASE. This was done via RAD experimentation using the digital library projects. These projects are good RAD examples, as our semester constraints require them to be fully architected in 11 weeks, and fully developed and transitioned in another 12 weeks.

Thus, the emerging extensions of COCOMO II discussed in Chapter 5 of

Software Cost Estimation with COCOMO II (COCOTS, CORADMO, Applications Composition, and other models) represent hypotheses of how to model the cost, schedule, and quality effects of current and future trends in software engineering practice. As we gather more data, we will be able to test and refine these models, and to identify further models or extensions likely to be important for future software engineering practice.
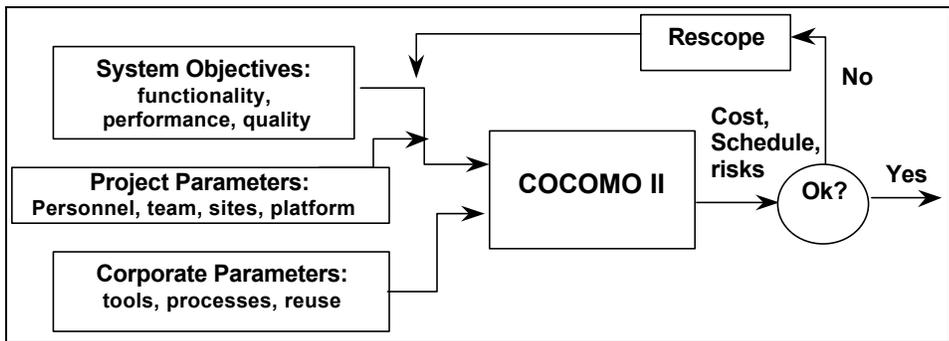
unnoticed: via personnel changes; COTS product, reusable component, or tool shortfalls; requirements creep; or platform discontinuities. In such cases, COCOMO II phase and activity distributions can be used to develop a quantitative milestone plan or an earned-value system [7] for the project, which enable plan deviations to be detected, and appropriate corrective actions to be taken (Figure 3) involving COCOMO II in project rescoping.

Figure 2. *Using COCOMO II to Cope With Change*

Coping with Change: COCOMO II and Your Organization

COCOMO II can be a useful tool for your organization to use in adapting to future change, both at the project level and at the organizational level.

Coping with Change During Project Definition

Figure 2 shows how COCOMO II can be used to help address issues of change at the project definition level. You can enter your organization's customary values via the COCOMO II parameters, and indicate which ones will undergo change. COCOMO II will estimate how these changes will affect the project's expected cost and schedule, and will provide you and your stakeholders with a framework for rescoping the project if estimated cost and schedule are unsatisfactory.

Coping with Change During Project Execution

Frequently, changes in project objectives, priorities, available componentry, or personnel occur during project execution. If these are anticipated, COCOMO II can support a variant of the project definition process above to converge on a stakeholder-satisfactory rescoping of the project.

A more serious case occurs when the changes are unanticipated and largely

Coping with Required COCOMO II Model Changes

At times, unanticipated project changes are indications that your COCOMO II model needs to be recalibrated or extended. The more management data you collect on actual project costs and schedules, the better you will be able to do this (see Figure 4).

Recalibration might be appropriate, for example, if your organization is acquired by or merged into an organization with different definitions of project endpoints, or with different definitions of which types of employees are directly charged to the project vs. being changed to overhead. As described in Chapter 4 of *Software Cost Estimation with COCOMO II*, techniques are available to recalibrate COCOMO II's base coefficients and exponents for cost and schedule estimation. Some COCOMO II tools such as USC COCOMO II and COSTAR, a commercial product from SoftStar Systems, provide such calibration features.

