COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 12

PRODUCT METRICS

M. MAHMUDUL HASAN
ASSISTANT PROFESSOR, CS, AIUB
http://www.dit.hua.gr/~m.hasan

RG Google Scholar LinkedIn

FUNCTION-BASED METRICS

- The function point metric (FP), first proposed by Albrecht, can be used effectively as a means for measuring the functionality delivered by a system (e.g. size)
- Function points are derived using an empirical relationship based on countable measures of software's information domain and assessments of software complexity

- Information domain values are defined in the following manner:
- Number of external inputs (EIs): input transactions that update internal computer files
- Number of external outputs (EOs): transactions where data is output to the user, e.g. printed reports
- Number of internal logical files (ILFs): group of data that is usually accessed together, e.g. purchase order file
- Number of external interface files (EIFs): file sharing among different applications to achieve a common goal
- Number of external inquiries (EQs): transactions that provide information but do not update internal file

FUNCTION POINTS METRICS

Information Domain Value (FPunadjusted)	**Count**		**Simple**	**Average**	**Complex**		
Number of external inputs (EIs)		X	3	4	6	=	
Number of external outputs (EOs)		X	4	5	7	=	
Number of external inquiries (EQs)		X	3	4	6	=	
Number of internal logical files (ILFs)		X	7	10	15	=	
Number of external interface files (EIFs)		X	5	7	10	=	
					Count Total		

METRICS FOR OO DESIGN

Whitmire describes nine distinct and measurable characteristics of an OO design:

- **Size:** size is defined in terms of volume, length, and functionality
- **Complexity:** how classes of an OO design are interrelated to one another
- **Coupling:** the physical connections between elements of the OO design
- **Cohesion :** the degree to which all operations working together to achieve a single, well-defined purpose

METRICS FOR OO DESIGN

- **Sufficiency:** the degree to which an abstraction possesses the features required of it, or the degree to which a design component possesses features in its abstraction, from the point of view of the current application. (e.g. deals with interface and hide internals to the users)
- **Completeness:** an indirect implication about the degree to which the abstraction or design component can be reused
- **Primitiveness:** applied to both operations and classes, the degree to which an operation is atomic
- **Similarity:** the degree to which two or more classes are similar in terms of their structure, function, behavior, or purpose
- **Volatility:** measures the likelihood that a change will occur

CLASS ORIENTED METRICS

Proposed by Chidamber and Kemerer:

- Weighted methods per class - number of functions in class (WMC)
- Depth of the inheritance tree (DIT)
- Number of children (NOC)
- Coupling between object classes (CBC)
- Lack of cohesion in methods (LCOM)

CLASS ORIENTED METRICS

Proposed by Lorenz and Kidd:

- Class size
- Number of operations overridden by a subclass
- Number of operations added by a subclass

OPERATION-ORIENTED METRICS

Proposed by Lorenz and Kidd:

- Average operation size
- Operation complexity
- Average number of parameters per operation

CODE METRICS

- Halstead's Software Science: a comprehensive collection of metrics all predicated on the number (count and occurrence) of operators and operands within a component or program

METRICS FOR TESTING

- Testing effort can also be estimated using metrics
- Binder suggests a broad array of design metrics that have a direct influence on the "testability" of an OO system.
  - Lack of cohesion in methods (LCOM).
  - Percent public and protected (PAP).
  - Public access to data members (PAD).
  - Number of root classes (NOR).
  - Number of children (NOC) and depth of the inheritance tree (DIT).

MAINTENANCE METRICS

- IEEE Std. 982.1-1988 suggests a **software maturity index (SMI)** that provides an indication of the stability of a software product (based on changes that occur for each release of the product).
  The following information is determined:
  - M_T = the number of modules in the current release
  - F_c = the number of modules in the current release that have been changed
  - F_a = the number of modules in the current release that have been added
  - F_d = the number of modules from the preceding release that were deleted in the current release
- The software maturity index is computed in the following manner:

  $$\textbf{SMI} = [MT - (F_a + F_c + F_d)]/MT$$
- As SMI approaches 1.0, the product begins to stabilize.

COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 13

SOFTWARE CONFIGURATION MANAGEMENT

M. MAHMUDUL HASAN
ASSISTANT PROFESSOR, CS, AIUB
http://www.dit.hua.gr/~m.hasan

RG Google Scholar LinkedIn

BASELINE

- A *baseline* is a software configuration management concept that helps you to control change without seriously impeding justifiable change.
- The IEEE (IEEE Std. No. 610.12-1990) defines a baseline as:

