Software Engineering
Lecture 04

Software Estimation

Referenced documents may be accessed via the URLs located on the course Angel page. Off-campus access will require authentication.

UAH
CPE 353

Outline

- Motivation
- Estimation Methods
  - Delphi Methods
  - Function Points
  - Use Case Points Estimation
  - PROBE

Motivation - 1

- ***Effort Estimation***
  – The anticipated amount of labor required to complete a project

- Estimation errors with real world projects
  – Published surveys report 30%-89% of project experience overruns

Motivation - 2

- ***Software Estimation*** **is used early-on to help construct bids for new business**
  - Rough Order-of-Magnitude estimate (ROM)
  - Accurate estimation can help management create reasonable budgets and schedules for upcoming work

Motivation - 3

- ***Earned-Value Management techniques utilize estimation to build project plans***
  - Estimated task size is used to calculate earned value
  - Better size estimates produce a more accurate EV plan

Delphi Method

- Rand Corporation, 1948
- Small team of experts given product specifications, constraints, and goals
- Team members produce anonymous individual estimates
- If consensus is not reached within 2-3 iterations, the highest and lowest estimates are discarded and the average of the remaining estimates determines the final estimate
- Several variations of method

Wideband-Delphi Estimation

1. Experts discuss program requirements.
2. Each expert creates a task list and makes an initial anonymous estimate.
3. Moderator tabulates results and distributes summary to each expert.
4. Experts discuss tasks associated with program.
5. Repeat process starting with step 2 until estimates converge.

Wideband Delphi Example

Project: __ABC___ Round # ___1______

Estimator: _Homer Simpson_

```
        X              X*      X!     X                  X
|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
0          20          40          60          80          100
```

X - estimates

X* - your estimate

X! - median estimate

Estimate for next round: _________

Rationale for estimate:

Wideband-Delphi Estimation

- Pros
  - Can produce accurate estimates
  - Discussion likely to highlight critical issues
- Cons
  - Requires access to expert estimators
  - Estimators may not accurately recall prior projects
  - Estimators may have biases
  - What happens if estimates don't converge?

Function Point Estimation

- Size estimation technique by Albrecht (1979)
- Based on the estimated occurrence and complexity of five types of components in the product
  - Inputs (Inp)
  - Outputs (Out)
  - Inquiries (Inq)
  - Master files (Maf)
  - Interfaces (Inf)
- Each complexity level (simple, average, or complex) scales that component instance by a different multiplier

Function Point Estimation

- The scaled summation UFP (Unadjusted Function Points) is adjusted further by a TCF (Technical Complexity Factor)

  $$FP = UFP \times TCF$$

- Each Technical Influence Factor's impact is rated from 0-5
- Influence factors summed to compute total degree of influence (DI) and then scaled to compute TCF

  $$TCF = 0.65 + 0.01 \times DI$$

UFP Example

Component Type	Complexity	Weight	Frequency	Totals
Input	simple	3	2	6
	average	4	1	4
	complex	6	1	6
Output	simple	4	3	12
	average	5	1	5
	complex	7	0	0
Inquiry	simple	3	1	3
	average	4	0	0
	complex	6	2	12
Master File	simple	7	2	14
	average	10	0	0
	complex	15	1	15
Interface	simple	5	4	20
	average	7	2	14

TCF Example

Technical Factors	Estimated Influence (0-none to 5-strong)
1. Data communication	0
2. Distributed data processing	2
3. Performance criteria	2
4. Heavily utilized hardware	1
5. High transaction rates	4
6. Online data entry	5
7. End-user efficiency	5
8. Online updating	2
9. Complex computations	0
10. Reusability	0
11. Ease of installation	2
12. Ease of operation	3
13. Portability	2
14. Maintainability	1
Total Degree of Influence (DI)	**29**

FP Example

- TCF = 0.65 + 0.01 x DI

  = 0.65 + 0.01 x 29 = 0.94
- FP = UFP x TCF

  = 111 x 0.94 = 104.34 function points

Function Point Estimation

- Pros
  - Some studies have shown FP estimation to be more accurate than LOC-based estimates
- Cons
  - Not automatically countable
  - Product maintenance does not necessarily lead to changes in function point count
  - Implementation independent??