Extending the model will be appropriate if some factor assumed to be constant or insignificant turns out to be a significant cost driver. For example, the COCOMO 81 TOOL Factor was not in the original 1978 TRW version of COCOMO, as previous TRW projects had operated with a relatively uniform set of mainframe tools. The TOOL Factor
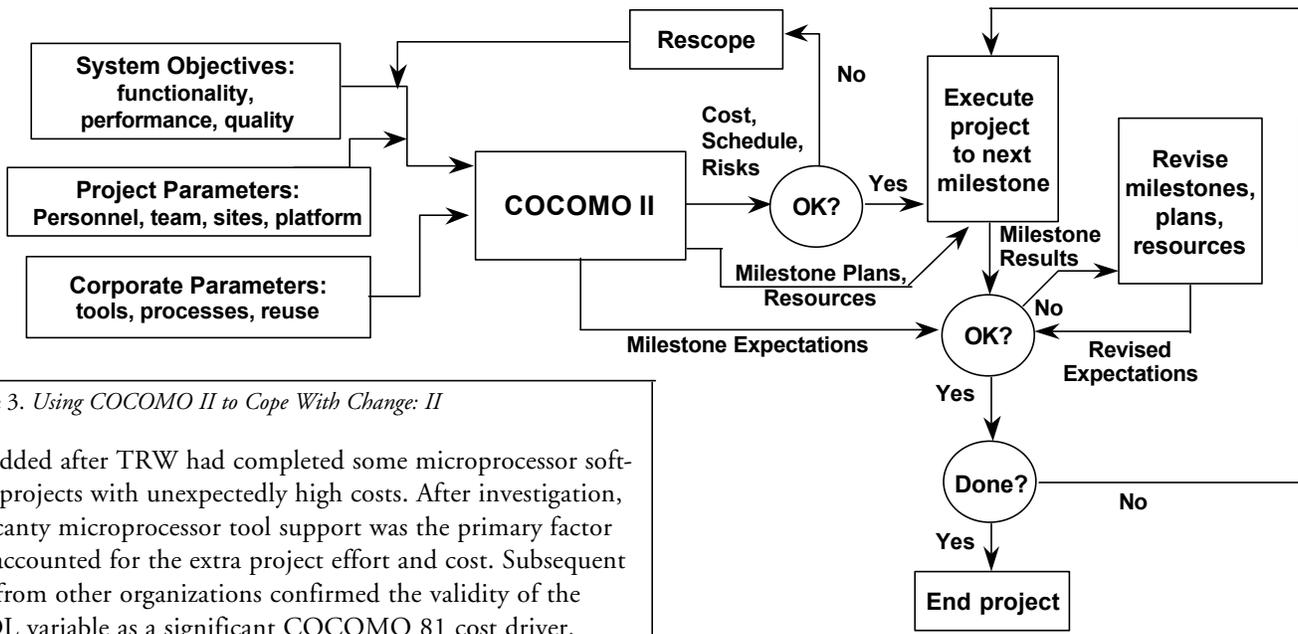
Figure 3. *Using COCOMO II to Cope With Change: II*

was added after TRW had completed some microprocessor software projects with unexpectedly high costs. After investigation, the scanty microprocessor tool support was the primary factor that accounted for the extra project effort and cost. Subsequent data from other organizations confirmed the validity of the TOOL variable as a significant COCOMO 81 cost driver.

Similarly, several variables were added to COCOMO 81 to produce COCOMO II, in response to affiliate indications of need and our confirmation via behavioral analysis.

## Proactive Organizational Change Management

Your organization will be much better off once it moves away from reacting to change, and toward proactive anticipation and management of change. This is what Level 5 of the SEI-CMM® is all about, particularly the key process areas of Technical Change Management and Process Change Management.

The COCOMO II model and parameters can help you evaluate candidate change management strategies. For example, investing in sufficient software tool acquisition and training to bring your projects' TOOL rating from nominal to high will replace a 1.0 effort multiplier by an 0.90, for a 10 percent productivity gain. Similar investments in improving process maturity, architecture and risk resolution, team cohesion, multisite

development, reuse, or any of the personnel factors can also have significant benefits that can be investigated via COCOMO II (See Figure 5). The cost, schedule, and quality drivers of COCOTS and CORADMO can be used similarly.

An integrated capability for using COCOMO II and CORADMO for evaluating the payoff of cost and schedule improvement strategies is provided by the Constructive Productivity Model (COPROMO) extension described in *Software Cost Estimation with COCOMO II*. It enables you to start from a current baseline of cost and schedule drivers from either your own organization's data or the COCOMO II database; and to express candidate cost and schedule improvement strategies in terms of achievable time-phased improvements in cost and schedule drivers. COPROMO will generate the resulting estimates and provide time histories of cost and schedule improvements for each of the candidate strategies.

