Introduction to Software Engineering

(CS350)

Lecture 03

Jongmoon Baik

KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY
KAIST
한국과학기술원
SINCE 1971

Estimation for Software Projects

Software Project Planning

The overall goal of project planning is to establish a pragmatic strategy for controlling, tracking, and monitoring a complex technical project.

Why?

So the end result gets done on time, with quality!

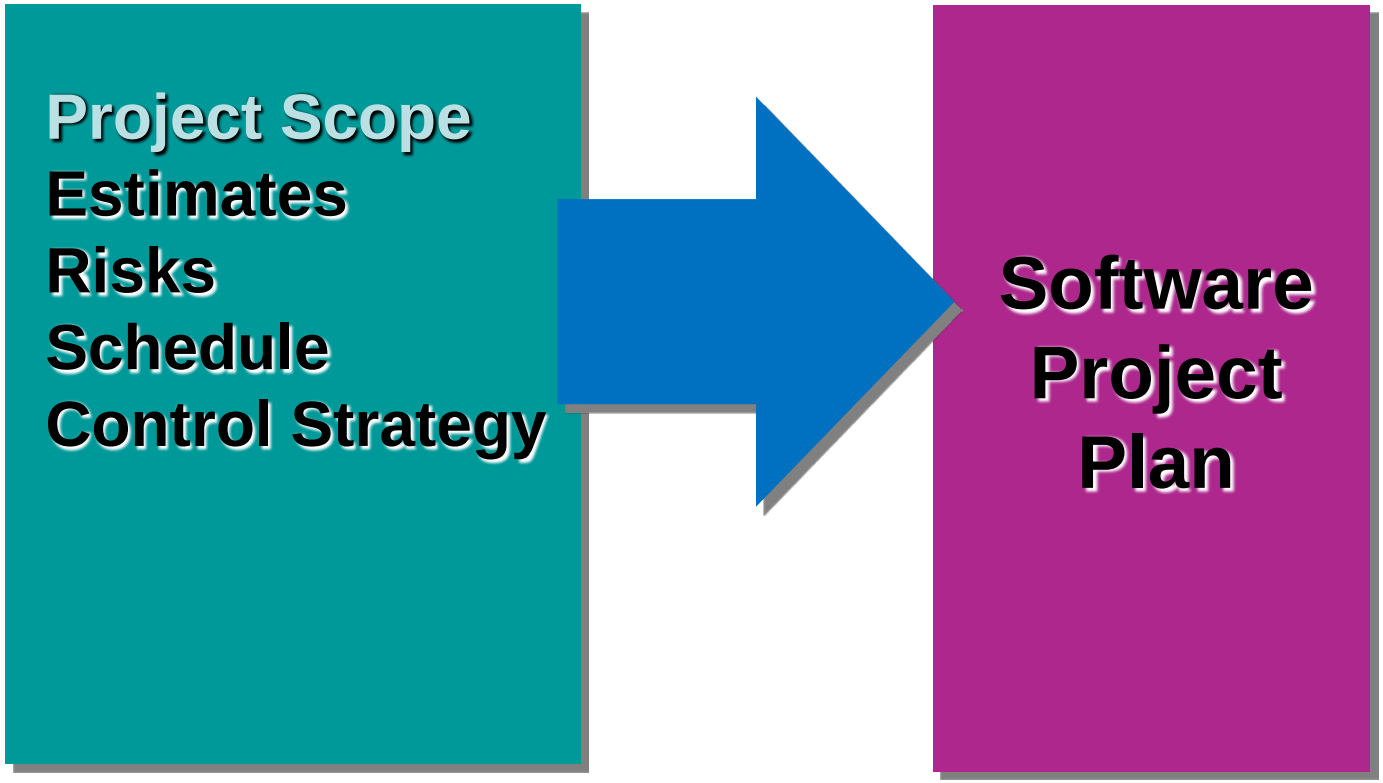
Project Planning Task Set-I

- Establish project scope
- Determine feasibility
- Analyze risks
- Define required resources
  - Determine required human resources
  - Define reusable software resources
  - Identify environmental resources

Project Planning Task Set-II

- Estimate cost and effort
  - Decompose the problem
  - Develop two or more estimates using size, function points, process tasks or use-cases
  - Reconcile the estimates
- Develop a project schedule
  - Scheduling is considered in detail in Chapter 27.
    - Establish a meaningful task set
    - Define a task network
    - Use scheduling tools to develop a timeline chart
    - Define schedule tracking mechanisms

Write it Down!



To Understand Scope ...

