Experience with Empirical Studies in Industry: Building Parametric Models

Barry Boehm, USC
boehm@usc.edu

CESI 2013
May 20, 2013
Outline

Types of empirical studies with Industry
  - Types, benefits, challenges
  - Comparative methods, emerging technologies, parametric modeling

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

• Conclusions
Types of Empirical Studies

• Comparative Methods: Inspection, Testing, Pair Programming
  – Benefits: Cost-effectiveness, Sweet spot insights
  – Challenges: Representative projects, personnel, environment

• Emerging Technologies: Agile, Model-Driven, Value-Based
  – Benefits: Maturity, Cost-effectiveness, Sweet spot insights
  – Challenges: Baselining, learning curve, subject skills

• Parametric Modeling: Cost, Schedule, Quality Estimation
  – Benefits: Budget realism, Progress monitoring, Productivity, quality improvement areas
  – Challenges: Community representativeness, Proprietary data, data consistency
Value-Based Testing: Qi Li at Galorath, Inc.

Business value of tests completed

- H-t1: the value-based prioritization does not increase APBIE
  - reject H-t1
- Value-based prioritization can improve the cost-effectiveness of testing

<table>
<thead>
<tr>
<th></th>
<th>APBIE-1</th>
<th>APBIE-2</th>
<th>APBIE-3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>70.99%</td>
<td>10.08%</td>
<td>32.10%</td>
</tr>
</tbody>
</table>

Stop Testing
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
Legend:
- Model has been calibrated with historical project data and expert (Delphi) data
- Model is derived from COCOMO II
- Model has been calibrated with expert (Delphi) data

Dates indicate the time that the first paper was published for the model.
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?
- 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?
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
USC-CSE Modeling Methodology

- concurrency and feedback implied

Step 1: Determine Model Needs

Step 2: Analyze existing literature

Step 3: Perform Behavioral analyses

Step 4: Define relative significance, data, ratings

Step 5: Perform expert-judgment Delphi assessment, formulate a priori model

Step 6: Gather project data

Step 7: Determine Bayesian A-Posteriori model

Step 8: Gather more data; refine model
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
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
Step 2: Analyze Existing Literature

• Understand underlying phenomenology
  – Sources of cost, defects, etc.

• Identify promising or unsuccessful model forms using Model Success Criteria
  – Narrow scope, inadequate detail
  – Linear, discontinuous software cost models
  – Model forms may vary by source of cost, defects, etc.
  – Invalid assumptions (queueing models)

• Identify most promising outcome-driver parameters
Nonlinear Reuse Effects

Data on 2954 NASA modules [Selby, 1988]

Usual Linear Assumption

Cost fraction

Fraction modified
# Reuse Cost Increment for Software Understanding

<table>
<thead>
<tr>
<th>Structure</th>
<th>Very Low</th>
<th>Low</th>
<th>Nom</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Clarity</td>
<td>No match between program and application world views.</td>
<td>Some correlation between program and application.</td>
<td>Moderate correlation between program and application.</td>
<td>Good correlation between program and application.</td>
<td>Clear match between program and application world views.</td>
</tr>
<tr>
<td>Self-Descriptiveness</td>
<td>Obscure code; documentation missing, obscure or obsolete.</td>
<td>Some code commentary and headers; some useful documentation.</td>
<td>Moderate level of code commentary, headers, documentation.</td>
<td>Good code commentary and headers; useful documentation; some weak areas.</td>
<td>Self-descriptive code; documentation up-to-date, well-organized, with design rationale.</td>
</tr>
</tbody>
</table>

| SU Increment to ESLOC | 50 | 40 | 30 | 20 | 10 |
Step 3: Perform Behavioral Analysis

• **Behavior Differences: Required Reliability Levels**

<table>
<thead>
<tr>
<th>Rating</th>
<th>Rqts and Product Design</th>
<th>Integration and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>• Little detail&lt;br&gt;• Many TBDs&lt;br&gt;• Little Verification&lt;br&gt;• Minimal QA, CM, draft user manual, test plans&lt;br&gt;• Minimal PDR</td>
<td>• No test procedures&lt;br&gt;• Many requirements untested&lt;br&gt;• Minimal QA, CM&lt;br&gt;• Minimal stress, off-nominal tests&lt;br&gt;• Minimal as-built documentation</td>
</tr>
<tr>
<td>Very High</td>
<td>• Detailed verification, QA, CM, standards, PDR, documentation&lt;br&gt;• IV&amp;V interface&lt;br&gt;• Very detailed test plans, procedures</td>
<td>• Very detailed test procedures, QA, CM, standards, documentation&lt;br&gt;• Very extensive stress, off-nominal tests&lt;br&gt;• IV&amp;V interface</td>
</tr>
</tbody>
</table>
USC-CSE Modeling Methodology

- concurrency and feedback implied

Step 1: Determine Model Needs

Step 2: Analyze existing literature

Step 3: Perform Behavioral analyses

Step 4: Define relative significance, data, ratings

Step 5: Perform expert-judgment Delphi assessment, formulate a priori model

Step 6: Gather project data

Step 7: Determine Bayesian A-Posteriori model

Step 8: Gather more data; refine model
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
1.2  L – M  Database size
____  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
Step 4: Define Relations, Data, Rating Scales

\[ PM_{estimated} = 3.67 \times (Size)^{(SF)} \times \left( \prod_{i} EM_i \right) \]

\[ SF = 0.91 + 0.01 \times \sum w_i \]

<table>
<thead>
<tr>
<th>Scale Factors ((W))</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>thoroughly unprecedented</td>
<td>largely unprecedented</td>
<td>somewhat unprecedented</td>
<td>generally familiar</td>
<td>largely familiar</td>
<td>thoroughly familiar</td>
</tr>
<tr>
<td>FLEX</td>
<td>rigorous</td>
<td>occasional relaxation</td>
<td>some relaxation</td>
<td>general conformity</td>
<td>some conformity</td>
<td>general goals</td>
</tr>
<tr>
<td>RESL</td>
<td>little (20%)</td>
<td>some (40%)</td>
<td>often (60%)</td>
<td>generally (75%)</td>
<td>mostly (90%)</td>
<td>full (100%)</td>
</tr>
<tr>
<td>TEAM</td>
<td>very difficult interactions</td>
<td>some difficult interactions</td>
<td>basically cooperative interactions</td>
<td>largely cooperative</td>
<td>highly cooperative</td>
<td>seamless interactions</td>
</tr>
<tr>
<td>PMAT</td>
<td>weighted sum of 18 KPA achievement levels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
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* (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
COCOTRS Effort Distribution: 20 Projects

Mean % of Total COTS Effort by Activity (+/- 1 SD)

- Assessment: 49.07% (±7.57%)
- Tailoring: 50.99% (±7.48%)
- Glue Code: 61.25% (±0.88%)
- System Volatility: 20.27% (±2.35%)

% Person-months
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
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

- Literature, behavioral analysis
- Noisy data analysis

A-priori Bayesian update

- A-priori Experts’ Delphi
- Literature, behavioral analysis

Productivity Range = Highest Rating / Lowest Rating

Language and Tool Experience (LTEX)
## Step 8 Example: Software Understanding

Increment Too Large

- Needed to add a Programmer Unfamiliarity factor

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

• Build on small empirical-study homework assignments to local-industry students

• Assign empirical studies in industry short courses

• Look for industry pain points
  – COSYSMO: Need to be CMMI Level 3 in systems engineering

• Have your grad students do empirical studies as summer interns
  – Or do a similar internship yourself