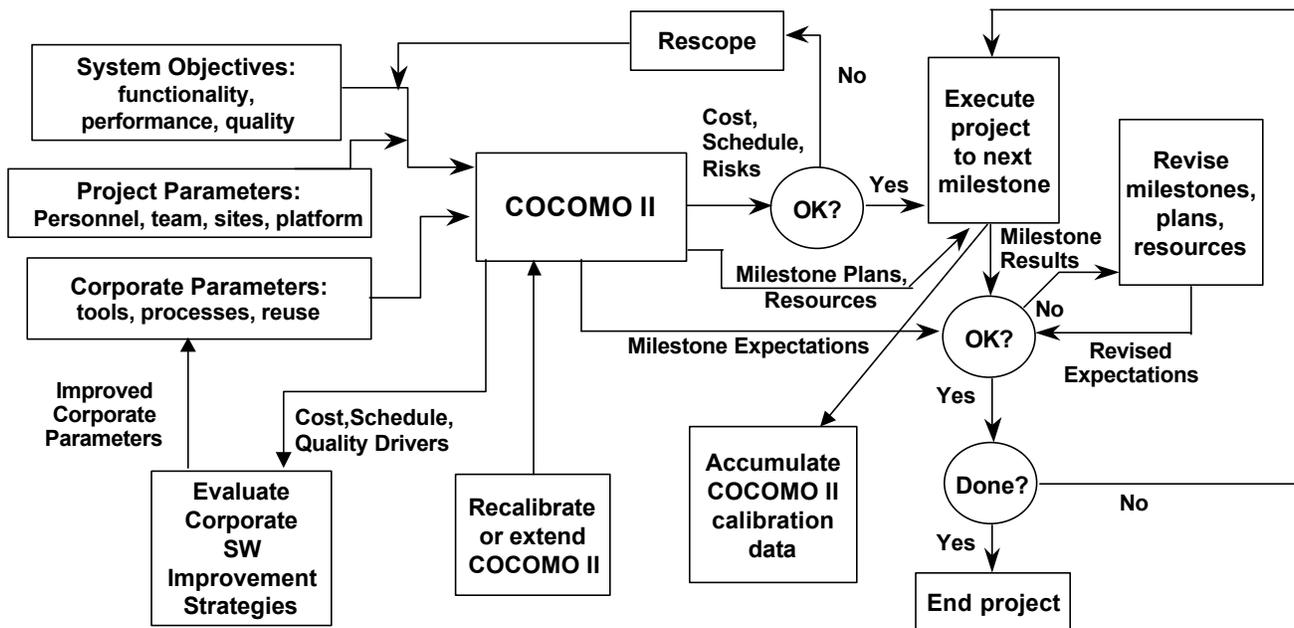Figure 4. *Using COCOMO II to Cope With Change: III*
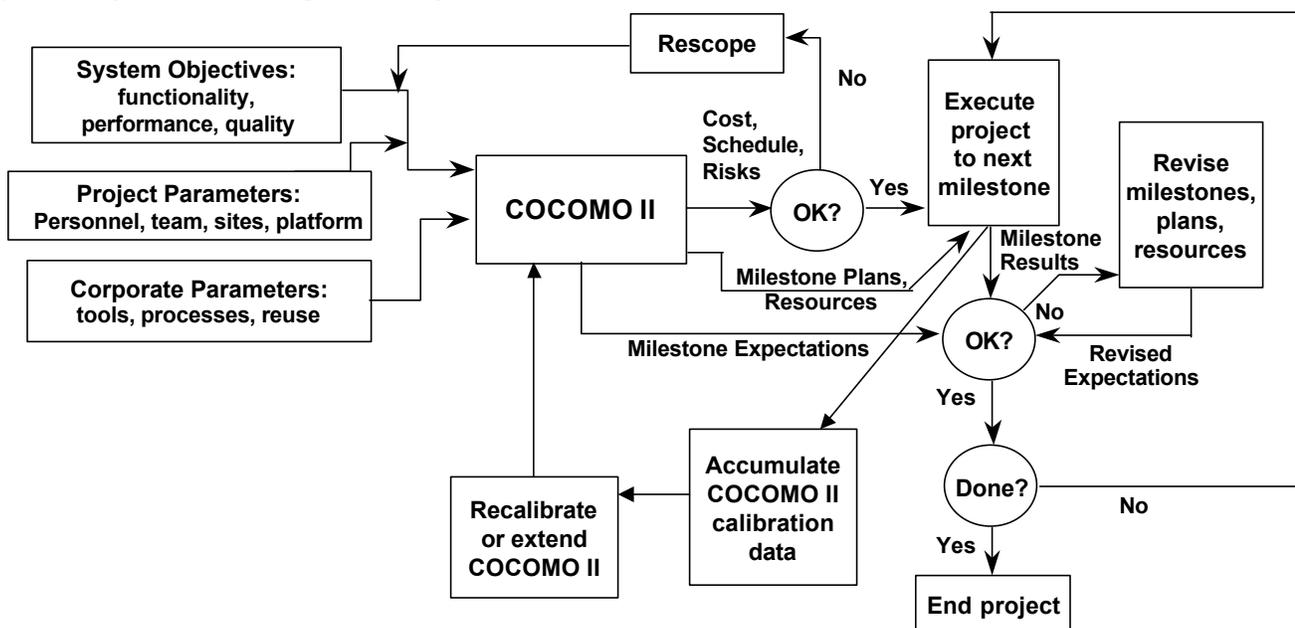
Figure 5. *Using COCOMO II to Cope With Change: IV*