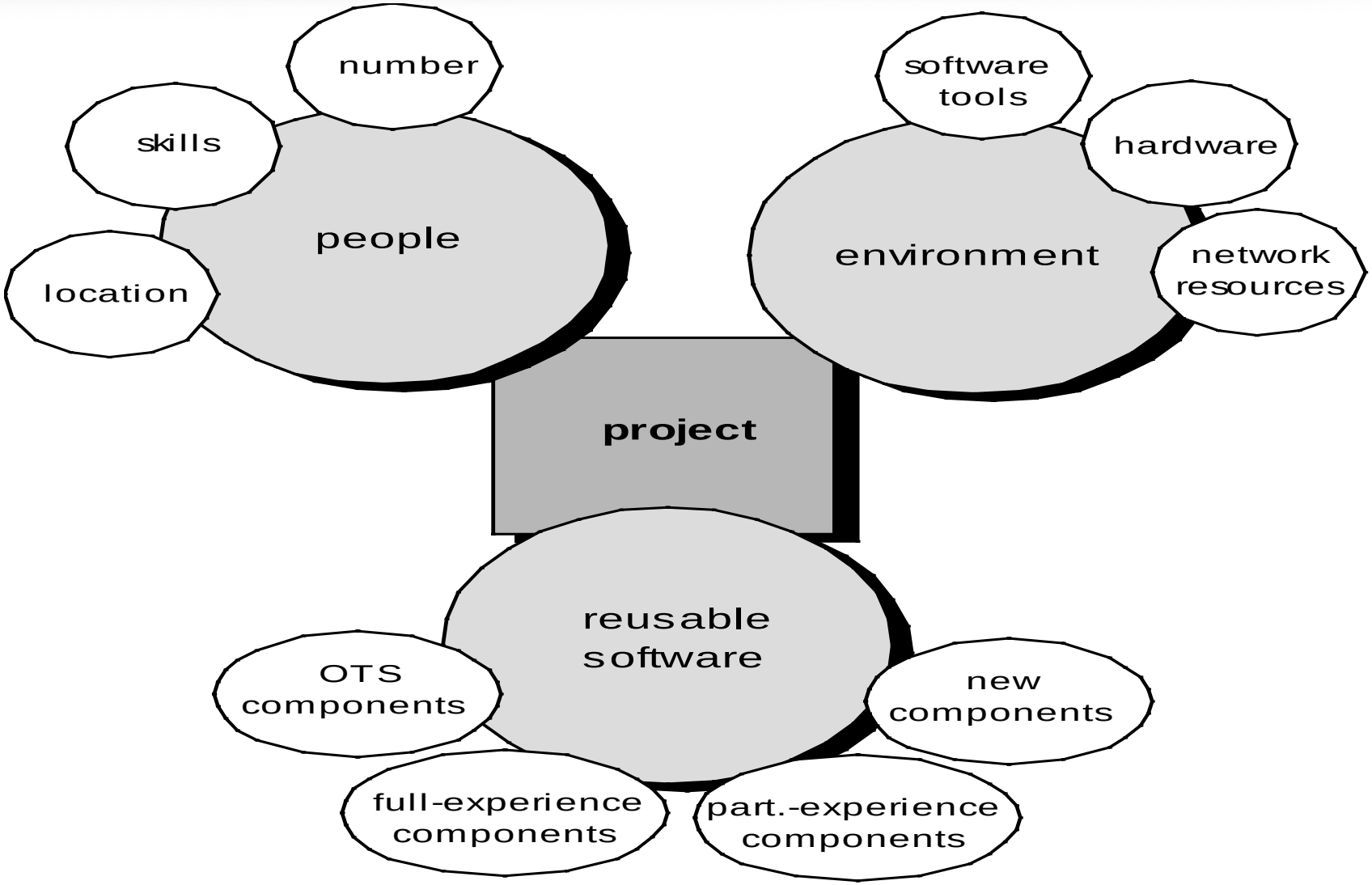
- Understand the customers needs
- Understand the business context
- Understand the project boundaries
- Understand the customer's motivation
- Understand the likely paths for change
- Understand that ...

Even when you understand, nothing is guaranteed!

What is Scope?

- ***Software scope*** describes
  - the functions and features that are to be delivered to end-users
  - the data that are input and output
  - the "content" that is presented to users as a consequence of using the software
  - the performance, constraints, interfaces, and reliability that *bound* the system.
- Scope is defined using one of two techniques:
  - "A narrative description of software scope is developed after communication with all stakeholders".
  - "A set of use-cases is developed by end-users".

