UNIT-5

Managing Software Project

The management spectrum: The People, The Product and The Process.

The project planning process, Project resources, Software Project Estimation, Decomposition techniques, Empirical Estimation Models.

Project Scheduling: Basic concepts and Principles, Defining a task network, Scheduling.

Software Project Management Concepts

The Four P’s

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

Stakeholders

- *Senior managers* who define the business issues that often have significant influence on the project.
- *Project (technical) managers* who must plan, motivate, organize, and control the practitioners who do software work.
- *Practitioners* who deliver the technical skills that are necessary to engineer a product or application.
- *Customers* who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- *End-users* who interact with the software once it is released for production use.

Software Teams

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

Team Leader

- The MOI Model
  - **Motivation.** The ability to encourage technical people to produce to their best ability.

  - **Organization.** The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.

  - **Ideas or innovation.** The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Software Teams

The following factors must be considered when selecting a software project team structure ...

- the difficulty of the problem to be solved
- the size of the resultant program(s) in LOC or function points
- the time that the team will stay together (team lifetime)
- the degree to which the problem can be modularized
- the required quality and reliability of the system to be built
- the rigidity of the delivery date
- the degree of sociability (communication) required for the project

FOUR KEY TRAITS: AN EFFECTIVE PROJECT MANAGER

- **Problem solving:**
  - diagnose the technical and organizational issues
  - systematically structure a solution or properly motivate other practitioners to develop the solution
  - apply lessons learned from past projects to new situations.
  - remain flexible enough to change direction if initial attempts at problem solution

- **Managerial identity: .**
  - must take charge of the project.
  - must have the confidence to assume control when necessary
  - the assurance to allow good technical people to follow their instincts.

FOUR KEY TRAITS….

- **Achievement:**
  - A competent manager must reward initiative to get productivity of a project team.
  - Must demonstrate through own actions that controlled risk taking will not be punished.

- **Influence and team building:**
  - must be able to “read” people;
  - must be able to understand verbal and nonverbal signals
  - react to the needs of the people sending these signals.
  - The manager must remain under control in high-stress situations.

Team Coordination & Communication issues

- *Formal, impersonal approaches* include software engineering documents and work products, technical memos, project milestones, schedules, and project control tools ,change requests & related documentation, error tracking reports & repository data.
- *Formal, interpersonal procedures* focus on quality assurance activities applied to SE work products. These include status review meetings and design and code inspections.
- *Informal, interpersonal procedures* include group meetings for information dissemination and problem solving and collocation of requirements.
- *Electronic communication* electronic mail, electronic bulletin boards, & by extension, video-based conferencing systems.
- *Interpersonal networking* includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

The Product Scope

- Scope
  - **Context.** How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
  - **Information objectives.** What customer-visible data objects are produced as output from the software? What data objects are required for input?
  - **Function and performance.** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?
- Software project scope must be unambiguous and understandable at the management and technical levels.

Problem Decomposition

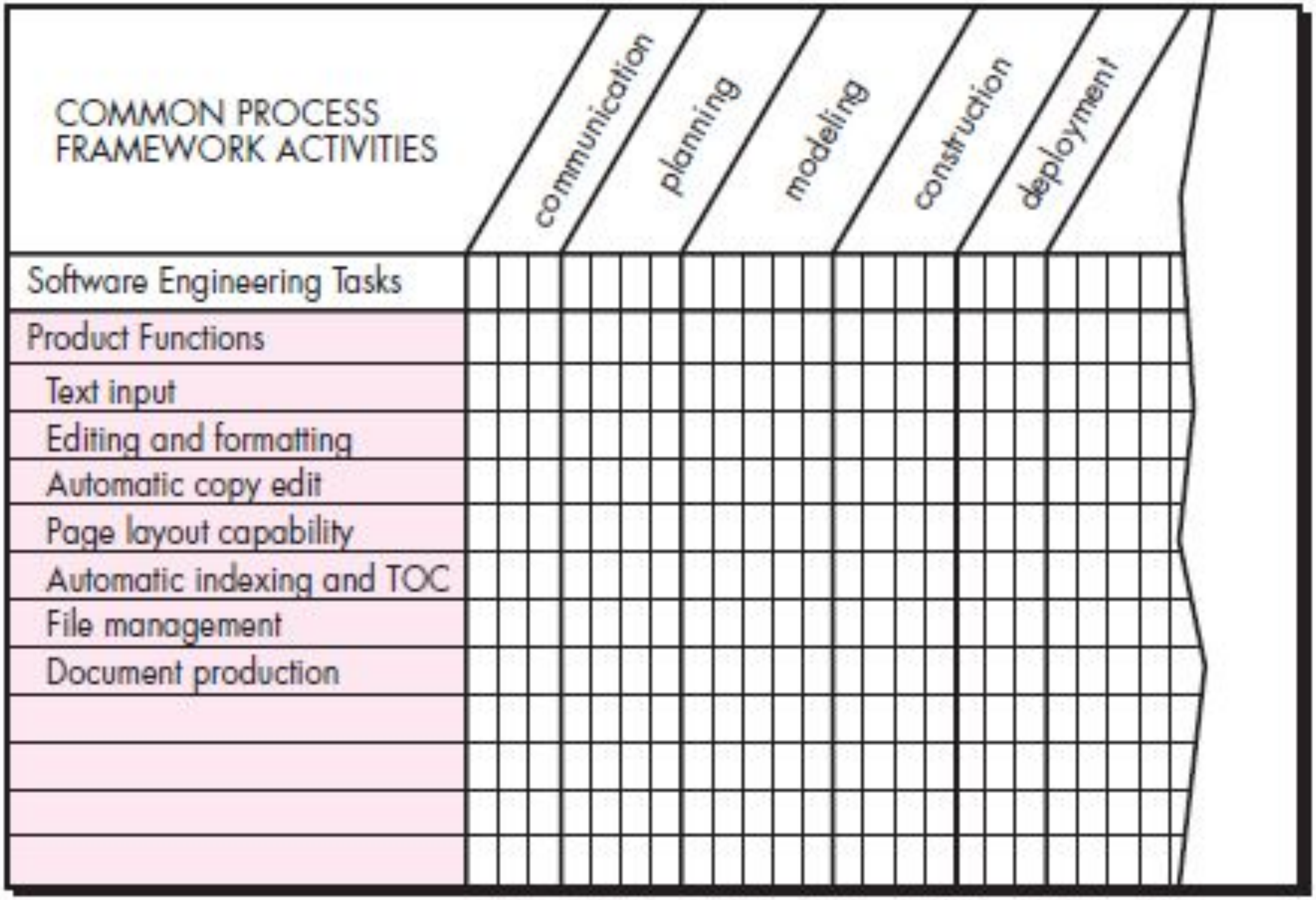
- Sometimes called *partitioning* or *problem elaboration*
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

  or
  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

The Process

- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

Melding the Problem and the Process



COMMON PROCESS FRAMEWORK ACTIVITIES	communication	planning	modeling	construction	deployment
Software Engineering Tasks					
Product Functions					
Text input					
Editing and formatting					
Automatic copy edit					
Page layout capability					
Automatic indexing and TOC					
File management					
Document production					

The Project

- *Projects get into trouble when …*
  - Software people don’t understand their customer’s needs.
  - The product scope is poorly defined.
  - Changes are managed poorly.
  - The chosen technology changes.
  - Business needs change [or are ill-defined].
  - Deadlines are unrealistic.
  - Users are resistant.
  - Sponsorship is lost [or was never properly obtained].
  - The project team lacks people with appropriate skills.
  - Managers [and practitioners] avoid best practices and lessons learned.

