CSCI 5828: Foundations of Software Engineering

Lecture 9 and 10: Planning the Software Project

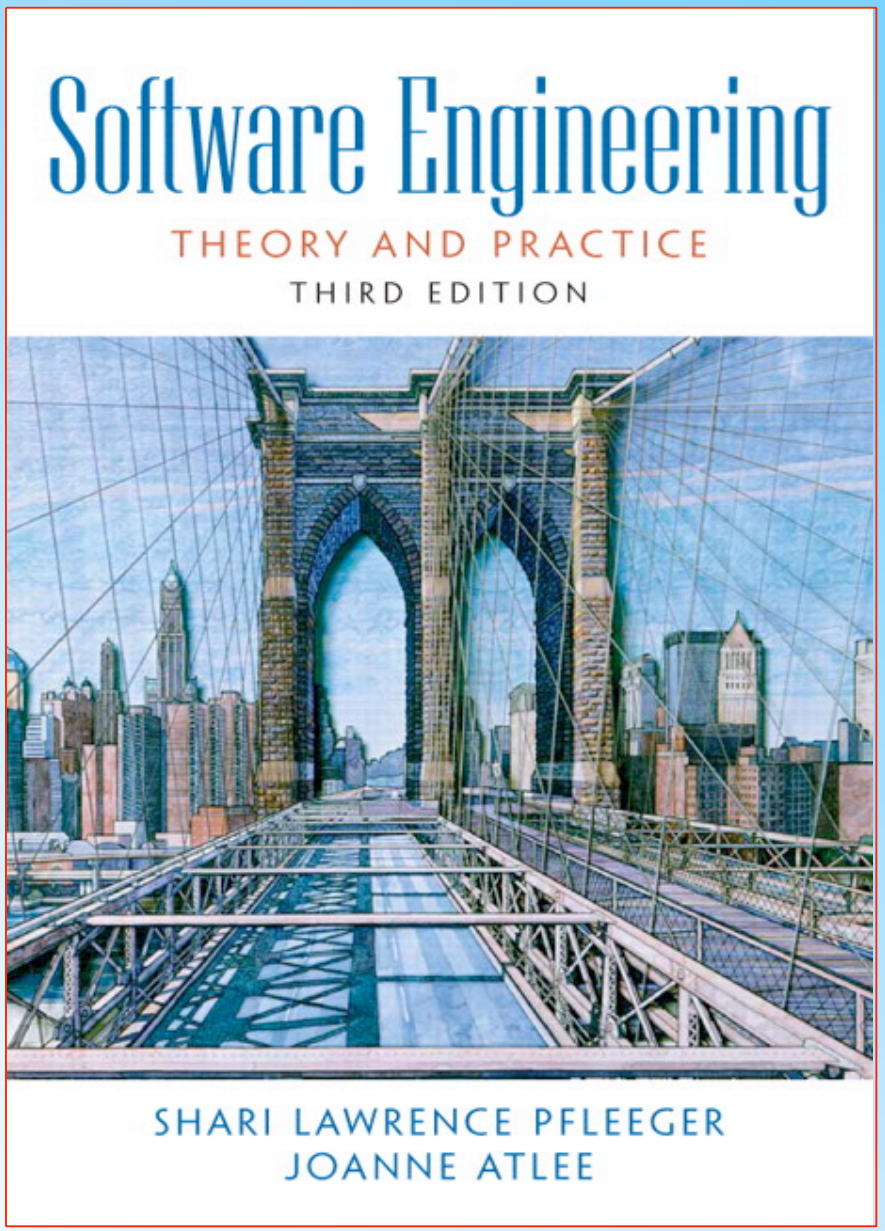
Slides created by Pfleeger and Atlee for the SE textbook

Some modifications to the original slides have been made by Ken Anderson for clarity of presentation

02/12/2008 — 02/14/2008

Chapter 3

Planning and Managing the Project



ISBN 0-13-146913-4
Prentice-Hall, 2006

Contents

Chapter 3 Objectives

- How do you track the progress of a software project?
- How should you organize project personnel?
- How do you make estimates of project effort and schedule?
- How do you manage risk?
- How do you integrate process modeling (Chapter 2) with project planning?

3.1 Tracking Progress

- Do you understand a customer's problems and needs?
- Can you design a system to solve a customer's problems or satisfy a customer's needs?
- How long will it take you to develop the system?
- How much will it cost to develop the system?

3.1 Tracking Progress
Project Schedule

- Describes the life cycle for a project by
  - enumerating the phases or stages of the project
  - decomposing each phase into tasks or activities to be completed
- Portrays the interactions among the activities
- Estimates the time that each task will take

3.1 Tracking Progress
Project Schedule: Approach

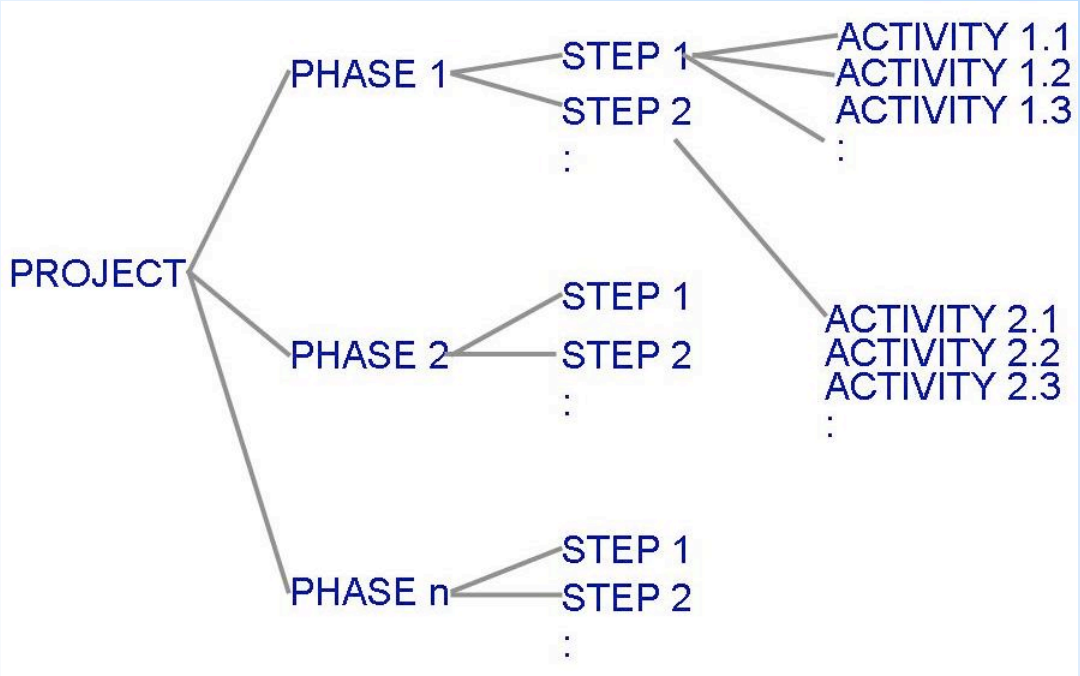
- Understanding customer's needs by listing all project deliverables
  - Documents
  - Demonstrations of function
  - Demonstrations of subsystems
  - Demonstrations of accuracy
  - Demonstrations of reliability, performance or security
- Determining milestones and activities to produce the deliverables

3.1 Tracking Progress
Milestones and activities

- **Activity**: takes place over a period of time
- **Milestone**: completion of an activity
  - a particular point in time
- **Precursor**: set of events that must occur to start an activity
- **Duration**: length of time needed to complete an activity
- **Due date or Deadline**: date by which an activity must be completed

3.1 Tracking Progress
Project Schedule (continued)

- Project development can be separated into a succession of phases which are composed of steps, which are composed of activities



3.1 Tracking Progress
Project Schedule (continued)

- Table 3.1 shows the phases, steps and activities to build a house
  - landscaping phase
  - building the house phase
- Table 3.2 lists milestones for building the house phase

3.1 Tracking Progress

Phases, Steps, and Activities in Building a House

Phase 1: Landscaping the lot	**Phase 2: Building the house**
Step 1.1: Clearing and grubbing	*Step 2.1: Prepare the site*
Activity 1.1.1: Remove trees	Activity 2.1.1: Survey the land
Activity 1.1.2: Remove stumps	Activity 2.1.2: Request permits
Step 1.2: Seeding the turf	Activity 2.1.3: Excavate for the foundation
Activity 1.2.1: Aerate the soil	Activity 2.1.4: Buy materials
Activity 1.2.2: Disperse the seeds	*Step 2.2: Building the exterior*
Activity 1.2.3: Water and weed	Activity 2.2. 1: Lay the foundation
Step 1.3: Planting shrubs and trees	Activity 2.2.2: Build the outside walls
Activity 1.3.1: Obtain shrubs and trees	Activity 2.2.3: Install exterior plumbing
Activity 1.3.2: Dig holes	Activity 2.2.4: Exterior electrical work
Activity 1.3.3: Plant shrubs and trees	Activity 2.2.5: Exterior siding
Activity 1.3.4: Anchor the trees and mulch around them	Activity 2.2.6: Paint the exterior
	Activity 2.2.7: Install doors and fixtures
	Activity 2.2.8: Install roof
	Step 2.3: Finishing the interior
	Activity 2.3.1: Install the interior plumbing
	Activity 2.3.2: Install interior electrical work
	Activity 2.3.3: Install wallboard
	Activity 2.3.4: Paint the interior
	Activity 2.3.5: Install floor covering
	Activit y 2.3.6: Install doors and fixtures

3.1 Tracking Progress
Milestones in Building a House

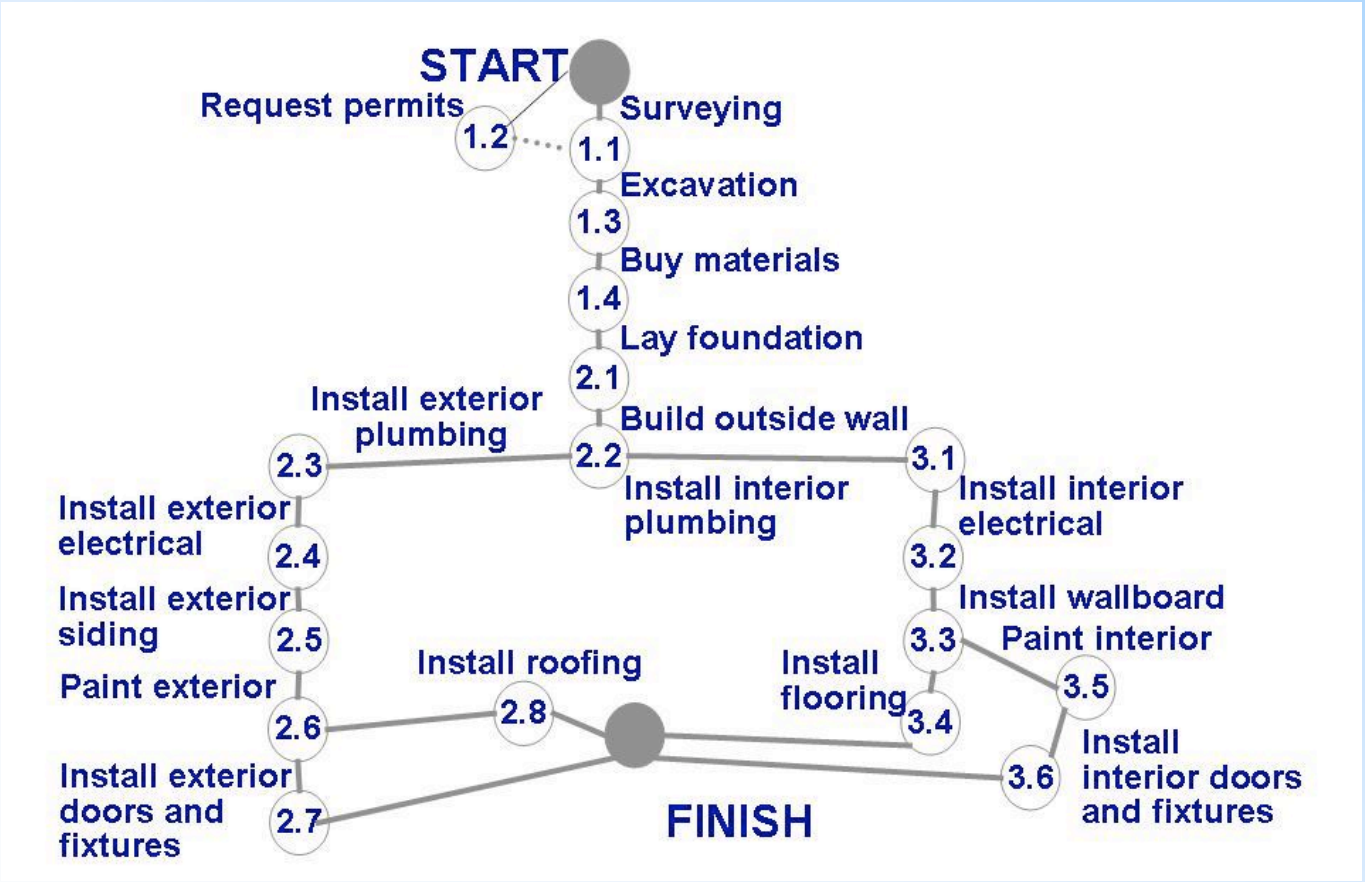
1.1. Survey complete
1.2. Permits issued
1.3. Excavation complete
1.4. Materials on hand
2.1. Foundation laid
2.2. Outside walls complete
2.3. Exterior plumbing complete
2.4. Exterior electrical work complete
2.5. Exterior siding complete
2.6. Exterior painting complete
2.7. Doors and fixtures mounted
2.8. Roof complete
3.1. Interior plumbing complete
3.2. Interior electrical work complete
3.3. Wallboard in place
3.4. Interior painting complete
3.5. Floor covering laid
3.6. Doors and fixtures mounted