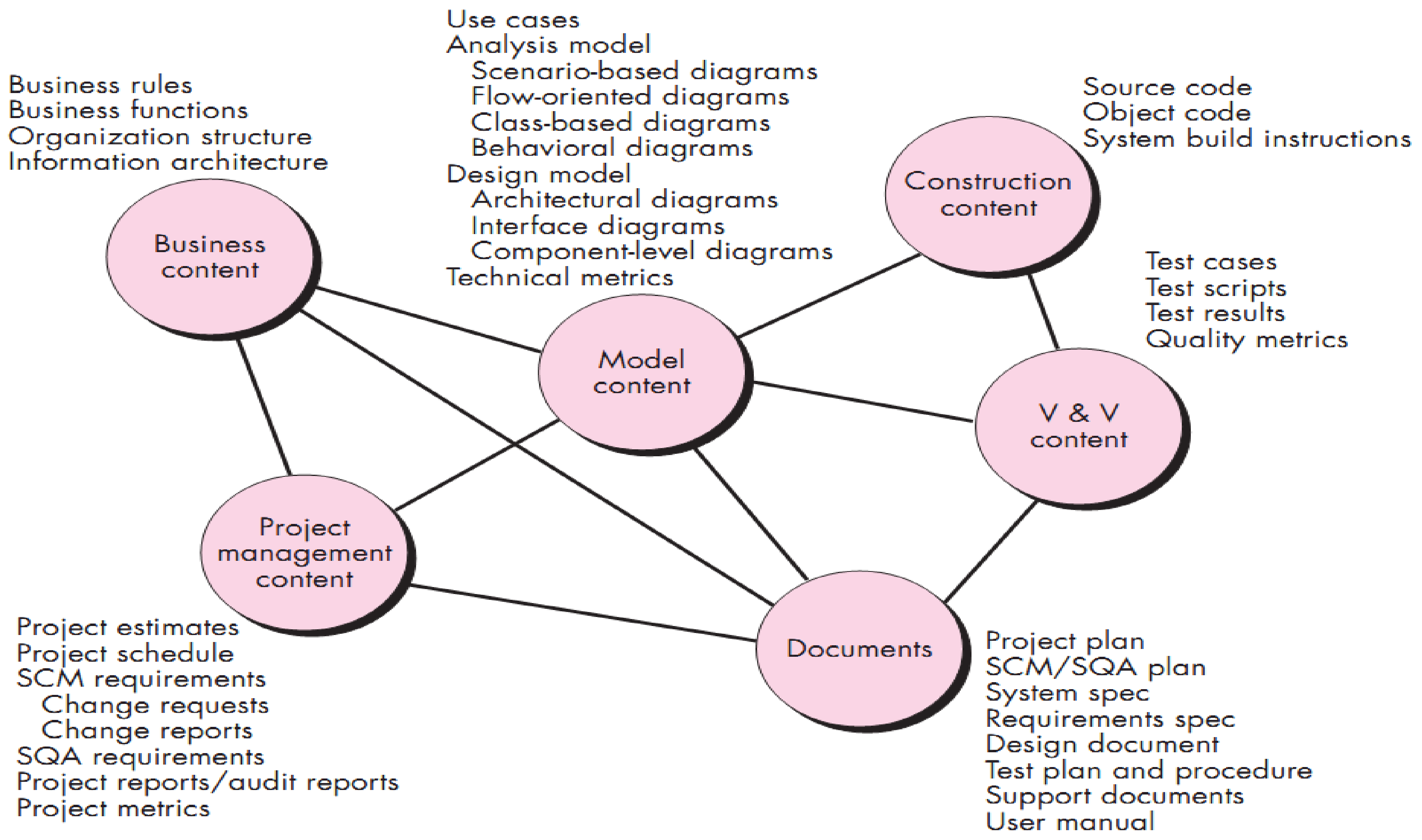
  A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

- For example, the elements of a design model have been documented and reviewed. Errors are found and corrected. Once all parts of the model have been reviewed, corrected, and then approved, the design model becomes a baseline.

