Experience with Empirical Studies in Industry: Building Parametric Models

Barry Boehm, USC
boehm@usc.edu

CESI 2013
May 20, 2013

Outline

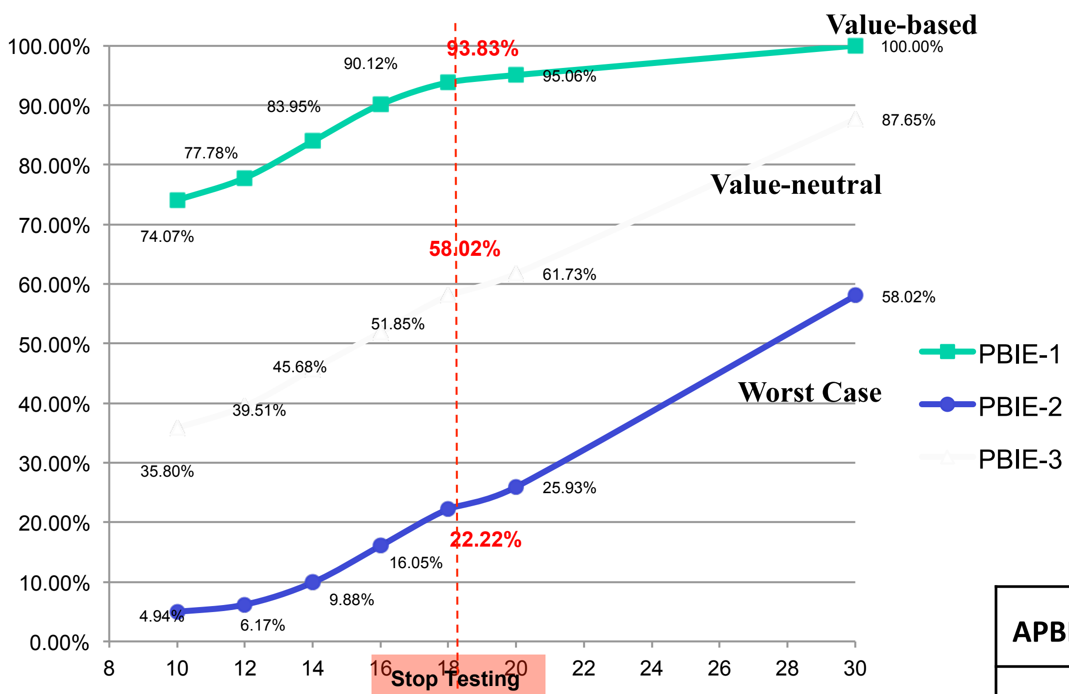
- Types of empirical studies with Industry
  - Types, benefits, challenges
  - Comparative methods, emerging technologies, parametric modeling
- Experiences with parametric modeling
  - Range of software engineering parametric models and forms
  - Goals: Model success criteria
  - 8-step model development 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



