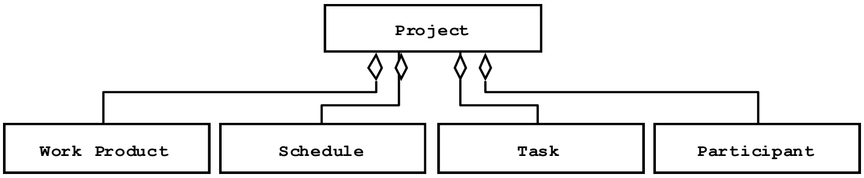
Project Management

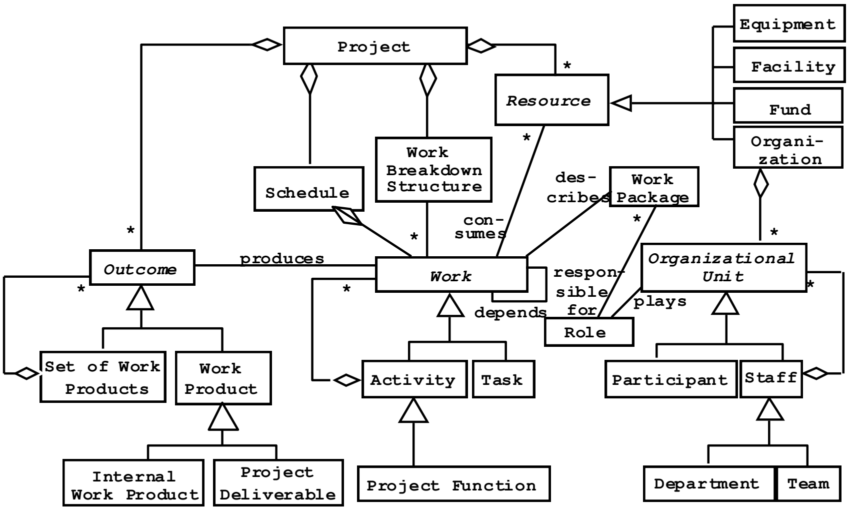
Basic Definitions: Project and Project Plan

- Software Project:
  - All *technical* and *managerial* activities required to deliver the deliverables to the client.
  - A software project has a specific duration, consumes resources and produces *work products*.
  - Management categories to complete a software project:
    - Tasks, Activities, Functions
- Software Project Management Plan:
  - The controlling document for a software project.
  - Specifies the technical and managerial approaches to develop the software product.
  - Companion document to requirements analysis document:
    - Changes in either document may imply changes in the other document.
  - The SPMP *may* be part of the project agreement.

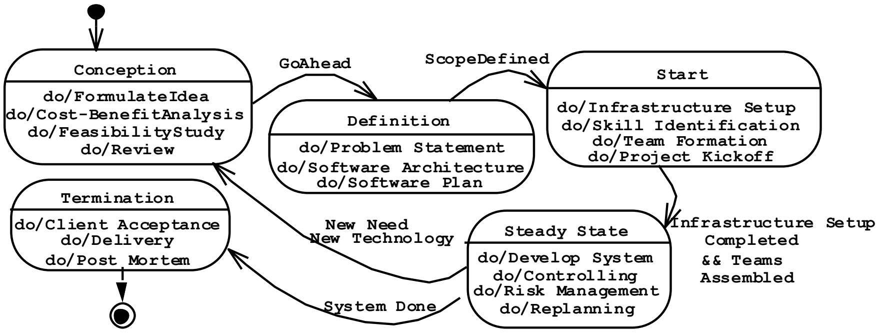
Components of a Project



A More Complex Model



States of a Project



Conception
do/FormulateIdea
do/Cost-BenefitAnalysis
do/FeasibilityStudy
do/Review

GoAhead

Definition
do/Problem Statement
do/Software Architecture
do/Software Plan

ScopeDefined

Start
do/Infrastructure Setup
do/Skill Identification
do/Team Formation
do/Project Kickoff

Infrastructure Setup
Completed
&& Teams
Assembled

Steady State
do/Develop System
do/Controlling
do/Risk Management
do/Replanning

New Need
New Technology

System Done

Termination
do/Client Acceptance
do/Delivery
do/Post Mortem

Capability Maturity Model

- **Model produced by the Software Engineering Institute to rate an organization’s software development process**

- Level 1: Initial - Lowest level, chaotic
- Level 2: Repeatable – Project tracking of costs, schedule, and functionality. Able to repeat earlier successes.
- Level 3: Defined – A documented and standardized software process. All development accomplished using the standard processes.
- Level 4: Managed – Quantitatively manages the process and products.
- Level 5: Optimizing – Uses the quantitative information to continuously improve and manage the software process.

