Basic Software Cost and Schedule Estimation: Guessing at how long and how much.

Bryce L. Meyer
St. Louis UNIX Users Group
14 June 2017

Overview

- Why can’t we predict our software projects?
- Approaches to schedule and cost prediction
- Tricks of the Trade
- Data Availability and Dangerous Data
- A few Common Methods and their limits

Why can’t we predict our software projects?

- Software is HARD!
- Complexity-Entropy Law (expanded from Thermodynamics): The more complex or massive a project is, the more chaos will reign!
  - Chaos/Entropy=uncertainty (usually in a bad way) in costs and schedule!
- You can usually get only 2 of (sometimes 1 of): Good Performance, Short Schedule, Low Cost.
- By the time you have enough data to confidently predict how much a project will cost, the project is over!
- Quality, Planning, and Architecture mitigate chaos!
  - All these play into not just predicting cost and schedule, but performance of the software, and happiness of the workers too!
- Most software cost is AFTER beta delivery.
  - No software survives first contact with the users!

Ways to Estimate Costs and Schedule for Software and Software Heavy Systems.

There are a few ways to estimate cost and schedule for software efforts:

- **Comparative:** Find something that allows comparison to other projects, then interpolate or extrapolate
- **Parametric:** Collect data to feed a series of equations with coefficients from other past projects (a historical data set).

https://opensource.com/business/14/1/top-project-management-tools-2014

Traditional Estimation (Decomposition + Comparative):

1. Break down the project into small enough parts that someone can guess based experience
   - Hopefully their experience is relevant?
   - Usually a hierarchy of parts, i.e. parts of parts.
2. Add up the guesses (using MS Project, Open Project, other tool) and add 'project manager discretion' (i.e. a multiplier of 2.4+/-).

https://opensource.com/business/14/1/top-project-management-tools-2014

Basic Types of Software Cost Models

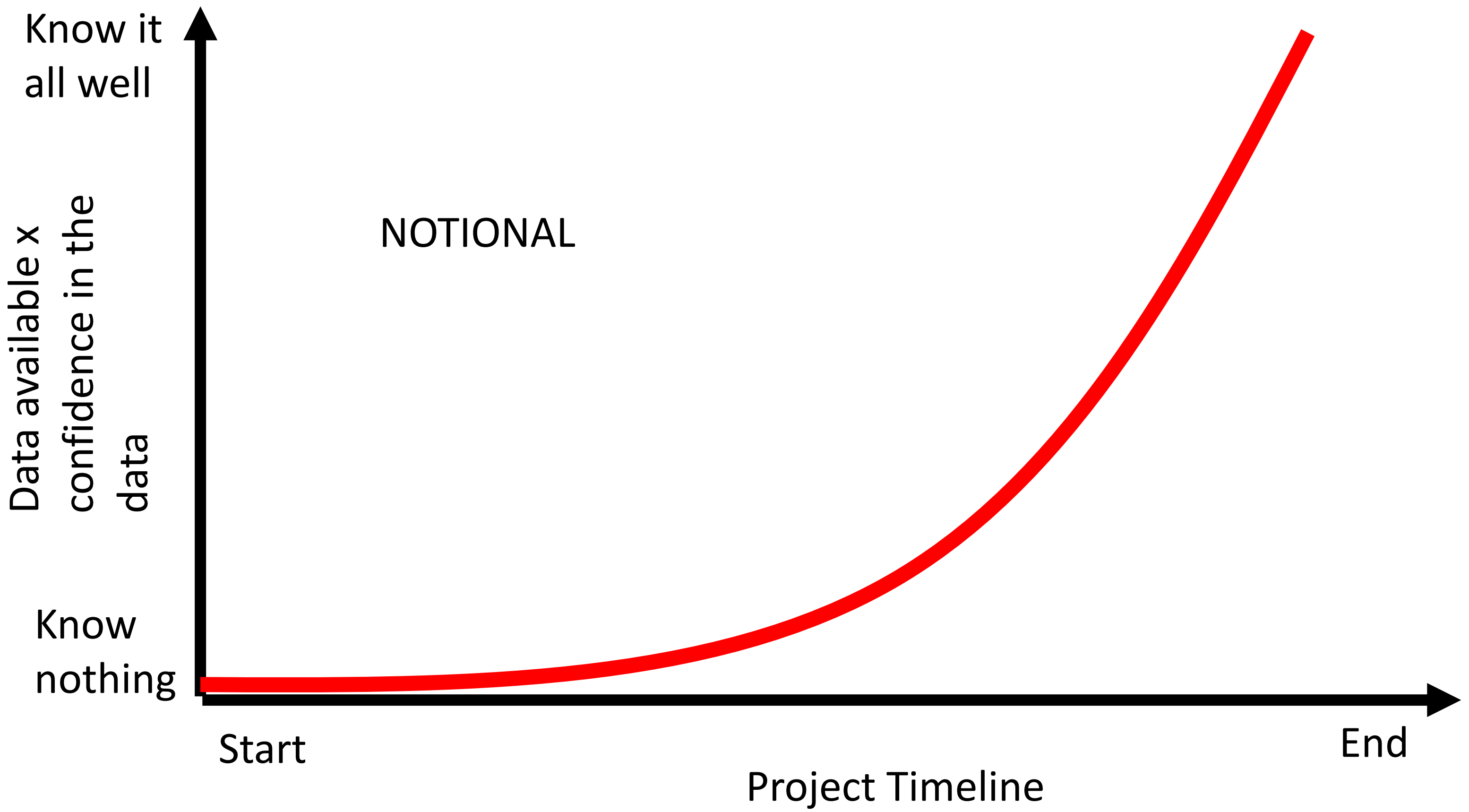
Software targeted cost models fall into combinations of 4 basic modes, as in the matrix below:

Approach / Estimation Method	**Decomposition:** Break down Software/System into design or project parts small enough to estimate costs	**Systematic:** Estimate at a high level by system or project.
Comparative (Analogy): Estimate by interpolation or extrapolation from historically similar efforts	Comparative estimation of each element in a Decomposition (Traditional Method)	Comparative estimation at the System/Project level (also Traditional…the W.A.G.)
Parametric: Estimate with a mathematical model that incorporates factors from the effort	Parametric estimation of each element in a Decomposition	Parametric estimation at the System/Project level

I put expert guesses as Comparative.

