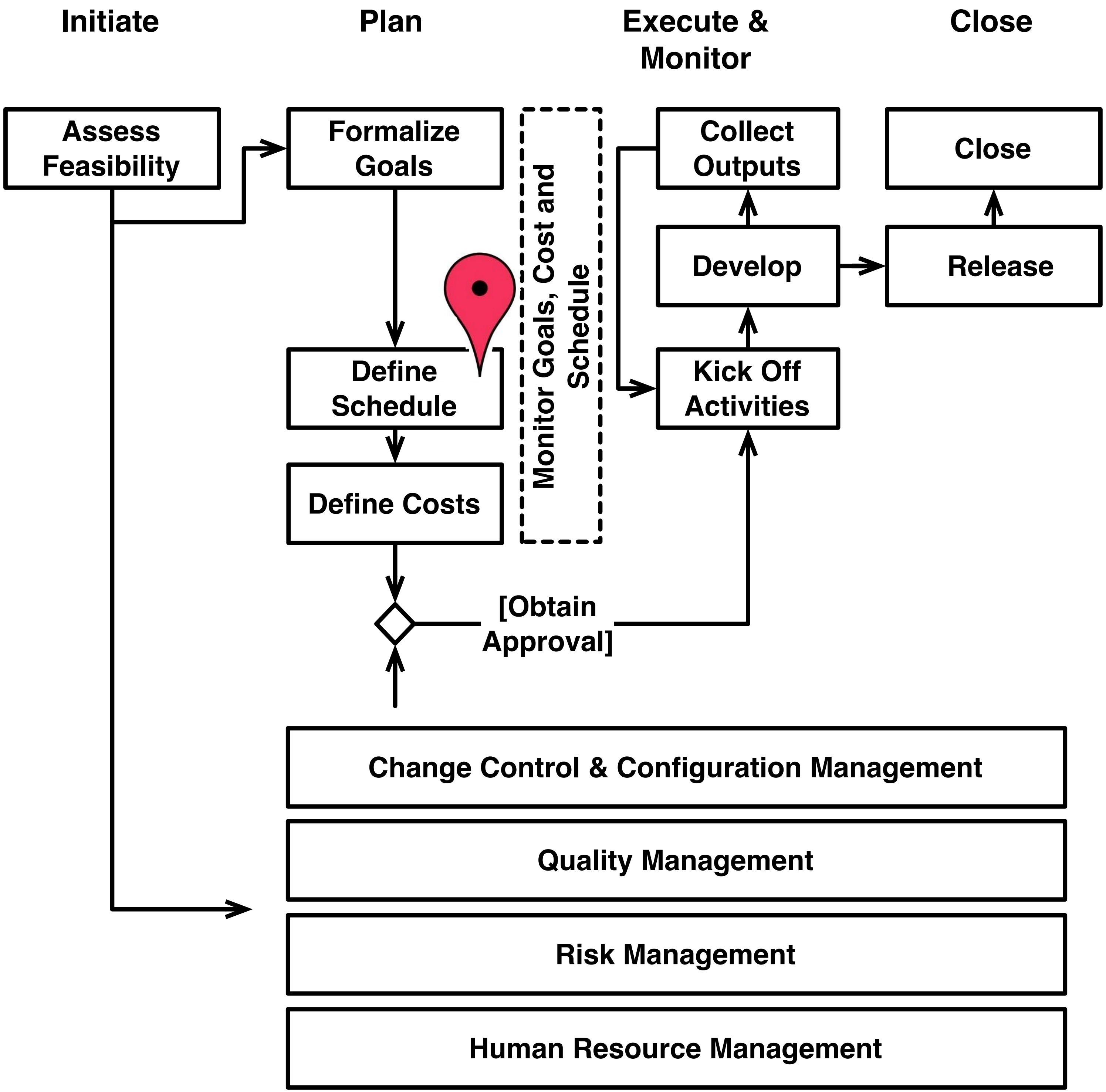
Project Scheduling

Goals of the Unit

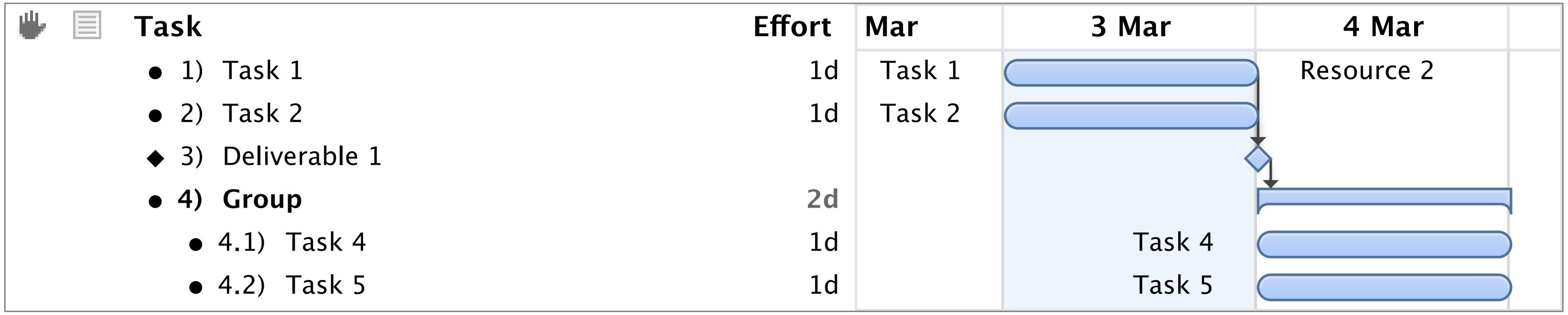
- Making the WBS into a schedule
- Understanding dependencies between activities
- Learning the Critical Path technique
- Learning how to level resources


