

# Prospectus

## **COCOMO 2.0 Program**

USC Center for Software Engineering (USC-CSE)

and

UCI Irvine Research Unit in Software (IRUS)

September 1995

(version 1.5)

Executive Summary .....	1
COCOMO 2.0 Business Motivation and Program Objectives .....	2
Technical Approach: COCOMO Baseline .....	3
Technical Approach: Future Software Marketplace Model .....	4
COCOMO 2.0 Models for the Software Marketplace Sectors .....	6
Technical Approach: Model Improvements .....	6
Technical Approach: Activity Sequence .....	8
Affiliates' Program .....	9
Artifacts Currently Available to COCOMO 2.0 Affiliates .....	11
Related USC and UCI Affiliates' Programs .....	11
COCOMO 2.0 Program Principals .....	14
COCOMO 2.0 Affiliates' Membership Form .....	19

**For further information please contact**

the USC-CSE Administrator:

Karen Prouten:

phone (213) 740 - 5703

fax (213) 740 - 4927

cocomo-info@sunset.usc.edu

[http://sunset.usc.edu/Ongoing\\_Research/topics](http://sunset.usc.edu/Ongoing_Research/topics)

or the COCOMO 2.0 Principals:

Prof. Barry Boehm:

boehm@usc.edu

phone (213) 740 - 8163

fax (213) 740 - 4927

Prof. Ellis Horowitz:

horowitz@usc.edu

phone (213) 740 - 8056

fax (213) 740 - 7285

Prof. Richard Selby:

selby@ics.uci.edu

phone (714) 856 - 6326

fax (714) 856 - 4056

Prof. Chris Westland

westland@usc.edu

phone (213) 740 - 0195;

fax (213) 740 - 7313

## **Executive Summary**

A new generation of software processes and products is changing the way organizations develop software. These new approaches -- evolutionary, risk-driven, and collaborative software processes; fourth generation languages and application generators; commercial off-the-shelf (COTS) and reuse-driven software approaches; fast-track software development approaches; software process maturity initiatives -- lead to significant benefits in terms of improved software quality and reduced software cost, risk, and cycle time.

However, these new approaches have not been matched to date by complementary new models for estimating software costs and schedules. This makes it difficult for organizations to conduct effective planning, analysis, and control of projects using the new approaches.

The COCOMO 2.0 Project has been formed to address this need. The current Constructive Cost Model (COCOMO) provides a strong starting point for this effort. It is arguably the most frequently-used software cost estimation model worldwide. It is available in a number of commercial packages and hundreds of private implementations. Since its publication in 1981, it has been calibrated to many organizations' special circumstances, and extended to cover such approaches as incremental development, Ada software processes, and new classes of software engineering environments. It is also an open model with public interfaces and internals.

However, a major rethinking of COCOMO, being undertaken by this project, was judged to be necessary to represent the effects of the new generation of software approaches. This rethinking will also help ensure that the results are applicable to all major business sectors of software development.

The COCOMO 2.0 Affiliates' Program enables interested organizations to participate in COCOMO 2.0's development. The Program will provide Affiliates with evolving COCOMO 2.0 model definitions and tools for experimentation and feedback. The Program will also provide Affiliates with associated reports, tutorials, model tailoring guidelines, and participation in workshops on COCOMO 2.0 and related issues. For easier, uniform data collection, the Program will provide Affiliates with a copy of the Amadeus® software metrics tool.

To date, 26 organizations have joined the COCOMO 2.0 Affiliates' Program: 10 aerospace companies, 8 commercial companies, 3 government organizations, and 5 consortia and nonprofit organizations.

This package also includes a COCOMO 2.0 Affiliates' membership form. We hope you will choose to join the Project and add your contribution to strengthen its results.

## **COCOMO 2.0 Business Motivation and Program Objectives**

“We are becoming a software company,” is an increasingly-repeated phrase in organizations as diverse as finance, transportation, aerospace, electronics, and manufacturing firms. Competitive advantage is increasingly dependent on the development of smart, tailorable products and services, and on the ability to develop and adapt these products and services within competitors' adaptation times.

Production of this new generation of software-intensive products and services must take into account business workflows and the potential for business process reengineering to adjust workflows, tasks, and software. Tomorrow's systems need to meet tight market windows with critical-mass product functionality. This highlights the need for concurrent product and process determination, and for the ability to conduct trade-off analyses among software and system life cycle costs, cycle times, functions, performance, and qualities.

Dramatic reductions in computer hardware platform costs, and the prevalence of commodity software solutions have indirectly put downward pressure on systems development costs. This makes cost-benefit calculations even more important in selecting the correct components for construction and life cycle evolution of a system, and in convincing skeptical financial management of the business case for software investments.

These business motivations lead to the three primary COCOMO 2.0 program objectives:

- To develop a software cost, schedule and quality estimation model tuned to the life cycle practices of the 1990's and 2000's.
- To develop software cost database and tool support capabilities for continuous model improvement.
- To develop associated procedures enabling Affiliates to strengthen their process management and improvement capabilities.
- To provide a quantitative analytic framework, and set of tools and techniques for evaluating the effects of software technology improvements on software life cycle costs and schedules.

In particular, these objectives emphasize the ability to support business case analyses for future corporate software investments. In addition, the data, tools, and analytic framework will enable COCOMO 2.0 Program Affiliates to develop and calibrate related models focused on their own high-leverage decision issues.

