Project Scheduling

Lecture 16

Why Dependency Diagrams?

Dependency Diagrams are a formal notation to help in the construction and analysis of complex schedules

Example:

A project consists of 5 tasks; each of these takes one week to complete.

How long does the project take?

Example:

A project consists of 5 tasks. Task 1 has to be finished before any other tasks can start. Task 2 and task 3 can be done in parallel, task 4 and 5 cannot. Task 4 and 5 both depend on task 2.

Dependency Diagrams (Overview)

Dependency diagrams consist of three elements

- Event: A significant occurrence in the life of a project.
- Activity: Amount of work required to move from one event to the next.
- Span time: The actual calendar time required to complete an activity.
  - Span time parameters: availability of resources, parallelizability of the activity

Dependency diagrams are drawn as a connected graph of nodes and arrows. Two commonly used notations to display dependency diagrams are:

- Activity-on-the-arrow
- Activity-in-the-node

PERT

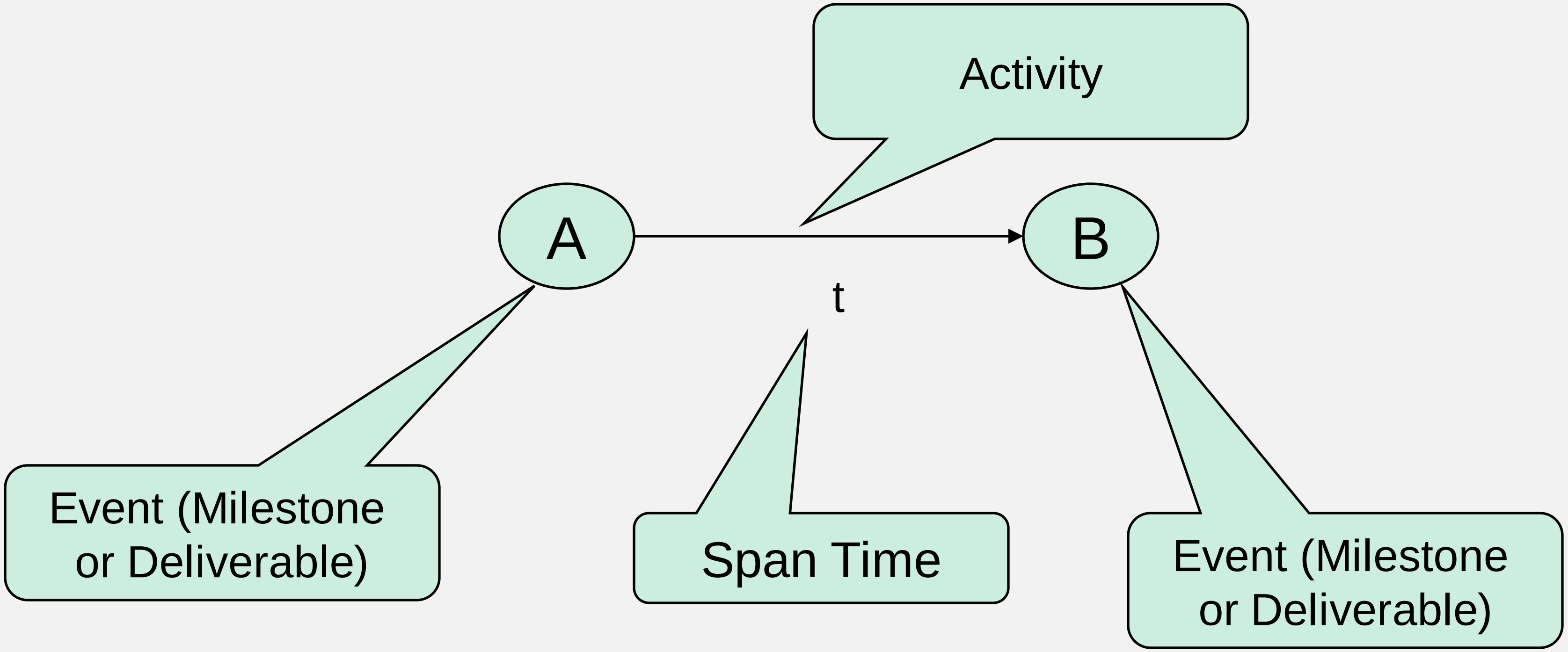
PERT stands for “Program Evaluation and Review Technique”

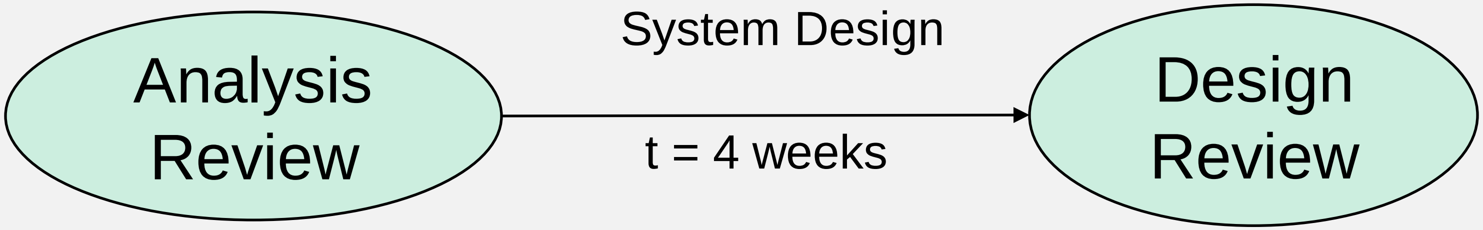
PERT uses an *activity-on-the-arrow* notation

Algorithm:

1. Assign optimistic, pessimistic and most likely estimates for the span times of each activity.
2. Compute the probability that the project duration will fall within specified limits.
3. At first the method did not take cost into consideration but was later extended to cover cost as well.
4. Still more fitting for projects where duration matters more than cost.