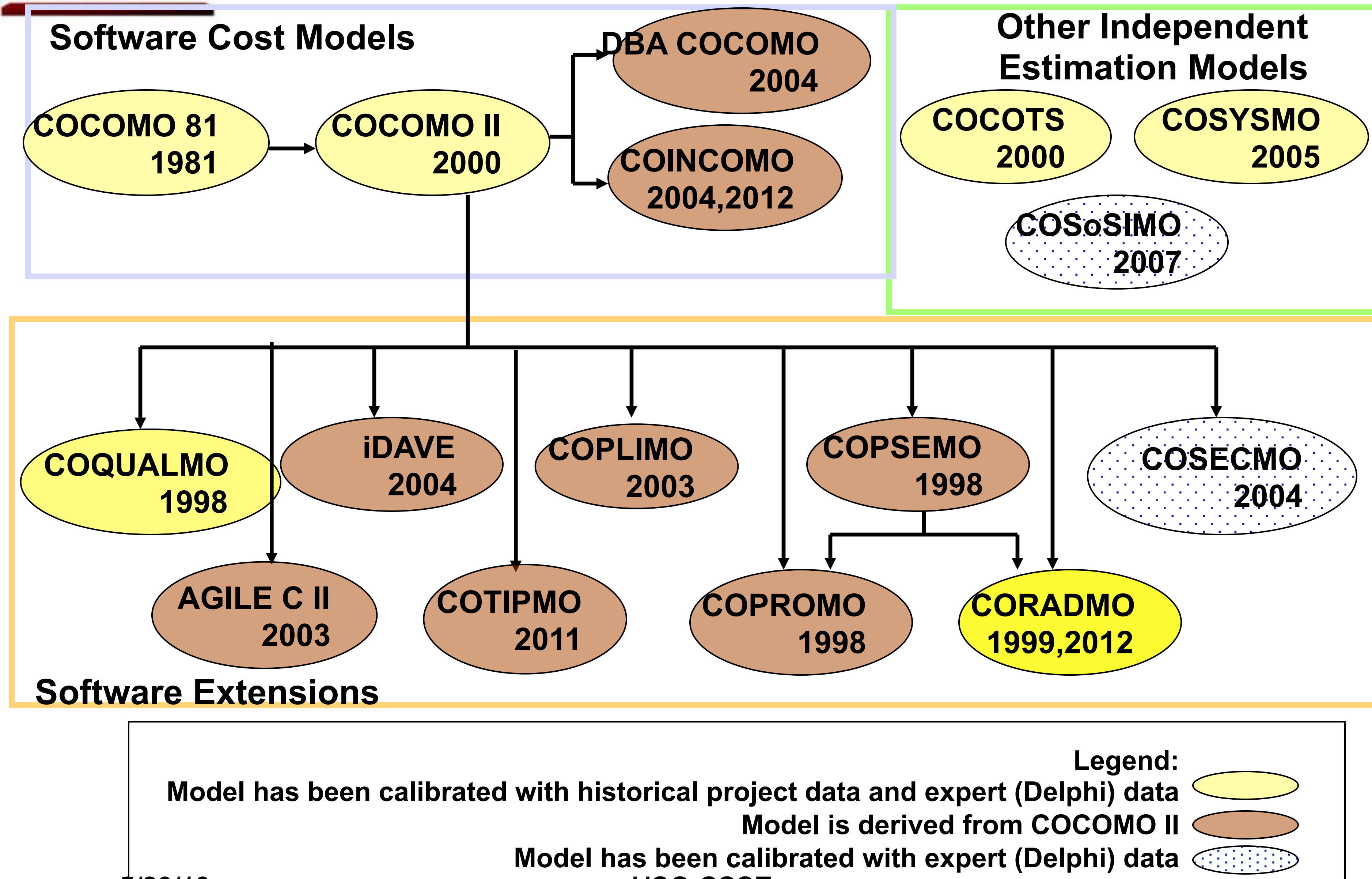
- H-t1: the value-based prioritization does not increase APBIE
  - reject H-t1
- Value-based prioritization can improve the cost-effectiveness of testing

