

## Building Parametric Models

Barry Boehm, USC

[boehm@sunset.usc.edu](mailto:boehm@sunset.usc.edu)

IASESE 2003  
September 29, 2003

9/29/03

USC-CSE

1

## Outline

- ➔ • Range of software engineering parametric models and forms
- Goals: Model success criteria
- 8-step model development process
  - Examples from COCOMO family of models
- Conclusions

9/29/03

USC-CSE

2

## Range of SE Parametric Models

- Outcome =  $f$  (Outcome-driver parameters)
- Most frequent outcome families
  - Throughput, response time; workload
  - Reliability, defect density; usage
  - Project cost, schedule; sizing
  - Other costs: facilities, equipment, services, licenses, installation, training
  - Benefits: sales, profits, operational savings
  - Return on investment = (Benefits-Costs)/Costs

9/29/03

USC-CSE

3

## Parametric Model Forms

- Analogy: Outcome =  $f$ (previous outcome, differences)
  - Example: yesterday's weather
- Unit Cost: Outcome =  $f$ (unit costs, unit quantities)
  - Example: computing equipment
- Activity-Based: Outcome =  $f$ (activity levels, durations)
  - Examples: operational cost savings, training costs
- Relationship-Based: Outcome =  $f$ (parametric relationships)
  - Examples: queuing models, size & productivity cost models

9/29/03

USC-CSE

4

## Goals: Model Success Criteria

- Scope: Covers desired range of situations?
- Granularity: Level of detail sufficient for needs?
- Accuracy: Estimates close to actuals?
- Objectivity: Inputs repeatable across estimators?
- Calibratability: Sufficient calibration data available?
- Constructiveness: Helps to understand job to be done?
- Ease of use: Parameters easy to understand, specify?
- Prospectiveness: Parameters values knowable early?
- Parsimony: Avoids unnecessary parameters, features?
- Stability: Small input changes mean small output changes?
- Interoperability: Easy to compare with related models?

9/29/03

USC-CSE

5

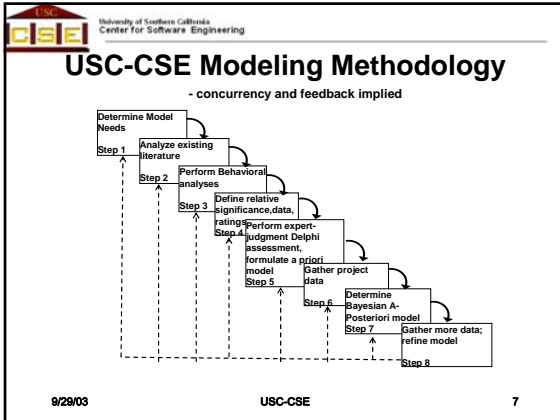
## Outline

- Range of software engineering parametric models and forms
- Goals: Model success criteria
- ➔ • 8-step model development process
  - Example from COCOMO family of models
- Conclusions

9/29/03

USC-CSE

6



USC-CSE

## Step 1: Determine Model Needs

- Similar to software requirements determination
  - Identify success-critical stakeholders
    - Decision-makers, users, data providers
  - Identify their model needs (win conditions)
  - Identify their ability to provide inputs, calibration data
  - Negotiate best achievable (win-win) model capabilities
- Prioritize capabilities for incremental development
- Use Model Success Criteria as checklist

9/29/03 USC-CSE 8

USC-CSE

## Major Decision Situations Helped by COCOMO II

- Software investment decisions
  - When to develop, reuse, or purchase
  - What legacy software to modify or phase out
- Setting project budgets and schedules
- Negotiating cost/schedule/performance tradeoffs
- Making software risk management decisions
- Making software improvement decisions
  - Reuse, tools, process maturity, outsourcing