## Technical Approach: COCOMO Baseline

Figure 1 shows a top-level black box description of COCOMO. It is an open model, in that:

- All of its relationships and algorithms are publicly available;
- All of its interfaces are public, well-defined, and parametrized, so that complementary preprocessors (function point or other size estimation models), post-processors (project planning and control tools, project dynamics models, risk analyzers), and higher level packages (project management packages, product negotiation aids), can be combined straightforwardly with COCOMO.

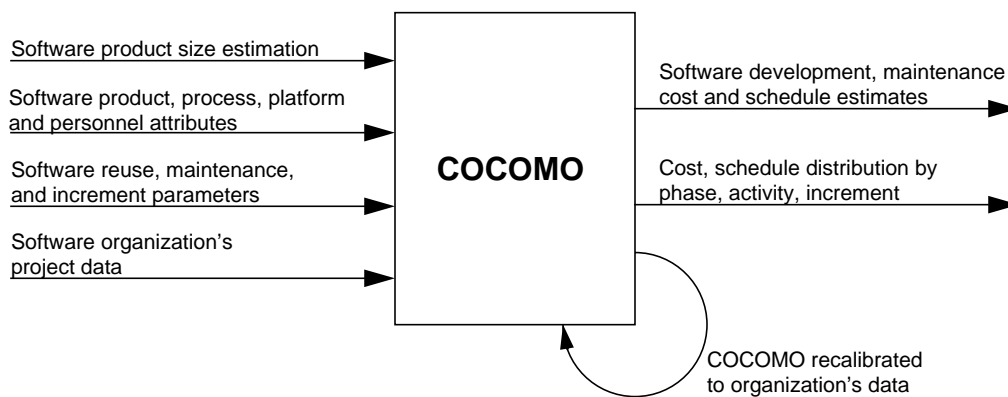


Figure 1. COCOMO Black Box Overview

The original COCOMO model was calibrated to 56 project data points in 1978, validated on 7 new project data points in 1979, and published in the book, Software Engineering Economics (by Barry Boehm, Prentice-Hall) in 1981. COCOMO is used as the standard model for software life cycle cost estimation by the U.S. Army, NATO, and numerous corporations.

Annual COCOMO users' group meetings have been held since 1985 (they are now called COCOMO/ Software Cost Estimation Forums). At these meetings, a number of corroborations, recalibrations, and improvements of the model have been reported and discussed. These meetings have also identified a candidate agenda of improvements for aligning COCOMO with 1990's - 2000's software practices. The next section discusses the top candidate model improvements in the COCOMO 2.0 technical agenda. The following section summarizes the COCOMO 2.0 program's planned sequence of activities for pursuing the technical agenda.

## Technical Approach: Future Software Marketplace Model

Figure 2 summarizes the model of the future software practices marketplace that we are using to guide the development of COCOMO 2.0. It includes a large upper “end-user programming” sector with roughly 55 million practitioners in the U.S. by the year 2005; a lower “infrastructure” sector with roughly 0.75 million practitioners; and three intermediate sectors, involving the development of applications generators and composition aids (0.6 million practitioners), the development of systems by applications composition (0.7 million), and system integration of large-scale and/or embedded software systems (0.7 million)<sup>1</sup>.

End-User Programming (55,000,000 performers in US)		
Application Generators and Composition Aids (600,000)	Application Composition (700,000)	System Integration (700,000)
Infrastructure (750,000)		

**Figure 2. Future Software Practices Marketplace Model**

End-User Programming will be driven by increasing computer literacy and competitive pressures for rapid, flexible, and user-driven information processing solutions. These trends will push the software marketplace toward having users develop most information processing applications themselves via application generators. Some example application generators are spreadsheets, extended query systems, and simple, specialized planning or inventory systems. They enable users to determine their desired information processing application via domain-familiar options, parameters, or simple rules. Every enterprise from Fortune 100 companies to small businesses and the U.S. Department of Defense will be involved in this sector.

Typical Infrastructure sector products will be in the areas of operating systems, database management systems, user interface management systems, and networking systems. Increasingly, the Infrastructure sector will address “middleware” so-

1. These figures are judgement-based extensions of the Bureau of Labor Statistics moderate-growth labor distribution scenario for the year 2005 [CSTB 1993; Silvestri and Lukaseiwicz 1991]. The 55 million End-User programming figure was obtained by applying judgement based extrapolations of the 1989 Bureau of the Census data on computer usage fractions by occupation [Kominski 1991] to generate end-user programming fractions by occupation category. These were then applied to the 2005 occupation-category populations (e.g., 10% of the 25M people in “Service Occupations”; 40% of the 17M people in “Marketing and Sales Occupations”). The 2005 total of 2.75 M software practitioners was obtained by applying a factor of 1.6 to the number of people traditionally identified as “Systems Analysts and Computer Scientists” (0.829M in 2005) and “Computer Programmers (0.882M). The expansion factor of 1.6 to cover software personnel with other job titles is based on the results of a 1983 survey on this topic [Boehm 1983]. The 2005 distribution of the 2.75 M software developers is a judgement-based extrapolation of current trends.

lutions for such generic problems as distributed processing and transaction processing. Representative firms in the Infrastructure sector are Microsoft, NeXT, Oracle, SyBase, Novell, and the major computer vendors.