Conundrum of Software Estimation

- You have the most and best data when you are done!



Software Cost Models: Required Data

- For each type below, different data is required to complete the estimation. What you can get may determine what models you use.

Approach / **Estimation Method**	**Decomposition:** Break down Software/System into design or project parts small enough to estimate costs	**Systematic:** Estimate at a high level by system or project.
Comparative(Analogy): Estimate by interpolation or extrapolation from historically similar efforts	1) Complete breakdown at a low enough level for comparison 2) Comprehensive set of costs of comparative elements and data for correct comparison	Comprehensive set of costs of comparative systems/projects and data for correct comparison. Must be in the same variables as used to compare
Parametric: Estimate with a mathematical model that incorporates factors from the effort	1) Complete breakdown at a low enough level for comparison 2) Factors to feed the parametric model for each element.	Factors to feed the parametric model for the system/project

Data Collection is HARD!

(Good) Data for use in cost estimation is hard to get:

- People in legacy programs impacted by new efforts due to integration or replacement:
  - Are not usually the ones who developed the system, nor were around when specified
  - Fear for their jobs therefore resent external parties asking about software they own. (Don't hurt my baby!)
  - Fear criticism for technical decisions made about the software. (Don't call my baby ugly!)
- Even if the program is long gone or unaffected by the estimate, they will still react as in impacted programs!

Data Collection is HARD! (Again)

More reasons why data for cost estimation and cost models is hard to get:

- May be proprietary to the company who made the software, or under Intellectual Property concerns.
- May be under security concerns
- Is often not shared in corporate knowledge bases (due to territoriality or mistrust or obscurity)
- Is rarely available on the general internet in detail enough for good estimation.

More Real World Estimation Experiences

- BS in = BS out.
  - If variables or comparison data is bad, the cost estimate will also be bad. Coarse guesses = low confidence results.
  - Trick: Reduce opinion range by expanding questions. That way a single error is mitigated in the model.
- Managers and Engineers are OPTIMISTIC when estimating effort and cost!
  - In the Real World, most coding shops produce less than 300 equivalent lines of code (final) per person per MONTH!
- Estimators must soothe each data holding group to gain their trust, and be willing to at least feel their concerns
- Estimators need a back-up data source to generate a ceiling and a floor for estimates in the event data is impossible to get in time, or is flippant.
- Use multiple models to compare results.

More Real World Estimation Experiences

- If you are estimating some else's project: estimators need to understand the software technologies in the system.
  - MBA does not mean I get software!
- If you are estimating some else's project: Commandment: THOU SHALT NOT WASTE THE TIME OF DATA HOLDERS!
  - Be empathetic!
  - Go in with a solid story, explanation of method to the equation level, and series of bullets for common questions
  - Know the ranges of effects for each answer and the sensitivity of variables.
  - Have a strong, well developed strategy with options to get to a solid set and range of costs.
- NOTE: Integration always gets short-changed! Integrating chunks of a complex project takes lots of testing and fix time.

How Do I Break Out A Software Project?

- An effort can be broken down two ways:
  - By Time phase
  - By Component
    - Example: by service, by object, by function, etc.
  - The Work Breakdown Structure (WBS) can be either or both.
- Core functions may be working long before test and fix is complete for software projects esp. due to:
    - Security + Resiliency
    - User Interface alterations
    - Interfaces to other systems
    - Non-core functions
- Project Tools

Comparing Historical Data To Get At Costs/Schedule

- To compare a project you have, against historical data, you need a common set of comparison variables.
- Common comparison items:
  - Organizational:
    - Team Experience
    - Team Cohesion
    - Management Effectiveness
    - Quality Process
    - Requirements knowledge
    - Ability to convert requirements to design
    - User Involvement
    - Size of User Base and Threats
  - Technical:
    - Expected software lines of code or functional elements
    - Complexity of objects/functions/services
    - Testability of Requirements
    - Limits on Bandwidth/Performance/Storage
    - Criticality, Required Availability (ex: Real Time, Safety Critical, **Cybersecurity**)

Sizing in Software

- Sizing (Size, functional size measure) is useful for comparing data
- Sizing is a way to determine the magnitude of a software project effort and time
  - It is NOT just Lines of Code, but may use Source Lines of Code (SLOC) with multipliers based on models and experience to match projects.
- Function Points may be used as a size metric also
  - Function Points are measurable characteristics of a software project, that indicate size.
  - Many Standards can be used to calculate function points:
    - **COSMIC: ISO/IEC 19761:2011 (most commonly used)**
    - FiSMA: ISO/IEC 29881:2010
    - IFPUG: ISO/IEC 20926:2009
    - Mark-II: ISO/IEC 20968:2002
    - NESMA: ISO/IEC 24570:2005
  - Best to have both tools and expertise when determining function points
- In all cases, an understanding of the software effort, and a series of rules for deriving characteristics, are required.

http://cosmic-sizing.org/cosmic-fsm/

Function Points the Easy Way

- Get all these:
  - Interfaces out of, and into, the software.
    - Multiple each by a complexity (1 = normal, 3=complex)
  - Core Functions or Services (aka Algorithms)
    - Again use a multiplier for each before adding
  - High Level User Scenarios/Use Cases
    - Again use a multiplier for each before adding
  - High Level Requirements
    - Again use a multiplier for each before adding
- Add them up to get Function Points!
- Ex: RICEF(W)S are how SAP does Function Points….

Lines of Code? SLOC? ESLOC?

