COMP 5700/6700/6706
Software Process

Spring 2016

David Umphress

Estimation

- Lesson: Estimation
- Strategic Outcomes:
  - To lay a foundation for estimating end-product size and effort
  - To learn the mechanics of estimating by component
- Tactical Outcomes:
  - To know common size estimating approaches in use today
  - To comprehend the use of components in estimation
  - To comprehend how to map size to effort
  - To apply an estimation technique to a sample problem

- Support material:
  - "Estimation" Stellman, A. and J. Greene. 2005. *Applied Software Project Management. O'Reilly.*
  - Fingers in the Air: A Gentle Introduction to Software Estimation. Asproni, G. 2005. *Methods and Tools, Winter 2005*. pp 2-13
  - Fundamentals of Function Point Analysis. Longstreet, D. Downloaded Jan 09. www.softwaremetrics.com.
- Instant take-aways:
  - High-level coverage of common size estimation approaches
  - Component-based estimation

- Bookshelf items
  - McConnell, S. 2006. Software Estimation: Demystifying the Black Art. Microsoft Press
  - Humphrey, W. 1995. *A Discipline for Software Engineering*. Addison Wesley.
  - Boehm, B.W., 1981. *Software Engineering Economics*. Prentice-Hall

Syllabus

- Software engineering raison d'être
- Process foundations
- Common process elements
- Construction
- Reviews
- Refactoring
- Analysis
- Architecture
- Estimation
- Scheduling
- Integration
- Repatterning
- Measurements
- Process redux
- Process descriptions*
- Infrastructure*
- Retrospective

Estimation:

- **Rationale**
- **Principles of estimation**
- **Common techniques**
- **Component-based estimation**
  - **process**
    - **size**
    - **effort**
    - **confidence**
  - **setup**

COMP5700/6700/6706 Goal Process

Minimal Guiding Indicators

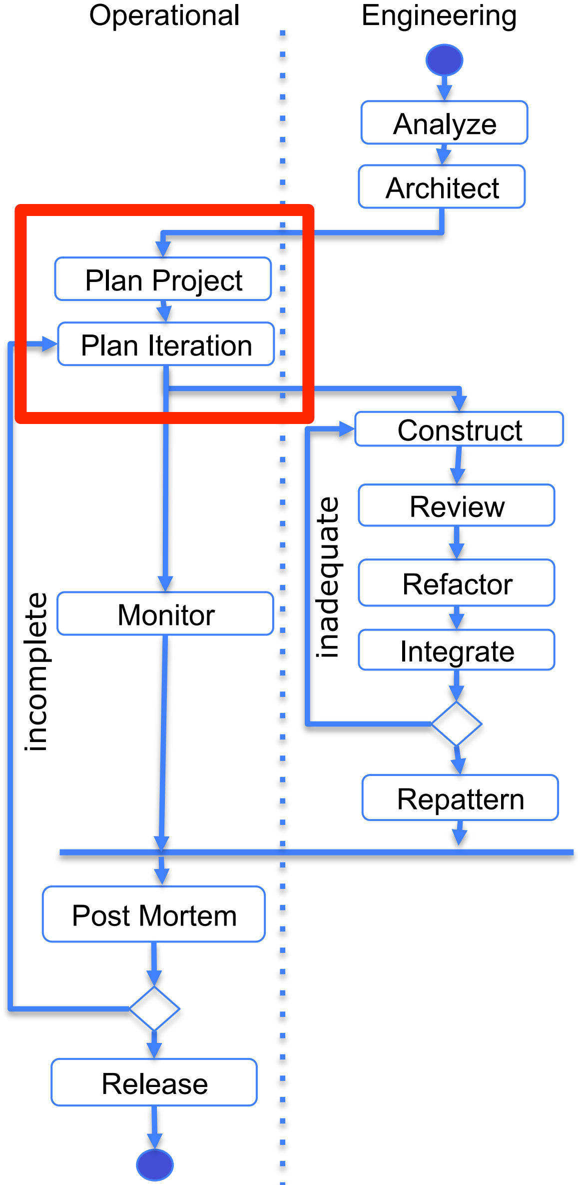
Goal	Indicator
Cost:	None
Schedule:	PV/EV >.75
Performance:	
Product:	
NFR:	none
FR:	100% BVA
Process:	pain < value

Minimal Sufficient Activities

Engineering Activities
- Envision
  - Analyze
- Synthesize
  - Architect
- Articulate
  - Construct
  - Refactor
- Interpret
  - Review
  - Integrate
  - Repattern

Operational Activities
- Plan
  - Plan project
  - Plan iteration
- Monitor
- Release

Minimal Viable Process



Minimal Effective Practice

MSA	MEP
Analyze	Scenarios
Architect	CRC
Plan Project	Component-based estimation
Plan Iteration	Component-iteration map
Construct	TDD
Review	Review checklist Test code coverage
Refactor	Ad hoc sniffing
Integrate	Ad hoc
Repattern	Ad hoc
Monitor	Time log Change log Burndown
Post Mortem	PV/EV
Release	Eclipse zip spreadsheets

How do you estimate effort?

Rationale

- Why do we need to estimate effort?
  - Effort estimate ~ defect rate
    - can predict quality of product, testing effort
  - Effort estimate ~ resources
    - can forecast hardware/software/peopleware needed
  - Effort estimate ~ cost
    - can predict bucks
  - Effort estimate ~ assessment mechanism
    - can measure progress using estimated vs actual
  - Effort estimate ~ buy-in
    - can use estimating efforts to gain worker/management buy-in

