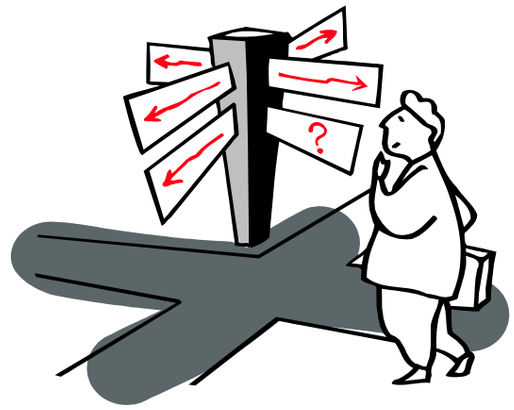
CHAPTER 2 – Project Management



Introduction
- When, Why and What?

Planning & Monitoring
- PERT charts
- Gantt charts
- Uncertainty ⇒ Risk to the schedule

- Dealig with delays
- Monitoring: earned value analysis
  + Tasks completes, Timesheets
  + Slip Lines, Timelines

Organisation, Staffing, Directing
- Team Structures
- Directing Teams

Conclusion
- Correctness & Traceability

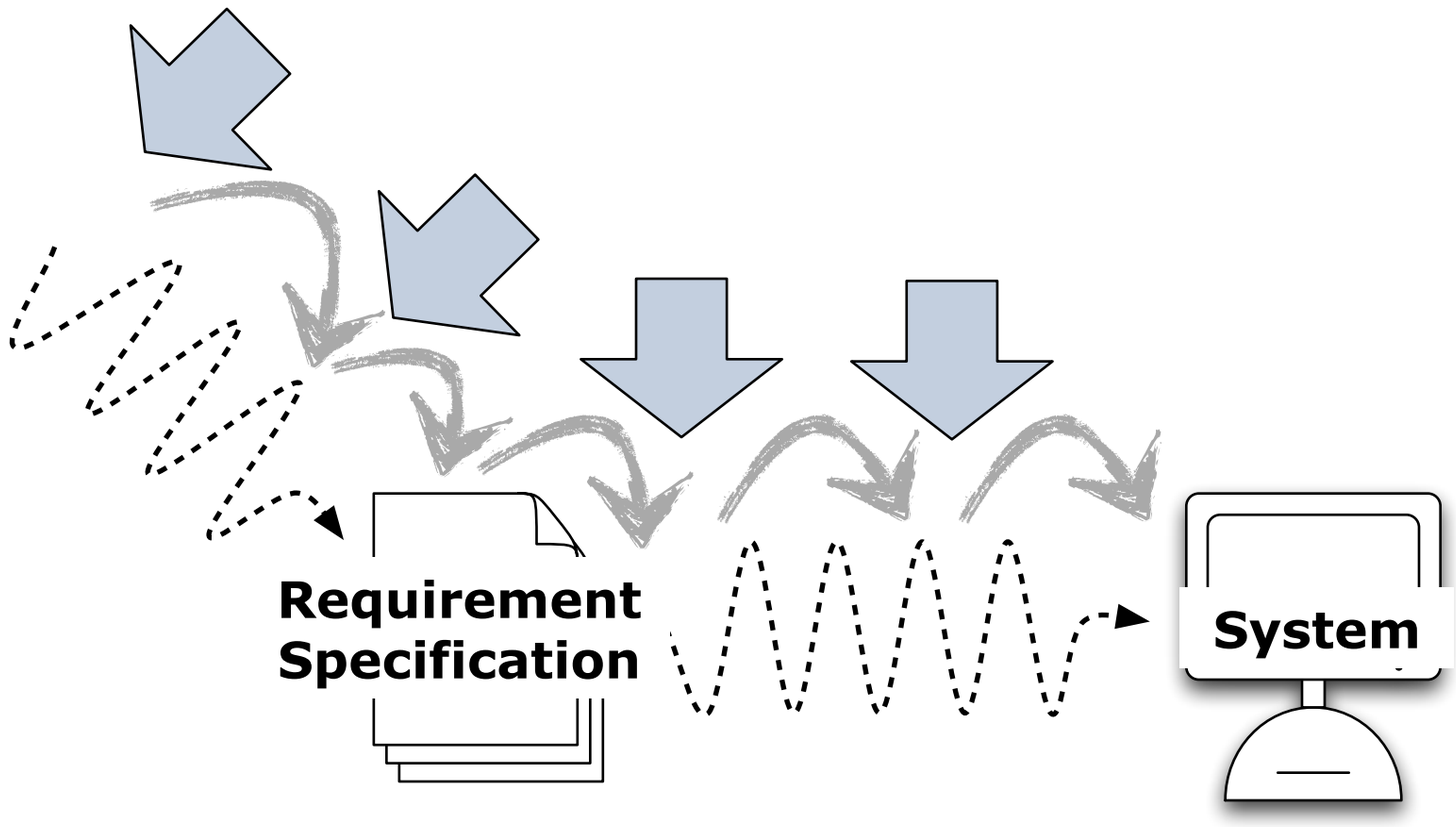
Literature

- [Ghez02a] In particular, “Management of Software Engineering”
- [Pres01a] In particular, “Software Project Planning” & “Project Scheduling and Tracking”
- [Somm04a] In particular, “Software Cost Estimation” & “Managing People”

Other

- [Broo75a]The Mythical Man-Month, F. Brooks, Addison-Wesley, 1975.
  ➡ “adding people to a late project makes it later” + lots of timeless wisdom
- [Dema98a] Peopleware - Productive Projects and Teams (2nd ed.), Tom DeMarco and Timothy Lister, Dorset House Publishing Company, 1998
  ➡ Lots of advice on how the sociology in teams affect productivity
- [Hugh99a] Software Project Management, B. Hughes and M. Cotterell, McGraw Hill, 1999.
  ➡ Good practical examples on PERT, Gantt, Time-sheets, ...
- [Gold95a] Succeeding with Objects: Decision Frameworks for Project Management, A. Goldberg and K. Rubin, Addison-Wesley, 1995.
  ➡ Explains how to define your own project management strategy

When Project Management



Ensure *smooth* process

Why Project Management ?