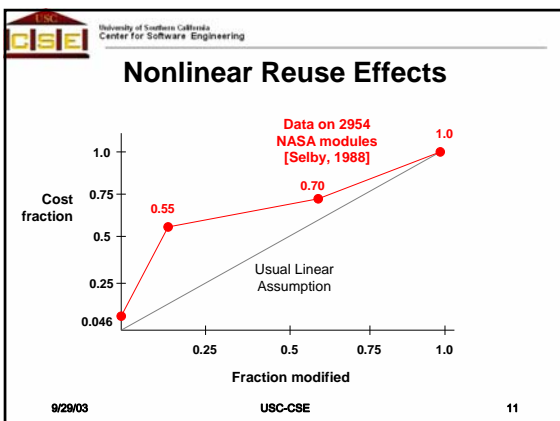
9/29/03 USC-CSE 9

USC-CSE

## Step 2: Analyze Existing Literature

- Understand underlying phenomenology
  - Sources of cost, defects, etc.
- Identify promising or unsuccessful model forms
  - Linear, discontinuous software cost models
  - Model forms may vary by source of cost, defects, etc.
  - Invalid assumptions (queuing models)
- Identify most promising outcome-driver parameters

9/29/03 USC-CSE 10




USC-CSE

## Reuse Cost Increment for Software Understanding

	Very Low	Low	Nom	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured; some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data/control structures.
Application Clarity	No match between program and application world views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world views.
Self-Descriptiveness	Obscure code; documentation missing, obscure or obsolete.	Some code commentary and headers; some useful documentation.	Moderate level of code commentary, headers, documentation.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptive code; documentation up-to-date, well-organized, with design rationale.
SU Increment to ESLOC	50	40	30	20	10

9/29/03 USC-CSE 12



University of Southern California  
Center for Software Engineering

### Step 3: Perform Behavioral Analysis


- Behavior Differences: Required Reliability Levels

Rating	Rqts and Product Design	Integration and Test
Very Low	<ul style="list-style-type: none"> <li>• Little detail</li> <li>• Many TBDs</li> <li>• Little Verification</li> <li>• Minimal QA, CM, draft user manual, test plans</li> <li>• Minimal PDR</li> </ul>	<ul style="list-style-type: none"> <li>• No test procedures</li> <li>• Many requirements untested</li> <li>• Minimal QA, CM</li> <li>• Minimal stress, off-nominal tests</li> <li>• Minimal as-built documentation</li> </ul>
Very High	<ul style="list-style-type: none"> <li>• Detailed verification, QA, CM, standards, PDR, documentation</li> <li>• IV&amp;V interface</li> <li>• Very detailed test plans, procedures</li> </ul>	<ul style="list-style-type: none"> <li>• Very detailed test procedures, QA, CM, standards, documentation</li> <li>• Very extensive stress, off-nominal tests</li> <li>• IV&amp;V interface</li> </ul>

9/29/03

USC-CSE

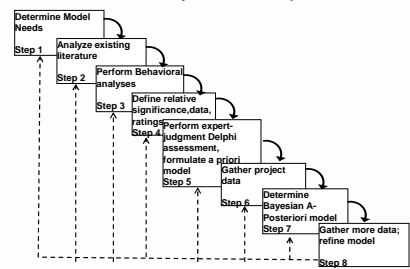
13



University of Southern California  
Center for Software Engineering

### USC-CSE Modeling Methodology


- concurrency and feedback implied



9/29/03

USC-CSE

14



University of Southern California  
Center for Software Engineering

### Step 4: Relative Significance: COSYSMO

Rate each factor H, M, or L depending on its relatively high, medium, or low influence on system engineering effort. Use an equal number of H's, M's, and L's.

**N=6 Application Factors**

- 3.0 **H** Requirements understanding
- 2.5 **M** - **H** Architecture understanding
- 2.3 **L** - **H** Level of service rqts. criticality, difficulty
- 1.5 **L** - **M** Legacy transition complexity
- 1.7 **L** - **M** COTS assessment complexity
- 1.7 **L** - **H** Platform difficulty
- 1.5 **L** - **M** Required business process reengineering
- TBD :Ops. concept understanding (N=H)
- TBD


**Team Factors**

- 1.5 **L** - **M** Number and diversity of stakeholder communities
- 2.7 **M** - **H** Stakeholder team cohesion
- 2.7 **M** - **H** Personnel capability/continuity
- 3.0 **H** Personnel experience
- 2.0 **L** - **H** Process maturity
- 1.5 **L** - **M** Multisite coordination
- 2.0 **L** - **H** Degree of system engineering ceremony
- 1.3 **L** - **M** Tool support
- TBD
- TBD