- In order to get an Apples to Apples comparison of Lines of Code, a reference language is picked, and a set of rules are applied.
  - Rules for: Comment Lines, Declaration Lines, etc.
  - What about configuration items and data sets (ex: css, xml DTD, XML in general, etc.)
  - Even in the same language, all lines are not equal!
  - Count of Lines of Code (CLOC) is the raw count of every line in your code.
  - Source Lines of Code (SLOC) is the count of code in the language you used using some rules.
  - Equivalent Source Lines of Code (ESLOC) is SLOC adjusted against a ruleset and common equivalent language.
    - Given a big count, the worth of each line averages out, as long as the language is ruled in.
- Most Models assume you made a series of choices, or guesses:
  - In modifying existing code, you can run a tool to get a count. Most modern software management tools will calculate a SLOC, maybe an ESLOC.
- Fortunately, there are tools that help:
  - Unified Code Counter: http://csse.usc.edu/ucc_new/wordpress/
  - CLOC: http://cloc.sourceforge.net/

Old resource: https://resources.sei.cmu.edu/asset_files/SpecialReport/1995_003_001_16358.pdf

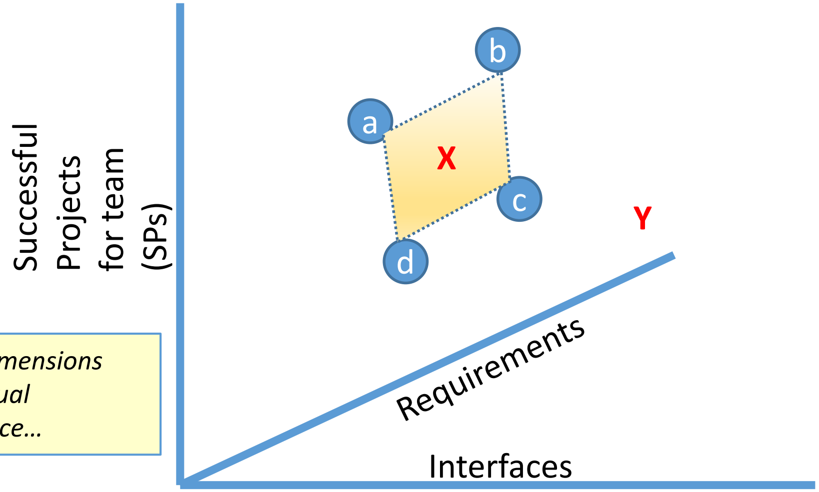
Statistics, Interpolation, and Extrapolation

- Most (all) models use a historical database that has been standardized to a series of variables (the N dimensional space previously mentioned).
- Nearest Neighbor, Liner Interpolation, Polynomial Interpolation, and Multivariate interpolation is used in many models to estimate values from historical data.
- Coefficients from the historical data can then be analyzed using ANOVA or other statistical methods to arrive at coefficients for (parametric) models or extrapolation.
- Monte Carlo methods are often used to explore variables for early phase programs with sparse data to make estimates using the historical statistics and get a confidence interval.
- Monte Carlo can also be used to find most probable scheduling curve.
- Armed with coefficients and confidence, planners can explore cost and schedule options.

Interpolation and Extrapolation (for Sizing)

- Can be used to generate sizing for early stage projects to feed other models

These have to be standardized using weights because not all requirements or interfaces are of the same risk/complexity



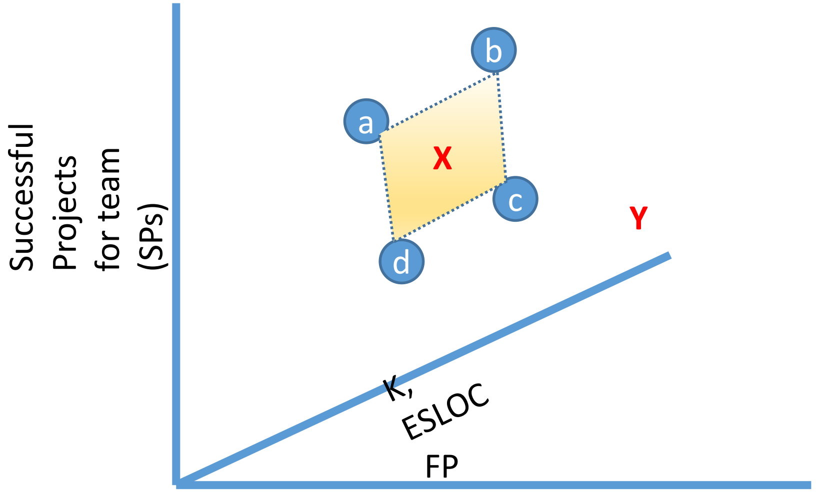
Not all dimensions are of equal importance…

project	Requirements	Interfaces	SPs	ESLOC	FPs
a	2067	203	19	1023	12
b	3102	410	21	1493	20
c	3420	530	9	3230	48
d	1920	190	8	2301	30
X	2000	300	16	**? Between 1023-3230**	**? Between 12-48**
Y	4000	550	2	**? >3230**	**? >48**

Note: Far more dimensions and comparatives required to be anywhere near accurate.
Can use linear or other extrapolation models to get Y

Interpolation and Extrapolation (for cost/duration)

- The characteristics form an N dimensional space
- Databases that contain comparison data are often proprietary



project	KESLOC	FP	SPs	ppm cost	Duration *(months)*
a	2067	203	19	1023	12
b	3102	410	21	1493	20
c	3420	530	9	3230	48
d	1920	190	8	2301	30
X	2000	300	16	**? Between 1023-3230**	**? Between 12-48**
Y	4000	550	2	**? >3230**	**? >48**

Note: Far more dimensions and comparatives required to be anywhere near accurate.