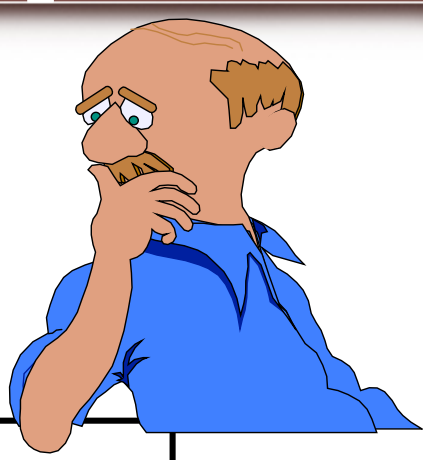
Resources


number
skills
people
location
software tools
hardware
environment
network resources
project
reusable software
OTS components
new components
full-experience components
part.-experience components

Software Project Estimation

- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process

Software Estimation Techniques

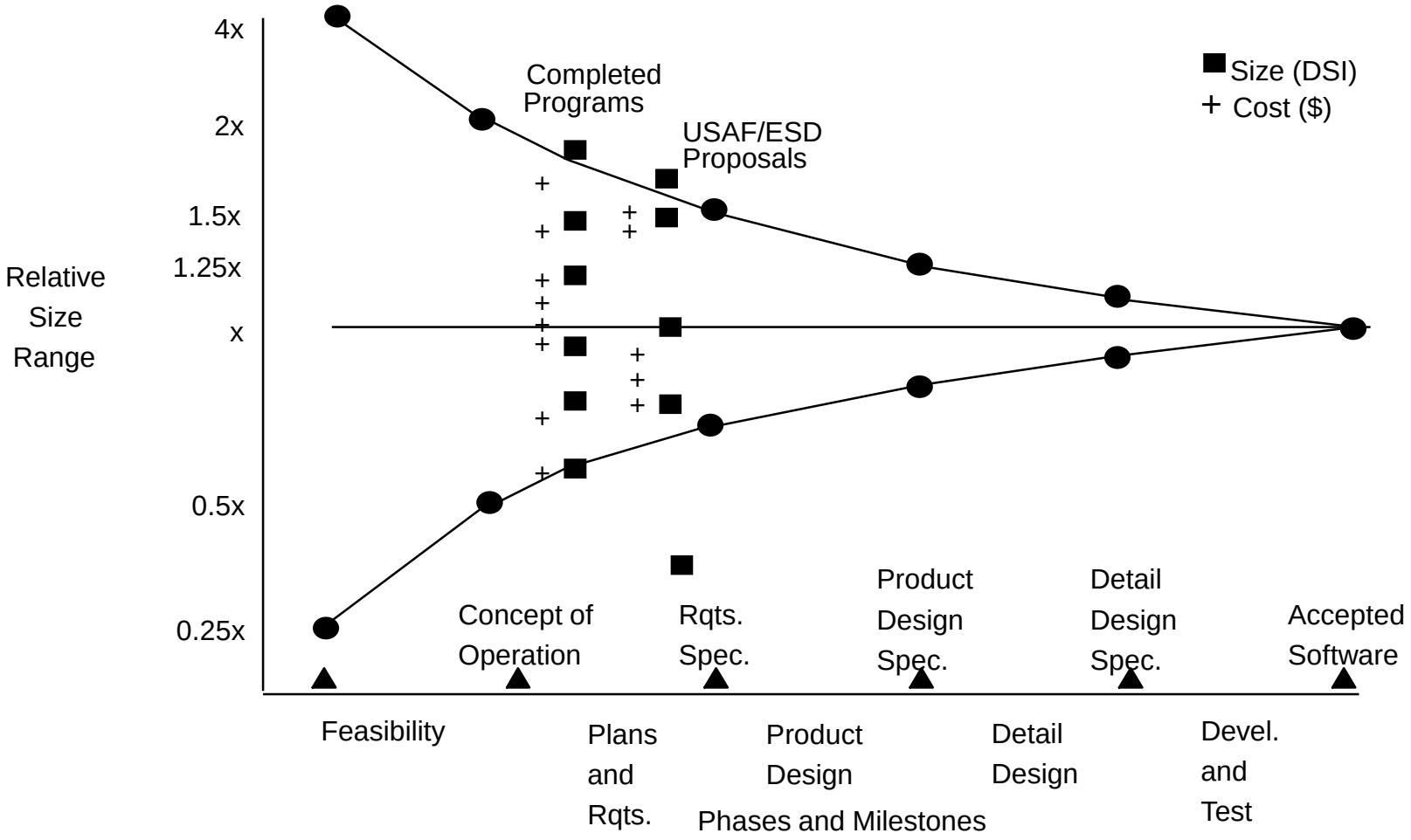


Estimation Accuracy

- Predicated on …
  - the degree to which the planner has properly estimated the size of the product to be built
  - the ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects)
  - the degree to which the project plan reflects the abilities of the software team
  - the stability of product requirements and the environment that supports the software engineering effort.