Common-Sense Approach to Projects

- *Start on the right foot.* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
- *Maintain momentum. The* project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
- *Track progress.* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
- *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
- *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

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

Estimation

- Estimation of
  - resources,
  - cost, and
  - schedule

  for a software engineering effort requires

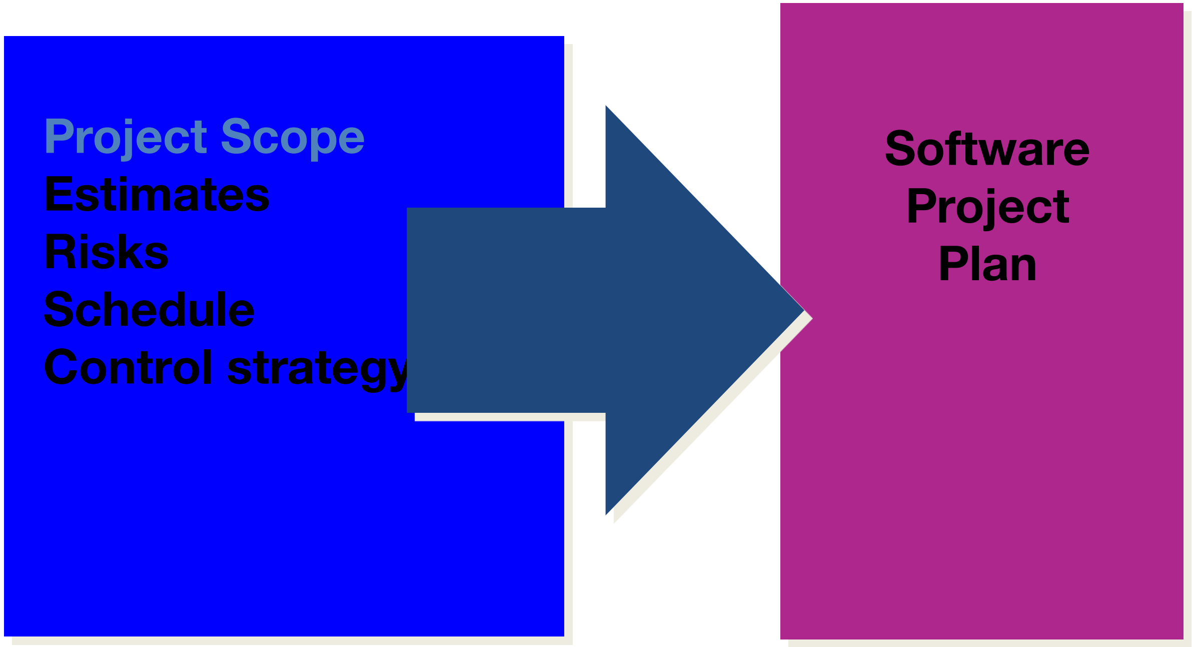
  > experience

  > access to good historical information (metrics)

  > the courage to commit to quantitative predictions when qualitative information is all that exists

- Estimation carries inherent risk and this risk leads to uncertainty

Write it Down!



Project Scope
Estimates
Risks
Schedule
Control strategy

Software Project Plan

To Understand Scope ...

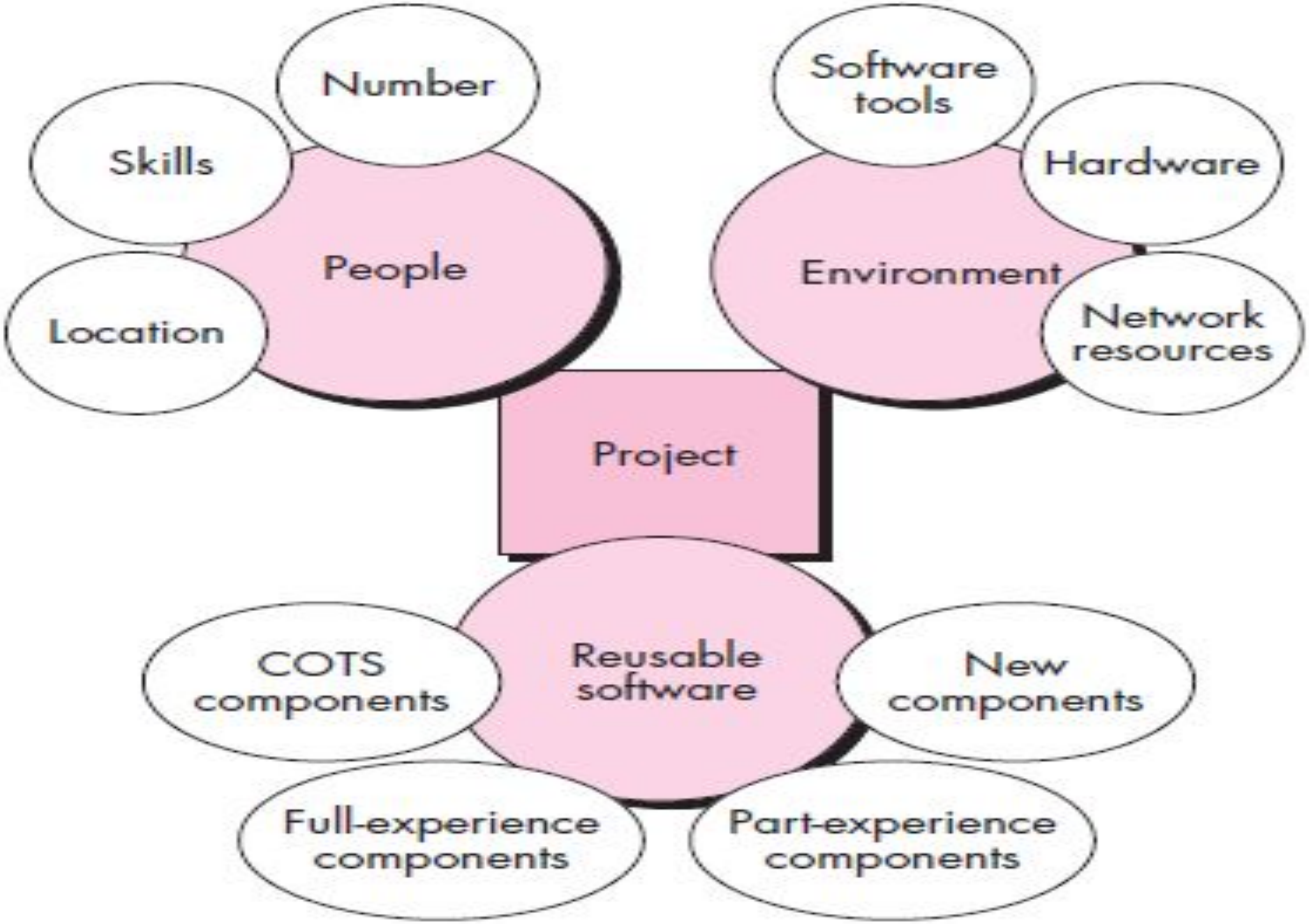
- Understand the customers needs
- understand the business context
- understand the project boundaries
- understand the customer’s motivation
- understand the likely paths for change
- understand that ...

