

Estimation for Software Projects

- Project planning
- Scope and feasibility
- Project resources
- Estimation of project cost and effort
- Decomposition techniques
- Empirical estimation models

For non-profit educational use only

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach*, 7/e. Any other reproduction or use is prohibited without the express written permission of the author.

These slides are designed to accompany *Software Engineering: A Practitioner's Approach*, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

Some of the contents are taken from other sources. Those are explicitly indicated

Project Planning

Software Project Planning

- Software project planning encompasses five major activities
 - Estimation, scheduling, risk analysis, quality management planning, and change management planning
- Estimation determines how much money, effort, resources, and time it will take to build a specific system or product
- The software team first estimates
 - The work to be done
 - The resources required
 - The time that will elapse from start to finish
- Then they establish a project schedule that
 - Defines tasks and milestones
 - Identifies who is responsible for conducting each task
 - Specifies the inter-task dependencies

Observations on Estimation

- Planning requires technical managers and the software team to make an initial commitment
- Process and project metrics can provide a historical perspective and valuable input for generation of quantitative estimates
- Past experience can aid greatly
- Estimation carries inherent risk, and this risk leads to uncertainty. Risk increases with
 - Project Complexity
 - Project Size
 - The degree of structural uncertainty
- The availability of historical information has a strong influence on estimation risk

(More on next slide)

Observations on Estimation (continued)

- When software metrics are available from past projects
 - Estimates can be made with greater assurance
 - Schedules can be established to avoid past difficulties
 - Overall risk is reduced
- Estimation risk is measured by the degree of uncertainty in the quantitative estimates for cost, schedule, and resources
- Nevertheless, a project manager should not become obsessive about estimation
 - Plans should be iterative and allow adjustments as time passes and more is made certain

"It is the mark of an instructed mind to rest satisfied with the degree of precision that the nature of the subject admits, and not to seek exactness when only an approximation of the truth is possible." ARISTOTLE

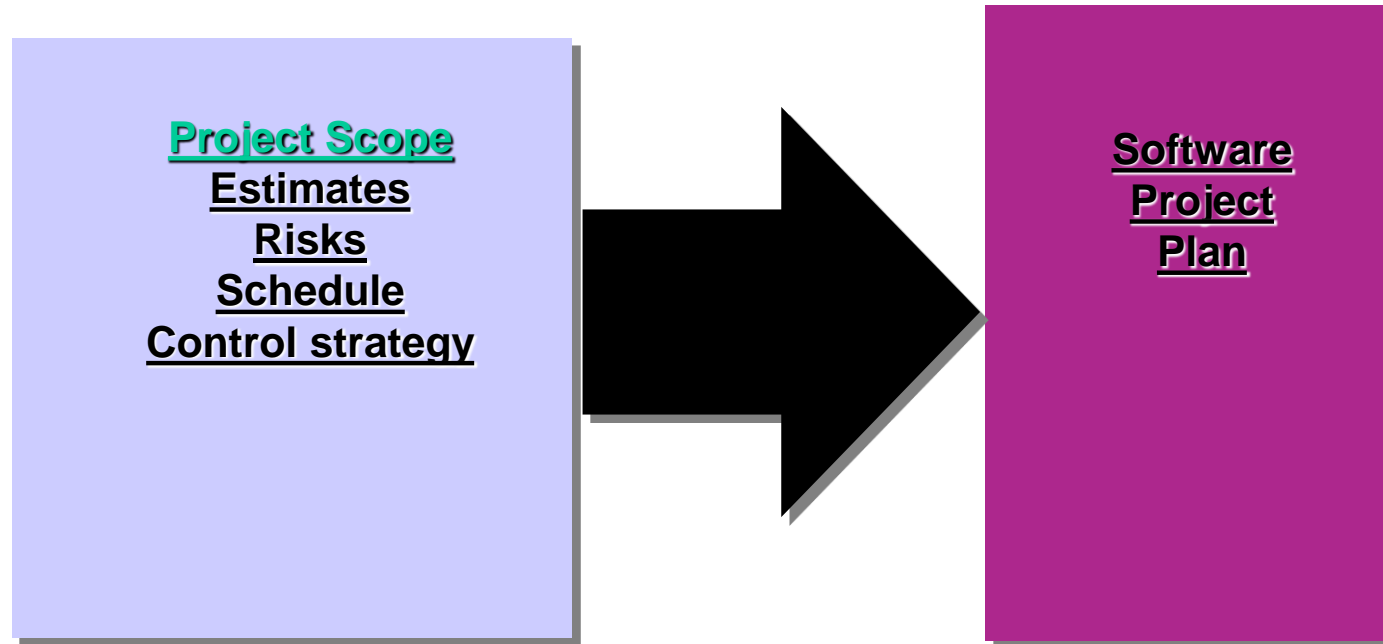
Project Planning Task Set-I

- 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 next session.
 - Establish a meaningful task set
 - Define a task network
 - Use scheduling tools to develop a timeline chart
 - Define schedule tracking mechanisms

In Short...