Parametric Models vs Comparative Models

- Parametric models are useful:
  - When Data is Limited for a new project, so not available for comparison
  - When modifying or integrating existing projects, i.e. projects in progress
    - Most Comparison is for either phases, or whole projects.
  - When a WBS is not available in detail
  - When the estimate has to be done quick (no time to fill in a decomposition)
- Parametric Models are only as good as the data used in development
  - The Historical Database is used to tune the model, the better and closer it is to your project, the more accurate
- Parametric Models: Good for tracking current State against previous estimates.
  - Are we still tracking?

SUPER SIMPLE (and Risky) Parametric Method

- Cost = Adjusted Size * (Some Coefficient from a table or your org’s history)
- Schedule = Adjusted Size * (Some Coefficient from a table or your org’s history)

- You could use exponents and additive methods to make it more complex.

COCOMO and COSYSMO Parametric Model Family

- Dr. Barry Boehm started work on Constructive Cost Model (COCOMO) in the 1970s, with COCOMO out in 1981. Work lead to formation of the Center for Systems and Software Engineering @ University of So. Cal.
- All in the family are multiplicative models, i.e. coefficients from a knowledge matrix are multiplied by inputs, then a product (**Π**) is used to combine them with a 'size' estimate.
  - Exponents are also used.
- COCOMO lead to COCOMO II (~1995)
- COSYSMO (Constructive Systems Model)(for software systems, 2005 (v1), 2009(v2))
  - Jared Fortune and Ricardo Valerdi enhanced COSYSMO to get COSYSMO v2.0 w/aid of Dr. Boehm
- COCOTS is a newer model that wraps in COTS integration
  - http://csse.usc.edu/csse/research/COCOTS/index.html
- Free to use for Gov't/Academia

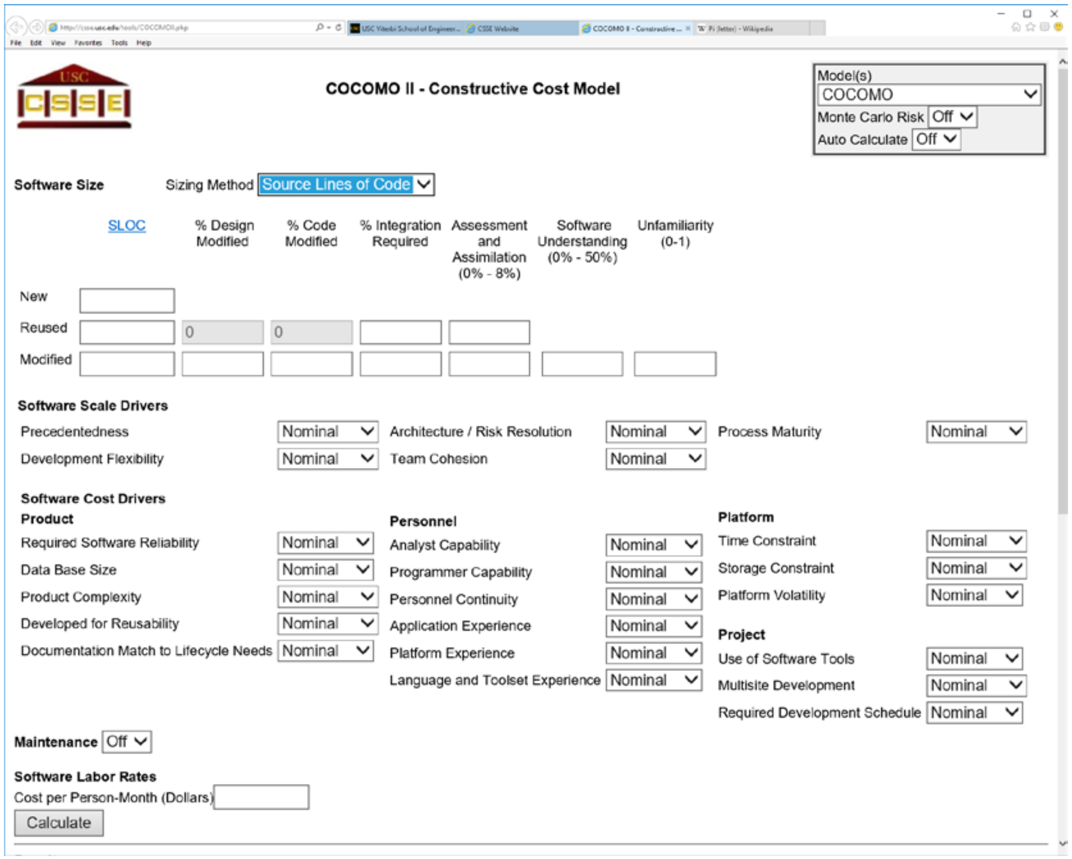
http://csse.usc.edu/csse/index.html

http://sunset.usc.edu/csse/research/cocomoii/cocomo_main.html

COSYSMO v. COCOMO

- COCOMO Family requires either Function Points or Source Lines of Code (SLOC), therefore requires a more mature design
- COSYSMO requires only architecture parameters, so works for very early development exploration.
- Can be used in concert

