Software Project Estimation

Software project estimation is a critical part of project planning.

Before the project can begin, the software team should estimate the work to be done, the resources that will be required, and the time that will elapse from start to finish.

Project Planning Task Set-I

- Establish project scope
- Determine feasibility
- Analyze risks
- Define required resources
  - Determine require 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

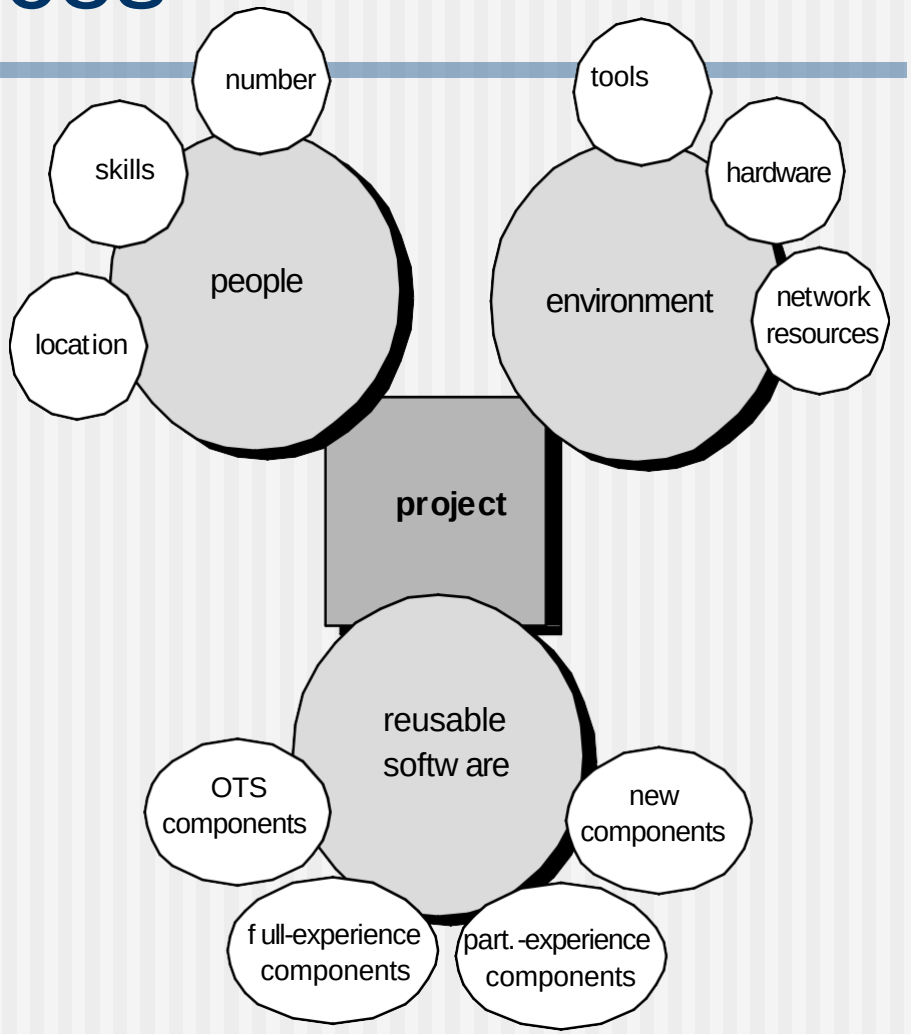
What is Scope?

- *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.

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

Resources


number
skills
people
location
tools
hardware
environment
network resources
project
reusable softw are
OTS components
new components
f ull-experience components
part. -experience components

Estimation Techniques

- Past (similar) project experience
- Conventional estimation techniques
  - task breakdown
  - size (e.g., LOC, FP) estimates
- Empirical models

Conventional 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 function (PCF)	2,100
Design analysis modules (DAM)	8,400
Estimated lines of code	*33,200*

If Average productivity for the system = 620 LOC/pm and labor rate =$8000 /pm:

Cost per LOC = $8,000 / 620 = $13

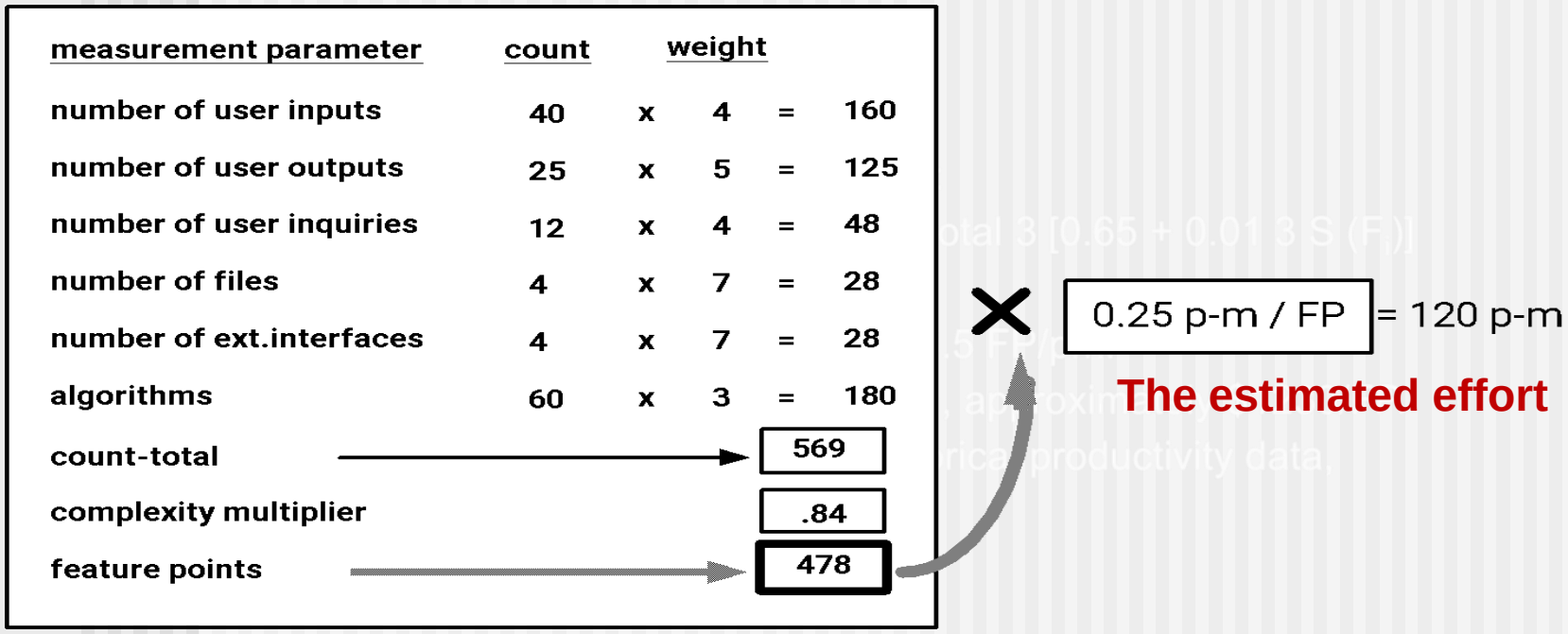
Total estimated project cost = 33,200 x $13 = $431,000

The estimated effort = 33,200 / 620

= 54 person-months.

Example: FP Approach

Effort = Feature Points × Productivity Rate



Empirical Estimation Models

An estimation model for computer software uses empirically derived formulas to predict effort as a function of LOC or FP.

COCOMO-II

- COCOMO-II (***CO****nstructive* ***CO****st* ***MO****del*) is an **empirical estimation model** used in software engineering to predict the effort, cost, and schedule required for a software project.
- It offers three sub-models, each tailored to different stages of software development:
  - *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.

COCOMO-II

Application Composition Model

Effort (Person-Months) =(Object Points) / Productivity Rate

Object Points: A count of the number and complexity of screens or reports.
Productivity Rate: Adjusted based on team experience and tool efficiency.

COCOMO-II

Application Composition Model

Developing an employee management system with:

- **20 screens** (medium complexity, weight = 2)
- **5 reports** (high complexity, weight = 3)

Step 1: Calculate Object Points:

Object Points=(20×2)+(5×3)=40+15=55

Step 2: Select Productivity Rate:

Productivity rate = 7 (Assuming moderate skill levels and standard tools)

Step 3: Calculate Effort:

Effort (Person-Months)==55/7≈7.86 person-months.