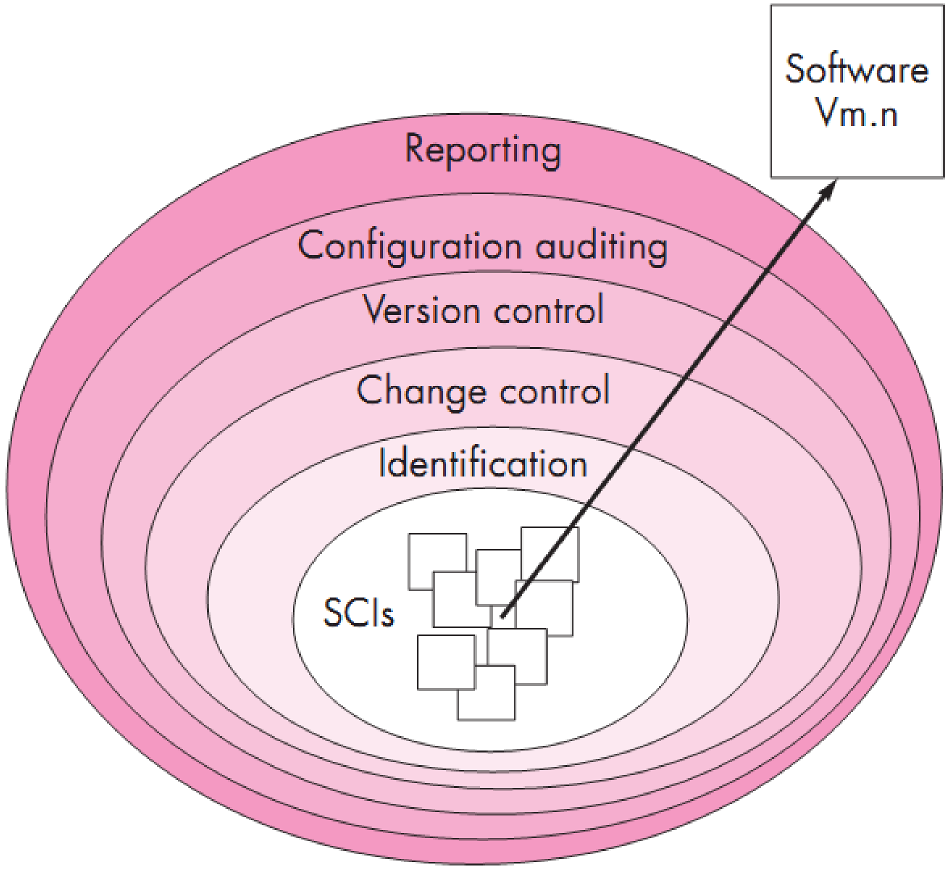
SCM REPOSITORY

- The items that comprise all information produced as part of the software development process are collectively called a *software configuration*.
- A database that acts as the center for both accumulation and storage of software engineering information
- The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner

CONTENT OF SCM REPOSITORY



SCM LAYERS



- S/w configuration items (SCIs) flow outward through these layers throughout their lifetime
- When a new SCI is created, it must be identified, however, if no changes are requested for the SCI, the changer control layer does not apply.
- Each SCI created is assigned to a specific version of s/w
- A version control captures all changes to all files in the configuration along with the reason for changes and details of who made the changes and when
- The record of all these SCI's (i.e. name, creation date, version, etc.) is maintained for configuration auditing purpose.

CHANGE CONTROL PROCESS

1. Need for change is recognized
2. Change request from user
3. Developers Evaluates
4. Change report is generated
5. Change control authority decides
6. If (Request is queued for action)….else (change request is denied)→ user is informed
7. Assign individuals to configuration objects
8. “Check out” configuration objects (items)
9. Make the change (in design)

CHANGE CONTROL PROCESS

12. Establish a baseline for testing
13. Perform quality assurance and testing activities (regression testing)
14. Promote changes for inclusion in next release (revision)
15. Rebuild appropriate version of software
16. Review the change to all configuration
17. Include changes in new version
18. Distribute the new version

COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 14

SOFTWARE PROJECT ESTIMATION

M. MAHMUDUL HASAN
ASSISTANT PROFESSOR, CS, AIUB
http://www.dit.hua.gr/~m.hasan

RG Google Scholar LinkedIn

SOFTWARE PROJECT PLANNING

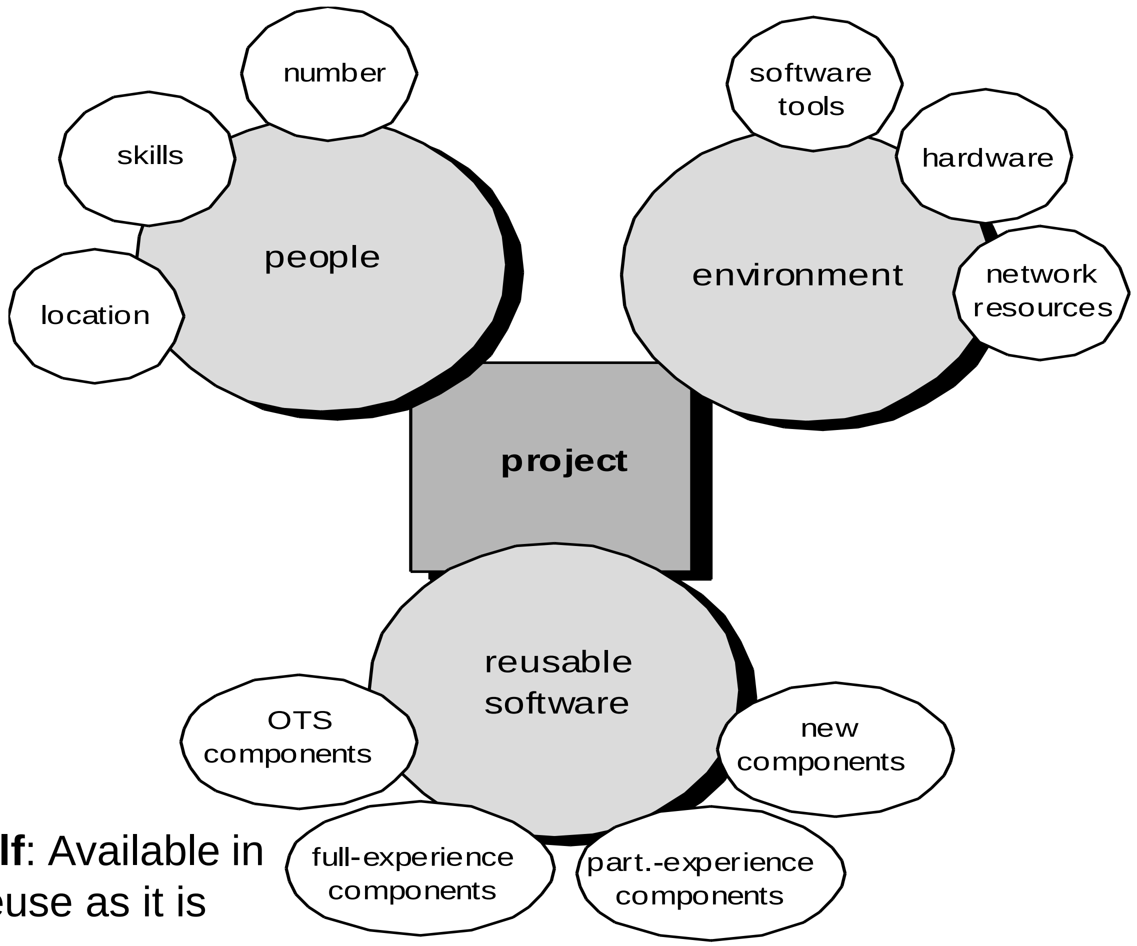
- The overall goal of project planning is to establish a pragmatic strategy for controlling, tracking, and monitoring a complex technical project.