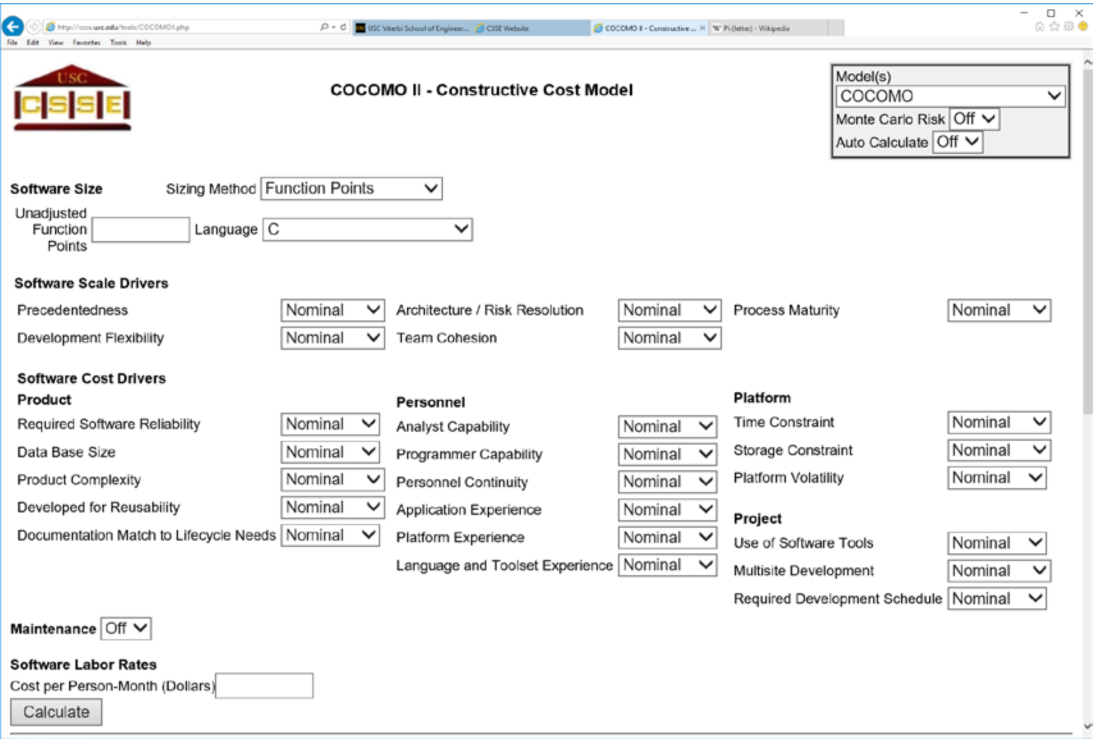
COCOMO II



- Can use Source Lines of Code (SLOC) or Function Points.
  - SLOC can be Equivalent lines against a reference language
- 13 Quantitative Inputs (not counting $$/person/month) in SLOC mode
- SLOC is primary driver, rest multipliers in SLOC Mode
- 22 Qualitative Inputs: 5 for size scale, 17 for cost drivers