Almost all software products are obtained via projects.
⇒ Every product is unique

(as opposed to manufactured products)

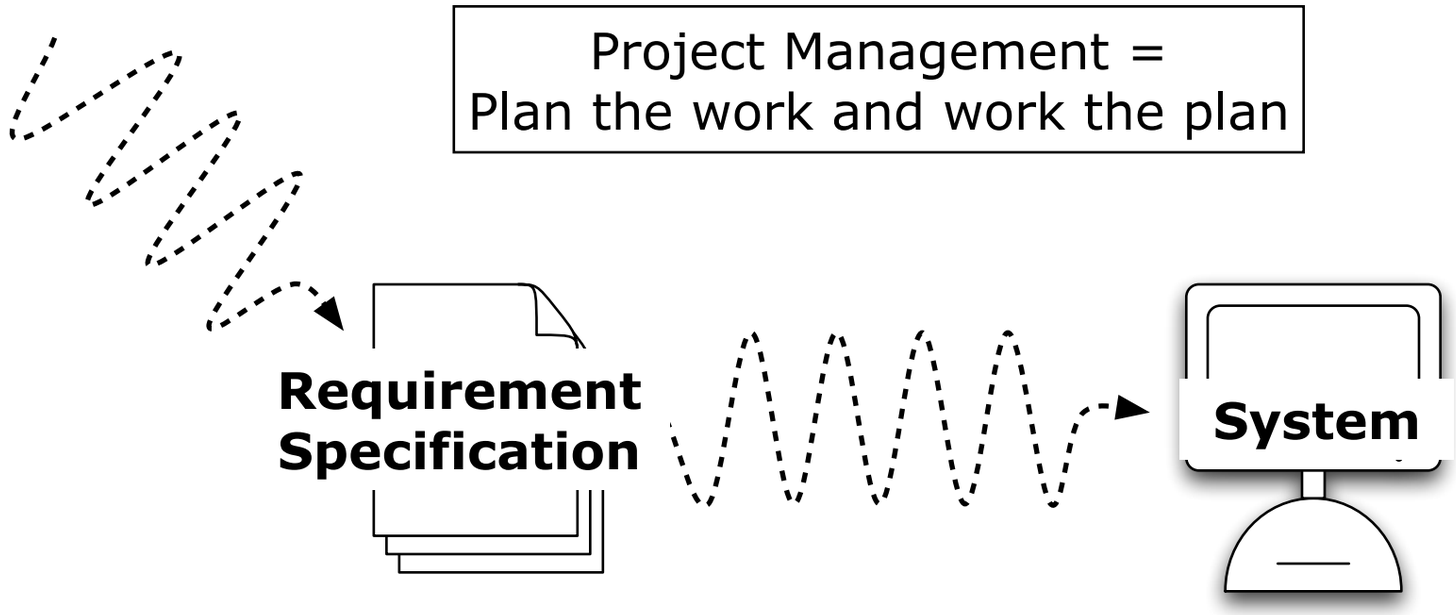
Software Project = <u>Deliver</u> on <u>time</u> and within <u>budget</u>

Achieve interdependent & conflicting goals …

… with limited resources.

Your project team is a resource !

What is Project Management ?



Management Functions

- Planing: Breakdown into tasks + Schedule resources.
- Organization: Who does what ?
- Staffing: Recruiting and motivating personnel.
- Directing: Ensure team acts as a whole.
- Monitoring (Controlling): Detect plan deviations + take corrective actions.

Focus of this lecture is Planning & Monitoring.
(Other functions are best learned in real life.)

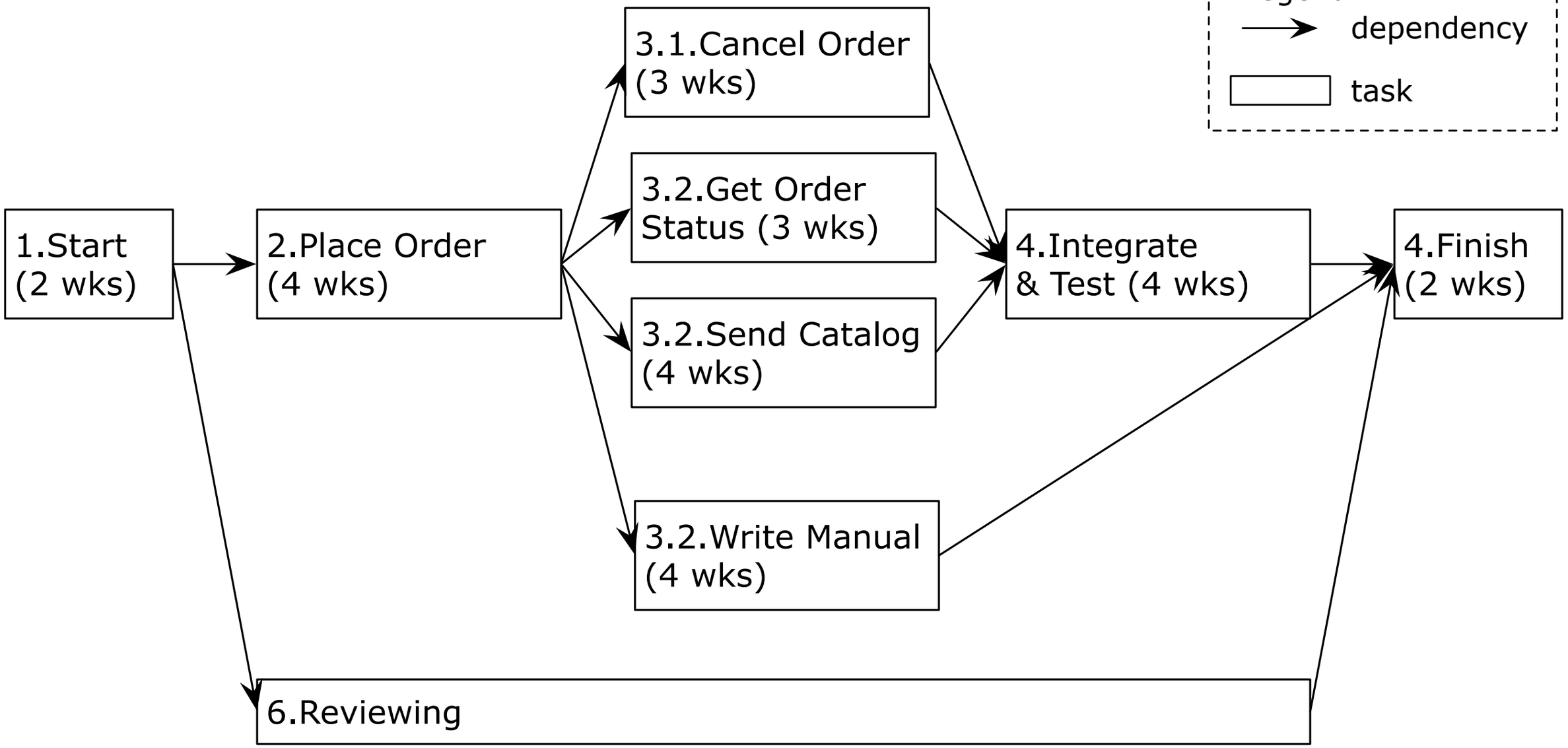
Tasks & Milestones

Good planning depends a lot on project manager's intuition and experience!

- Split project into tasks
  - Tasks into subtasks etc.
- For each task, estimate the *time*
  - Define tasks small enough for reliable estimation.
- Most tasks should end with a *milestone*.
  - Milestone =
    A verifiable goal that must be met after task completion
    ➡ Verifiable? .... by the customer
  - Clear unambiguous milestones are a necessity!
    ("80% coding finished" is a meaningless statement)
  - Monitor progress via milestones
- Organize tasks concurrently to make optimal use of workforce
- Define dependencies between project tasks
  + Total time depends on longest (= critical) path in activity graph
  + Minimize task dependencies to avoid delays

Planning is iterative ⇒ monitor and revise schedules during the project!