3.1 Tracking Progress
Work Breakdown and Activity Graphs

- Work breakdown structure depicts the project as a set of discrete pieces of work
- Activity graphs depict the dependencies among activities
  - *Nodes*: project milestones
  - *Lines*: activities involved

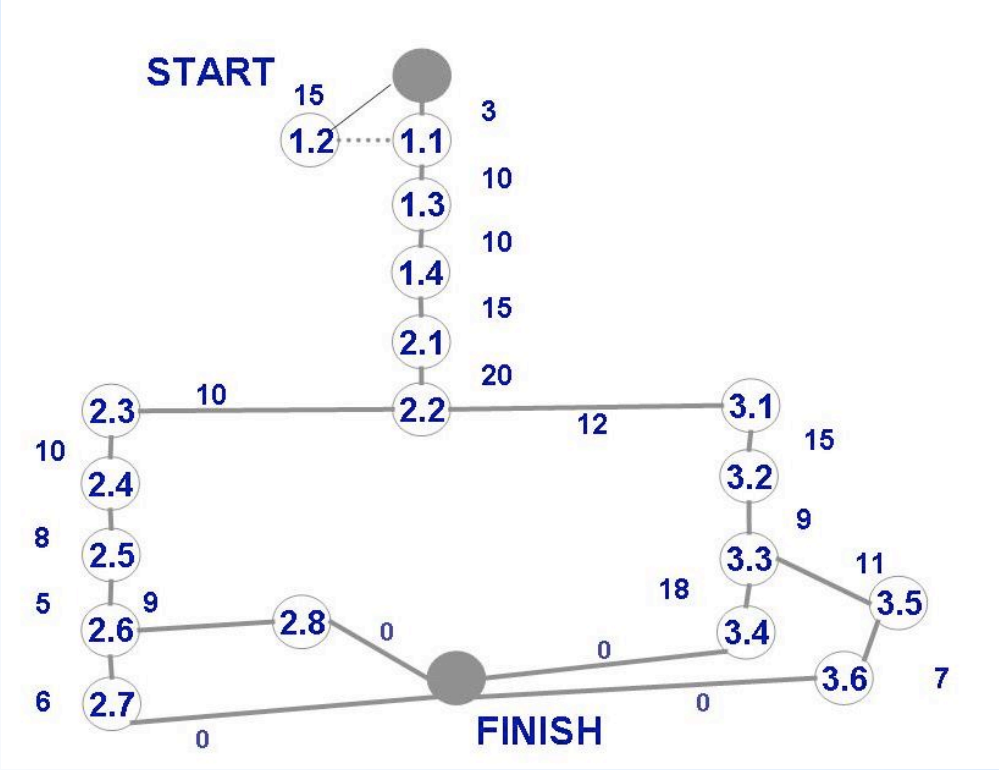
3.1 Tracking Progress
Work Breakdown and Activity Graphs (continued)

- Activity graph for building a house



3.1 Tracking Progress
Estimating Completion

- Adding estimated time in activity graph of each activity to be completed tells us more about the project's schedule



3.1 Tracking Progress
Estimating Completion for Building a House

Activity	Time estimate (in days)
Step 1: Prepare the site	
Activity 1.1: Survey the land	3
Activity 1.2: Request permits	15
Activity 1.3: Excavate for the foundation	10
Activity 1.4: Buy materials	10
Step 2: Building the exterior	
Activity 2.1: Lay the foundation	15
Activity 2.2: Build the outside walls	20
Activity 2.3: Install exterior plumbing	10
Activity 2.4: Exterior electrical work	10
Activity 2.5: Exterior siding	8
Activity 2.6: Paint the exterior	5
Activity 2.7: Install doors and fixtures	6
Activity 2.8: Install roof	9
Step 3: Finishing the interior	
Activity 3.1: Install the interior plumbing	12
Activity 3.2: Install interior electrical work	15
Activity 3.3: Install wallboard	9
Activity 3.4: Paint the interior	18
Activity 3.5: Install floor covering	11
Activity 3.6: Install doors and fixtures	7

3.1 Tracking Progress
Critical Path Method (CPM)

- Minimum amount of time it will take to complete a project
  - Reveals those activities that are most critical to completing the project on time
- **Real time (actual time):** estimated amount of time required for the activity to be completed
- **Available time:** amount of time available in the schedule for the activity's completion
- **Slack time:** the difference between the available time and the real time for that activity

3.1 Tracking Progress
Critical Path Method (CPM) (continued)

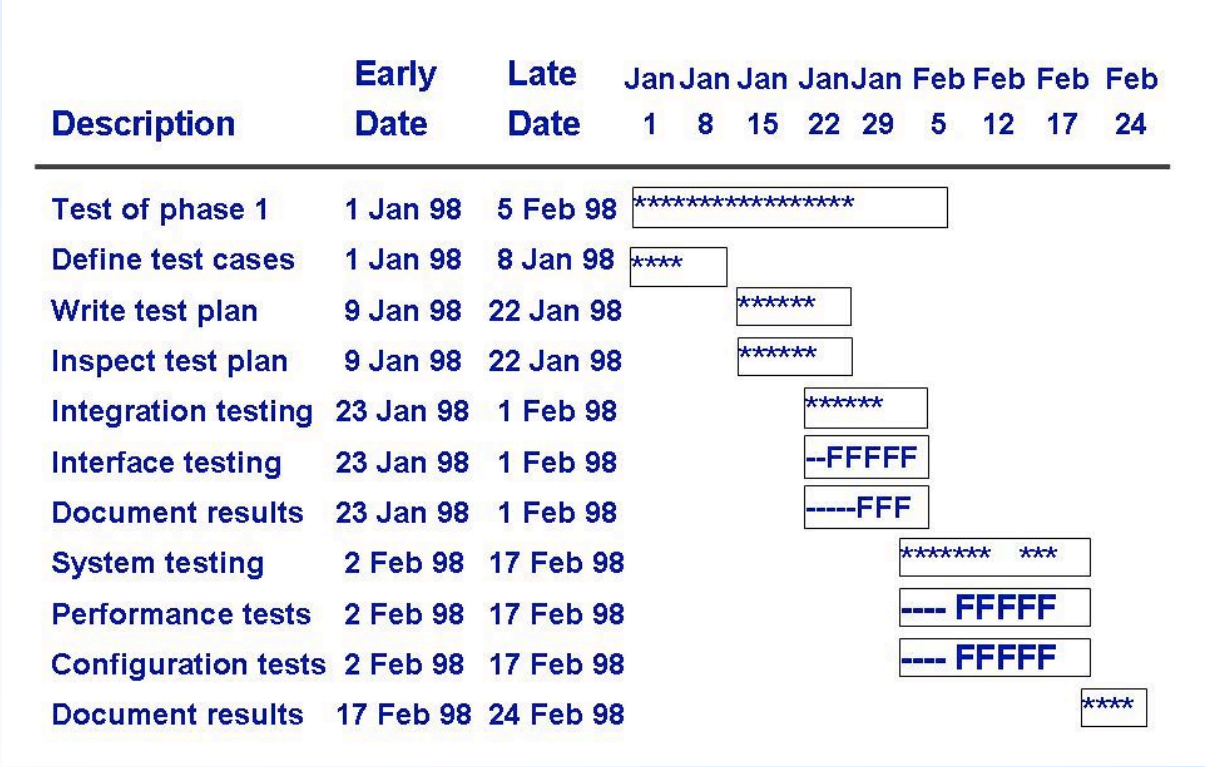
- **Critical path**: the slack at every node is zero
  - can be more than one in a project schedule
- **Slack time** = available time – real time
  = latest start time – earliest start time

Slack Time for Activities of Building a House

Activity	Earliest start time	Latest start time	Slack
1.1	1	13	12
1.2	1	1	0
1.3	16	16	0
1.4	26	26	0
2.1	36	36	0
2.2	51	51	0
2.3	71	83	12
2.4	81	93	12
2.5	91	103	12
2.6	99	111	12
2.7	104	119	15
2.8	104	116	12
3.1	71	71	0
3.2	83	83	0
3.3	98	98	0
3.4	107	107	0
3.5	107	107	0
3.6	118	118	0
Finish	124	124	0

3.1 Tracking Progress
CPM Bar Chart

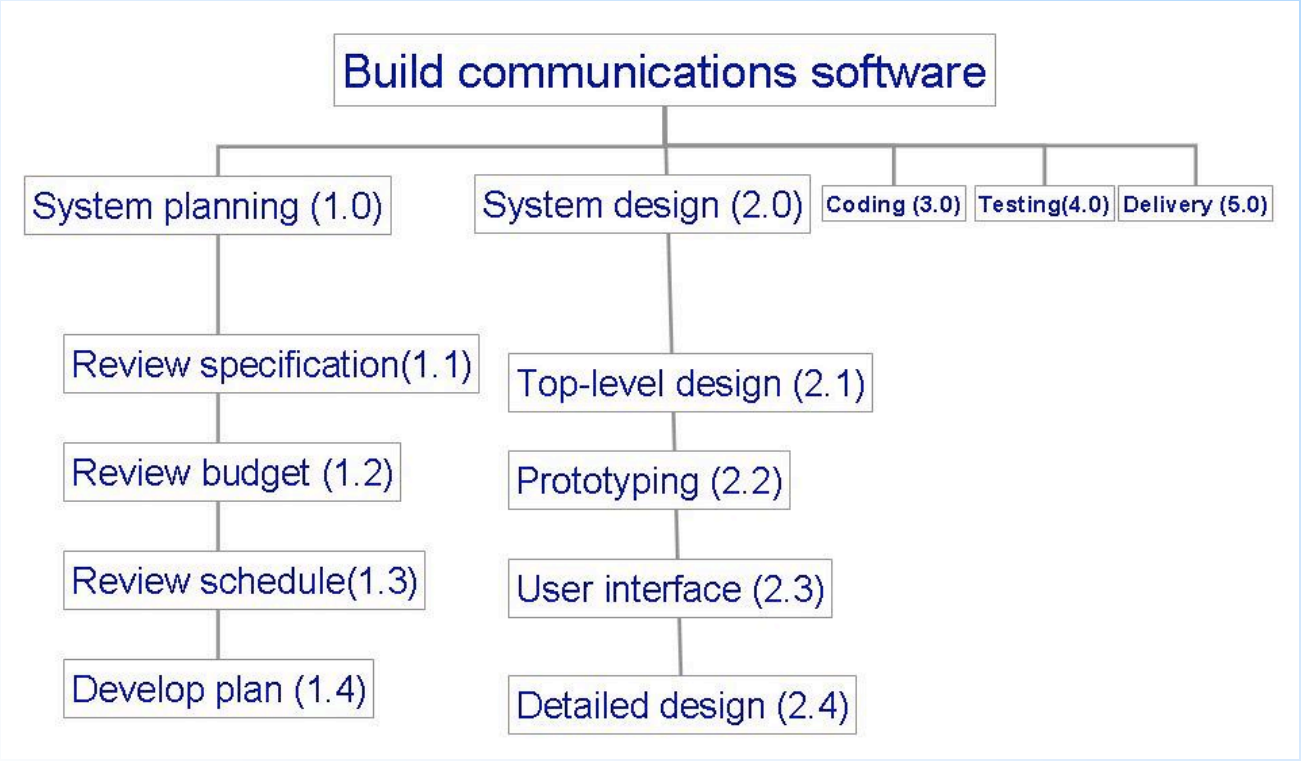
- Includes info about the early and late start dates
- Asterisks indicate the critical path