Initiate
Plan
Execute & Monitor
Close
Assess Feasibility
Formalize Goals
Define Schedule
Define Costs
Monitor Goals, Cost and Schedule
Collect Outputs
Develop
Kick Off Activities
Close
Release
[Obtain Approval]
Change Control & Configuration Management
Quality Management
Risk Management
Human Resource Management

Overview

- We have:
  - A WBS (activities)
  - Effort (duration) estimations for each element of the WBS
- We want to schedule activities, so that we know when each activity starts and ends, when we need resources, when we deliver
- Process:
  - Identify constraints (dependencies)
  - Allocate and level resources
  - Find the critical path and iterate till the plan is satisfactory
- Output: Gantt Chart

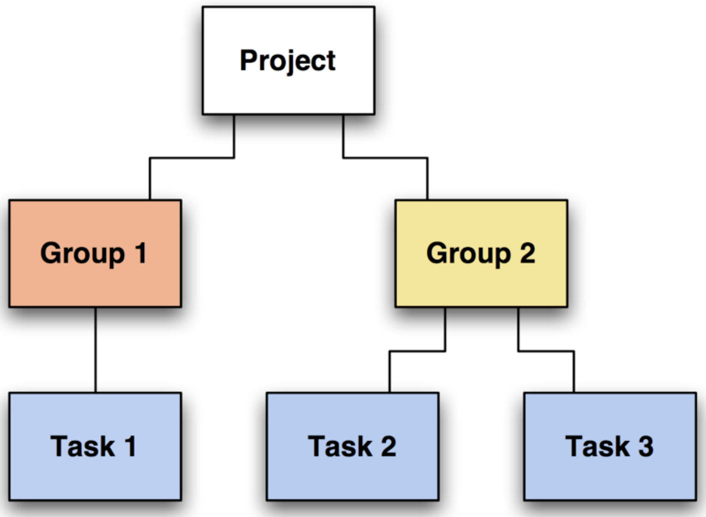
The modern Gantt chart



Task	Effort
● 1) Task 1	1d
● 2) Task 2	1d
◆ 3) Deliverable 1	
● **4) Group**	**2d**
● 4.1) Task 4	1d
● 4.2) Task 5	1d

- Textual Outline + Calendar Graph
- Activities as bars (possibly annotated with names and resources)
- Deliverable (as diamonds)
- Activities can be grouped (information of group is derived by lower level activities)
- Dependencies among tasks

The modern Gantt chart and the WBS



Project

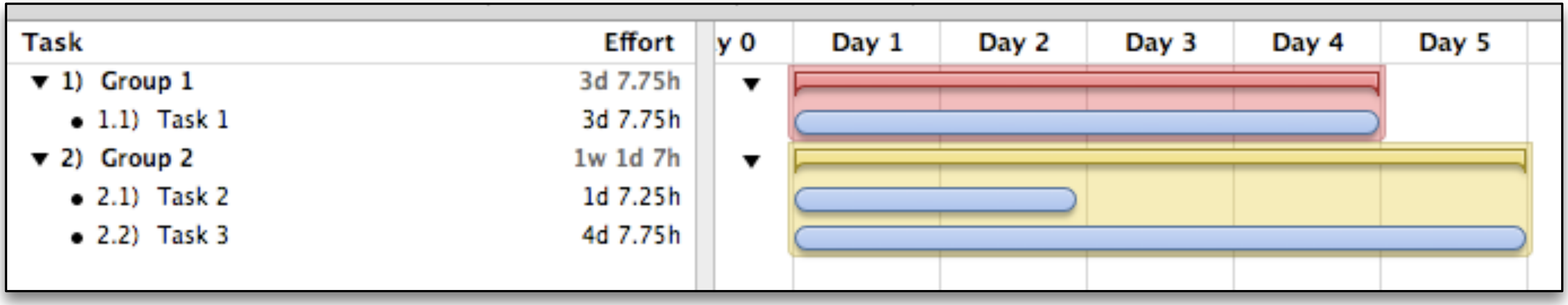
1. Group 1

1.1 Task 1

2. Group 2

2.1 Task 2

2.2 Task 3



Task	Effort
▼ 1) Group 1	3d 7.75h
• 1.1) Task 1	3d 7.75h
▼ 2) Group 2	1w 1d 7h
• 2.1) Task 2	1d 7.25h
• 2.2) Task 3	4d 7.75h

Identify the constraints (dependencies)

Identify Dependencies

- The execution of activities is constrained by the logic of the plan (you do not build the roof before the foundations and structure of a house are laid completed)
- Hard and soft dependencies (definition in the next two slides).
- When using planning tools:
  - Specify only "hard" dependencies
  - "Soft" dependencies are typically inserted by the planning tool

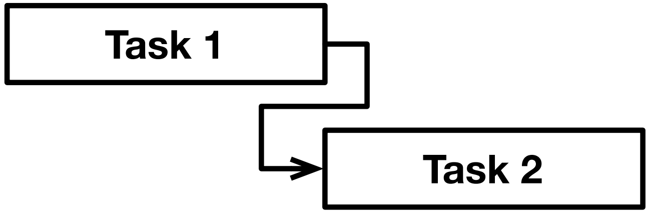
Hard Dependencies

- Not much you can do about it...
- They might either derive:
  - From the project “logic” (e.g. testing has to come after coding)
  - From external dependencies (e.g. a contract sign-off; a particular alignment of planets is necessary to launch a spacecraft)
- Eliminating hard dependencies can be done, at a cost (e.g., increased risk, re-work)