Personal Software Process

- Can use the CMM idea and apply it to an individual software developer. Watts Humphrey developed PSP in 1997.
  - Use personal time logs to measure productivity; errors timed and recorded

Date	Start	Stop	Delta	Interrupt	Task
1/1	09:00	15:30	360	30 lunch	50 LOC
1/3	09:00	14:00	270	30 lunch	60 LOC
1/4	09:00	11:30	150		50 LOC
1/5	12:00	02:00	120		Testing

900 minutes to write/test a program of 160 LOC. Assuming 5 hrs/day this is 3 days to write/test 160 LOC. Productivity = 53 LOC/day

Earned Value Analysis

- Basic measures to calculate how much has been accomplished
  - Percent of the estimated time that has been completed
- Basic Measures
  - Budgeted Cost of Work (BCW)
    - The estimated effort for each work task
  - Budgeted Cost of Work Scheduled (BCWS)
    - The sum of the estimated effort for each work task that was scheduled to be completed by the specified time
  - Budget at Completion (BAC)
    - The total of the BCWS and thus the estimate of the total effort of the project

Earned Value Analysis

- Basic Measures
  - Planned Value (PV)
    - PV = BCW/BAC
    - The percentage of the total estimated effort assigned to a particular work task
  - Budgeted Cost of Work Performed (BCWP)
    - The sum of the estimated efforts for the work tasks completed by the specified time
  - Actual Cost of Work Performed (ACWP)
    - Sum of the actual efforts for the work tasks that have been computed

Earned Value Analysis

- Progress Indicators
  - Earned Value (EV) or Percent Complete (PC)
    - EV = BCWP/BAC
    - The sum of the Planned Value for all completed work tasks
  - Schedule Performance Index (SPI)
    - SPI = BCWP / BCWS
    - 100% = perfect schedule
  - Schedule Variance (SV)
    - SV = BCWP – BCWS
    - Negative is behind schedule, Positive ahead

Earned Value Analysis

- Progress Indicators
  - – Cost Performance Index (CPI)
    - CPI = BCWP / ACWP
    - 100% = perfect cost
  - – Cost Variance (CV)
    - CV = BCWP – ACWP
    - Negative is behind on cost, positive ahead on cost

Earned Value Analysis Example

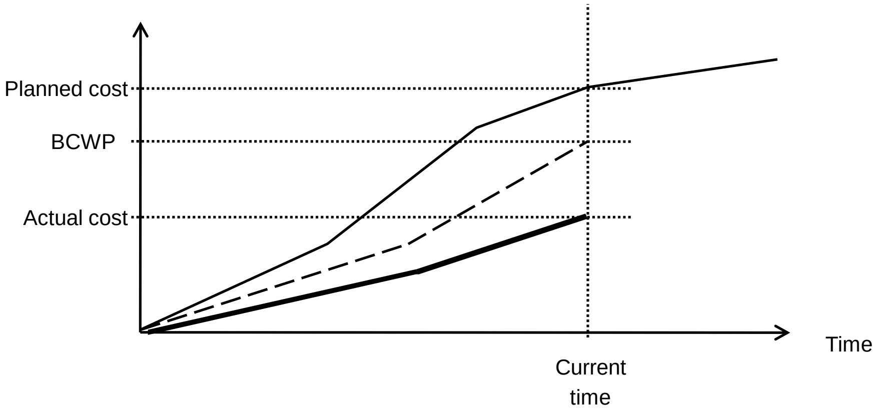
Task	Estimated Effort (days)	Actual Effort To Date	Estimated Completion	Actual Completion
1	5	10	1/25	2/1
2	25	20	2/15	2/15
3	120	80	5/15	
4	40	50	4/15	4/1
5	60	50	7/1	
6	80	70	9/1	

Today is 4/1

BAC = sum of estimations = 5 + 25 + 120 + … = 330 days
BCWP = estimate of completed work = 5 + 25 + 40 = 70 days
EV or PC = 70/330 = 21.2%
BCWS = sum of estimates scheduled to be done = 5+25 = 30
SPI = BCWP/BCWS = 70/30 = 233%
SV = 70 – 30 = 40 days   (ahead)
ACWP = sum of actual work done = 10+20+50 = 80
CPI = BCWP / ACWP = 70/80 = 87.5%
CV = BCWP – ACWP = 70-80 = -10 programmer days (behind)

Track Status Over Time

- Comparison of planned costs against actual costs allows the manager to assess the health of the project



Other Measurement Tools

- Error Tracking
  - – We generally expect error rates to go down over time
- Postmortem Reviews
  - – Assemble key people to discuss quality, schedule, software process. Results should not be sanitized.