COCOMO-II

Application Composition Model

Component Type	Weight for Low Complexity	Weight for Medium Complexity	Weight for High Complexity
Screens	1	2	3
Reports	2	3	4

Factor	Range (Object Points/Person-Month)	Description
High Productivity	12–15	Skilled team with advanced tools and efficient processes.
Medium Productivity	7–12	Average team capability and standard tool support.
Low Productivity	3–7	Inexperienced team or limited tools and process support.

COCOMO-II

Early Design Model & Post-Architecture Model

$$\text{Effort} = A \times \text{Size}^{B} \times \prod(\text{EM})$$

Size: Measured LOC
A: Tuning coefficient
B: Scale factor
∏(EM) : Product of effort multipliers for the project.

COCOMO-II

Inputs

- **Size of the Project (LOC)**:
  1. Estimated size is **25,000 lines of code (LOC)**.
- **Tuning Coefficient (A)**:
  1. A=2.8 (Based on the project complexity and domain)
- **Scale Factor (B)**:
  1. B=1.12 ((medium-sized project with moderate complexity)
- **Effort Multipliers (∏EM)**:
  1. **Required Software Reliability (RELY)**: High reliability is needed (1.10).
  2. **Platform Volatility (PVOL)**: Low platform changes expected (0.87).
  3. **Analyst Capability (ACAP)**: Highly skilled analysts (0.85).
  4. **Use of Modern Tools (TOOL)**: Moderately advanced tools (0.90).
  5. **Required Development Schedule (SCED)**: Nominal (1.00).

∏EM = 1.10×0.87×0.85×0.90×1.00=0.817

Effort (Person-Months)=*A*×Size^*B*×∏*EM* = 2.8×(25)^1.12×0.817 = 70.89

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

t = project duration in months

B = “special skills factor”

P = “productivity parameter”

The Software Equation

B (special skills factor)

- *B* increases slowly as "the need for integration, testing, quality assurance, documentation, and management skills grows" [Put92]. For small programs (KLOC 5 to 15), *B* 0.16. For programs greater than 70 KLOC, *B* 0.39.

*P (*productivity parameter)

- Reflects process maturity, engineering practices, programming language level, software environment, team expertise, and application complexity.
- Typical P values: 2000 (real-time embedded software), 10000 (telecom systems), 28000 (business applications).

The Software Equation

Simplified Version

$$t_{\min} = 8.14 \frac{\text{LOC}}{p^{0.43}} \text{ in months for } t_{\min} > 6 \text{ months}$$

$$E = 180\, Bt^3 \text{ in person-months for } E \geq 20 \text{ person-months}$$

Example

$$t_{\min} = 8.14 \times \frac{33{,}200}{12{,}000^{0.43}} = 12.6 \text{ calendar months}$$

$$E = 180 \times 0.28 \times (1.05)^3 = 58 \text{ person-months}$$

The Make-Buy Decision

Acquisition Options

- **Off-the-Shelf Software:**
  Pre-built software purchased or licensed for immediate use.
- **Modified Components:**
  "Full-experience" or "Partial-experience" components are acquired, then customized and integrated.
- **Custom Software:**
  Built by an external contractor to meet specific requirements.