APBIE-1	70.99%
APBIE-2	10.08%
APBIE-3	32.10%

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

COCOMO Family of Cost Models



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

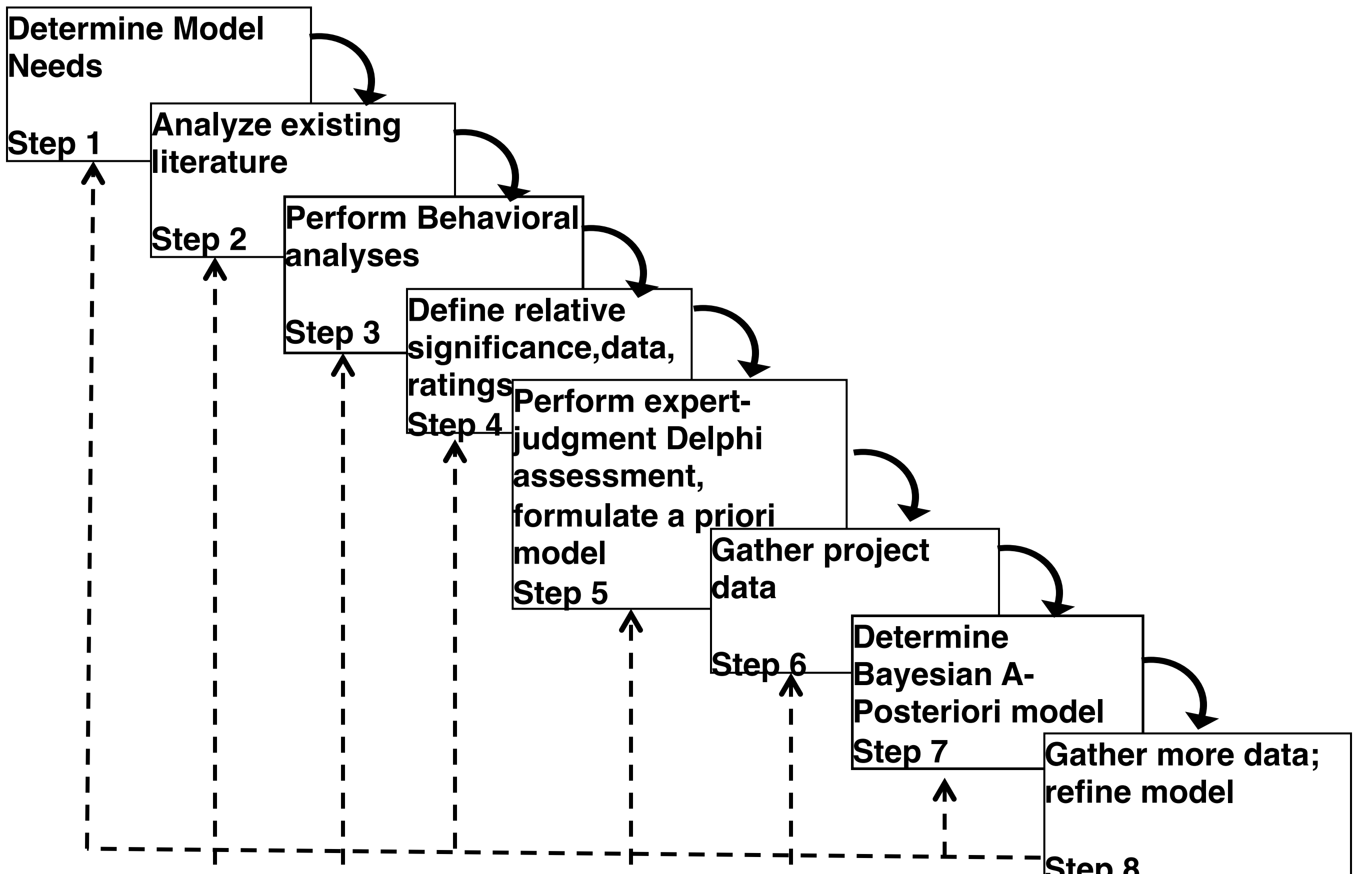
- Scope: Covers desired range of situations?
- Granularity: Level of detail sufficient for needs?
- Accuracy: Estimates close to actuals?
- Objectivity: Inputs repeatable across estimators?
- Calibratability: Sufficient calibration data available?
- Contructiveness: Helps to understand job to be done?
- 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?