PERT Chart: Task Dependencies



- 1 start node & 1 end node
- time flows from left to right
- node numbering preserves time dependencies
- no loops, no dangling nodes

Remember: small tasks & milestones verifiable by curstomer !

Finding the Critical Path

Forward Pass: compute "earliest start-date" (ESD)

- ESD (start-node) := start-date project
- Breadth-first enumeration (use node numbering)
- For each task n: compute earliest start-date
  = Latest of all incoming paths
  ➡ ESD (n) := latest of (
  ESD (preceding) + estimated time (preceding))

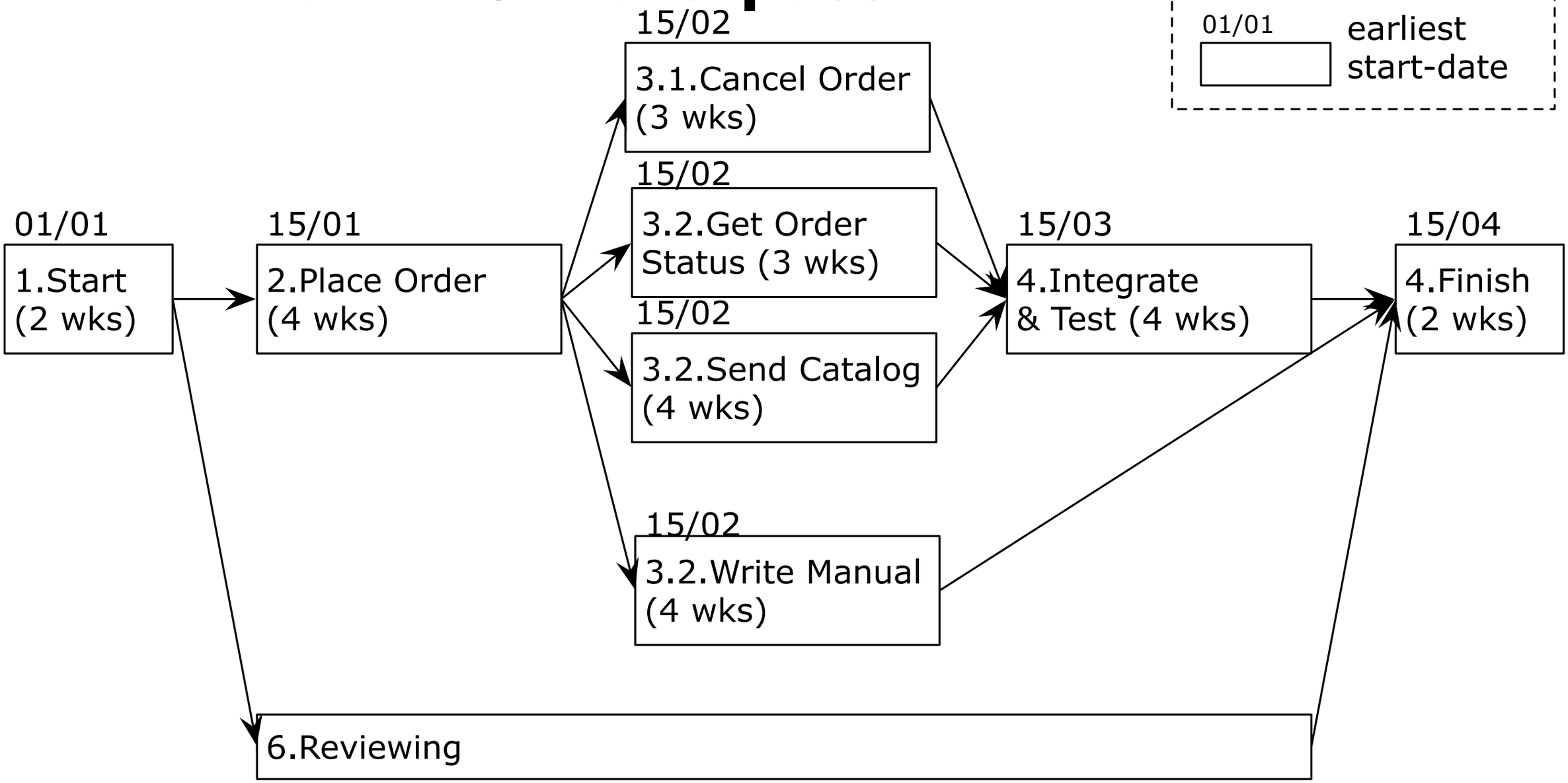
Backward Pass: compute "latest end-date" (LED)

- LED (end-node) := ESD (end-node) + estimated time
- Breadth-first enumeration (node numbering in reverse order)
- For each task n: compute latest end-date
  = Earliest of all outgoing paths
  ➡ LED (n):= earliest of (
  LED (subsequent) - estimated time (subsequent))

Critical Path

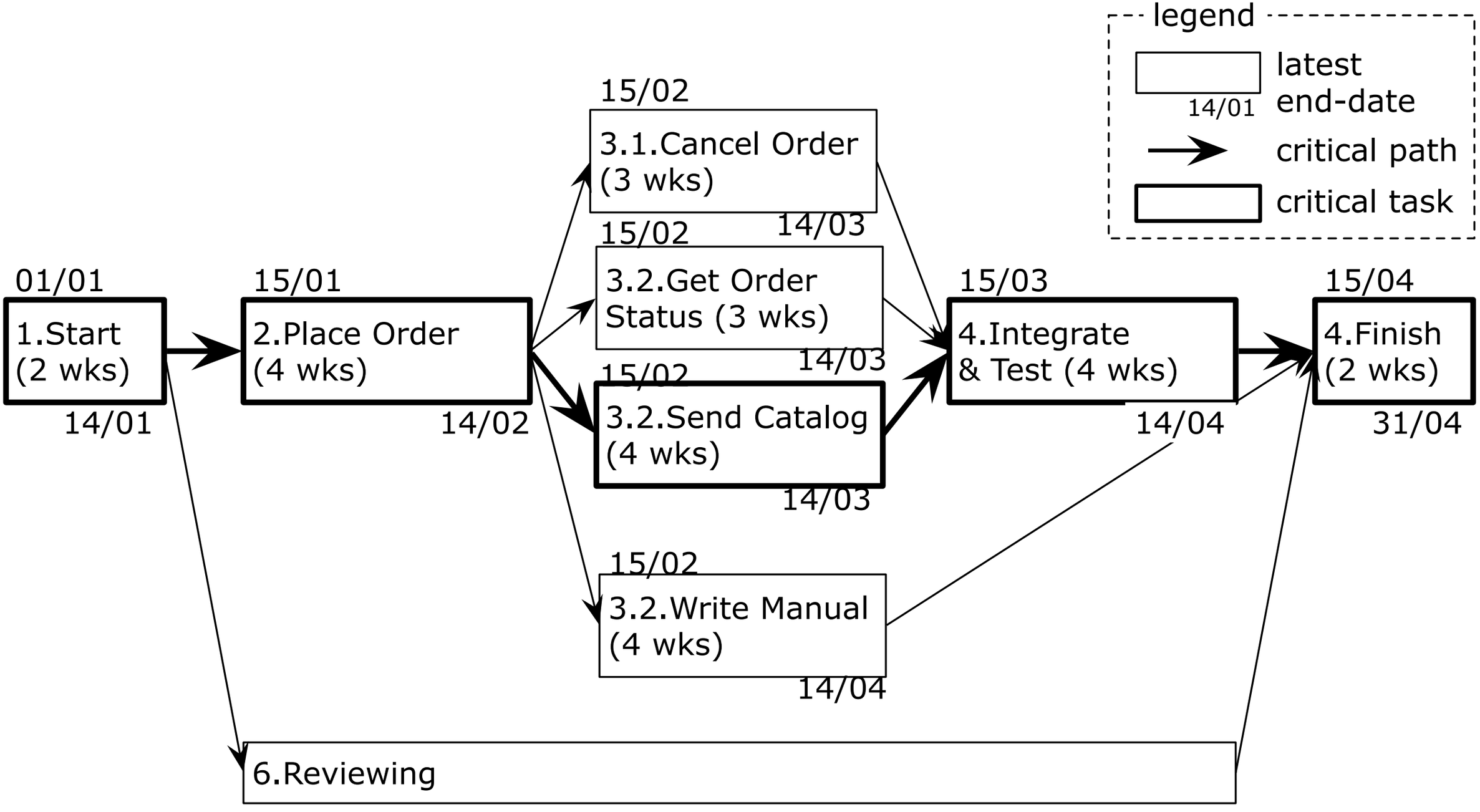
- = path where delay in one task will cause a delay for the whole project
- path where for each node n:
  ESD(n) + estimated time (n)= LED(n)