WHY?

- So the end result gets done on time, on budget, and with quality

PROJECT PLANNING TASK SET-I

- 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

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”

COCOMO (CONSTRUCTIVE COST MODEL)

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

- Effort = PM = $\text{Coefficient}_{\text{<Effort Factor>}}$*(SLOC/1000)^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

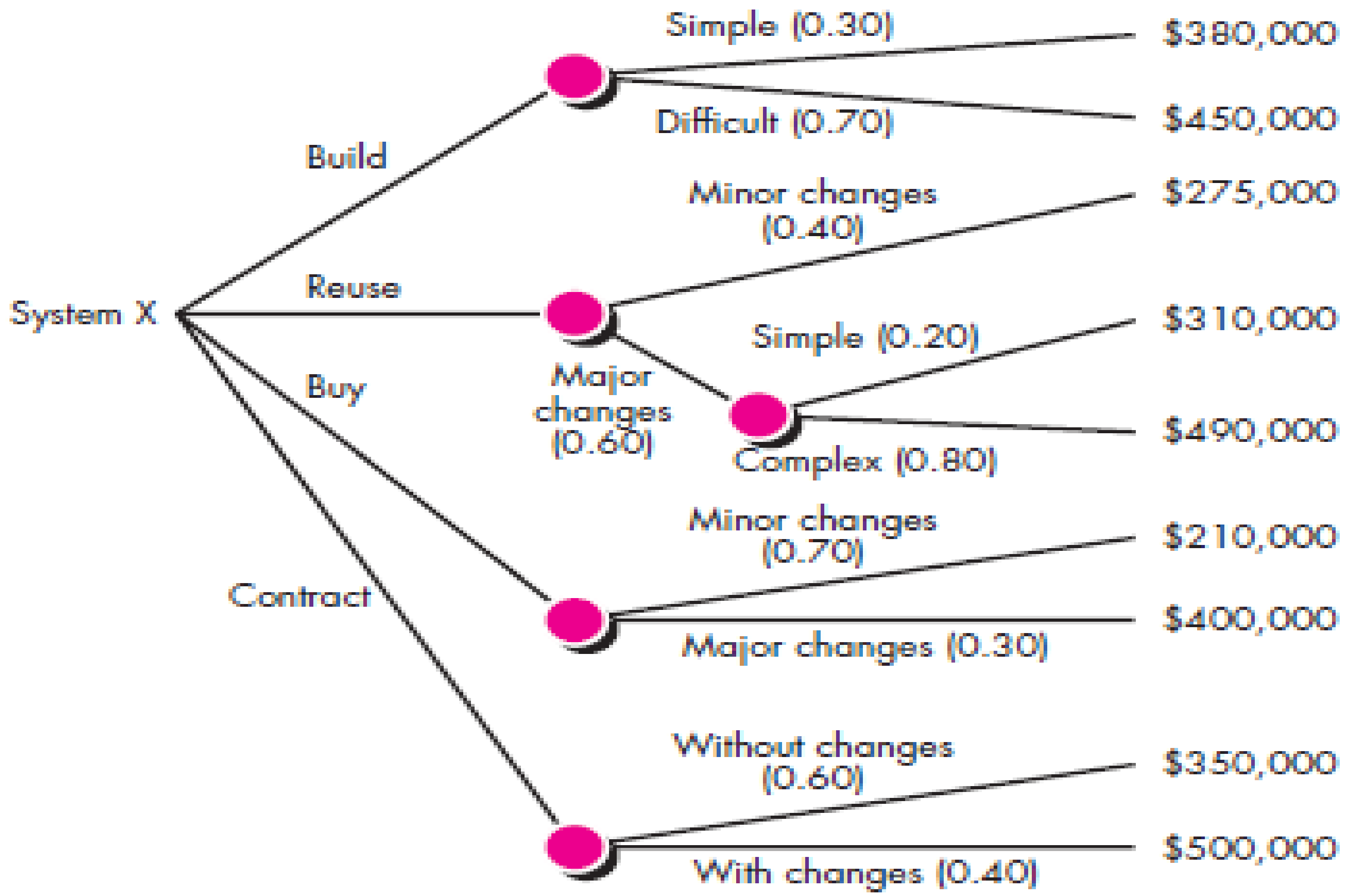
COCOMO (CONSTRUCTIVE COST MODEL)