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

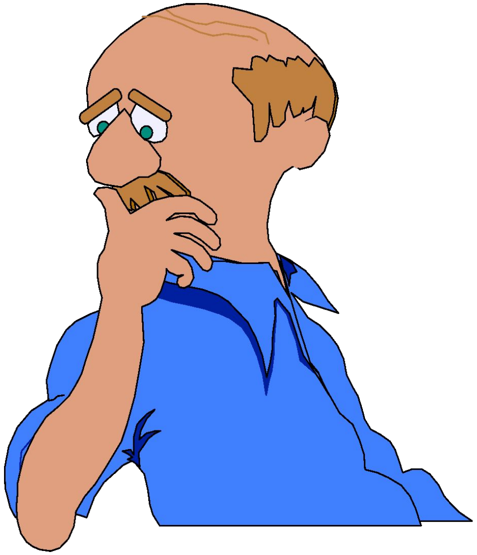

Number
Skills
People
Location
Software tools
Hardware
Environment
Network resources
Project
COTS components
Reusable software
New components
Full-experience components
Part-experience components

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

Estimation Techniques

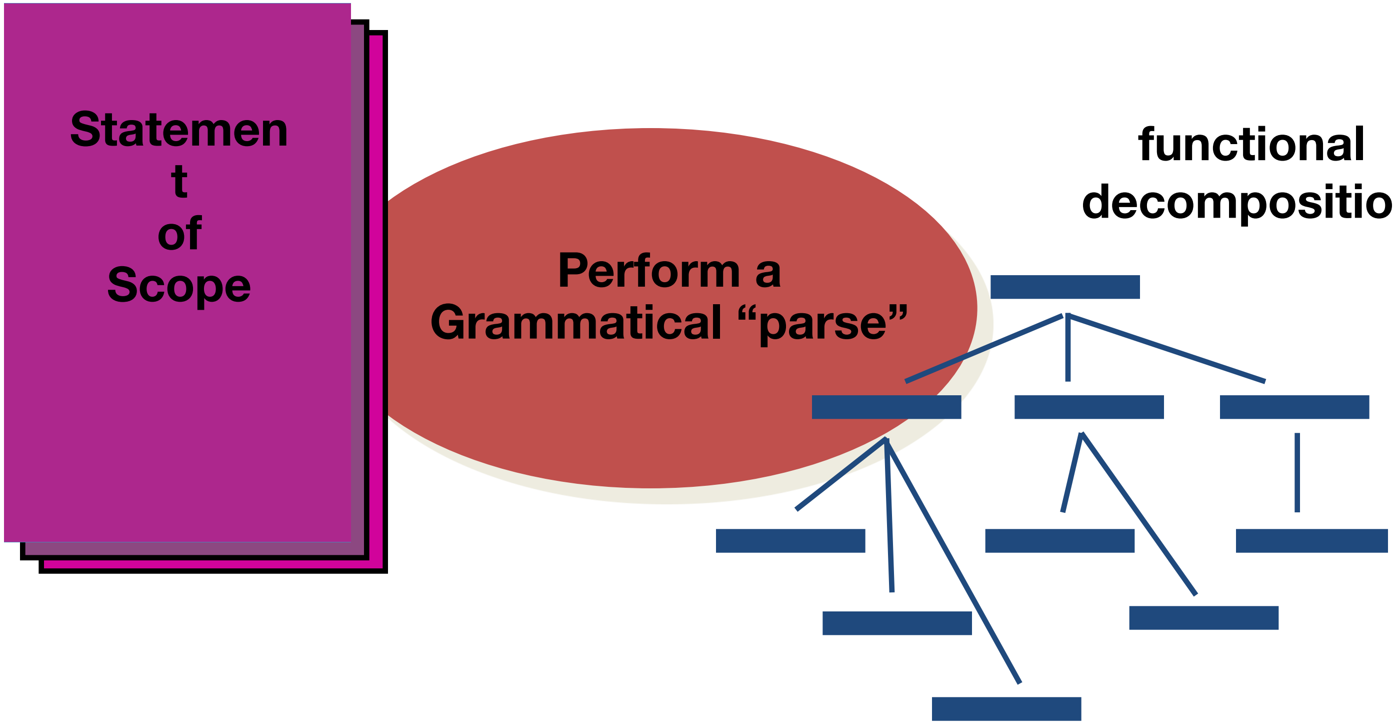
- Past (similar) project experience
- Different ways
  - Conventional estimation techniques
    - ✔ task breakdown and effort estimates
    - ✔ size (e.g., FP) estimates
  - Empirical models
  - Automated tools



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.

Functional Decomposition



Statemen
t
of
Scope

Perform a
Grammatical “parse”

functional
decomposition

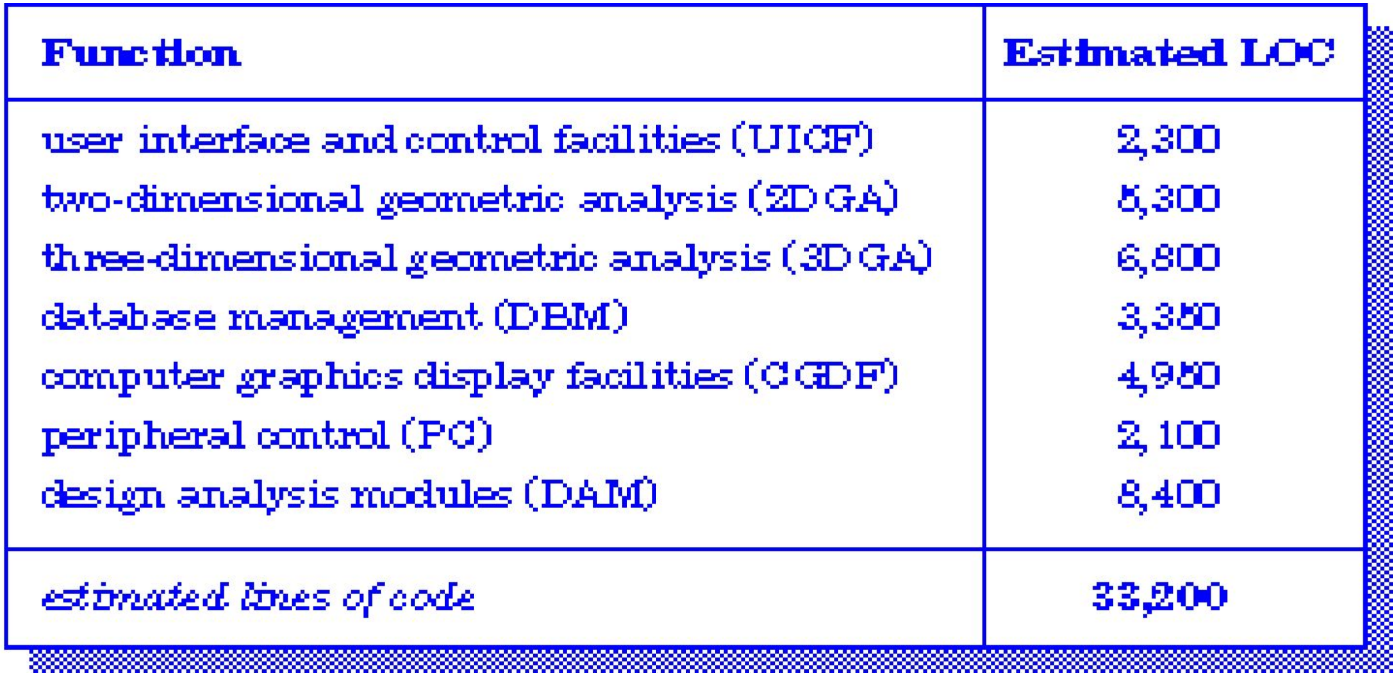
Conventional Methods:

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 (2D GA)	5,300
three-dimensional geometric analysis (3D GA)	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.