S/W Estimation Accuracy vs. Phase

"Corn of Software Cost Estimation"



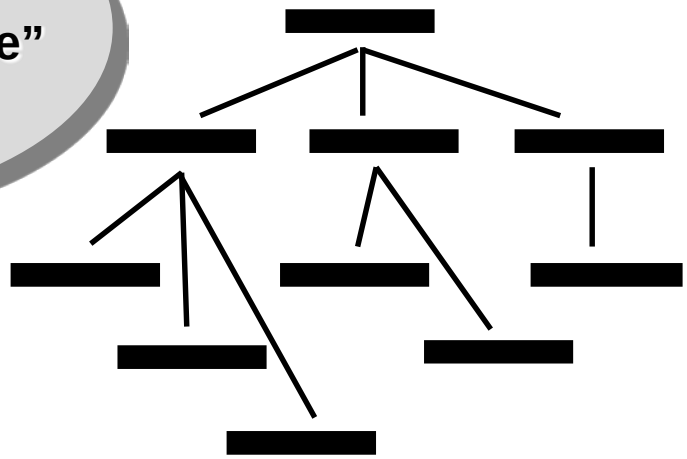
- Estimation carries inherent risk and this risk leads to uncertainty

Functional Decomposition

Statement of Scope

Perform a Grammatical “parse”

Functional Decomposition



Sizing Methods

- Source Lines of Code (SLOC)
  - SEI Definition Check List
- Unadjusted Function Points (UFP)
  - IFPUG

Source Lines of Code

- Best Source : Historical data form previous projects
- Expert-Judged Lines of Code
- Expressed in thousands of source lines of code (KSLOC)
- Difficult Definition – Different Languages
- Logical Source Statement
  - SEI Source Lines of Code Check List
  - Excludes COTS, GFS, other products, language support libraries and operating systems, or other commercial libraries

SEI Source Lines of Code Checklist

Definition Checklist for Source Statements Counts							
Definition name:	Logical Source Statements (basic definition)			Date:_____ Originator: COCOMO II			
Measurement unit:		**Physical source lines**					
		Logical source statements	√				
Statement type	**Definition**	√	**Data Array**			**Includes**	**Excludes**
When a line or statement contains more than one type, classify it as the type with the highest precedence.							
1 Executable			**Order of precedence:**		1	√	
2 Nonexecutable							
3 Declarations					2	√	
4 Compiler directives					3	√	
5 Comments							
6 On their own lines					4		√
7 On lines with source code					5		√
8 Banners and non-blank spacers					6		√
9 Blank (empty) comments					7		√
10 Blank lines					8		√
How produced	**Definition**	√	**Data array**			**Includes**	**Excludes**
1 Programmed						√	
2 Generated with source code generators							√
3 Converted with automated translators						√	
4 Copied or reused without change						√	
5 Modified						√	
6 Removed							√
Origin	**Definition**	√	**Data array**			**Includes**	**Excludes**
1 New work: no prior existence						√	
2 Prior work: taken or adapted from							
3 A previous version, build, or release						√	
4 Commercial, off-the-shelf software (COTS), other than libraries							√
5 Government furnished software (GFS), other than reuse libraries							√
6 Another product							√
7 A vendor-supplied language support library (unmodified)							√
8 A vendor-supplied operating system or utility (unmodified)							√
9 A local or modified language support library or operating system							√
10 Other commercial library							√
11 A reuse library (software designed for reuse)						√	
12 Other software component or library						√	
Usage	**Definition**	√	**Data array**			**Includes**	**Excludes**

Unadjusted Function Points - I

- Based on the amount of functionality in a software project and a set of individual project factors.
- Useful since they are based on information that is available early in the project life-cycle.
- Measure a software project by quantifying the information processing functionality associated with major external data or control input, output, or file types.

Unadjusted Function Points - II

Step 1. Determine function counts by type. The unadjusted function point counts should be counted by a lead technical person based on information in the software requirements and design documents. The number of each the five user function types should be counted (Internal Logical File (ILF), External Interface File (EIF), External Input (EI), External Output (EO), and External Inquiry (EQ)).