3.1 Tracking Progress

Tools to Track Progress

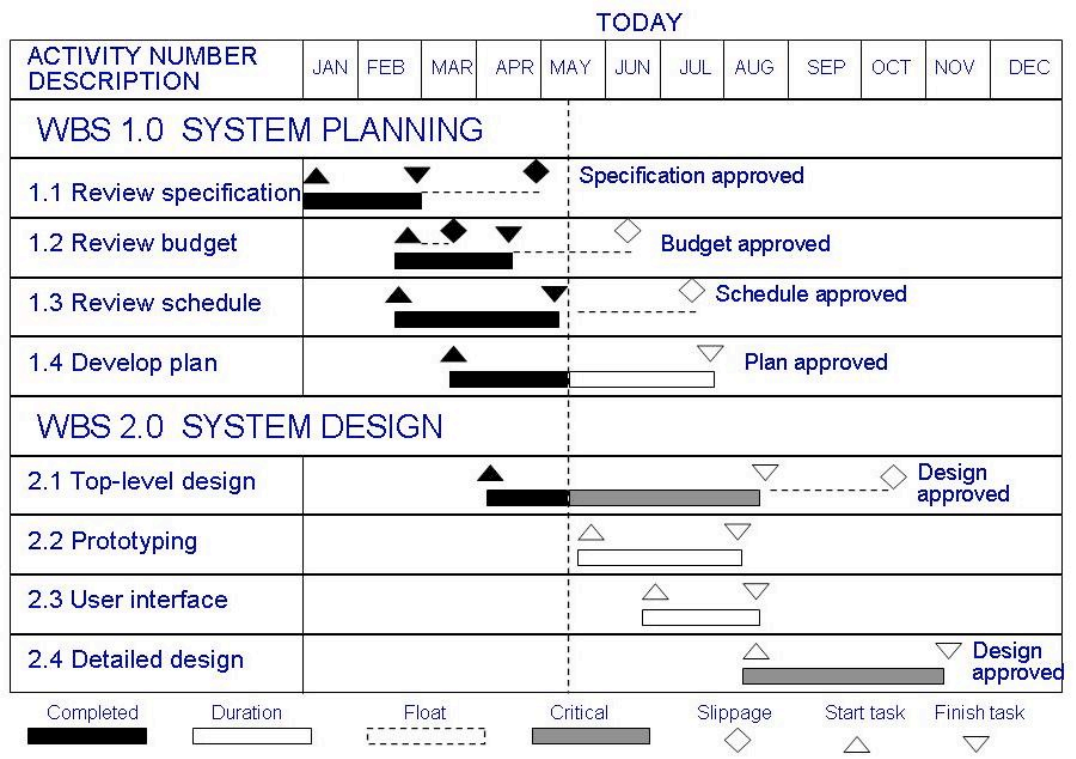
- Example: to track progress of building a communication software



3.1 Tracking Progress

Tools to Track Progress: Gantt Chart

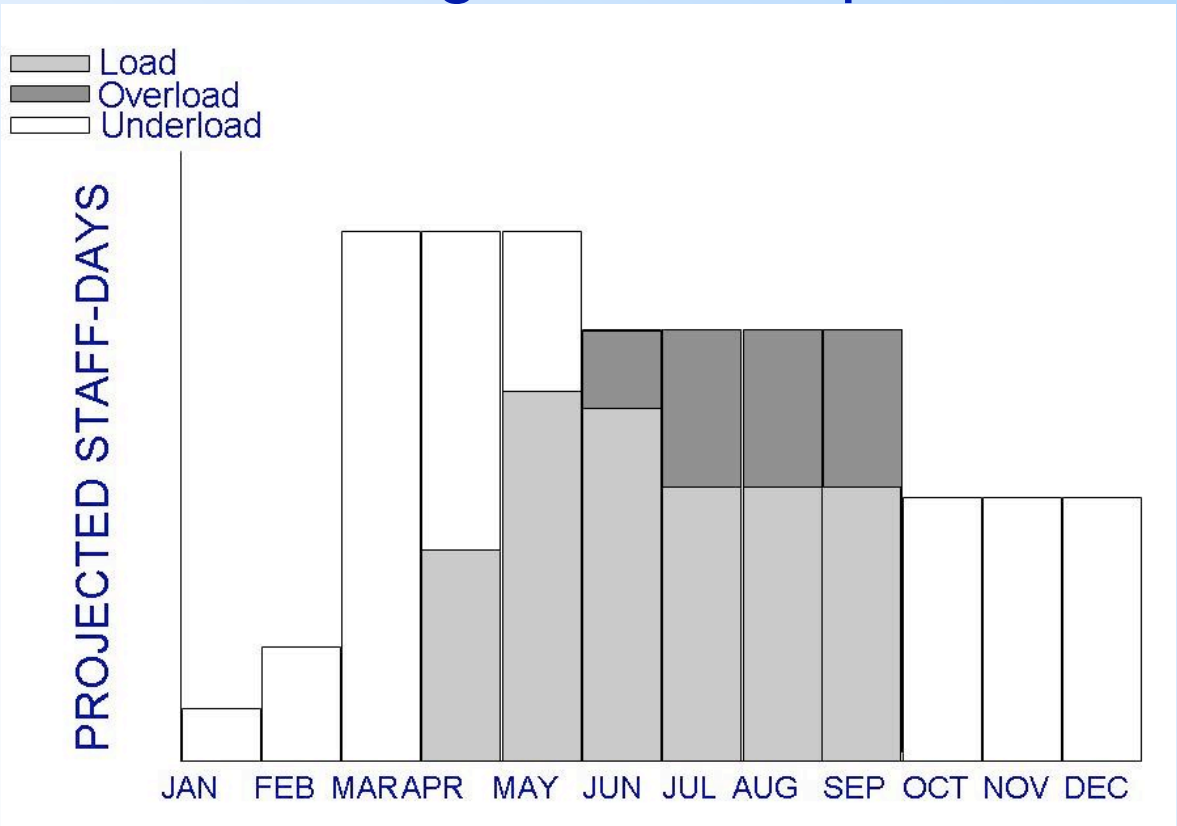
- Activities shown in parallel
  - helps understand which activities can be performed concurrently



3.1 Tracking Progress

Tools to Track Progress: Resource Histogram

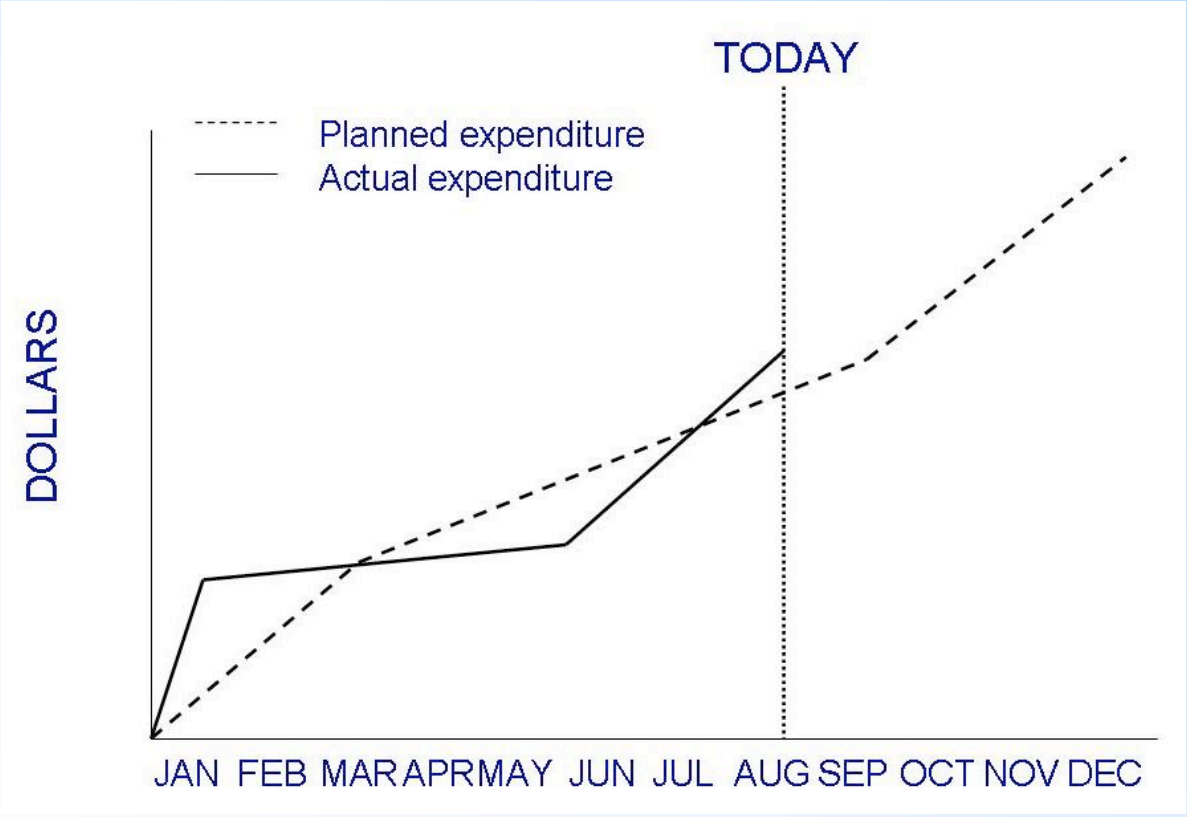
- Shows people assigned to the project and those needed for each stage of development



3.1 Tracking Progress

Tools to Track Progress: Expenditures Tracking

- An example of how expenditures can be monitored



3.2 Project Personnel

- Key activities requiring personnel
  - requirements analysis
  - system design
  - program design
  - program implementation
  - testing
  - training
  - maintenance
  - quality assurance
- There is a great advantage in assigning different responsibilities to different people

3.2 Project Personnel
Choosing Personnel

- Ability to perform work
- Interest in work
- Experience with
  - similar applications
  - similar tools, languages, or techniques
  - similar development environments
- Training
- Ability to communicate with others
- Ability to share responsibility
- Management skills

3.2 Project Personnel
Communication

- A project's progress is affected by
  - degree of communication
  - ability of individuals to communicate their ideas
- Software failures can result from breakdown in communication and understanding

- Sidebar: The Mythical Man-Month
  - Chapter 2 of Fred Brooks' book of the same name

The Mythical Man-Month (I)

- Books looks at the “man-month”, i.e. “person-month”, which is sometimes used to help schedule large projects
- There are several reasons why projects go beyond their initial schedule estimates
  - Developers are optimists
  - Our estimating techniques confuse “effort with progress, hiding the assumption that [people] and months are interchangeable”
  - Because we are uncertain about our estimates, we are unwilling to defend them
  - When schedule slippage is detected, we add more people to the project which is like “dousing a fire with gasoline”

The Mythical Man-Month (II)

- The unit of a person-month implies that workers and months are interchangeable
  - However, this is only true when a task can be partitioned among many workers with NO communication among them!
- Brooks points out that cost does indeed vary as the product of the number of workers and the number of months. Progress does not!

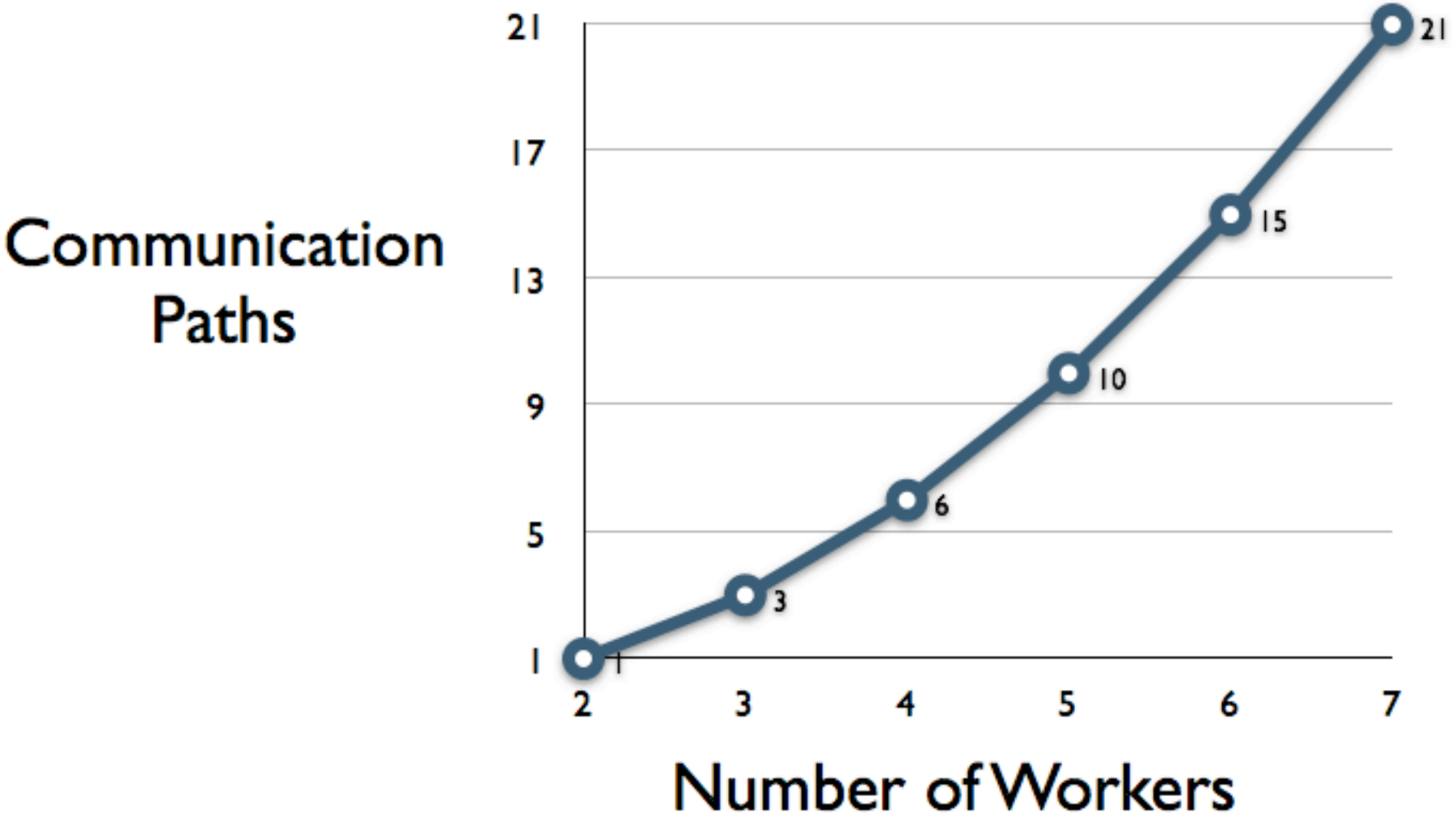
The Mythical Man-Month (III)