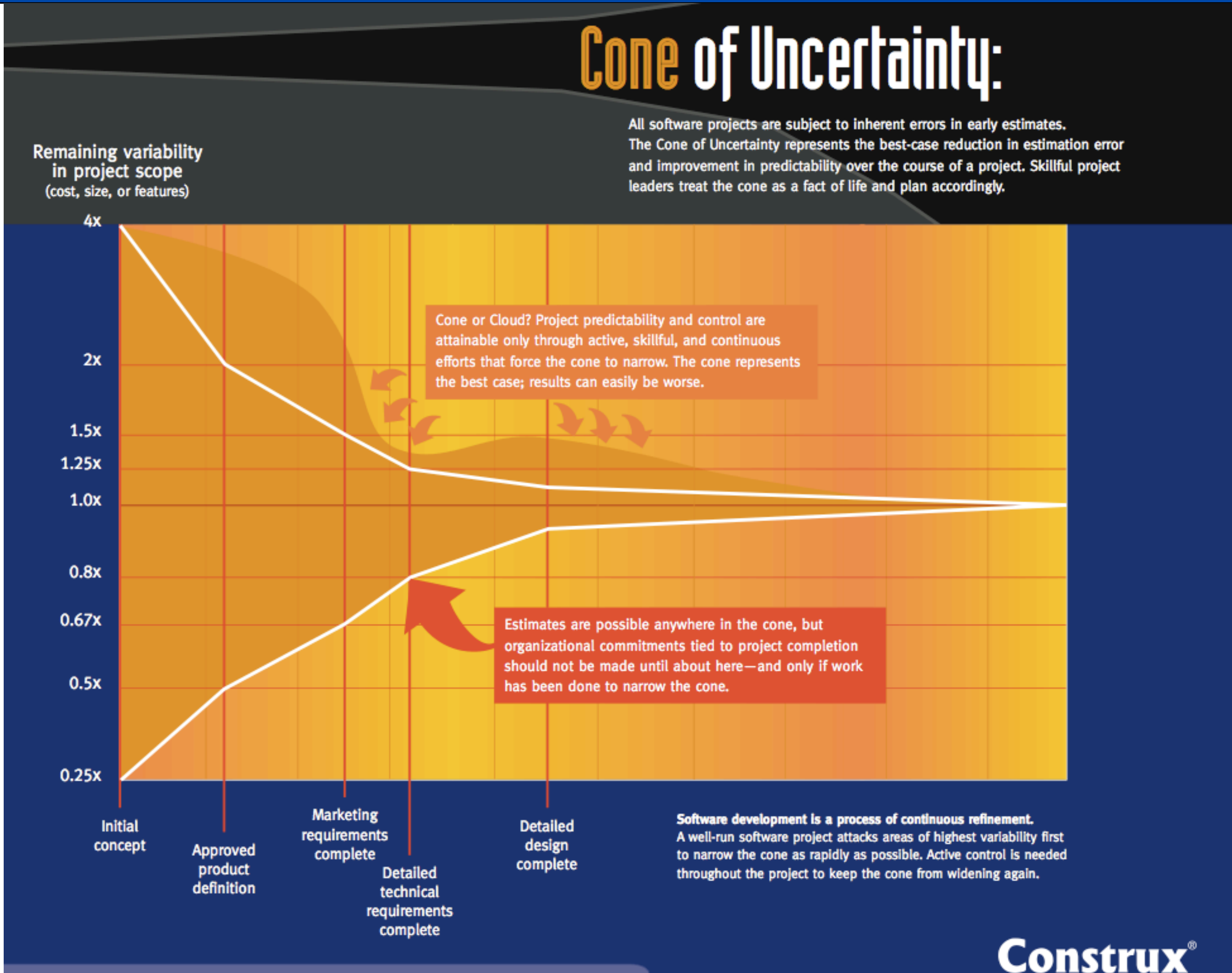
Rationale

- What makes us think we can?
  - Estimating models in other fields
    - large base of history
    - in wide use
    - generate detailed planning data
  - Software estimating experience
    - 100% + errors are normal
    - few developers make estimates
    - fewer still use orderly methods
  - But ...
    - empirical studies show relationship between size and resources
    - we must choose and appropriate method
      - based on empirical evidence within organization
      - may (will?) differ from place to place, culture to culture

A Good Estimate is

- "… an estimate that provides a clear enough view of the project reality to allow the project leadership to make good decisions about how to control the project to hit its targets."

Steve McConnell



Cone of Uncertainty:
All software projects are subject to inherent errors in early estimates.
The Cone of Uncertainty represents the best-case reduction in estimation error and improvement in predictability over the course of a project. Skillful project leaders treat the cone as a fact of life and plan accordingly.
Remaining variability in project scope
(cost, size, or features)
4x
2x
1.5x
1.25x
1.0x
0.8x
0.67x
0.5x
0.25x
Cone or Cloud? Project predictability and control are attainable only through active, skillful, and continuous efforts that force the cone to narrow. The cone represents the best case; results can easily be worse.
Estimates are possible anywhere in the cone, but organizational commitments tied to project completion should not be made until about here—and only if work has been done to narrow the cone.
Initial concept
Approved product definition
Marketing requirements complete
Detailed technical requirements complete
Detailed design complete
Software development is a process of continuous refinement.
A well-run software project attacks areas of highest variability first to narrow the cone as rapidly as possible. Active control is needed throughout the project to keep the cone from widening again.
Construx®

Effort Estimation

- Foundation
  - Planning process requires us to gauge the magnitude of work.
  - Our approach will be to
    - identify components, then
    - estimate size, then
    - estimate effort
  - Our goal in do so
    - measure functionality delivered to customer
    - provide a measure of product quantity understood by users and developers
    - provide a measure that is available early in the product life-cycle
    - serve as a metric for project planning, project control, process improvement
  - But, must decide what “size” means

[Humphrey 1995]

Size

- Common measures of size
  - physical:
    - LOC
    - normalized LOC
    - screens
    - pages
    - scripts
    - reports
  - function:
    - function points
    - feature points

Common Estimation Techniques

- By guessing
- By expert opinion
  - correlate estimates of size from multiple sources
  - e.g., Wideband-delphi
- By function
  - predict by examining features
  - e.g., Function points
- By analogy
  - compare to similar system of known size
  - e.g., Fuzzy-logic method
- By decomposition
  - divide and conquer
  - e.g., Standard component method
- By parametric models
  - mathematically model historical data
  - e.g., Proxy estimation

By guessing

- RHE

By expert opinion

- Background
  - Uses several estimators
  - “size estimation by consensus”
- Method: Wideband-Delphi
  - multiple estimators make independent estimate, submit to coordinator
  - coordinator calculates average estimate, indicates (anonymously) other estimates
  - repeat until estimates stabilize
  - size estimate = average estimate; variation = range of original estimates

Wideband-Delphi

- E.g.
  - Round 1
    - Estimates:
      - A - 13,800 LOC
      - B - 15,700 LOC
      - C - 21,000 LOC
    - Coordinator calculates average of 16,833 LOC and resubmits
  - Round 2
    - Estimators then meet and discuss the estimates. New estimates:
      - A - 18,500 LOC
      - B - 19,500 LOC
      - C - 20,000 LOC
    - Coordinator calculates average of 19,333 LOC and resubmits
  - Round 3
    - Estimators agree on Round 2 figures

Wideband-Delphi

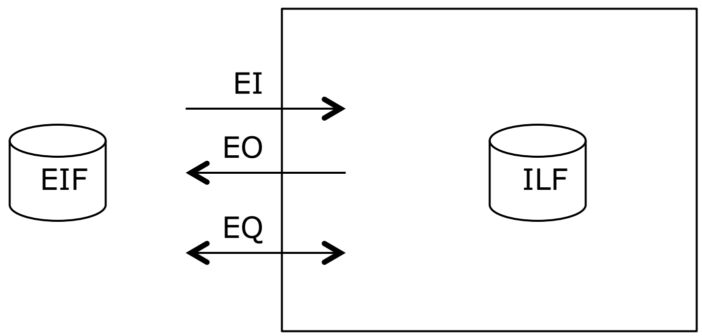
- Agile equivalent: Planning Poker

Wideband-Delphi

- Perspective
  - pros
    - can produce very accurate results
    - utilizes organization’s skills
    - can work for any sized product
  - cons
    - relies on experts
    - is time consuming
    - is subject to common biases

By function

- Background
  - function point is an arbitrary unit of “functionality”
  - function point analysis seeks to assess a measure of functionality by examining interfaces to system
  - feature point analysis is fp + algorithmic analysis
- Method: function point estimation
  - determine project boundary
  - count files (data)
    - internal logical files
    - external interface files
  - count transactions
    - external inputs
    - external outputs
    - external inquiries
  - determine system complexity factor



Function Point Estimation

CHECKLIST FOR IDENTIFYING INTERNAL LOGICAL FILES (ILFs)				
Data Group:				
Definitions: An Internal Logical File (ILF) is a *user identifiable* group of logically related data or *control information* maintained within the boundary of the application. *User identifiable* refers to the specific user requirements that an experienced user would define. *Control information* is data used by the application to ensure compliance with business function requirements specified by the user. *Maintained* refers to the ability to modify data through an *elementary process*. An *elementary process* is the smallest unit of activity that is meaningful to the end-user in the business (e.g., add, change, delete).				
Rules:	**Yes**	**No**	**N/A**	**Remarks**
Is the group of data or control information a logical, or user identifiable, group of data that fulfils specific user requirements?				
Is the group of data maintained within the application boundary?				
Is the group of data modified, or maintained, through an elementary process of the application?				