Step 2. Determine complexity-level function counts. Classify each function count into Low, Average, and High complexity levels depending on the number of data element types contained and the number of file types reference. Use the following scheme.

For ILF and EIF				For EO and EQ				For EI			
Record Elements	Data Elements			File Types	Data Elements			File Types	Data Elements		
	1-19	20-50	51+		1-5	6-19	20+		1-4	5-15	16+
1	Low	Low	Avg	0 or 1	Low	Low	Avg	0 or 1	Low	Low	Avg
2-5	Low	Avg	High	2-3	Low	Avg	High	2-3	Low	Avg	High
6+	Avg	High	High	4+	Avg	High	High	4+	Avg	High	High

Step 3. Apply complexity weights. Weight the number in each cell using the following scheme. The weight reflect the relative value of the function to the user.

Function Type	Complexity Weight		
	Low	Average	High
Internal Logical File (ILF)	7	10	15
External Interface Files (EIF)	5	7	10
External Inputs (EI)	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Step 4. Compute Unadjusted Function Points. Add all the weight functions counts to get one number, the Unadjusted Function Points.

Sizing Methods: LOC/FP Approach

- compute LOC/FP using estimates of information domain values
- use historical data to build estimates for the project

Example: LOC Approach

Function	Estimated LOC
user interface and control facilities (UICF)	2,300
two-dimensional geometric analysis (2DGA)	5,300
three-dimensional geometric analysis (3DGA)	6,800
database management (DBM)	3,350
computer graphics display facilities (CGDF)	4,950
peripheral control (PC)	2,100
design analysis modules (DAM)	8,400
estimated lines of code	33,200

Average productivity for systems of this type = 620 LOC/pm.

Burdened labor rate =$8000 per month, the cost per line of code is approximately $13.

Based on the LOC estimate and the historical productivity data, the total estimated project cost is **$431,000 and the estimated effort is 54 person-months.**

Example: FP Approach

Information Domain Value	opt.	likely	pess.	est. count	weight	FP-count
number of inputs	20	24	30	24	4	97
number of outputs	12	15	22	16	5	78
number of inquiries	16	22	28	22	5	88
number of files	4	4	5	4	10	42
number of external interfaces	2	2	3	2	7	15
count-total						321

The estimated number of FP is derived:

$$FP_{estimated} = count\_total \times [0.65 + 0.01 \times \sum (F_i)] = 375$$

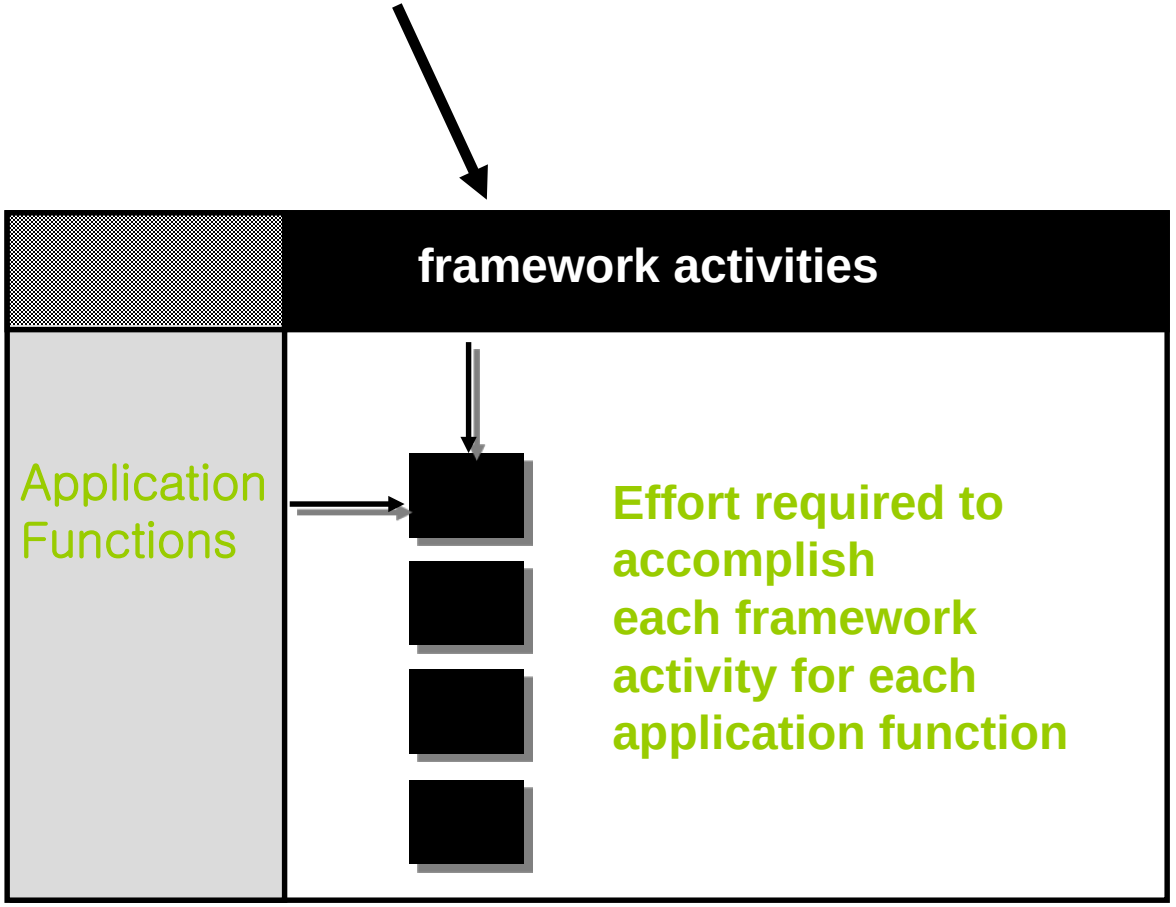
organizational average productivity = 6.5 FP/pm.