Use Case Points Estimation Method

** Unless otherwise noted, lecture notes are based on
Gustav Karner, “Resource Estimation for Objectory Projects”, Objective Systems SF AB, 1993.
Roy K. Clemmons, “Project Estimation with Use Case Points”, *CrossTalk,* February 2006, pp. 18-22.

Sample Use Case Description

Use Case Number: 001

Use Case Name: Borrow Item

Actors: Librarian, Patron

Summary:

Patron selects an item, and the librarian assists the patron with the process of borrowing the item from the library.

Description:

Patron selects item from the library's holdings.

1. Librarian requests validation of patron's eligibility to borrow an item.
2. System indicates patron is eligible.
3. Librarian inputs item identifier.
4. System indicates item has been borrowed by patron.

Librarian deactivates security tag.

…

Alternative Flow #1: Patron is ineligible - no library card

2*. System indicates patron is ineligible due to lack of library card.

3*. System prompts librarian to Cancel transaction or Suspend while issuing card.

…

Alternative Flow #2: Patron is ineligible – unpaid overdue fines

…

Author: Homer J. Simpson

Use Cases: Requirements in Context, by Daryl Kulak and Eamonn Guiney

Use Case Points

- A measure of software product complexity derived from the use case model
- This model examines complexity from various perspectives
  - Use case scenarios
  - Actors involved
  - Technical factors
  - Environmental factors
- Based upon work by Gustav Karner, 1993

UUCP

- UUCP = Unadjusted Use Case Points
  - Based upon Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW)
  - UUCP = UUCW + UAW

Unadjusted Use Case Weight

- For each Use Case, determine its UUCW from the table below

Use Case Category	User Interface	Steps in Scenario (including alternative courses)	Implemented Using the Following Number of Classes	Database Entries Touched**	Weight
Simple	Simple	<= 3	< 5	1	5
Average	More elaborate	4-7	5 - 10	>= 2	10
Complex	Complex / significant processing	>7	More than 10	>= 3	15

** This column not listed by Gustav Karner

Example UUCW Calculation

Use Case Type	Weight	# of Use Cases per Type	Totals
Simple	5	5	25
Average	10	8	80
Complex	15	2	30

UUCW Total = 135

Unadjusted Actor Weight

- For each Use Case, evaluate each actor involved in that Use Case

Actor Type	Description	Weight
Simple	**Another system with a defined interface (API)**	**1**
Average	**Another system interacting through a protocol (ex. TCP/IP) or a human interaction with a line terminal.**	**2**
Complex	**Person interacting through a graphical user interface.**	**3**

Example UAW Calculation

Actor Type	Weight	# of Actors per Type	Totals
Simple	1	2	2
Average	2	4	8
Complex	3	2	6

UAW Total = 16

Example UUCP Calculation

- UUCP = UUCW + UAW

  = 135 + 16

UUCP = 151

Technical Complexity Factor (TCF)

- Used to scale use case points based upon thirteen technical factors
- Rank 0 (irrelevant) to 5 (critical) to project success; if unknown, use rank of 3 = average
- **TCF = 0.6 + 0.01*TTF where TTF = Technical Total Factor**

Technical Factor	Description	Weight
T1	Distributed system	2
T2	Performance (response or throughput)	1
T3	End user efficiency	1
T4	Complex internal processing	1
T5	Reusability	1
T6	Easy to install	0.5
T7	Operational ease, usability	0.5
T8	Portability	2
T9	Easy to change	1
T10	Concurrency	1
T11	Special security features	1
T12	Provides direct access for third parties	1
T13	Special user training facilities required	1

Example TTF Calculation

Technical Factor	Weight	Perceived Complexity	Calculated Factor (Weight x Perceived Complexity)
T1	2	0	0
T2	1	1	1
T3	1	1	1
T4	1	2	2
T5	1	4	4
T6	0.5	4	2
T7	0.5	4	2
T8	2	2	4
T9	1	1	1
T10	1	0	0
T11	1	1	1
T12	1	0	0
T13	1	1	1