- When a task is sequential, more effort has no effect on the schedule
  - "The bearing of a child takes nine months, no matter how many women are assigned!"
- And, unfortunately, many tasks in software engineering have sequential constraints!
  - Especially debugging and system testing
    - Although, open source development challenges this notion a bit

The Mythical Man-Month (IV)

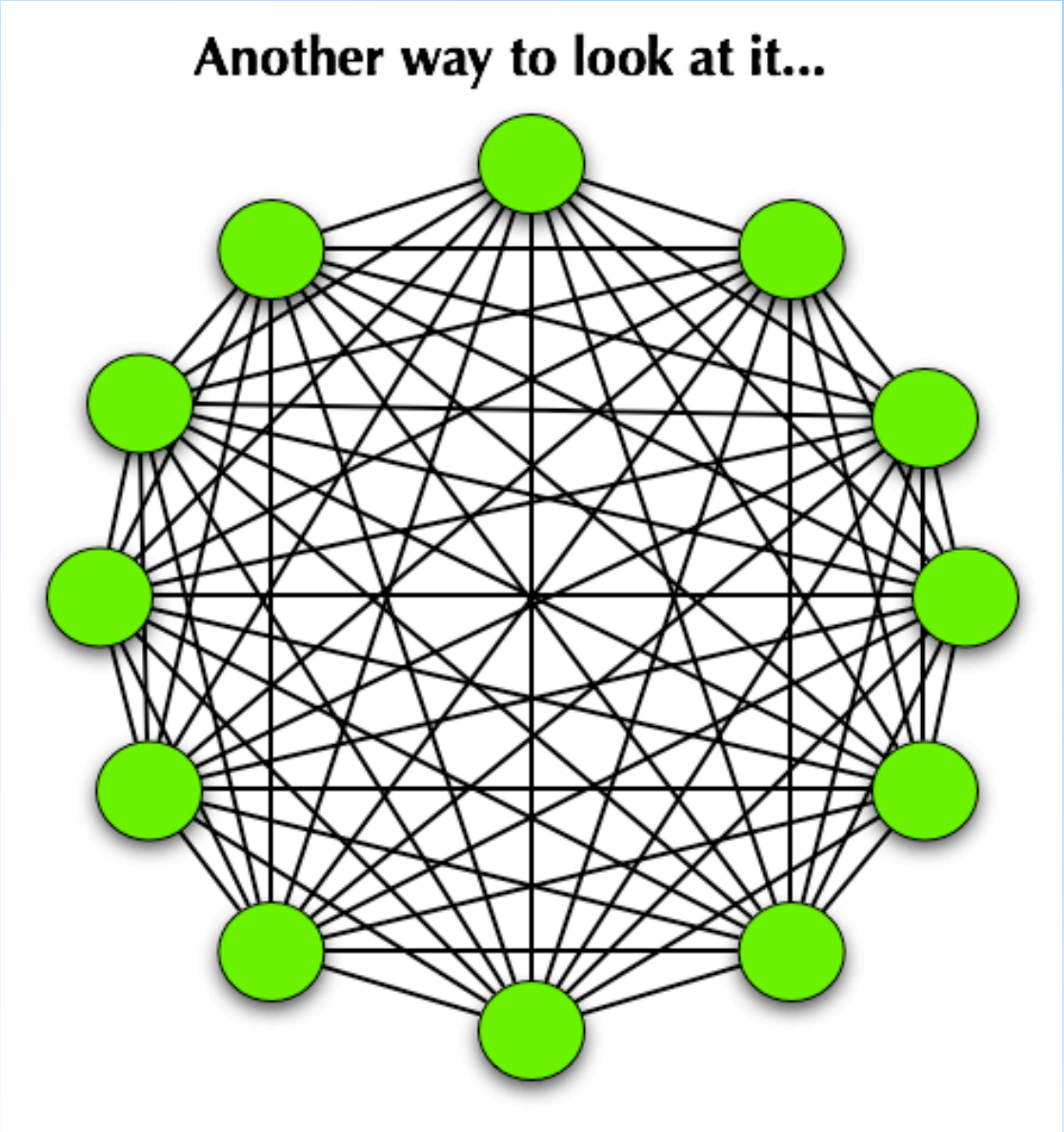
- In addition, most tasks require communication among workers
  - In software development, communication consists of
    - training
    - sharing information (intercommunication)
- Training will effect effort at worst linearly
  - if you have N people to train individually, it will take N*trainingTime minutes to train them
- Intercommunication on the other hand affects effort in a non-linear fashion, if each worker has to communicate with every other worker
  - i.e. if there are N workers there are N(N-1)/2 paths between them

The Mythical Man-Month (V)



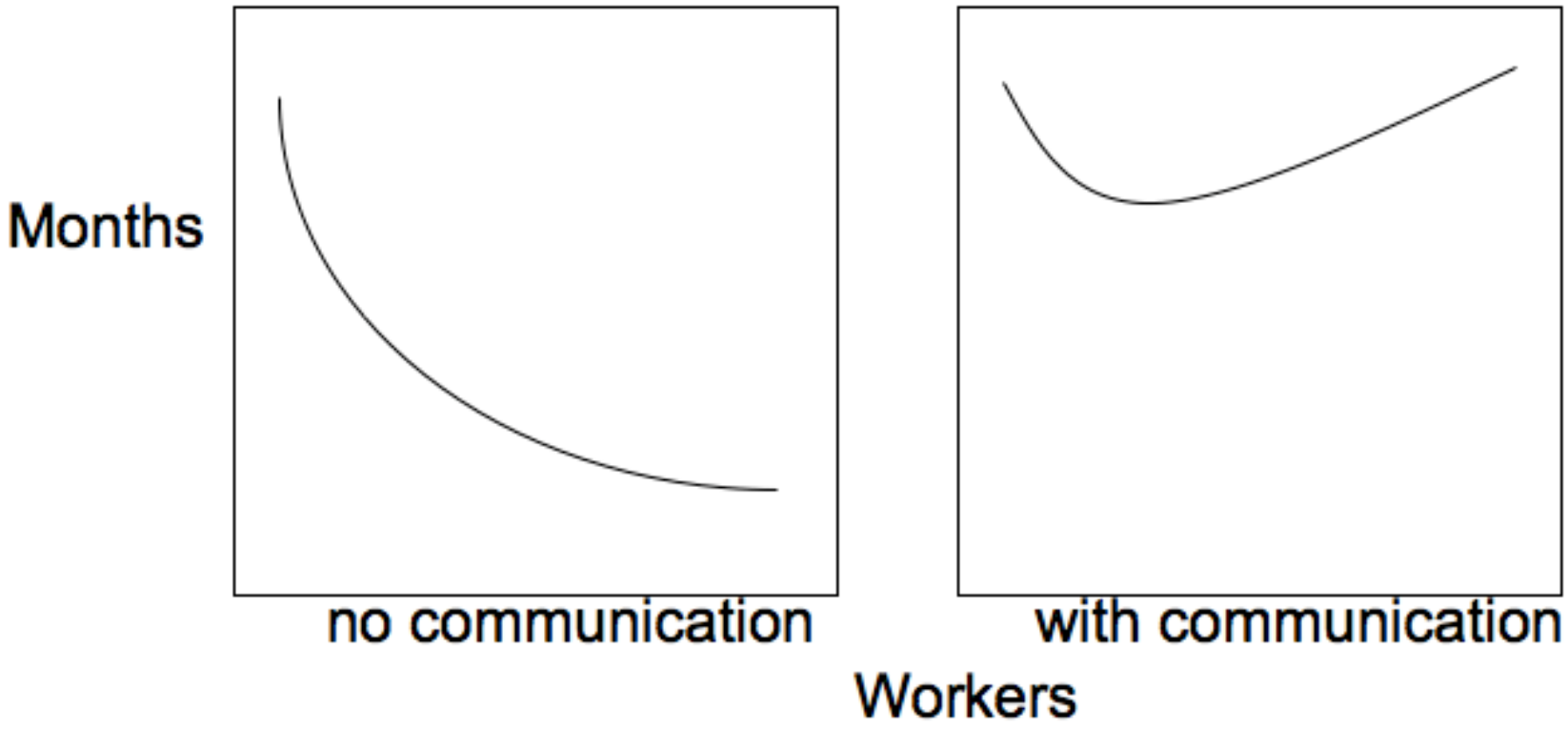
The Mythical Man-Month (VI)

- 12 workers
  - 66 paths!



The Mythical Man-Month (VII)

"Adding more people then lengthens, not shortens, the schedule!"
-- (A paraphrase of) Brooks' Law



The Mythical Man-Month (VIII)

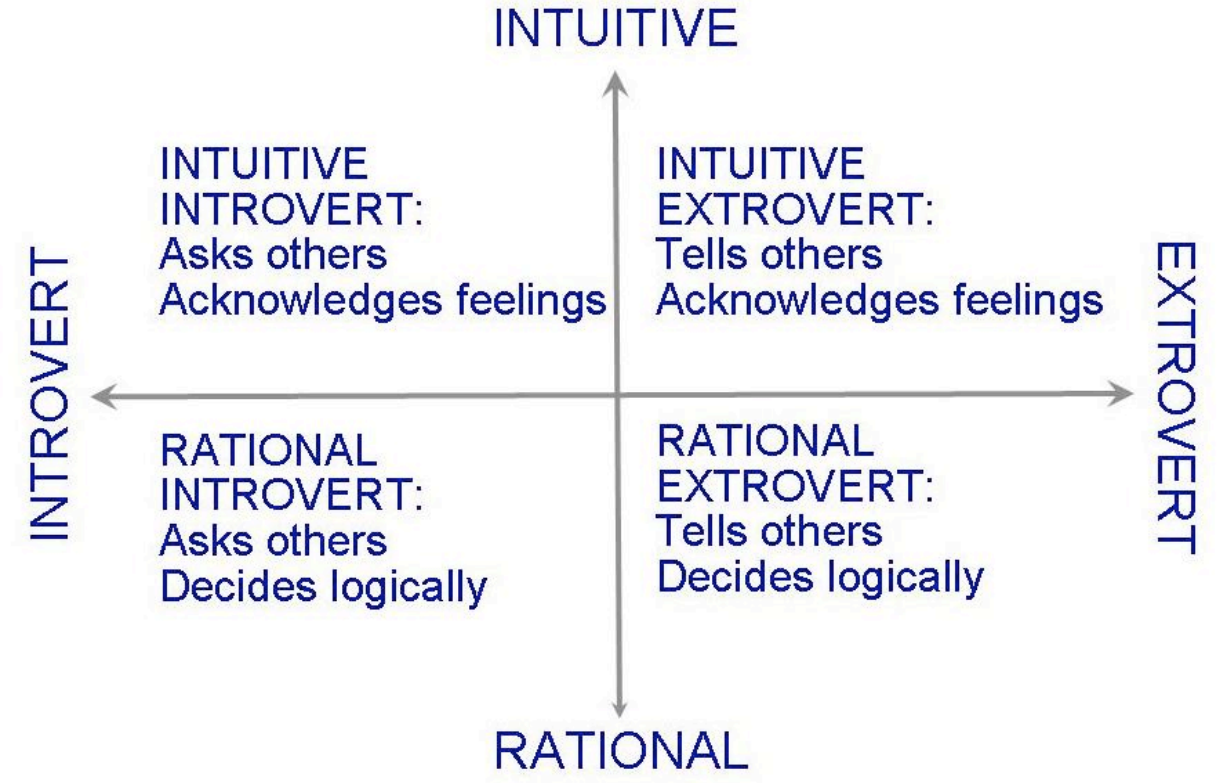
- How do we deal with this?
  - Team organization
  - Scheduling: Need better estimation techniques
- Brooks's Rule of Thumb for Scheduling Software Projects
  - 1/3 planning
  - 1/6 coding
  - 1/4 component test
  - 1/4 system test
- More time spent planning than normal
  - 50% of time allocated to testing!