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

expected cost = (path probability)$_i$ x (estimated path cost)$_i$

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

similarly,

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

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

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

COURSE NAME

SOFTWARE ENGINEERING

CSC 3114

(UNDERGRADUATE)

CHAPTER 15

PROJECT SCHEDULING

M. MAHMUDUL HASAN
ASSISTANT PROFESSOR, CS, AIUB
http://www.dit.hua.gr/~m.hasan

RG Google Scholar LinkedIn

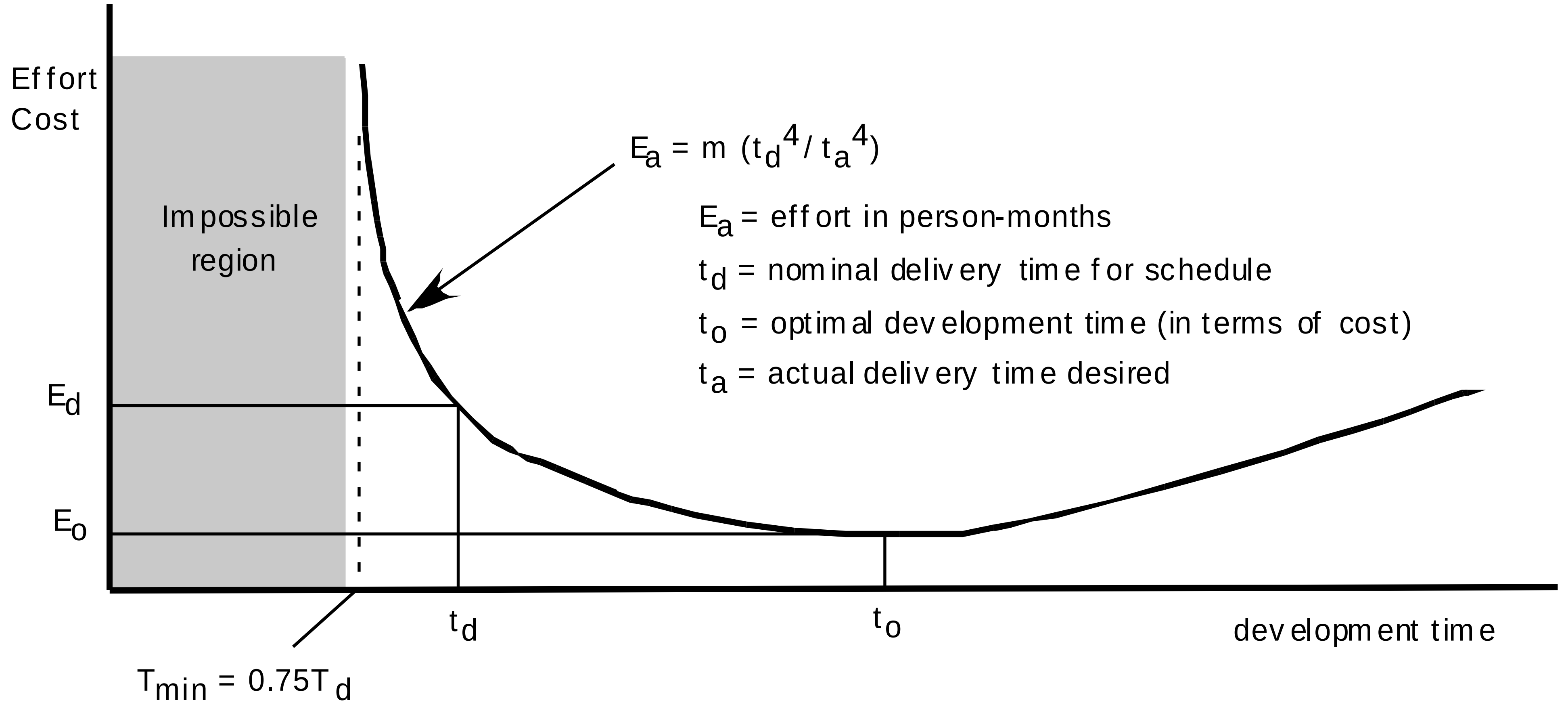
WHY ARE PROJECTS LATE?