Example ILFs:

- logical file of error messages
- logical file of audit information
- logical file of application data

Not ILFs:

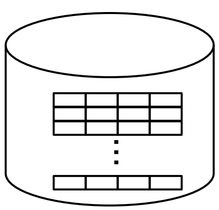
- temporary/working files
- data for normal backup and recovery

[it.toolbox.com]

Function Point Estimation

- Sample counting: Internal logical files
  - Quantifying functionality to service input file

Number of Record Element Types	Number of Data Element Types: 1-19	20-50	51+
1	7fp	7fp	10fp
2 to 5	7fp	10fp	15fp
6+	10fp	15fp	15fp



1 record type
4 data element types
= 7fp

[it.toolbox.com]

Function Point Estimation

- E.g.,
  - New program has the following requirements
    - analyze marketing performance
    - project the likely sales
    - allocate sales to regions and time periods
    - produce monthly report
  - Raw function points estimate is 54

System Components		Complexity			Total
		Low	Average	High	
Inputs	EI	3	4	6	
Outputs	EO	4	5	7	
Inquiries	EQ	3	4	6	
Files	ILF	7	10	15	
Interfaces	EIF	5	7	10	

Function Point Estimation

- E.g., (con't)
  - function point estimate adjusted for environment is
    adjusted count = raw x (system char x .01 + .65)

General System Characteristics			
Data communications	**2**	On-line update	0
Distributed functions	0	Complex processing	0
Performance	0	Reusability	**3**
Heavily-used configuration	0	Installation ease	0
Transaction rate	0	Operational ease	0
On-line data entry	**4**	Multiple sites	0
End user efficiency	0	Facilitate change	0
		Total	**9**

Not present = 0
Incident influence = 1
Moderate influence = 2
Average influence = 3
Significant influence = 4
Strong influence = 5

adjusted count =
54 x (9 x .01 + .65)
= 54 x .74
= 39.96
= 40 fp

[IFPUG 1990]

Function Point Estimation

- Perspective
  - pros
    - usable in the earliest requirements phases
    - independent (sorta) of programming language, product design, or development style
    - large body of historical data
    - well documented method
    - active users group
  - cons
    - automated counting not always possible
    - do not reflect language, design, or style differences
    - designed for estimating commercial data processing applications

By analogy

- Method: fuzzy-logic size estimation
  - Gather size data on previously developed programs
  - Subdivide data into size categories:
    - very large, large, medium, small, very small
    - establish size ranges
    - include all existing and expected products
  - Subdivide each range into subcategories
  - Allocate the available data to the categories
  - Establish subcategory size ranges
  - When estimating a new program, judge which category and subcategory it most closely resembles

Fuzzy-Logic Size Estimation

- E.g.
  - given 5 programs:
    - a file utility of 1,844 LOC
    - a file management program of 5,834 LOC
    - a personnel record keeping program of 6,845 LOC
    - a report generating package of 18,386 LOC
    - an inventory management program of 25,943 LOC
  - establish 5 size ranges VS, S, M, L, VL:

Range	**Low**	**Midpoint**	**High**	**Representative System**
Very Small	1,325	1,946	2,566	file utility
Small	2,566	3,768	4,970	no members
Medium	4,970	9,298	9,626	file mgmt; personnel records sys
Large	9,626	14,134	18,641	report generator
Very Large	18,641	27,373	36,104	inventory mgmt

Fuzzy-Logic Size Estimation

- E.g. (con’t)
  - In comparing the new program to the historical data you observe
    - more complex application than file management or personnel programs
    - not as complex as the inventory management program
    - more function than the report package
  - You conclude that the new program is in the lower end of “very large,” or from 18 to 26 KLOC.

Fuzzy-Logic Size Estimation

- Perspective
  - pros
    - based on relevant historical data
    - easy to use
    - requires no special tools or training
    - provides reasonably good estimates where new work is similar to prior experience
  - cons
    - requires familiarity with historical data
    - not useful for new program types
    - not useful for programs much larger or smaller than historical data

By decomposition

- Background
  - estimates size based on principal units of decomposition
- Method: standard component sizing
  - identify common building blocks (subsystems, modules, screens, etc.)
  - associate sizes with building blocks
  - decompose new development into blocks and compare with history
- E.g.
  - assume a historical database of
    - data input component 1,108 LOC
    - output component 675 LOC
    - file component 1,585 LOC
    - control component 2,550 LOC
    - computation component 475 LOC

Standard Component Sizing

- E.g. (con't)
  - estimate of maximum, minimum, and likely numbers of components of new product
    - data input component 1 4 7
    - output component 1 3 5
    - file component 2 4 8
    - control component 1 2 3
    - computation component 1 3 7
  - calculate likely code sizes based on history
    - data input component 1108 4432 7756
    - output component 675 2025 3375
    - file component 3170 6340 12680
    - control component 2550 5100 7650
    - computation component 475 1425 3325

Standard Component Sizing

- E.g. (con't)
  - calculate the minimum, likely, and maximum LOC of the new product
    - minimum 7,978
    - likely 19,322
    - maximum 34,786
  - distribute the size*
    - LOC = (7978 + 4*19322 + 34786)/6 = 20,009 LOC
    - the standard deviation is (34786-7978)/6=4468
    - the estimate range is 20,009+4468 = 15,500 to 24,500 LOC

*(low + 4*mid + high)/6 gives Gaussian distribution with a standard deviation of (high - low)/6

[Humphrey 1995]

Standard Component Sizing

- Perspective
  - pros
    - based on relevant historical data
    - easy to use
    - requires no special tools or training
    - provides a rough estimate range
  - cons
    - decomposition must mirror end product to be accurate
    - must use large components early in a project

[Humphrey 1995]

By parametric model

- Background
  - use statistical model of past estimation efforts to predict size
- Method: Component-based estimation
  - Concept
    - estimators rarely think in detail at project beginning
    - use two-phase approach to estimating:
      - identify major components, then
      - determine size/duration based on components, then
      - adjust according to past performance
  - Component characteristics
    - should correlate closely to development costs
    - should be easy to visualize early in development
    - should be countable by automated means

[derived from Humphrey 1995]

Component-Based Estimation

- Method
  - begin
    identify components from architecture
    forecast effort based on historical components
    adjust effort via stat model of past projects
    end

Principles of Size Estimation

- General principles
  - repeatability
    - separate estimation efforts must have same result $\pm$ delta
  - consistency
    - must be performed and used the same way across the organization
  - empiricism
    - must have measure of estimation error*

What makes an estimation technique "good"?