Project Management Concepts

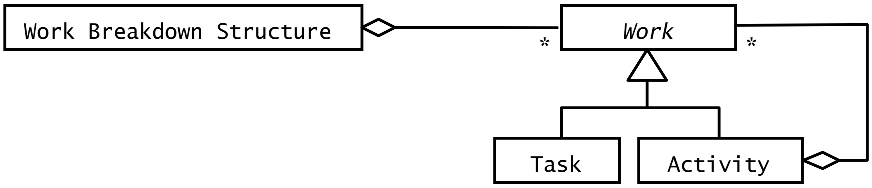
- Follow critical / best practices
- Divide and conquer approach generally taken to decompose work into smaller, more manageable pieces
- Key Tasks
  - Hierarchical representation of all the tasks in a project called the Work Breakdown Structure (WBS)
  - Task model or Network model
  - Mapping of the task model to the project schedule
  - Development of a Software Project Management Plan (SPMP)

Work Packages

- Work packages are assignment to participants to do the work
  - Small work package: an action item
  - Larger work packages:
    - Create the object model
    - Class diagram
    - Etc.
  - Any work product delivered to the customer is a deliverable; All other work products are internal work products

Work Breakdown Structure

- Simple hierarchical model of the work to be performed; uses aggregation only



Creating Work Breakdown Structures

- Two major philosophies
  - Activity-oriented decomposition ("Functional decomposition")
    - Write the book
    - Get it reviewed
    - Do the suggested changes
    - Get it published
  - Result-oriented ("Object-oriented decomposition")
    - Chapter 1
    - Chapter 2
    - Chapter 3
- Which one is best for managing? Depends on project type:
  - Development of a prototype
  - Development of a product
  - Project team consist of many unexperienced beginners
  - Project team has many experienced developers

Estimates for establishing WBS

- Establishing a WBS in terms of percentage of total effort:
  - Small project (7 person-month): at least 7% or 0.5 PM
  - Medium project (300 person-month): at least 1% or 3 PMs
  - Large project (7000 person-month): at least 0.2 % or 15 PMs
  - (From Barry Boehm, Software Economics)

Example: Let‘s Build a House

- What are the activities that are needed to build a house?

Typical activities when building a house

- Surveying
- Excavation
- Request Permits
- Buy Material
- Lay foundation
- Build Outside Wall
- Install Exterior Plumbing
- Install Exterior Electrical
- Install Interior Plumbing
- Install Interior Electrical
- Install Wallboard
- Paint Interior
- Install Interior Doors
- Install Floor
- Install Roof
- Install Exterior Doors
- Paint Exterior
- Install Exterior Siding
- Buy Pizza

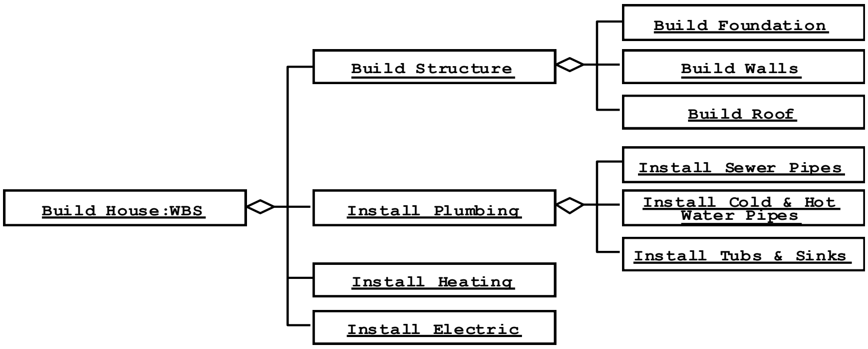
Finding these activities is a brainstorming activity.
It requires similar activities used during requirements analysis

Hierarchical organization of the activities

- Building the house consists of
  - Prepare the building site
  - Building the Exterior
  - Building the Interior

- Preparing the building site consists of
  - Surveying
  - Excavation
  - Buying of material
  - Laying of the foundation
  - Requesting permits