PERT Chart: Forward pass



- ESD(1) := start-date project
- ESD(2) := ESD(1) + time(1) := 01/01 + 2 weeks := 15/01
- ESD(4) := latest (ESD(3.1) + 3 wks, ESD(3.2) + 2 wks, ESD(3.3) + 4 wks) := 15/03

PERT Chart: Backward pass + Critical path



- LED(7) := ESD(7) + time(7) := 15/04 + 2 wks := 31/04
- LED(6) := LED(7) - time(7) := 31/04 - 2 wks := 14/04
- LED(2) := earliest (LED(3.1) - 3 wks, LED(3.2) - 2 wks, LED(3.3) - 4 wks) := 14/02

When to use PERT Charts ?

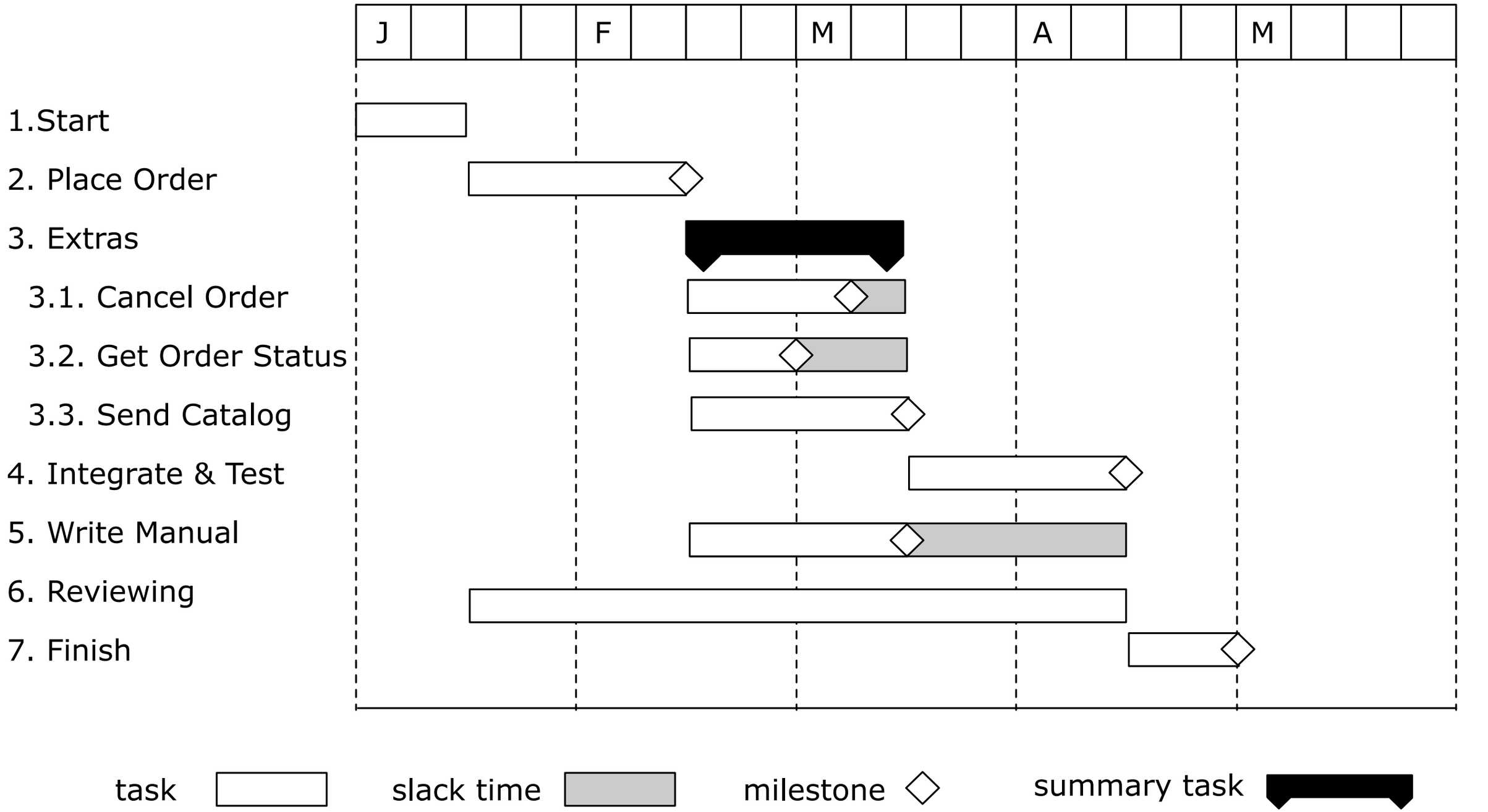
- Good for: Task interdependencies
  - ➡ shows tasks with estimated time
  - ➡ links task that depend on each other
    (depend = cannot start before other task is completed)
  - ➡ optimise task parallelism
  - ➡ monitor complex dependencies

- Good for: Critical Path Analysis
  - ➡ calculate for each task: earliest start-date, latest finish-date
    (latest start-date, latest finish-date)
  - ➡ optimise resources allocated to critical path
  - ➡ monitor critical path

- Not for: Time management

(N.B.: PERT = Program Evaluation and Review Technique)