In contrast to end-user programmers, who will generally know a good deal about their applications domain and relatively little about computer science, the infrastructure developers will generally know a good deal about computer science and relatively little about applications. Their product lines will have many reusable components, but the pace of technology (new processor, memory, communications, display, and multimedia technology) will require them to build many components and capabilities from scratch.

Performers in the three intermediate sectors in Figure 2 will need to know a good deal about computer science-intensive Infrastructure software and also one or more applications domains. Creating this talent pool is a major national challenge.

The Application Generators sector will create largely prepackaged capabilities for user programming. Typical firms operating in this sector are Microsoft, Lotus, Novell, Borland, and vendors of computer-aided planning, engineering, manufacturing, and financial analysis systems. Their product lines will have many reusable components, but also will require a good deal of new-capability development from scratch. Application Composition Aids will be developed both by the firms above and by software product-line investments of firms in the Application Composition sector.

The Application Composition sector deals with applications which are too diversified to be handled by prepackaged solutions, but which are sufficiently simple to be rapidly composable from interoperable components. Typical components will be graphic user interface (GUI) builders, database or object managers, middleware for distributed processing or transaction processing, hypermedia handlers, smart data finders, and domain-specific components such as financial, medical, or industrial process control packages.

Most large firms will have groups to compose such applications, but a great many specialized software firms will provide composed applications on contract. These range from large, versatile firms such as Andersen Consulting and EDS, to small firms specializing in such specialty areas as decision support or transaction processing, or in such applications domains as finance or manufacturing.

The Systems Integration sector deals with large scale, highly embedded, or unprecedented systems. Portions of these systems can be developed with Application Composition capabilities, but their demands generally require a significant amount of up-front systems engineering and custom software development. Aerospace firms operate within this sector, as do major system integration firms such as EDS and Andersen Consulting, large firms developing software-intensive products and services (telecommunications, automotive, financial, and electronic products firms), and firms developing large-scale corporate information systems or manufacturing support systems.

## COCOMO 2.0 Models for the Software Marketplace Sectors

The User Programming sector does not need a COCOMO 2.0 model. Its applications are typically developed in hours to days, so a simple activity-based estimate will generally be sufficient.

The COCOMO 2.0 model for the Application Composition sector is based on Object Points. Object Points are a count of the screens, reports and third-generation-language modules developed in the application, each weighted by a three-level (simple, medium, difficult) complexity factor [Banker et al. 1994, Kauffman and Kumar 1993]. This is commensurate with the level of information generally known about an Application Composition product during its planning stages, and the corresponding level of accuracy needed for its software cost estimates (such applications are generally developed by a small team in a few weeks to months).

The COCOMO 2.0 capability for estimation of Application Generator, System Integration, or Infrastructure developments is based on a tailorable mix of the Application Composition model (for early prototyping efforts) and two increasingly detailed estimation models for subsequent portions of the life cycle.

### Technical Approach: Model Improvements

Further COCOMO 2.0 model improvements are summarized below in terms of the current top 7 research hypotheses. The program will proceed incrementally, providing some useful new capabilities (e.g., a more accurate nonlinear software reuse model) in the near term, and more complex or refined improvements later. As the research proceeds, and Affiliate feedback is accumulated, other promising hypotheses will be identified and integrated into the current agenda.

1. Nonlinear Effects of Reuse. The Selby nonlinear software reuse effects<sup>2</sup> in Figure 3 are a good starting point for an improved software reuse cost model. In the current COCOMO reuse model, the cost of reusing software is basically a linear function of the extent that the reused software needs to be modified. However, Selby's analysis of reuse costs across nearly 3000 reused modules in the NASA Software Engineering Laboratory indicates that the reuse cost function is nonlinear in two significant ways:

- It does not go through the origin. There is generally a cost of about 5% for assessing, selecting, and assimilating the reusable component.
- Small modifications generate disproportionately large costs. This is primarily due to the cost of understanding the software to be modified.

We have formulated an initial revision of the COCOMO reuse model to accommodate these nonlinearities, which will be in the initial increment of COCOMO

---

2. Selby, R. (1988), "Empirically Analyzing Software Reuse in a Production Environment," In Software Reuse: Emerging Technology, W. Tracz (Ed.), IEEE Computer Society Press, 1988., pp. 176-189

## 2.0.

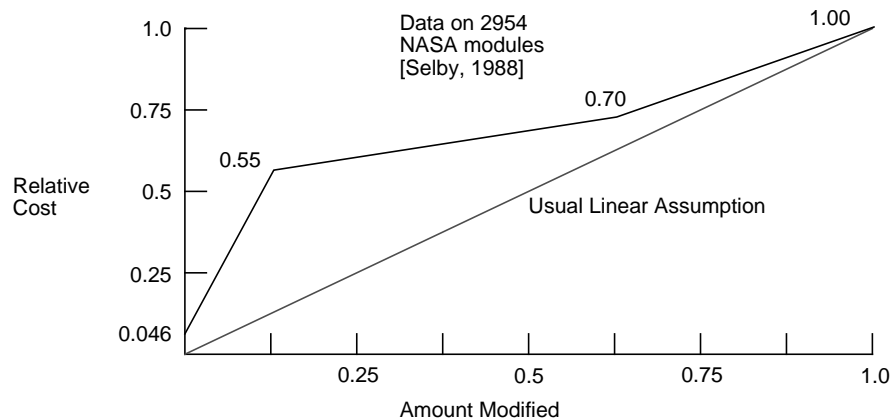


Figure 3. Nonlinear Reuse Effects

2. New phase and activity distribution models. New models are needed for re-use-driven software development and very high level languages (VHLL's). The Verner-Tate data<sup>3</sup> and model for software development with fourth generation languages is a good starting point for estimating overall VHLL development effort, overall schedule, and their distribution by phase.

