SCHEDULING & ESTIMATION

IT 3201: Software Engineering

Why Are Projects Late?

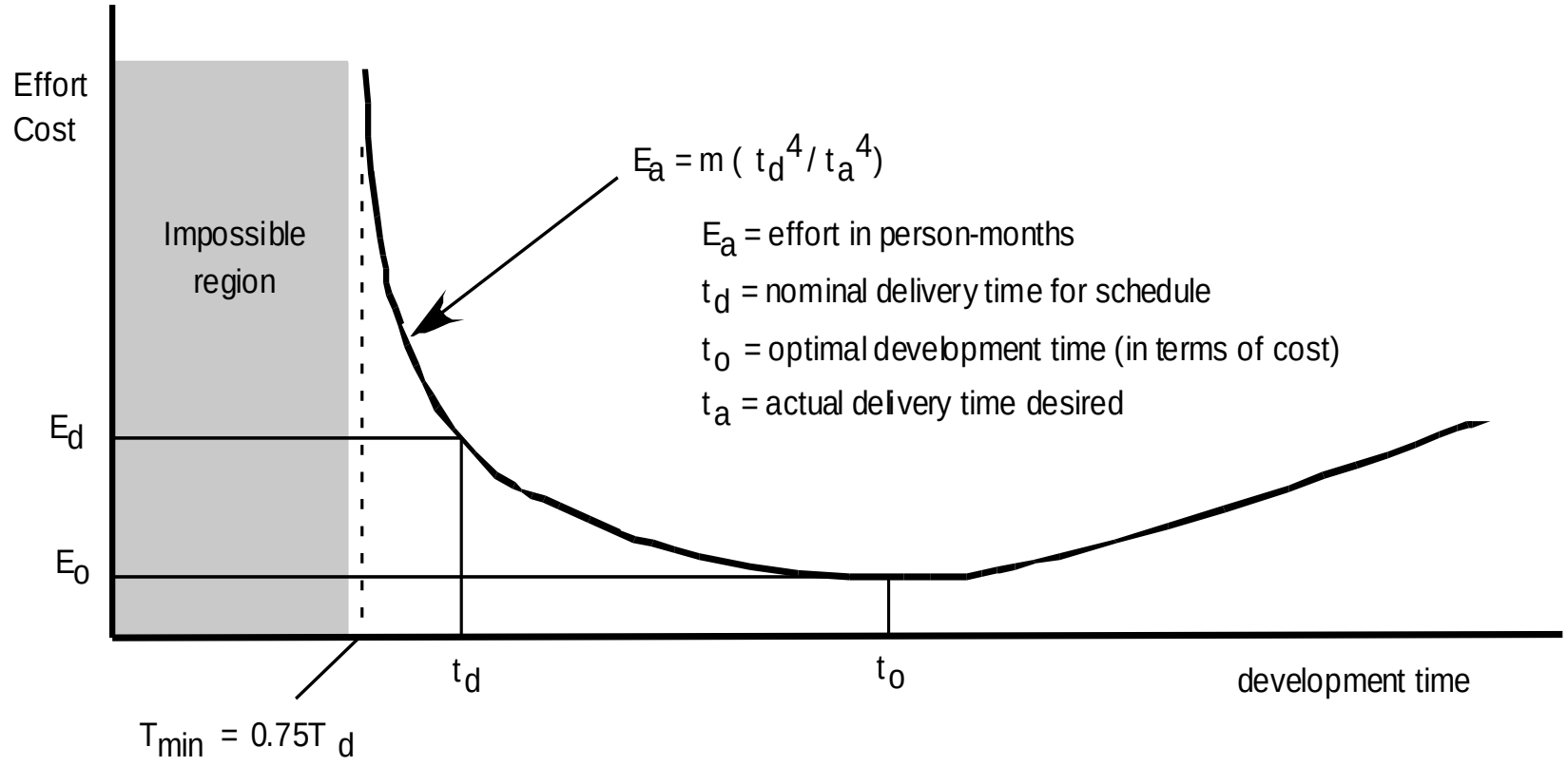
- an **unrealistic deadline** established by someone outside the software development group
- **changing customer requirements** that are not reflected in schedule changes;
- an **honest underestimate** of the amount of effort and/or the number of resources that will be required to do the job;
- **predictable and/or unpredictable risks** that were not considered when the project commenced;
- **technical difficulties** that could not have been foreseen in advance;
- **human difficulties** that could not have been foreseen in advance;
- **miscommunication** among project staff that results in delays;
- a failure by project management to recognize that **the project is falling behind schedule** and **a lack of action to correct the problem**