*What is a reasonable bound on estimation error?

A measure of "goodness":

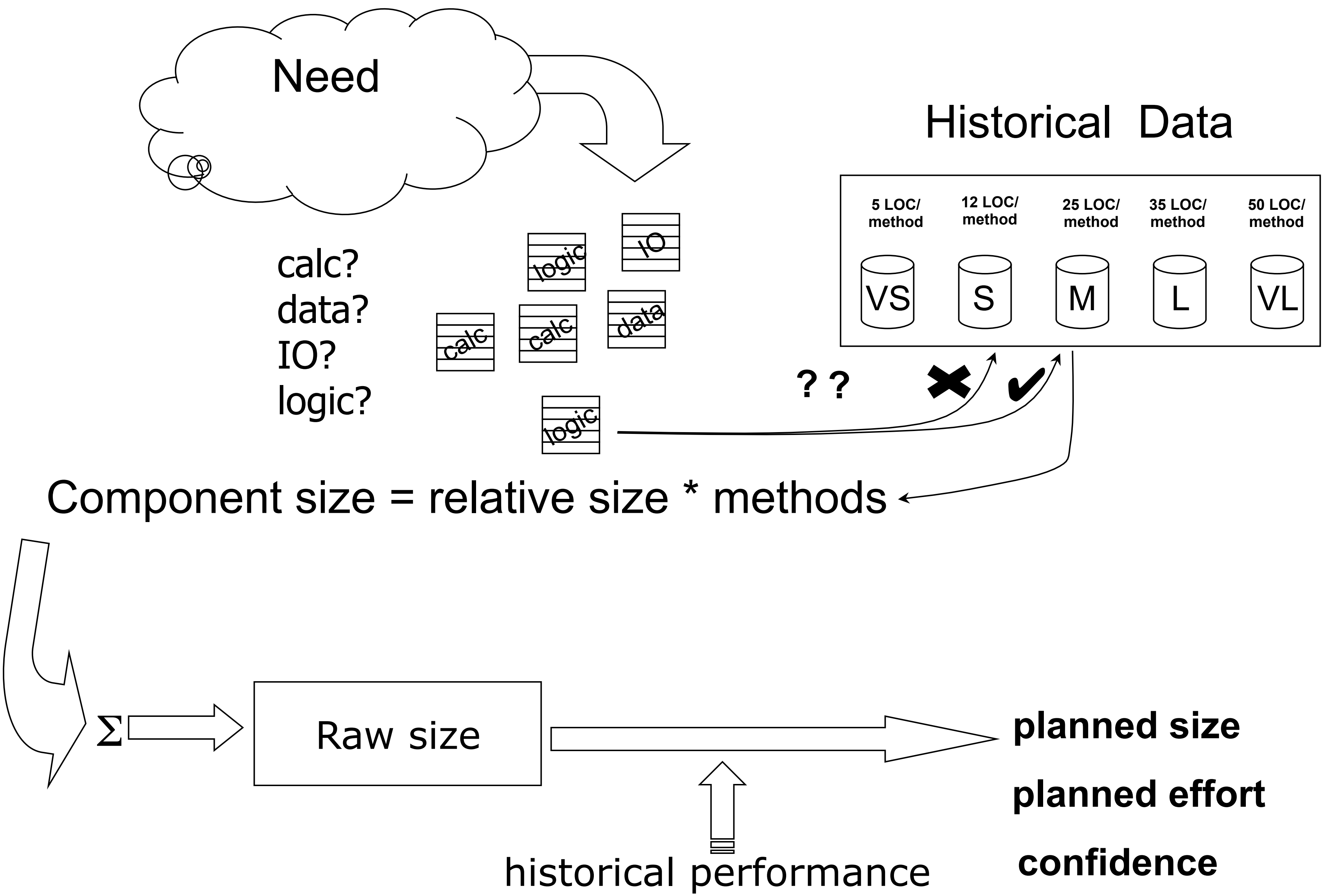
- **V is actual**
- **V_i is estimate for project at time i**
- **n is the number of estimating points**
- **the lower the value of ε, the better the estimation process**
- **high ε's across projects indicate an estimation problem**

$$\varepsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{(V_i - V)}{V}\right)^2}$$

Let’s Pick ...

- By RHE
- By expert opinion
- By function
- By analogy
- By decomposition
- By component

... this one



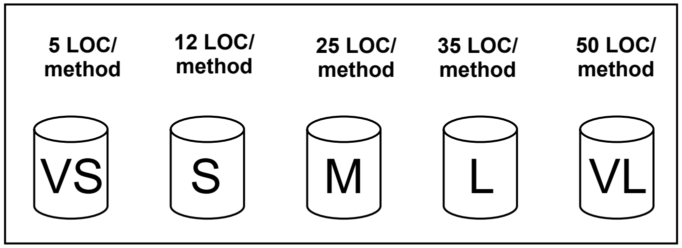
Need
calc?
data?
IO?
logic?
logic
IO
calc
calc
data
logic
Historical Data
5 LOC/ method
12 LOC/ method
25 LOC/ method
35 LOC/ method
50 LOC/ method
VS
S
M
L
VL
? ?
Component size = relative size * methods
Σ
Raw size
planned size
planned effort
confidence
historical performance

Component-Based Estimation Setup

- Need the following information to estimate size:
  - Data on historical objects
    - Actual lines of code (LOCa)
  - Data on historical projects:
    - Raw lines (LOCr)
    - Actual lines (LOCa)
    - Actual effort (Ea)

1. Putting stuff in buckets

Historical Data



Building relative size matrix

- Goal: to determine bucket sizes
- To calculate size ranges
  - determine ln of LOC per method for each component
  - calculate the mean and standard deviation
  - distribute sizes as*

Size	Lower	Midpoint	Upper
VS	1	$e^{mean-2*std}$	$e^{mean-1.5*std}$
S	$e^{mean-1.5*std}$	$e^{mean-std}$	$e^{mean-0.5*std}$
M	$e^{mean-0.5*std}$	e^{mean}	$e^{mean+0.5*std}$
L	$e^{mean+0.5*std}$	$e^{mean+std}$	$e^{mean+1.5*std}$
VL	$e^{mean+1.5*std}$	$e^{mean+2*std}$	big

* All values are rounded up to nearest integer

Size matrix (con't)

- Example
  - given the following components
    - 3 methods, 39 total LOC
    - 5 methods, 127 total LOC
    - 2 methods, 64 total LOC
    - 3 methods, 28 total LOC
    - 1 method, 23 LOC
    - 2 methods, 44 total LOC
  - LOC per method =
    13, 25.4, 32, 9.3, 23, 22
  - ln of LOC per method =
    2.56, 3.23, 3.46, 2.23,
    3.13, 3.09;
    mean = 2.954; std = 0.461