→ http://csse.usc.edu/tools/COCOMOII.php

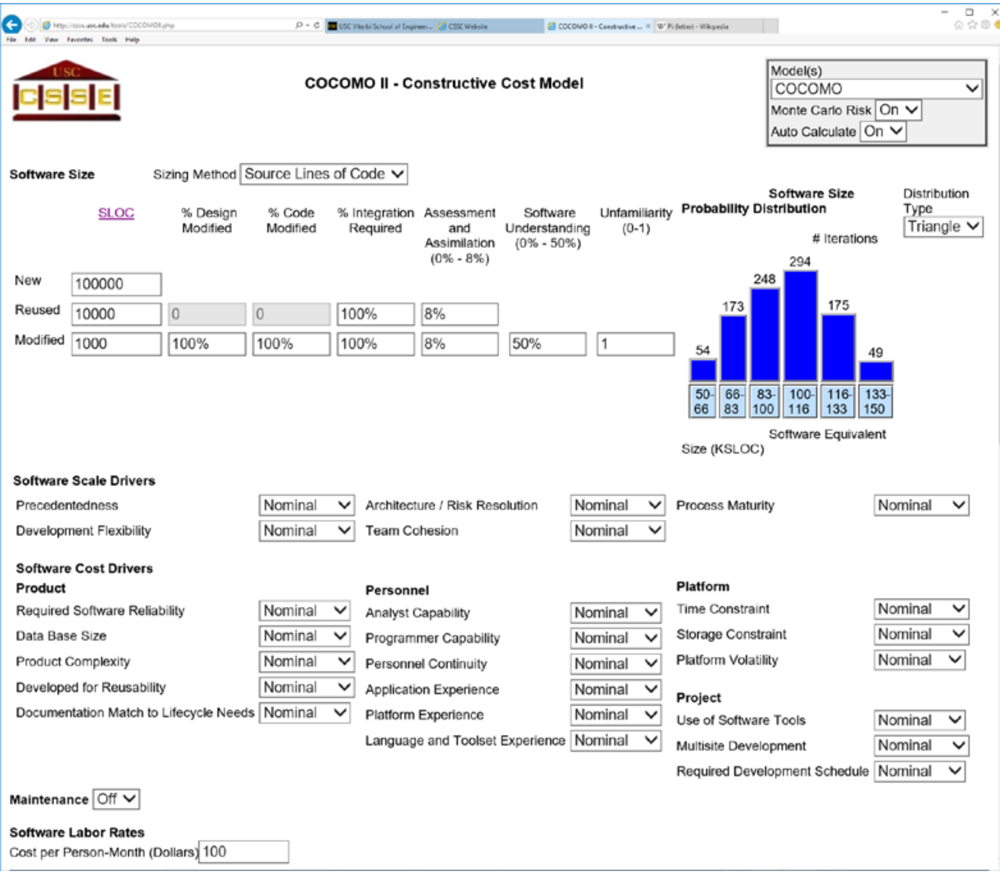
COCOMO II

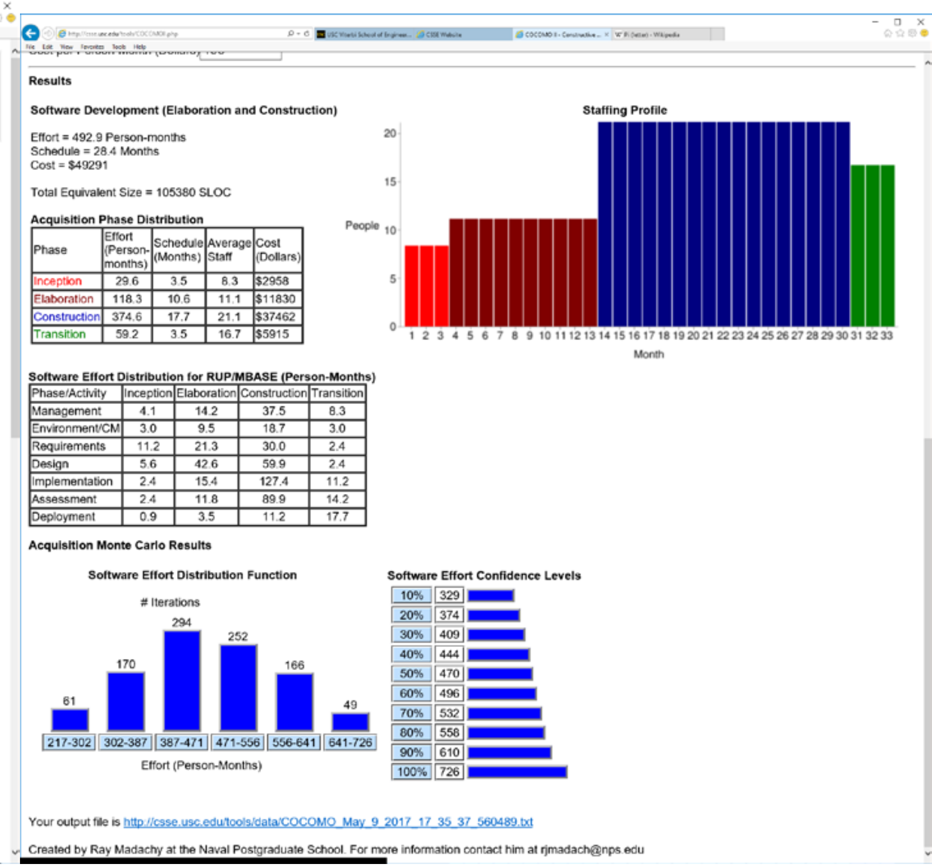


- Function Points depend on equivalent language (i.e. function points leveled to common source language equivalent…many options)
- 1 Quantitative Input (not counting $$/person/month)
- 22 Qualitative Inputs: 5 for size scale, 17 for cost drivers (not counting language)