Project scheduling

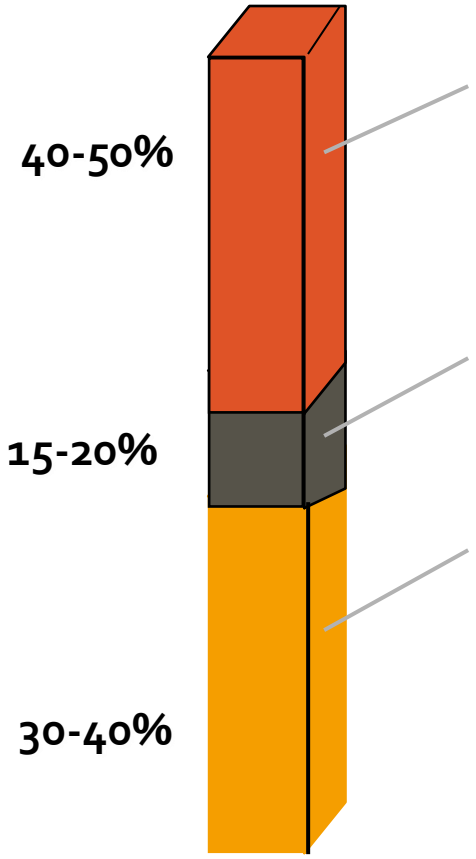
- Project scheduling is the process of deciding how the work in a project will be organized as separate tasks, and when and how these tasks will be executed.
- You estimate
  - the calendar time needed to complete each task,
  - the effort required
  - who will work on the tasks that have been identified.
- You also have to estimate the resources needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be.

The project scheduling process

Effort and Delivery Time



Effort Allocation

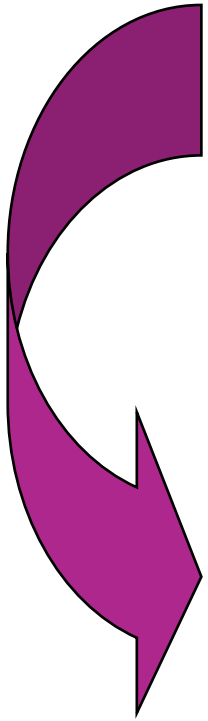


- "front end" activities
  - customer communication
  - analysis
  - design
  - review and modification
- construction activities
  - coding or code generation
- testing and installation
  - unit, integration
  - white-box, black box
  - regression

Defining Task Sets

- determine type of project
- assess the degree of rigor required
- identify adaptation criteria
- select appropriate software engineering tasks

Task Set Refinement



is refined to

1.1 **Concept scoping** determines the overall scope of the project.

Task definition: Task 1.1 Concept Scoping
- 1.1.1 Identify need, benefits and potential customers;
- 1.1.2 Define desired output/control and input events that drive the application;

  Begin Task 1.1.2
  - 1.1.2.1 FTR: Review written description of need

    FTR indicates that a formal technical review (Chapter 26) is to be conducted.
  - 1.1.2.2 Derive a list of customer visible outputs/inputs
  - 1.1.2.3 FTR: Review outputs/inputs with customer and revise as required;

  endtask Task 1.1.2
- 1.1.3 Define the functionality/behavior for each major function;

  Begin Task 1.1.3
  - 1.1.3.1 FTR: Review output and input data objects derived in task 1.1.2;
  - 1.1.3.2 Derive a model of functions/behaviors;
  - 1.1.3.3 FTR: Review functions/behaviors with customer and revise as required;

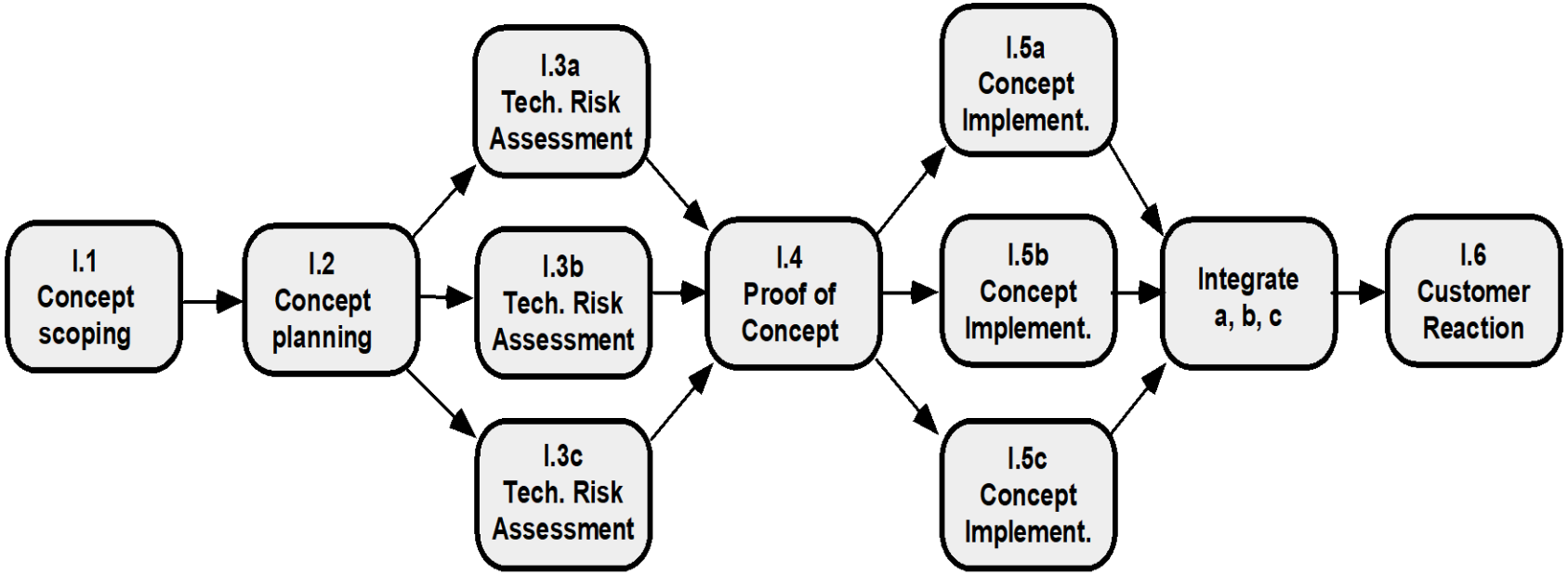
  endtask Task 1.1.3
- 1.1.4 Isolate those elements of the technology to be implemented in software;
- 1.1.5 Research availability of existing software;
- 1.1.6 Define technical feasibility;
- 1.1.7 Make quick estimate of size;
- 1.1.8 Create a Scope Definition;