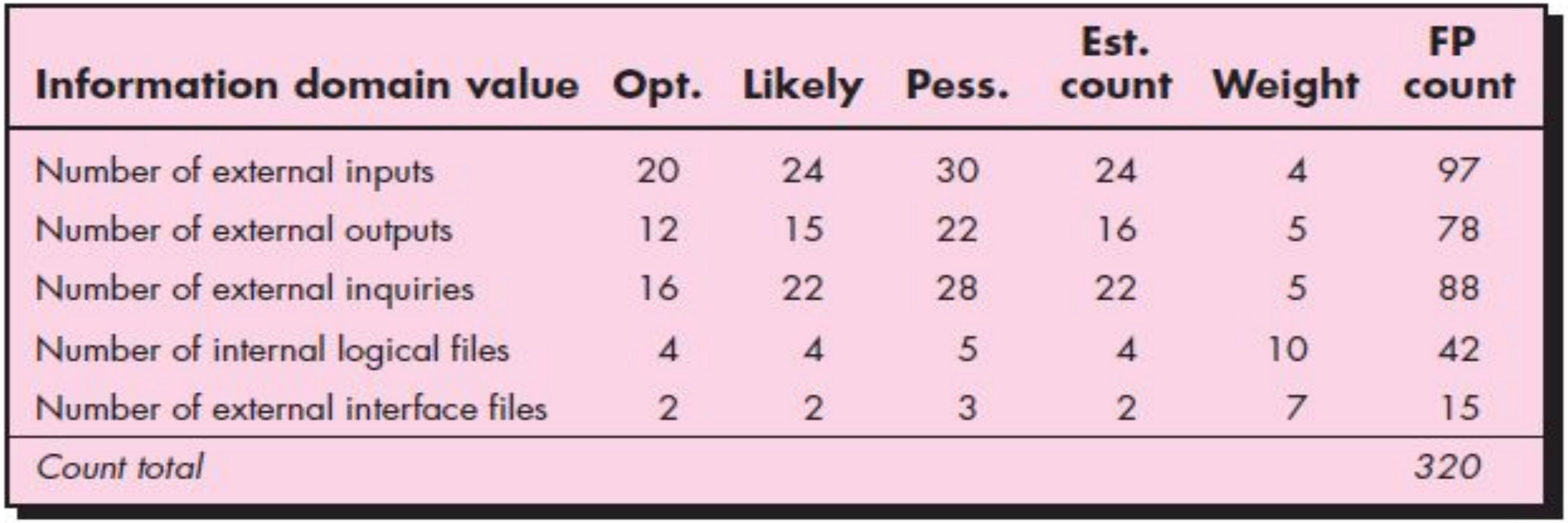
Burdened labor rate =$8000 per month,

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 (=33200X13) and**

the estimated effort is 54 (=33200/620) person-months.

Example: Functional Point(FP) Approach



Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						*320*

The estimated number of FP is derived:

$FP_{estimated}$ = count-total 3 [0.65 + 0.01 3 S (F_i)]

$FP_{estimated}$ = 375

organizational average productivity = 6.5 FP/pm.

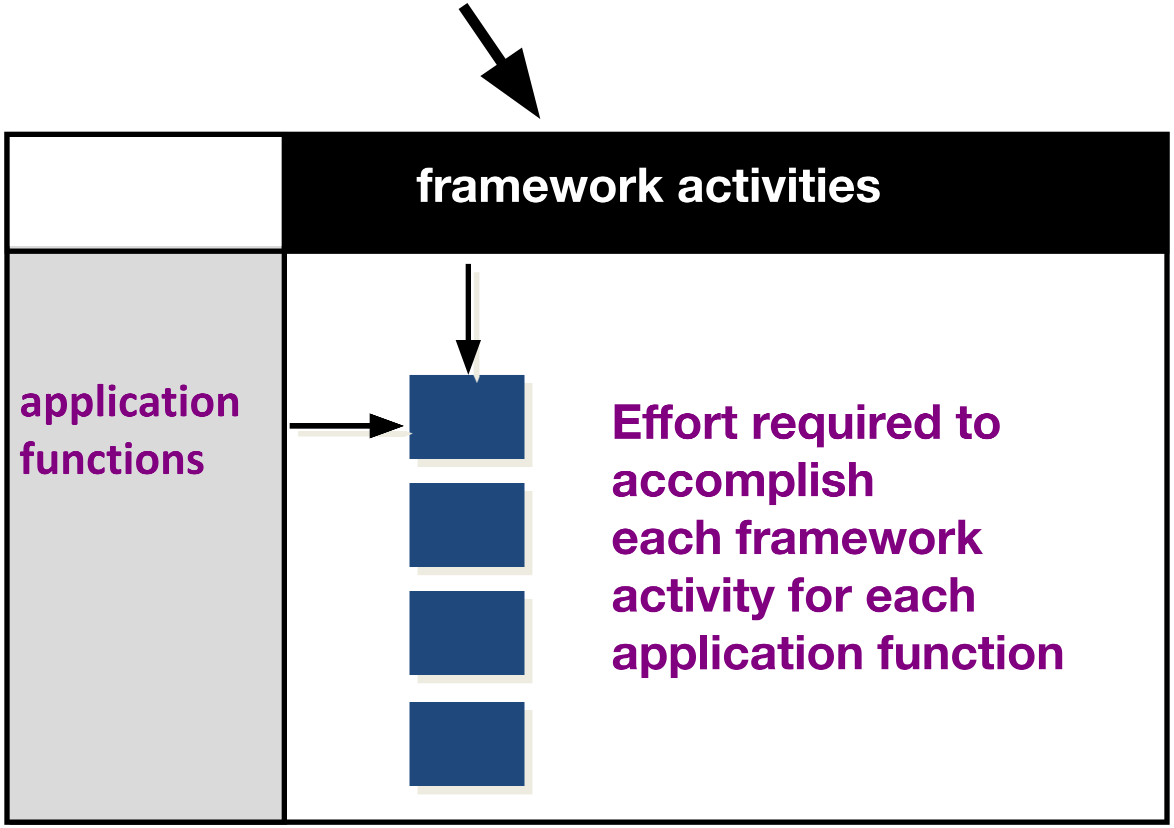
burdened labor rate = $8000 per month, approximately $1230/FP.