Scope and Feasibility

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.

Software Scope (continued)

- After the scope has been identified, two questions are asked
 - Can we build software to meet this scope?
 - Is the project feasible?
- Software engineers too often rush (or are pushed) past these questions
- Later they become mired in a project that is doomed from the onset


Feasibility

- After the scope is resolved, feasibility is addressed
- Software feasibility has four dimensions, (as per Putnam & Meyer)
 - **Technology** – Is the project technically feasible? Is it within the state of the art? Can defects be reduced to a level matching the application's needs?
 - **Finance** – Is it financially feasible? Can development be completed at a cost that the software organization, its client, or the market can afford?
 - **Time** – Will the project's time-to-market beat the competition?
 - **Resources** – Does the software organization have the resources needed to succeed in doing the project?

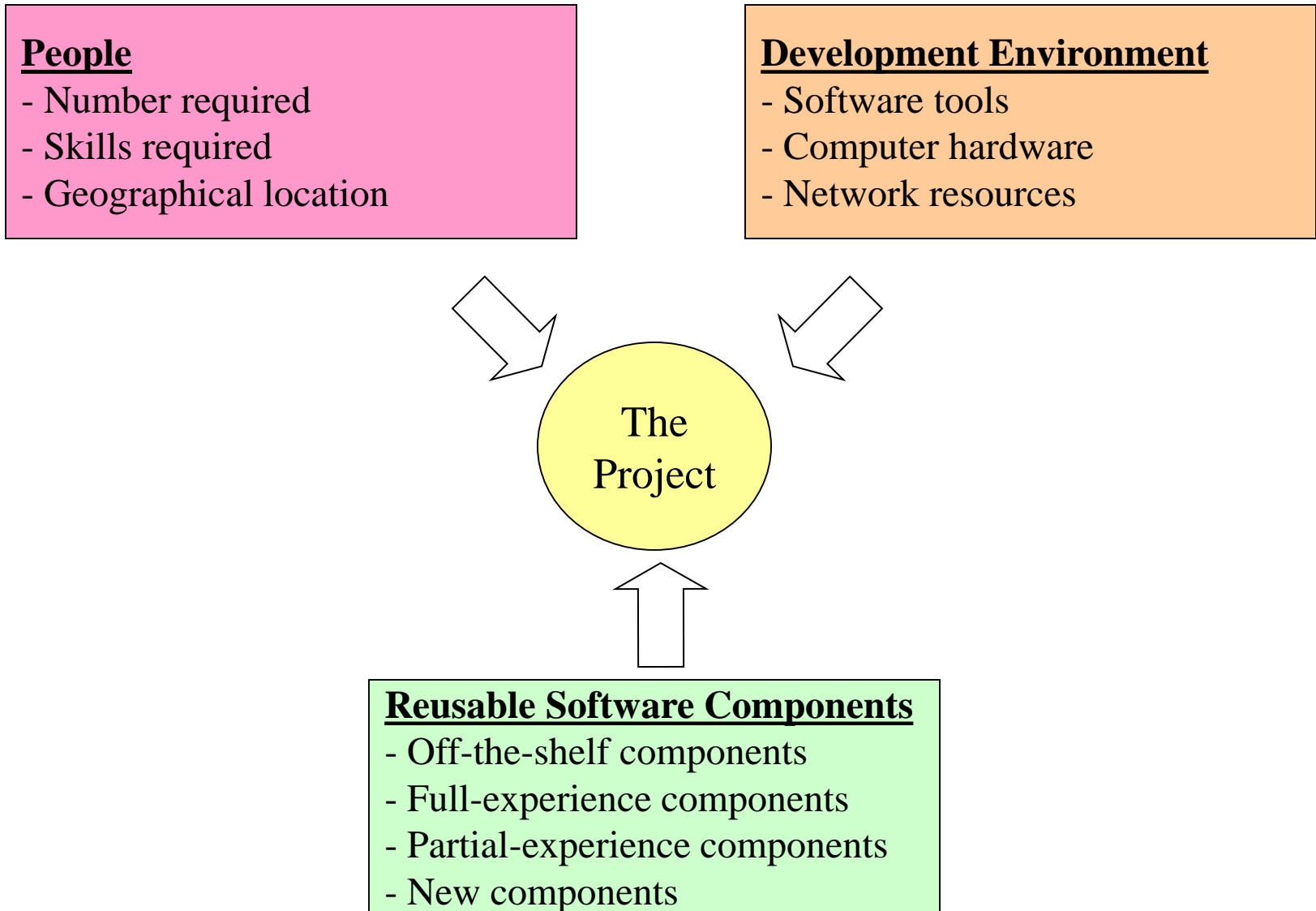
Another view recommends the following feasibility dimensions: technological, economical, **legal**, **operational**, and schedule issues (TELOS)