endTask definition: Task 1.1

Schedule representation

- Graphical notations are normally used to illustrate the project schedule.
- These show the project breakdown into tasks. Tasks should not be too small. They should take about a week or two.
- Bar charts are the most commonly used representation for project schedules. They show the schedule as activities or resources against time.

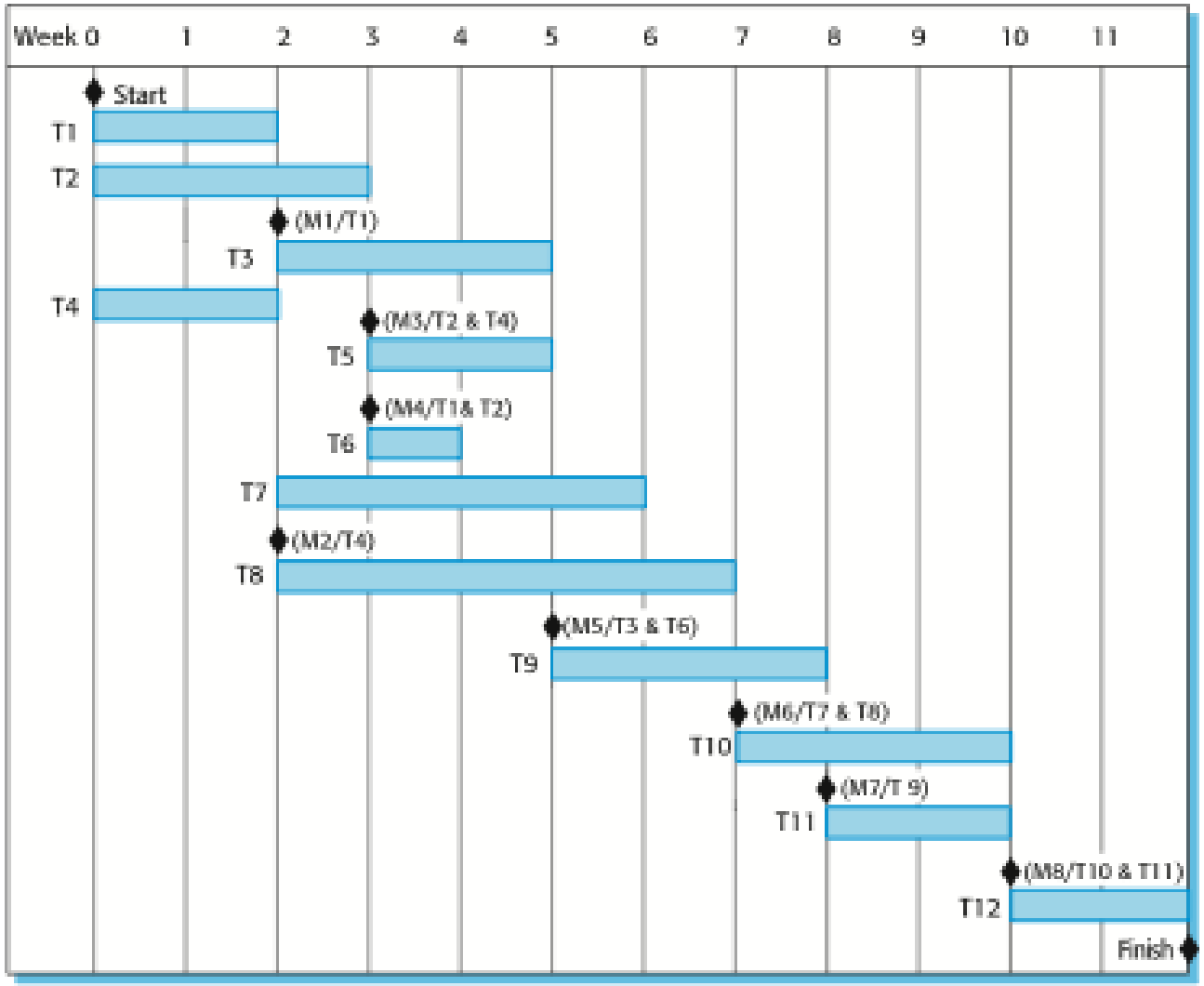
Define a Task Network



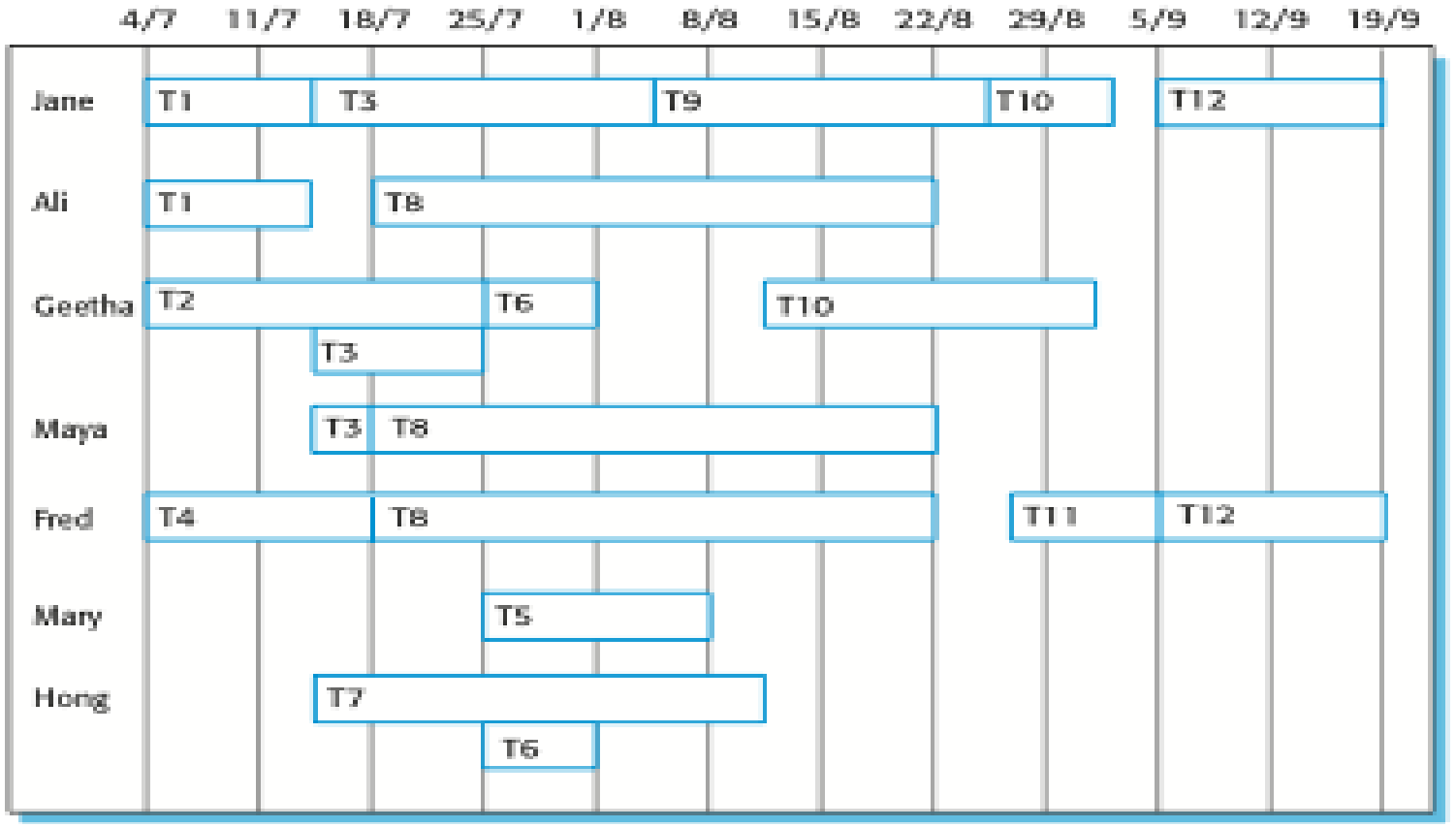
Tasks, durations, and dependencies

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

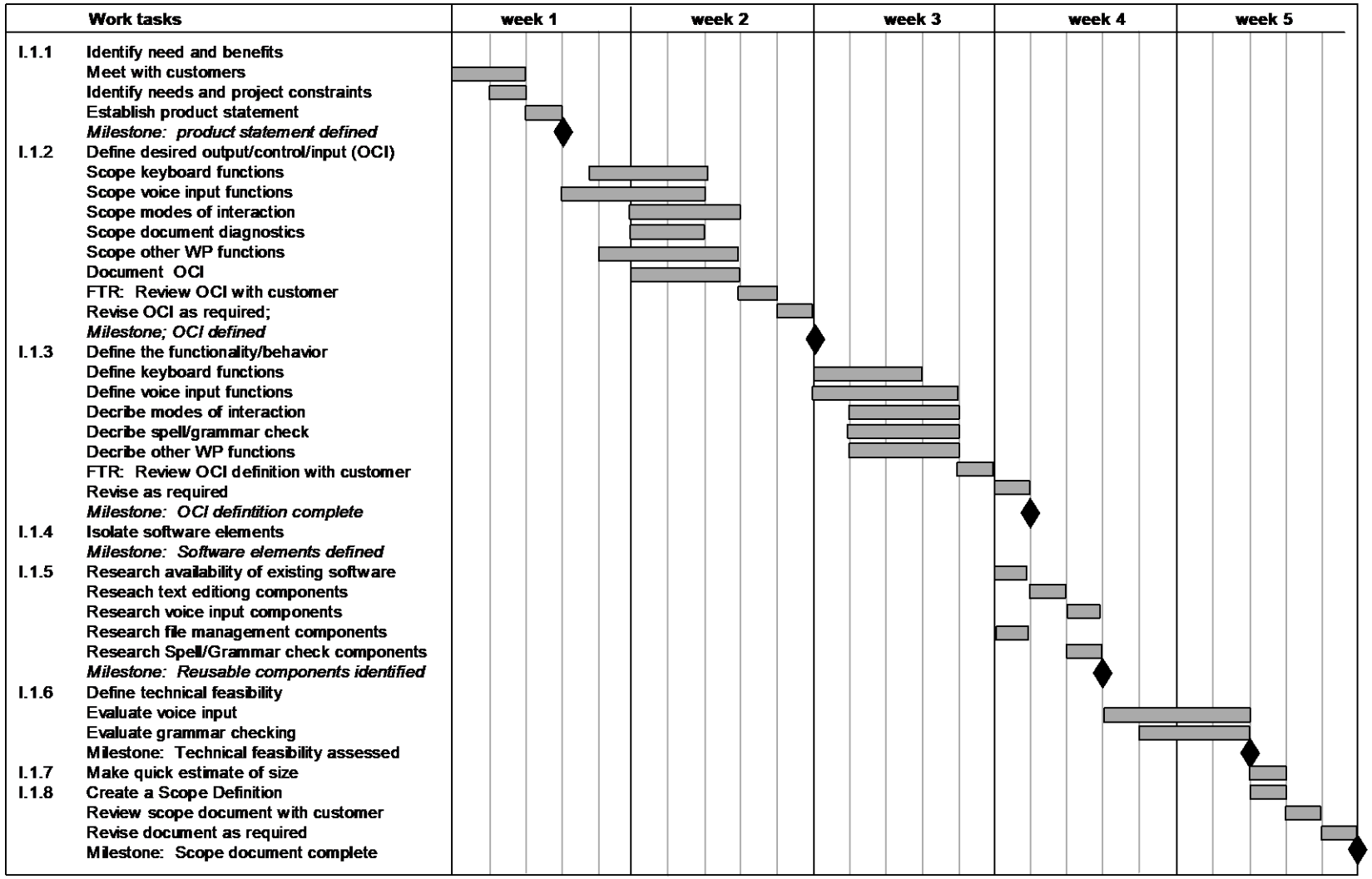
Activity bar chart



Staff allocation chart



Use Automated Tools to Derive a Timeline Chart



Progress on an OO Project-I

- *Technical milestone: OO analysis completed*
  - All classes and the class hierarchy have been defined and reviewed.
  - Class attributes and operations associated with a class have been defined and reviewed.
  - Class relationships (Chapter 8) have been established and reviewed.
  - A behavioral model (Chapter 8) has been created and reviewed.
  - Reusable classes have been noted.
- *Technical milestone: OO design completed*
  - The set of subsystems (Chapter 9) has been defined and reviewed.
  - Classes are allocated to subsystems and reviewed.
  - Task allocation has been established and reviewed.
  - Responsibilities and collaborations (Chapter 9) have been identified.
  - Attributes and operations have been designed and reviewed.