Based on the FP estimate and the historical productivity data,

total estimated project cost is $461,000 and estimated effort is 58 person-months.

Process-Based Estimation

Obtained from "process framework"

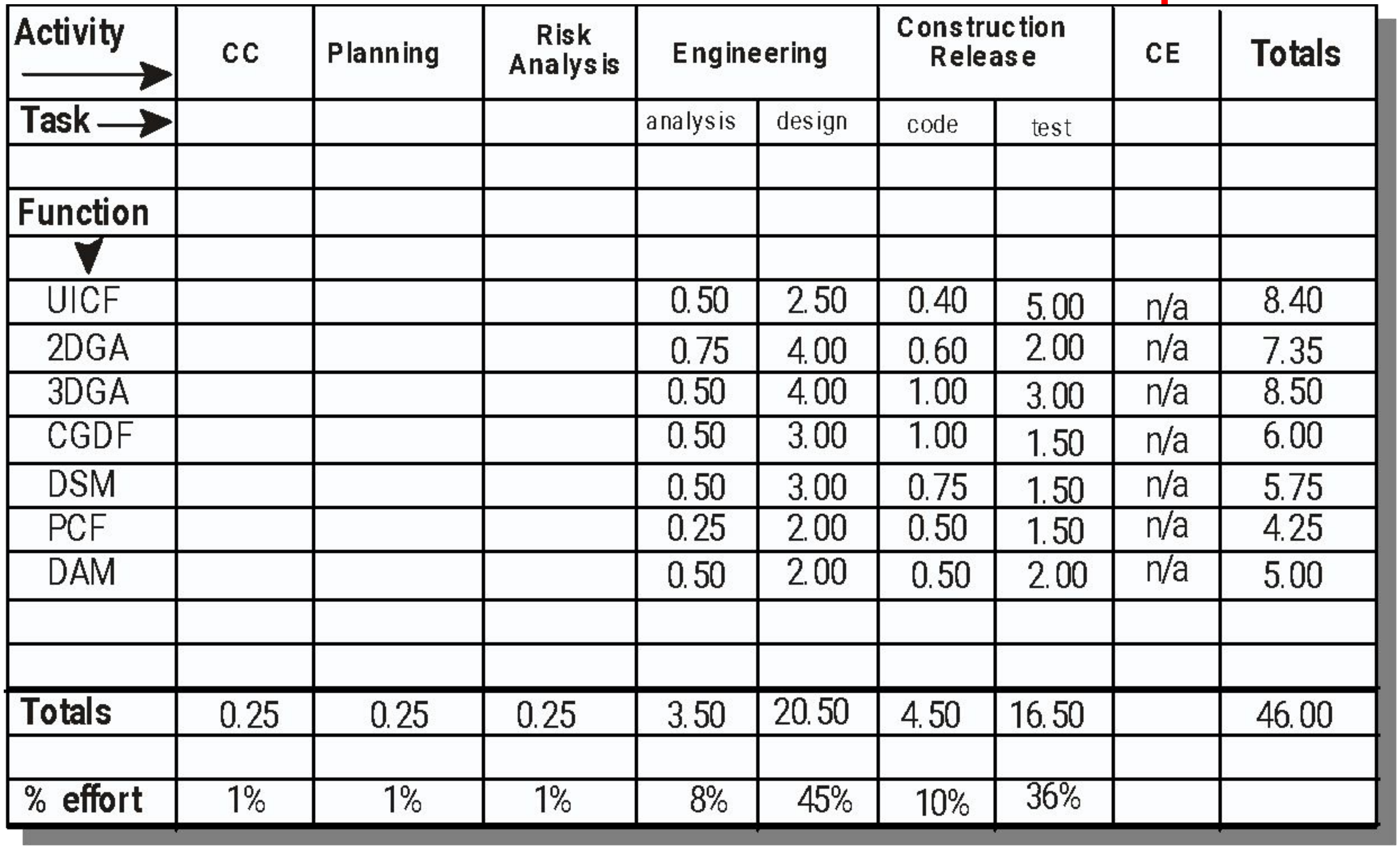


framework activities

application functions

Effort required to accomplish each framework activity for each application function

Process-Based Estimation Example



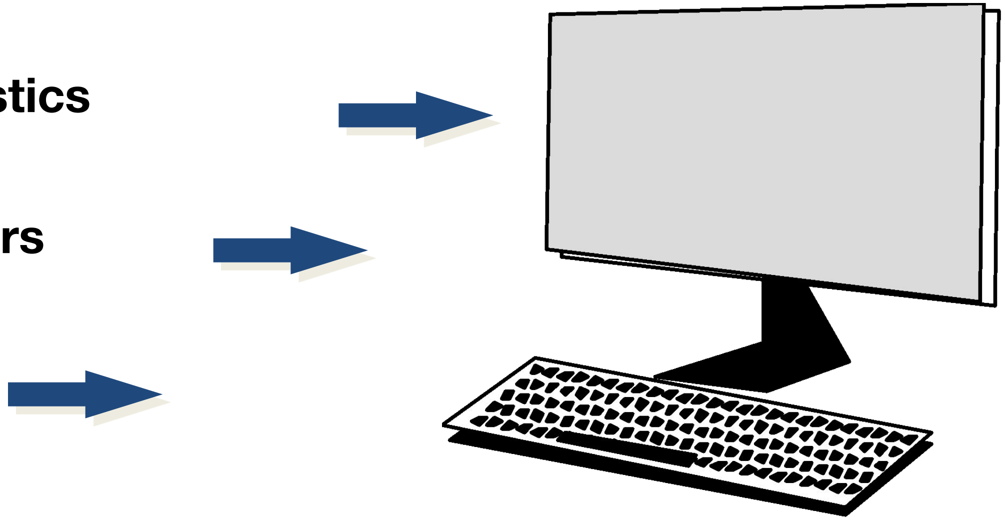
Activity →	CC	Planning	Risk Analysis	Engineering		Construction Release		CE	Totals
Task →				analysis	design	code	test		
Function									
↓									
UICF				0.50	2.50	0.40	5.00	n/a	8.40
2DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	4.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DSM				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