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

- **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**

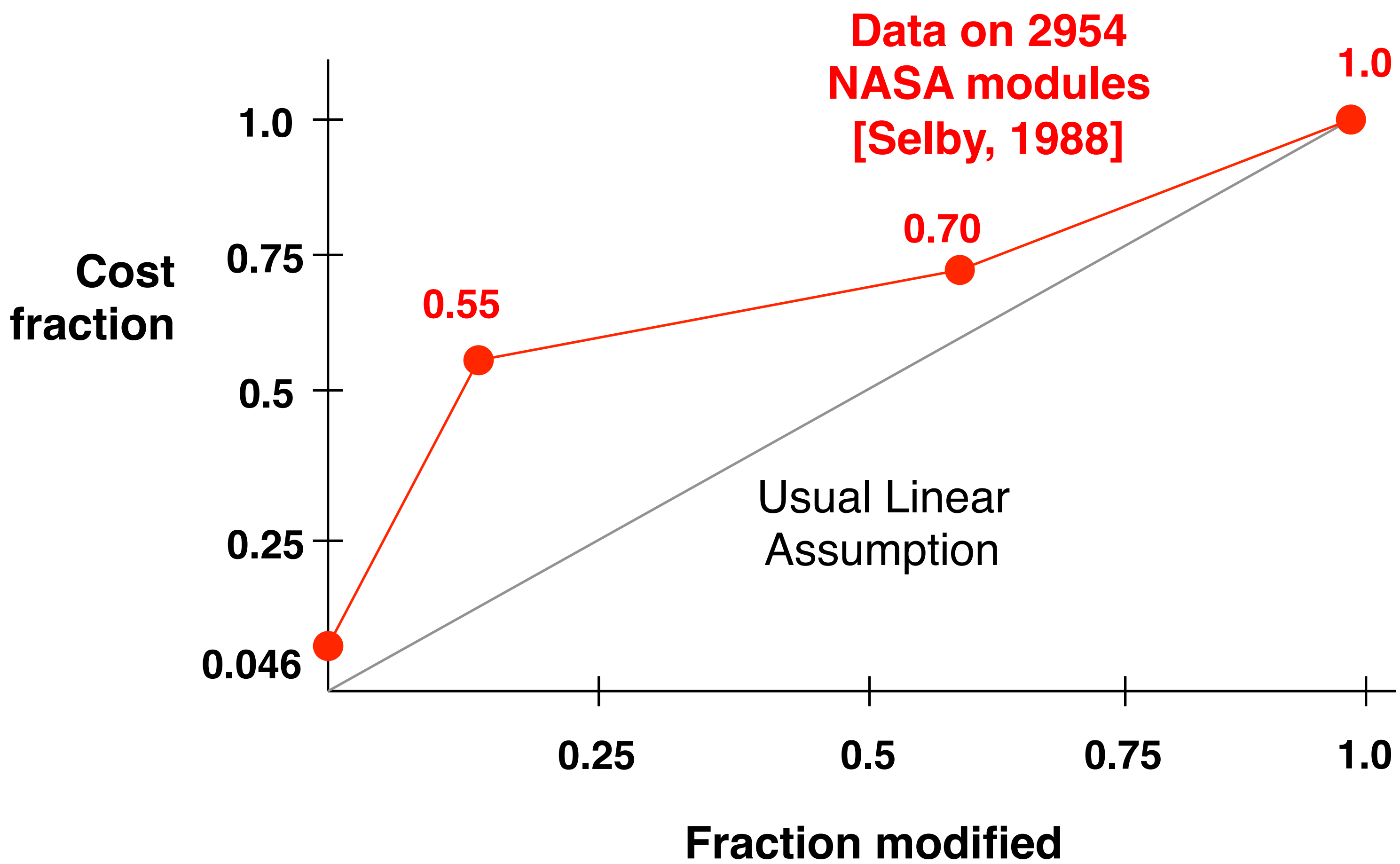
Major Decision Situations Helped by COCOMO II

- **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 (queuing models)
- Identify most promising outcome-driver parameters