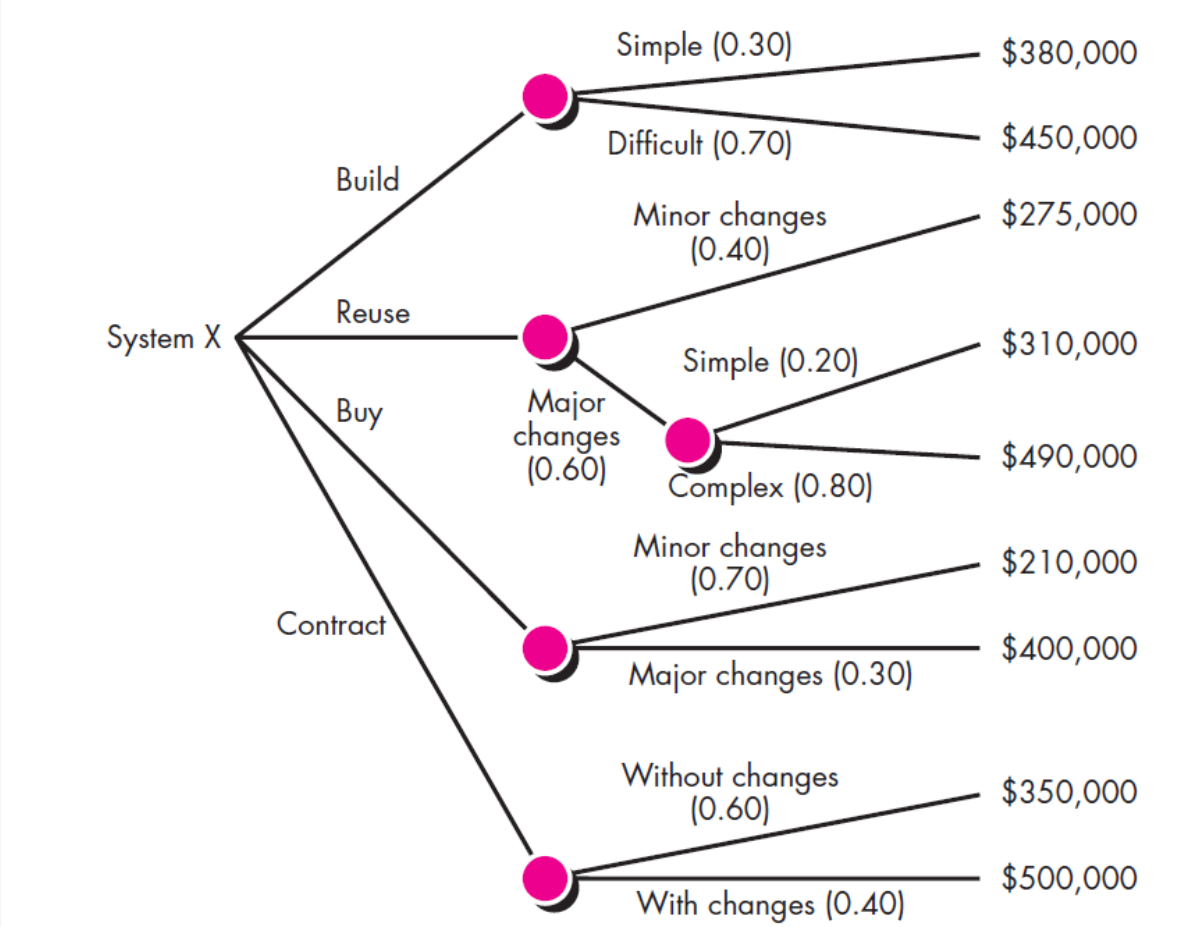
The Make-Buy Decision

Key Factors for Make/Buy Decision

- **Delivery Timeline:**
  Will purchased software be available sooner than internally developed software?
- **Cost Effectiveness:**
  Is the cost of acquisition + customization less than internal development costs?
- **Support Costs:**
  Will external maintenance contracts be cheaper than internal support?

The Make-Buy Decision

- **Decision Tree**



The Make-Buy Decision

- **Expected Cost Estimation:**

$$\text{Expected cost} = \Sigma\,(\text{path probability})_i \times (\text{estimated path cost})_i$$

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

$$\text{Expected cost}_{\text{reuse}} = 0.40\ (\$275\text{K}) + 0.60\ [0.20\ (\$310\text{K}) + 0.80\ (\$490\text{K})] = \$382\text{K}$$

$$\text{Expected cost}_{\text{buy}} = 0.70\ (\$210\text{K}) + 0.30\ (\$400\text{K}) = \$267\text{K}$$

$$\text{Expected cost}_{\text{contract}} = 0.60\ (\$350\text{K}) + 0.40\ (\$500\text{K}) = \$410\text{K}$$

- The lowest expected cost is the “buy” option.

The Make-Buy Decision

Additional Consideration Other than the Cost

- **Availability:**
  - Is the software or component readily available to meet project needs?
- **Experience of Vendor/Developer:**
  - Does the provider have the expertise and a reliable track record?
- **Requirement Conformance:**
  - Does the software align with the functional and non-functional requirements?
- **Local Considerations ("Politics"):**
  - Are there organizational or regional preferences impacting the decision?
- **Likelihood of Change:**
  - How adaptable is the solution to future changes in requirements or technology?

Reference

- Chapter 26 **- Estimation for Software Projects**

  Software Engineering: A Practitioner's Approach, 7/e

  by Roger S. Pressman