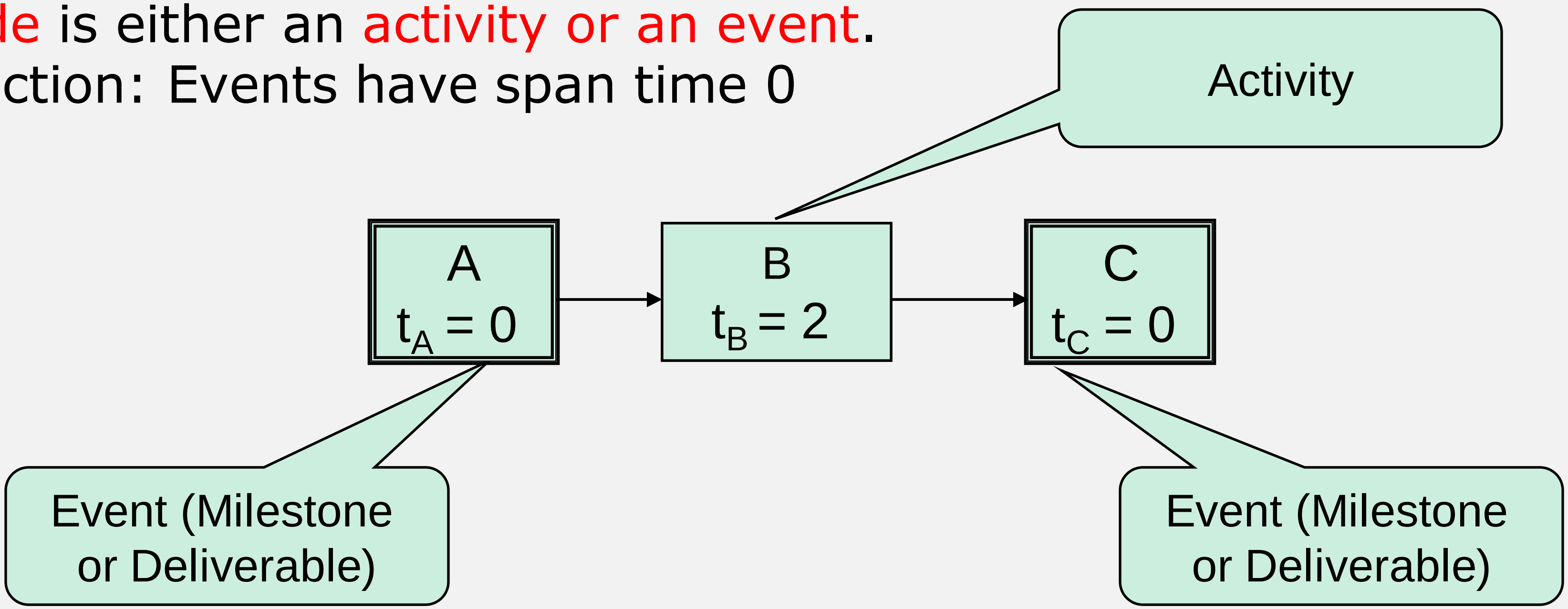
1) Activity-on-the-arrow Diagram Notation


Activity
A
B
t
Event (Milestone or Deliverable)
Span Time
Event (Milestone or Deliverable)



2) Activity-in-the-node Diagram Notation

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



Activity

A
$t_A = 0$

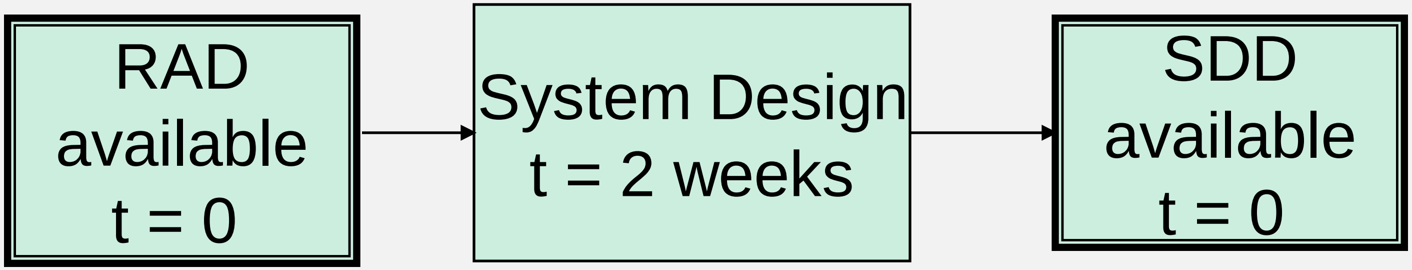
B
$t_B = 2$

C
$t_C = 0$

Event (Milestone or Deliverable)

Event (Milestone or Deliverable)