3. Schedule Cost Drivers; Schedule/Quality Trade-offs. Some software cost drivers should be independently used as schedule drivers as well. The current COCOMO assumes that software cost drivers affect schedule only through their contribution to software effort and cost. However, some cost drivers probably have a more pronounced effect on schedule (e.g., applications experience, analyst capability, requirements volatility). We will investigate these candidate effects via conceptual models and exploratory data collection and analysis.

Also, as a number of Affiliates have indicated a desire to assess trade-offs between early delivery and delivered product quality in terms of error rates, we plan to address this issue via exploratory modeling and data analysis as well.

4. Software Process Maturity. The Ada COCOMO  $\Sigma$  factor<sup>4</sup> is a good starting point for modeling the effects of software process maturity. Ada COCOMO modeled the effect of certain factors (stable requirements; stable, consistent design specifications; low residual risks; and experience with advanced process models) on the economies of scale in software development. There was general confirmation via project data that improvements in these directions did correlate with re-

---

3. Verner, J., G. Tate, "Estimating Size and Effort in Fourth-Generation Development," IEEE Software, July 1988, pp. 15-22.

4. Boehm, B., and W. Royce (1989), "Ada COCOMO and the Ada Process Model," *Proceedings, Fifth COCOMO Users' Group Meeting*, Software Engineering Institute, Pittsburgh, PA, November 1989.

duced project rework, process thrashing, and diseconomies of scale. In modeling such effects, Ada COCOMO used a factor called  $\Sigma$  to modify the exponent relating size to effort in the equation:

$$Effort = A \times (Size)^{B + \Sigma} \quad EQ 1.$$

Our current hypothesis is that software process maturity (as measured by the SEI Capability Maturity Model or ISO 9000) has similar effects on diseconomies of scale, which can be modeled by a generalization of the  $\Sigma$  factor. We are collecting and analyzing data to test and refine this hypothesis.

5. Risk Assessment. COCOMO 2.0 should have risk assessment features. Examples are Monte Carlo confidence levels and Madachy's recent Expert COCOMO tool for software risk assessment based on COCOMO cost driver ratings.

6. Additional Cost Drivers. Additional software cost drivers need consideration. Based on Affiliate inputs to date, we have modeled and plan to collect data on such cost drivers as the project's environment (customer/user/developer roles, organizational/geographical distribution), documentation, and personnel continuity. We have also rethought some other cost drivers such as modern programming practices, tool capabilities, and turnaround time. Further cost drivers to investigate will result from our continuing analysis of leading current cost models, and from suggestions provided by our Affiliates.

7. Maintenance Model. Based on discussions with Affiliates, we have decided to unify the COCOMO 2.0 software reuse and maintenance cost models. Also, the COCOMO 2.0 agenda includes formulating models for other weakly-supported segments of the software life cycle: system engineering, hardware-software integration, operational test and evaluation, installation, and reengineering. Parametric models may work for some of the segments, but activity-based approaches may be more appropriate in some cases. Further data is needed to determine good models for these life-cycle segments

Besides the research agenda specific to the COCOMO 2.0 software life cycle cost-schedule estimation model, we plan to investigate and develop related cost-schedule models where Affiliate interest, good candidate models, and available data support developing useful results. A leading candidate area involves organization-wide business-case analysis of software domain engineering and product line management investments and their associated reuse payoffs. Some others involve modeling cost-benefit relations for information system business case analysis, and cost/schedule/quality models.

## **Technical Approach: Activity Sequence**

1. Cost Model Review. We have completed the initial review. It involved obtaining and creating an archive of latest information on leading software cost models (e.g., COCOMO, Ada COCOMO, PRICE S, SEER, Softcost-R, Softcost-Ada, Checkpoint, Estimacs), comparing their input factors used and outputs produced, and identifying their capabilities to address the 1990's - 2000's practices above. The

proprietors of these models have been most cooperative in furnishing us with information. We have provided them with the results of the review and the initial definition of COCOMO 2.0, in the spirit of COCOMO as an open model available to everyone in the software costing community.

2. Cost Model Hypothesis Formulation. Some hypotheses, such as revising the linear COCOMO reuse model to accommodate nonlinear software reuse effects, are already well-formulated. Others, such as preferred counting rules for object-oriented software and visual programming applications, need further exploratory work.

3. Data Definitions. A preliminary set of data definitions have been documented in a COCOMO 2.0 Model User's Manual. A data collection document has been created that sets up collection procedures.

4. Data Collection Support. We provide a confidential data collection procedure to interested Affiliates. A data collection document defines record definitions that can be used to convert Affiliate historical data to the COCOMO 2.0 format. We provide contributing Affiliates the Amadeus Metric collection tool from Amadeus Software Research, Inc.