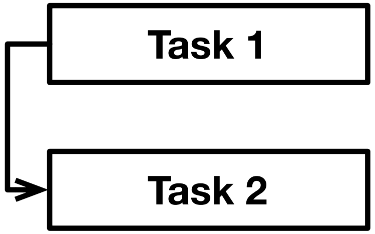
Soft Dependencies

- Due to a choice among all possible alternative plans
- They might either derive:
  - From discretionary choices (e.g., the PM chooses the order in which modules are to be developed)
  - From resource availability and leveling (e.g., the PM or the planning tool sequences two tasks relying on the same resource)
- Notice that, as time progresses, it might become difficult or impossible to “undo” soft dependencies (e.g. a resource is shared by different projects)

Task Dependency Relationships



- Finish-to-Start (FS)
  - B cannot start till A finishes
  - Most commonly used



- Start-to-Start (SS)
  - B cannot start till A starts
  - Perform experiment; monitor experiment
- Finish-to-Finish (FF)
  - B cannot finish till A finishes
- Start-to-Finish (SF)
  - B cannot finish till A starts (rare)

Lead and Lag Time

- Dependencies between activities can have a non zero duration
- **Lag time** = delay introduced by the dependency is positive (some time passes between the two tasks)
- **Lead time** = the duration of the dependency is negative (the activities partially overlap)

Some rules of the thumb

- Use milestones (and deliverables) to clearly mark "phase" transitions (or some important transitions from an activity to another)
- Try and minimize task dependencies (to minimize delays due to some activities waiting for some other activities to end)
- Evaluate alternatives
- Certain activities might just depend on calendar (and be constrained by dates)
- Take into account all dimensions (cost, quality, and time): minimize time might increase costs, risks, and compromise quality

Critical Path Method

Critical Path

- Not all activities are equally important or critical in a plan
- The critical path method looks at those activities which determine the duration of a plan
- These activities constitute the critical path
- Any arbitrarily small delay in any activity in the critical path will delay the finish date of a project
- The computation is based on Network Diagrams (a graph representation of the plan)

Network Diagrams

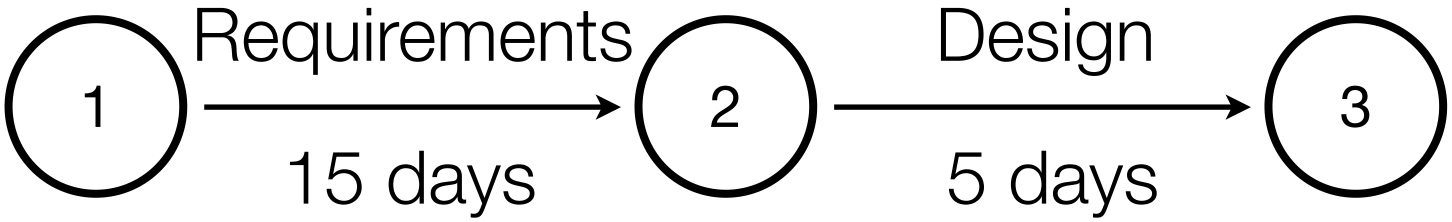
- Developed in the 1950's
- A graphical representation of the tasks necessary to complete a project (plan as graph)
- Visualize the flow of tasks & relationships
- Two classic formats
  - AOA: Activity on Arc (or Activity on Arrow)
  - AON: Activity on Node
- Conventions:
  - Each task labeled with an identifier and a duration (in std. unit like days)... variations are possible
  - There is one start and one end event
  - Time goes from left to right

Network Diagrams

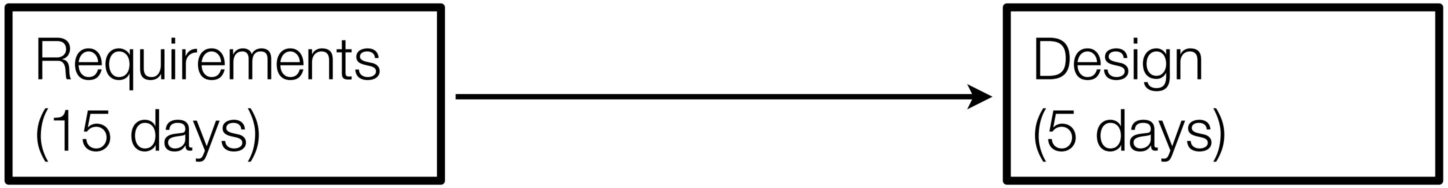
- AOA (Activity on Arrow)
  a.k.a ADM (Activity Diagramming Method):
  - Circles represents Events (e.g. 'start' or 'end' of a given task)
  - Lines representing Tasks, such as 'Design'
- AON (Activity on Node)
  a.k.a. PDM (Precedence Diagramming Method):
  - Tasks are on Nodes
  - Arcs represents dependencies between task

Graphical Formats

AOA: Activity on Arc



AON: Activity on Node



... which one is better?