	Lower	Mid	Upper
VS	1	8	10
S	10	12	15
M	15	19	24
L	24	30	38
VL	38	48	big

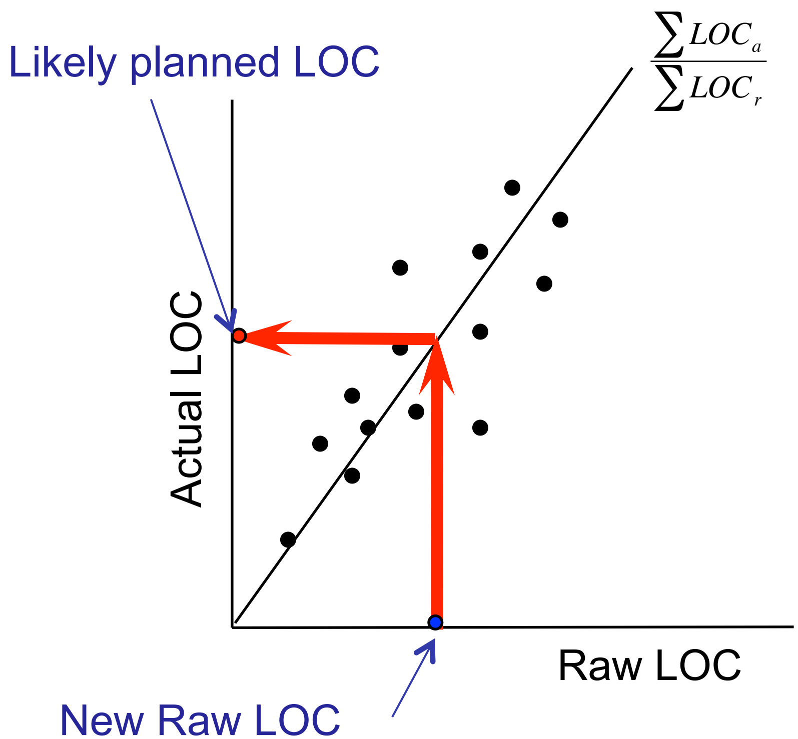
$e^{(mean-0.5*std)}$
$=e^{(2.954-0.5*0.461)}$

$e^{(mean)}=e^{2.954}$

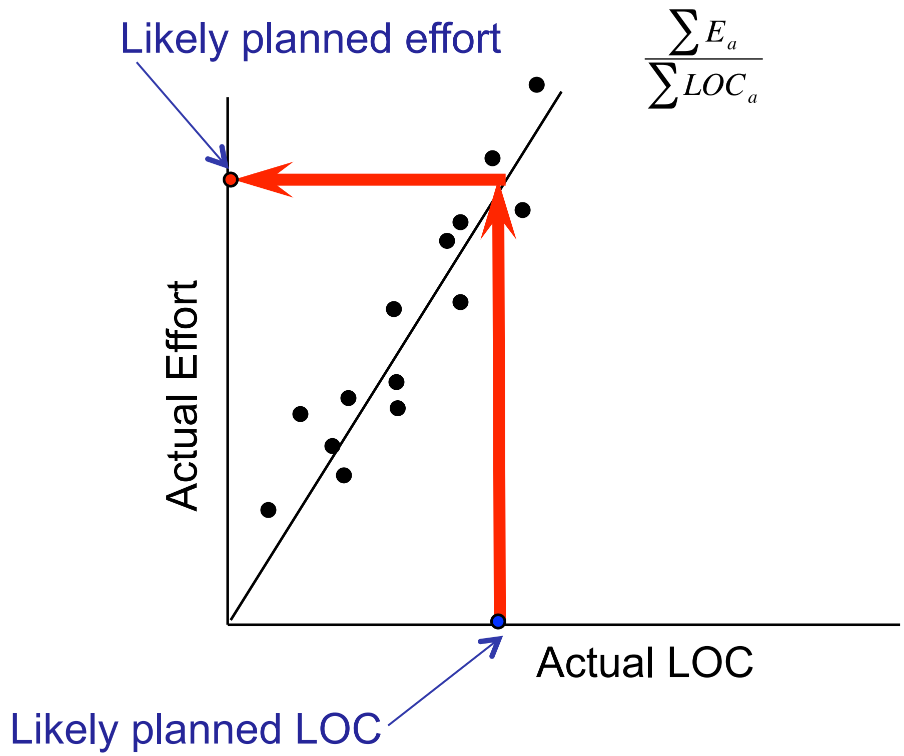
$e^{(mean+0.5*std)}$
$=e^{(2.954+0.5*0.461)}$

2. Calculating historical trend

Estimation fundamentals



$$\left[NewLOC_r \times \frac{\sum LOC_a}{\sum LOC_r} \right] = NewLOC_p$$



$$\left[NewLOC_p \times \frac{\sum E_a}{\sum LOC_a} \right] = NewE_p$$

Historical trends

- Example
  - Given the historical database:

LOCr	LOCa	Ea
229	240	1100
382	255	1020
215	270	900
336	240	760
316	227	800
316	322	1200
133	120	260
433	412	1500
266	170	500
296	353	1700

$$\frac{\sum LOC_a}{\sum LOC_r} = .89$$

If the raw size is 342 LOC, then the planned size is ceil(342*.89)=306

$$\frac{\sum E_a}{\sum LOC_a} = 3.73$$

and the planned effort is ceil(306*3.73)=1142

3. Calculating confidence

Confidence

- Prediction interval
  - ... provides a likely range around the estimate
  - Note: Humphrey suggests determining linear regression coefficients, then calculating statistical range:

$$Range = t(\alpha/2, n-2)\sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2}\sqrt{1 + \frac{1}{n} + \frac{(x_k - x_{avg})^2}{\sum_{i=1}^{n}(x_i - x_{avg})^2}}$$

    - My opinion: this is statistical overkill
    - We will take a different, although Humphrey-compatible approach

Confidence

- Prediction interval … our approach
  - ... involves calculation of lower prediction interval (LPI) and upper prediction interval (UPI) …
    - for time estimates
      - LPI = Estimate * fastest historical productivity
      - UPI = Estimate * slowest historical productivity
- Confidence on prediction interval
  - … involves calculating correlation coefficient
    - $1.0 \geq r^2 \geq .75$ high
    - $.75 > r^2 \geq .50$ med
    - $.50 > r^2$ low

Confidence

- Size prediction interval

LOCr=342
LOCp=306

LOCr	LOCa	Ea
229	240	1100
382	255	1020
215	270	900
336	240	760
316	227	800
316	322	1200
133	120	260
433	412	1500
266	170	500
296	353	1700

240/229

LOCa/LOCr
1.05
0.67
1.26
0.71
0.72
1.02
0.90
0.95
0.64
1.19

worst underestimate 342*1.26= **UPI=431**

worst overestimate 342*0.64= **LPI=219**

Note:
set UPI to LOCp if UPI < LOCp
set LPI to 1 if LPI > LOCp
also set confidence to low