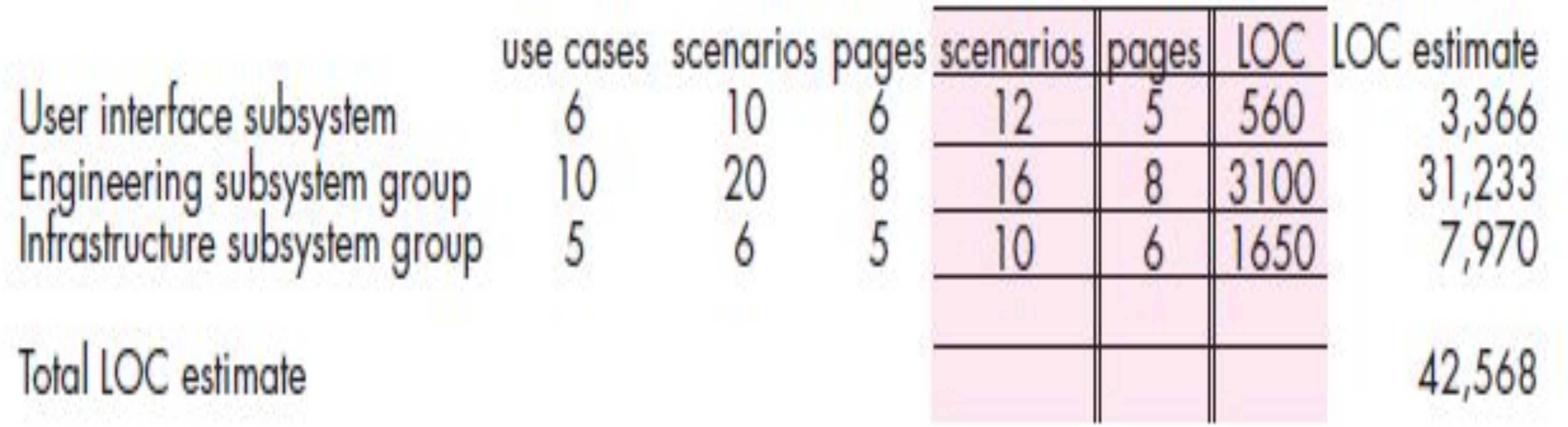


project characteristics

calibration factors

LOC/FP data

Estimation with Use-Cases



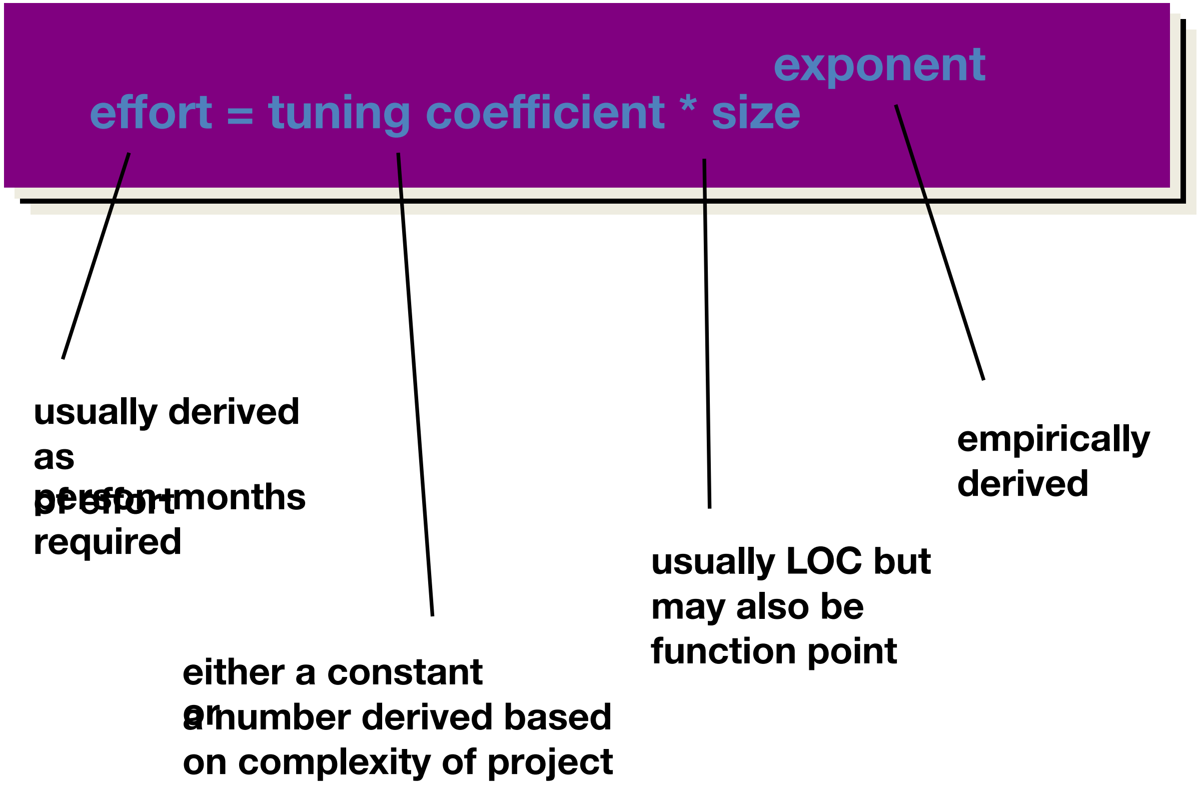
	use cases	scenarios	pages	scenarios	pages	LOC	LOC estimate
User interface subsystem	6	10	6	12	5	560	3,366
Engineering subsystem group	10	20	8	16	8	3100	31,233
Infrastructure subsystem group	5	6	5	10	6	1650	7,970
Total LOC estimate							42,568

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:



$$\text{effort} = \text{tuning coefficient} * \text{size}^{\text{exponent}}$$

usually derived as person-months of effort required

either a constant or a number derived based on complexity of project

usually LOC but may also be function point

empirically derived

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”

Project Scheduling

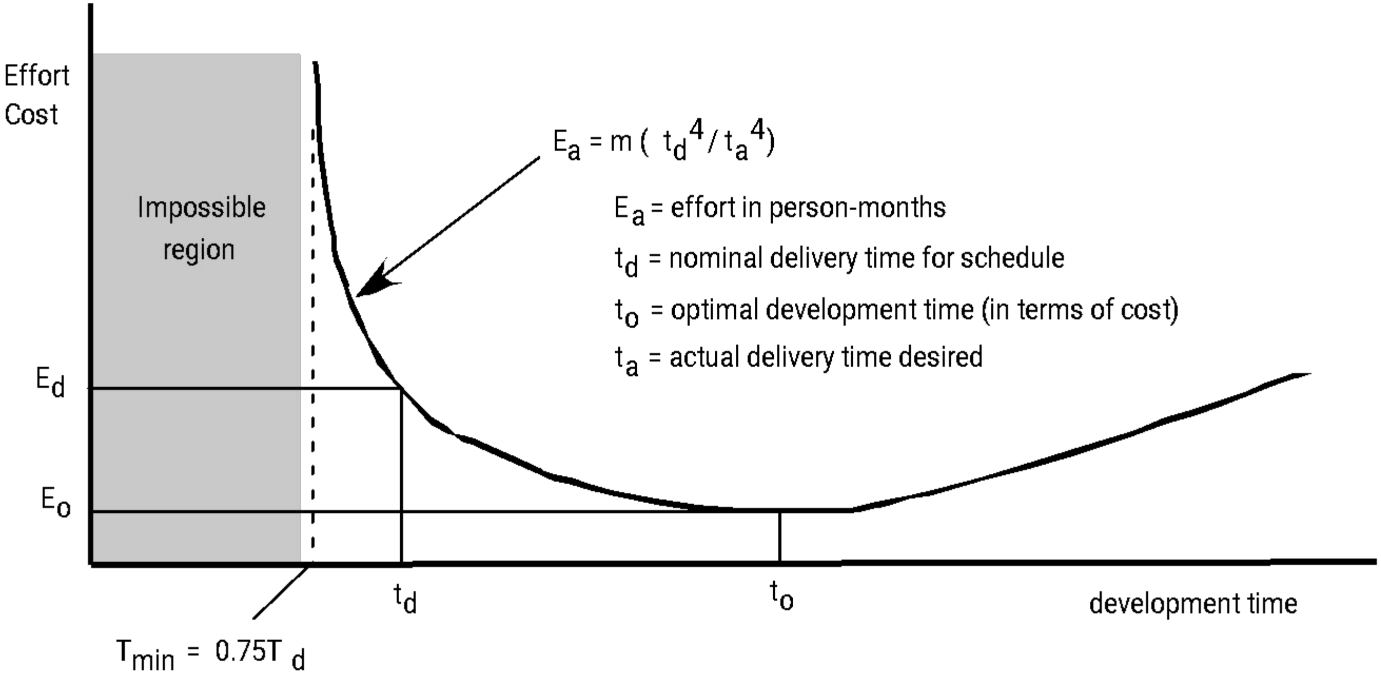
Why Are Projects Late?