Product

Your Best Estimate

TTF Total = 19

Environmental Complexity Factor (ECF)

- Used to scale use case points based upon eight environmental factors
- Rank 1 (strong negative impact) to 5 (strong positive impact) on project success,
- 0 = no impact
- **ECF = 1.4 - 0.03*ETF where ETF = Environmental Total Factor**

Environmental Factor	Description	Weight
E1	Familiarity with UML (was Objectory)	1.5
E2	Part-time workers	-1
E3	Analyst capability	0.5
E4	Application experience	0.5
E5	Object-oriented experience	1
E6	Motivation	1
E7	Difficult programming language	-1
E8	Stable Requirements	2

Example ETF Calculation

Environmental Factor	Weight	Perceived Impact	Calculated Factor
E1	1.5	5	7.5
E2	-1	0	0
E3	0.5	5	2.5
E4	0.5	4	2
E5	1	5	5
E6	1	5	5
E7	-1	3	-3
E8	2	3	6

Product

Your Best Estimate

ETF Total = 25

Example UCP Calculation

- Recall
  - UUCW = 135
  - UAW = 16
  - UUCP = UUCW + UAW = 151
  - TTF = 19
  - TCF = 0.6 + 0.01*TTF = 0.6 + 0.01*19 = 0.79
  - ETF = 25
  - ECF = 1.4 - 0.03*ETF = 1.4 - 0.03*25 = 0.65
- UCP = UUCP * TCF * ECF

  = 151 * 0.79 * 0.65 = 77.54 ≈ 78 use case points

Productivity Factor (PF)

- Productivity factor is the amount of development time per use case point

- Typical PF values for large projects
  - 15 <= PF <= 30
  - Smaller number = more efficient
  - Use PF = 20 for new team on first project

- For your team project, to estimate labor required, use the following productivity factor

  PF = 3 hours per use case point

Total Use Case Point Time Estimate

- Estimated Time = UCP * PF

- For our example,
  - Estimated Time = UCP * PF

    = 78 [use case points] * 3 [hours / use case point]

    = 234 hours total estimated time

Challenges of UCP Estimation

- Use cases may be written from various perspectives so which is appropriate to use??
  - External users of system
  - Subsystem to subsystem
  - Etc.
- Productivity factor
  - Can vary depending upon team composition, skills and experience, methodologies used, etc.
  - Ideally, use historical data to calculate PF
  - May have to manually count prior products to determine UCP to Development Time numbers