Milestone boxes are often highlighted by double-lines

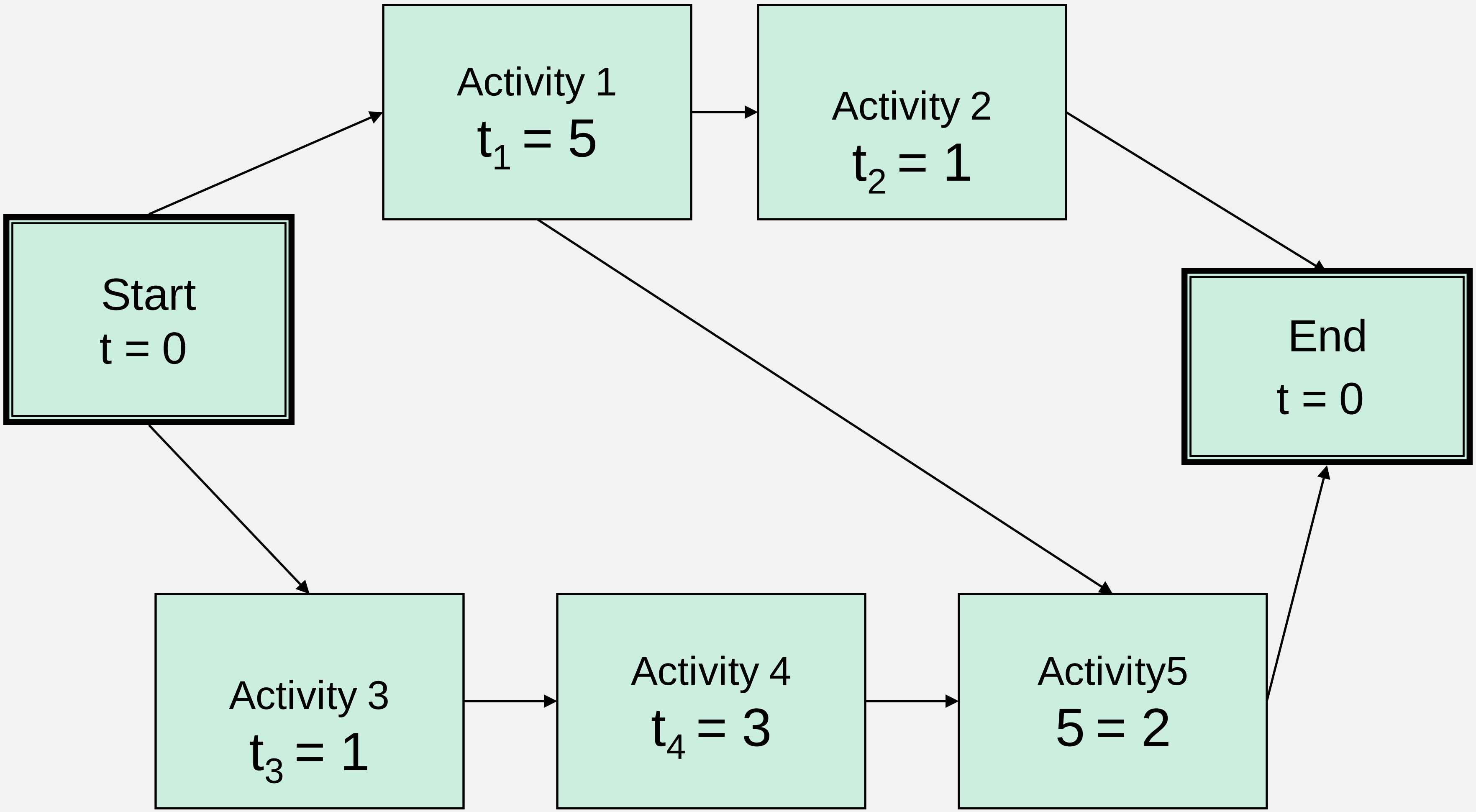


RAD available
t = 0

System Design
t = 2 weeks

SDD available
t = 0

Example of an Activity-in -the -Node Diagram



Start

t = 0

Activity 1

$t_1 = 5$

Activity 2

$t_2 = 1$

End

t = 0

Activity 3

$t_3 = 1$

Activity 4

$t_4 = 3$

Activity5

5 = 2

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.

What do we do with these diagrams?

1. Compute the project duration
2. Determine activities that are critical to ensure a timely delivery

Analyze the diagrams

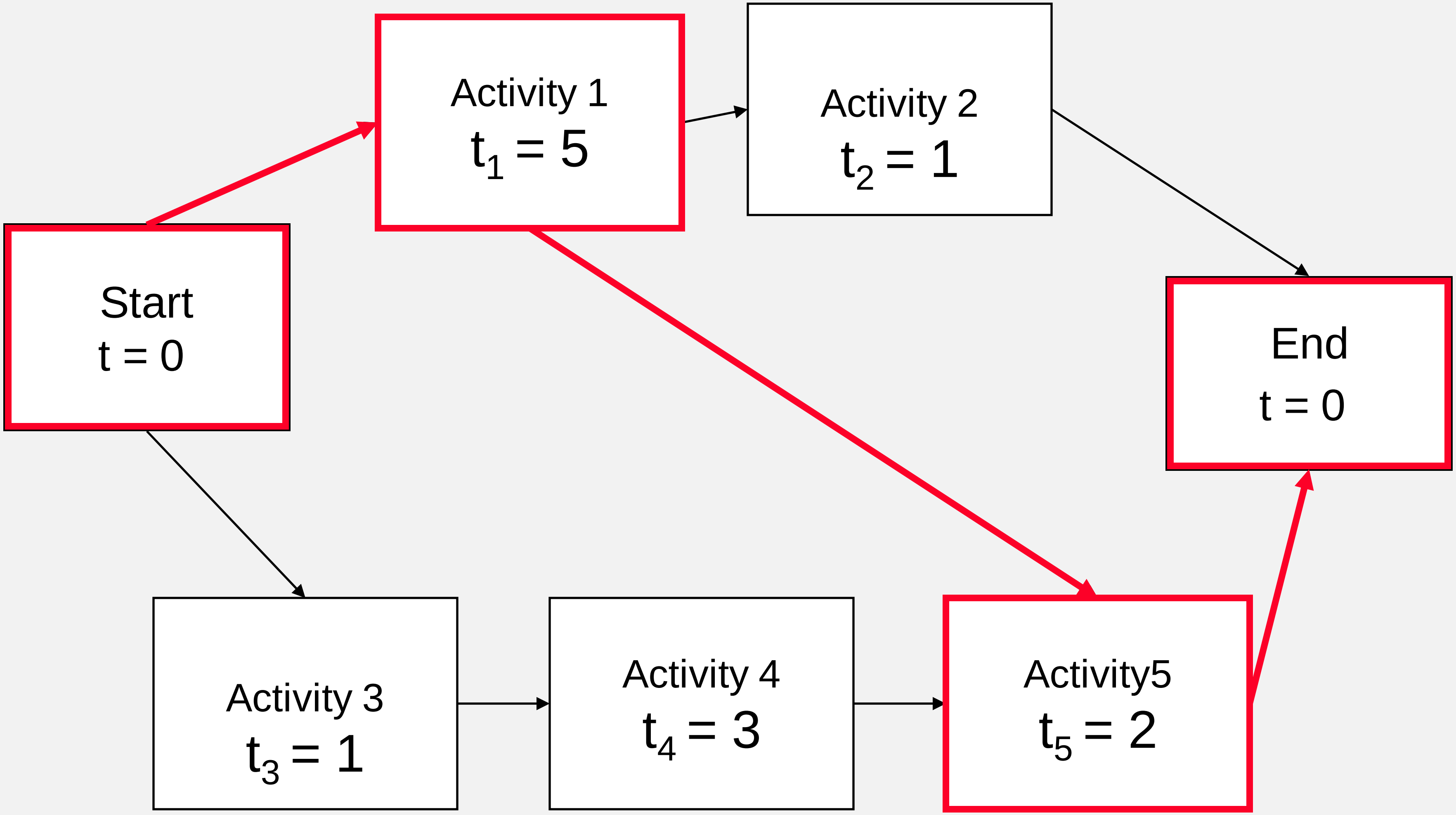
- To find ways to shorten the project duration
- To find ways to do activities in parallel

2 techniques are used

Forward pass (determine critical path)

Backward pass (determine slack time)

Example of a critical path



Activity 1

$t_1 = 5$

Activity 2

$t_2 = 1$

Start

t = 0

End

t = 0

Activity 3

$t_3 = 1$

Activity 4

$t_4 = 3$

Activity5

$t_5 = 2$

Critical path with bold and red arrows

Analyzing Dependency Graphs

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 path)

Backward pass (determine slack time)

Analyzing Dependency Graphs

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 activities along each path until you reach the final project milestone.