- 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

Scheduling Principles

- compartmentalization—define distinct tasks
- interdependency—indicate task interrelationship
- effort validation—be sure resources are available
- defined responsibilities—people must be assigned
- defined outcomes—each task must have an output
- defined milestones—review for quality

Effort and Delivery Time



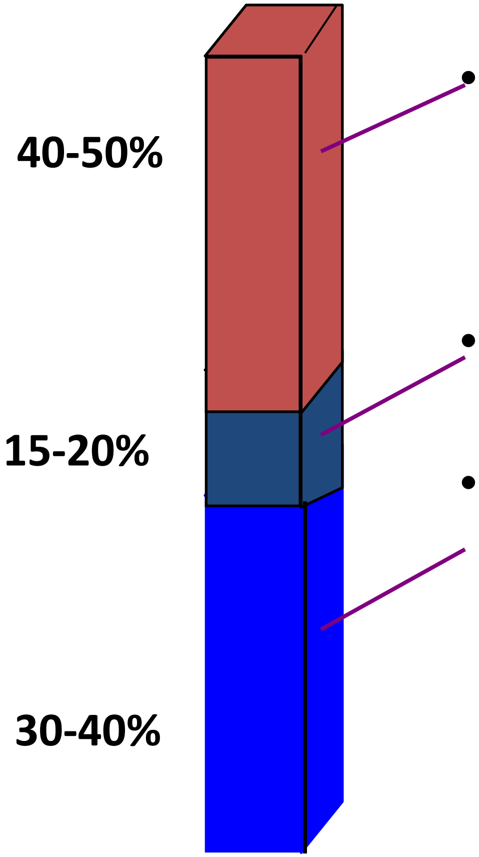
E_a = effort in person-months

t_d = nominal delivery time for schedule

t_o = optimal development time (in terms of cost)

t_a = actual delivery time desired

Effort Allocation



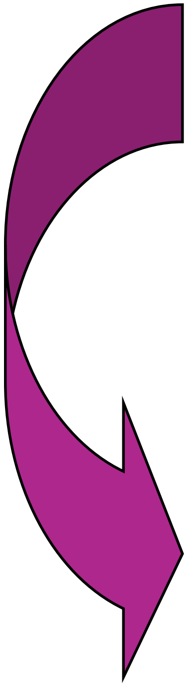
- “front end” activities
  - customer communication
  - analysis
  - design
  - review and modification
- construction activities
  - coding or code generation
- testing and installation
  - unit, integration
  - white-box, black box
  - regression

Defining Task Sets

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

Task Set Refinement

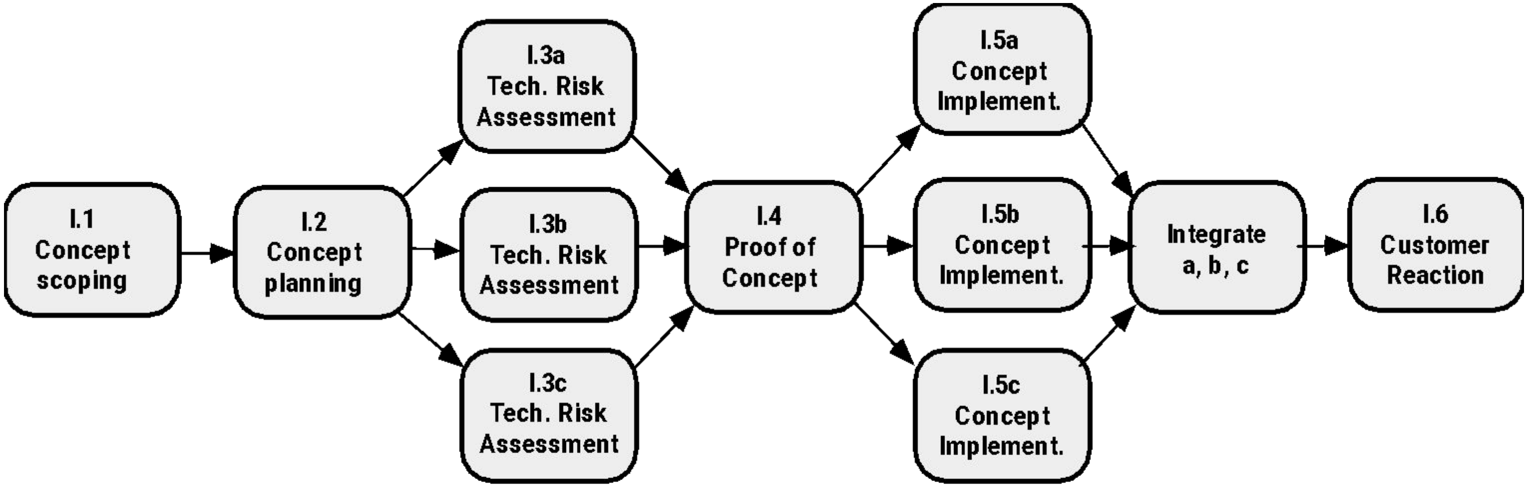
1.1 **Concept scoping** determines the overall scope of the project.