Gantt Chart: Time Management



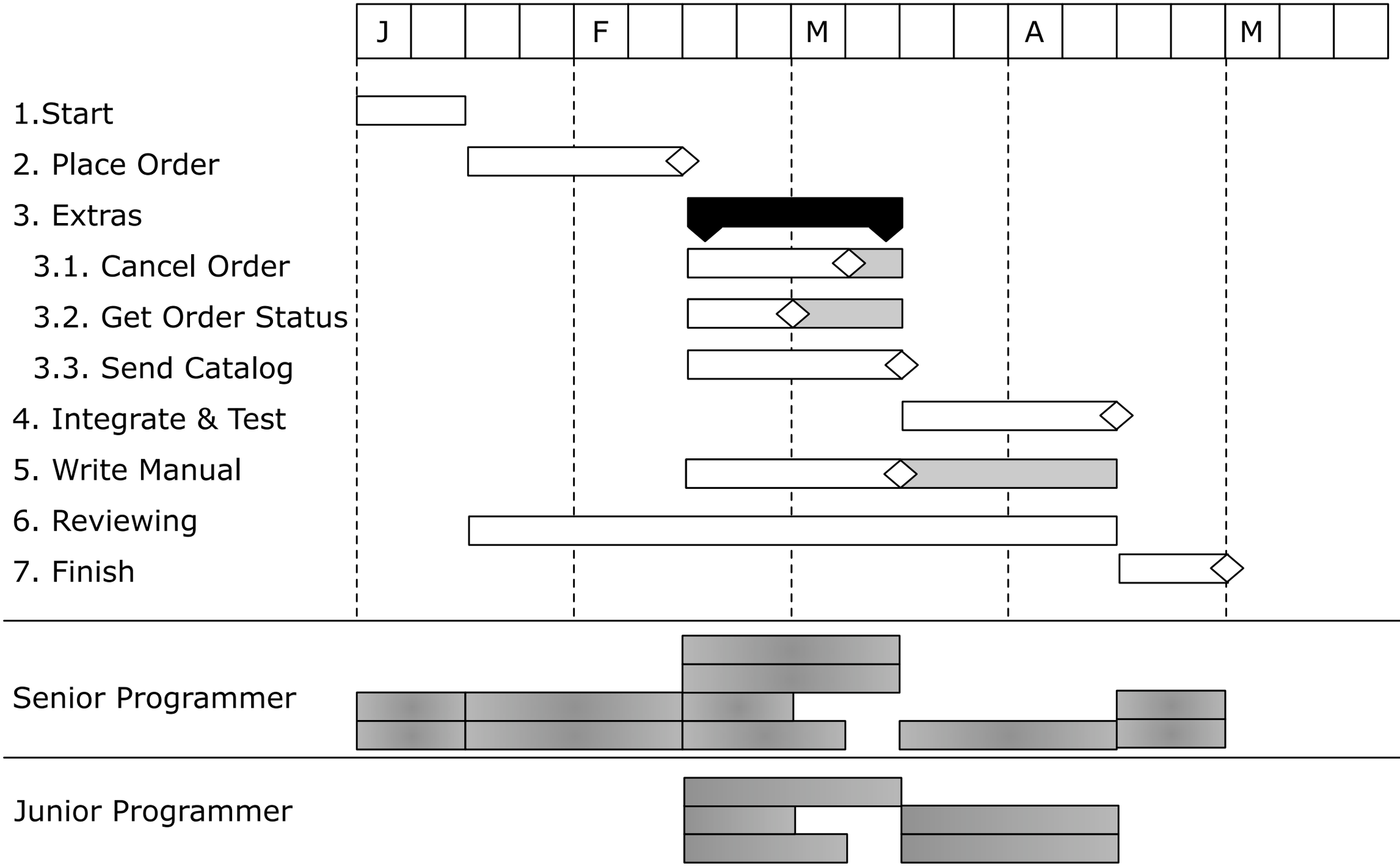
Resource Allocation

For each task, list the required resources.

- Mainly staff (incl. type of skills required)
- ... and special equipment

Activity	Resource	Time	Quantity	Notes
1	Senior Programmer	2 wks	2	Initially senior programmers only
2	Senior Programmer	4 wks	2	
31	Senior Programmer	3 wks	1	
	Junior Programmer	3 wks	1	Implementation: extra junior staff
32	Senior Programmer	2 wks	1	
	Junior Programmer	2 wks	1	
33	Senior Programmer	4 wks	1	
	Junior Programmer	4 wks	1	
4	Senior Programmer	4 wks	1	
	Junior Programmer	4 wks	2	
5	Senior Programmer	4 wks	1	
	Writer	4 wks	1	Manual
6	Quality Engineer	1 day/wk	1	Assistance from QA department
7	Senior Programmer	2 wks	2	