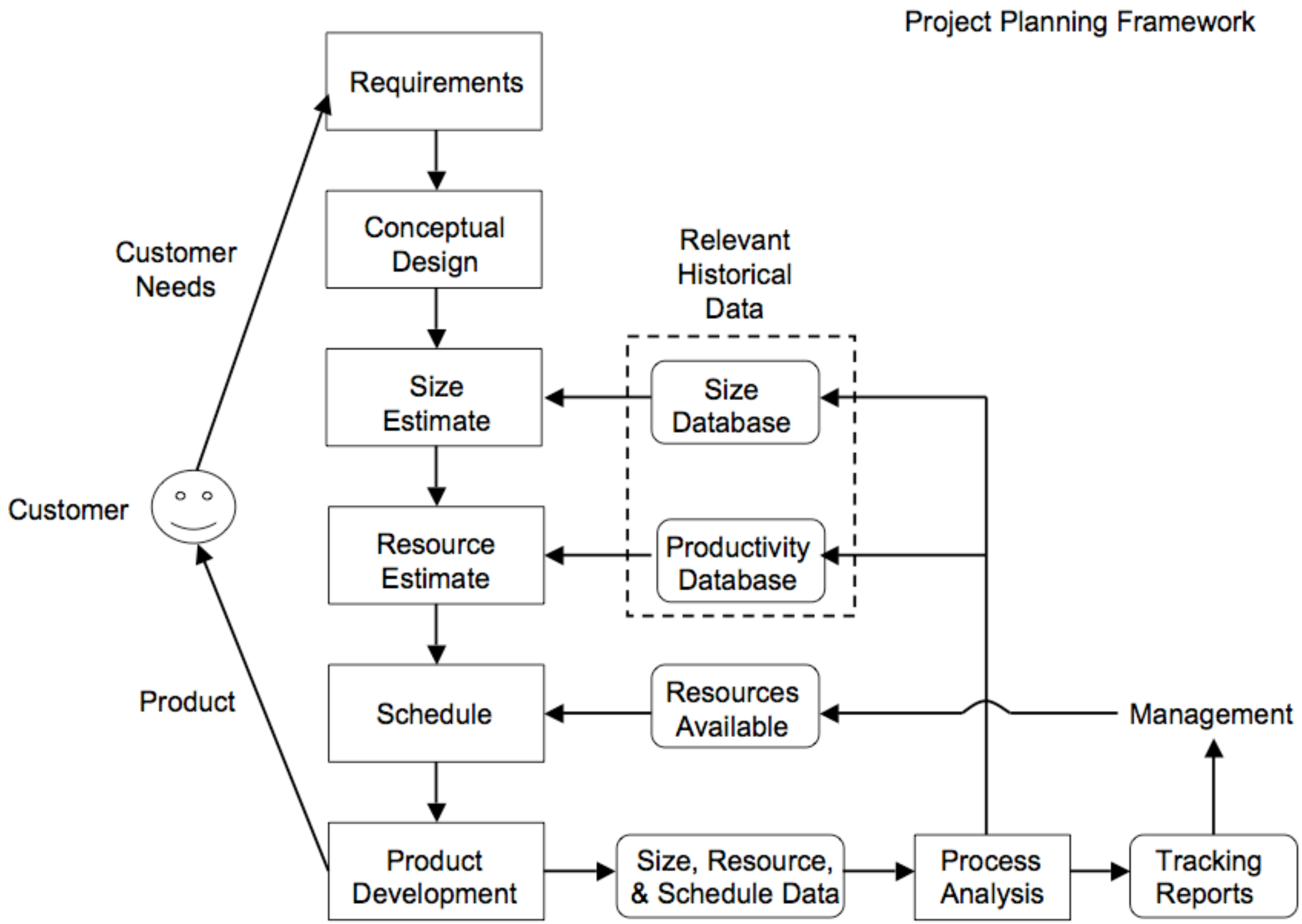
Proxy-Base Estimation

PROBE

PROxy-Based Estimation

- What is a Proxy?
  - A stand-in
- Some characteristics of a good proxy
  - Easy to visualize early in development
  - Closely related to development effort
  - Automatically countable
- Possible proxies
  - Files, screens, function points, scripts, etc.
- Personal Software Process (PSP)
  - Proxy = Estimated Object LOC
  - High correlation
  - Low significance