Confidence

- Correlation on predication interval
  - calculate correlation coefficient

LOCr	LOCa	Ea
229	240	1100
382	255	1020
215	270	900
336	240	760
316	227	800
316	322	1200
133	120	260
433	412	1500
266	170	500
296	353	1700

$$r(x,y)^2 = \left(\frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2 \right]}} \right)^2$$

= 0.52

= medium confidence

$1.0 \geq r^2 \geq .75$	high
$.75 > r^2 \geq .50$	med
$.50 > r^2$	low

Confidence

- Time prediction interval - 1

LOCr=342
LOCp=306
LPI=219
UPI=431

LOCr	LOCa	Ea
229	240	1100
382	255	1020
215	270	900
336	240	760
316	227	800
316	322	1200
133	120	260
433	412	1500
266	170	500
296	353	1700

1100/240

Ea/LOC
4.58
4.00
3.33
3.17
3.52
3.73
2.17
3.64
2.94
4.82

min
306*2.17

max
306*4.82

Ep=1142

LPI=665

UPI=1475

Note:
set UPI to Ep if UPI < Ep
set LPI to 1 if LPI > Ep
also set confidence to low

Confidence

- Correlation on predication interval
  - calculate correlation coefficient

LOCr	LOCa	Ea
229	240	1100
382	255	1020
215	270	900
336	240	760
316	227	800
316	322	1200
133	120	260
433	412	1500
266	170	500
296	353	1700

$$r(x,y)^2 = \left(\frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2 \right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i \right)^2 \right]}} \right)^2$$

= 0.85

= high confidence

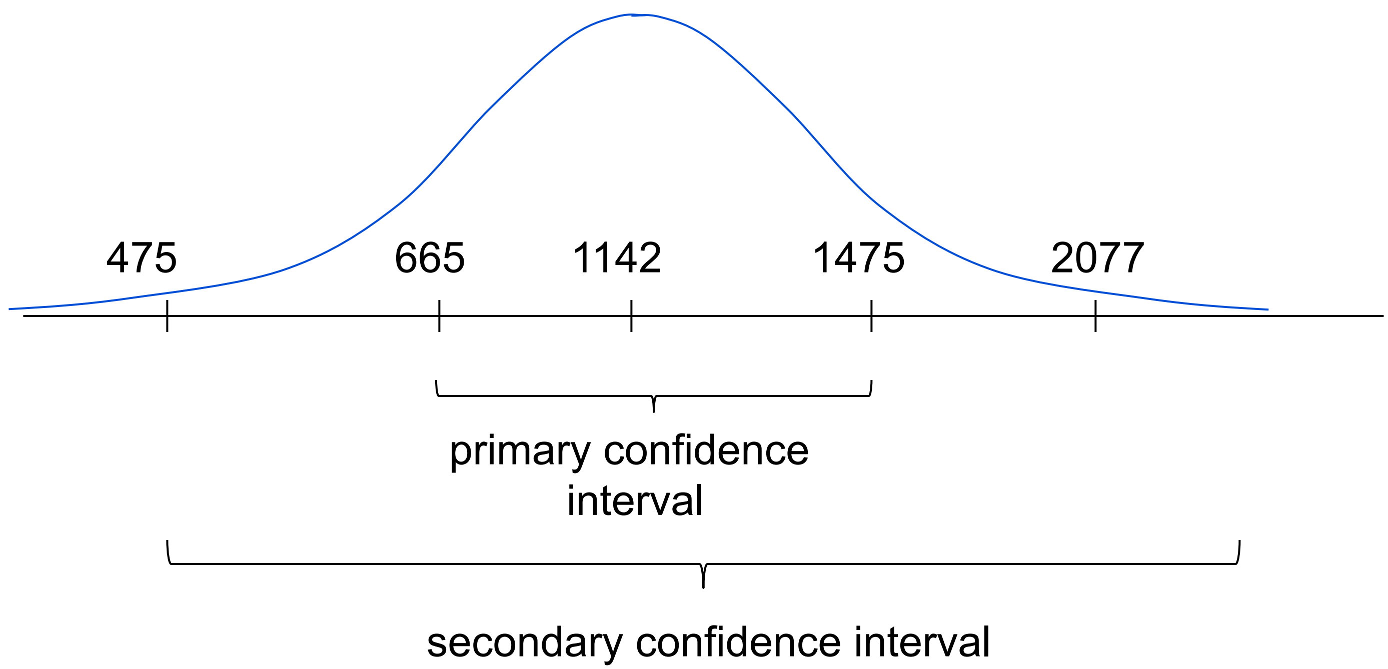
$1.0 \geq r^2 \leq .75$	high
$.75 > r^2 \leq .50$	med
$.50 > r^2$	low

Confidence

- Time prediction interval -2
  - recall size estimate:
    - LOCp = 306
    - UPI = 431
    - LPI = 219
  - calculate a "fallback" prediction
    - Ep LLPI = LOCp LPI*2.17 = 475
    - Ep UUPI = LOCp UPI*4.82 = 2077