5. Data Archiving. A data management system is used to archive the data and support analysis activities for COCOMO 2.0. The data is stored on a stand alone computer in a locked room.

6. Incremental Model Development and Refinement. Increments will be sequenced based on strength of the hypotheses being tested, availability of data, and relative utility of the results. Refinement will be based on usage feedback from the COCOMO 2.0 Affiliate organizations. The USC COCOMO tool will be evolved to support new model capabilities. Periodic updates will be provided to Affiliates via Internet File Transfer Protocol or MS DOS floppy disks. USC-CSE holds the copyright for USC COCOMO; Affiliates may create extensions and tailored versions under appropriate conditions.

7. Related Research. Additional related hypotheses will be formulated and tested based on insights accumulated on COCOMO 2.0, and relative utility of the results. Candidates include software product line management return on investment models, and related technology-impact assessment models.

## **Affiliates' Program**

Figure 4 illustrates the interaction between the COCOMO 2.0 project and its affiliates. Affiliates contribute funding support, sanitized data, and feedback on models and tools. They can also arrange special projects or visits to USC and/or UCI by their staff. The COCOMO 2.0 Project provides Affiliates with COCOMO 2.0 models and tools, the Amadeus® metrics package, tutorials, reports, related research, collaboration on special projects, and graduating students trained in COCOMO 2.0 development and use. The project also works with SEI and other organizations to use standard metric and process definitions, and maintains the CO-

COMO 2.0 data repository.

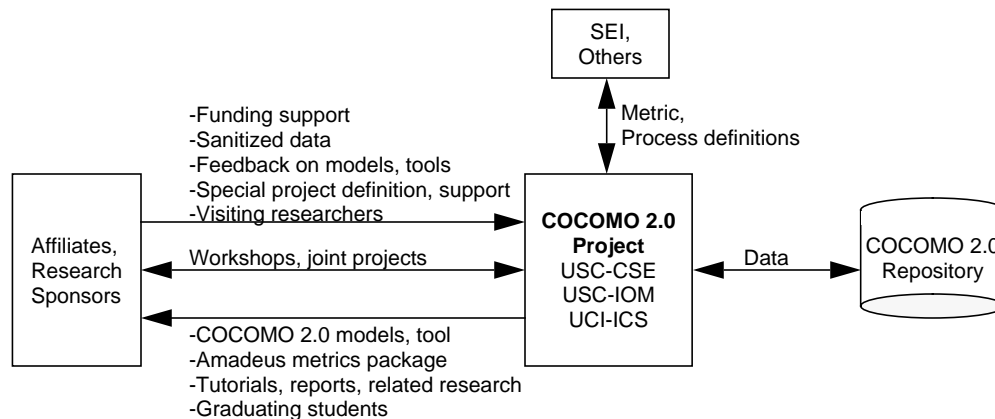


Figure 4. COCOMO 2.0 Affiliates' Program

### Affiliates' Program: Initial Activities and Contributions

The initial COCOMO 2.0 activities involving Affiliates are the following:

- Identifying one or more organizational focal points for active participation in the project.
- Reviewing the COCOMO 2.0 User's Manual and Data Collection documents.
- Participating in data collection and analysis (after Amadeus training).
- Participating in USC Center for Software Engineering/SEI-sponsored COCOMO/Software Cost Estimation Forums, COCOMO 2.0 Affiliate meetings and COCOMO 2.0 Workshops. These activities include tutorials on COCOMO 2.0 and Amadeus, and discussion of the model.

Primary Affiliates' contributions to the COCOMO 2.0 project include:

- funding (\$15,000 per year for COCOMO Affiliates),
- software cost-related data, and
- feedback on models and tools.

Additional optional Affiliate contributions can include:

- definition and support of special research projects,
- collaboration on joint projects, and
- visiting researchers at USC and/or UCI.

The general USC-CSE and UCI-IRUS Affiliate programs function similarly to the COCOMO 2.0 program in offering technology transfer and collaboration opportunities in such areas as software architectures; advanced software processes, tools, and environments; and human-computer interaction software, particularly

groupware. Large Affiliates enjoy economies of scale in joining one or both of these programs. The relative contribution levels for large organizations are:

- COCOMO 2.0 Affiliates Program: \$15,000 per year
- Either USC-CSE or UCI-IRUS Affiliates Program: \$25,000 per year
- Both USC-CSE and UCI-IRUS Affiliates Programs: \$40,000 per year

A membership form is included at the end of this document.

### **Artifacts Currently Available to COCOMO 2.0 Affiliates**

- Paper submitted to Annals of Software Engineering
- COCOMO 2.0 Model Users' Manual
- Data Collection Procedures
- Knowledge Summary: COCOMO 2.0 Focused Workshop
- Amadeus Measurement System
- COCOMO 2.0 Prototype Software
- USC COCOMO Software (based on 1981 model)

### **Related USC and UCI Affiliates' Programs**

#### **USC Center for Software Engineering Affiliates' Program**

The USC Center for Software Engineering (CSE) research focuses on the technologies necessary to achieve scalable collaborative concurrent engineering capabilities for large-scale software intensive systems. "Concurrent engineering" refers to the ability to support concurrent reasoning about and negotiation of software and systems concerns, product and process concerns, and concerns reflecting all of the system's life-cycle stakeholders. As indicated in Figure 5, there are a number of synergies among major research components involved in the CSE core concurrent engineering capability. COCOMO 2.0 benefits from the process modeling and architecture research, and provides them with a powerful tool for stakeholder negotiation and architectural reasoning.