3.2 Project Personnel
Work Styles

- **Extroverts:** tell their thoughts
- **Introverts:** ask for suggestions
- **Intuitives:** base decisions on feelings
- **Rationals:** base decisions on facts, options

3.2 Project Personnel
Work Styles (continued)

- Horizontal axis: communication styles
- Vertical axis: decision styles



3.2 Project Personnel
Work Styles (continued)

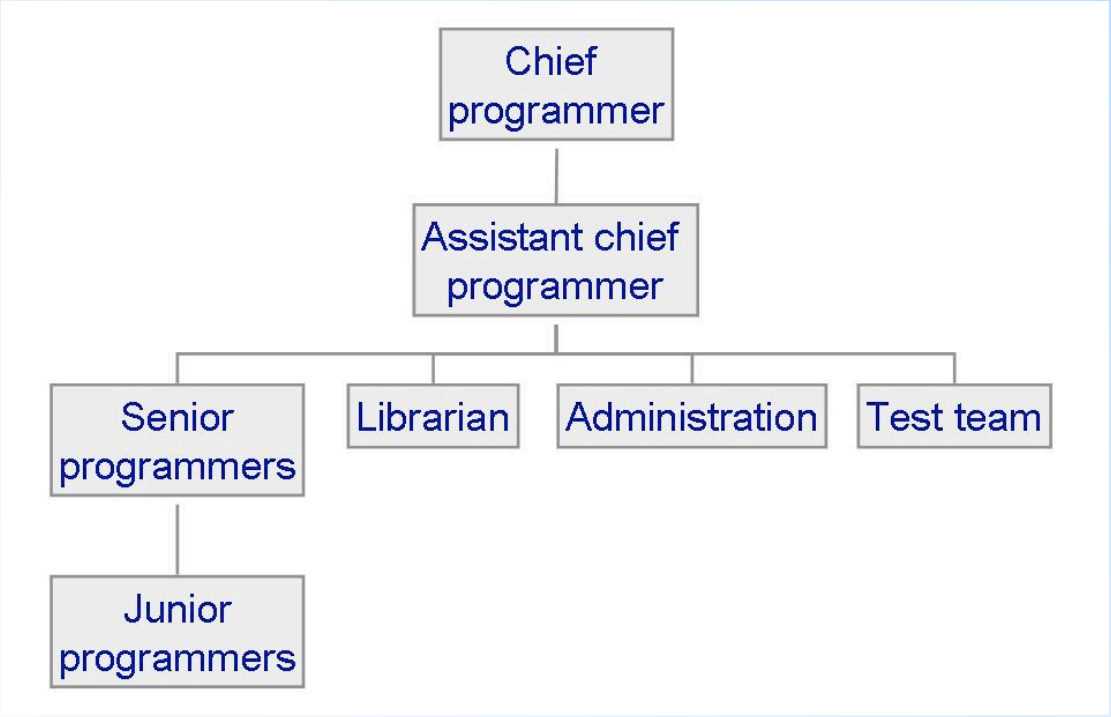
- Work styles determine communication styles
- Understanding workstyles
  - Helps you to be flexible
  - give information about other's priorities
- Affect interaction among customers, developers and users

3.2 Project Personnel
Project Organization

- Depends on
  - backgrounds and work styles of team members
  - number of people on team
  - management styles of customers and developers
- Examples:
  - *Chief programmer team*: one person totally responsible for a system's design and development
  - *Egoless approach:* hold everyone equally responsible

3.2 Project Personnel
Project Organization: Chief Programmer Team

- Each team member must communicate often with chief, but not necessarily with other team members

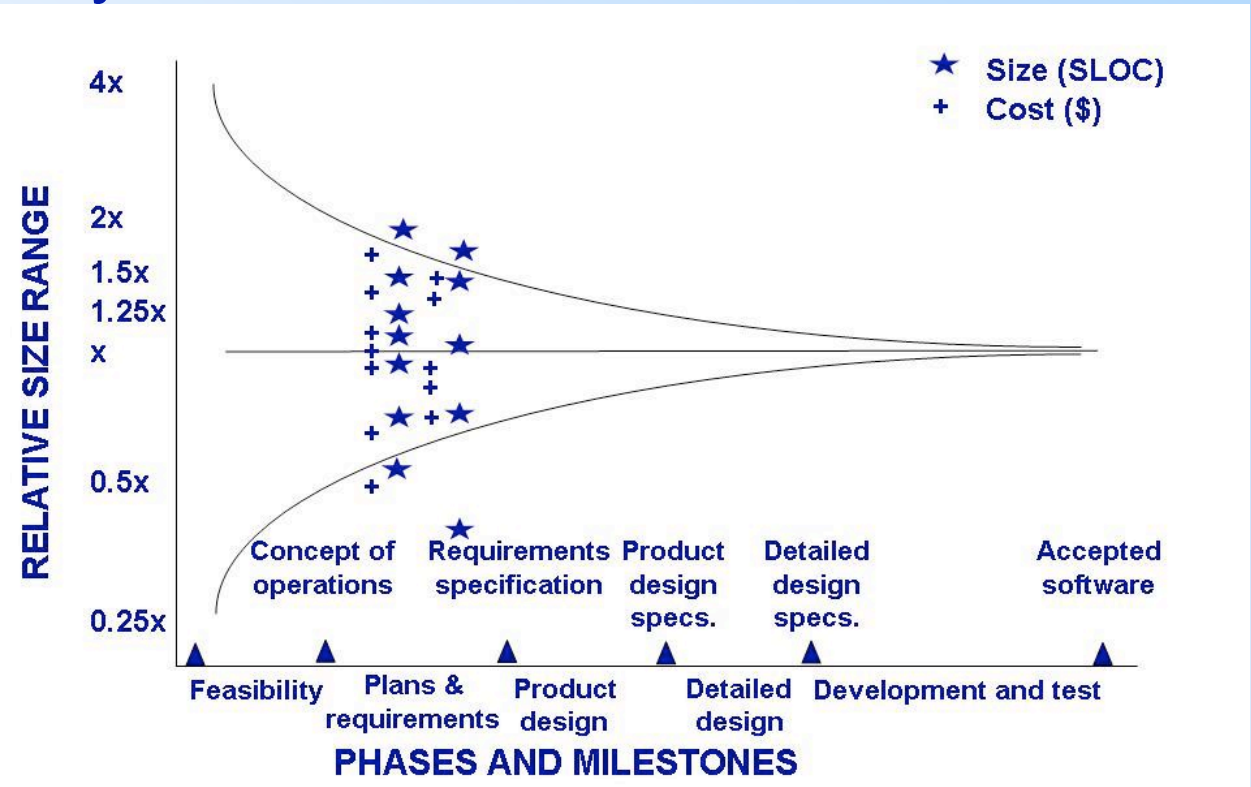


3.3 Effort Estimation

- Estimating project costs is one of the crucial aspects of project planning and management
- Estimating cost has to be done as early as possible during the project life cycle
- Type of costs
  - facilities: hardware space, furniture, telephone, etc
  - methods and tools
  - staff (effort): the biggest component of cost

3.3 Effort Estimation
Estimation Should be Done Repeatedly

- Uncertainty early in the project can affect the accuracy of cost and size estimations



3.3 Effort Estimation
Sidebar 3.3 Causes of Inaccurate Estimates

- Key causes
  - Frequent request for change by users
  - Overlooked tasks
  - User's lack of understanding of the requirements
  - Insufficient analysis when developing estimate
  - Lack of coordination of system development, technical services, operations, data administration, and other functions during development
  - Lack of an adequate method or guidelines for estimating

3.3 Effort Estimation

Sidebar 3.3 Causes of Inaccurate Estimates (continued)

- **Key influences**
  - Complexity of the proposed application system
  - Required integration with existing system
  - Complexity of the program in the system
  - Size of the system expressed as number of functions or programs
  - Capabilities of the project team members
  - Project team's experience with the application, the programming language, and hardware
  - Capabilities of the project team members
  - Database management system
  - Number of project team member
  - Extent of programming and documentation standards

3.3 Effort Estimation
Type of Estimation Methods

- Expert judgment
  - Top-down or bottom-up
  - Analogy: pessimistic *(x)*, optimistic *(y)*, most likely *(z)*; estimate as $(x + 4y + z)/6$
  - Delphi technique: based on the average of "secret" expert judgments
  - Wolverton model: old (mid 70's)
- Algorithmic methods: $E = (a + bS^c)\, m(\mathbf{X})$
  - Walston and Felix model: $E = 5.25S^{0.91}$
  - Bailey and Basili model: $E = 5.5 + 0.73S^{1.16}$

3.3 Effort Estimation
Expert Judgement: Wolverton Model

- Two factors that affect difficulty
  - whether problem is old (O) or new (N)
  - whether it is easy (E) or moderate (M)

	Difficulty					
Type of software	*OE*	*OM*	*OH*	*NE*	*NM*	*NH*
Control	21	27	30	33	40	49
Input/output	17	24	27	28	35	43
Pre/post processor	16	23	26	28	34	42
Algorithm	15	20	22	25	30	35
Data management	24	31	35	37	46	57
Time-critical	75	75	75	75	75	75

3.3 Effort Estimation
Algorithmic Method: Watson and Felix Model

- A productivity index is included in the equation
- There are 29 factors that can affect productivity
  - 1 if it increases the productivity
  - 0 if it decreases the productivity

3.3 Effort Estimation
Watson and Felix Model Productivity Factors

1. Customer interface complexity	16. Use of design and code inspections
2. User participation in requirements definition	17. Use of top -down development
3. Customer -originated program design changes	18. Use of a chief programmer team
4. Customer experience with the application area	19. Overall complexity of code
5. Overall personnel experience	20. Complexity of application processing
6. Percentage of development programmers who participated in the design of functional specifications	21. Co mplexity of program flow
7. Previous experience with the operational computer	22. Overall constraints on program's design
8. Previous experience with the programming language	23. Design constraints on the program's main storage
9. Previous experien ce with applications of similar size and complexity	24. Design constraints on the program's timing
10. Ratio of average staff size to project duration (people per month)	25. Code for real -time or interactive operation or for execution under severe time constraints
11. Hardware under concurrent development	26. Percentage of code for delivery
12. Access to development computer open under special request	27. Code classified as nonmathematical application and input/output formatting programs
13. Acc ess to development computer closed	28. Number of classes of items in the database per 1000 lines of code
14. Classified security environment for computer and at least 25% of programs and data	29. Number of pages of delivered documentation per 1000 line s of code
15. Use of structured programming	

3.3 Effort Estimation
Agorithmic Method: Bailey-Basili technique

- Minimize standard error estimate to produce an equation such as $E = 5.5 + 0.73S^{1.16}$
- Adjust initial estimate based on the difference ratio
  - If R is the ratio between the actual effort, E, and the predicted effort, E', then the effort adjustment is defined as
  - $ER_{adj} = R - 1$ if $R > 1$
    $= 1 - 1/R$ if $R < 1$
- Adjust the initial effort estimate E_{adj}
  - $E_{adj} = (1 + ER_{adj})E$ if $R > 1$
    $= E/(1 + ER_{adj})$ if $R < 1$

3.3 Effort Estimation
Agorithmic Method: Bailey-Basily Modifier

Total methodology (METH)	*Cumulative complexity (CPLX)*	*Cumulative experience (EXP)*
Tree charts	Customer interface complexity	Programmer qualifications
Top-down design	Application complexity	Programmer machine experience
Formal documentation	Program flo w complexity	Programmer language experience
Chief programmer teams	Internal communication complexity	Programmer application experience
Formal training	Database complexity	Team experience
Formal test plans	External communication complexity	
Design forma lisms	Customer-initiated program design changes	
Code reading		
Unit development folders		

3.3 Effort Estimation
COCOMO model

- Introduced by Boehm
- COCOMO II
  - updated version
  - include models of reuse
- The basic models
  - $E = bS^c m(X)$
  - where
    - bS^c is the initial size-based estimate
    - $m(\mathbf{X})$ is the vector of cost driver information

3.3 Effort Estimation
COCOMO II: Stages of Development

- Application composition
  - prototyping to resolve high-risk user interface issues
  - size estimates in object points
- Early design
  - to explore alternative architectures and concepts
  - size estimates in function points
- Postarchitecture
  - development has begun
  - size estimates in lines of code

Three Stages of COCOMO II

Model Aspect	Stage 1: Application Composition	Stage 2: Early Design	Stage 3: Post-architecture
Size	Application points	Function points (FP) and language	FP and language or source lines of code (SLOC)
Reuse	Implicit in model	Equivalent SLOC as function of other variables	Equival ent SLOC as function of other variables
Requirements change	Implicit in model	% change expressed as a cost factor	% change expressed as a cost factor
Maintenance	Application Point Annual Change Traffic	Function of ACT, software understanding, unfamiliarity	Function of ACT, s oftware understanding, unfamiliarity
Scale (c) in nominal effort equation	1.0	0.91 to 1.23, depending on precedentedness, conformity , early architecture, risk resolution, team cohesion, and SEI process maturity	0.91 to 1.23, depending on precedentedness, conformity, early architecture, risk resolution, team cohesion, and SEI process maturity
Product cost drivers	None	Complexity, required reusability	Reliability, database size, documentation needs, required reuse, and product complexity
Platform cost drivers	None	Platform difficulty	Execution time constraints, main storage constraints, and virtual machine volatility
Personnel cost drivers	None	Personnel capability and experience	Analyst capability, applications experience, programmer capability, language and tool programmer experience, experience, and personnel continuity
Project cost drivers	None	Required development schedule, devel opment environment multisite development	Use of software tools, required development schedule, and

3.3 Effort Estimation
COCOMO II: Estimate Application Points

- To compute application points, first we need to count the number of screens, reports, and programming language used to determine the complexity level

For Screens				*For Reports*			
	Number and source of data tables				*Number and source of data tables*		
Number of views contained	Total < 4 (<2 server, <3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)	*Number of sections contained*	Total < 4 (<2 server, <3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)
<3	simple	simple	medium	0 or 1	simple	simple	medium
3 - 7	simple	medium	difficult	2 or 3	simple	medium	difficult
8 +	medium	difficult	difficult	4 +	medium	difficult	difficult