PSP Project Planning



PROBE Size Estimation Method

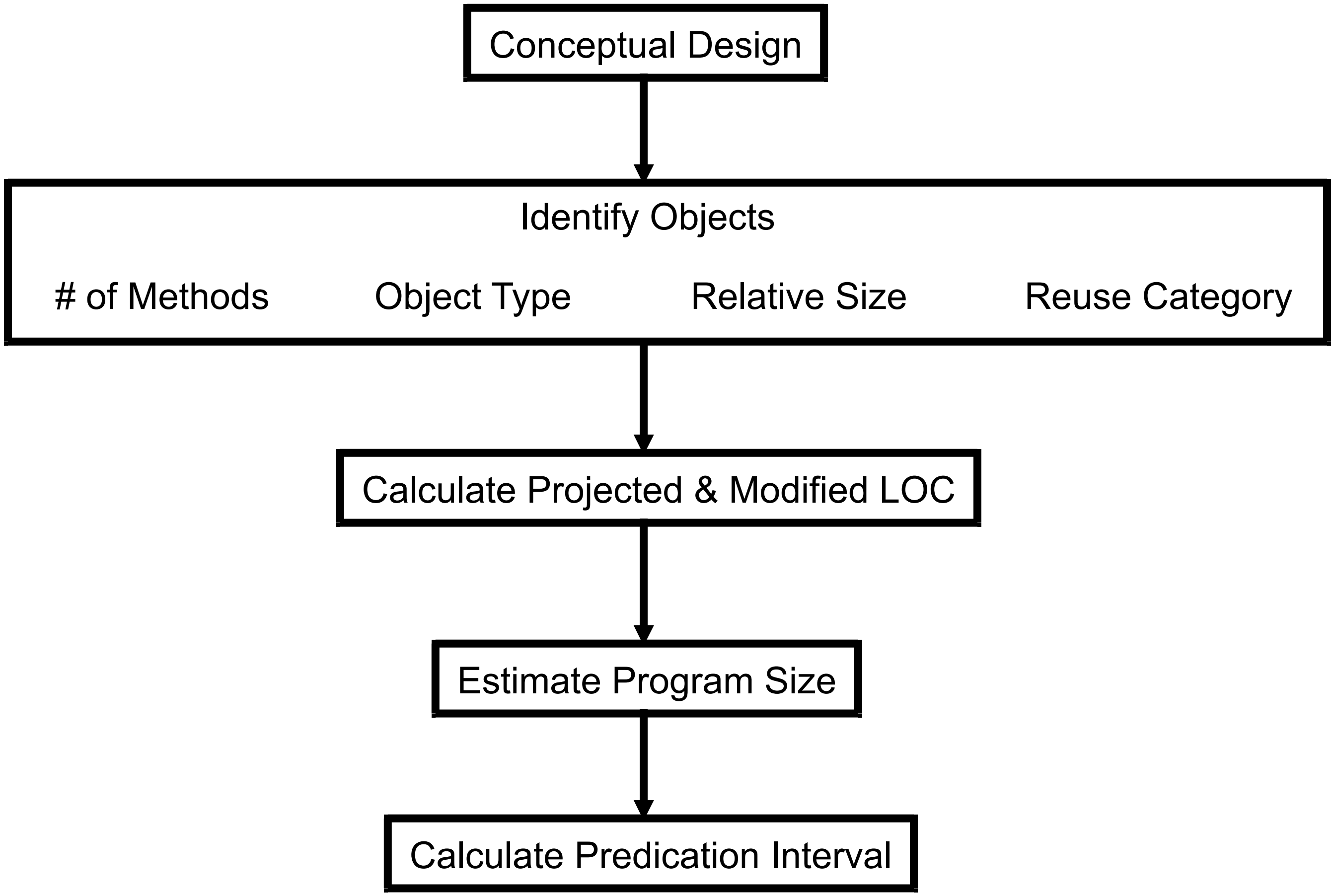


Figure 5.8, *A Discipline for Software Engineering*, Watts Humphrey

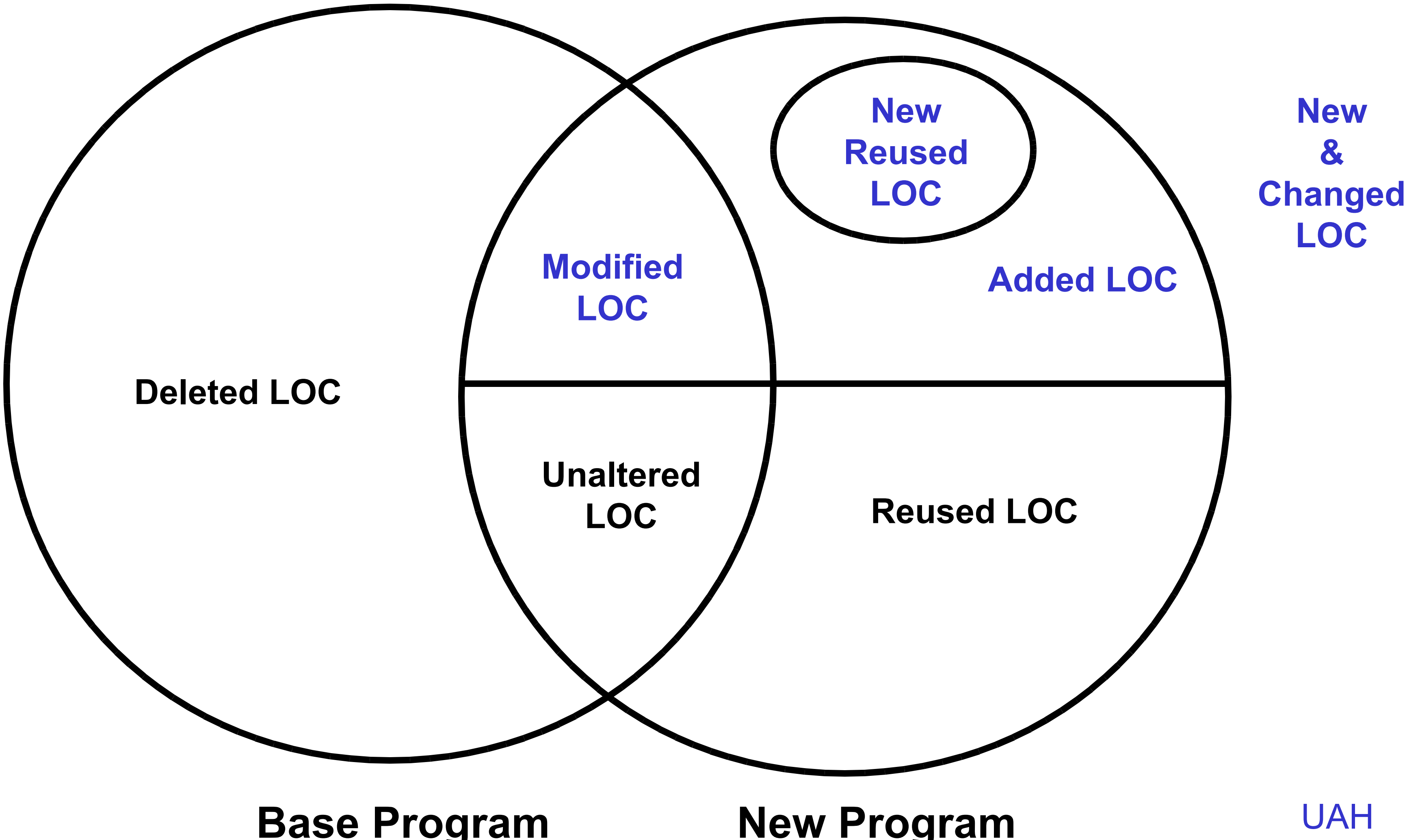
PROBE Methodology

- Conceptual Design
  - Preliminary design for estimation purposes
  - Design does not have to be complete
  - Hypothesize about objects and/or functions required
  - You may implement a different design!!
  - For Large Programs….
    - May need break down conceptual design into smaller pieces to improve estimation accuracy
- Estimate Object Size
  - Categorize Object by Type
  - Determine size based on historical data on similar objects

LOC-Based Estimation

- Physical Lines of Code
  - Varies based on the way the content has been entered
- Logical Lines of Code
  - Each language construct is counted according to a pre-defined standard regardless of the physical lines required to represent the construct
- Advantages of LOC-Based Estimation
  - Automatically counting

SEI LOC Counting Venn Diagram



Object LOC Estimation Example

Base Program				Estimated LOC
Base Size(B)				695
Deleted (D)				0
Modified (M)				5

Base Additions (BA)	Type	#Methods	Relative Size	Estimated LOC
				0

New Objects	Type	#Methods	Relative Size	Estimated LOC
Matrix	Data	13	Medium	115
Gaussian Elimination	Calc	8	Large	197
Priority Queue	Data	3	Large	49
	Total New Objects (NO)			361

Reused Objects		Estimated LOC
Linked List		73
Data Entry		96
	Total Reused Objects (R)	169

	Size	Time	
Estimated Object LOC (E) = BA + NO + M	366		
Beta0	62	108	(From linear regression)
Beta1	1.3	2.95	(From linear regression)
Estimated New and Changed LOC = Beta0 + Beta1*E	538		
Estimated Total LOC (T) = N + B - D - M + R	1397		
Estimated Development Time = Beta0 + Beta1*E		1186	