Gantt Chart: Resource Allocation

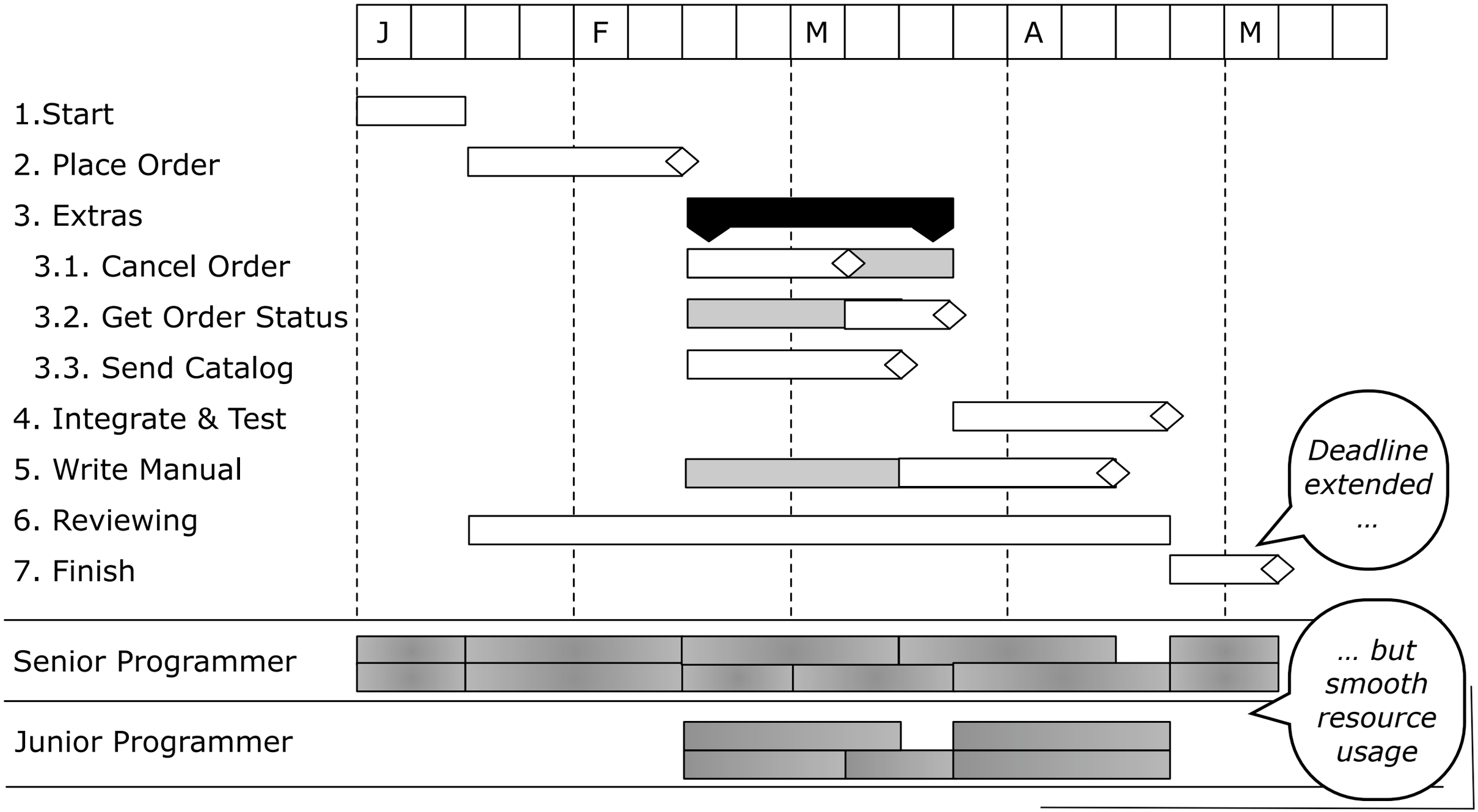


Scheduling tasks at earliest start dates typically gives uneven resource distribution!

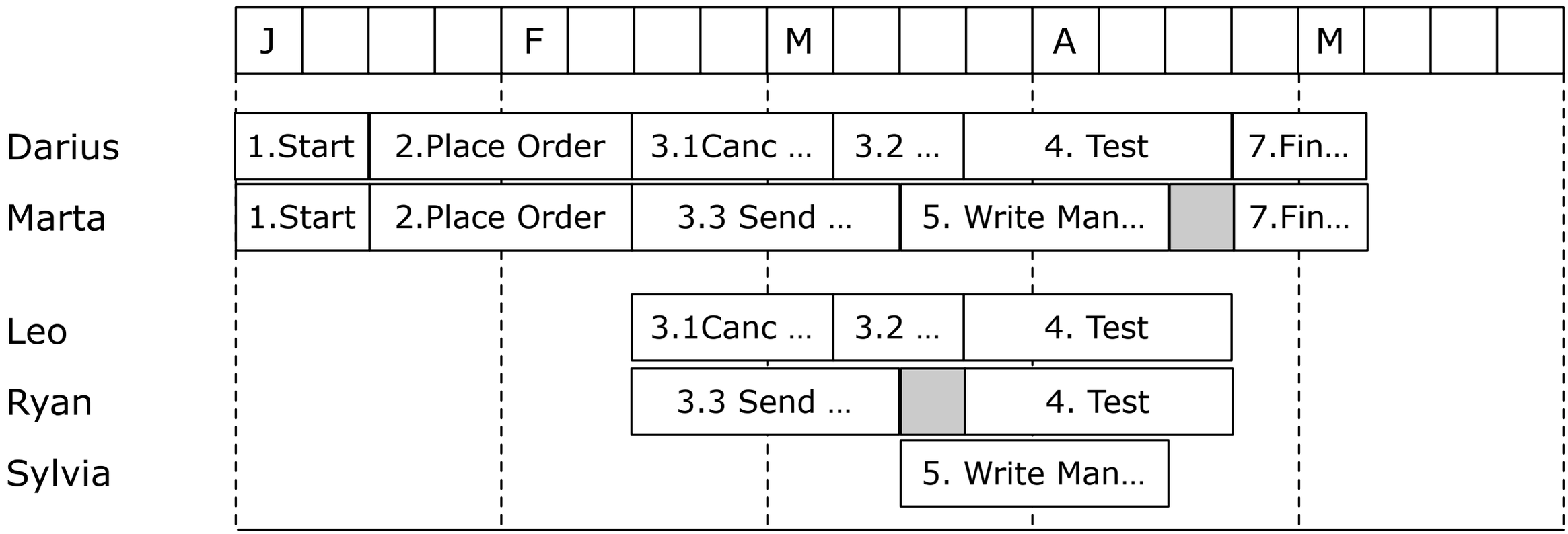
Gantt Chart: Optimized Resources

Shuffle tasks in time to optimise use of resources

- Distribute resources evenly (or with a smooth build-up and run-down)
- May require to extend termination date or to split tasks



Gantt Chart: Staff Allocation



(Overall tasks such as reviewing, reporting, ... are difficult to incorporate)

When to use Gantt Charts ?

- Good for: Time management
  - ➡ shows tasks in time
  - ➡ optimise resources by managing “slack time”
  - ➡ monitor critical tasks (= without slack time)

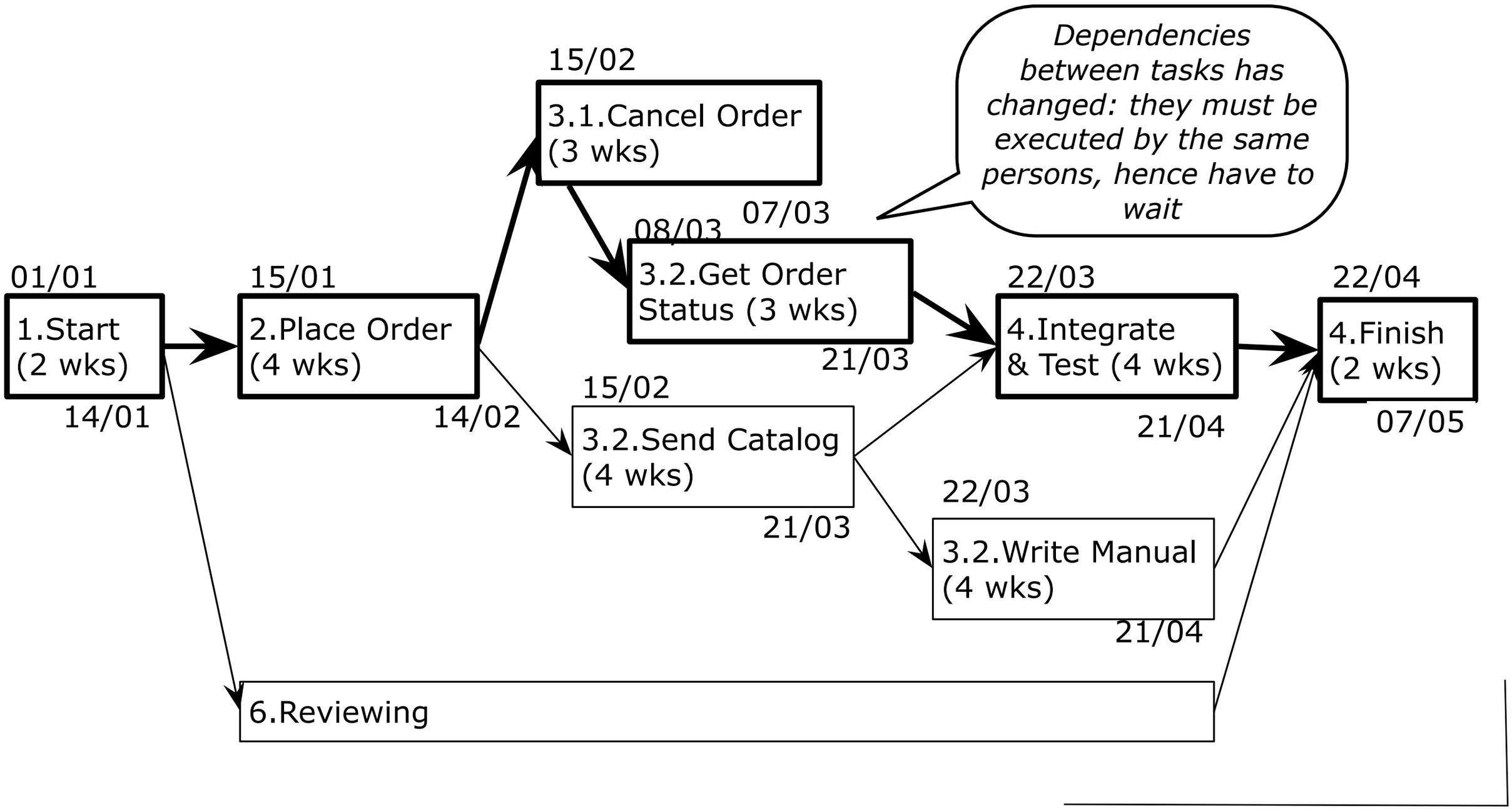
- Good for: Resource and staff allocation
  - ➡ shows resource/staff occupation
  - ➡ optimize “free time” (= time without occupation)
  - ➡ monitor bottlenecks (= fully occupied resources / staff)