Nonlinear Reuse Effects



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

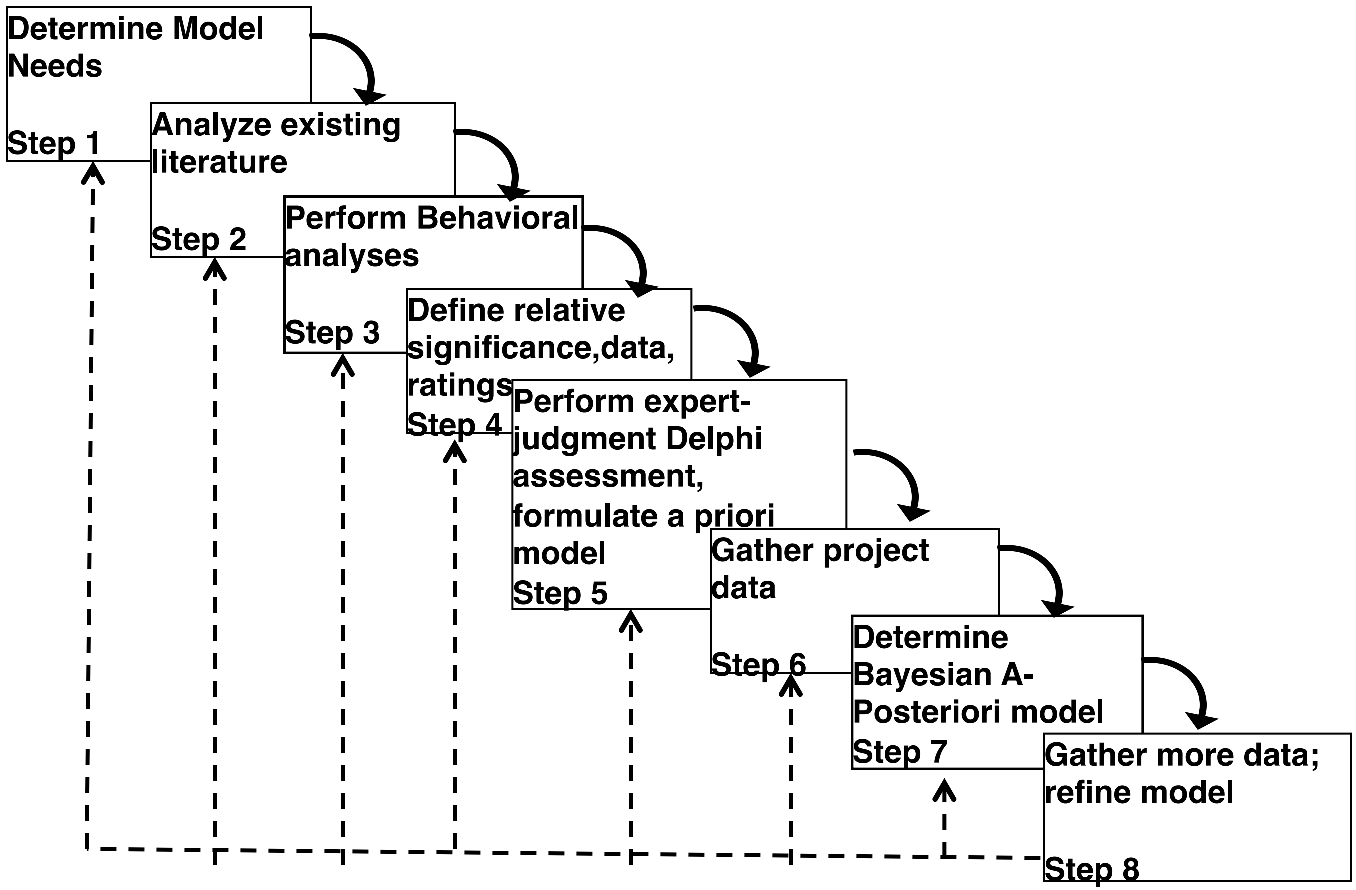
Step 3: Perform Behavioral Analysis

- **Behavior Differences: Required Reliability Levels**

Rating	Rqts and Product Design	Integration and Test
Very Low	•Little detail •Many TBDs •Little Verification •Minimal QA, CM, draft user manual, test plans •Minimal PDR	•No test procedures •Many requirements untested •Minimal QA, CM •Minimal stress, off-nominal tests •Minimal as-built documentation
Very High	•Detailed verification, QA, CM, standards, PDR, documentaion •IV&V interface •Very detailed test plans, procedures	•Very detailed test procedures, QA, CM, standards, documentaion •Very extensive stress, off-nominal tests •IV&V interface