- 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



$E_a = m\ (t_d^4 / t_a^4)$

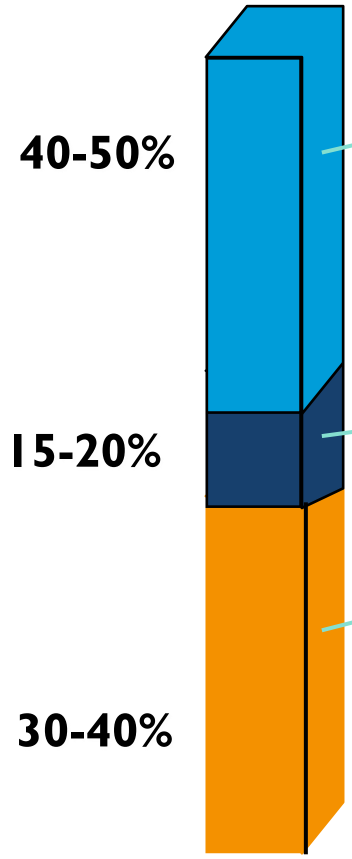
E_a = effort in person-months

t_d = nominal delivery time for schedule

t_o = optimal development time (in terms of cost)

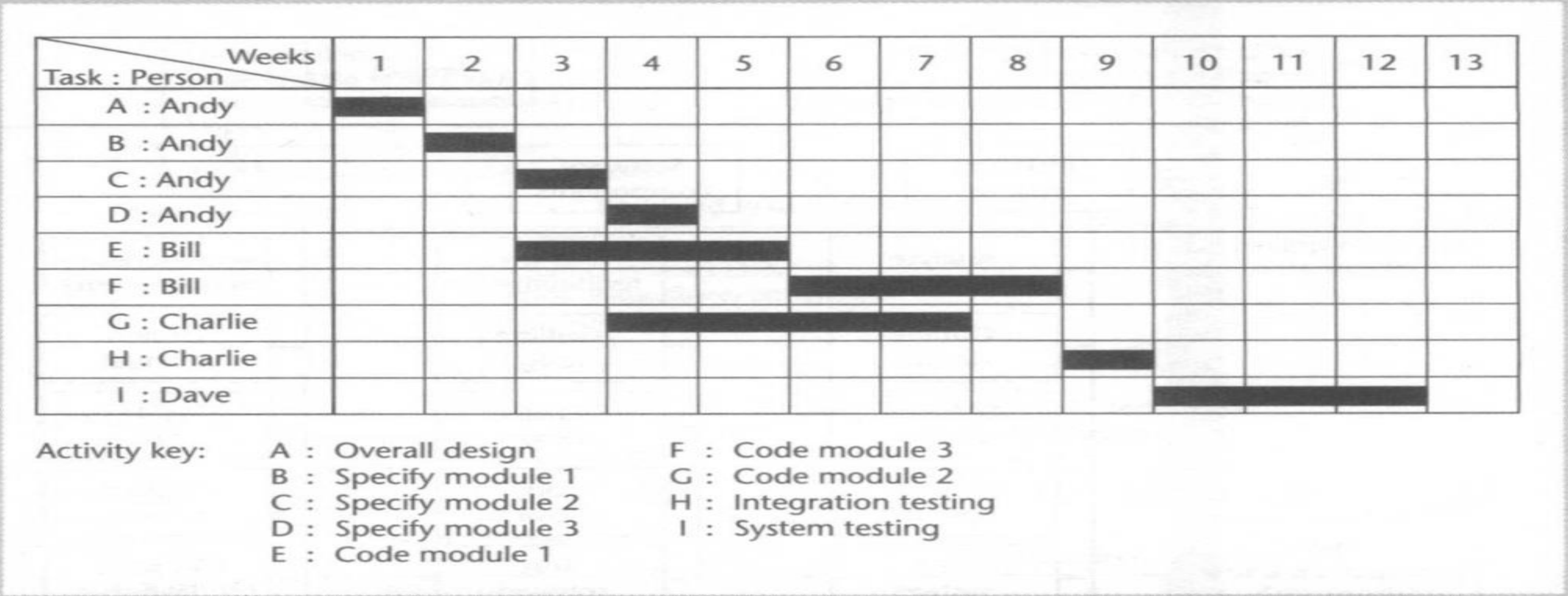
t_a = actual delivery time desired

EFFORT ALLOCATION (40-20-40 RULE)