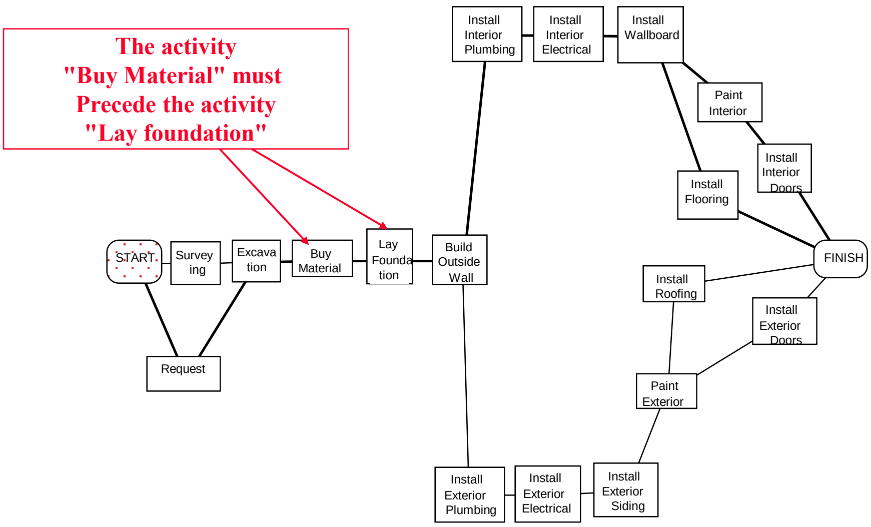
Partial Work Breakdown Structure



From the WBS to the Dependency Graph

- **The work breakdown structure does not show any temporal dependence among the activities/tasks**
  - **Can we excavate before getting the permit?**
  - **How much time does the whole project need if I know the individual times?**
    - **What can be done in parallel?**
  - **Are there any critical actitivites, that can slow down the project significantly?**
- **Temporal dependencies are shown in the dependency graph**
  - Nodes are activities
  - Lines represent temporal dependencies

Building a House (Dependency Graph)



The activity "Buy Material" must Precede the activity "Lay foundation"

Map tasks onto time

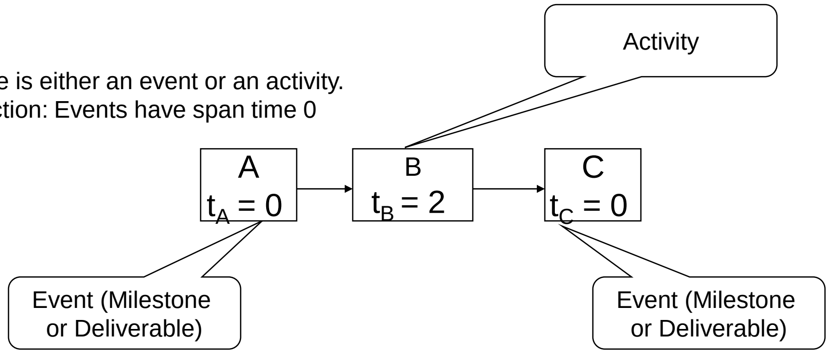
- **Estimate starting times and durations for each of the activities in the dependency graph**
- **Compute the longest path through the graph: This is the estimated duration of your project**

PERT

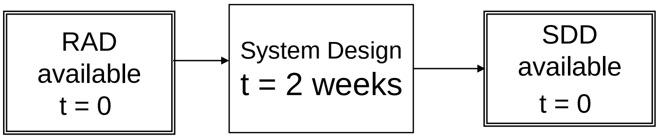
- PERT = Program Evaluation and Review Technique
- Developed in the 50s to plan the Polaris weapon system in the USA.
- PERT allows the manager to assign optimistic, pessimistic and most likely estimates for the span times of each activity.
- You can then compute the probability to determine the likelihood that overall project duration will fall within specified limits.

PERT Diagram Notation

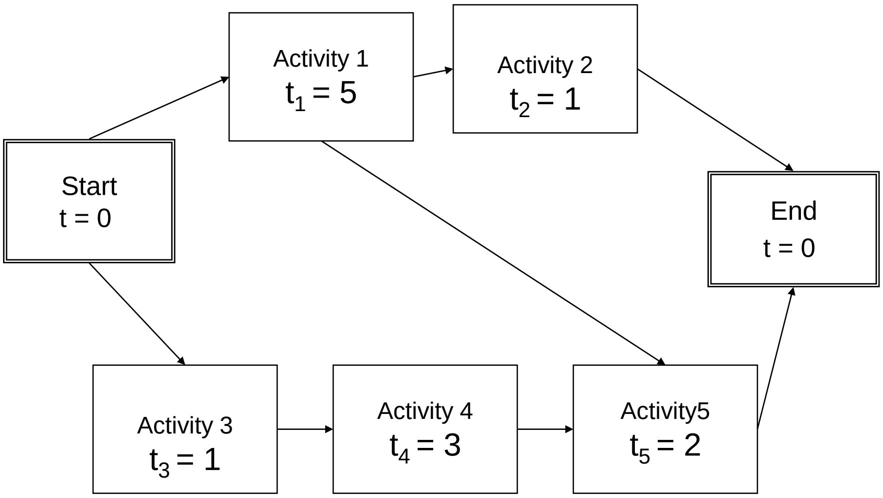
A Node is either an event or an activity.
Distinction: Events have span time 0


