CHAPTER 14

SOFTWARE PROJECT ESTIMATION

COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

PROF. DR. KAMRUDDIN NUR

PROFESSOR, CS, AIUB

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, on budget, and with quality

PROJECT PLANNING TASK SET-1

❑ Establish project scope

❑ Determine feasibility

❑ Analyze risks

❑ Define required resources

- Determine require human resources

- Define reusable software resources

- Identify environmental resources
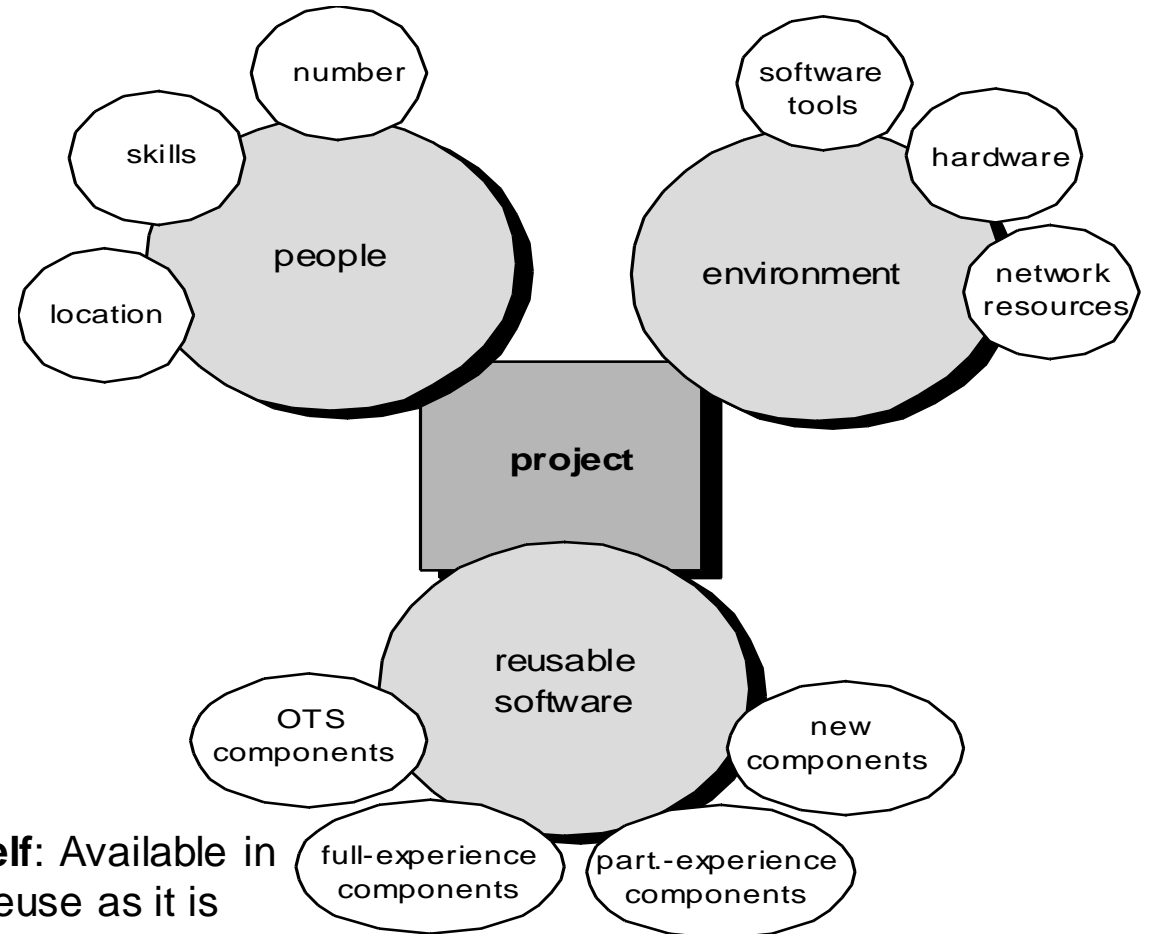
Off-the-shelf: Available in the stock, reuse as it is

MMH

PROJECT PLANNING TASK SET-II

❑ Estimate cost and effort

- Decompose the problem

- Develop two or more estimates using size, function points, process tasks (complexity),

- Reconcile the estimates

❑ Develop a project schedule

- Establish a meaningful task set

- Define a task network

- Use scheduling tools to develop a timeline chart

- Define schedule tracking mechanisms

PROJECT ESTIMATION PRINCIPLES

- Project scope must be understood

- Elaboration (decomposition) is necessary

- Historical metrics (pervious data) are very helpful

- At least two different techniques should be used

- Uncertainty is inherent in the process

CONVENTIONAL METHOD

❑ Conventional estimation techniques

- task breakdown and effort estimates

- size (e.g., LOC/FP) estimates

- compute LOC/FP using estimates of information domain values

- use historical data (previous experience) to build estimates for the project

❑ Automated tools

EFFORT ESTIMATION

- A dynamic multivariable model for effort estimation

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

Where,

E = effort in person-months or person-years (the amount of time, personnel devote to a specific project)

t = project duration in months or years

B = "special skills factor"

P = "productivity parameter"

MMH

COCOMO (CONSTRUCTIVE COST MODEL)

Based on SLOC characteristic, and operates according to the following equations:

- $\text{Effort} = \text{PM} = \text{Coefficient}_{<\text{Effort Factor}>} * (\text{SLOC}/1000)\char`\^P$ [100,000 SLOC/1000 = 100k SLOC]