Besides Profs. Boehm, Horowitz, and Westland, the CSE principals include leading researchers in the USC Information Science Institute (Profs. Robert Balzer, Lewis Johnson, and David Wile), the USC Business School (Prof. Walter Scacchi), and the USC Electrical Engineering Department (Prof. Alvin Despain and Eberhardt Rechtin). Prof. Scacchi is also a principal in the School of Business Administration's ATRIUM project, focused on business process redesign and innovation.

The CSE Affiliates' program currently includes twenty-six industry, government, and nonprofit organizations. They are Aerospace Corporation, AT&T Bell Laboratories, Bellcore, DISA, E-Systems, Electronic Data Systems Corporation, Hewlett-Packard Company, Hughes Aircraft Company, Institute for Defense

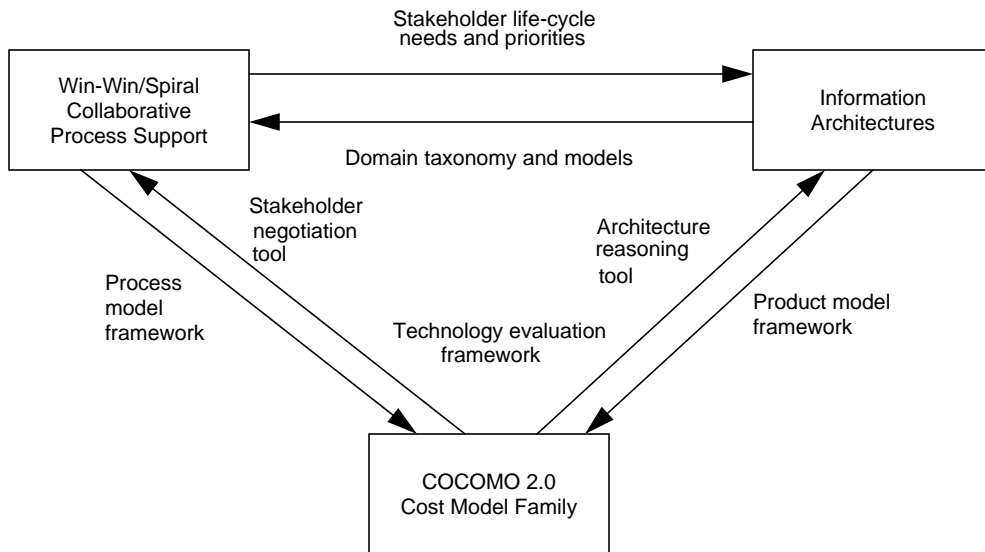


Figure 5. USC-CSE Core Concurrent Engineering Capability Overview

Analysis, Interactive Development Environments, Jet Propulsion Laboratory, Litton Data Systems, Lockheed Martin Corporation, Loral Corporation, Motorola, Northrop Grumman Corporation, Rockwell International, Rational, Inc., Science Applications International Corporation, Software Engineering Institute at Carnegie-Mellon University, Software Productivity Consortium, Texas Instruments, TRW, U.S. Air Force Rome Laboratory, U.S. Army Research Laboratory, and Xerox. Affiliates typically contribute a \$25,000 per year grant or equivalent research contract support to CSE, and participate in a program of focused workshops, executive workshops, research reviews, and special programs such as the COCOMO 2.0 program. For small organizations (less than \$100 million annual budget), the membership contribution is \$5,000 per year, but this does not include COCOMO 2.0 membership. The membership contribution for foreign organizations is \$75,000 per year.

Affiliates receive technical reports and executable versions of evolving CSE research tools for experimentation and feedback. Most of these are now available via Internet. Source code of Affiliates sponsored research efforts is copyrighted by USC-CSE and made available to Affiliates to extend. Affiliates also receive an annual visit by a selected USC-CSE principal and membership on the CSE Steering Committee.

### **Irvine Research Unit in Software**

IRUS, the Irvine Research Unit in Software, was chartered by the Chancellor of the University of California at Irvine in 1990 to promote interaction between academia and industry on current software issues, to facilitate cooperative problem solving, and to provide information sharing in the field of software engineering.

The primary goal of IRUS is to create an alliance by developing a local software community and fostering synergy between university faculty and regional software industry. IRUS provides organizational infrastructure and support to advance the state of the art and the state of the practice in computer software.

The role of IRUS in the regional software community is to:

- Provide a forum for communication and experience sharing, leading to improved coordination and a growing stimulus for joint community initiatives and action.
- Accelerate the rate of absorption by regional software practitioners of the best new software technologies developed.
- Facilitate active networking and interaction on key contemporary software issues and topics.
- Foster the rapid sharing and dissemination of the best ideas among regional industry leaders.
- Influence university research and state-of-the-art technology development.
- Expedite technology transition, evaluation, and insertion of university research into industrial practice.