- "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

TIMELINE CHARTS

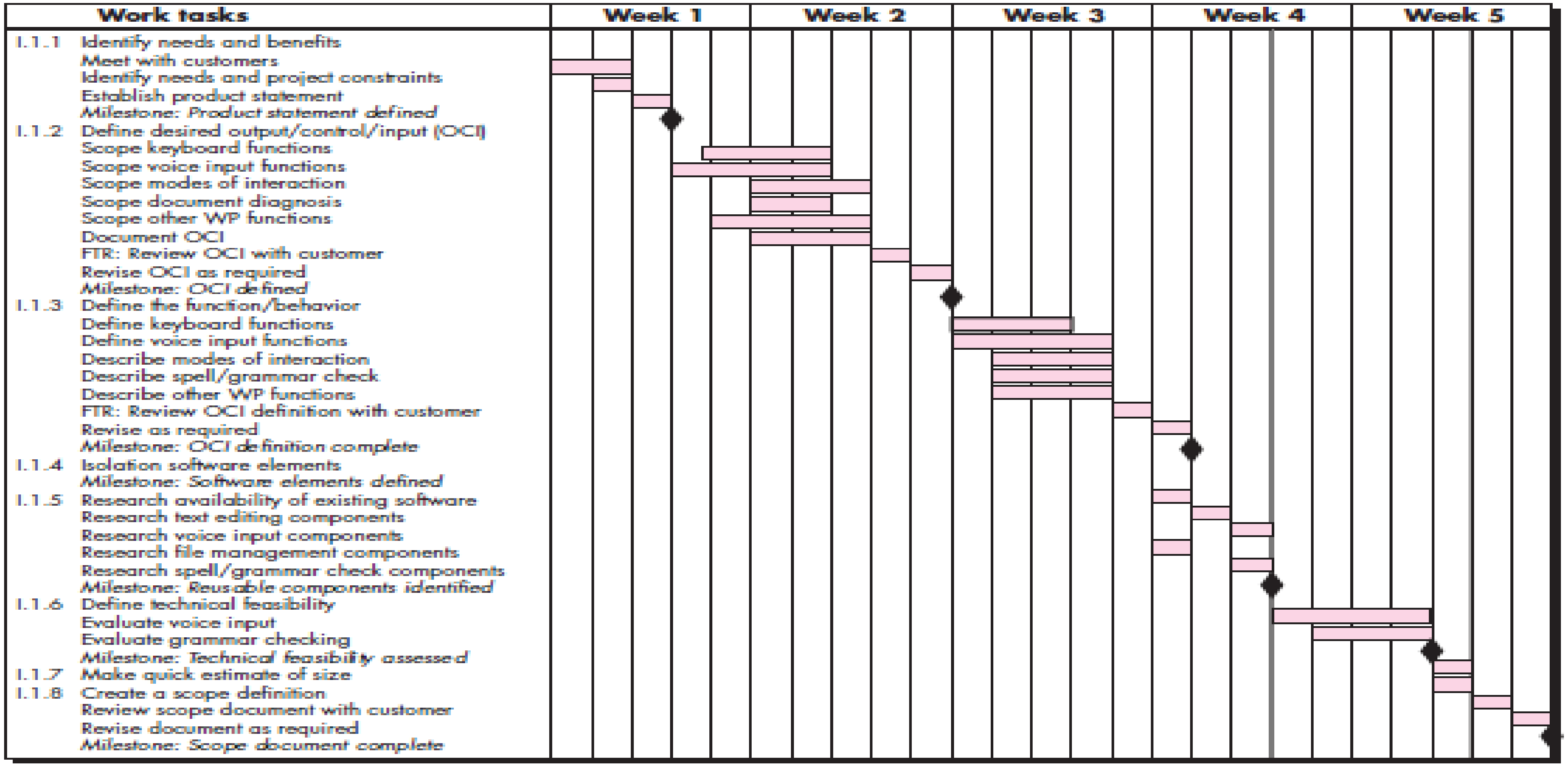


Task : Person \ Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13
A : Andy	■												
B : Andy		■											
C : Andy			■										
D : Andy				■									
E : Bill			■	■	■								
F : Bill						■	■	■					
G : Charlie				■	■	■	■						
H : Charlie									■				
I : Dave										■	■	■	

Activity key:		
	A : Overall design	F : Code module 3
	B : Specify module 1	G : Code module 2
	C : Specify module 2	H : Integration testing
	D : Specify module 3	I : System testing
	E : Code module 1	

Figure 6.6 A project plan as a bar chart

TIMELINE CHARTS



Work tasks	
I.1.1	Identify needs 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 diagnosis
	Scope other WP functions
	Document OCI
	FTR: Review OCI with customer
	Revise OCI as required
	Milestone: OCI defined
I.1.3	Define the function/behavior
	Define keyboard functions
	Define voice input functions
	Describe modes of interaction
	Describe spell/grammar check
	Describe other WP functions
	FTR: Review OCI definition with customer
	Revise as required
	Milestone: OCI definition complete
I.1.4	Isolation software elements
	Milestone: Software elements defined
I.1.5	Research availability of existing software
	Research text editing 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.
- Use **earned value analysis** to assess progress quantitatively.

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."