- `Development time = DM = 2.50*(PM)^T`

- `Required number of people = ST = PM/DM`

PM : person-months needed for project (labor working hours)

SLOC : source lines of code

P : project complexity (1.04-1.24)

DM : duration time in months for project (week days)

T : SLOC-dependent coefficient (0.32-0.38)

ST : average staffing necessary

| Software Project Type	Coefficient <Effort Factor>	P	T
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

COCOMO (CONSTRUCTIVE COST MODEL)

Organic: relatively small, simple software projects in which a small teams with good application experience work to a software development project (e.g. showing VUES information to webpage)

Semidetached: an intermediate (in size and complexity) software project in which teams with mixed experience levels works in a mix of hardware and software application (e.g. biometric log-in time saved in VUES database)

Embedded: A software project that must be developed within a strongly coupled to hardware environment (e.g. biometric device, elevator)

MAKE-BUY DECISION
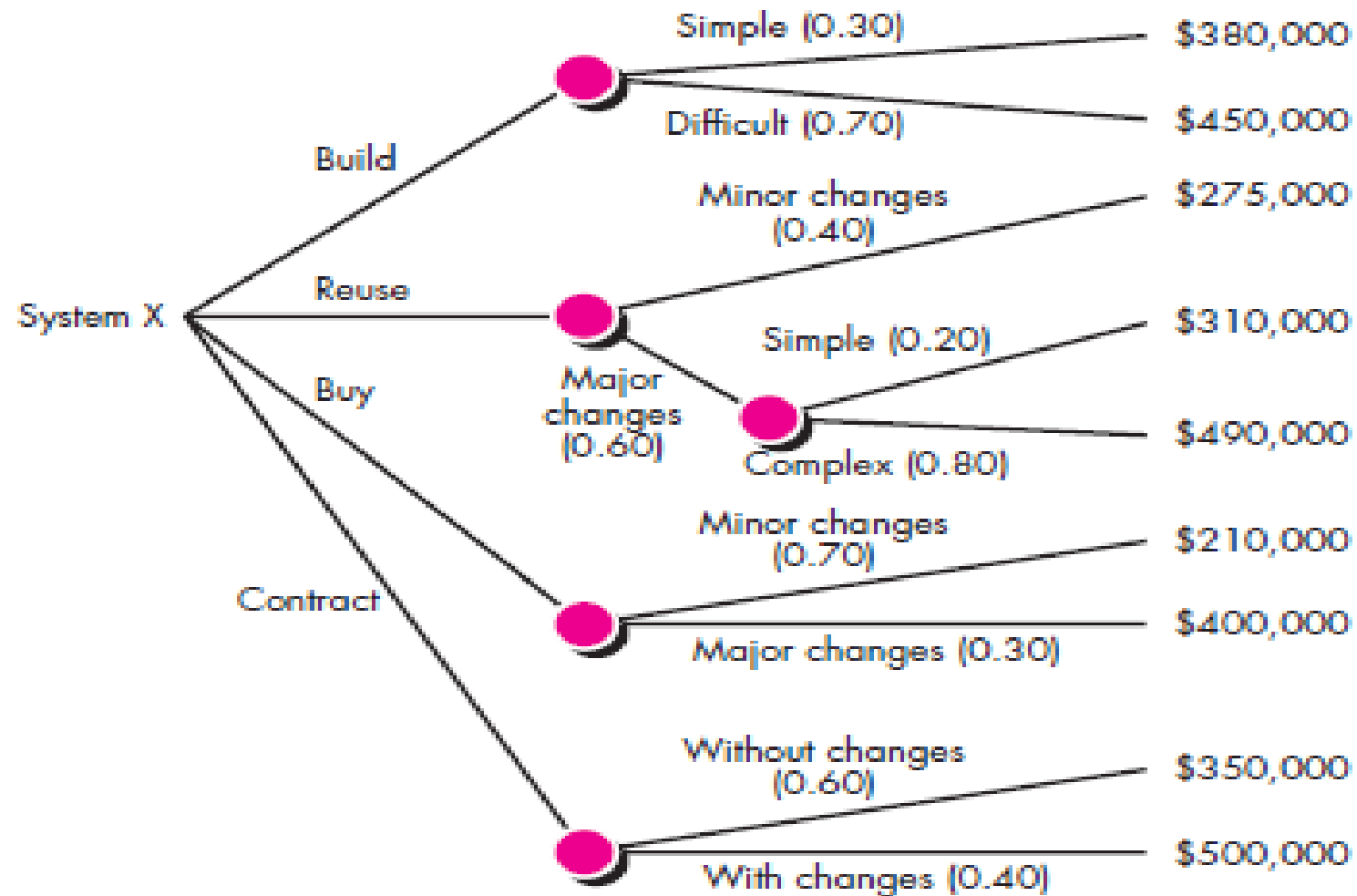
COMPUTING EXPECTED COST FROM DECISION TREE

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

For example, the expected cost to build is: expected cost = 0.30 ($380K) + 0.70 ($450K) = **$429 K**

similarly,

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

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

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

MMH

REFERENCES

- R.S. Pressman & Associates, Inc. (2010). *Software Engineering: A Practitioner's Approach.*

- Kelly, J. C., Sherif, J. S., & Hops, J. (1992). An analysis of defect densities found during software inspections. *Journal of Systems and Software, 17*(2), 111-117.

- Bhandari, I., Halliday, M. J., Chaar, J., Chillarege, R., Jones, K., Atkinson, J. S., & Yonezawa, M. (1994). In-process improvement through defect data interpretation. *IBM Systems Journal, 33*(1), 182-214.