3.3 Effort Estimation
COCOMO II: Estimate Application Point (continued)

- Determine the relative effort required to implement a report or screen simple, medium, or difficult
- Calculate the productivity factor based on developer experience and capability
- Determine the adjustment factors expressed as multipliers based on rating of the project

3.3 Effort Estimation
Complexity Weights for Application Points

Object type	*Simple*	*Medium*	*Difficult*
Screen	1	2	3
Report	2	5	8
3GL component	-	-	10

3.3 Effort Estimation
Productivity Estimate Calculation

Developers' experience and capability	Very low	Low	Nominal	High	Very high
CASE maturity and capability	Very low	Low	Nominal	High	Very high
Productivity factor	4	7	13	25	50

3.3 Effort Estimation
Tool Use Categories

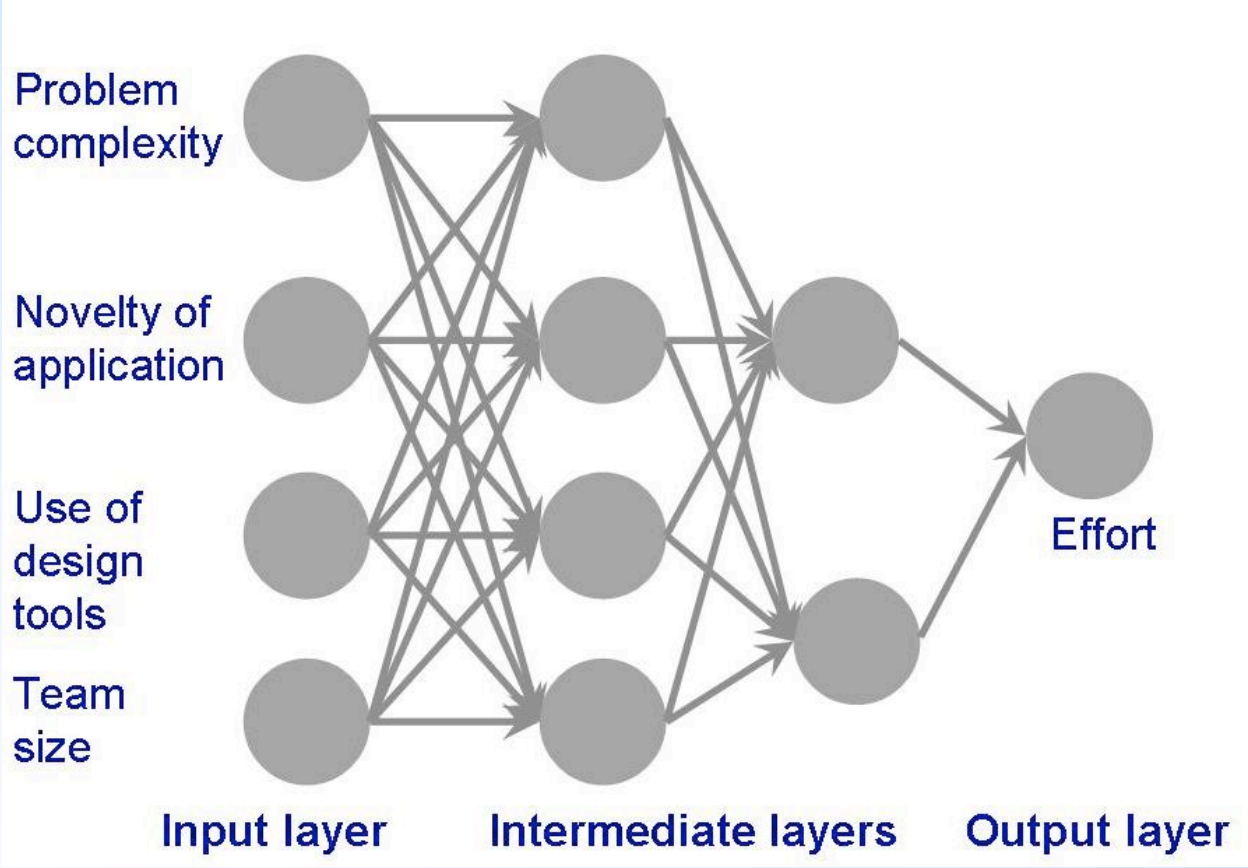
Category	*Meaning*
Very low	Edit, code, debug
Low	Simple front-end, back-end CASE, little integration
Nominal	Basic life-cycle tools, moderately integrated
High	Strong, mature life-cycle tools, moderately integrated
Very high	Strong, mature, proactive life-cycle tools, well-integrated with processes, methods, reuse

3.3 Effort Estimation
Machine Learning Techniques

- Example: case-based reasoning (CBR)
  - user identifies new problem as a case
  - system retrieves similar cases from repository
  - system reuses knowledge from previous cases
  - system suggests solution for new case
- Example: neural network
  - cause-effect network "trained" with data from past history

3.3 Effort Estimation
Machine Learning Techniques: Neural Network

- Neural network used by Shepperd to produce effort estimation



3.3 Effort Estimation
Machine Learning Techniques: CBR

- Involves four steps
  - the user identifies a new problem as a case
  - the system retrieves similar case from a respository of historical information
  - the system reuses knowledge from previous case
  - the system suggests a solution for the new case
- Two big hurdles in creating successful CBR system
  - characterizing cases
  - determining similarity

3.3 Effort Estimation
Finding the Model for Your Situation

- Mean magnitude of relative error (MMRE)
  - absolute value of mean of [(actual - estimate)/actual]
  - goal: should be .25 or less
- Pred(x/100): percentage of projects for which estimate is within *x%* of the actual
  - goal: should be .75 or greater for *x* = .25

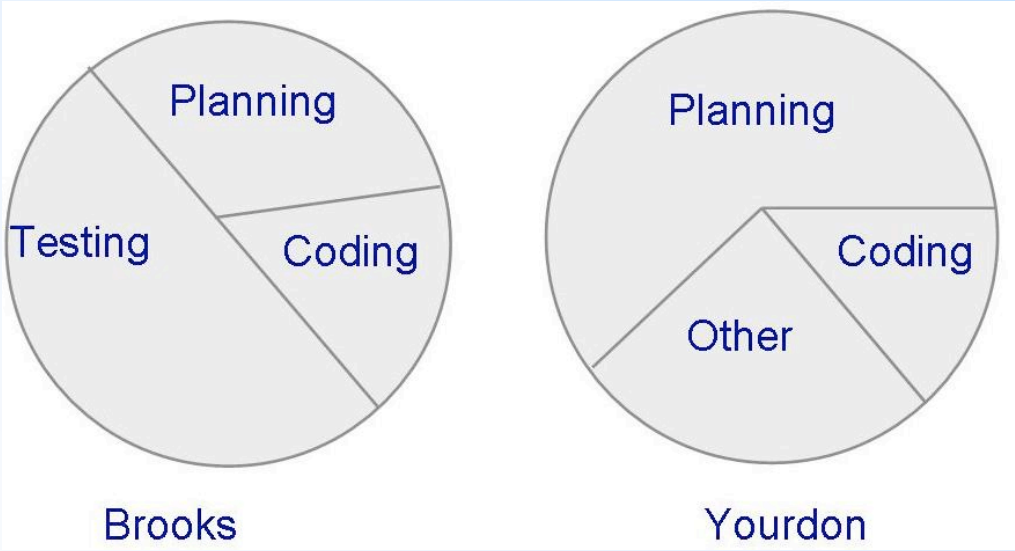
3.3 Effort Estimation
Evaluating Models (continued)

- No model appears to have captured the essential characteristics and their relationships for all types of development

Model	*PRED(0.25)*	*MMRE*
Walston-Felix	0.30	0.48
Basic COCOMO	0.27	0.60
Intermediate COCOMO	0.63	0.22
Intermediate COCOMO (variation)	0.76	0.19
Bailey-Basili	0.78	0.18
Pfleeger	0.50	0.29
SLIM	0.06-0.24	0.78-1.04
Jensen	0.06-0.33	0.70-1.01
COPMO	0.38-0.63	0.23-5.7
General COPMO	0.78	0.25

3.3 Effort Estimation
Evaluating Models (continued)

- It is important to understand which types of effort are needed during development even when we have reasonably accurate estimate
- Two different reports of effort distribution from different researchers



Planning Poker: What Agile Does

- A student pointed me to a website that is used by teams using agile life cycles to manage their software development
  - Planning Poker: <http://planningpoker.com/>
- Basic idea:
  - A feature is proposed; on-line discussion occurs
  - Once all questions have been asked, each person picks a card with an estimate of how long it will take to implement the feature
  - Once each person has picked a card, the estimates are shown to all people; discussion occurs again
  - repeat until the team has agreed to the estimate

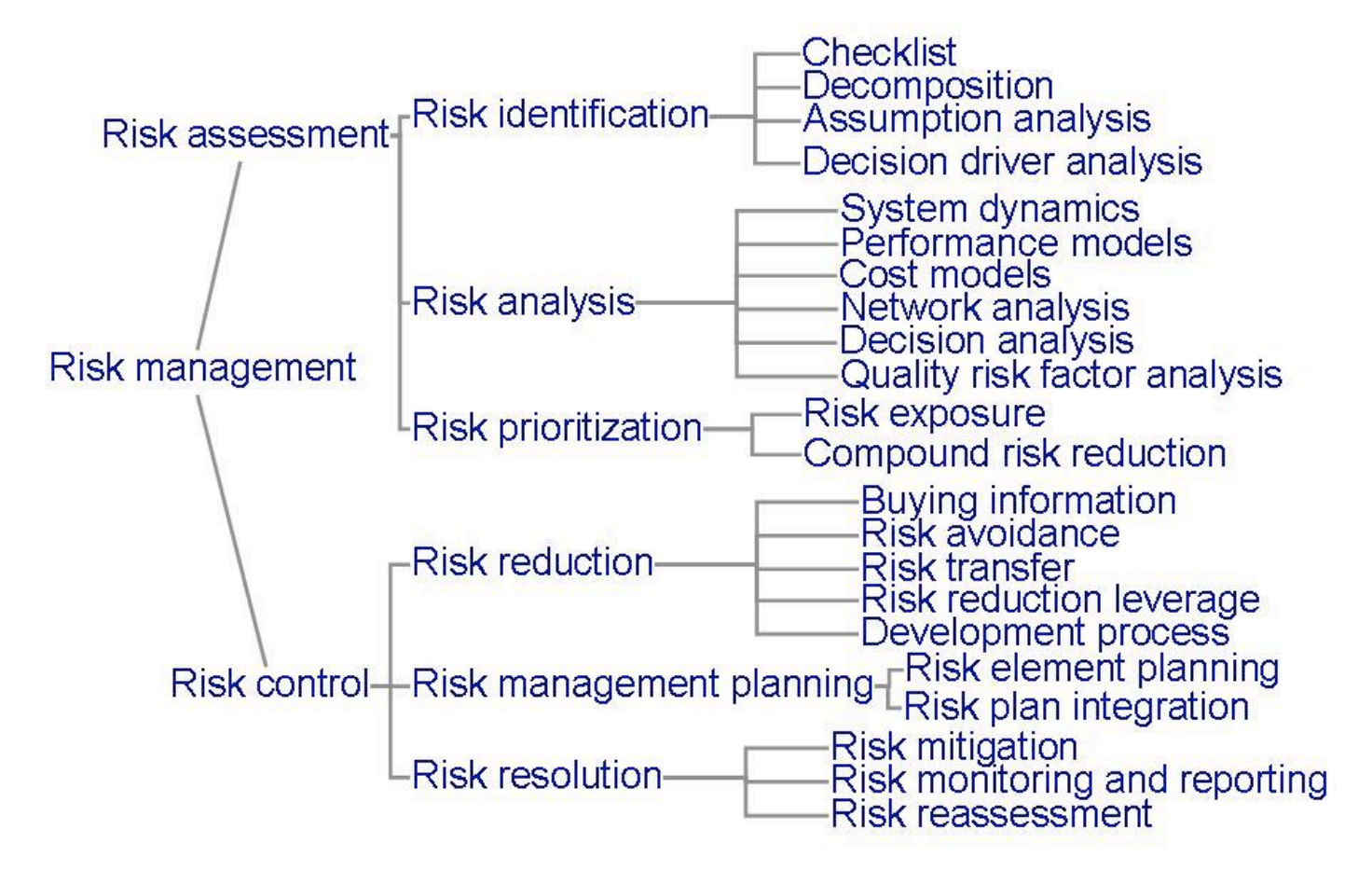
3.4 Risk Management
What is a Risk?