AOA/AON Comparison

- AOA initially used by Walker and Kelly for PERT
- AON more flexible and easier to draw
- AOA simpler to use for certain algorithms

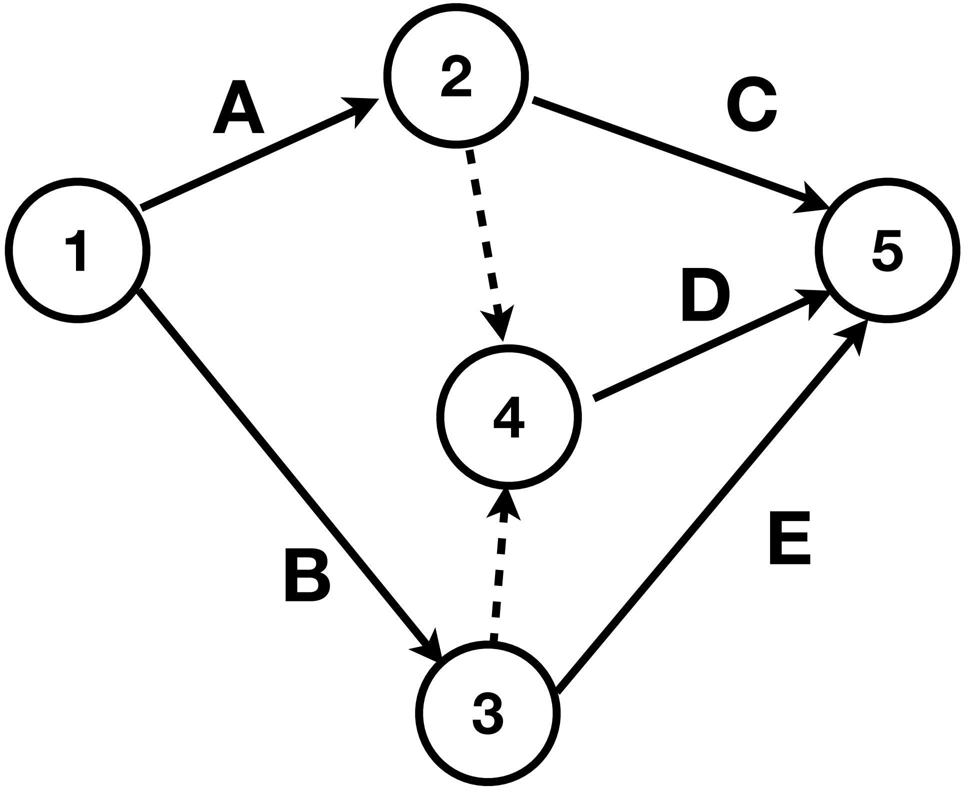
... we will stick (mostly) to AON

Example: AOA/AON Comparison

Consider the following plan:

Activity	**Predecessors**	**Duration**
A	None	3 months
B	None	4 months
C	A	3 months
D	A, B	1 month
E	B	2 months

Example: AOA/AON Comparison



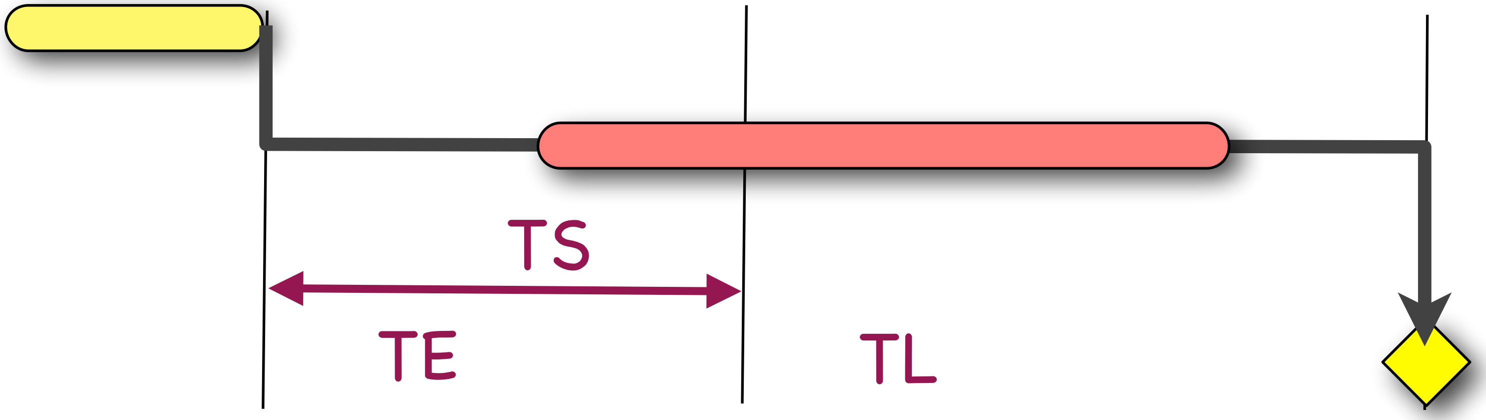
- In the AOA notation, some dependencies might require “dummy” arcs and nodes to be introduced (*)

(*) Notice that, since we can/have to add nodes and arcs, a plan does not have a unique AOA associated to it