  - The communication model has been created and reviewed.

Schedule Tracking

- conduct periodic project status meetings in which each team member reports progress and problems.
- evaluate the results of all reviews conducted throughout the software engineering process.
- determine whether formal project milestones (the diamonds shown in Figure 27.3) have been accomplished by the scheduled date.
- compare actual start-date to planned start-date for each project task listed in the resource table
- meet informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.
- use earned value analysis to assess progress quantitatively.

Progress on an OO Project-II

- *Technical milestone: OO programming completed*
  - Each new class has been implemented in code from the design model.
  - Extracted classes (from a reuse library) have been implemented.
  - Prototype or increment has been built.
- *Technical milestone: OO testing*
  - The correctness and completeness of OO analysis and design models has been reviewed.
  - A class-responsibility-collaboration network (Chapter 6) has been developed and reviewed.
  - Test cases are designed and class-level tests (Chapter 19) have been conducted for each class.
  - Test cases are designed and cluster testing (Chapter 19) is completed and the classes are integrated.
  - System level tests have been completed.

Earned Value Analysis (EVA)

- Earned value
  - is a measure of progress
  - enables us to assess the "percent of completeness" of a project using quantitative analysis rather than rely on a gut feeling
  - "provides accurate and reliable readings of performance from as early as 15 percent into the project." [Fle98]

Computing Earned Value-I

- The *budgeted cost of work scheduled* (BCWS) is determined for each work task represented in the schedule.
  - $BCWS_i$ is the effort planned for work task *i*.
  - To determine progress at a given point along the project schedule, the value of BCWS is the sum of the $BCWS_i$ values for all work tasks that should have been completed by that point in time on the project schedule.
- The BCWS values for all work tasks are summed to derive the *budget at completion,* BAC. Hence,

$$BAC = \sum (BCWS_k) \text{ for all tasks } k$$

Computing Earned Value-II

- Next, the value for *budgeted cost of work performed* (BCWP) is computed.
  - The value for BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.
- "the distinction between the BCWS and the BCWP is that the former represents the budget of the activities that were planned to be completed and the latter represents the budget of the activities that actually were completed." [Wil99]
- Given values for BCWS, BAC, and BCWP, important progress indicators can be computed:
  - Schedule performance index, SPI = BCWP/BCWS
  - Schedule variance, SV = BCWP – BCWS
  - SPI is an indication of the efficiency with which the project is utilizing scheduled resources.