9/29/03

USC-CSE

15



University of Southern California  
Center for Software Engineering


### Step 4 COCOMO II Result: New Scaling Exponent Approach

- Nominal person-months =  $A \times (\text{size})^B$
- $B = 0.91 + 0.01 \sum (\text{exponent driver ratings})$ 
  - B ranges from 0.91 to 1.23
  - 5 drivers; 6 rating levels each
- Exponent drivers:
  - Precedentedness
  - Development flexibility
  - Architecture/ risk resolution
  - Team cohesion
  - Process maturity (derived from SEI CMM)

9/29/03

USC-CSE

16



University of Southern California  
Center for Software Engineering

### Step 4: Define Relations, Data, Rating Scales

$$PM_{estimated} = 3.67 \times (\text{Size})^{(SF)} \times \left\{ \prod_i EM_i \right\}$$


$$SF = 0.91 + 0.01 \times \sum w_i$$

Scale Factors ( $w_i$ )	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	weighted sum of 18 KPA achievement levels					

9/29/03

USC-CSE

17



University of Southern California  
Center for Software Engineering

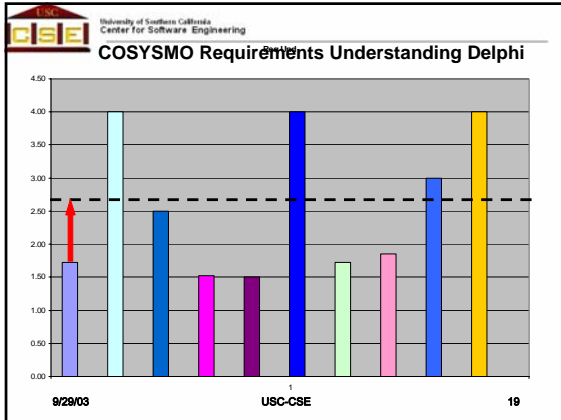
### Step 5: Initial Delphi Assessment

- Data definitions and rating scales established for significant parameters
- Convene experts, use wideband Delphi process
  - Individuals estimate each parameter's outcome-influence value
    - E.g. ratio of highest to lowest effort multiplier
  - Summarize results; group discussion of differences
    - Usually draws out significant experience
  - Individuals re-estimate outcome-influence values
  - Can do more rounds, but two generally enough
- Produces mean, standard deviation of outcome-influence values
- Often uncovers overlaps, changes in outcome drivers

9/29/03

USC-CSE

18



University of Southern California  
Center for Software Engineering

### Size Drivers vs. Effort Multipliers

- **Size Drivers: Additive, Incremental**
  - Impact of adding a new item inversely proportional to current size
    - 10 -> 11 rqls = 10% increase
    - 100 -> 101 rqls = 1% increase
- **Effort Multipliers: Multiplicative, system-wide**
  - Impact of adding a new item independent of current size
    - 10 rqls + high security = 40% increase
    - 100 rqls + high security = 40% increase

9/29/03 USC-CSE 20

University of Southern California  
Center for Software Engineering

### Step 6: Gather, Analyze Project Data

- **Best to pilot data collection with early adopters**
  - Identifies data definition ambiguities
  - Identifies data availability problems
  - Identifies need for data conditioning
- **Best to collect initial data via interviews**
  - Avoids misinterpretations
    - Endpoint milestones; activities included/excluded; size definitions
  - Uncovers hidden assumptions
    - Schedule vs. cost minimization; overtime effort reported