Also called *forward path analysis*

Determination of slack times

- 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.

Also called *back path analysis*

Definitions: Start and Finish Dates

Earliest start date (ES):

The earliest date you can start an activity

Earliest finish date (EF):

The earliest date you can finish an activity

Latest start date (LS):

The latest date you can start an activity and still finish the project in the shortest time

Latest finish date (LF):

The latest date you can finish an activity and still finish the project in the shortest time.

Computing Start and Finish Times

To compute start and finish times, we apply two rules

Rule 1: After a node is finished, we can proceed to the next node(s) that is (are) reachable via a transition from the current node.

Rule 2: To start a node, all nodes from which transitions to that node are possible must be complete.

Summary: Analyzing Dependency Diagrams

Forward pass: Goal is the determination of critical paths

- Compute earliest start and finish dates for each activity

Backward pass: Goal is the determination of slack times

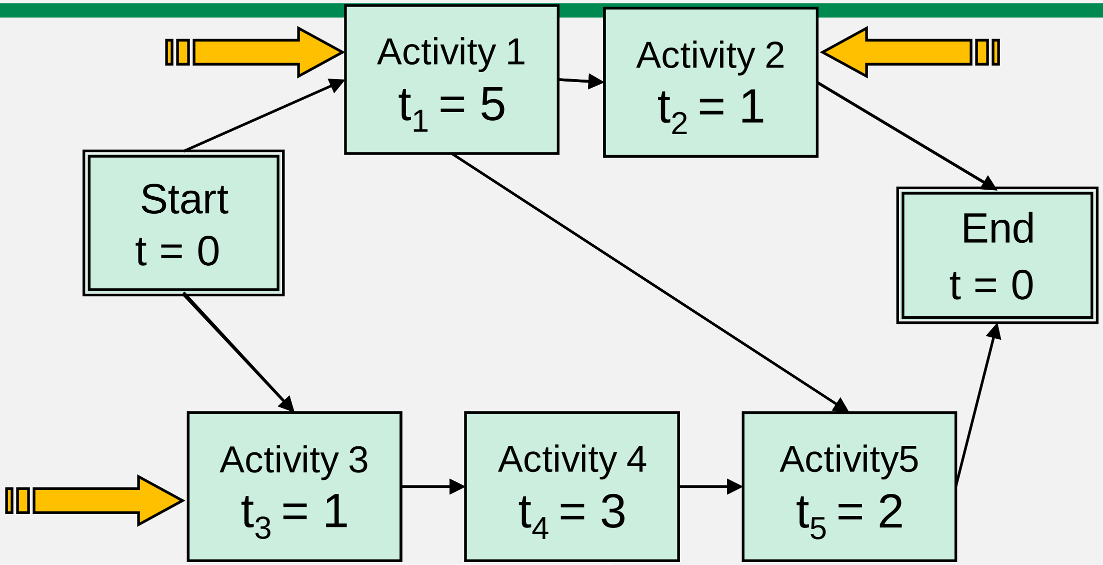
- Compute latest start and finish dates for each activity

Rules for computing start and finish times

- Rule 1: After a node is finished, proceed to the next node that is reachable via a transition from the current node.
- Rule 2: To start a node all nodes from which transitions to that node are possible must be complete.

Forward Path Analysis

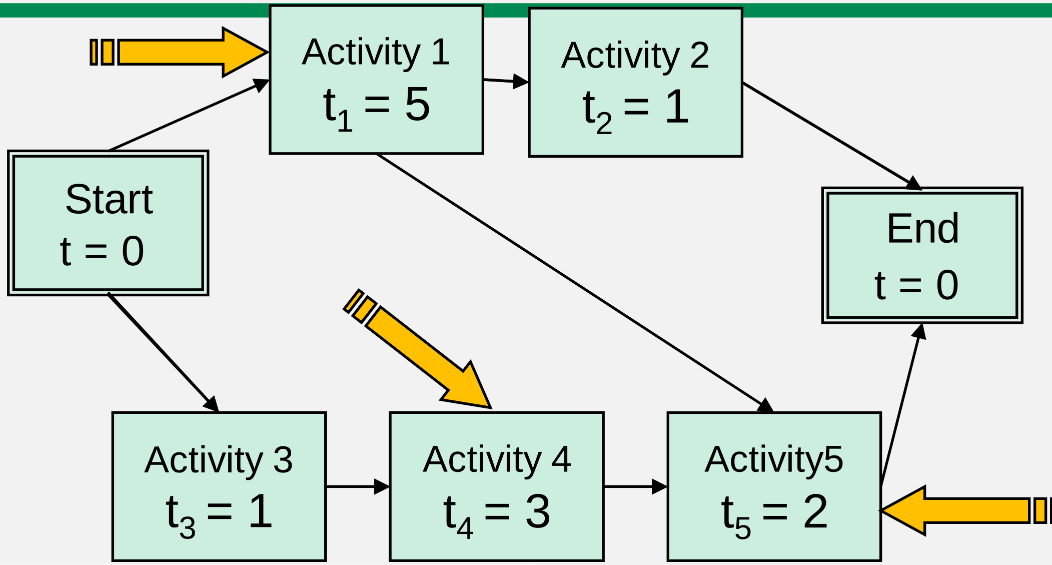
Project Duration = 7



Activity	ES (earliest start)	EF (earliest finish)
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 Analysis

Project Duration = 7



Activity	LS (Latest start)	LF (Latest finish)
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:

$ST_A = LS_A - ES_A$

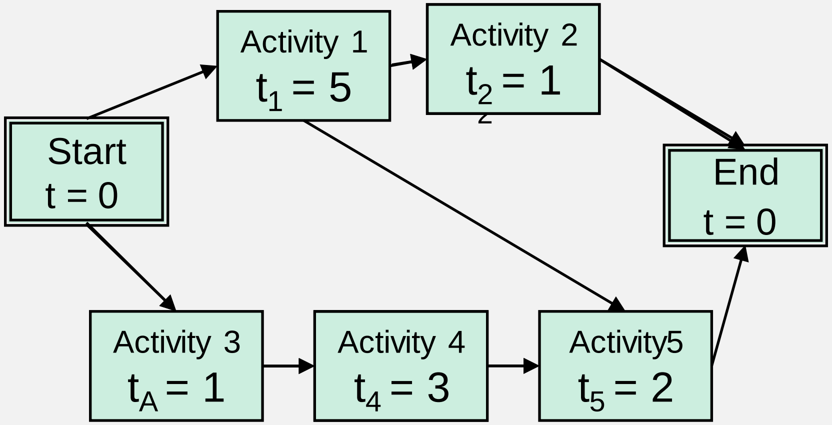
Example: $ST_{A4} = \ ?$

$ST_{A4} = \ 3 - 2 = 1$

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



Path types in dependency graphs

- Critical path: Any path in a dependency diagram, in which all activities have zero slack time.
- Noncritical path: Any path with at least one activity that has a nonzero slack time.
- Overcritical path: A path with at least one activity that has a negative slack time.
    - Overcritical paths should be considered as serious warnings: Your plan contains unrealistic time estimates

Path types in dependency graphs cont.

- Any dependency diagram with no fixed intermediate milestones has at least one critical path.
- A project schedule with fixed intermediate milestones might not have a critical path

  Example:

  - The analysis review must be done 1 month after project start
  - The estimated time for all activities before the review is less than 4 weeks.

Frequently used formats for schedules

Milestone View:

A table that lists milestones and the dates on which you plan to reach them.

Activities View:

A table that lists the activities and the dates on which you plan to start and end them

Gantt chart View:

A graphical view illustrating on a timeline when each activity will start, be performed and end.

Combined Gantt Chart and Milestone View:

The Gantt Chart contains activities as well as milestones.

PERT Chart View:

A graphical representation of task dependencies and times.

Burndown Chart View:

A graph showing the number of open tasks over time.

Milestone View (Key-Events Report)

Date	Milestones
August 26	Project Kickoff (with Client)
October 16	Analysis Review
October 26	System Design Review
November 7	Internal Object Design Review
November 20	Project Review (with Client)
November 26	Internal Project Review
December 11	Acceptance Test (with Client)

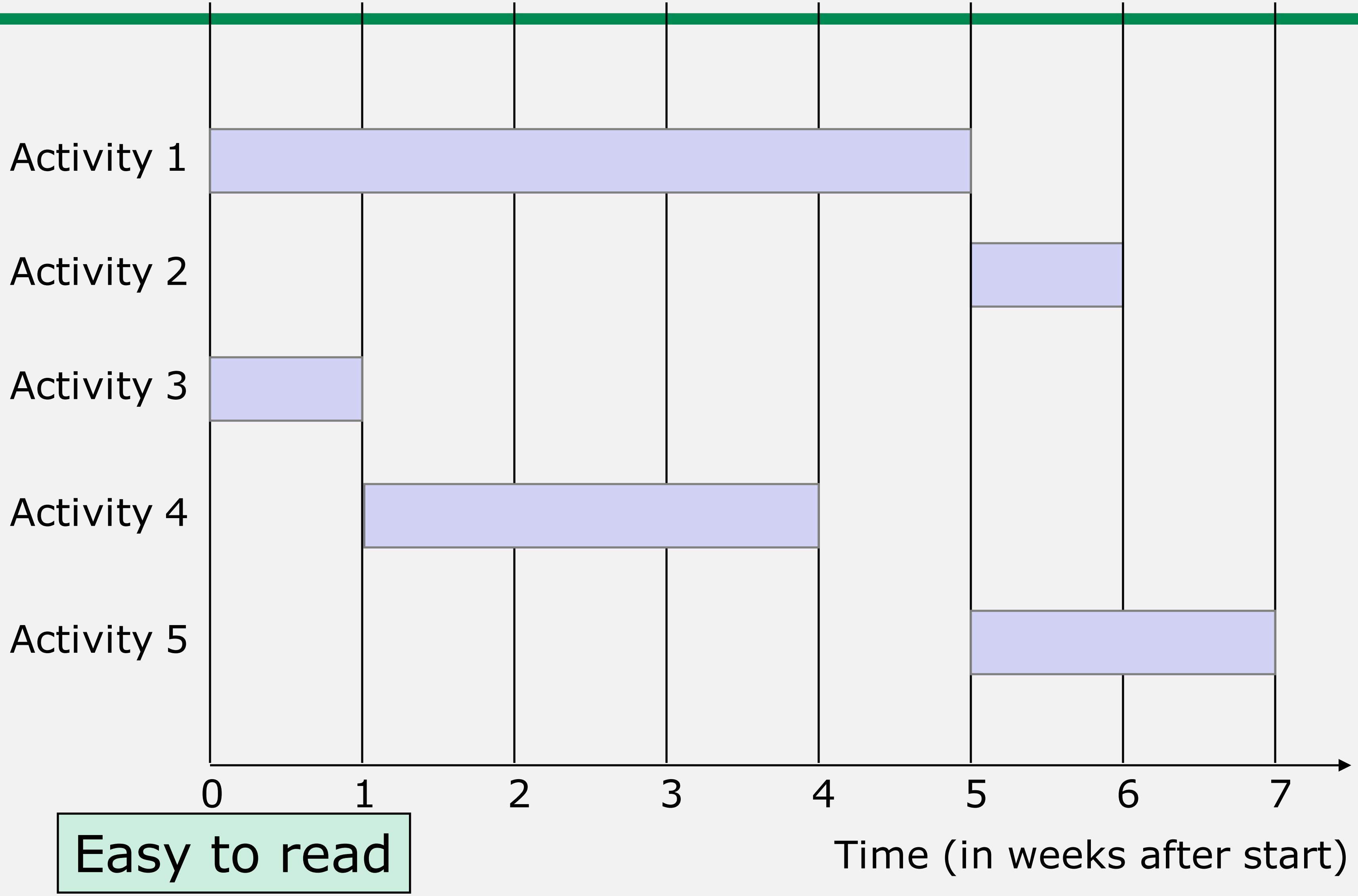
Good for introduction of project
and high executive briefings

Activities View

Date	**Project Phase**
Jul 17 - Aug 23	Preplanning Phase
Aug 26 - Sep 24	Project Planning
Sep 11 - Oct 8	Requirements Analysis
Oct 9 - Oct 26	System Design
Oct 28 - Nov 7	Object Design
Nov 8 - Nov 20	Implementation & Unit Testing
Nov 22 - Dec 4	System Integration Testing
Dec 4 - Dec 10	System Testing
Dec 11- Dec 18	Post-Mortem Phase