USC-CSE Modeling Methodology

- concurrency and feedback implied



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

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

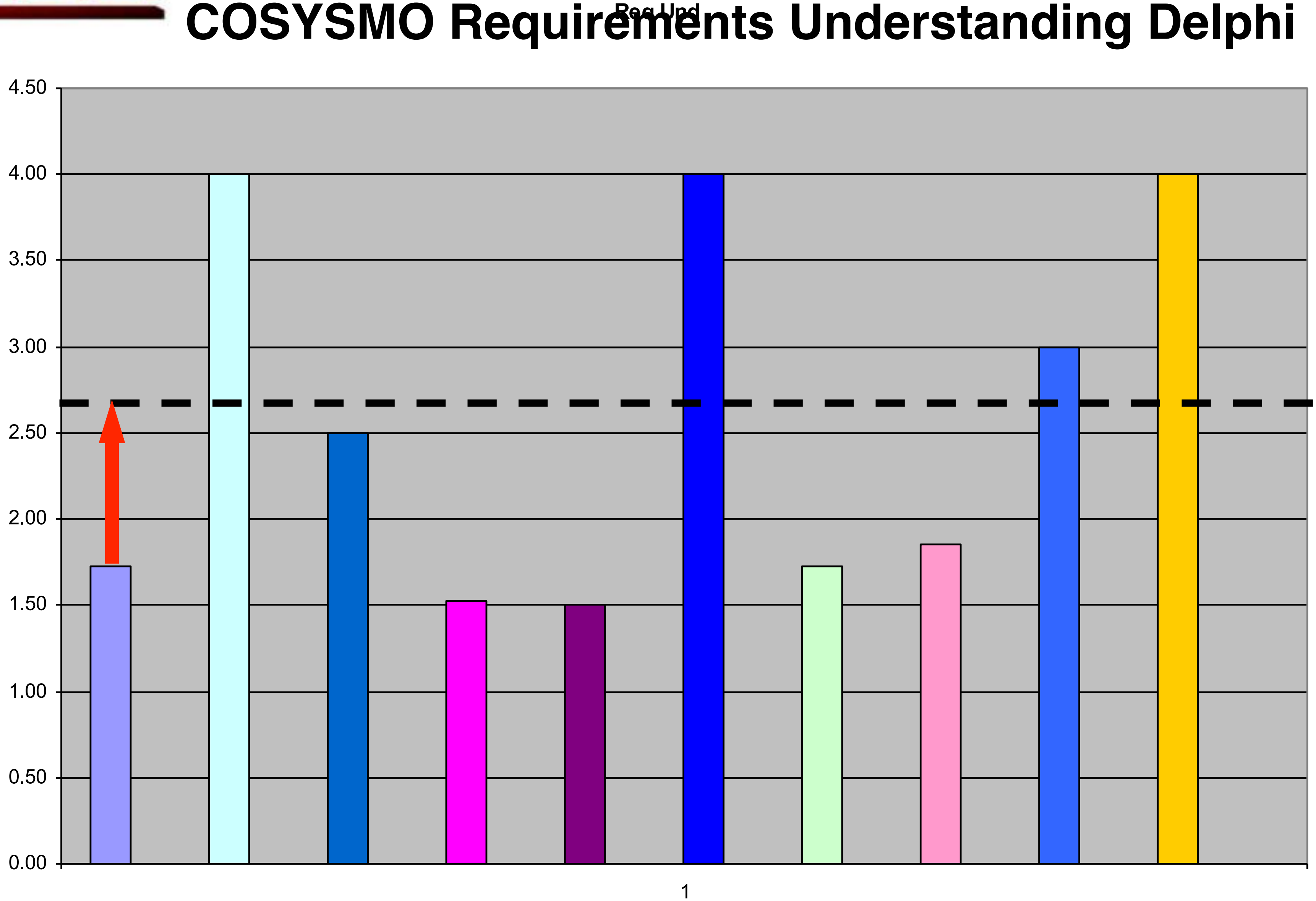
$$SF = 0.91 + 0.01 \times \sum_i w_i$$

Scale Factors (*Wi*)	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	throughly 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					