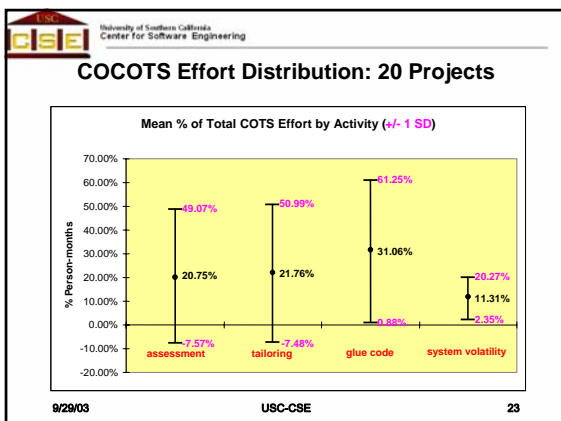
9/29/03 USC-CSE 21

University of Southern California  
Center for Software Engineering

### Initial Data Analysis May Require Model Revision

- **Initial COCOTS model adapted from COCOMO II, with different parameters**
  - $\text{Effort} = A * (\text{Size})^B * \prod (\text{Effort Multipliers})$
- **Amount of COTS integration glue code used for Size**
- **Data analysis showed some projects with no glue code, much effort**
  - Effort devoted to COTS assessment, tailoring

9/29/03 USC-CSE 22



University of Southern California  
Center for Software Engineering

### Revised COCOTS Model

- **COCOMO-like model for glue code effort**
- **Unit cost approach for COTS assessment effort**
  - Number of COTS products to assess
  - Number of attributes to assess, weighted by complexity
- **Activity-based approach for COTS tailoring effort**
  - COTS parameters setting, script writing, reports layout, GUI tailoring, protocol definitions

9/29/03 USC-CSE 24

## New Glue Code Submodel Results

- New calibration results
  - Excluding projects with very large, very small amounts of glue code
    - [0.5 - 100 KLOC]: Pred (.30) = 9/17 = 53%
    - [2 - 100 KLOC]: Pred (.30) = 8/13 = 62%
  - Previous calibration results:
    - [0.1 - 390 KLOC]: Pred (.30) = 4/13 = 31%
- Pred(.30) = percent of projects with estimates within 30% of actuals

9/29/03

USC-CSE

25

## Step 7: Bayesian Calibration

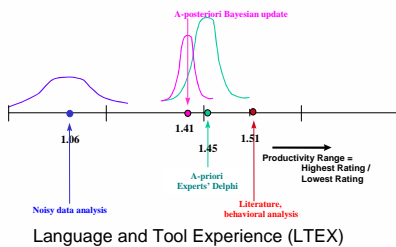
- Multiple regression analysis of project data points (model inputs, actual outputs) produces outcome-influence values
  - Mean, variance, statistical significance
- For COCOMO II, 161 data points produced mostly statistically significant parameters values
  - Productivity ranges of cost drivers
  - One with wrong sign, low significance (RUSE)
- Bayesian approach favors experts when they agree, data where results are significant
  - Result: RUSE factor with correct sign