is refined to

```
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
```

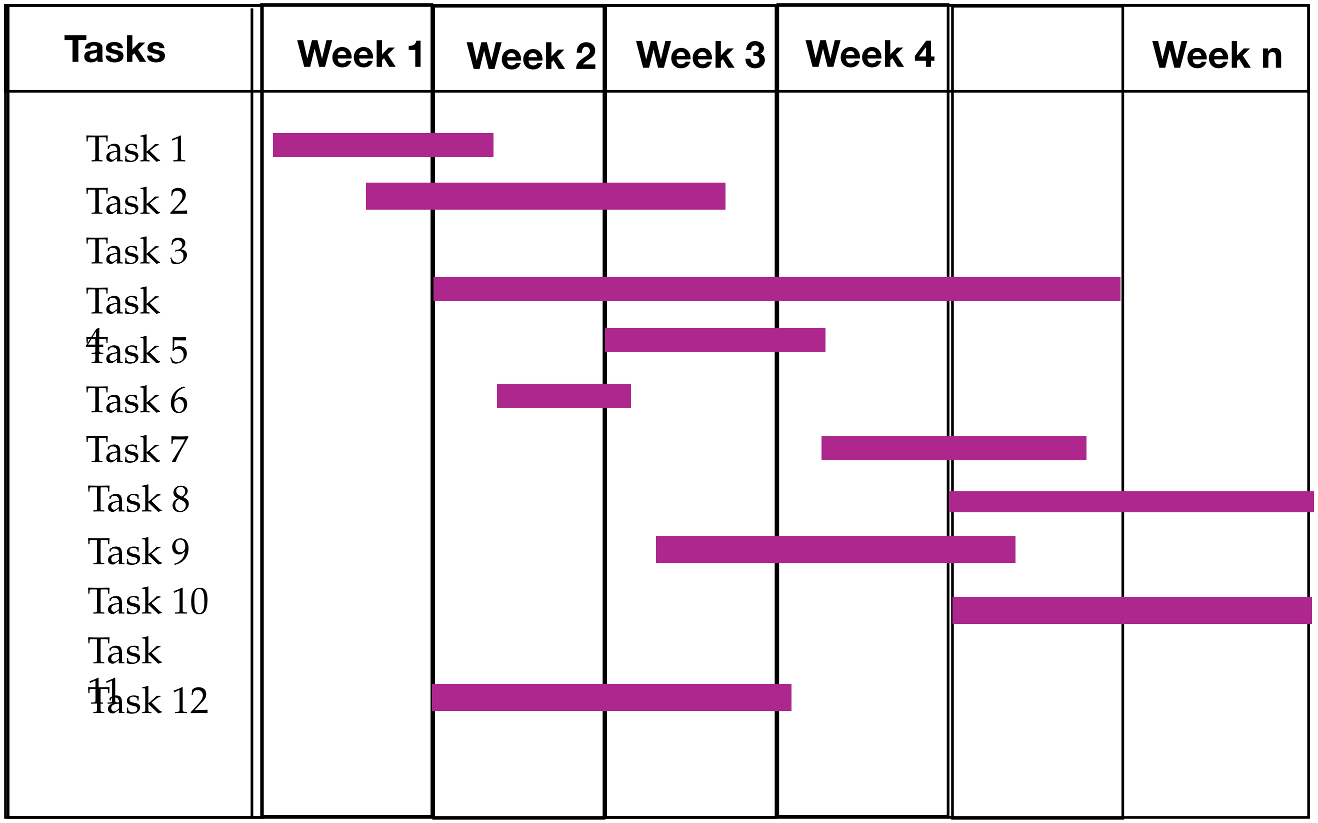
Define a Task Network



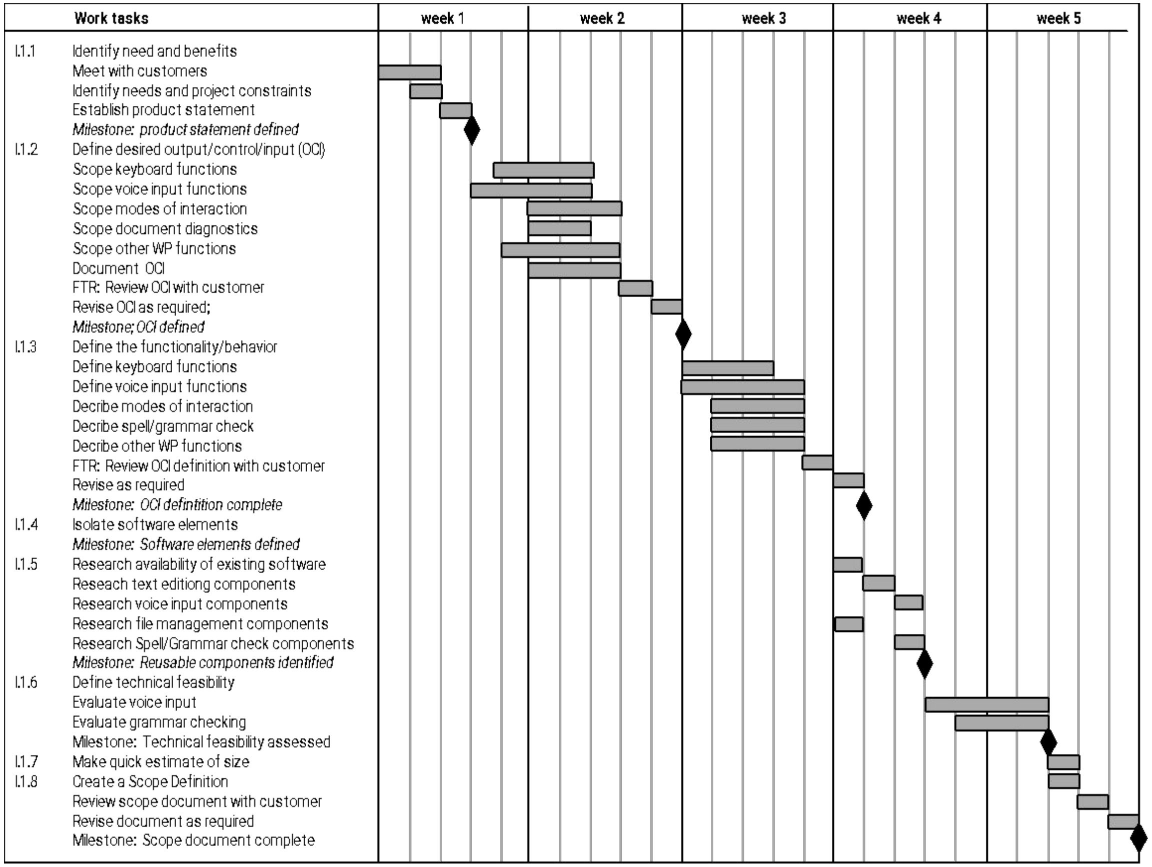
Three I.3 tasks are applied in parallel to 3 different concept functions

Three I.3 tasks are applied in parallel to 3 different concept functions

Timeline Charts



Use Automated Tools to Derive a Timeline Chart



	Work tasks
I.1.1	Identify need and benefits
	Meet with customers
	Identify needs and project constraints
	Establish product statement
	Milestone: product statement defined
I.1.2	Define desired output/control/input (OCI)
	Scope keyboard functions
	Scope voice input functions
	Scope modes of interaction
	Scope document diagnostics
	Scope other WP functions
	Document OCI
	FTR: Review OCI with customer
	Revise OCI as required;
	Milestone; OCI defined
I.1.3	Define the functionality/behavior
	Define keyboard functions
	Define voice input functions
	Decribe modes of interaction
	Decribe spell/grammar check
	Decribe other WP functions
	FTR: Review OCI definition with customer
	Revise as required
	Milestone: OCI defintition complete
I.1.4	Isolate software elements
	Milestone: Software elements defined
I.1.5	Research availability of existing software
	Reseach text editiong components
	Research voice input components
	Research file management components
	Research Spell/Grammar check components
	Milestone: Reusable components identified
I.1.6	Define technical feasibility
	Evaluate voice input
	Evaluate grammar checking
	Milestone: Technical feasibility assessed
I.1.7	Make quick estimate of size
I.1.8	Create a Scope Definition
	Review scope document with customer
	Revise document as required
	Milestone: Scope document complete

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 (Figure 27.4).
- meet informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.
- use earned value analysis (Section 27.6) to assess progress quantitatively.