Activity
A
$t_A = 0$
B
$t_B = 2$
C
$t_C = 0$
Event (Milestone or Deliverable)
Event (Milestone or Deliverable)

Milestone boxes are often highlighted by double-lines



Example of a Node Diagram



What do we do with these diagrams?

- Compute the project duration
- Determine activities that are critical to ensure a timely delivery

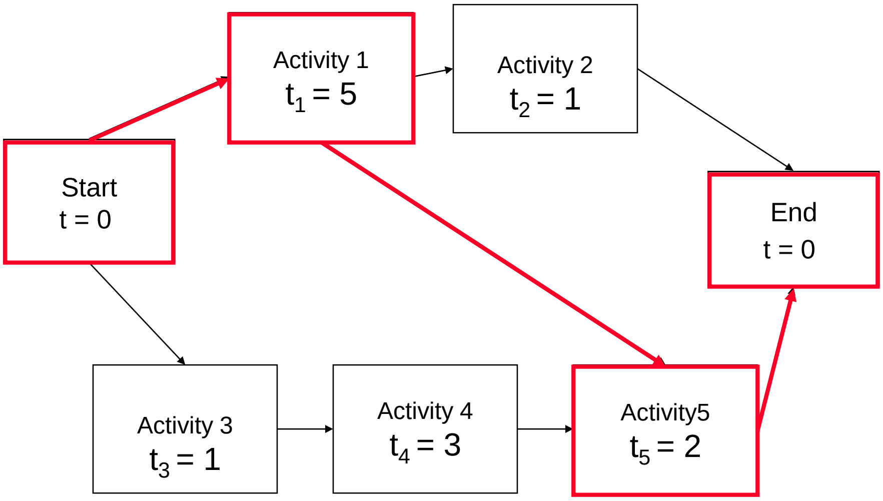
- Analyze the diagrams
  - to find ways to shorten the project duration
  - To find ways to do activities in parallel

- 2 techniques are used
  - Forward pass (determine critical paths)
  - Backward pass (determine slack time)

Definitions: Critical Path and Slack Time

- Critical path:
  - A sequence of activities that take the longest time to complete
  - The length of the critical path(s) defines how long your project will take to complete.
- Noncritical path:
  - A sequence of activities that you can delay and still finish the project in the shortest time possible.
- Slack time:
  - The maximum amount of time that you can delay an activity and still finish your project in the shortest time possible.

Example of a critical path



Critical path in bold face

Definitions: Start and Finish Dates

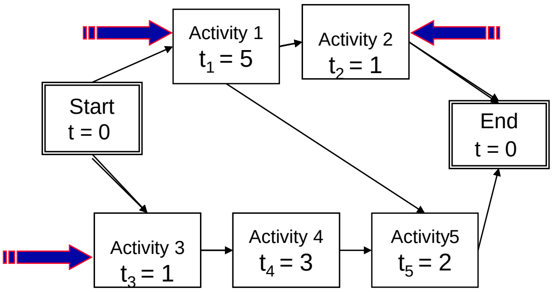
- Earliest start date:
  - The earliest date you can start an activity
- Earliest finish date:
  - The earliest date you can finish an activity
- Latest start date:
  - The latest date you can start an activity and still finish the project in the shortest time.
- Latest finish date:
  - The latest date you can finish an activity and still finish the project in the shortest time.

2 Ways to Analyze Dependency Diagrams

- Forward pass: Goal is the determination of critical paths
  - Compute earliest start and finish dates for each activity
  - Start at the beginning of the project and determine how fast you can complete the activites along each path until you reach the final project milestone.
- Backward pass: Goal the determination of slack times
  - Compute latest start and finish dates activity
  - Start at the end of your project, figure out for each activity how late it can be started so that you still finish the project at the earliest possible date.
- To compute start and finish times, we apply 2 rules
  - Rule 1: After a node is finished, we can proceed to the next node(s) that is reachable via a transition from the current node.
  - Rule 2: To start a node all nodes must be complete from which transitions to that node are possible.