Computing Earned Value-III

- Percent scheduled for completion = BCWS/BAC
  - provides an indication of the percentage of work that should have been completed by time *t.*
- Percent complete = BCWP/BAC
  - provides a quantitative indication of the percent of completeness of the project at a given point in time, *t.*
- *Actual cost of work performed*, ACWP, is the sum of the effort actually expended on work tasks that have been completed by a point in time on the project schedule. It is then possible to compute
  - Cost performance index, CPI = BCWP/ACWP
  - Cost variance, CV = BCWP – ACWP

Estimation techniques

- Organizations need to make software effort and cost estimates. There are two types of technique that can be used to do this:
  - ***Experience-based techniques*** The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort requirements are likely to be.
  - ***Algorithmic cost modeling*** In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.

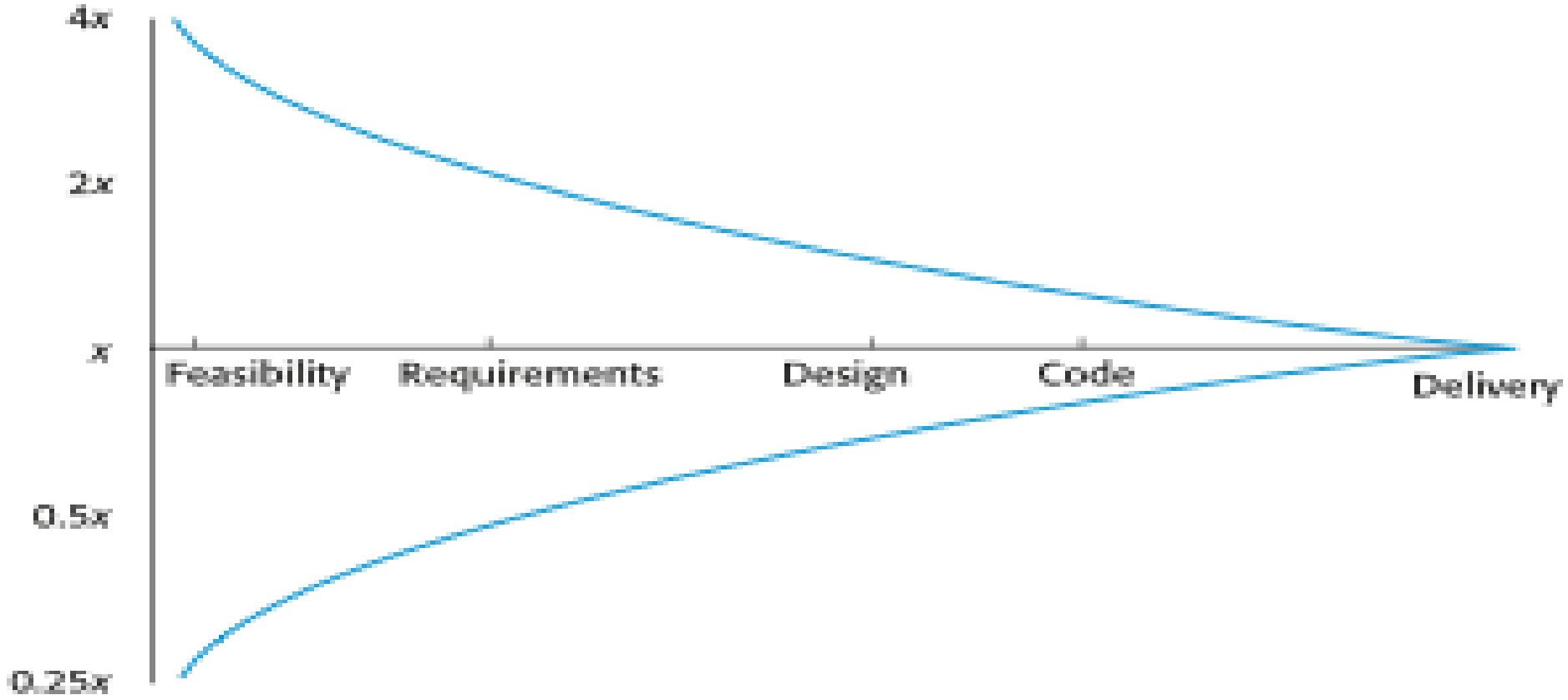
Experience-based approaches

- Experience-based techniques rely on judgments based on experience of past projects and the effort expended in these projects on software development activities.
- Typically, you identify the deliverables to be produced in a project and the different software components or systems that are to be developed.
- You document these in a spreadsheet, estimate them individually and compute the total effort required.
- It usually helps to get a group of people involved in the effort estimation and to ask each member of the group to explain their estimate.