burdened labor rate = $8000 per month, approximately $1230/FP.

Based on the FP estimate and the historical productivity data, **total estimated project cost is $461,000 and estimated effort is 58 person-months.**

Process-Based Estimation

Obtained from “process framework”



Process-Based Estimation Example

Activity →	CC	Planning	Risk Analysis	Engineering		Construction Release		CE	Totals
Task →				analysis	design	code	test		
Function									
↓									
UICF				0.50	2.50	0.40	5.00	n/a	8.40
2DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	4.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DSM				0.50	3.00	0.75	1.50	n/a	5.75
PCF				0.25	2.00	0.50	1.50	n/a	4.25
DAM				0.50	2.00	0.50	2.00	n/a	5.00
Totals	0.25	0.25	0.25	3.50	20.50	4.50	16.50		46.00
% effort	1%	1%	1%	8%	45%	10%	36%		

CC = customer communication CE = customer evaluation

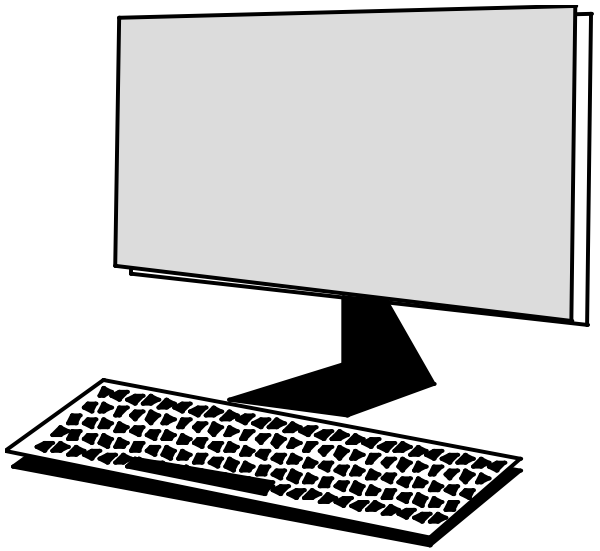
Based on an average burdened labor rate of $8,000 per month, **the total estimated project cost is $368,000 and the estimated effort is 46 person-months.**