Critical Path Computation

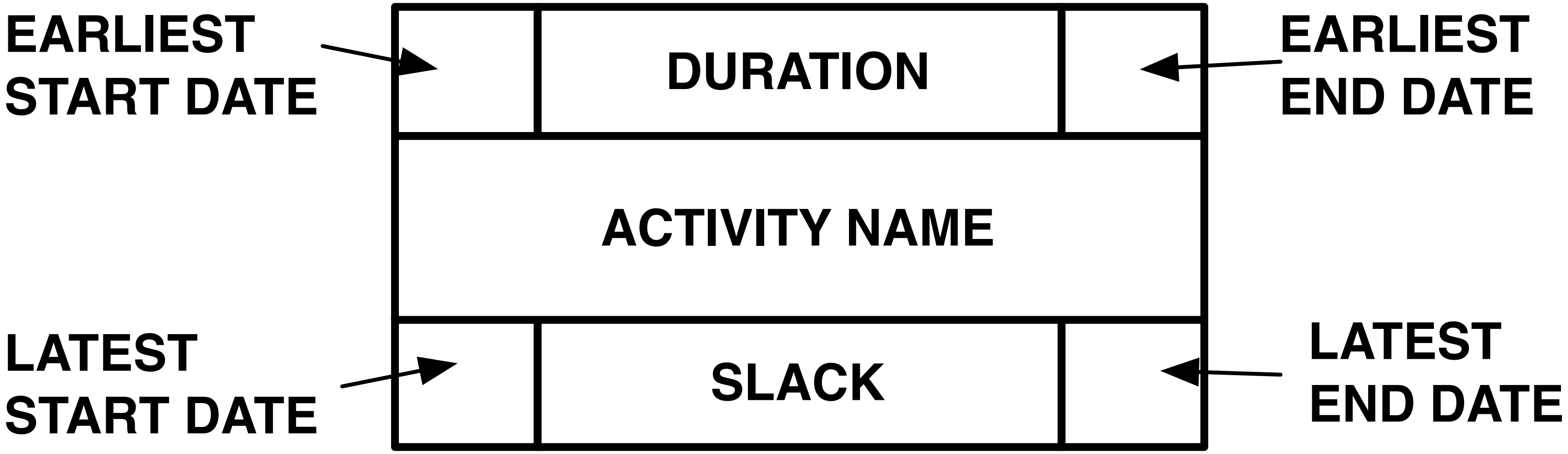
Slack & Float (synonyms)

- Free Slack
  - Slack an activity has before it delays next task
- Total Slack
  - Slack an activity has before delaying whole project
- Slack Time TS = TL – TE
  - TE = earliest time an event can take place
  - TL = latest date it can occur w/o extending project's completion date or next activity



Critical Path Computation

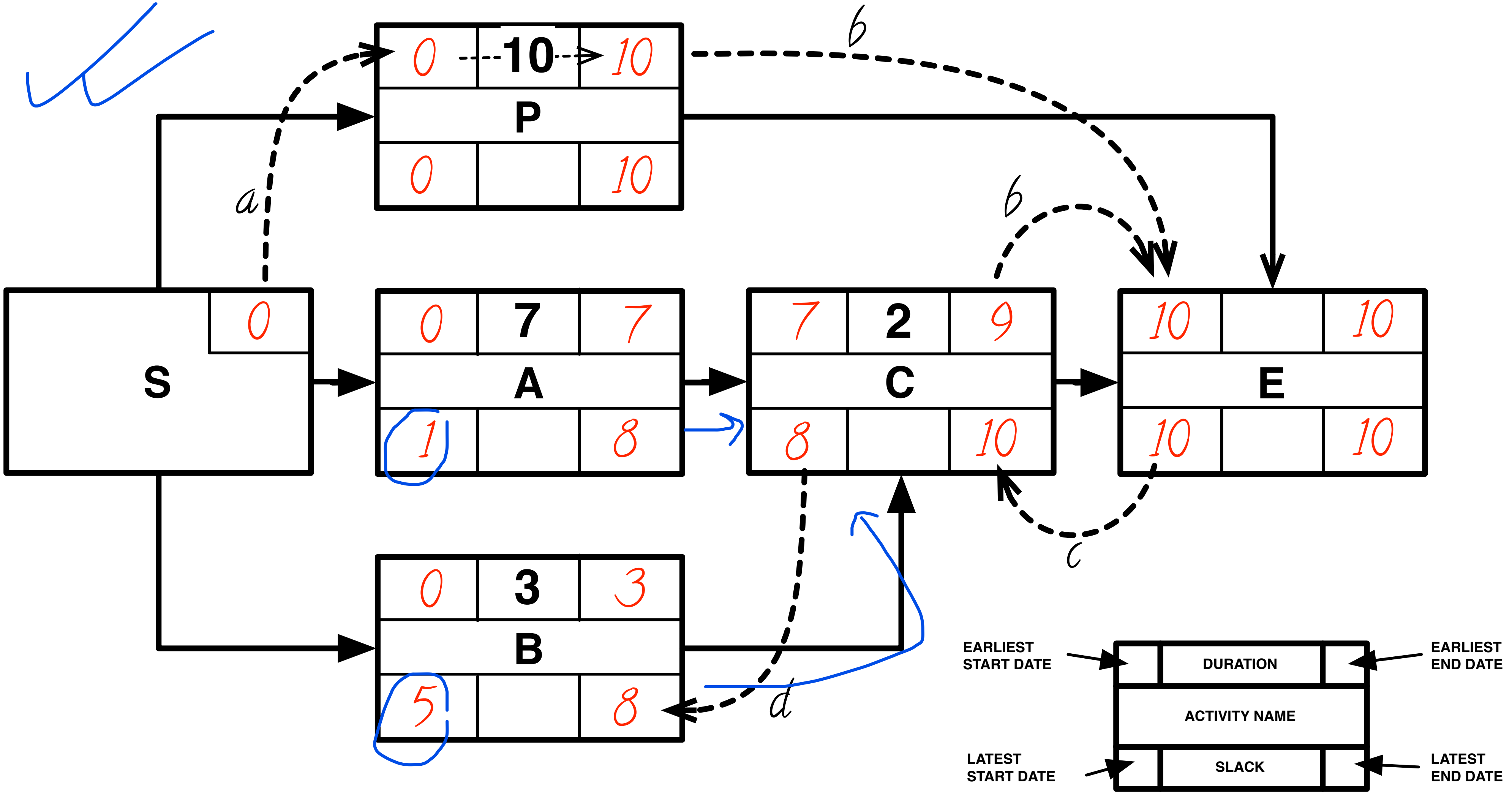
- Goal: given a plan (activities, duration, and dependencies), determine Slack, Earliest and Latest dates of each activity
- Notation: AON with nodes represented as follows