Algorithmic cost modelling

- Cost is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers:
  - $\text{Effort} = A \times \text{Size}^B \times M$
  - A is an organisation-dependent constant, B reflects the disproportionate effort for large projects and M is a multiplier reflecting product, process and people attributes.
- The most commonly used product attribute for cost estimation is code size.
- Most models are similar but they use different values for A, B and M.

Estimate uncertainty



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*

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	96
number of outputs	12	15	22	16	5	80
number of inquiries	16	22	28	22	4	88
number of files	4	4	5	4	10	40
number of external interfaces	2	2	3	2	7	14
count-total						**318**

$FP_{estimated}$ = count-total × [0.65 + 0.01 × S (F_i)]

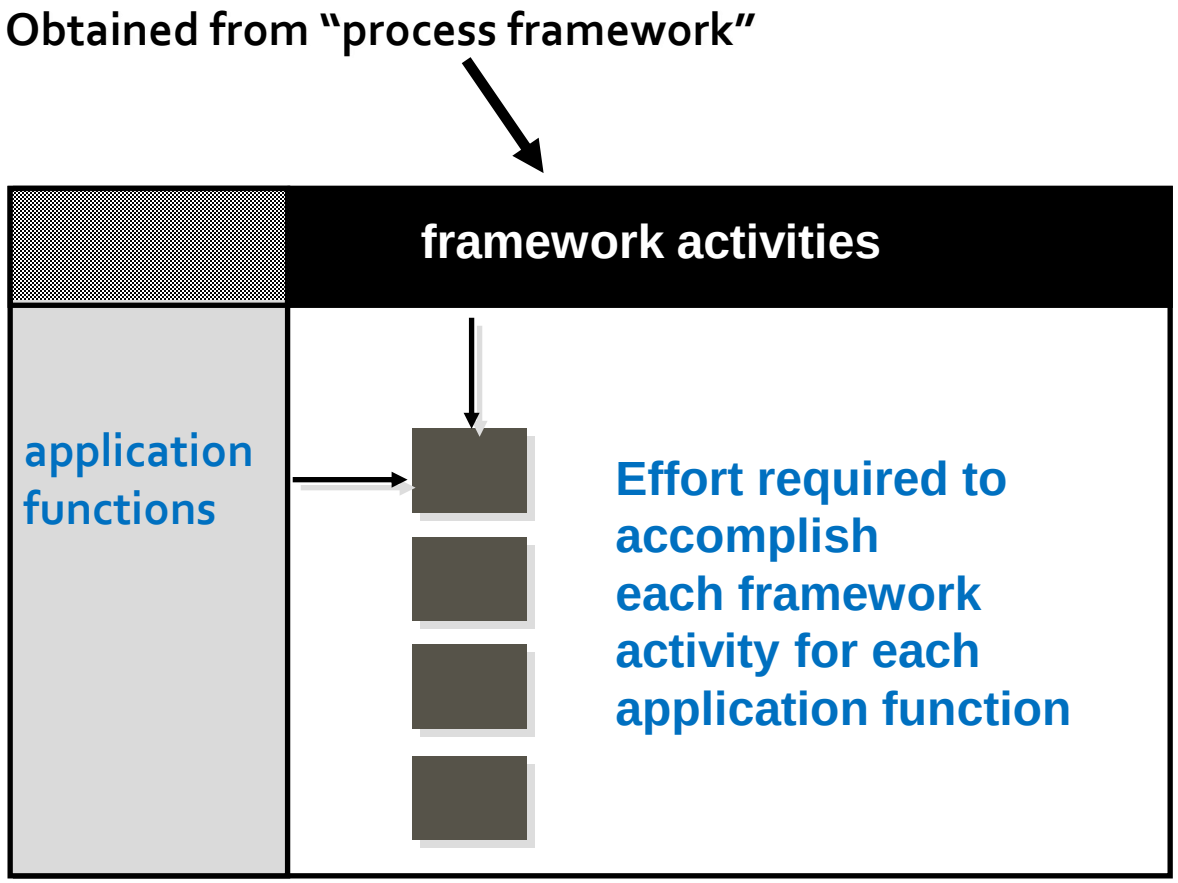
$FP_{estimated}$ = 372

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 $457,560 and estimated effort is 49 person-months.**

Process-Based Estimation



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

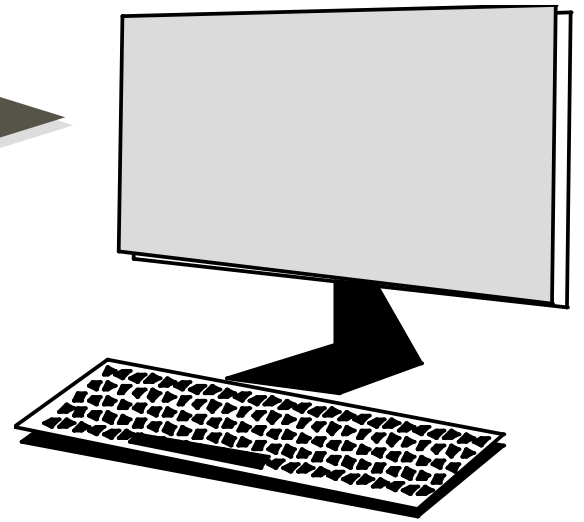
Based on an average burdened labor rate of $8,000 per month, **the total estimated project cost is $368,000 and the estimated effort is 46 person-months.**