Project Resources

Resource Estimation

- Three major categories of software engineering resources
 - People
 - Development environment
 - Reusable software components
 - Often neglected during planning but become a paramount concern during the construction phase of the software process
 - Each resource is specified with
 - A description of the resource
 - A statement of availability
 - The time when the resource will be required
 - The duration of time that the resource will be applied
- 
- Time window

Categories of Resources



Human Resources

- Planners need to select the number and the kind of people skills needed to complete the project
- They need to specify the organizational position and job specialty for each person
- Small projects of a few person-months may only need one individual
- Large projects spanning many person-months or years require the location of the person to be specified also
- The number of people required can be determined only after an estimate of the development effort

Development Environment Resources

- A software engineering environment (SEE) incorporates hardware, software, and network resources that provide platforms and tools to develop and test software work products
- Most software organizations have many projects that require access to the SEE provided by the organization
- Planners must identify the time window required for hardware and software and verify that these resources will be available

Reusable Software Resources

- Off-the-shelf components
 - Components are from a third party or were developed for a previous project
 - Ready to use; fully validated and documented; virtually no risk
- Full-experience components
 - Components are similar to the software that needs to be built
 - Software team has full experience in the application area of these components
 - Modification of components will incur relatively low risk
- Partial-experience components
 - Components are related somehow to the software that needs to be built but will require substantial modification
 - Software team has only limited experience in the application area of these components
 - Modifications that are required have a fair degree of risk
- New components
 - Components must be built from scratch by the software team specifically for the needs of the current project
 - Software team has no practical experience in the application area
 - Software development of components has a high degree of risk

– E. M. Bennatan

Estimation of Project Cost and Effort

Estimation Techniques

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

Factors Affecting Project Estimation

- The accuracy of a software project estimate is predicated on
 - The degree to which the planner has properly estimated the size (e.g., KLOC) of the product to be built
 - The ability to translate the size estimate into human effort, calendar time, and money
 - The degree to which the project plan reflects the abilities of the software team
 - The stability of both the product requirements and the environment that supports the software engineering effort

Project Estimation Options

- Options for achieving reliable cost and effort estimates
 - 1) Delay estimation until late in the project (we should be able to achieve 100% accurate estimates after the project is complete)
 - 2) Base estimates on similar projects that have already been completed
 - 3) Use relatively simple decomposition techniques to generate project cost and effort estimates
 - 4) Use one or more empirical estimation models for software cost and effort estimation
- Option #1 is not practical, but results in good numbers
- Option #2 can work reasonably well, but it also relies on other project influences being roughly equivalent
- Options #3 and #4 can be done in tandem to cross check each other

Project Estimation Approaches

- Decomposition techniques
 - These take a "divide and conquer" approach
 - Cost and effort estimation are performed in a stepwise fashion by breaking down a project into major functions and related software engineering activities
- Empirical estimation models
 - Offer a potentially valuable estimation approach if the historical data used to seed the estimate is good

Decomposition Techniques

Introduction

- Before an estimate can be made and decomposition techniques applied, the planner must
 - Understand the scope of the software to be built
 - Generate an estimate of the software's size
- Then one of two approaches are used
 - Problem-based estimation
 - Based on either source lines of code or function point estimates
 - Process-based estimation
 - Based on the effort required to accomplish each task

Approaches to Software Sizing

- Function point sizing
 - Develop estimates of the information domain characteristics
 - Standard component sizing
 - Estimate the number of occurrences of each standard component
 - Use historical project data to determine the delivered LOC size per standard component
 - Change sizing
 - Used when changes are being made to existing software
 - Estimate the number and type of modifications that must be accomplished
 - Types of modifications include reuse, adding code, changing code, and deleting code
 - An effort ratio is then used to estimate each type of change and the size of the change
 - Fuzzy Logic sizing
 - Requires the planner to derive size from understanding of qualitative aspects
- Putnam & Myers

The results of these estimates are used to compute an optimistic (low), a most likely, and a pessimistic (high) value for software size