Critical Path Computation

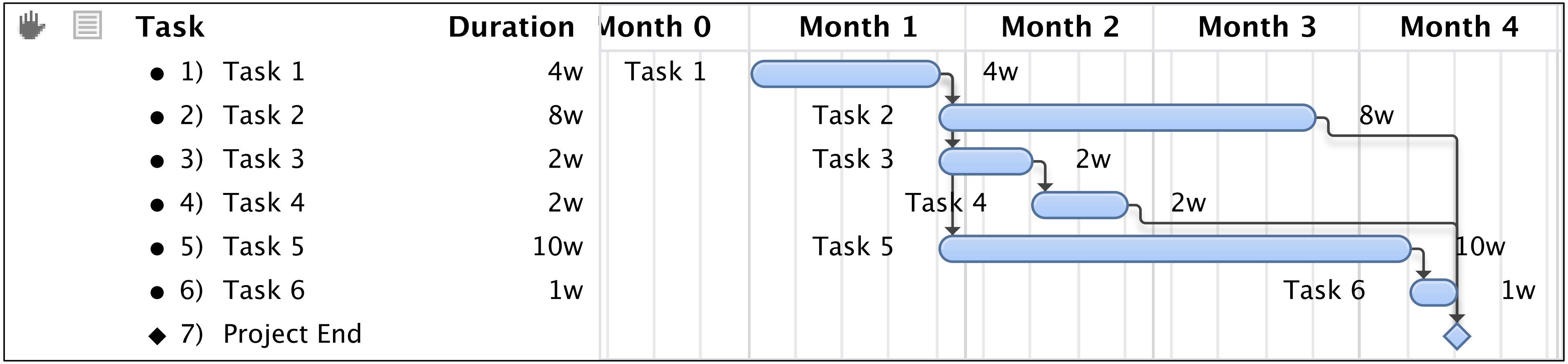
- A **forward pass** determines the earliest start and end dates of each activity in the plan
- A **backward pass** determines the latest start and end dates of each activity in the plan
- The difference between earliest start (end) and latest start (end) is the slack of an activity
- The **critical path** is the path in which all activities have zero slack
- A plan always has a critical path... changing the plan changes what activities are in the critical path

Example 1



0
10
10
P
0
10
a
b
b
S
0
0
7
7
A
1
8
7
2
9
C
8
10
10
10
E
10
10
c
0
3
3
B
5
8
d
EARLIEST START DATE
DURATION
EARLIEST END DATE
ACTIVITY NAME
LATEST START DATE
SLACK
LATEST END DATE

Example 2



	Task	Duration
●	1) Task 1	4w
●	2) Task 2	8w
●	3) Task 3	2w
●	4) Task 4	2w
●	5) Task 5	10w
●	6) Task 6	1w
◆	7) Project End	

- “Informal approach”: have a look at what activities can slide in a plan without moving the end date of a project (e.g. Task 3 is not in the critical path)
- CPM highlighted automatically by many Gantt charting tools

Critical Path Method Remarks

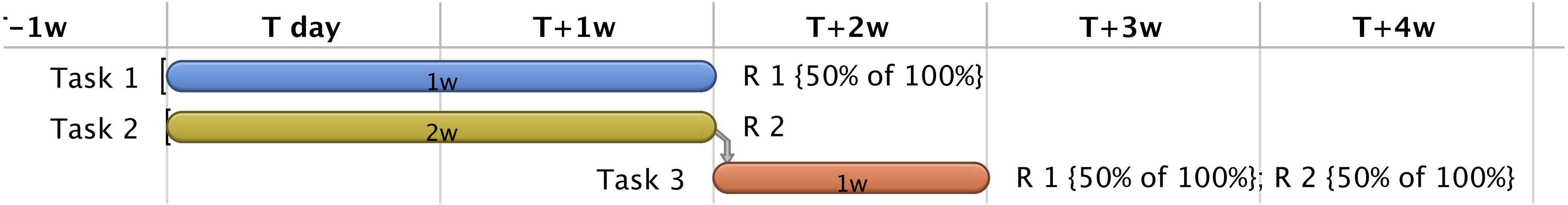
- Critical path refers just to duration and not to other characteristics such as risk or difficulty
- Activities which are not in the critical path **can** delay a plan, if the delay is long enough.
- Watch out for (nearly) critical paths: a delay in an activity in a non-critical path may make another path critical

Resource Allocation and Resource Leveling

A (simplified) Process

- Inputs:
  - the plan: activities, constraints, effort for each activity
  - project team (number, types, and availability of resources)
  - delivery dates (project constraints)
- Resource allocation:
  - the process by which a resource is assigned to a task, that is, is tasked with carrying out part of the work (effort) defined in a task
- Constraints:
  - according to availability and needs (e.g. the type of resource required for a given activity): no over-allocation (above maximum availability) (resource leveling)
- If no solution is found, if you may, variate some hypotheses (e.g. increase team size, relax constraints) and iterate

Resource Allocation Examples



- Legenda:
  - each slot: 1week
  - R1 assigned to Task 1 at 50% of his time
  - R2 allocated full time to Task 2
  - R1 and R2 allocated @ 50% of their time to Task 3
- What it means:
  - R1 will work 20 hours on week 1 and 2 and 20 hours on week 3
  - R2 will work 40 hours on week 1 and 2 and 20 hours on week 3