- Not for: Task Interdependencies

(N.B. Charts are developed by Henry Gantt; hence the name)

PERT Chart: Including Resources

Due to allocated resources, implicit dependencies are added...
- may give rise to different critical path
- may break "encapsulation" between groups of project tasks



Uncertainty

Planning under uncertainty

- State clearly what you know and don't know
- State clearly what you will do to eliminate unknowns
- Make sure that all early milestones can be met

➡ However: tackle critical risks early

Get commitment

- from main parties involved, incl. management
- The difference between "involvement" and "commitment" is like an eggs-and-ham breakfast: the chicken was involved; the pig is committed.[Anonymous]

Build confidence

- within the team
- with the customer

➡ ... re-planning will not be considered harmful

(See [Gold95a])

A software project is like skiing down a black piste.
The ultimate goal is clear: getting down in one piece.
The way to reach the goal? ... One turn at a time.

Knowns & Unknowns

[This is terminology used for planning military campaigns.]

Known knowns

- = the things you know you know
  You can safely make assumptions here during planning

Known unknowns

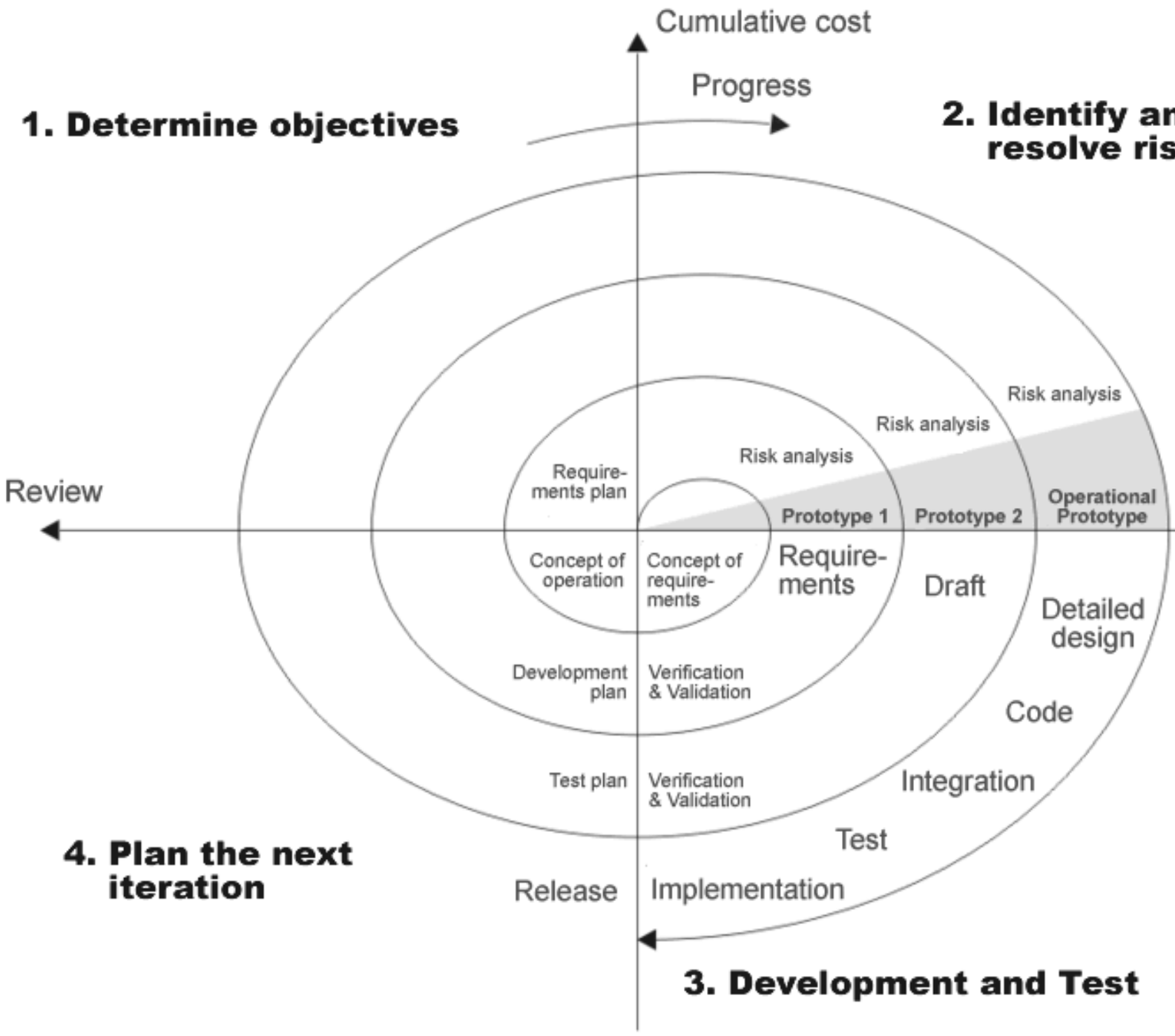
- = the things you know, you don’t know
  You can prepare for these during planning

Unknown unknowns

- = the things you do not know, you don’t know
  These you cannot prepare for during planning
  … the best you can do is being aware and spot opportunities
  + do a thorough *risk analysis*

- software projects (compared to other engineering projects) have lots of “unknown unknowns”
  + Not constrained by physical laws
  + Many stakeholders ⇒ strong political forces around project