Problem-Based Estimation

- 1) Start with a bounded statement of scope
- 2) Decompose the software into problem functions that can each be estimated individually
- 3) Compute an LOC or FP value for each function
- 4) Derive cost or effort estimates by applying the LOC or FP values to your baseline productivity metrics (e.g., LOC/person-month or FP/person-month)
- 5) Combine function estimates to produce an overall estimate for the entire project

(More on next slide)

Problem-Based Estimation (continued)

- In general, the LOC/pm and FP/pm metrics should be computed by project domain
 - Important factors are team size, application area, and complexity
- LOC and FP estimation differ in the level of detail required for decomposition with each value
 - For LOC, decomposition of functions is essential and should go into considerable detail (the more detail, the more accurate the estimate)
 - For FP, decomposition occurs for the five information domain characteristics and the 14 adjustment factors
 - External inputs, external outputs, external inquiries, internal logical files, external interface files

pm = person month

Problem-Based Estimation (continued)

- For both approaches, the planner uses lessons learned to estimate an optimistic, most likely, and pessimistic size value for each function or count (for each information domain value)
- Then the expected size value S is computed as follows:

$$S = (S_{\text{opt}} + 4S_{\text{m}} + S_{\text{pess}}) / 6$$

- Historical LOC or FP data is then compared to S in order to cross-check it

Example: LOC Approach

Function	Estimated LOC
user interface and control facilities (UICF)	2,300
two-dimensional geometric analysis (2D GA)	5,300
three-dimensional geometric analysis (3D GA)	6,800
database management (DBM)	3,300
computer graphics display facilities (CGDF)	4,900
peripheral control (PC)	2,100
design analysis modules (DAM)	8,400
<i>estimated lines of code</i>	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	poss.	est. count	weight	FP-count
number of inputs	20	24	30	24	4	97
number of outputs	12	18	22	16	8	78
number of inquiries	16	22	28	22	8	88
number of files	4	4	8	4	10	42
number of external interfaces	2	2	3	2	7	18
count-total						321

The estimated number of FP is derived:

$$FP_{\text{estimated}} = \text{count-total} * [0.65 + 0.01 * \text{Sum}(F_i)]$$

Assuming $\text{Sum}(F_i)$ for the System is 52, $FP_{\text{estimated}} = 375$

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.

Does the system require reliable backup and recovery? - 4

Are specialized data communications required to transfer information to or from the application? - 2

Are there distributed processing functions? - 0

Is performance critical? - 4

Will the system run in an existing, heavily utilized operational environment? - 3

Does the system require on-line data entry? - 4

Does the on-line data entry require the input transaction to be built over multiple screens or operations? - 5

Are the internal logical files updated on-line? - 3

Are the inputs, outputs, files, or inquiries complex? - 5

Is the internal processing complex? - 5

Is the code designed to be reusable? - 4

Are conversion and installation included in the design? - 3

Is the system designed for multiple installations in different organizations? - 5

Is the application designed to facilitate change and for ease of use by the user? - 5

Process-Based Estimation

- 1) Identify the set of functions that the software needs to perform as obtained from the project scope
- 2) Identify the series of framework activities that need to be performed for each function
- 3) Estimate the effort (in person months) that will be required to accomplish each software process activity for each function

(More on next slide)