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**

COSYSMO Requirements Understanding Delphi



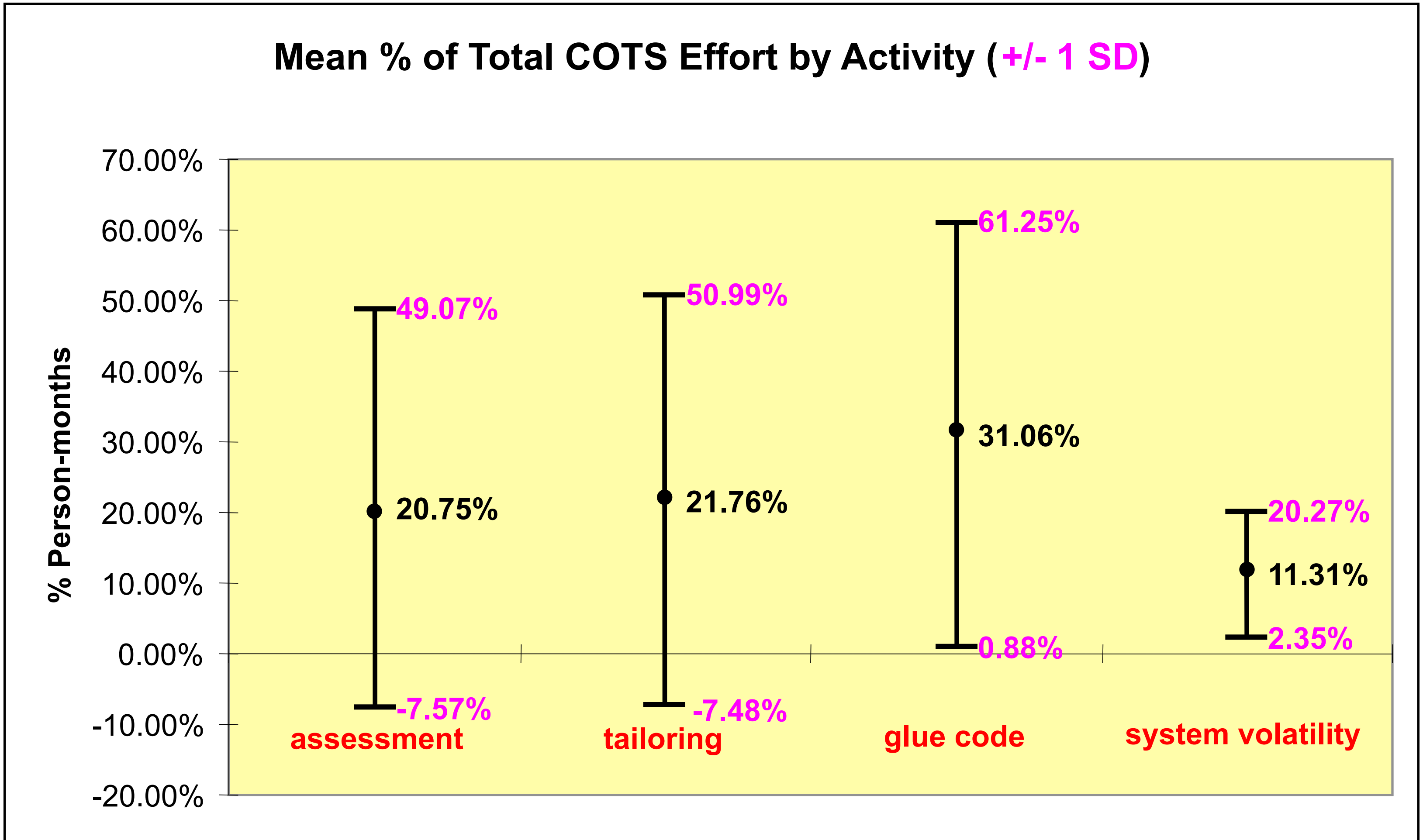
Step 6: Gather, Analyze Project Data

- **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* $(\text{Size})^{B}$* $\prod$(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