Forward Path Example

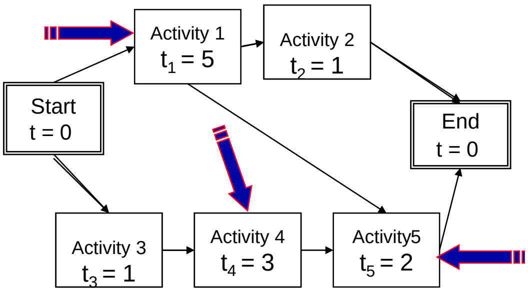
Project Duration = 7



Activity	Earliest Start(ES)	Earliest Finish(EF)
A1	Start of week 1	End of week 5
A2	Start of week 6	End of week 6
A3	Start of week 1	End of week 1
A4	Start of week 2	End of week 4
A5	Start of week 6	End of week 7

Backward Path Example

Project Duration = 7



Activity	Latest Start(LS)	Latest Finish(LF)
A1	Start of week 1	End of week 5
A2	Start of week 7	End of week 7
A3	Start of week 2	End of week 2
A4	Start of week 3	End of week 5
A5	Start of week 6	End of week 7

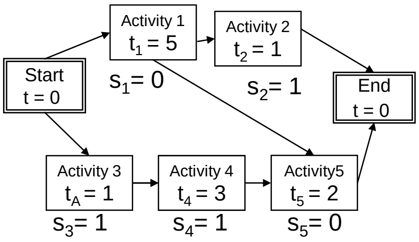
Computation of slack times

- Slack time ST of an activity A:
  - $ST_A = LS_A - ES_A$
  - Subtract the earliest start date from the latest start date for each activity

Example: $ST_{A4} = 3 - 2 = 1$

Slack times on the same path influence each other.
Example: When Activity 3 is delayed by one week, activity 4 slack time becomes zero weeks.

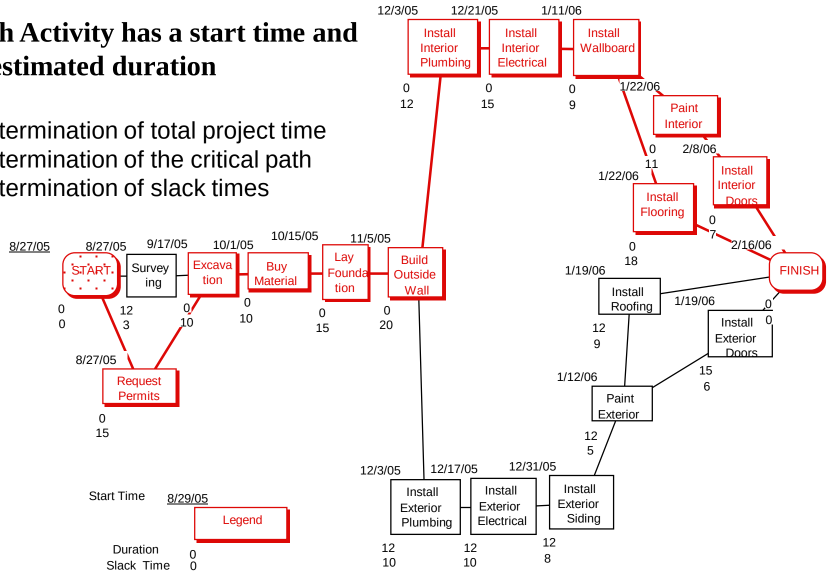
Activity	Slack time
A1	0
A2	1
A3	1
A4	1
A5	0



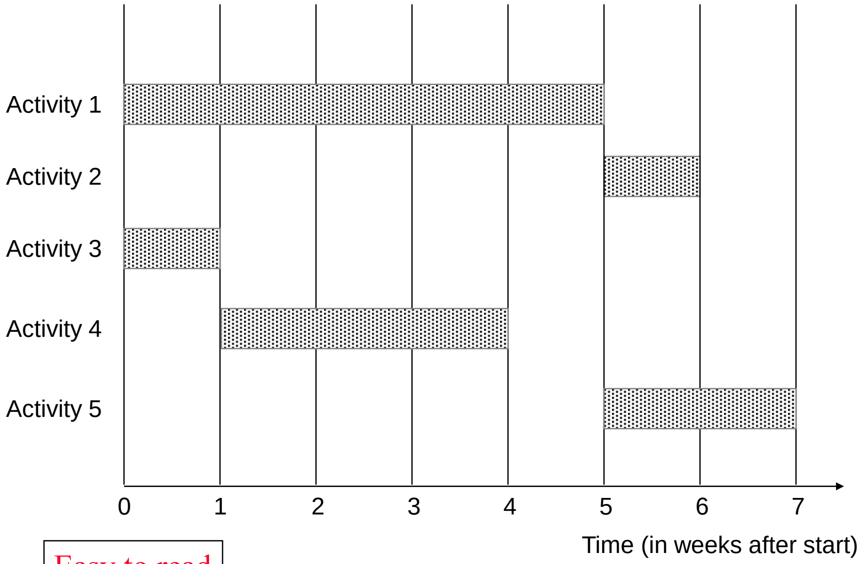
Building a House (PERT Chart)