Tool-Based Estimation

project characteristics

calibration factors

LOC/FP data



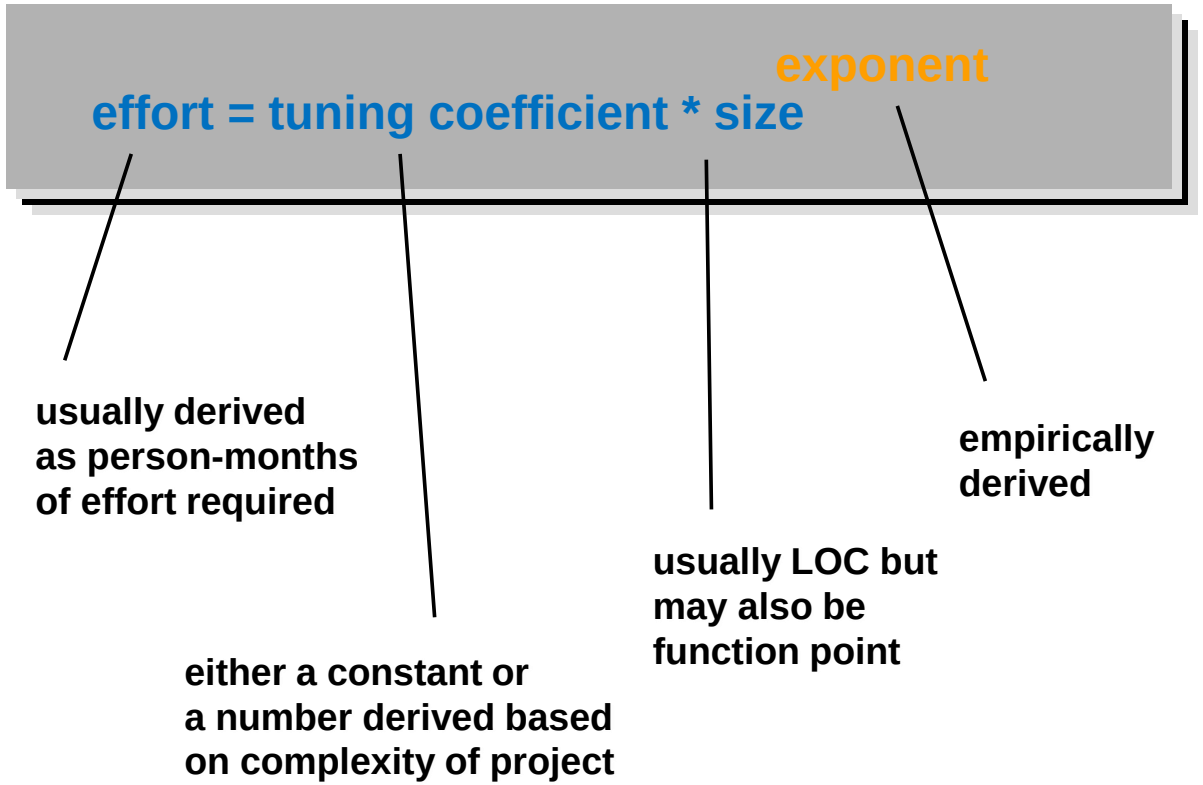
Estimation with Use-Cases

	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”

The effect of cost drivers on effort estimates

Exponent value	**1.17**
System size (including factors for reuse and requirements volatility)	128,000 DSI
Initial COCOMO estimate without cost drivers	**730 person-months**
Reliability	Very high, multiplier = 1.39
Complexity	Very high, multiplier = 1.3
Memory constraint	High, multiplier = 1.21
Tool use	Low, multiplier = 1.12
Schedule	Accelerated, multiplier = 1.29
Adjusted COCOMO estimate	**2,306 person-months**

The effect of cost drivers on effort estimates

Exponent value	**1.17**
Reliability	Very low, multiplier = 0.75
Complexity	Very low, multiplier = 0.75
Memory constraint	None, multiplier = 1
Tool use	Very high, multiplier = 0.72
Schedule	Normal, multiplier = 1
Adjusted COCOMO estimate	**295 person-months**

Project duration and staffing

- As well as effort estimation, managers must estimate the calendar time required to complete a project and when staff will be required.
- Calendar time can be estimated using a COCOMO 2 formula
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
  - PM is the effort computation and B is the exponent computed as discussed above (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project.
- The time required is independent of the number of people working on the project.

Staffing requirements

- Staff required can’t be computed by diving the development time by the required schedule.
- The number of people working on a project varies depending on the phase of the project.
- The more people who work on the project, the more total effort is usually required.
- A very rapid build-up of people often correlates with schedule slippage.

Reference

- Sommerville, Chapter 23 – Project planning, 9th edition
- Pressman, 7th edition
  - Chapter 27- Project Scheduling
  - Chapter 26 - Estimation for Software Projects

THANK YOU