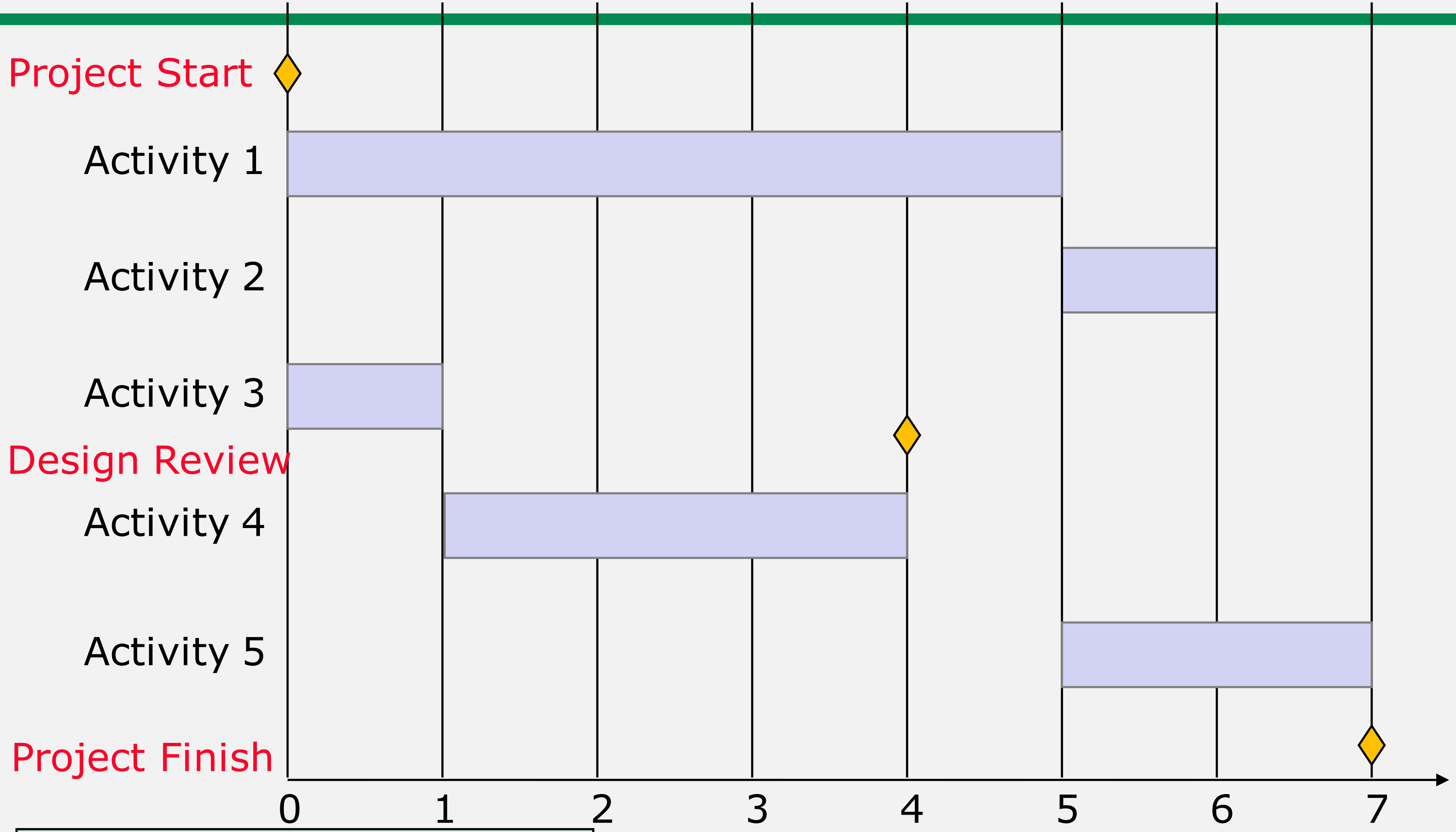
Good for documentation and during developer meetings

Gantt Chart



Easy to read

Gantt Chart with milestones



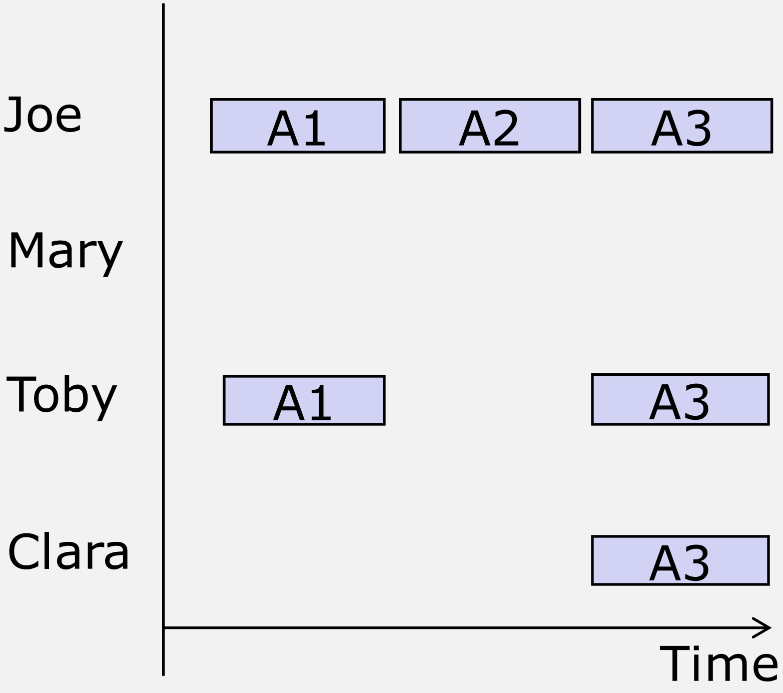
Good for reviews

Time (in weeks after start)

