



SMC-IT Cost Risk Tutorial



Model Based Estimate

Presented by:

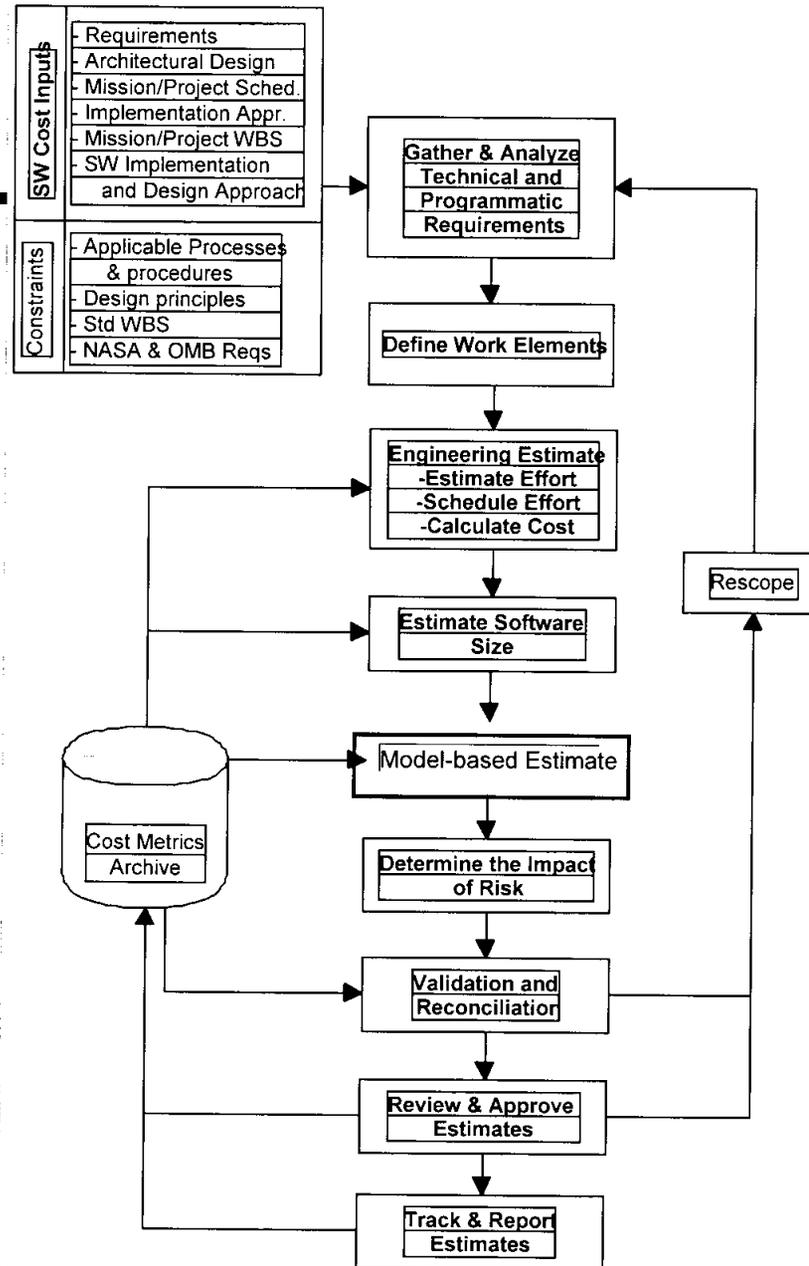
Jairus Hihn

JPL Software Quality Improvement Project

International Conference on
Space Mission Challenges for Information Technology
July 13, 2003



Software Estimation Steps





PARAMETRIC SOFTWARE COST ESTIMATION



- Model-based estimates are estimates made using parametric cost models
- Model-based estimates can be used
 - As a primary estimate early in life cycle
 - As a secondary backup estimate for validation
 - To help you “reason about the cost and schedule implications of software decisions you may need to make”
- Cost methodology using parametric models as described in this tutorial has been applied at JPL since 1990
- In the tutorial we will use JPL’s Software Cost Analysis Tool (SCAT) a probabilistic version of COCOMO II



Parametric Model – COCOMO II



- COCOMO II is a tool developed by the Center for Software Engineering (CSE), headed by Dr. Barry Boehm at the University of Southern California (USC)
- COCOMO II is an open model, so all of the details are published
- There are different versions of the model,
 - the Early Design Model
 - the Post-Architecture Model
 - primary supported version
 - version taught in class