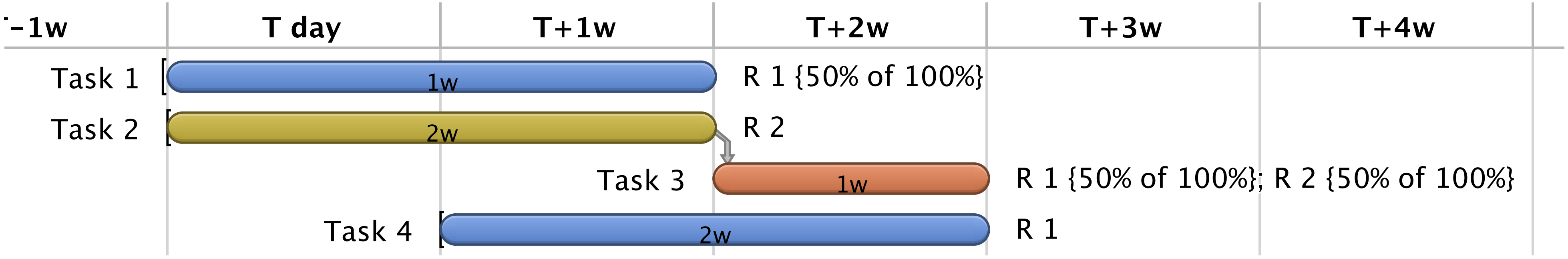
Resource Usage

- For manpower: the amount of time each resource is needed at a given time
- For equipment: the number of items that are necessary at any given time
- For material: the amount of material which is required (consumed) at any given time

How is it computed?

- Resource usage is computed by summing the amount of work required for any given period
- That is a "vertical" sum over work assignments
- Overallocation: a situation in which a resource is used above his/her/its maximum capability

Example

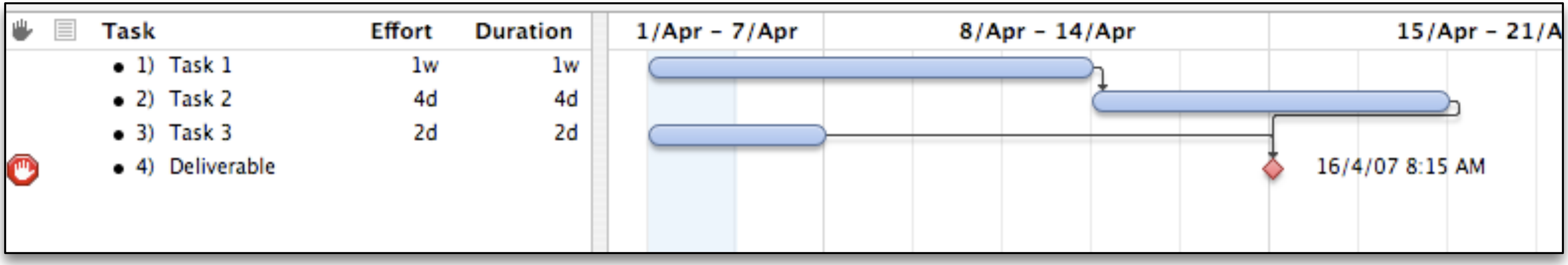


	hours	**hours**	**hours**	**hours**	**hours**
R1	20	20		T1	
			20	T3	
		40	40	T4	
Total R1	**20**	**60**	**60**		
R2	40	40		T2	
			20	T3	
Total R2	**40**	**40**	**20**		

R1 is over allocated in W2 (T+1w) and W3 (T+2w)

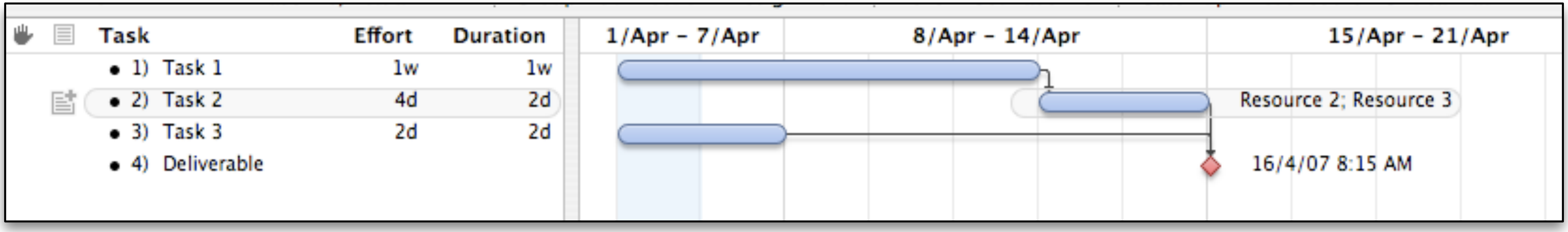
More Complete Example

We draw the plan highlighting hard constraints. Deliverable has a unmovable delivery date



Task	Effort	Duration
• 1) Task 1	1w	1w
• 2) Task 2	4d	4d
• 3) Task 3	2d	2d
• 4) Deliverable		