http://csse.usc.edu/tools/COCOMOII.php

COCOMO II EXAMPLE

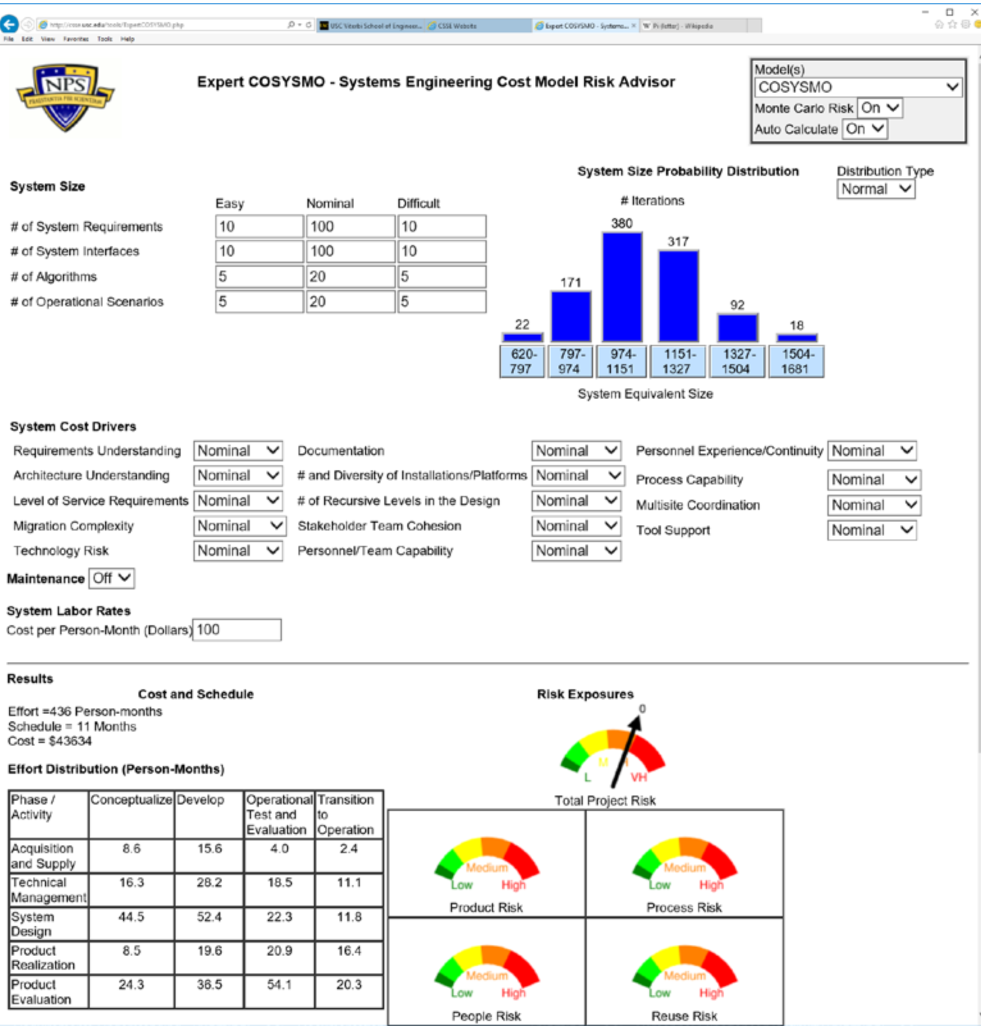


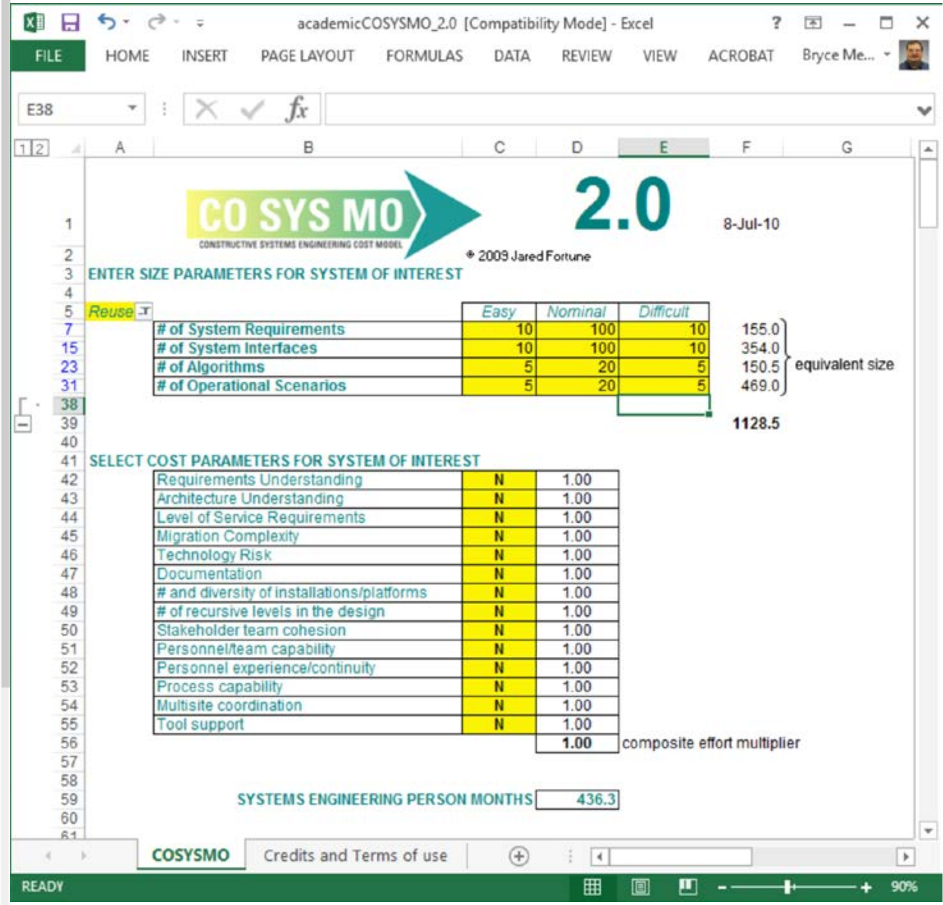


Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	29.6	3.5	8.3	$2958
Elaboration	118.3	10.6	11.1	$11830
Construction	374.6	17.7	21.1	$37462
Transition	59.2	3.5	16.7	$5915

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	4.1	14.2	37.5	8.3
Environment/CM	3.0	9.5	18.7	3.0
Requirements	11.2	21.3	30.0	2.4
Design	5.6	42.6	59.9	2.4
Implementation	2.4	15.4	127.4	11.2
Assessment	2.4	11.8	89.9	14.2
Deployment	0.9	3.5	11.2	17.7

COSYSMO





http://csse.usc.edu/tools/ExpertCOSYSMO.php http://cosysmo.mit.edu/downloads/

SEER-SEM (SEER for software) (COMMERCIAL)

- Developed by Dan Galorath in 1988. Commercial Product of Galorath inc.
- Emerged from efforts in the cost community in the 1970s’-1980’s (cross flow between this effort and similar work at TRW/USC, etc.)
- Incorporates decomposition then estimation by a family of models, but are linked (as in other parametric models) to software size.
- Has a long series of questions hierarchal questions to probe various risk areas.
- Can estimate with initial answers, then probe deeper, but requires extensive data to be effective.
- Links to other software packages (MS Office + Project, Oracle, etc.)

https://en.wikipedia.org/wiki/SEER-SEM

http://galorath.com/products/software/extended-capabilities-seer-sem
http://galorath.com/products/software/SEER-Software-Cost-Estimation

PRICE TruePlanning (COMMERCIAL)

- Programmed Review of Information for Costing and Evaluation (PRICE).
- Commercial Tool Set, one of the first parametric models
  - From RCA, then Lockheed-Martin, David Shore, Frank Freiman and William Rapp 1970s. Spun out as PRICE.
- System Engineering and Software Cost Estimation Toolbox
  - WBS System Engineering approach meshed with various proprietary parametric and interpolation methods using sizing and complexity values at levels for available data. System is broken down, and questions are asked for each item.
  - Uses a knowledge base and assumptions based on provided data.
  - Sensitivity: Like other WBS models, requires a fairly detailed design and refined requirements, and a heavier data requirement. Less data, more inaccurate (due to inaccurate complexity and size).
    - THE LESS A PROJECT MATCHES THE HISTORICAL DATA, THE LESS ACCURATE THE ESTIMATE.

For a system:
https://www.nasa.gov/sites/default/files/files/44_JPL_Calibration_Process_08102015_final_CAD.pdf

https://en.wikipedia.org/wiki/PRICE_Systems
https://en.wikipedia.org/wiki/Software_development_effort_estimation#Development_estimation_software
http://www.pricesystems.com/trueplanning-framework

CONCLUSION

- Three Things We Need to Estimate Well:
  - Cost
  - Schedule
  - Performance
- Accuracy of Estimate = Accuracy and Availability of Data
  - From Software Project
  - From Organization
  - From History
  - How close does it match?
- Cost Estimation is great for determining how well your team is doing software quality!