Each Activity has a start time and an estimated duration

- Determination of total project time
- Determination of the critical path
- Determination of slack times

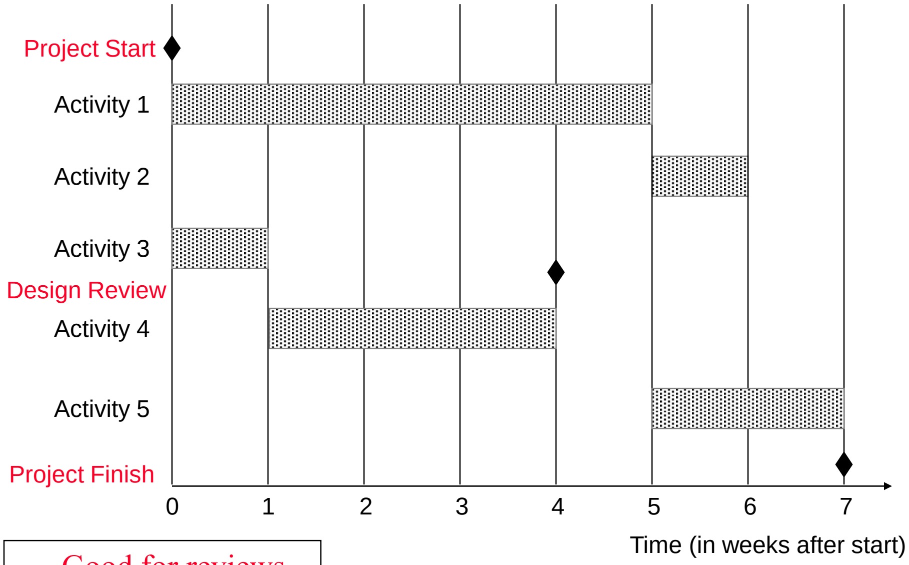


Gantt Chart



Easy to read

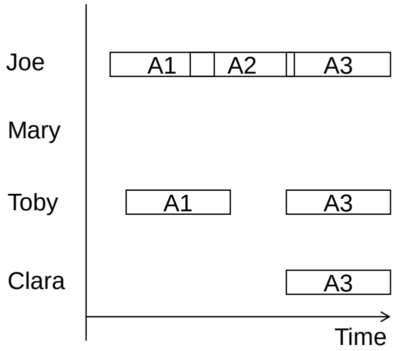
Gantt Chart with Milestones



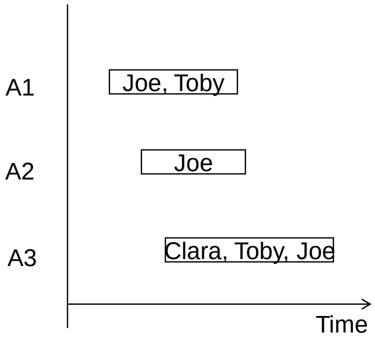
Good for reviews.

Two Types of Gantt Charts

- Person-Centered View
  - To determine people's load



- Activity-Centered View
  - To identify teams working together on the same tasks



Choose one view, stay with it. Usually base the view on the WBS structure

Managing Experienced Teams: Person-centered view

Managing Beginners: Activity oriented view

Heuristics for WBS

- The project manager may find the following heuristics useful to create the work breakdown structure
  - Reuse an existing WBS
    - Consult people who have worked on similar projects
  - Involve key developers
    - Developers with knowledge in the solution domain should participate in the development
    - If they join after the WBS is developed they should be able to review and critique it
  - Identify work gaps.
    - All work to be performed must be mapped onto tasks
    - Work associated with an activity must be addressed by at least one task
  - Identify work overlaps
    - The same task should not be included in more than one activity

Creating the Initial Schedule

- Impossible to generate a precise schedule for the entire project at the beginning of the project
- One solution: initial schedule with deadlines mutually agreed by the client and project manager
- Detailed for the first few weeks of the project
  - Kick-off meetings
  - Initial team meetings
  - Tutorials
  - Individual teams could start working on a revision of the initial schedule after the initial team meetings