- Risk is an unwanted event that has negative consequences
- Distinguish risks from other project events
  - *Risk impact:* the loss associated with the event
  - *Risk probability*: the likelihood that the event will occur
  - *Risk control*: the degree to which we can change the outcome
- Quantify the effect of risks
  - *Risk exposure* = (risk probability) x (risk impact)
- Risk sources: generic and project-specific

3.4 Risk Management
Sidebar 3.4 Boehm's Top Ten Risk Items

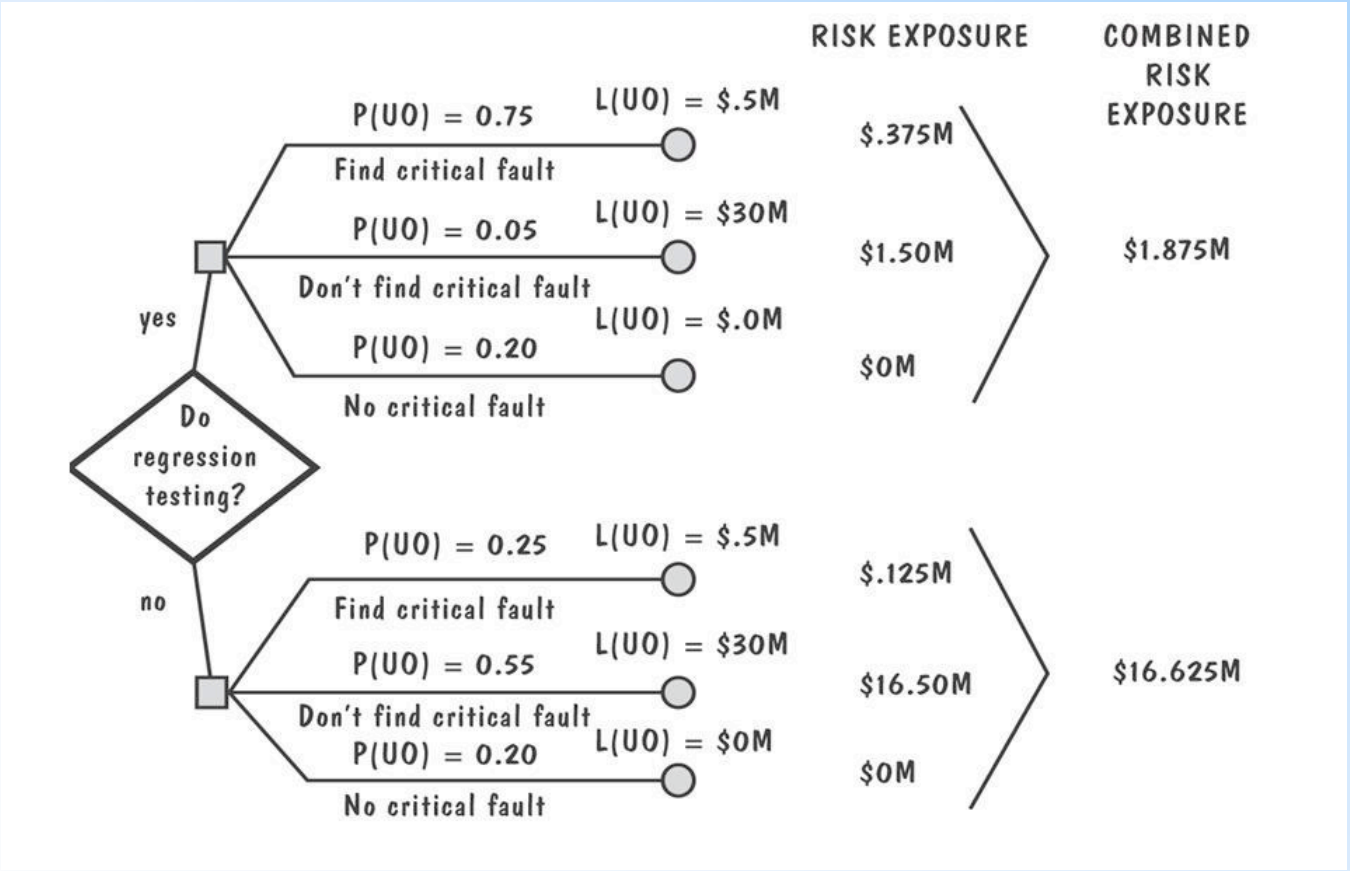
- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions
- Developing the wrong user interfaces
- Gold-plating (adding more to a system than specified in the requirements)
- Continuing stream of requirements changes
- Shortfalls in externally-performed tasks
- Shortfalls in externally-furnished components
- Real-time performance shortfalls
- Straining computer science capabilities

3.4 Risk Management
Risk Management Activities



3.4 Risk Management
Risk Management Activities (continued)

- Example of risk exposure calculation



3.4 Risk Management
Risk Management Activities (continued)

- Three strategies for risk reduction
  - *Avoiding the risk*: change requirements for performance or functionality
  - *Transferring the risk*: transfer to other system, or buy insurance
  - *Assuming the risk*: accept and control it
- Cost of reducing risk
  - *Risk leverage* = (risk exposure before reduction – (risk exposure after reduction) / (cost of risk reduction)
  - Example: <http://syque.com/improvement/Risk%20Reduction%20Leverage.htm>

Project Plan

- Created to communicate risk analysis and management, project cost estimates, schedule, and other important information about a proposed project to customers
- Will vary across organizations
- Note: It is NOT the same as requirements documents, design documents, etc.
- Instead, it's the document that organizes the management of the development project

3.5 Project Plan
Project Plan Contents

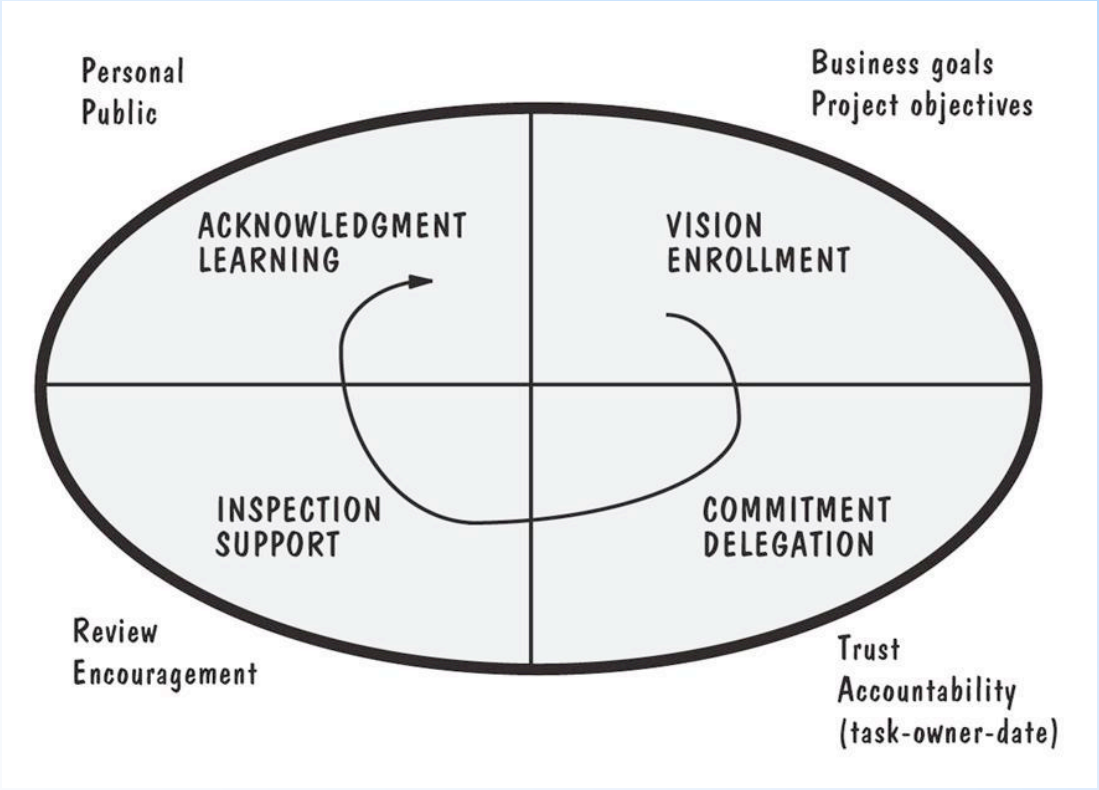
- Project scope
- Project schedule
- Project team organization
- Technical description of system
- Project standards and procedures
- Quality assurance plan
- Configuration management plan
- Documentation plan
- Data management plan
- Resource management plan
- Test plan
- Training plan
- Security plan
- Risk management plan
- Maintenance plan

3.6 Process Models and Project Management
Enrollment Management Model: Digital Alpha AXP

- Digital's Alpha AXP Project
  - Created new system architecture involving the creation of four new operating systems, 22 development teams, many products
- Developed process/project management approach that
  - Established an appropriately large shared vision for entire team
  - Delegated decisions completely and elicited specific commitments from participants
  - Inspected vigorously and provided supportive feedback
  - Acknowledged every advance and learn as the program progresses
    - Rewards based on recognition, not money (sim. to open src)

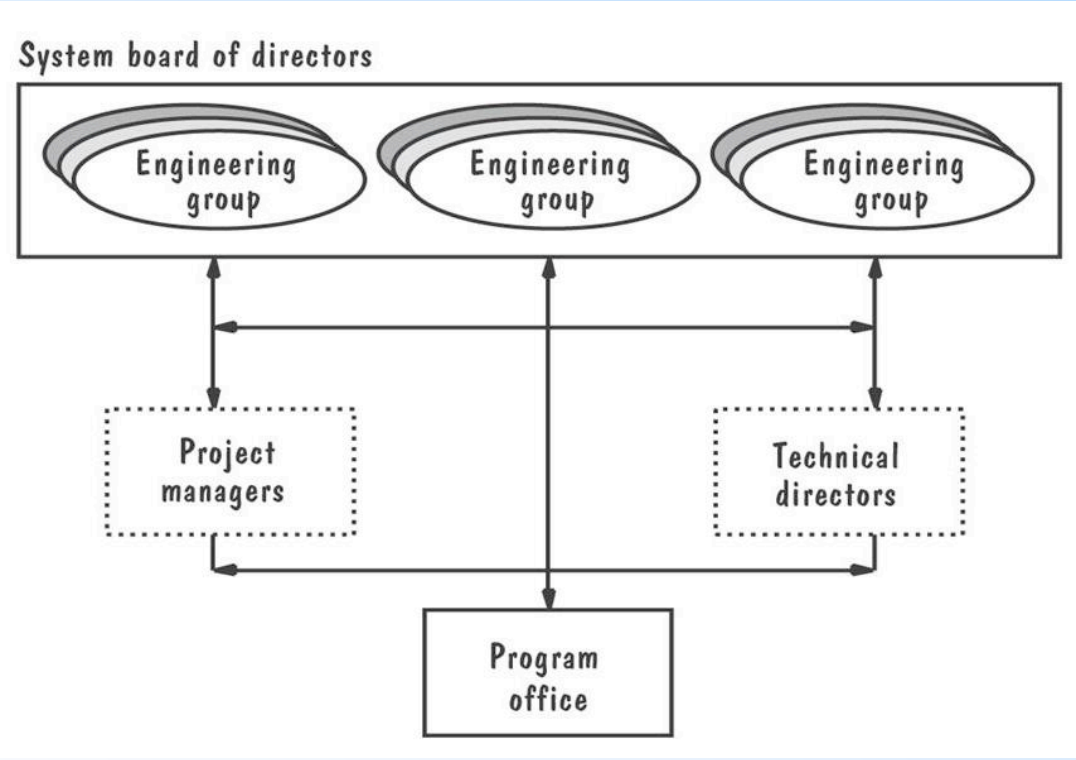
3.6 Process Models and Project Management Digital Alpha AXP (continued)

- Vision: to "enroll" the related programs, so they all shared common goals



3.6 Process Models and Project Management Digital Alpha AXP (continued)

- An organization that allowed technical focus and project focus to contribute to the overall program