Parametric Model – COCOMO II ⁽²⁾

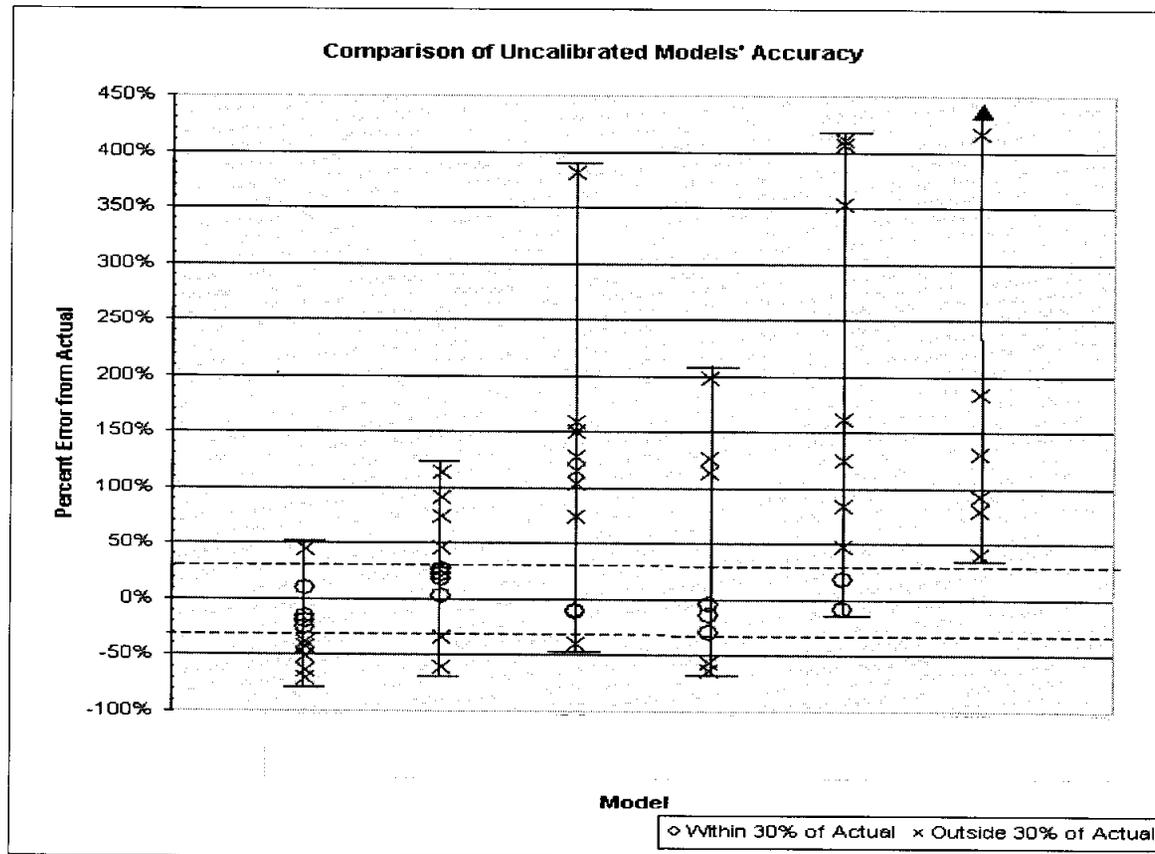


- SCAT uses Monte Carlo simulation to combine uncertainty distributions for each model input to produce total project cost probability distribution
 - Cost model built-in Monte Carlo simulation
- Can be used to develop confidence level of estimate relative to historical data set in the cost models
- Estimates include known and unknown unknowns



Models Must be Validated

Model Estimates vs. Actual Effort





COCOMO II Details



Standard Functional Form



- $E = [A (\text{Size})^B (\text{EM})]$
 - **E** is estimated effort in work-months.
 - **A** is a constant that reflects a measure of the basic organizational/technology costs.
 - **Size** is the equivalent number of new logical lines of code. Equivalent lines are the new lines of code and the new lines of adapted code. Equivalent lines of code takes into account the additional effort required to modify reused/adapted code for inclusion into the software product. The tools automatically compute the equivalent lines of code from the size and percentage inputs. Size also takes into consideration any code growth from requirements evolution/volatility.
 - **B** is a scaling factor of size. It is a variable exponent whose values represent economies/diseconomies of scale.
 - **EM** is the product of a group of effort multipliers that measure environmental factors used to adjust effort (E). The set of factors comprising EM are commonly referred to as cost drivers because they adjust the final effort estimate up or down.