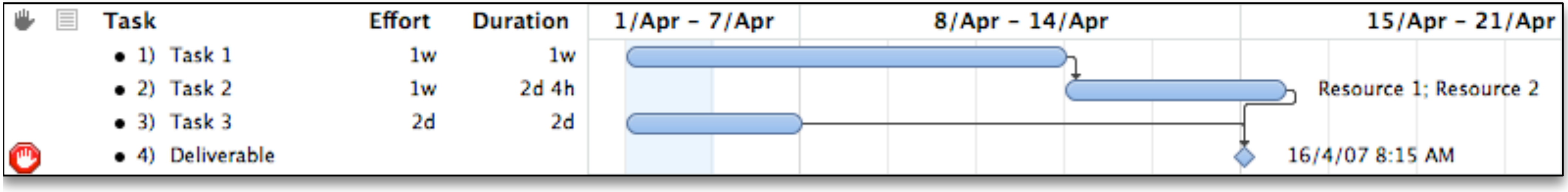
Allocating two resources to Task 2 allows to satisfy the constraints



Task	Effort	Duration
• 1) Task 1	1w	1w
• 2) Task 2	4d	2d
• 3) Task 3	2d	2d
• 4) Deliverable		

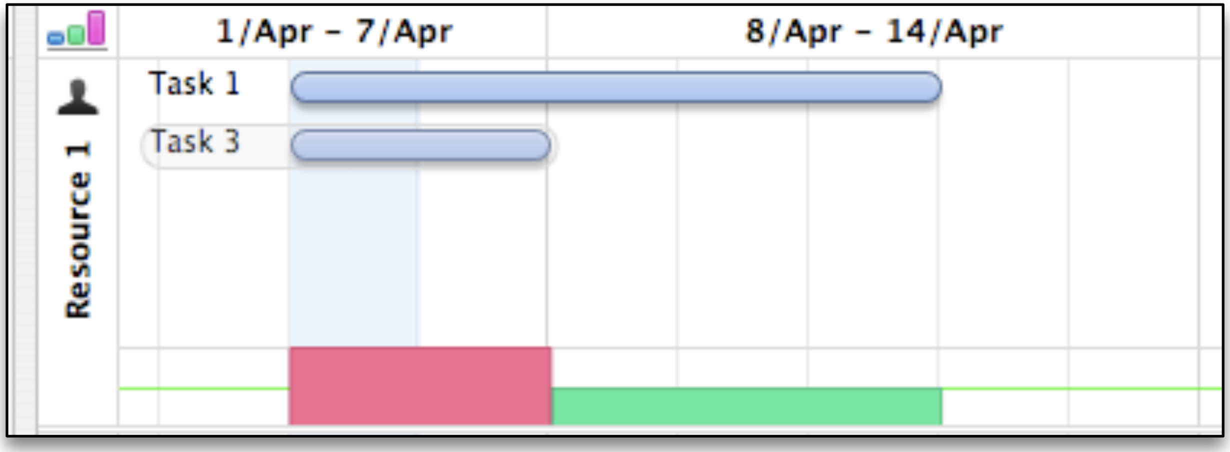
Example

Problem: Task 1 and Task 3 require the same resource



Task	Effort	Duration
1) Task 1	1w	1w
2) Task 2	1w	2d 4h
3) Task 3	2d	2d
4) Deliverable		

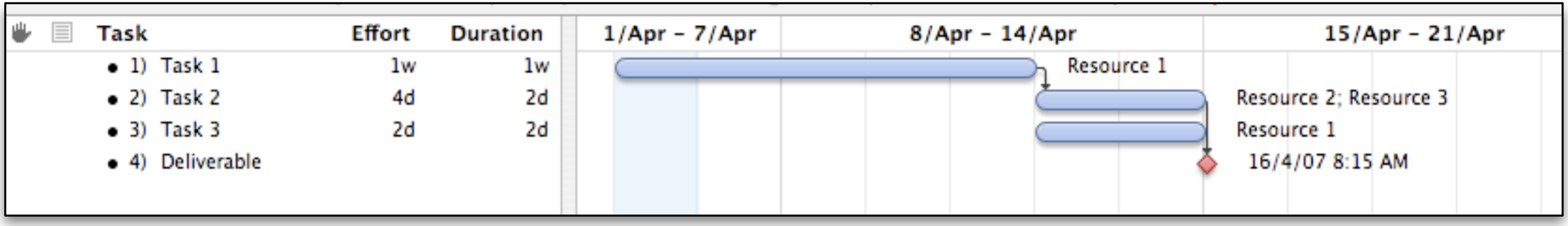
... we are over-allocating Resource 1



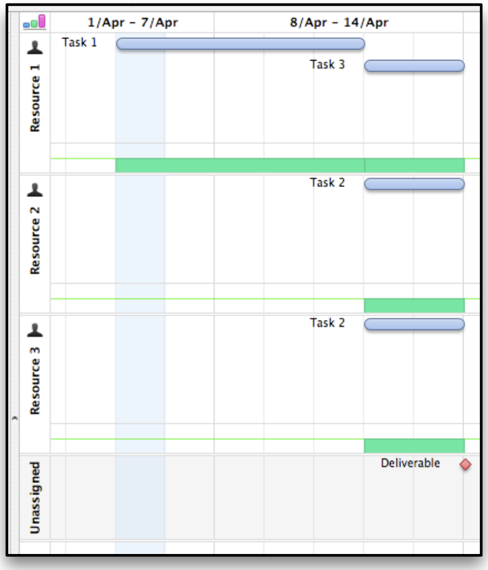
Example

Solution 1. Resource leveling... insert soft constraints in your plan so that no resource is over allocated (does not work above 100%)

Solution 2. Compression techniques (in a few lessons)



Task	Effort	Duration
• 1) Task 1	1w	1w
• 2) Task 2	4d	2d
• 3) Task 3	2d	2d
• 4) Deliverable		



Some considerations:

- Resource 1 will work on the project full time.
- Resource 2 and Resource 3 needed just towards the end of the project (for Task 2)