Ea/LOC
4.58
4.00
3.33
3.17
3.52
3.73
2.17
3.64
2.94
4.82

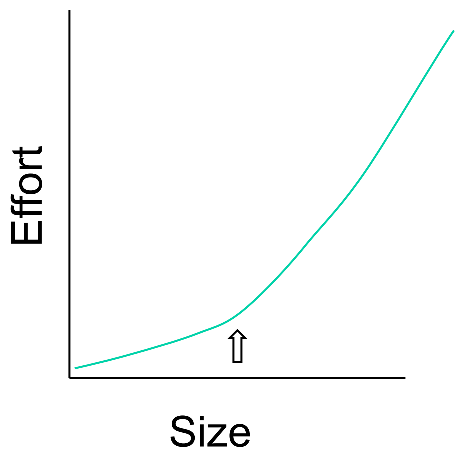
Result

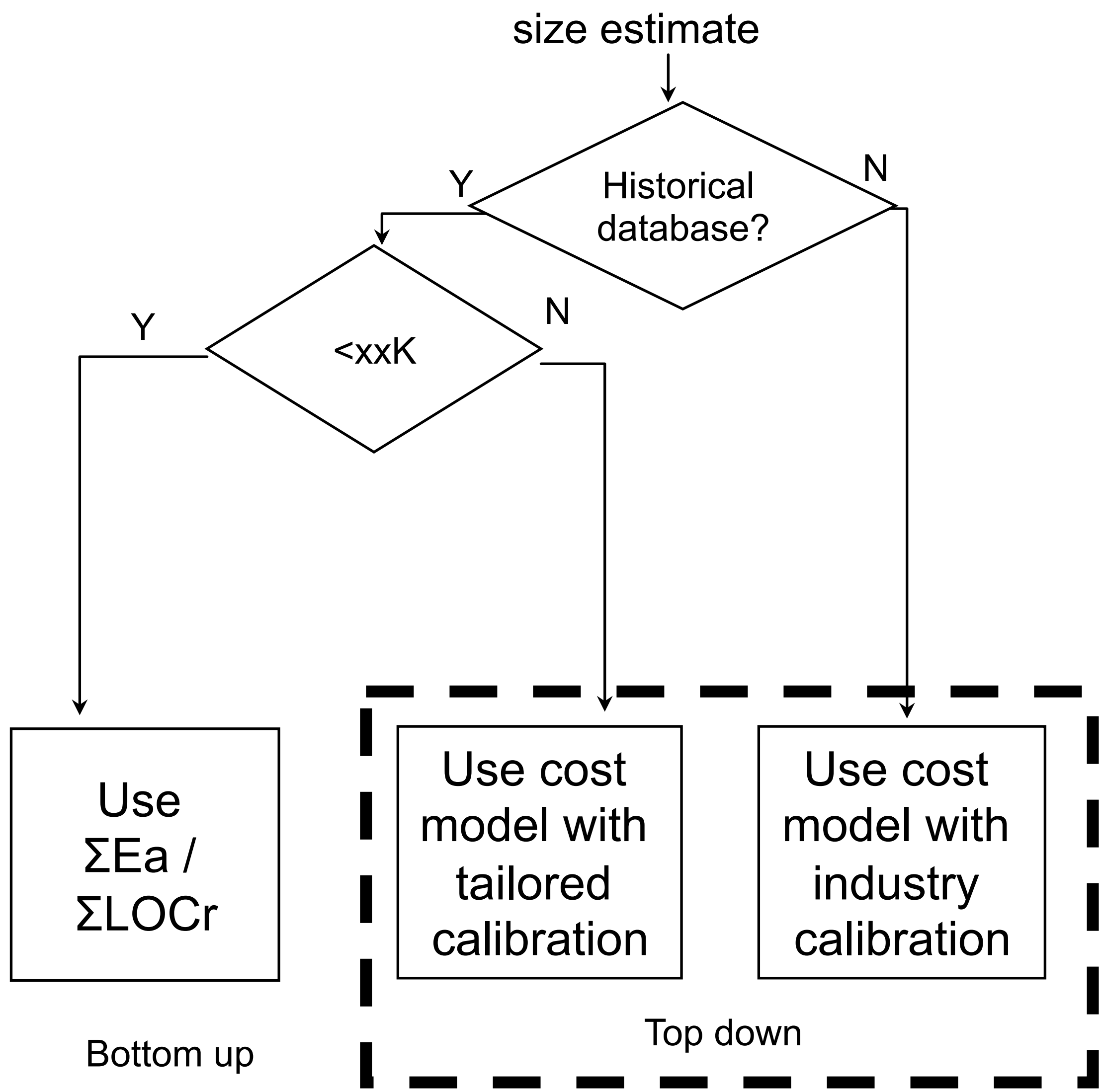


475
665
1142
1475
2077
primary confidence interval
secondary confidence interval

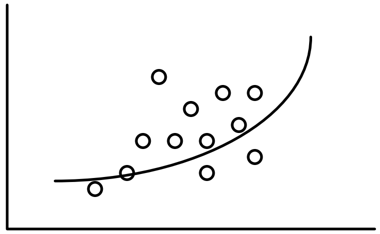
4. Calculating Ep when LOCr is very large

Project Duration





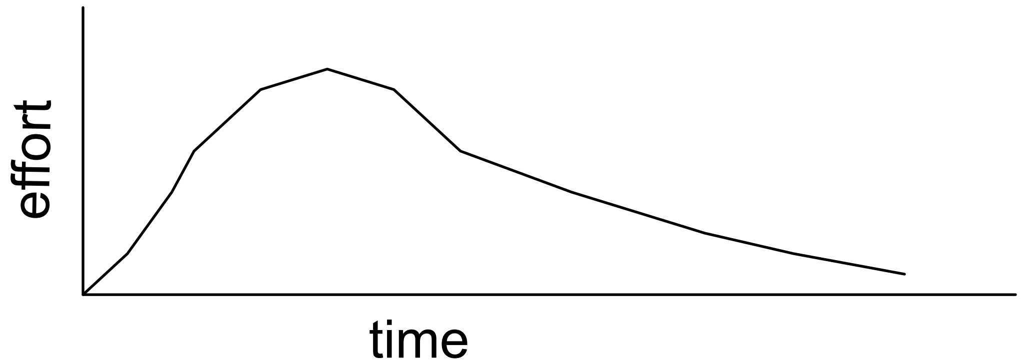
Cost Models



- Size/cost models
  - history based
    - static single-variable model
      - Resource = c_1 x (estimated characteristic)^c_2
        - » where resource is effort, project duration, staff size, etc.
        - » c_1 and c_2 are constants derived from historical data
        - » estimated characteristic describes production environment
    - static multivariable model
      - Resource = $c_1e_1 + c_2e_2 + ...$
        - » where e_i is the ith software characteristic
        - » c_1, c_2 are empirically derived constants for the ith characteristic

Cost Models

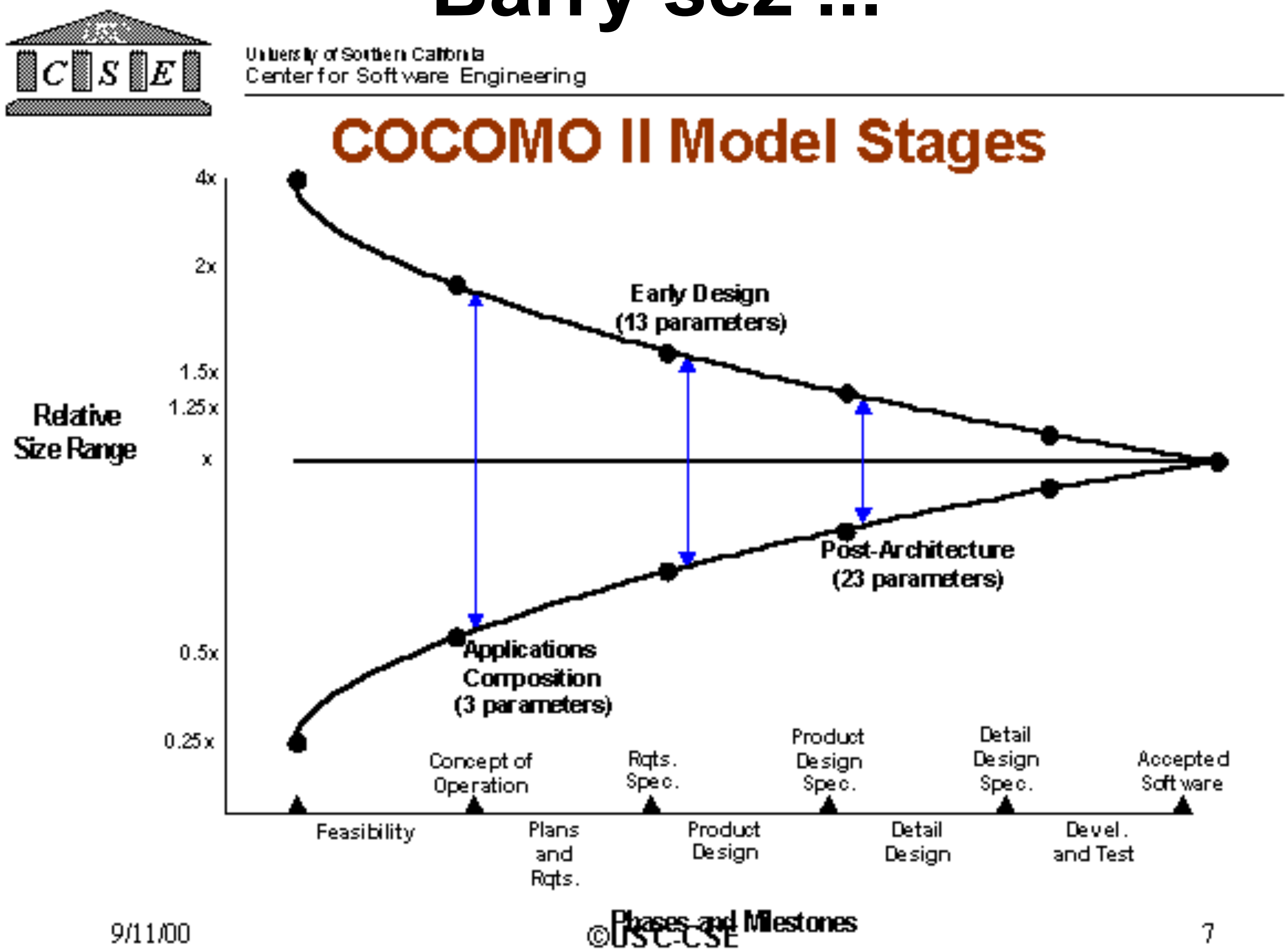
- Size/cost models
  - theoretic based
    - projects resource requirements as a function of time (typically using Rayleigh-Norden curve)