COCOMO II Inputs



- COCOMO II's post architecture model requires 22 inputs
 - 17 effort multipliers
 - 5 scale factors
- Scale factors capture features of a software project that can account for relative economies or diseconomies of scale
- Effort multipliers characterize the product, platform, personnel, and project attributes of the software project under development



JPL Excel-Based COCOMO II



Microsoft Excel - cococomontecarlo3-11

File Edit View Insert Format Tools Data MonteCarlo Window Help

A31 APEX

	A	B	C	D	E	F	G
1	Name of Module	Module 1					
2		Inputs Enter in yellow colored cells					
3		Low	Most Likeli	High			
4	New SLOC	10000	20000	30000			
5	REVL %	5	6	5			
6	Adapted SLOC						
7	AA	+	+	+			
8	SU						
9	SU-Structure	Nomina	Nomina	Nomina			
10	SU-Application Clarity	Nomina	Nomina	Nomina			
11	SU-Self-Descriptiveness	Nomina	Nomina	Nomina			
12	UNFM	0.4	0.4	0.4			
13	% Design Modified	10.0	10.0	10.0			
14	% Code Modified	50.0	50.0	50.0			
15	% Integration Modified	100.0	100.0	100.0			
16	EFFORT MULTIPLIERS						
17	RELY	VH	VH	VH			
18	DATA	N	N	N			
19	DOCU	N	N	N			
20	CPLX						
21	CPLX- Control Operations	N	N	N			
22	CPLX- Computational Operations	N	N	N			
23	CPLX- Device Dependent Operations	N	N	N			
24	CPLX- Data Management Operations	N	N	N			
25	CPLX- User Interface Management Operation	N	N	N			
26	RUSE	N	N	N			
27	TIME	H-50	H-50	H-50			
28	STOR	H-75	H-75	H-75			
29	PVGL	N	N	N			
30	ACAP	N	N	N			
31	APEX	N	N	N			
32	PCAP	N	N	N			
33	PLEX	N	N	N			
34	LTEX	N	N	N			
35	PCON	N	N	N			
36	TOOL	N	N	N			
37	SCED	N	N	N			
38	SITE	N	N	N			
39	SCALE FACTORS						
40	PREC	N	N	N			
41	FLEX	N	N	N			
42	RESL	N	N	N			
43	TEAM	N	N	N			
44	PMAT	N	N	N			
45							
46							
47	Add New Module	Run Monte Carlo	Clear Estimate				
48							
49							

Finished Monte Carlo Simulation
11/10/2000

- Software size is the primary parameter
 - Logical SLOC
 - Function Points
- You provide estimates of new, reused and modified SLOC and the tool will calculate equivalent SLOC from which the COCOMO II equations will calculate effort



Inputs – COCOMO II Parameters



- Each of the COCOMO parameters is associated with up to six ratings – “very low,” “low,” “nominal,” “high,” “very high,” and “extra high.”
- Each rating has a corresponding real number based upon the factor and the degree to which the factor can influence productivity
- A rating equal to 1 does not increase nor decrease the schedule and effort (this rating is called “nominal”)
- A rating less than 1 denotes a factor that can decrease the schedule and effort
- A rating greater than 1 denotes a factor that increases the schedule or effort



COCOMO II EFFORT MULTIPLIERS (1)



Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
RELY Required Software Reliability	Effect of SW failure = slight inconvenience (0.82)	Effect of SW failure = low, easily recoverable losses (0.92)	Effect of SW failure = moderate, easily recoverable losses (1.00)	Effect of SW failure = high financial loss (1.10)	Effect of SW failure = risk to human life/public safety requirements (1.26)	
DATA Database Development Size		Testing DB Bytes/Program SLOC < 10 (0.90)	10 ? D/P < 100 (1.00)	100 ? D/P < 1000 (1.14)	D/P ? 1000 (1.28)	
DOCU Documentation Match to Life-Cycle Needs	Many life-cycle needs uncovered (0.81)	Some life-cycle needs uncovered (0.91)	Right-sized to life-cycle needs (1.00)	Excessive for life-cycle needs (1.11)	Very excessive for life-cycle needs (1.23)	
RUSE Developed for Reusability		None (0.95)	Across project (1.00)	Across program (1.07)	Across product line (1.15)	Across multiple product lines (1.24)
TIME Execution Time Constraint			?50% use of available execution time (1.00)	70% use of available execution time (1.11)	85% use of available execution time (1.29)	95% use of available execution time (1.63)
STOR Main Storage Constraint			?50% use of available storage (1.00)	70% use of available storage (1.05)	85% use of available storage (1.17)	95% use of available storage (1.46)
PVOL Platform Volatility		Major change every 12 mo.; Minor change every 1 mo. (0.87)	Major change every 6 mo.; Minor change every 2 wk. (1.00)	Major change every 2 mo.; Minor change every 1 wk. (1.15)	Major change every 2 wk.; Minor change every 2 days (1.30)	
ACAP Analyst Capability	15 th percentile (1.42)	35 th percentile (1.19)	55 th percentile (1.00)	75 th percentile (0.85)	90 th percentile (0.71)	



COCOMO II EFFORT MULTIPLIERS (2)



Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
PCAP Programmer Capability	15 th percentile (1.34)	35 th percentile (1.15)	55 th percentile (1.00)	75 th percentile (0.88)	90 th percentile (0.76)	
PCON Personnel Continuity	Annual personnel turnover: 48%/year (1.29)	24%/year (1.12)	12%/year (1.00)	6%/year (0.90)	3%/year (0.81)	
APEX Applications Experience	?2 months (1.22)	6 months (1.10)	1 year (1.00)	3 years (0.88)	6 years (0.81)	
PLEX Platform Experience	?2 months (1.19)	6 months (1.09)	1 year (1.00)	3 years (0.91)	6 years (0.85)	
LTEX Language and Tool Experience	?2 months (1.20)	6 months (1.09)	1 year (1.00)	3 years (0.91)	6 years (0.84)	
TOOL Use of Software Tools	Edit, code, debug (1.17)	Simple, frontend, backend, CASE, little integration (1.09)	Basic life-cycle tools, moderately integrated (1.00)	Strong, mature life-cycle tools, moderately integrated (0.90)	Strong, mature, proactive life- cycle tools, well integrated with processes, methods, reuse (0.78)	
SITE Multisite Development	Collocation: international; Communications: some phone, mail (1.22)	Collocation: multicity and multicompany; Communications: individual phone, fax (1.09)	Collocation: multicity or multicompany; Communications: narrow band email (1.00)	Collocation: same city or metro area; Communications: wideband electronic communication (0.96)	Collocation: same building or complex; Communications: wideband electronic communication, occasional video conf. (0.86)	Collocation: Fully collocated; Communications: Interactive multimedia (0.80)
SCED Required Development Schedule	75% of nominal (1.43)	85% of nominal (1.14)	100% of nominal (1.00)	130% of nominal (1.00)	160% of nominal (1.00)	



COCOMO II Required Reliability Check List



Rating	Requirements and Product Design	Detailed Design	Code and Unit Test	Integration and Test
Very Low	Little Detail, Many TBDs, Little verification, Minimal QA and CM, Draft user manual and test plans, Informal design inspections	Basic Design information, Minimal QA and CM, Draft user manual and test plans, Informal design inspections	No test procedures, Minimal path test, standard checks, Minimal QA and CM, Minimal I/O and off-nominal tests, Minimal user manual	No test procedures, Many requirements untested, Minimal QA and CM, Minimal stress and off nominal tests, Minimal "as built" documentation
Low	Basic information and verification, Frequent TBDs, Basic QA and CM, Standards, draft users manual and, test plans	Moderate detail, Basic QA and CM, draft users manual and test plans	Minimal test procedures, Partial path test and standards check, Basic QA, CM and user manual, Partial I/O and off-nominal tests	Minimal test procedures, frequent requirements untested, Basic QA, CM and user manual, Partial stress and off nominal tests
Nominal	Nominal project V&V	Nominal project V&V	Nominal project V&V	Nominal project V&V
High	Detailed verification, QA, CM, Standards, PDR and documentation, Detailed test plans and procedures	Detailed verification, QA, CM, Standards, CDR and documentation, Detailed test plans and procedures	Detailed test procedures, QA, CM and documentation, Extensive off-nominal tests	Detailed test procedures, QA, CM and documentation, Extensive stress and off-nominal tests
Very High	Detailed verification, QA, CM, Standards, PDR and documentation, IV&V interface, Very detailed test plans and procedures	Detailed verification, QA, CM, Standards, CDR and documentation, Very through design inspections, IV&V interface, Very detailed test plans and procedures	Detailed test procedures, QA, CM and documentation, Very thorough code inspections, Very extensive off-nominal tests, IV&V interface	Very detailed test procedures, QA CM and documentation, Very extensive stress and off-nominal tests, IV&V interface



