Building Parametric Models
Barry Boehm, USC
boehm@sunset.usc.edu
IASESE 2003
September 29, 2003

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

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

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

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?
• Contructiveness: 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?
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

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

Nonlinear Reuse Effects

<table>
<thead>
<tr>
<th>Fraction modified</th>
<th>Cost fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.75</td>
</tr>
<tr>
<td>0.75</td>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Data on 2954 NASA modules [Selby, 1988]

Reuse Cost Increment for Software Understanding

<table>
<thead>
<tr>
<th>SU Increment to ESLOC</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

SU Increment to ESLOC

- Very Low: Obscure code; documentation missing, obsolete.
- Low: Some code commentary and headers; some useful documentation.
- Moderately Low: Good code commentary and headers; useful documentation; some weak areas.
- Reasonably Well-structured: Strong modularity, information hiding in data/control structures.
- High: Clear match between program and application world views.
Step 3: Perform Behavioral Analysis

- Behavior Differences: Required Reliability Levels

<table>
<thead>
<tr>
<th>Rating</th>
<th>High and Product Design Integration and Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>- Little detail</td>
</tr>
<tr>
<td></td>
<td>- Many TBDs</td>
</tr>
<tr>
<td></td>
<td>- Little Verification</td>
</tr>
<tr>
<td></td>
<td>- Minimal QA, CM, draft user plan</td>
</tr>
<tr>
<td></td>
<td>- Wasteful PM</td>
</tr>
<tr>
<td>Very High</td>
<td>- Skip detail</td>
</tr>
<tr>
<td></td>
<td>- No text procedures</td>
</tr>
<tr>
<td></td>
<td>- Many requirements oriented</td>
</tr>
<tr>
<td></td>
<td>- Minimal QA, CM, employment</td>
</tr>
<tr>
<td></td>
<td>- Wasteful stress, all non-trivial tests</td>
</tr>
<tr>
<td></td>
<td>- Wasteful no built documentation</td>
</tr>
</tbody>
</table>

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.

Step 4 COCOMO II Result:
- New Scaling Exponent Approach
  - Nominal person-months = A*(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:
    - Precedentness
    - Development flexibility
    - Architecture/ risk resolution
    - Team cohesion
    - Process maturity (derived from SEI CMM)

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
Size Drivers vs. Effort Multipliers

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

---

**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

---

**Initial Data Analysis May Require Model Revision**

- Initial COCOTS model adapted from COCOMO II, with different parameters
  - Effort = $A \times (\text{Size})^B \times \prod \text{(Effort Multipliers)}$
- 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

---

**COCOTS Effort Distribution: 20 Projects**

<table>
<thead>
<tr>
<th>Percent of Total COTS Effort by Activity (+/- 1 SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment</td>
</tr>
<tr>
<td>3.67%</td>
</tr>
</tbody>
</table>

---

**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
New Glue Code Submodel Results

- New calibration results
  - Excluding projects with very large, very small amounts of glue code
    - [0.5 - 100 KLOC]: Pred(0.30) = 9/17 = 53%
    - [2 - 100 KLOC]: Pred(0.30) = 8/13 = 62%
- Previous calibration results:
  - [0.1 - 390 KLOC]: Pred(0.30) = 4/13 = 31%

Pred(0.30) = percent of projects with estimates within 30% of actuals

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

Results of Bayesian Update: Using Prior and Sampling Information

Language and Tool Experience (LTEX)

COCOMO II. 2000 Productivity Ranges

USC-CSE Modeling Methodology

COCOMO II Experience Factory: I
Critical success factors for utility tools, processes, reuse

All models are wrong; some are more useful than others.

Albert Einstein? George Box?

System objectives: fcn’y, perf., quality

Corporate parameters:

Objectivity: Inputs repeatable across estimators?

Accuracy: Estimates close to actuals?

Calibratability: Sufficient calibration data available?

Rescambility: Helps to understand job to be done?

Ease of use: Parameters easy to understand, specify?

Prospectiveness: Parameters values knowable early?

Stability: Small input changes mean small output changes?

Interoperability: Easy to compare with related models?

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?

Contructiveness: 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?