Sample Model

- COCOMO II
  - COnstructive COst Model
  - Widely known
  - Regression analysis of industry projects
  - History
    - Introduced in 1981
    - Analysis of TRW projects
    - Three models
      - Basic COCOMO: computes effort as a function of size
      - Intermediate COCOMO: computes effort as a function of size and cost drivers
      - Advanced COCOMO: intermediate COCOMO plus assessment of cost driver impact on lifecycle phase
    - Revised in 2000
      - referred to as COCOMO 2000 or COCOMO II (vs COCOMO 81)
      - Same basic models

Barry sez ...


University of Southern California
Center for Software Engineering
COCOMO II Model Stages
Relative Size Range
4x
2x
1.5x
1.25x
x
0.5x
0.25x
Early Design (13 parameters)
Post-Architecture (23 parameters)
Applications Composition (3 parameters)
Concept of Operation
Rqts. Spec.
Product Design Spec.
Detail Design Spec.
Accepted Software
Feasibility
Plans and Rqts.
Product Design
Detail Design
Devel. and Test
Phases and Milestones
9/11/00
©USC-CSE
7

COCOMO II formula

- Schedule
  - $PM = A \times Size^{(B + 0.01 \times \Sigma SFi)} \times \Pi EM_i$
    - where
      - PM = person months
      - Size = estimated size of system in KSLOC
      - SF = scale factors
      - EM = effort multipliers
      - A = calibration data = 2.94
      - B = calibration data = 0.91

COCOMO II formula

- Schedule … basic model
  - $PM = A \times Size^{(B + 0.01 \times \Sigma SFi)}$
    - where
      - PM = person months
      - Size = estimated size of system in KSLOC
      - SF = scale factors
      - A = calibration data = 2.94
      - B = calibration data = 0.91

Scale Factors

Scale Factor	Very Low	Low	Nom	High	V High	Extra High
Precedent	Surprise! 6.20	4.96	3.72	2.48	1.24	very familiar 0.00
Flexibility	rigorous 5.07	4.05	3.04	2.03	1.01	general goals 0.00
Risk resolution	little 7.07	5.65	4.24	2.83	1.41	full 0.00
Team Cohesion	difficult interaction 5.48	4.38	3.29	2.19	1.10	seamless interactions 0.00
Process Level	CMMI L1 7.80	6.24	4.68	3.12	1.56	CMMI L5 0.00

COCOMO II formula

- Schedule … intermediate model
  - $PM = A \times Size^{(B + 0.01 \times \Sigma SFi)} \boxed{\times \Pi EM_i}$
    - where
      - PM = person months
      - Size = estimated size of system in KSLOC
      - SF = scale factors
      - EM = effort multipliers
      - A = calibration data = 2.94
      - B = calibration data = 0.91

Effort Multipliers - Full Dev Model

- Product factors
  - Required software reliability
  - Database size
  - Product complexity
  - Reuse development
  - Documentation
- Platform factors
  - Execution time constraint
  - Storage constraint
  - Platform volatility
- Personnel factors
  - Analyst capability
  - Programming capability
  - Personnel continuity
  - Applications experience
  - Platform experience
  - Language/tool experience
- Project factors
  - Use of software tools
  - Multisite development
  - Required dev schedule

For example ...

Effort Multiplier	Very Low	Low	Nom	High	V High	Extra High
RELY	annoyance 0.82	0.92	1.00	1.10	1.26	human rated 1.26
ACAP	15th pctile 1.42	35th 1.19	55th 1.00	75th 0.85	90th 0.71	100th 0.71
PCAP	15th perc 1.34	35th 1.15	55th 1.00	75th 0.88	90th 0.76	100th 0.76
PCON	48%/yr 1.29	24% 1.12	12% 1.00	6% 0.90	3% 0.81	0% 0.81
APEX	≤2 mo 1.22	6 mo 1.10	1 yr 1.00	3 yrs 0.88	6 yrs 0.81	>6 yrs 0.81

Case Study -- Airborne Radar System

- Scale factors
  - Precedentedness — Nominal
  - Flexibility: — Very Low
  - Risk Resolution: — Very High
  - Team Cohesion: — Nominal
  - Process Maturity: — High

given KSLOC = 94

PM = A × $Size^{(B \times 0.01 \times \Sigma SFi)}$ × ΠEM_i

= 526.5 person months

- Effort Multipliers

RELY	VH	STOR	H	PLEX	VH
DATA	N	PVOL	N	LTEX	N
CPLX	VH	ACAP	H	TOOL	VH
RUSE	VH	PCAP	H	SITE	N
DOCU	N	PCON	H	SCED	VL
TIME	VH	APEX	VH		

COCOMO II formula II

- Time
  - $TDEV = C \times PM^{(D + 0.2 \times 0.01 \times \Sigma SFi)}$
    - where
      - TDEV = development calendar time
      - PM = person months
      - SF = Scale factor
      - C = calibration data = 3.67
      - D = calibration data = 0.28
- Airborne Radar System
  - TDEV = 17.5 months

Summary

Topics

- Rationale
- Principles of estimation
- Common techniques
  - Wideband-delphi
  - Function points
  - Fuzzy-logic
  - Standard component
  - Component
- Component-based estimation
  - process
    - size
    - effort
    - confidence
  - setup

Key Points

- "Size" is a key measurement of the overall software process
- There are a number of ways to estimate size, all of which are tenable EXCEPT guessing
- Component-based estimation is our approach this semester
  - entails sizing by component
  - adjusted by historical bias