Organizing the Project

- The project manager needs to address the communication infrastructure
  - Scheduled modes of communication
    - Planned milestones, review, team meetings, inspections, etc.
    - Best supported by face-to-face communications
  - Event-based modes of communication
    - Problem reports, change requests, etc.
    - Usually arise from unforeseen problems or issues
    - E-mail, groupware, web databases the best mechanisms

Identifying Skills

- Skills for a software development project
  - Application domain skills
  - Communication skills
  - Technical skills
  - Quality skills
  - Management skills
- Assign management, technical roles
- 3-5 team members the best size for a group

Kick-off Meeting

- Project manager, team leaders, and the client officially start the project in a kick-off meeting with all developers present
- Purpose: Share information about the scope of the project, communication infrastructure, and responsibilities of each team
- Presentation split between client and project manager
  - Client: Requirements and scope of the project
  - Project manager: Project infrastructure, top-level design, and team responsibilities

Project Agreement

- Document that formally defines the scope, duration, cost, and deliverables
  - Contract or statement of work, business plan, or charter
  - Typically finalized after the analysis model is stabilized
- Should contain
  - List of deliverables
  - Criteria for demonstrations of functional requirements
  - Criteria for demonstration of nonfunctional requirements
  - Criteria for acceptance
- Represents the baseline of the client acceptance test
- Changes in the functionality, deadlines, or budget requires renegotiation of the project agreement

Controlling the Project

- The project manager must collect information to make effective decisions in the steady state phase of the project
- Tools to collect information
  - Meetings
    - Periodic status meetings, milestones, project reviews, code inspections, prototype demonstrations
  - Metrics
    - Lines of code, branching points, modularity
    - Defects, mean time between failures

Software Cost Estimation

- How many resources to complete the project?
  - For big projects, expressed in Programmer Months
  - Older approach: LOC estimation
  - Newer approach: Counting Function Points

LOC Estimation

- Estimate number of lines of code in the finished project
  - Use prior experience, similar products, etc.
- Standard approach:
  - For each piece i, estimate the max size, min size, and best guess. The estimate for the each piece is 1/6*(max + 4*guess + min)

Part	Min	Guess	Max
1	20	30	50
2	10	15	25
3	25	30	45

Whole = (20+4*30+50)/6 +
(10+4*15+25)/6 +
(25+4*30+45)/6
= 79 LOC

COCOMO

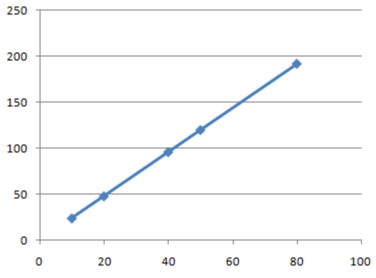
- COCOMO = Constructive Cost Model, developed by Boehm in the 70’s
  - Used thousands of delivered lines of code to determine a relationship between size and cost in Programmer Months (PM)
  - App Programs: PM = 2.4*(KLOC)$^{1.05}$
  - Utility Programs: PM = 3.0*(KLOC)$^{1.12}$
  - Systems Programs: PM = 3.6*(KLOC)$^{1.20}$

General LOC Estimation

In general: Cost = A * KLOC^{B} + C where A,B,C are constants

Can determine these values regressively if you measure your own efforts:

Project	KLOC	Effort (PM)
1	50	120
2	80	192
3	40	96
4	10	24
5	20	48



Function Point Analysis

- Identify and quantify the functionality required for the project. Some possibilities, but no standards for what is considered a function point:
  - Inputs
    - Logical input, not individual fields
  - Outputs
    - Displays of application dtaa
  - Inquiries
    - Request/response pairs
  - Internal files
    - Number of logical files
  - External interfaces
    - Data shared with other programs

Function Point Analysis

- Individual function points classified as simple, average, or complex, and weights are summed

	Simple	**Average**	**Complex**
Outputs	4	5	7
Inquiries	3	4	6
Inputs	3	4	6
Files	7	10	15
Interfaces	5	7	10

- Correlate total with PM; can capture effort for hidden items (e.g. one output, lots of internal work)

Conclusion

- Software Project Managers have a lot of challenging work that shouldn’t be ignored
  - Unlike the Pointy Haired Boss
  - Must deal with project outcomes, schedules, work products, work breakdown schedule, and resources
  - Development of a Software Project Management Plan
  - Much of this built into the Agile Development process in a simple way
- Project managers can deal with project complexity the same way developers deal with system complexity
  - Modeling of the domain
  - Communication
  - Analysis
  - Planning