***A Discipline for Software Engineering*, Watts Humphrey**

C++ Object Size in LOC per Method (Figure 5.7)

Category	Very Small	Small	Medium	Large	Very Large
Calculation	2.34	5.13	11.25	24.66	54.04
Data	2.60	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15	21.62	28.93
Logic	7.55	10.98	15.98	23.25	33.83
Set-up	3.88	5.04	6.56	8.53	11.09
Text	3.75	8.00	17.07	36.41	77.66

Figure 5.7, *A Discipline for Software Engineering*, Watts Humphrey

Linear Regression

Using Estimated Object LOC to predict program size/development time:

Step 1: β_0 and β_1 calculated from historical data

Historical Data: x_i = Estimated Object LOC (E)

y_i = Actual New and Changed LOC [or Actual Dev. Time]

Step 2: Determine x_k = Estimated Object LOC (E) using method illustrated in prior worksheet.

Step 3: Use x_k and beta values to calculate y_k where

y_k = Estimated New and Changed LOC [or Estimated Dev. Time]

$$y_k = \beta_0 + x_k \cdot \beta_1$$

$$\beta_1 = \frac{\sum_{i=1}^{n} x_i y_i - n x_{avg} y_{avg}}{\sum_{i=1}^{n} x_i^2 - n(x_{avg})^2}$$

$$\beta_0 = y_{avg} - \beta_1 \cdot x_{avg}$$

***A Discipline for Software Engineering*, Watts Humphrey**

Linear Regression Example

Program	Estimated Object LOC (Xi)	Actual New & Changed LOC (Yi)	Xi*Yi	Xi*Xi
1	30	40	1200	900
2	90	105	9450	8100
3	103	155	15965	10609
4	127	171	21717	16129
5	173	205	35465	29929
Sum	523.0	676.0	83797	65667
Average	104.6	135.2		
Beta1	1.19397511			
Beta0	10.3102033			
Correlation	**r**	0.95218249		
	r*r	0.9066515		

Proxy-Based Estimation

- Pros
  - Use of Estimated Object LOC complements Object-Oriented Analysis/Design
  - Utilizes historical data to improve estimation accuracy
  - Also, uses historical data which can be automatically counted in prior projects!!
  - Statistical techniques can give upper/lower bounds on estimates to a specified degree of confidence
- Cons
  - Requires adequate amount of historical size and effort data on similar projects
  - May also need additional info to identify outliers

Composite Estimation Example

Component	Estimated Size	Actual Size	Estimate Error
GUI Interface	2500	3680	-1180
Database	7500	6890	+610
Simulator (Reused)	15770	15303	+467
Scheduler	1200	1017	+183
Report Generator	2000	2390	-390
Total	28970	29280	-310

Sources of Estimation Error

- Realistic Engineering Estimate versus Management's Ideal Target
  - Estimators feel pressured to bias their estimate so that it will agree with Management's desired outcome
  - "Are you asking me for an answer or telling me what the answer needs to be?" **
- Estimators inability to support their estimate
- Evolving requirements
- Poor quality

****Software Measurement and Estimation: A Practical Approach*,
By Linda M. Laird and M. Carol Brennan

Dealing with an Unrealistic Target - 1

- Common Approach
  - Adjust estimation model parameters to force the estimate to converge with the target
- Why is this approach appealing?
  - May satisfy management’s immediate goals
- *Why is this approach is a recipe for disaster?*
  - The actual work to be done has not changed
  - Project failure is preplanned!!
  - If you have followed this approach, any future estimates you produce will be suspect.

Dealing with an Unrealistic Target - 2

- A Better Approach
  - Suppose you have been given the desired effort budget by management
  - Use the estimation methods to help you adjust the scope of the project to fit the desired budget
    - How? Re-estimate for various What-If scenarios.
- Why is this approach better?
  - Work has been scaled to satisfy management's goals
  - Now management's plan is realistic for the restricted scope
  - You have planned for success!!
- Beware!!
  - This approach is not always possible.
  - Unfortunately, some projects have been set up to fail before your input has been solicited