Chapter 27

- **Project Scheduling**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 7/e
by Roger S. Pressman

Slides copyright © 1996, 2001, 2005, 2009 by Roger S. Pressman

For non-profit educational use only

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

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

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

2

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
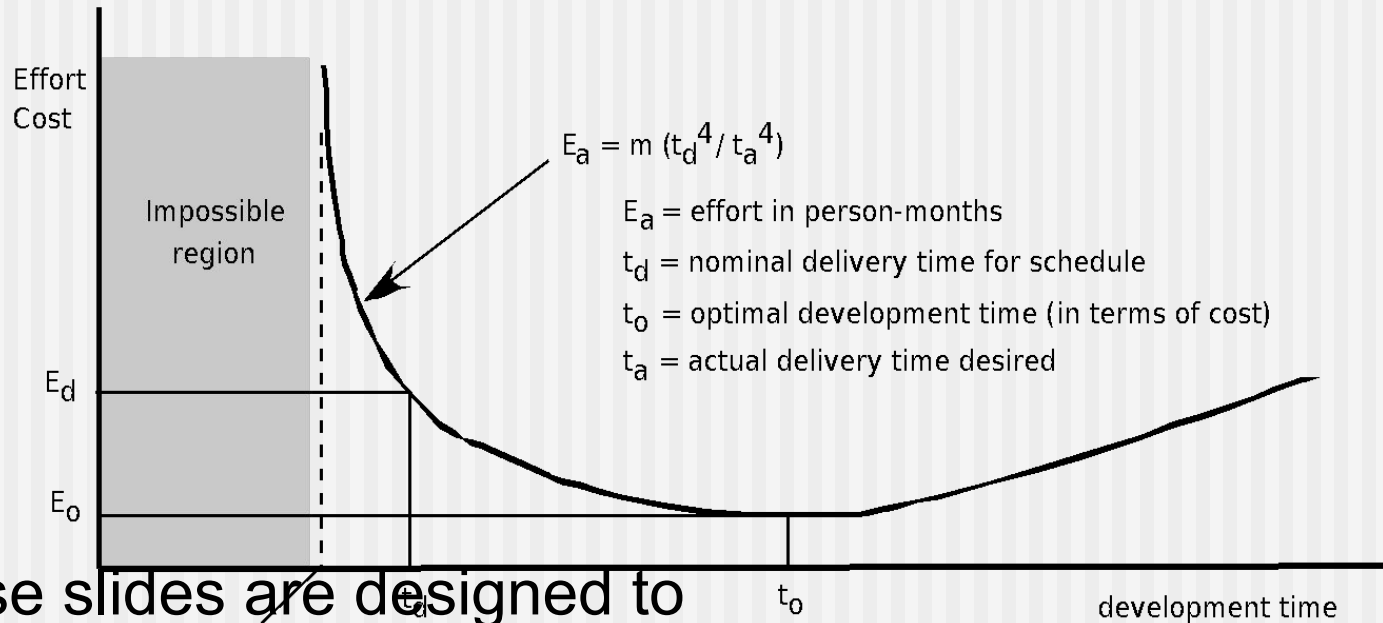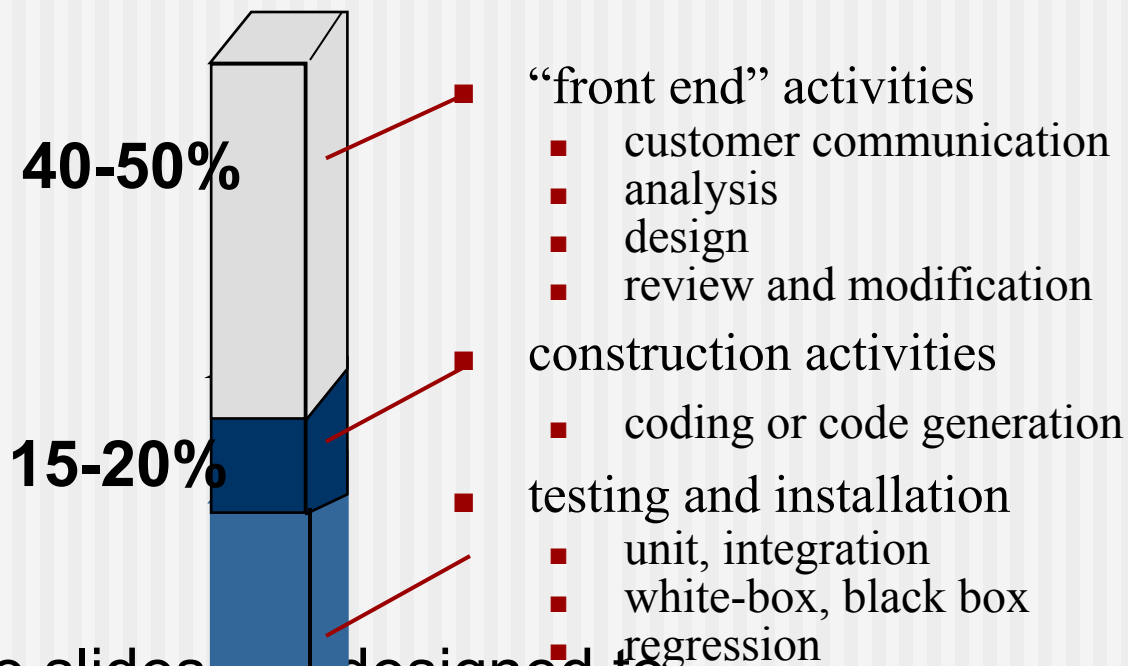- defined milestones—review for quality