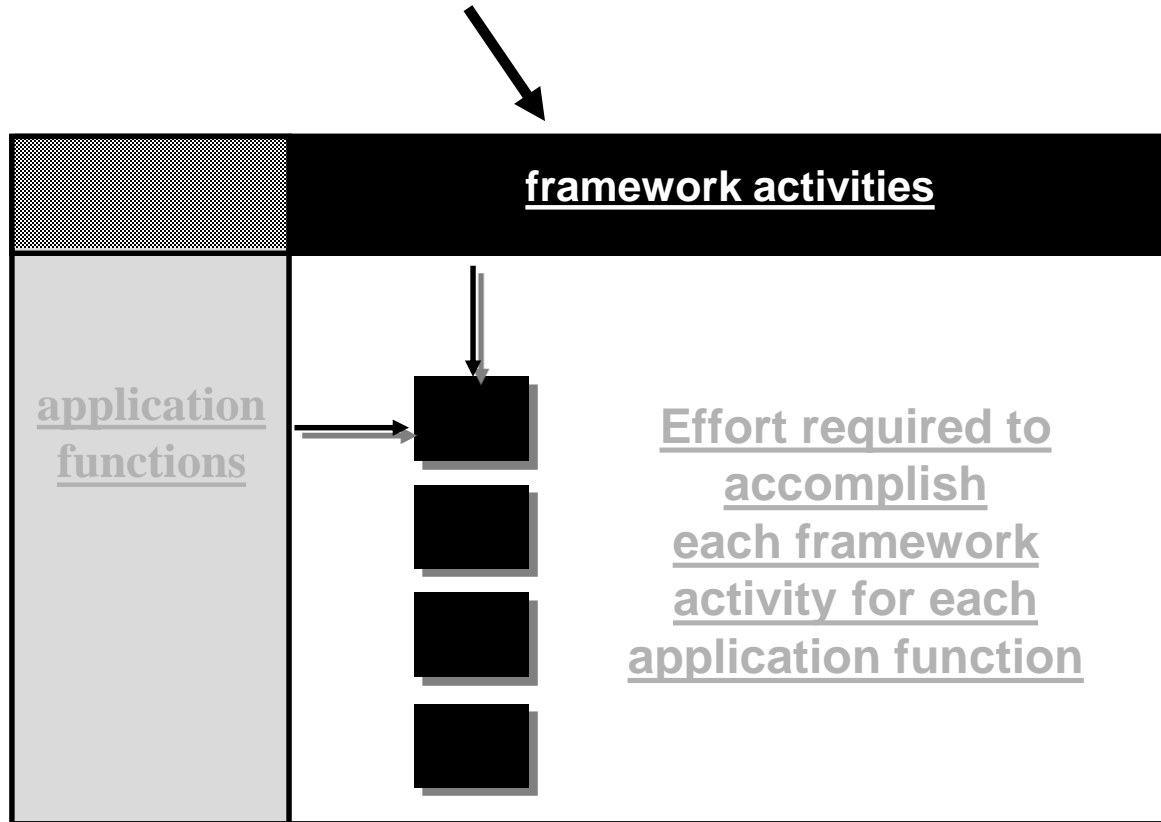
Process-Based Estimation (continued)

- 4) Apply average labor rates (i.e., cost/unit effort) to the effort estimated for each process activity
- 5) Compute the total cost and effort for each function and each framework activity
- 6) Compare the resulting values to those obtained by way of the LOC and FP estimates
 - If both sets of estimates agree, then your numbers are highly reliable
 - Otherwise, conduct further investigation and analysis concerning the function and activity breakdown

This is the most commonly used of the two estimation techniques (problem and process)

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.

Estimation with Use Cases

Use Cases are widely used. But estimation with use cases is problematic

- Many different formats and Styles

- Many levels of abstraction

- Do not specify complexity of functions

Smith(Rational Corp) suggests using “structural hierarchy” of use cases, scenarios for types of subsystems (e.g. real-time, business, scientific etc.)

Estimation with Use Cases (contd)

An empirical formula for estimation can be

LOC estimate =

$$N * LOC_{avg} + [(S_a/S_h - 1) + (P_a/P_h - 1)] * LOC_{adjust}$$

Where

N = actual no of use cases

LOC_{avg} = historical LOC based on use case of this type

S_a and S_h = actual and historical scenarios per use case

P_a and P_h = actual and historical pages per use case

LOC_{adjust} = adjustment factor (a fraction of LOC_{avg})

Estimation with Use-Cases (example)

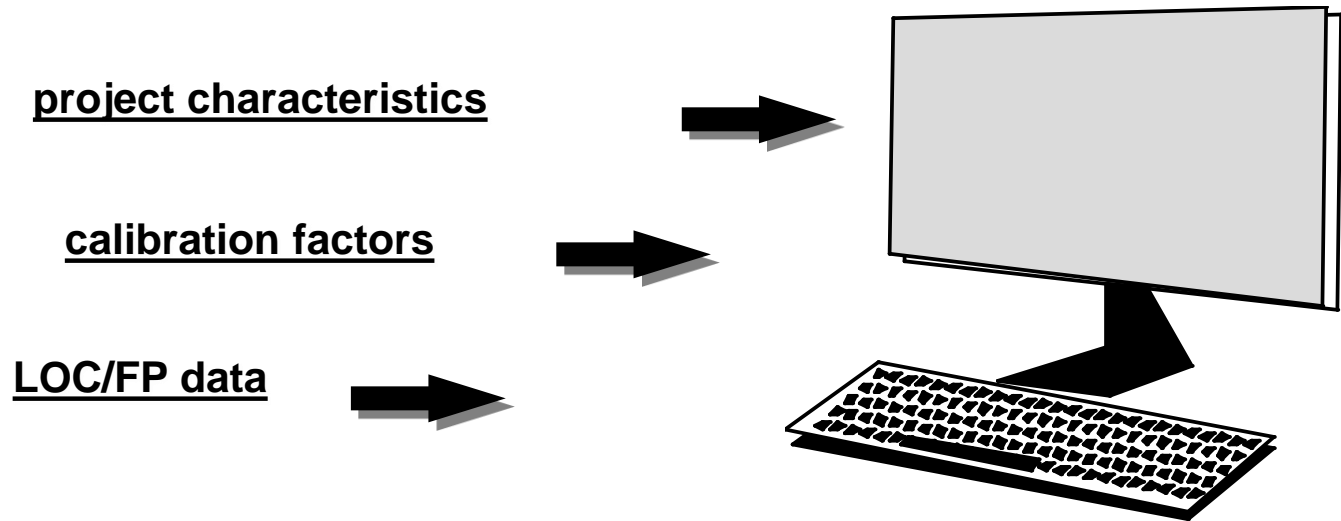
				Historical Information			
	Use cases	Scenarios	Pages	Scenarios	Pages	LOC	LOC estimate
Subsystem1	6	10	6	12	5	560	3366
Subsystem2	10	20	8	16	8	3100	31233
Subsystem3	5	6	5	10	6	1650	7970
Total LOC estimate							42568