Tool-Based Estimation

Project Characteristics →

Calibration Factors →

LOC/FP data →



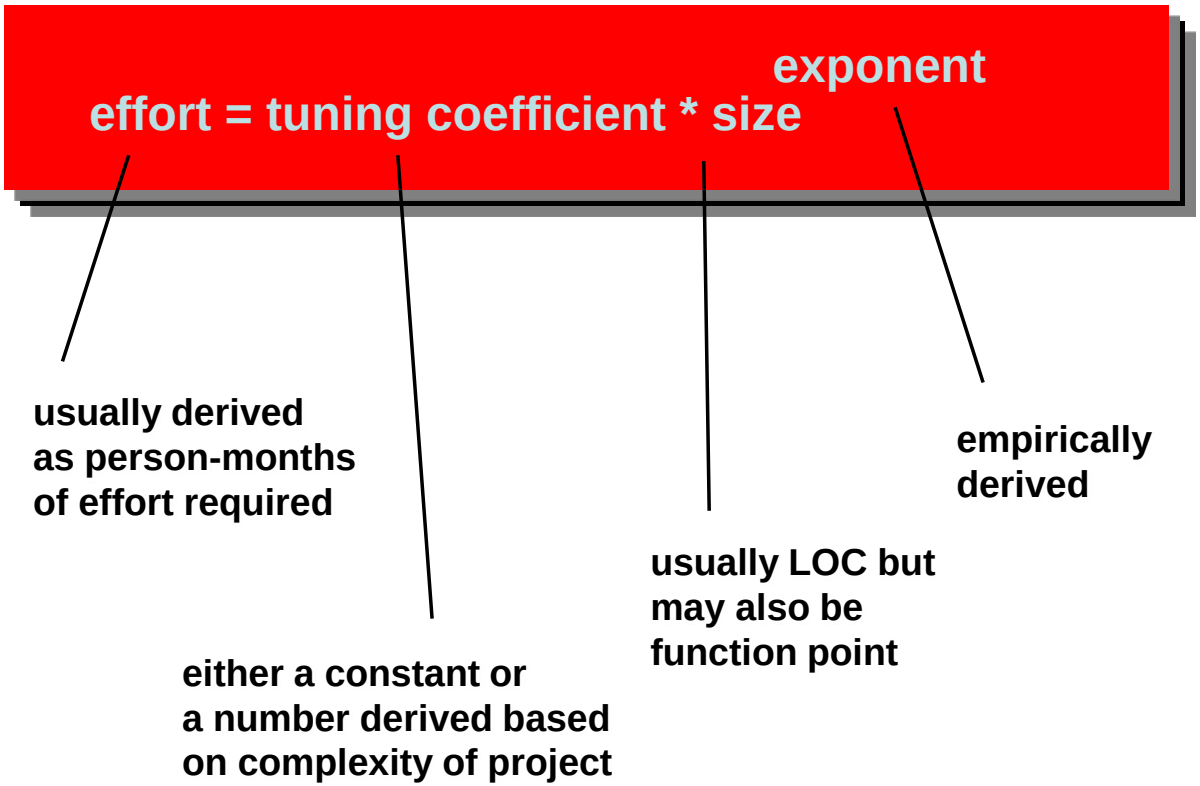
Estimation with Use-Cases

	use cases	scenarios	pages	Ê	scenarios	pages	LOC	LOC estimate
User interface subsystem	6	10	6	Ê	12	5	560	3,366
Engineering subsystem group	10	20	8	Ê	16	8	3100	31,233
Infrastructure subsystem group	5	6	5	Ê	10	6	1650	7,970
				Ê	Ê	Ê	Ê	
Total LOC estimate				Ê	Ê	Ê	Ê	42,568

Using 620 LOC/pm as the average productivity for systems of this type and a burdened labor rate of $8000 per month, the cost per line of code is approximately $13. Based on the use-case estimate and the historical productivity data, **the total estimated project cost is $552,000 and the estimated effort is 68 person-months.**

Empirical Estimation Models

General form:



COCOMO-II

- COCOMO II is actually a hierarchy of estimation models that address the following areas:
  - *Application composition model.* Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
  - *Early design stage model.* Used once requirements have been stabilized and basic software architecture has been established.
  - *Post-architecture-stage model.* Used during the construction of the software.

The Software Equation

A dynamic multivariable model

$$E = [LOC \times B^{0.333}/P]^3 \times (1/t^4)$$

where

E = effort in person-months or person-years

t = project duration in months or years

B = “special skills factor”

P = “productivity parameter”

Estimation for OO Projects-I

- Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
- Using object-oriented requirements modeling (Chapter 6), develop use-cases and determine a count.
- From the analysis model, determine the number of key classes (called analysis classes in Chapter 6).
- Categorize the type of interface for the application and develop a multiplier for support classes:

Interface type	Multiplier
No GUI	2.0
Text-based user interface	2.25
GUI	2.5
Complex GUI	3.0