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

3

Effort and Delivery Time

Effort
Cost

Impossible
region

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

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

E_d

E_o

t_d

t_o

development time

$T_{min} = 0.75T_d$

4

Effort Allocation

40-50%

15-20%

30-40%

"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.

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;
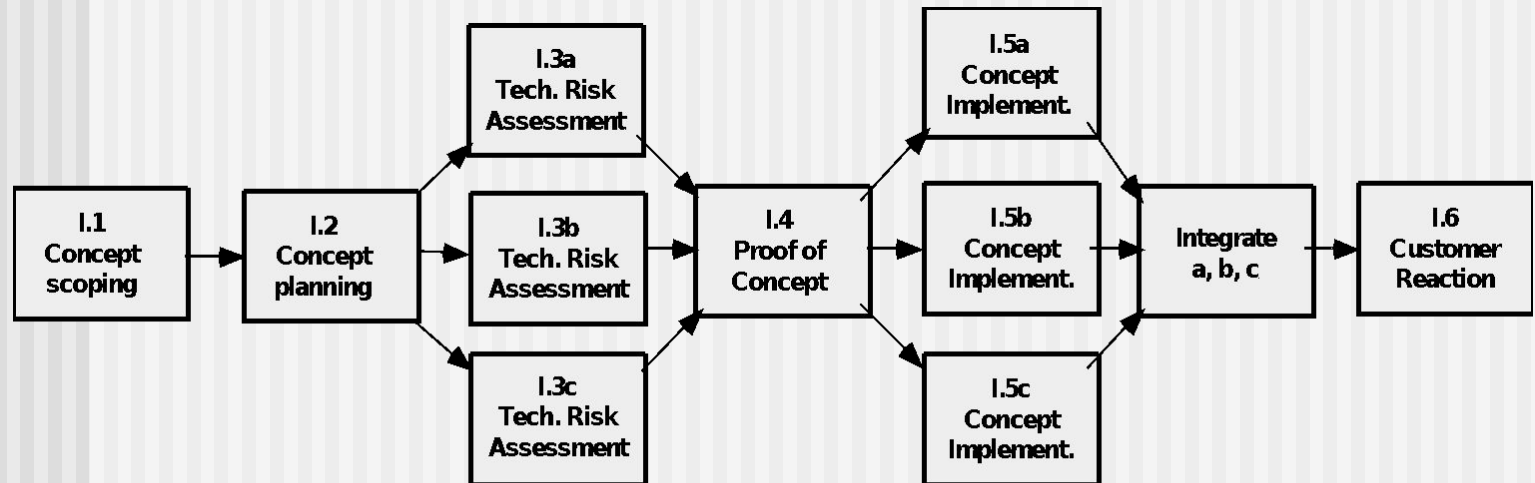
1.1.6   Define technical feasibility;