For Subsystem1, LOC estimate without adjustment = $6 \times 560 = 3360$

Adjustment count with 0.3 multiplier = $(10/12 - 1 + 6/5 - 1) \times 560 \times 0.3 \approx 6$

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.**

Tool-Based Estimation



Reconciling Estimates

- The results gathered from the various estimation techniques must be reconciled to produce a single estimate of effort, project duration, and cost
- If widely divergent estimates occur, investigate the following causes
 - The scope of the project is not adequately understood or has been misinterpreted by the planner
 - Productivity data used for problem-based estimation techniques is inappropriate for the application, obsolete (i.e., outdated for the current organization), or has been misapplied
- The planner must determine the cause of divergence and then reconcile the estimates

State of the Art for estimating Large Systems

- Formal sizing approaches based on FP
- Secondary sizing approach based on LOC
- Tertiary sizing approaches using information such as screens, reports etc.
- Inclusion of reusable materials in the estimates
- Inclusion of supply chains if multiple companies are involved
- Inclusion of travel costs for international distributed teams
- Comparison with historical benchmark data

Rank Order of Large System Software Cost Elements

1. Defect Removal (inspections, static analysis, testing, finding, and fixing bugs)
2. Producing paper documents (plans, architecture, specifications, user manuals)
3. Meetings and communication (clients, team members, managers)
4. Programming
5. Project Management

Rank Order of Agile Software Cost Elements

1. Programming
2. Meetings and communication (clients, team members, managers)
3. Defect Removal (inspections, static analysis, testing, finding, and fixing bugs)
4. Project Management
5. Producing paper documents (plans, architecture, specifications, user manuals)