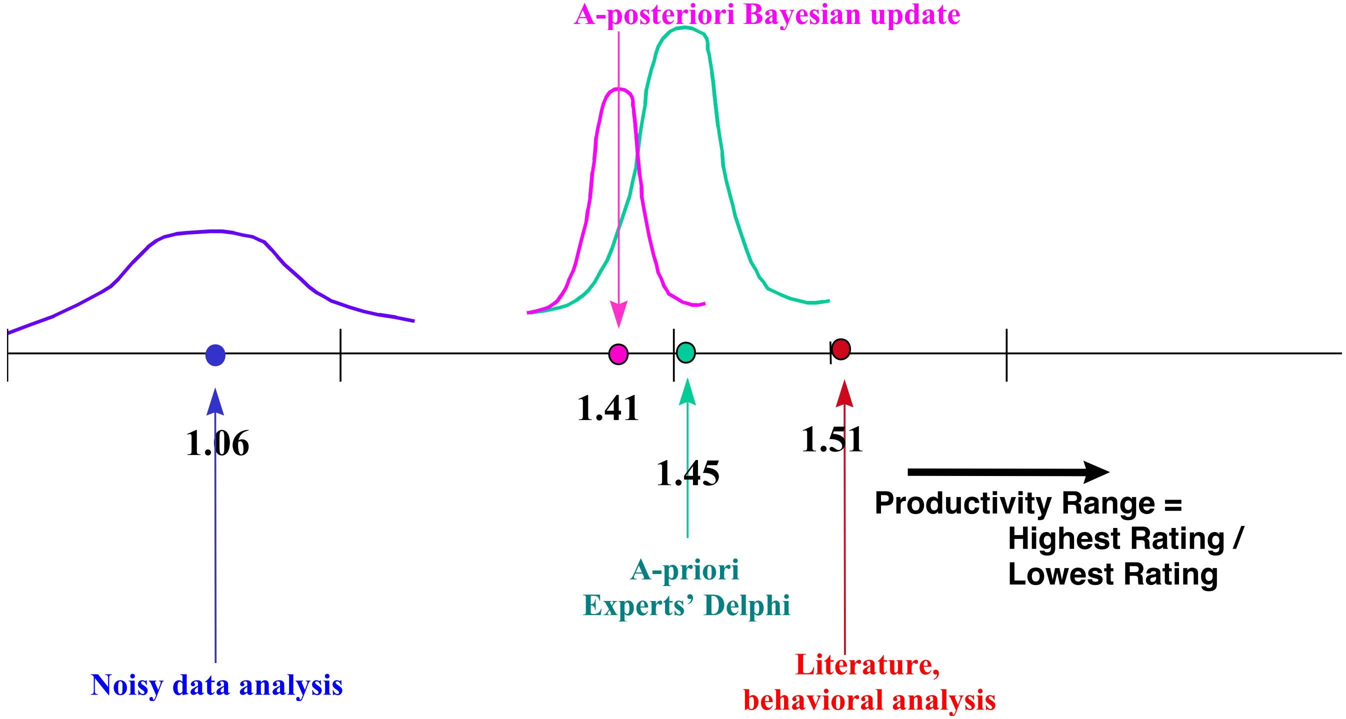
Revised COCOTS Model

- 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



Language and Tool Experience (LTEX)

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

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