9/29/03

USC-CSE

26

## Results of Bayesian Update: Using Prior and Sampling Information

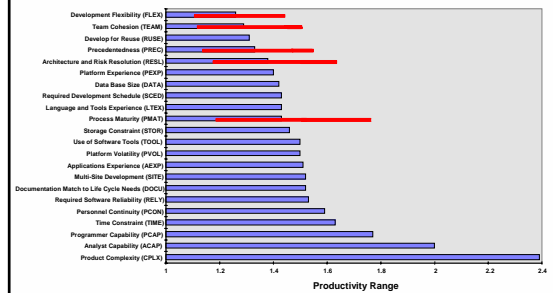


9/29/03

USC-CSE

27

## COCOMO II. 2000 Productivity Ranges



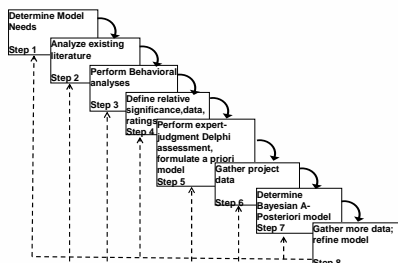
9/29/03

USC-CSE

28

## USC-CSE Modeling Methodology

- concurrency and feedback implied

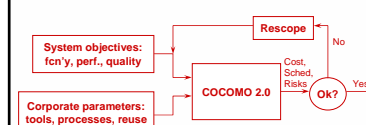


9/29/03

USC-CSE

29

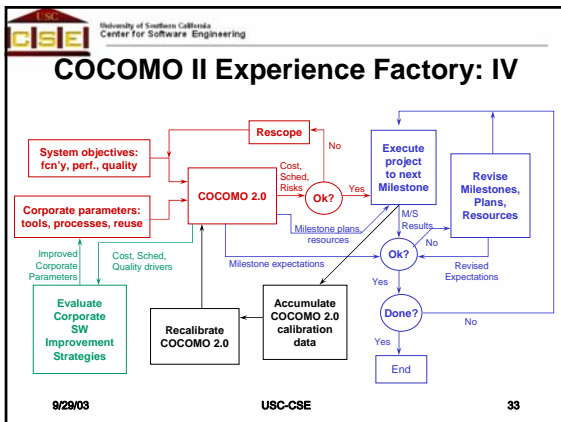
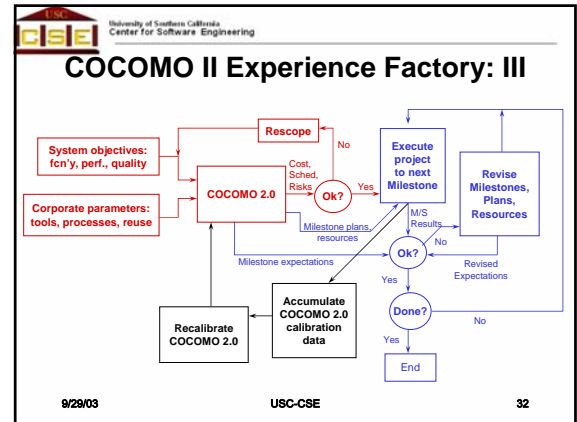
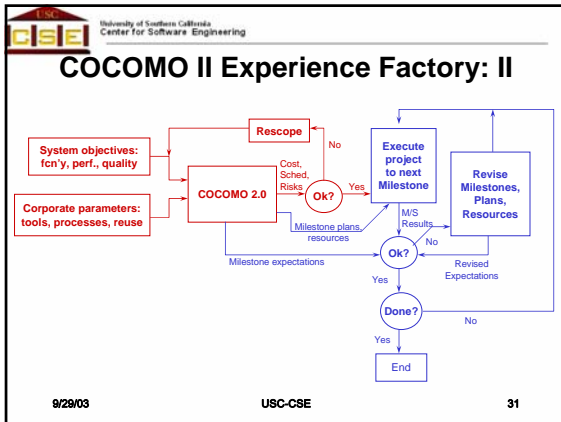
## COCOMO II Experience Factory: I



9/29/03

USC-CSE

30



University of Southern California  
Center for Software Engineering

### Step 8 Example: Software Understanding Increment Too Large

- Needed to add a Programmer Unfamiliarity factor

	Very Low	Low	Nom	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured, some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data/control structures.
Application Clarity	No match between program and application world views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world views.
Self-Descriptiveness	Obscure code; documentation missing, obscure or obsolete.	Some code commentary and headers; some useful documentation.	Moderate level of code commentary, headers, documentation.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptive code; documentation up-to-date, well-organized, with design rationale.
SU Increment to ESLOC	50	40	30	20	10

9/29/03 USC-CSE 34

University of Southern California  
Center for Software Engineering

## Conclusions

- All models are wrong; some are more useful than others.
  - Albert Einstein? George Box?
- Critical success factors for utility
  - Evidence of model demand, user willingness to support model definition, provide calibration data
  - Model focused on supporting major decision situations
  - Good match of estimation relationships to underlying phenomenology
  - Clear definition of inputs, outputs, assumptions
  - Careful conditioning of calibration data
  - Flexibility in adapting model to explain mismatches, outliers
  - Good balance of Model Success Criteria

9/29/03 USC-CSE 35

University of Southern California  
Center for Software Engineering

## Goals: Model Success Criteria

- Scope: Covers desired range of situations?
- Granularity: Level of detail sufficient for needs?
- Accuracy: Estimates close to actuals?
- Objectivity: Inputs repeatable across estimators?
- Calibratability: Sufficient calibration data available?
- Constructiveness: Helps to understand job to be done?
- Ease of use: Parameters easy to understand, specify?
- Prospectiveness: Parameters values knowable early?
- Parsimony: Avoids unnecessary parameters, features?
- Stability: Small input changes mean small output changes?
- Interoperability: Easy to compare with related models?

9/29/03 USC-CSE 36