Put together, the four COCOMO II feedback cycles in Figure 5 can enable your organization to determine and evolve a project-level and organization-level set of project analysis, management, and improvement strategies based on your own quantitative metrics. These strategies will enable you to determine appropriate objectives and approaches for each project, to manage projects to more successful completion, and to improve your organization's software productivity, speed, and quality by anticipating and capitalizing on change rather than being a reactive victim of change.

References

1. Boehm, B.; Abts, A.; Brown, W.; Chulani, S.; Clark, B.; Horowitz, E.; Madachy R.; Reifer, D.; and Steece, B. *Software Cost Estimation with COCOMO II,* Prentice Hall (to appear in June 2000).
2. Boehm, B.; Kellner, M. and Perry, D. (eds.), *Proceedings, ISPW 10: Process Support of Software Product Lines,* IEEE Computer Society, 1998.
3. Reifer, D. *Practical Software Reuse,* John Wiley and Sons, 1997.
4. Maier, M. Architecting Principles for Systems-of-Systems, *Systems Engineering* Vol. 1 No. 4 (1998), pp. 267-284.
5. Boehm, B. and Port, D. Escaping the Software Tar Pit: Model Clashes and How to Avoid Them, *ACM Software Engineering Notes,* Jan. 1999, pp. 36-48.
6. Boehm, B.; Egyed, A.; Port, D.; Shah, A.; Kwan, J.; and Madachy R., A Stakeholder Win-Win Approach to Software Engineering Education, *Annals of Software Engineering,* Vol. 6(1998), pp. 295-321.

About the Authors

Barry Boehm is the TRW Professor of Software Engineering and Director of the Center for Software Engineering at USC. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, and the Office of the Secretary of Defense as the Director of Defense Research & Engineering Software and Computer Technology Office. Besides COCOMO, he originated the spiral model of the software process and the stakeholder win-win approach to software management and requirements negotiation.

University of Southern California
Center for Software Engineering
Los Angeles, Calif. 90089-0781
Voice: 213-740-8163
Fax: 213-740-4927
E-mail: boehm@sunset.usc.edu

Dr. Ellis Horowitz is professor of computer science and electrical engineering at the University of Southern California.He is also Director of the Distance Education Network in the School of Engineering. He was chair-man of the Computer Science Department at USC for nine years. He is the author of 10 books and more than 100 research arti-cles on computer science subjects ranging from data structures, algorithms, and soft-ware design to computer science education.

Dr. Raymond Madachy is an adjunct assistant profes-sor in the computer science and industrial and systems engineering departments and research collaborator with the USC Center for Software Engineering. He is also the manag-er of the Software Engineering Process Group at Litton Guidance and Control Systems, which achieved SEI CMM Level 4 in December 1998. He completed his Ph.D. in Industrial and Systems Engineering at USC in 1994.

Chris Abts holds Bachelor's and Master's degrees in industrial engineering and is a research assistant/ Ph.D. Candidate at the USC Center for Software Engineering. He has worked as a lead systems engineer/analyst for Logicon in the area of Mission Planning, including development of automated mis-sion planning tools, algorithm development, mission scenario analysis, and simulation modeling. His research interests have been in software metrics and cost models, risk man-agement, and systems architecting, including the development of the COCOTS COTS modeling extension to COCOMO II.

Information on other authors of this article may be found at http://sunset.usc.edu/ Research_Group/People.html