Two Types of Gantt Charts

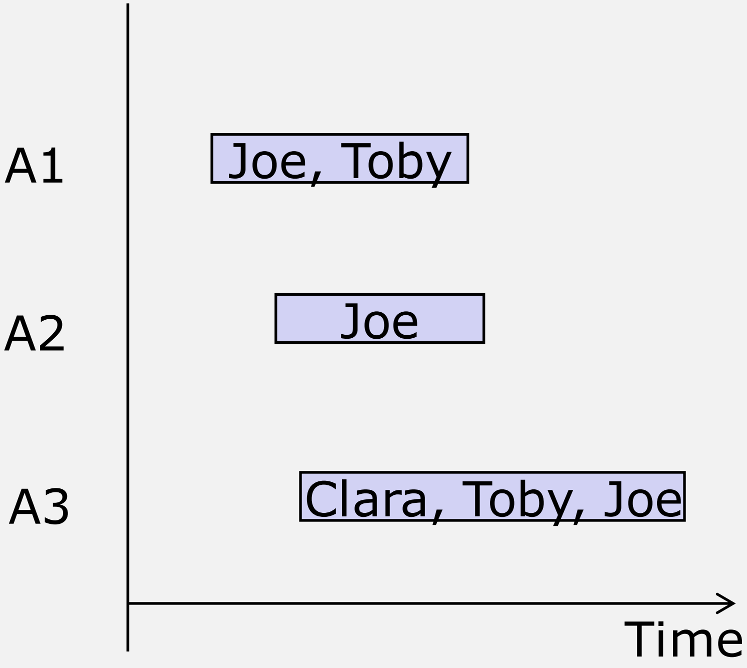
Person-Centered View

To determine people‘s work load

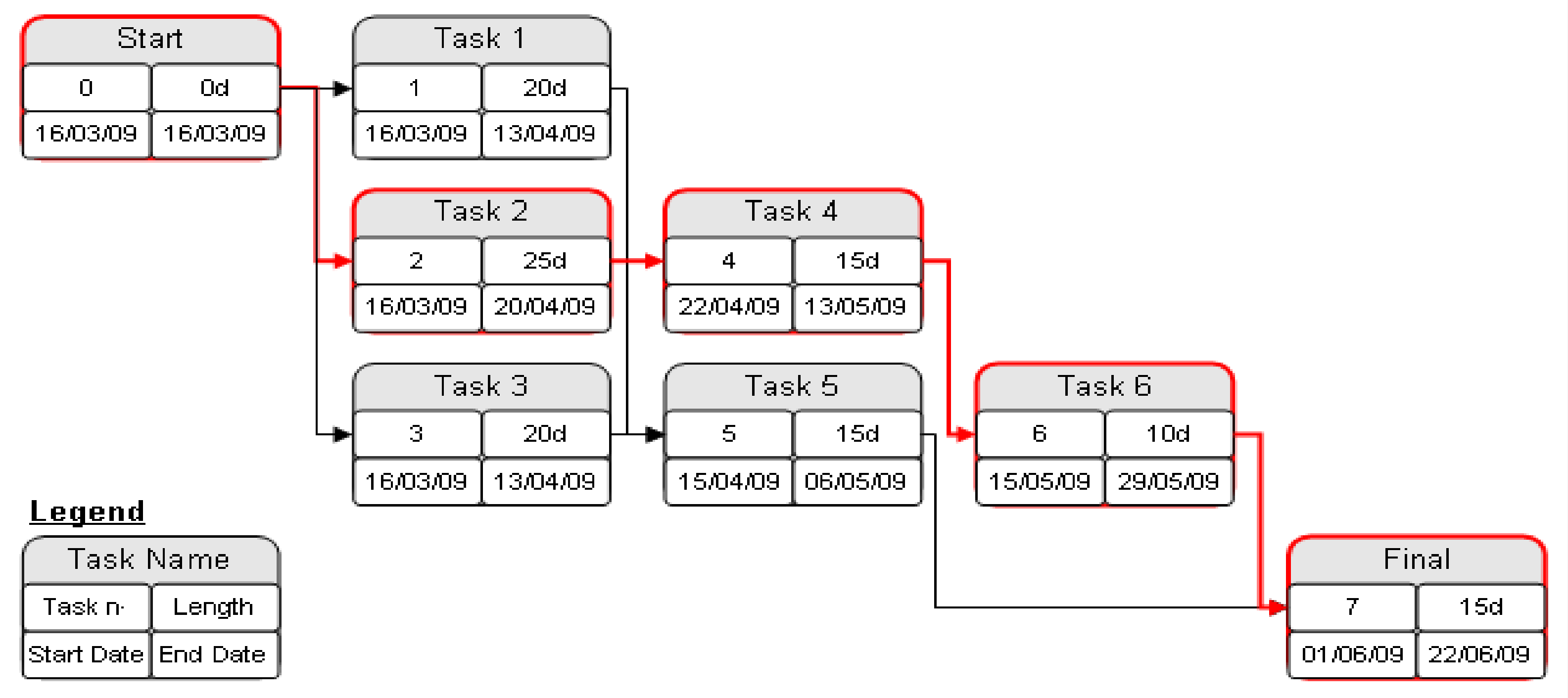


Activity-Centered View

To identify teams working together on the same tasks



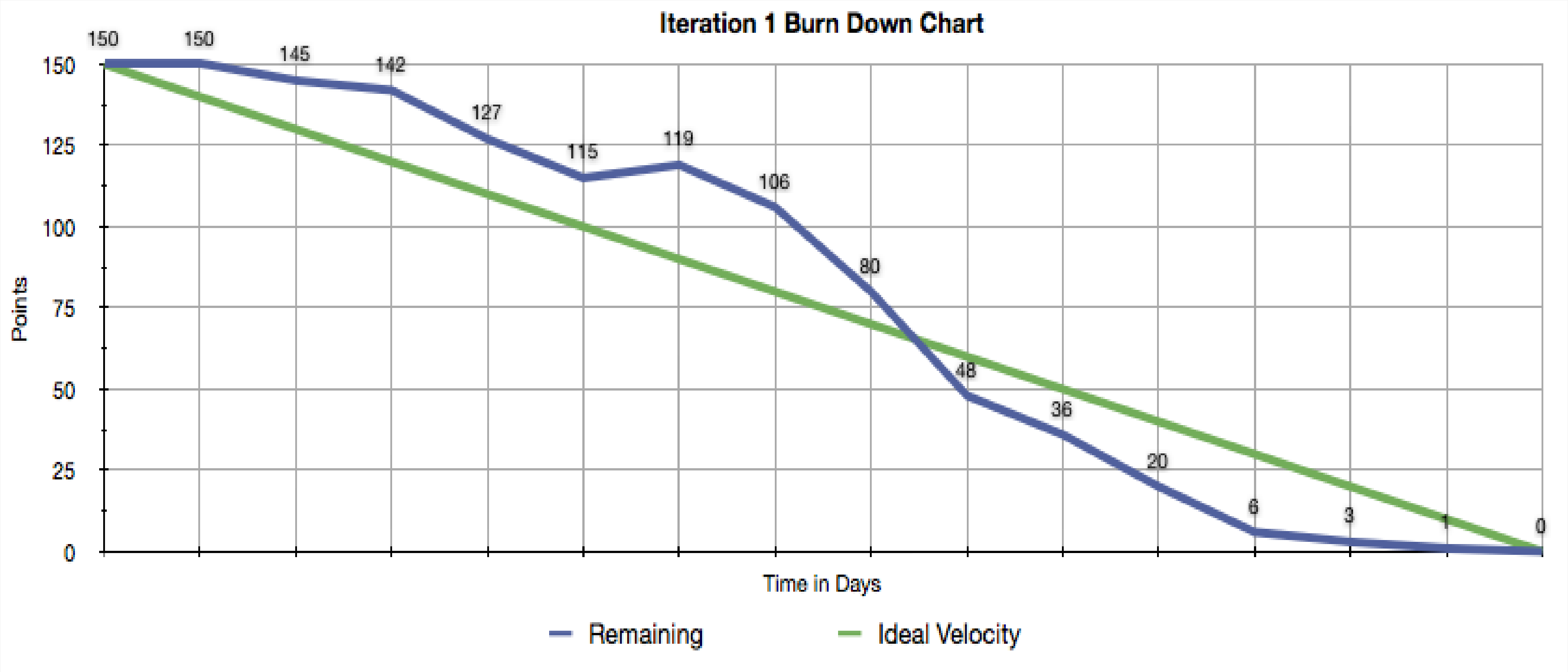
PERT Chart



Good overview of task dependencies

Burndown Chart

number of open tasks over time



Good for project controlling

Which view should you use?

Milestone View:

Good for introduction of project and high executive briefings

Activity View:

Good for developer meetings

Gantt Chart Views:

Base the view on the experience of the participants:

Managing experienced teams - use a person-centered view

Managing beginners - use an activity oriented view

PERT Chart View:

Good for clear illustration of task dependencies.

Burndown Chart View:

Good for progress reports, project controlling.

Scheduling Heuristics

- How to develop an initial project schedule
- How to shorten the project duration
- Mistakes made during preparation of schedules
- The danger of fudge factors
- How to identify when a project goes off-track (actual project does not match the project plan).
- How to become a good software project manager

How to develop an Initial Project Schedule

Identify all your activities

Identify intermediate and final dates that must be met

Assign milestones to these dates

Identify all activities and milestones outside your project that may affect your project’s schedule

Identify “depends on” relationships between the activities

Draw a dependency diagram for the activities and relationships

Determine critical paths and slack times of noncritical paths.

Reducing the planned project time

Recheck the original span time estimates

- Ask other experts to check the estimates
- Has the development environment changed? (batch vs. interactive systems, desktop vs. laptop development)

Consider different strategies to perform the activities

- Consider to Buy a work product instead of building it (Trade-off: Buy-vs.-build)
- Consider extern subcontractor instead of performing the work internally

Reducing the planned project time (2)

Hire more experienced personnel to perform the activities

- Trade-off: Experts work fast, but cost more

Try to find parallelizable activities on the critical path

- Continue coding while waiting for the results of a review
- Risky activity, portions of the work may have to be redone.

Develop an entirely new strategy to solve the problem

Mistakes when Developing Schedules

1. The “Backing in” Mistake
2. Using Fudge Factors

The “Backing in” Mistake

Definition “Backing In”:

- You start at the last milestone of the project and work your way back toward the starting milestone, while estimating durations that will add up to the amount of the available time

Problems with Backing in:

- You probably miss activities because your focus is on meeting the time constraints rather than identifying the required work
- Your span time estimates are based on what you allow activities to take, not what they actually require