IRUS has established a track record of moving the community forward through its increasingly popular activities. In particular, the Irvine Software Symposium, which features peer reviewed, high quality presentations, is well attended and includes international participation; the Southern California Software Process Improvement Network (SPIN) has been a model for the founding of SPINs across the country and around the world; the Software Engineering Tools and Technologies Network, IRUS's newest activity, fosters sharing of ideas and experiences; IRUS Roundtables have been well received and have enriched technical interchange; the Fall Research Review features current IRUS research activities, both academic and industrial; Focused Technology Symposia have increased the community's awareness on contemporary software topics; Technical Research Reviews have critically evaluated current software issues and led to cooperative problem solving. All events have featured internationally recognized speakers.

The foundation of IRUS is the software engineering faculty at the University of California at Irvine. The IRUS core faculty has an established track record for working together harmoniously and for obtaining and administering large-scale joint funding. The faculty also has established software related research projects in cooperation with the local industry and connections to software research groups elsewhere in the University of California system, across the country, and around the world.

For information regarding IRUS please contact the individual whose address appears on the following page:

Debra A. Brodbeck  
Technical Relations Director  
Irvine Research Unit in Software  
Dept. of Information and Computer Science  
University of California, Irvine  
Irvine, CA 92717-3425  
(714) 725 - 2260  
brodbeck@ics.uci.edu

*The Irvine Research Unit in Software wishes to thank its corporate sponsors:*

Sustaining Members:

Hughes Aircraft Company, McDonnell Douglas Aerospace West,  
Northrop Corporation, Rockwell International, TRW

Sponsoring Members:

Applied Technology Associates, Beckman Instruments, Computing  
Experience Corporation, FileNet Corporation, GTE, Hewlett-Packard  
Laboratories, Loral, NASA Ames Research Center, Relsys, Tandem

## **COCOMO 2.0 Program Principals**

The COCOMO 2.0 program principals are Profs. Barry Boehm and Ellis Horowitz of the USC Computer Science Department, Prof. Chris Westland of the USC Business School's Information and Operations Management Department, and Prof. Richard Selby of the UC Irvine Information and Computer Science Department. They have a strong track record of significant contributions to software cost modeling, software metrics, information economics, and practical software tool development. They will use the current USC COCOMO tool and the Amadeus Software Research, Inc. Amadeus software metrics package to support their research and model development.

**Barry Boehm, TRW Professor of Software Engineering,  
Computer Science Department, USC**

Barry Boehm received his B.A. degree from Harvard in 1957, and his M.S and Ph.D. degrees from UCLA in 1961 and 1964, all in Mathematics. Between 1989 and 1992, he served within the U.S. Department of Defense (DoD) as Director of the DARPA Information Science and Technology Office and the Software and Intelli-

gent Systems Technology Office, as Director of the DDR&E Software and Computer Technology Office, and as Director of two major DoD software initiatives: the DoD Software Technology Plan and the DDR&E Software Action Plan. He worked at TRW from 1973 to 1989, culminating as Chief Scientist of the Defense Systems Group, and at the Rand Corporation from 1959 to 1973, culminating as Head of the Information Sciences Department. He is currently Director of the USC Center for Software Engineering. His current research interests include software process modeling, software requirements engineering, software architectures, software metrics and cost models, software engineering environments, and knowledge-based software engineering. His contributions to the field include the Constructive Cost Model (COCOMO), the Spiral Model of the software process, and two advanced software engineering environments: the TRW Software Productivity System and Quantum Leap Environment. He has served on the editorial boards of several scientific journals, including the IEEE Transactions on Software Engineering, IEEE Computer, IEEE Software, ACM Computing Reviews, and Information and Software Technology. He has served as chair of the AIAA Technical Committee on Computer Systems, Chair of the IEEE Technical Committee on Software Engineering, and as a member of the Governing Board of the IEEE Computer Society. His honors and awards include Guest Lecturer of the USSR Academy of Sciences (1970), the AIAA Information Systems Award (1979), the J.D. Warnier Prize for Excellence in Information Sciences (1984), the ISPA Freiman Award for Parametric Analysis (1988), and the NSIA Grace Murray Hopper Award (1989). He is an AIAA Fellow and an IEEE Fellow.

#### Selected Publications

1. *Characteristics of Software Quality*, North Holland, with J.R. Brown, H. Kaspar, M. Lipow, G. McLeod, and M. Merritt, 1978.
2. *Software Engineering Economics*, Prentice Hall, 1981.
3. *Software Risk Management*, IEEE Computer Society Press, 1989.

#### **Ellis Horowitz, Professor, Computer Science and Electrical Engineering Departments, USC**

Ellis Horowitz received his M.S. and Ph. D. degrees in Computer Science from the University of Wisconsin at Madison. He served his industrial apprenticeship working for IBM in Poughkeepsie, New York. He was also associated with IBM's World Trade Corporation in Paris. Dr. Horowitz's initial research interests were in the field of mathematical symbol manipulation. At Wisconsin, he worked on SAC 1, a system for the manipulation of mathematical expressions. Afterwards Dr. Horowitz became interested in the fields of data structures and algorithms. He wrote many papers developing and analyzing exact and approximate algorithms for solving a wide variety of common data processing problems. In 1975, Dr. Horowitz edited the book *Practical Strategies for Developing Large Software Systems*, Addison Wesley, which was based upon his interaction with computer scientists in industry. Dr. Horowitz is also the co-author of two widely used texts, *Fundamentals of Data Structures* and *Fundamentals of Computer Algorithms* and is the author of

*Fundamentals of Programming Languages* and *Programming Languages: A Grand Tour*, all published by W.H. Freeman. He has been a Visiting Associate Professor at both the Technion and MIT. His recent interests include SScriptWriter, a programming environment designed expressly for the creation of educational software and Computer Aided Software Engineering (CASE) tools. He is currently an IBM Scholar.