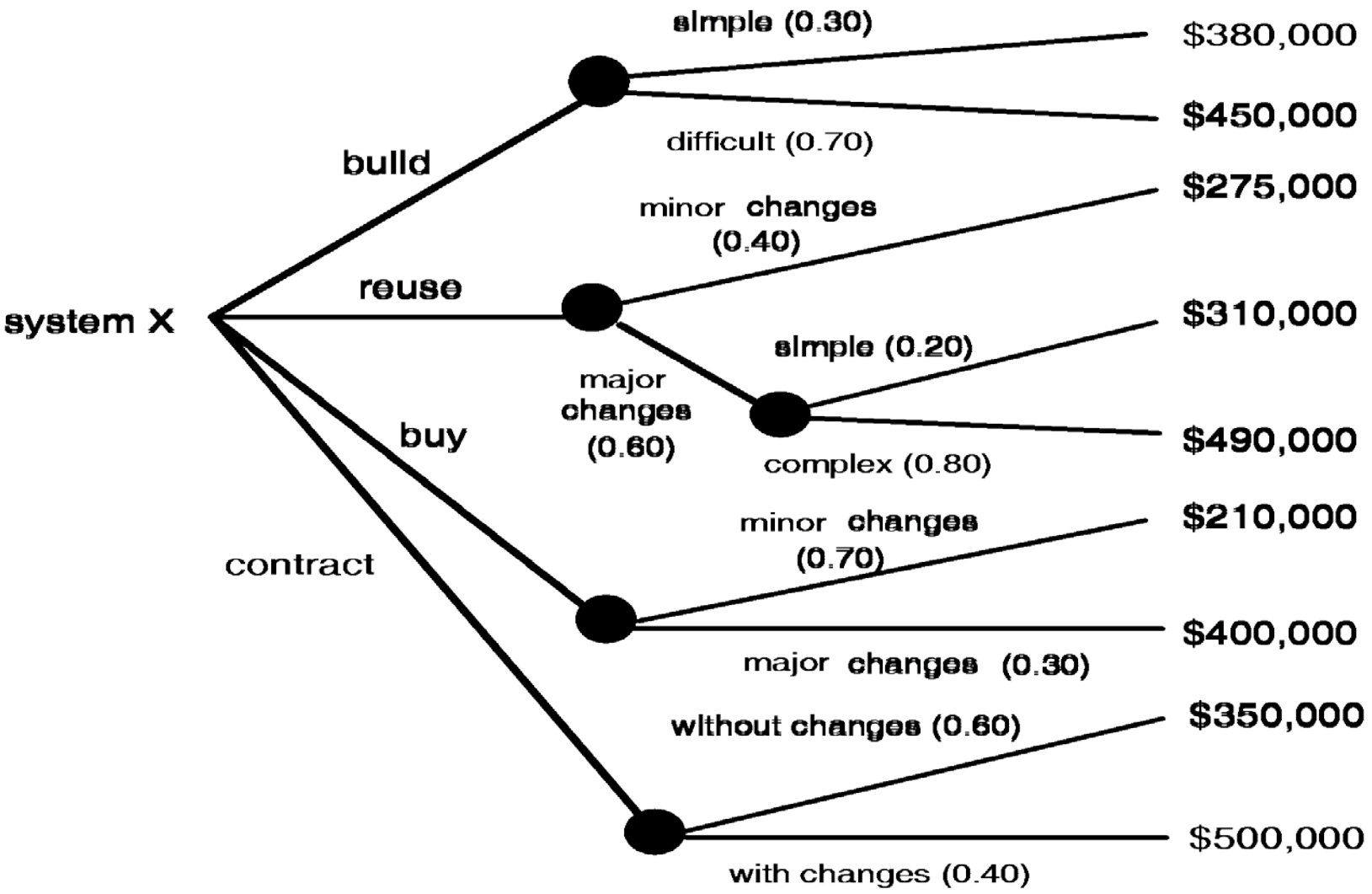
Estimation for OO Projects-II

- Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
- Multiply the total number of classes (key + support) by the average number of work-units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.
- Cross check the class-based estimate by multiplying the average number of work-units per use-case

Estimation for Agile Projects

- Each user scenario (a mini-use-case) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- Each task is estimated separately. Note: estimation can be based on historical data, an empirical model, or “experience.”
  - Alternatively, the ‘volume’ of the scenario can be estimated in LOC, FP or some other volume-oriented measure (e.g., use-case count).
- Estimates for each task are summed to create an estimate for the scenario.
  - Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.

The Make-Buy Decision



Computing Expected Cost

$$\text{expected cost} = \sum_i (\text{path probability})_i \times (\text{estimated path cost})_i$$

For example, the expected cost to build is:

$$\text{expected cost}_{build} = 0.30 (\$380K) + 0.70 (\$450K)$$
$$= \$429\ K$$

similarly,

$$\text{expected cost}_{reuse} = \$382K$$

$$\text{expected cost}_{buy} = \$267K$$

$$\text{expected cost}_{contr} = \$410K$$

Q & A