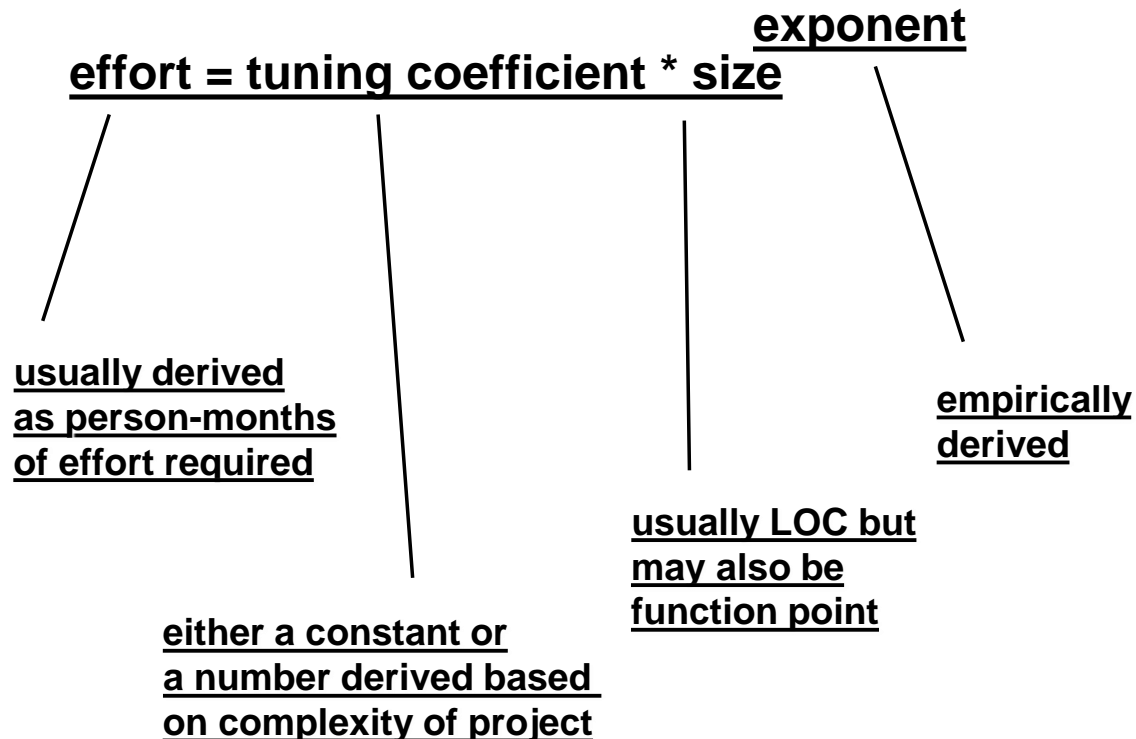
Empirical Estimation Models

Introduction

- Several estimation models for computer software use empirically derived formulas to predict effort as a function of LOC or FP
- Resultant values computed for LOC or FP are entered into an estimation model
- The empirical data for these models are derived from a limited sample of projects
 - Consequently, the models should be calibrated to reflect local software development conditions

Empirical Estimation Models

General form:



COCOMO

- Stands for COnstructive COst MOdel
- Introduced by Barry Boehm in 1981 in his book “Software Engineering Economics”
- Became one of the well-known and widely-used estimation models in the industry
- It has evolved into a more comprehensive estimation model called COCOMO II
- COCOMO II is actually a hierarchy of three estimation models
- As with all estimation models, it requires sizing information and accepts it in three forms: object points, function points, and lines of source code

(More on next slide)

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.

Project Modes (project types)

- Organic
 - Relatively small software teams develop software in a highly familiar, in-house environment
- Semi-detached
 - between organic and embedded
- Embedded
 - Needs to operate within tight constraints. The product must operate within (is embedded in) a strongly coupled complex of hardware, software, regulations, and operational procedures.

Basic Effort Formula

$$E = a \times \text{Size}^b$$

E = person-months

Size = KLOC

	Organic	Semi	Embedded
a	2.4	3.0	3.6
b	1.05	1.12	1.20

Cost Drivers

	very low				extra high			
Product attributes								
Required software reliability	0.75	0.88	1.00	1.15	1.40			
Size of application database		0.94	1.00	1.08	1.16			
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65		
Hardware attributes								
Run-time performance constraints	1.00	1.11	1.30	1.66				
Memory constraints		1.00	1.06	1.21	1.56			
Virtual machine environment volatility		0.87	1.00	1.15	1.30			
Required turnaround time	0.87	1.00	1.07	1.15				

Cost Drivers

	very low			extra high	
Personnel attributes					
Analyst capability	1.46	1.19	1.00	0.86	0.71
Software engineer capability	1.29	1.13	1.00	0.91	0.82
Applications experience	1.42	1.17	1.00	0.86	0.70
Virtual machine experience	1.21	1.10	1.00	0.90	
Programming language experience	1.14	1.07	1.00	0.95	
Project attributes					
Use of software tools	1.24	1.10	1.00	0.91	0.82
Application of SwEng methods	1.24	1.10	1.00	0.91	0.83
Required development schedule	1.23	1.08	1.00	1.04	1.10

Effort Formula with Cost Drivers

$$E = a \times \text{Size}^b \times C$$

E = person-months

Size = KLOC

C = 15 Cost Drivers

	Organic	Semi	Embedded
a	3.2	3.0	2.8
b	1.05	1.12	1.20

The Software Equation

A dynamic multivariable model

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

where

E = effort in person-months or person-years

t = project duration in months or years

B = “special skills factor”

P = “productivity parameter”

- Putnam et al.

Estimation for OO Projects-I

-Lorenz & Kidd

- Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
- Using object-oriented requirements modeling, develop use-cases and determine a count.
- From the analysis model, determine the number of key classes.
- 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