COCOMO II EFFORT MULTIPLIERS (3)



	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low (0.73)	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IF-THEN-ELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., $A = B + C * (D - E)$	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low (0.87)	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderately level expressions: e.g., $D = \text{SQRT}(B^2 - 4 * A * C)$	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal (1.00)	Mostly simple nesting. Some inter-module control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High (1.17)	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round off concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlaps.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O multimedia.
Very High (1.34)	Reentrant and recursive coding. Fixed-priority interrupt handling. Tasks synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High (1.74)	Multiple resource scheduling with dynamically changing priorities. Microcode-level	Difficult and unstructured numerical analysis: highly accurate analysis	Device timing-dependent coding, micro-programmed operations.	Highly coupled, dynamic relational and object structures.	Complex multimedia, virtual reality.



COCOMO II EFFORT MULTIPLIERS (3)



	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low (0.73)	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IF-THEN-ELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., $A = B + C * (D \text{ OR } E)$	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low (0.87)	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderately level expressions: e.g., $D = \text{SQRT}(B^2 - 4 * A * C)$	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal (1.00)	Mostly simple nesting. Some inter-module control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High (1.17)	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round off concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlaps.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O multimedia.
Very High (1.34)	Reentrant and recursive coding. Fixed-priority interrupt handling. Tasks synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High (1.74)	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.



COCOMO II Scale Factors



	Very Low	Low	Nominal	High	Very High	Extra High
PREC Precedentedness (See Table 2 for Details)	thoroughly unprecedented (6.20)	largely unprecedented (4.96)	Somewhat unprecedented (3.72)	generally familiar (2.48)	largely familiar (1.24)	thoroughly familiar (0.00)
FLEX Development Flexibility (See Table 2 for Details)	Rigorous (5.07)	occasional relaxation (4.05)	Some relaxation (3.04)	General conformity (2.03)	Some conformity (1.01)	general goals (0.00)
RESL Architecture/Risk Resolution (See Table 2 for Details)	little (20%) (7.07)	some (40%) (5.65)	often (60%) (4.24)	Generally (75%) (2.83)	mostly (90%) (1.41)	full (100%) (0.00)
TEAM (See Table 2 for Details)	very difficult interactions (5.48)	some difficult interactions (4.38)	Basically cooperative interactions (3.29)	Largely cooperative (2.19)	Highly cooperative (1.10)	Seamless interactions (0.00)
PMAT Process Maturity	CMM Level 1 (Lower half) (7.80)	CMM Level 1 (Upper half) (6.24)	CMM Level 2 (4.68)	CMM Level 3 (3.12)	CMM Level 4 (1.56)	CMM Level 5 (0.00)



COCOMO II Size and Reuse Parameters



	Low	Most Likely	High	Comments
# of New SLOC (Logical)				
# of Reused or Inherited or Adapted SLOC (Logical)				
Percentage of Reused SW Design Modified	%	%	%	
Percentage of Reused SW Code Modified	%	%	%	
% Effort Required for Integration and Test of SW relative to normal amount of integration and test for SW of comparable size	%	%	%	
Requirements Evolution (REVL): % of code thrown away due to requirements volatility	%	%	%	