- The order in which you propose activities may not be the most effective one.

Instead, start with computing all the required times and then try to shorten the project duration

Using Fudge Factors

Fudge factor:

- A fudge factor is the extra amount of time you add to your best estimate of span time “just to be safe”.
- Example: Many software companies double their span time estimates.

Don’t use fudge factors!

- If an activity takes 2 weeks, but you add a 50% fudge factor, chances are almost zero that it will be done in less then 3 weeks.

Heuristics for dealing with Time

1. First set the Project Start Time
   - Determines the planned project time
   - Determine the critical path(s)
2. Then try to reduce the planned project time
   - If you want to get your project done in less time, you need to consider ways to shorten the aggregate time it takes to complete the critical path.

Avoid fudge factors

Identifying when a Project goes Off-Track

Determine what went wrong: Why is your project got off track?

- Behind schedule
- Overspending of resource budgets
- Not producing the desired deliverables

Identify the reasons

Heuristics to get a Project back on Track

Reaffirm your plan

- Reaffirm your key people
- Reaffirm your project objectives
- Reaffirm the activities remaining to be done
- Reaffirm roles and responsibilities

Refocus team direction and commitment

- Revise estimates, develop a viable schedule
- Modify your personnel assignments
- Hold a mid-project kickoff session
- Closely monitor and control performance for the remainder of the project

Get practical experience

Become a better Software Project Manager

End User and Management involvement   35%

- Learn how to involve the customer and end users
- Learn how to get support from your upper management

Practice project management   30 %

- Do as many projects as possible
- Learn from your project failures

Focus on objectives and requirements   20%

- Distinguish between core, optional and fancy requirements

How to become a better project manager

Don’t assume anything

- Find out the facts.
- Use assumptions only as a last resort.
- With every assumption comes a risk that you are wrong.

Communicate clearly with your people.

- Being vague does not get your more leeway, it just increases the chances for misunderstanding.

Acknowledge performance

- Tell the person, the person’s boss, team members, peers.

View your people as allies not as adversaries

- Focus on common goals, not on individual agendas.
- Make people comfortable by encouraging brainstorming and creative thinking

Be a manager and a leader

- Deal with people as well as to deliverables, processes and systems.
- Create a sense of vision and excitement.

Summary

Dependency Graph:

Identification of dependency relationships between activities identified in the WBS

Schedule

Dependency graph decorated with time estimates for each activity

Critical path and slack time

Forward and Backward Path Analysis

PERT: Technique to analyze complex dependency graphs and schedules

Gantt Chart: Simple notation to visualize a schedule

Thank You