A Few References

- Jackson,. Meyer, Wessel, “Methodology for the Cost Benefit Analysis of a Large Scale Multi-phasic Software Enterprise Migration”, Software Solutions Symposium 2017, http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=495776
- COSYSMO at http://cosysmo.mit.edu/
- COCOMO II at http://csse.usc.edu/csse/tools/

More Links/References

- https://en.wikipedia.org/wiki/Cost_estimation_in_software_engineering
- **Quantifying Uncertainty for Early Lifecycle Cost Estimation (QUELCE),** CMU/SEI-2011-TR-026
- http://csse.usc.edu/TECHRPTS/2008/usc-csse-2008-836/usc-csse-2008-836.pdf
- https://en.wikipedia.org/wiki/Software_development_effort_estimation
- http://www.psmsc.com/UG2010/Presentations/p11-Packard-Improving%20ERP%20Estimation%20in%20the%20DOD%20(FINAL).pdf
- **COTS Integration Estimation: Enterprise Resource Planning Systems, Dr. Wilson Rosa, James Bilbro , USC CSSE Annual Research Review 2012, March 6, 2012,** http://csse.usc.edu/csse/event/2012/ARR/presentations/COTS%20Integration%20Estimation-ERP.pdf
- Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*. McGraw-Hill Osborne Media, April, 2008.
- ***The Failure of Risk Management: Why It's Broken and How to Fix It, ISBN:*** 978-0470387955, 2009, Douglas Hubbard
- **How to Measure Anything: Finding the Value of Intangibles in Business, 2008,** Douglas Hubbard
- SEER SEM:
  - https://en.wikipedia.org/wiki/SEER-SEM
  - http://galorath.com/products/software/extended-capabilities-seer-sem
  - http://galorath.com/products/software/SEER-Software-Cost-Estimation

More links

- http://sunset.usc.edu/csse/research/cocomoii/cocomo_main.html
- http://csse.usc.edu/tools/COCOMOII.php
- http://cosysmo.mit.edu/downloads
- http://csse.usc.edu/tools/ExpertCOSYSMO.php
- http://csse.usc.edu/csse/research/COCOTS/modeldesc.html
- http://galorath.com/products/software/extended-capabilities-seer-sem
- https://en.wikipedia.org/wiki/SEER-SEM
- http://galorath.com/products/software/SEER-Software-Cost-Estimation
- http://www.spminfoblog.com/post/141/slim---a-mathematical-model/

COCOMO II Model Mechanics

A function converts SLOC and FP inputs to ESLOC for *Size*
(See Equivalent Size in outputs)

PM is Person Months (i.e. Effort)

$$PM = a * Size^{E} * \prod_{i=1}^{17} EM_{i}$$

a is a constant built into the model

where

$$E = B + 0.01 * \sum_{j=1}^{5} SF_{j}$$

B is a constant built into the model

Software Cost Drivers (Feed ***EM***)
Required Software Reliability
Data Base Size
Product Complexity
Developed for Reusability
Documentation Match to Lifecycle Needs
Analyst Capability
Programmer Capability
Personnel Continuity
Application Experience
Platform Experience
Language and Toolset Experience
Time Constraint
Storage Constraint
Platform Volatility
Use of Software Tools
Multisite Development
Required Development Schedule

Software Scale Drivers (feed ***SF***)
Precedentedness
Development Flexibility
Architecture / Risk Resolution
Team Cohesion
Process Maturity

Software Cost Drivers and Software Scale Drivers are multiplied using a knowledge matrix from hundreds of software efforts

http://sunset.usc.edu/csse/research/cocomoii/cocomo_main.html

COSYSMO Engine Mapping

Numeric Inputs	Easy	Nominal	Difficult
# of System Requirements	##	##	##
# of System Interfaces	##	##	##
# of Algorithms	##	##	##
# of Operational Scenarios	##	##	##

PM is Person Months (i.e. Effort)

A, w are embedded in model (w is a matrix from initially 44 projects)

r is the breakout of Numeric inputs into the 6 reuse categories, *if Reuse is used.*

$$PM_{NS} = A \cdot \left[\sum_k \left(\sum_r w_r (w_{e,k}\Phi_{e,k} + w_{n,k}\Phi_{n,k} + w_{d,k}\Phi_{d,k}) \right) \right]^E \cdot \prod_{j=1}^{14} EM_j$$

PM_{NS} = effort in Person Months (Nominal Schedule)

A = calibration constant derived from historical project data

k = {Requirements, Interfaces, Algorithms, Scenarios}

w_x = weight for "easy", "nominal", or "difficult" size driver

r = {New, Design for Reuse, Modified, Deleted, Adopted, Managed}

w_r = weight for reuse category

Φ_x = quantity of "k" size driver

E = represents (dis)economies of scale

EM = effort multiplier for the j^{th} cost driver.

E is embedded in model (from initially 44 projects) ~1.06

Qualitative Inputs	Scale (*=risky side)
Requirements Understanding	***Very Low** To Very High
Architecture Understanding	***Very Low** To Very High
Level of Service Requirements	Very Low To **Very High***
Migration Complexity	***Extra High** To Nominal
Technology Risk	***Extra High** To Very Low
Documentation	Very Low To **Very High***
# and Diversity of Installations/Platforms	***Extra High** To Nominal
# of Recursive Levels in the Design	Very Low To **Very High***
Stakeholder Team Cohesion	***Very Low** To Very High
Personnel/Team Capability	***Very Low** To Very High
Personnel Experience/Continuity	***Very Low** To Very High
Process Capability	***Very Low** To Extra High
Multisite Coordination	***Very Low** To Extra High
Tool Support	***Very Low** To Very High

Equations from:
ESTIMATING SYSTEMS ENGINEERING REUSE WITH THE CONSTRUCTIVE SYSTEMS ENGINEERING COST MODEL (COSYSMO 2.0) by Jared Fortune, USC, 2009
And
COSYSMO 2.0: A Cost Model and Framework for Systems Engineering Reuse (Jared Fortune and Ricardo Valerdi), *2009 COCOMO Forum, Massachusetts Institute of Technology*
And
Valerdi, R. (2005). The Constructive Systems Engineering Cost Model. Ph.D. Dissertation. University of Southern California. Los Angeles, CA.