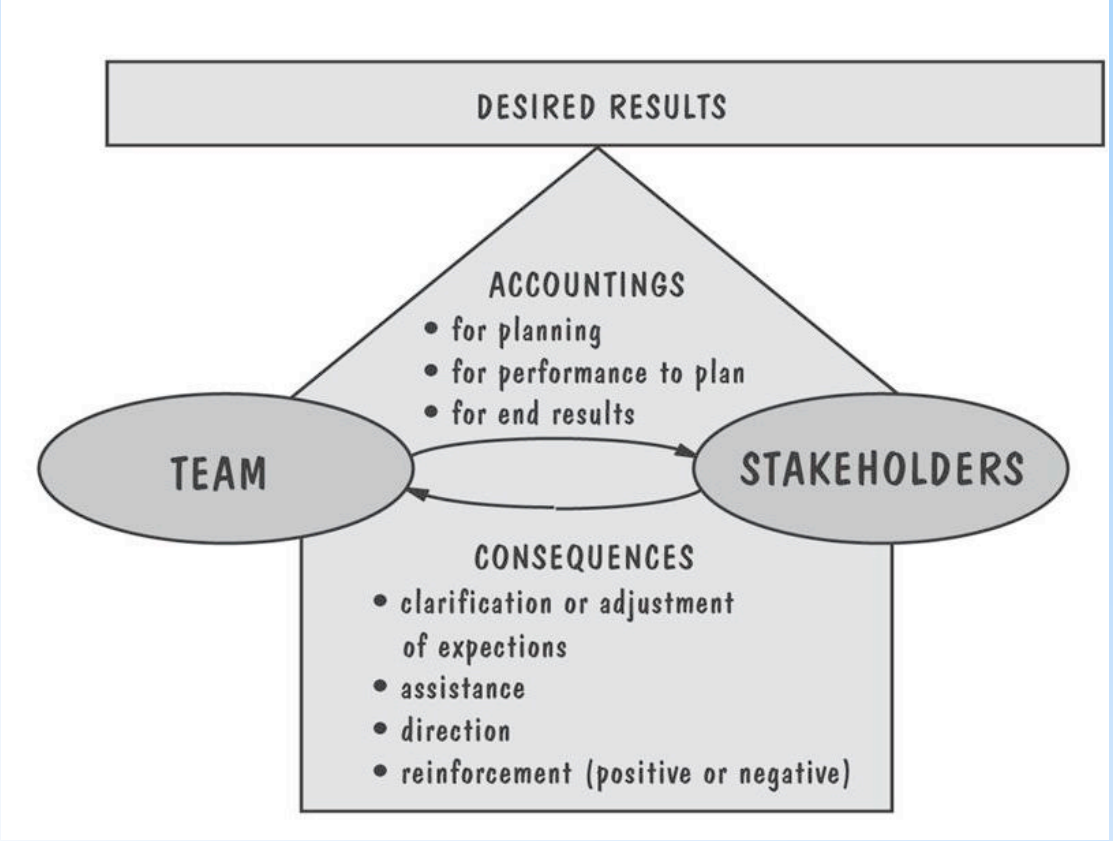


3.6 Process Models and Project Management
Accountability Modeling: Lockheed Martin

- Lockheed Martin's F16 project
  - produced 4M lines of code, 25% had real-time demands
  - 250 developers, 8 product teams, chief engineer, program man.
- Employees used to working in matrix organization
  - Each engineer belongs to a functional unit based on type of skill
- Project required integrated product development team
  - Combines people from different functional units into one interdisciplinary team
- Each activity tracked using cost estimation, critical path analysis, schedule tracking
  - Earned value a common measure for progress

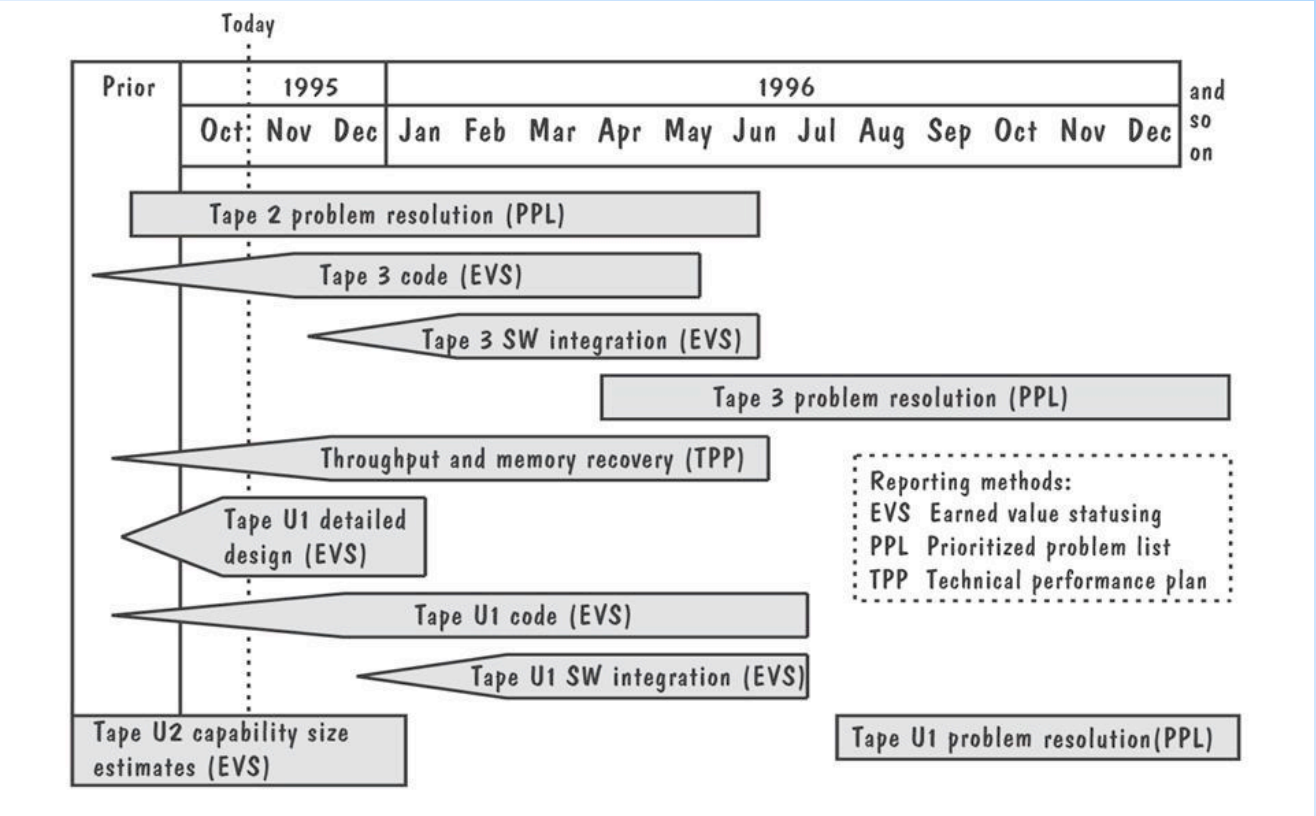
3.6 Process Models and Project Management
Accountability modeling:Lockheed Martin (continued)

- Accountability model used in F-16 Project
- Software written to track handoffs over time
  - Allowed coordination to be monitored by management



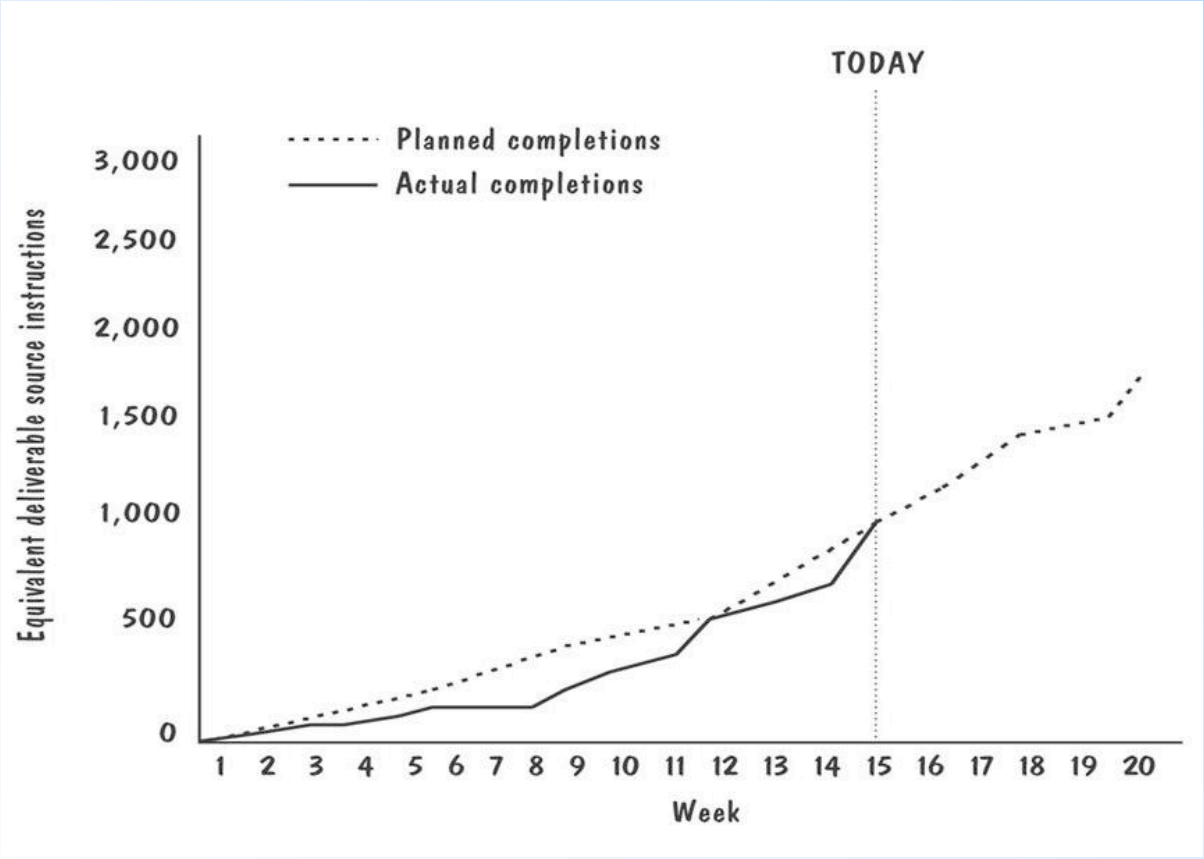
3.6 Process Models and Project Management
Accountability Modeling: Lockheed Martin (continued)

- Teams had multiple, overlapping activities
- An activity map used to illustrate progress on each activity



3.6 Process Models and Project Management
Accountability Modeling: Lockheed Martin (continued)

- Each activitiy's progress was tracked using earned value chart

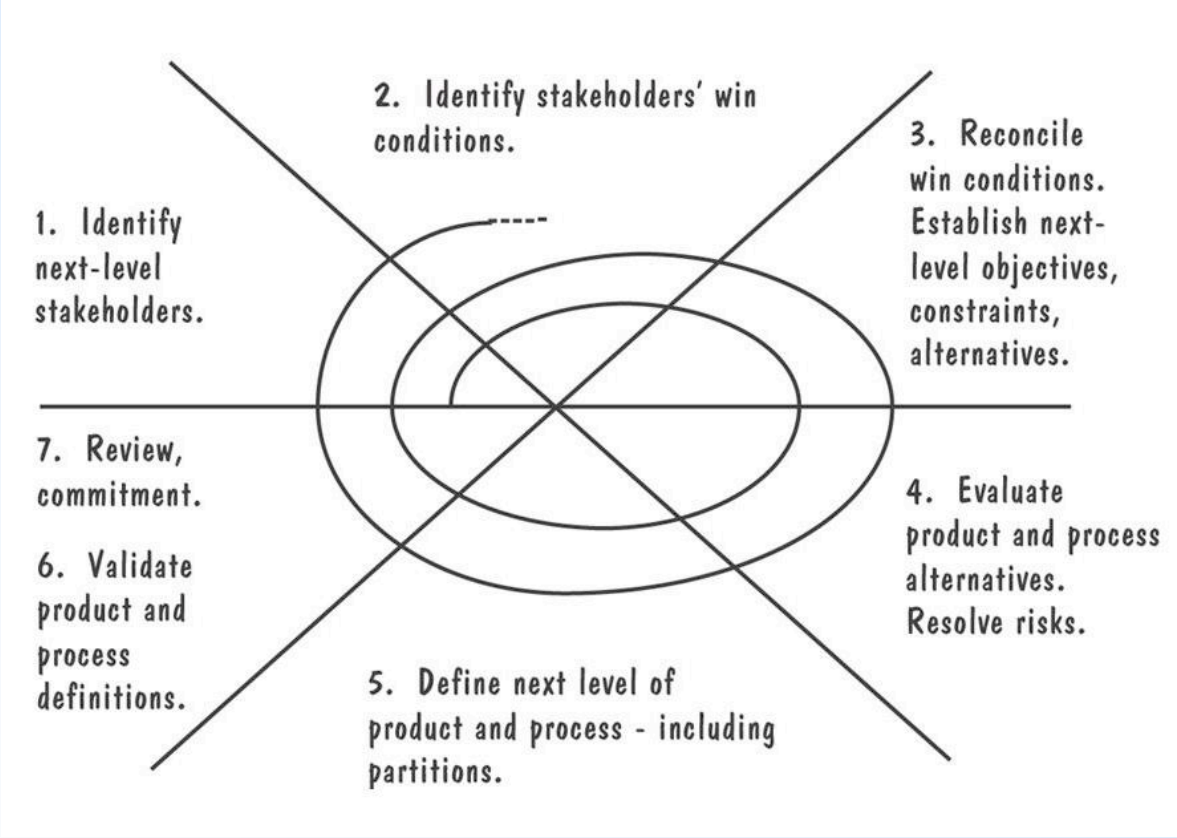


3.6 Process Models and Project Management Anchoring (Common) Milestones

- Life cycle objectives
  - **Objectives**: Why is the system being developed?
  - **Milestones and schedules**: What will be done by when?
  - **Responsibilities**: Who is responsible for a feature?
  - **Approach**: How will the job be done, technically and managerially?
  - **Resources**: How much of each resource is needed?
  - **Feasibility**: Can this be done, and is there a good business reason for doing it?
- Life-cycle architecture: define the system and software architectures and address architectural choices and risks
- Initial operational capability: readiness of software, deployment site, user training

3.6 Process Models and Project Management Anchoring Milestones (continued)

- The Win-Win spiral model suggested by Boehm is used as supplement to the milestones



3.9 What this Chapter Means for You

- Key concepts in project management
  - Project planning
  - Cost and schedule estimation
  - Risk management
  - Team Organization
- Project planning involves input from all team members
- Communication path grows as the size of the team increases and need to be taken into account when planning and estimating schedule