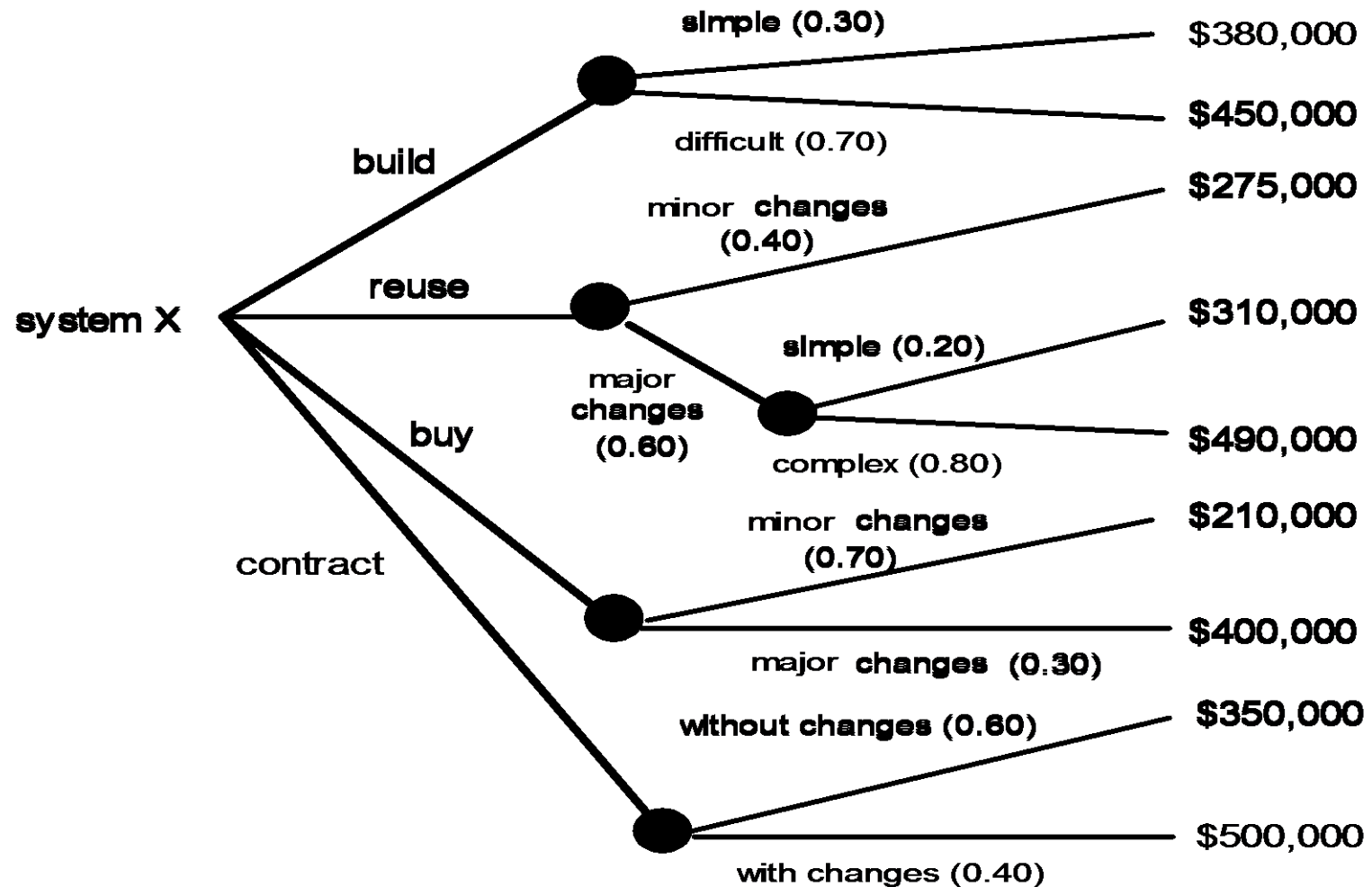
Estimation for Agile Projects

- 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.

Make/Buy Decision

- It is often more cost effective to acquire rather than develop software
- Managers have many acquisition options
 - Software may be purchased (or licensed) off the shelf
 - “Full-experience” or “partial-experience” software components may be acquired and integrated to meet specific needs
 - Software may be custom built by an outside contractor to meet the purchaser’s specifications
- The make/buy decision can be made based on the following conditions
 - Will the software product be available sooner than internally developed software?
 - Will the cost of acquisition plus the cost of customization be less than the cost of developing the software internally?
 - Will the cost of outside support (e.g., a maintenance contract) be less than the cost of internal support?

The Make-Buy Decision



Computing Expected Cost

expected cost =

$$\sum (\text{path probability}) \times (\text{estimated path cost})$$

For example, the expected cost to build is:

$$\begin{aligned} \text{expected cost}_{\text{build}} &= 0.30 (\$380\text{K}) + 0.70 (\$450\text{K}) \\ &= \underline{\$429 \text{ K}} \end{aligned}$$

similarly,

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

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

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

Summary of Estimation

- Software planning involves estimating how much time, effort, money, and resources will be required to build a specific software system.
- After the project scope is determined and the problem is decomposed into smaller problems, software managers use historical project data (as well as personal experience and intuition) to determine estimates for each.
- The final estimates are typically adjusted by taking project complexity and risk into account.
- Managers will not know that they have done a good job estimating until the project post mortem.
- It is essential to track resources and revise estimates as a project progresses