COCOMO II Reuse Parameters



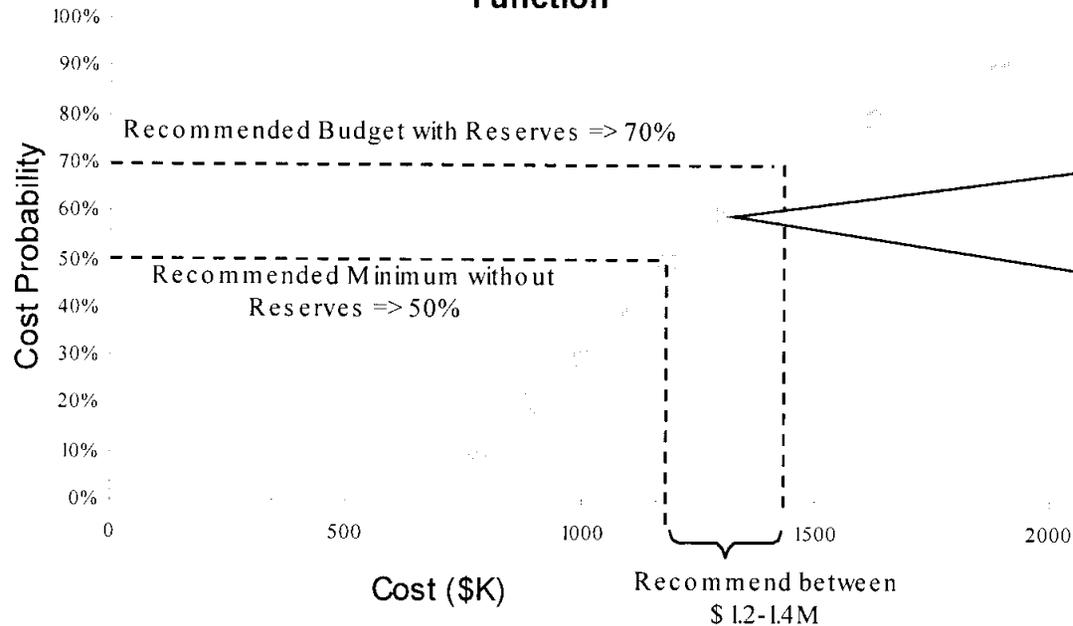
Assessment and Assimilation (AA)	None (0)	Basic module search and documentation (2)	Some module Test and Evaluation, documentation (4)	Considerable module Test and Evaluation, documentation (6)	Extensive module Test and Evaluation, documentation (8)		
Programmer Unfamiliarity with Software (UNFM)	Completely familiar (0)	Mostly familiar (0.2)	Somewhat familiar (0.4)	Considerably familiar (0.6)	Mostly unfamiliar (0.8)	Completely unfamiliar (1.0)	
	Very Low	Low	Nominal	High	Very High		
Software Understanding (SU):							
Structure	Very low cohesion, high coupling, spaghetti code (50)	Moderately low cohesion, high coupling (40)	Reasonably well-structured; some weak areas (30)	High cohesion, low coupling (20)	Strong modularity, information hiding in data/control structures (10)		
Application Clarity	No match between program and application worldviews	Some correlation between program and application	Moderate correlation between program and application	Good correlation between program and application	Clear match between program and application worldviews		
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		



Example Model Output



Software Development Cost Cumulative Distribution Function



- For tasks with 10% level of reserves or less recommend a range of 50% to 70% probability
- For tasks with 20% or greater reserves recommend 40-65% for other Subsystems



Limitations and Constraints (1)



- In addition, before using any parametric model, it is important to note that each tool provides cost and effort estimates that may include different activities/phases and different labor categories than your plan and budget
- Sometimes it may appear that a tool is overestimating by a large margin, but it may be found that the estimate includes field testing, concept study, formal Quality Assurance, and Configuration Management, while you did not require those activities and labor categories to be estimated



Limitations and Constraints (2)



- Many of the models also have limitations as to the size of a development project for which it can forecast effort. Most models cannot accurately forecast effort for development projects under and over a certain number of lines of code.
 - COCOMO II, for example, is not calibrated for projects below 2,000 SLOC in size. It is recommended that projects smaller than this limit not use commercial cost tools for estimating costs and effort.
- It is better to break down the system into smaller work elements
 - This is important because most models will take the size as one large function rather than many smaller work elements, and overestimate the effort



WRAP UP



- Models provide a method for quickly generating back-up estimates
- SCAT: Contact karen.t.lum@jpl.nasa.gov
- The USC version can be downloaded from <http://sunset.usc.edu/> and then select COCOMO suite
- ASK Pete developed at Glen Research Center provides a rule-based version of COCOMO II with other extensions and can be downloaded from <http://osat-ext.grc.nasa.gov/rmo/pete/index.html>