#### Selected Publications

1. *Object-Oriented Databases with Applications to CASE, Networks and VLSI*, Prentice-Hall, Englewood Cliffs New Jersey, 1990
2. "Cbase1.0: A CAD Database for VLSI Circuits Using Object Oriented Technology," *Proc. of the IC-CAD Conference*, (with M.A. Breuer, W. Cheng, R. Gupta, I. Hardonag, and S.Y.Lin), 1989.
3. "Database Support for Software Project Management," *Proc. ACM Conf. on Practical Software Development Environments*, Boston, Ma., November, 1988, pp. (with L. Liu)

#### **Richard W. Selby, Professor, Information and Computer Science, UCI**

Richard W. Selby received the B.A. degree in mathematics from Saint Olaf College, Northfield, MN, in 1981 and the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, in 1983 and 1985, respectively. Since 1991 he has been an Associate Professor of Information and Computer Science at the University of California, Irvine. From 1985--91, he was an Assistant Professor at the University of California, Irvine. His research interests include empirically based analysis techniques for large-scale software, software metrics, and software environments. He has written over 50 refereed publications and consulted for many government and commercial organizations. Selby and Prof. Kaji Torii co-edited the IEEE Transactions on Software Engineering special issue on software measurement that appeared in November 1992. He also serves on the editorial boards for the Journal of Software Testing, Verification, and Reliability and the Journal of Software Quality. In 1992 he founded Amadeus Software Research, Inc. to develop and market software CASE tools and services including a commercial version of the Amadeus Measurement System. Dr. Selby is a principal of the Arcadia Consortium and the Irvine Research Unit in Software (IRUS) and a member of the NASA Software Engineering Laboratory (SEL). He is a member of the Association for Computing Machinery, the IEEE, and the IEEE Computer Society.

#### Selected Publications

1. Richard W. Selby and Adam A. Porter, "Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis," *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 12, December 1988, pp. 1743-1757.
2. Richard W. Selby, "Empirically Based Analysis of Failures in Software Sys-

tems," *IEEE Transactions on Reliability*, Vol. 39, No. 4, October 1990, pp. 444-454.

3. Richard W. Selby, Adam A. Porter, Doug C. Schmidt and Jim Berney, "Metric-Driven Analysis and Feedback Systems for Enabling Empirically Guided Software Development," *Proceedings of the Thirteenth International Conference on Software Engineering*, Austin, TX, May 13-16, 1991.

**J. Christopher Westland, Professor,  
Information & Operations Management, USC**

Chris Westland received a B.A. in Mathematics in 1971 and an MBA in Accounting in 1973 from Indiana University. Following many years in industry as a Certified Public Accountant for Touche Ross in Chicago, and later as a computer security specialist for Rockwell International, he returned for a Ph.D. in Computers & Information Systems at the University of Michigan in 1987. Dr. Westland's research focuses on the economics of information technology. He has argued, in papers on a wide range of topics, that market forces -- more than available or even appropriate technology -- drive the creation and promotion of successful information technology. He is also studying the comparative performance of Multinomial-Dirichlet Bayesian revision with fuzzy sets and expert systems; and modifications of optical transfer functions to predict systems reliability. Dr. Westland has been awarded an Andersen Foundation Fellowship, a FLAS Fellowship by the U.S. Department of Education, several Paton Fellowships, a Dykstra Fellowship for Teaching Excellence, and a FRIF Grant to study politics and conflict resolution in information systems resource sharing. He is a member of the AICPA, TIMS, the Western Economic Association and Beta Gamma Sigma.

Selected Publications

1. Congestion and Network Externalities in the Short Run Pricing of Information Systems Services, *Management Science*, v. 38(6) July 1992.

2. The Marginal Analysis of Investments in Information Technology, in *Strategic and Economic Impacts of Information Technology Investment: Perspectives on Organizational Growth and Competitive Advantage*, Mahmood, Banker and Kauffman (eds.), Middletown, PA: Idea Group Publishing, September 1992.

3. Scaling Up Output Volumes Predicted by Information Systems Prototypes, *ACM/TODS*, v. 15(3), 341-358, 1990.



## COCOMO 2.0 Affiliates' Membership Form

Organization Name: \_\_\_\_\_

COCOMO 2.0 Focal Point: \_\_\_\_\_

Title and Address: \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Phone Number: \_\_\_\_\_ Fax Number: \_\_\_\_\_

E-mail Address: \_\_\_\_\_

\_\_\_\_\_ USC-CSE or UCI-IRUS Sustaining Affiliate (no charge)

\_\_\_\_\_ New COCOMO 2.0 Affiliate (enclose tax-deductible contribution check for \$15, 000., made out to the USC Center for Software Engineering).

\_\_\_\_\_  
(Signature)

\_\_\_\_\_  
(Title and Date)

Please return form and check (if applicable) to:  
**USC Center for Software Engineering**  
**Computer Science Department**  
**University of Southern California**  
**Los Angeles, CA 90089-0781**