1.1.7   Make quick estimate of size;

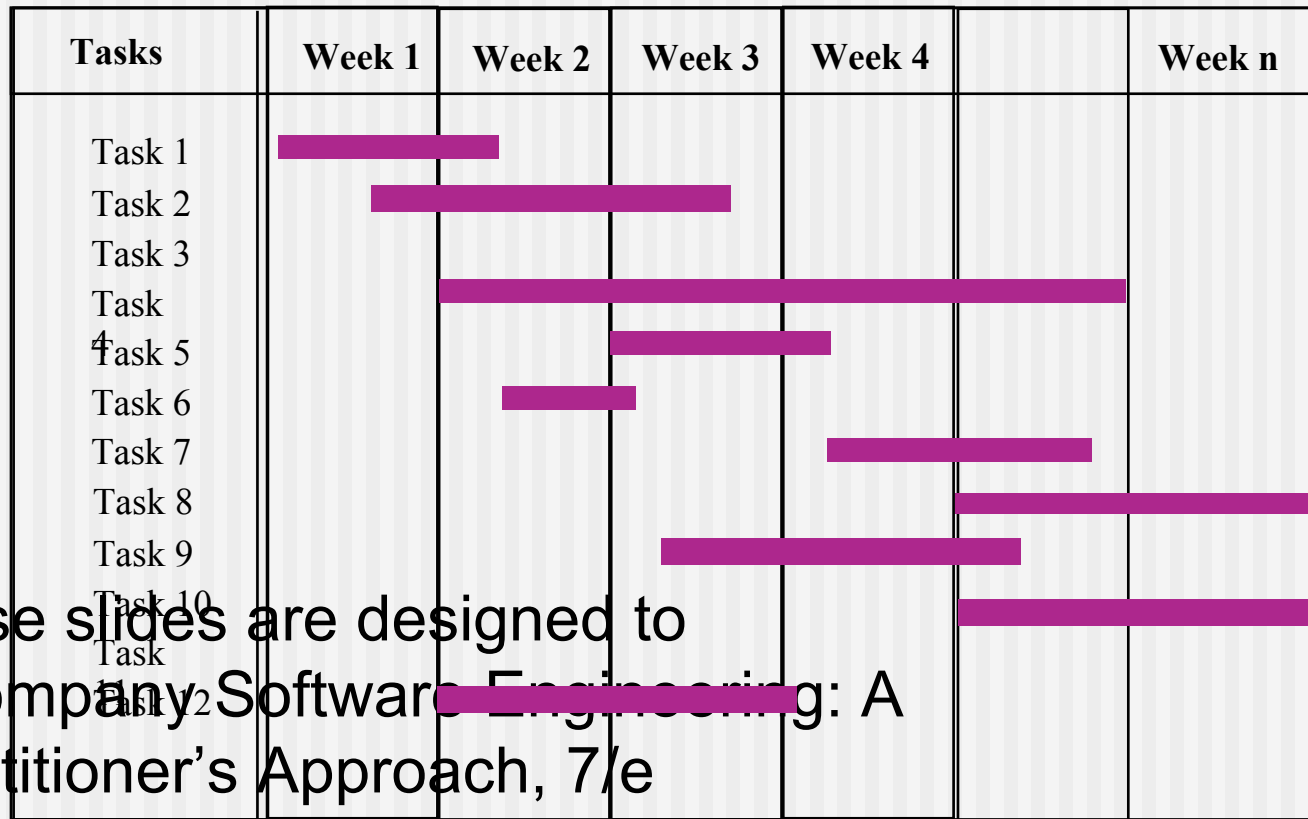1.1.8   Create a Scope Definition;

endTask definition: Task 1.1

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

is reducom

7

Define a Task Network

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

8

Timeline Charts

Tasks	Week 1	Week 2	Week 3	Week 4		Week n
Task 1						
Task 2						
Task 3						
Task 4						
Task 5						
Task 6						
Task 7						
Task 8						
Task 9						
Task 10						
Task 11						
Task 12						

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

9

Use Automated Tools to Derive a Timeline Chart

10

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.

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

11

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.

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

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]

These slides are designed to accompany Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.   14

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,

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

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

15

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

16

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

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

17