COMPUTING EARNED VALUE

- 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

- *Budgeted cost of work performed* (BCWP)
- 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."
- Given values for BCWS, BAC, and BCWP, important progress indicators can be computed:
- Schedule performance index, SPI = BCWP/BCWS
- SPI tells you how efficiently you are actually progressing compared to the planned progress.
- SPI is an indication of the efficiency with which the project is utilizing scheduled resources.
- Schedule variance, SV = BCWP – BCWS

COMPUTING EARNED VALUE

- 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

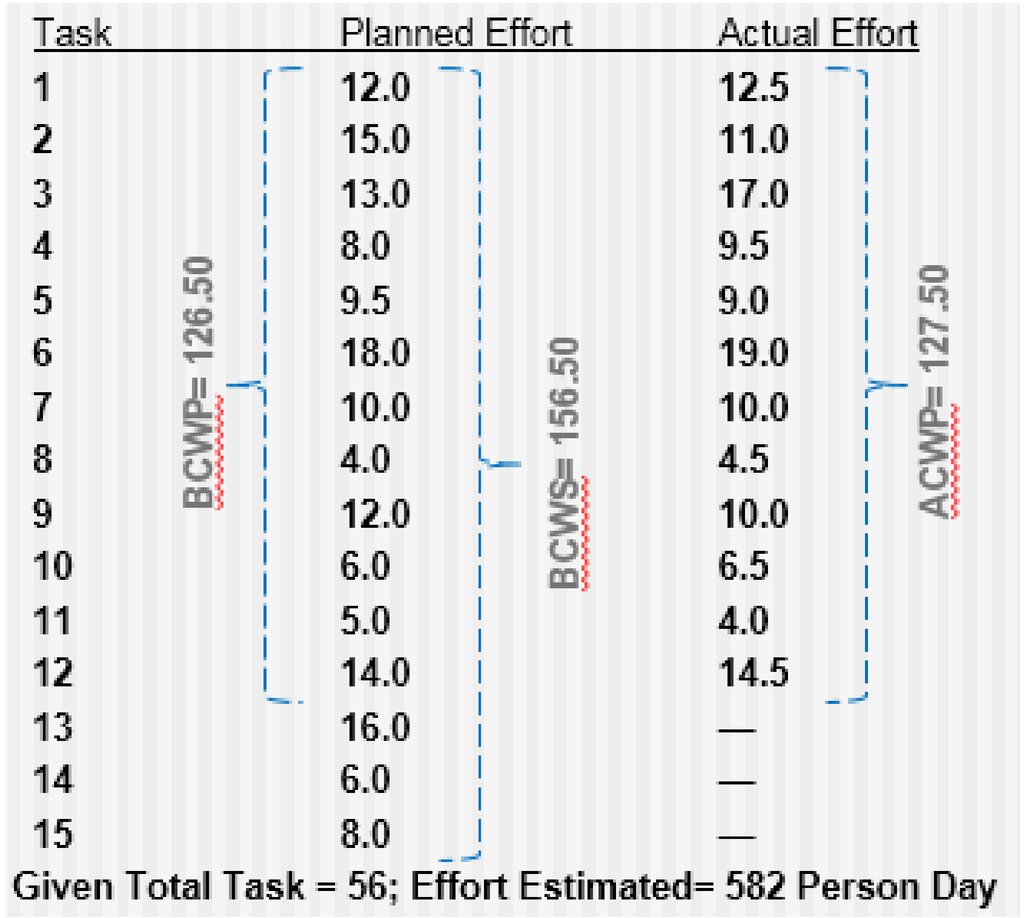
  The Cost Performance Index helps you analyze the efficiency of the cost utilized by the project.

- Cost variance, CV = BCWP – ACWP

EVA EXERCISE

- Assume you are a software project manager and you've been asked to compute earned value statistics for a small software project.
- The project has 56 planned work tasks that are estimated to require 582 person-days to complete.
- At the time that you've been asked to do the earned value analysis, 12 tasks have been completed.
- However the project schedule indicates that 15 tasks should have been completed.
- The following scheduling data (in person-days) are avalable:

EVA EXERCISE


BCWP= 126.50
BCWS= 156.50
ACWP= 127.50

Task	Planned Effort	Actual Effort
1	12.0	12.5
2	15.0	11.0
3	13.0	17.0
4	8.0	9.5
5	9.5	9.0
6	18.0	19.0
7	10.0	10.0
8	4.0	4.5
9	12.0	10.0
10	6.0	6.5
11	5.0	4.0
12	14.0	14.5
13	16.0	—
14	6.0	—
15	8.0	—

Given Total Task = 56; Effort Estimated= 582 Person Day

- BAC = 582.00

- SPI = BCWP/ BCWS = 126.5/ 156.5 = 0.808307
- SV = BCWP - BCWS = 126.5 - 156.5 = -30 person-day

- CPI = BCWP/ ACWP = 0.99
- CV = BCWP – ACWP = -1 person-day

- % schedule for completion = BCWS/ BAC = 156.5/ 582.00
  = 26.89%

  [% of work scheduled to be done at this time]

- % complete = BCWP/ BAC = 126.5/ 582.00 = 21.74%
  [% of work completed at this time]