Risk Analyis: Quantify Risks for Delays



Quantify Risk: Calculate the risk to the global schedule from the risk on the individual tasks

go, no-go decision

Calculating Risks to the Schedule

(This calculation is an advanced but crucially important part of PERT)

Estimate Task Time

- For each task n, estimate
  + likely time LT(n), optimistic time OT(n), pessimistic time PT(n)
  + deduce estimated time ET(n) := (OT(n) + 4 * LT(n) + PT(n)) / 6

Calculate Standard Deviation per Task

- For each task, calculate the degree of uncertainty for the task time
- standard deviation S(n) := (PT(n) - OT(n)) / 6

Forward Pass: Calculate Standard Deviation per Path

- For each path, calculate the degree of uncertainty for the path execution time
  - ➡ Paths with a high deviation are likely to slip.
- For each task n: compute standard deviation per path
  - ➡ = Maximum of all standard deviations for incoming paths
  - ➡ SP (n) := maximum of (√(Σ S(mi)2)) [where mi is a node in the path to n]

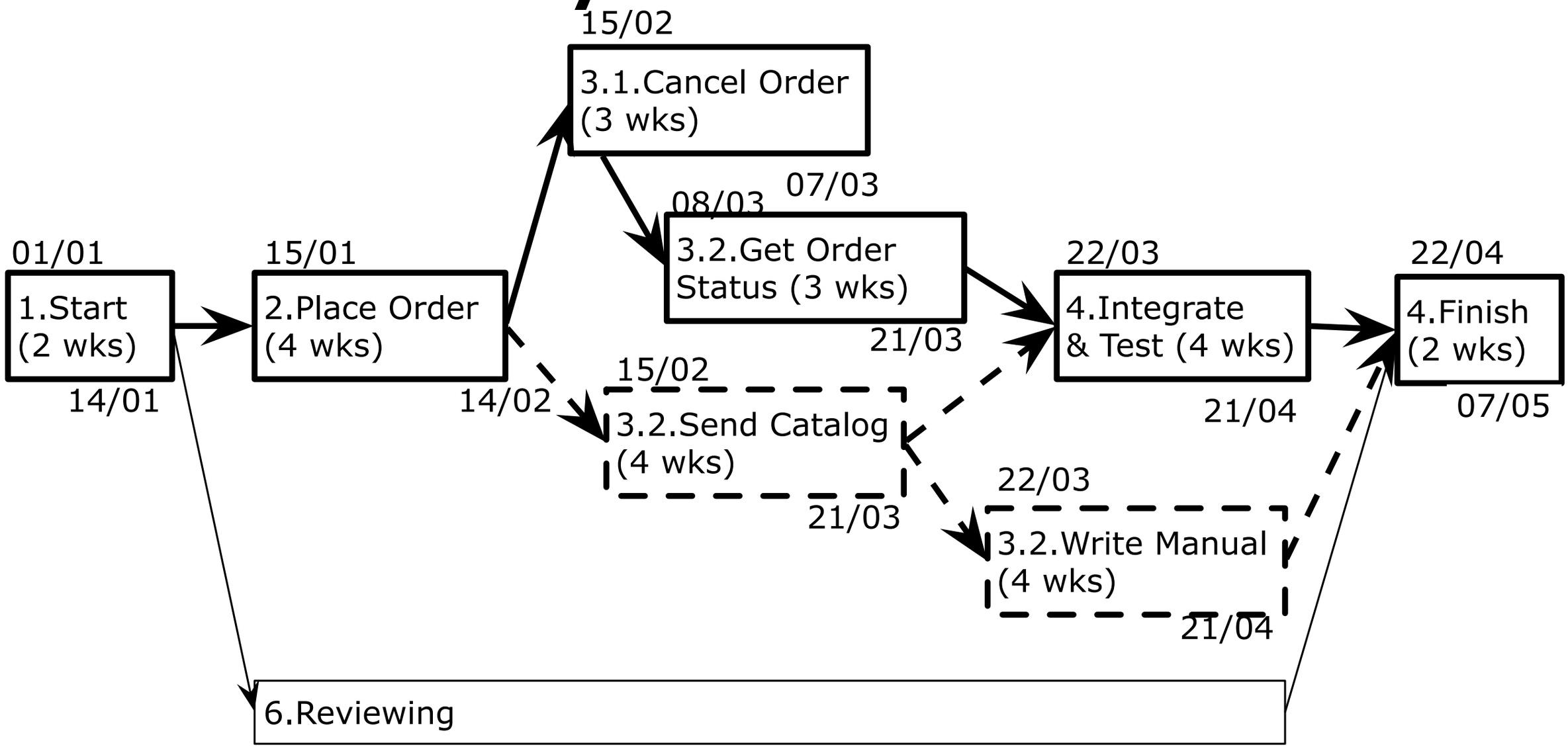
$$\sqrt{\sum_{i=1}^{n} \mathrm{S}(m_i)^2}$$

Example: Calculating Risk

	OT	LT	PT	ET	S	SP is maximum of columns		
1.Start	2	2	2	2,00	0,00	0,00		
					path	1		
2.Place O.	3	4	5	4,00	0,33	0,33		
					path	1,2		
3.1.Cancel	2	3	4	300	0,33	0,47		
					path	1,2,3.1		
3.2.Get O.	2	2	3	2,17	0,17	0,50		
					path	1,2,3.1,3.2		
3.3.Send C.	3	4	6	4,17	0,50	0,60		
					path	1,2,3.3		
4.Test	4	4	6	4,33	0,33	0,60	0,69	
					path	1,2,3.1,3.2,4	1,2,3.3,4	
5.Manual	3	4	5	4,00	0,33	0,69		
					path	1,2,3.3,5		
7.Finish	2	2	2	2,00	0,00	0,60	0,69	0,69
					path	1,2,3.1,3.2,4,7	1,2,3.3,4,7	1,2,3.3,5,7

Task 3.3 is riskiest task (interface with legacy database)
⇒ Paths 1,2,3.3,4,7 and 1,2,3.3,5,7 represent largest risk !

PERT Chart: Risky Path



Estimated scenario (use "estimated time" ET instead of "likely time" LT)
- 1,2,3.3,4,7: 0 + 0 + 0,17 + 0,33 + 0 = 0,5 extra weeks

Worst case scenario (use "pessimistic time" PT instead of "likely time" LT)
- 1,2,3.3,4,7: 0 + 1 + 2 + 2 + 0 = 5 extra weeks
- 1,2,3.3,5,7: 0 + 1 + 2 + 1 + 0 = 4 extra weeks

Risk analysis: can the project afford such delays ? Customers decision; if not … no-go !

Delays & Options

- Assume that you have the following two options

Early with big risk for delay	Later with small risk for delay
delivery of project within 4 (four) months … but can be 1 month early … or 4 months late !	delivery of project within 5 (five) months … at maximum 1 week late … or 1 week early.

- What would you choose ?
- What do you think upper management would choose ? (*)

(*) Most managers would choose option 2 !

Delays

Myth:

- "If we get behind schedule, we can add more programmers and catch up."

Reality:

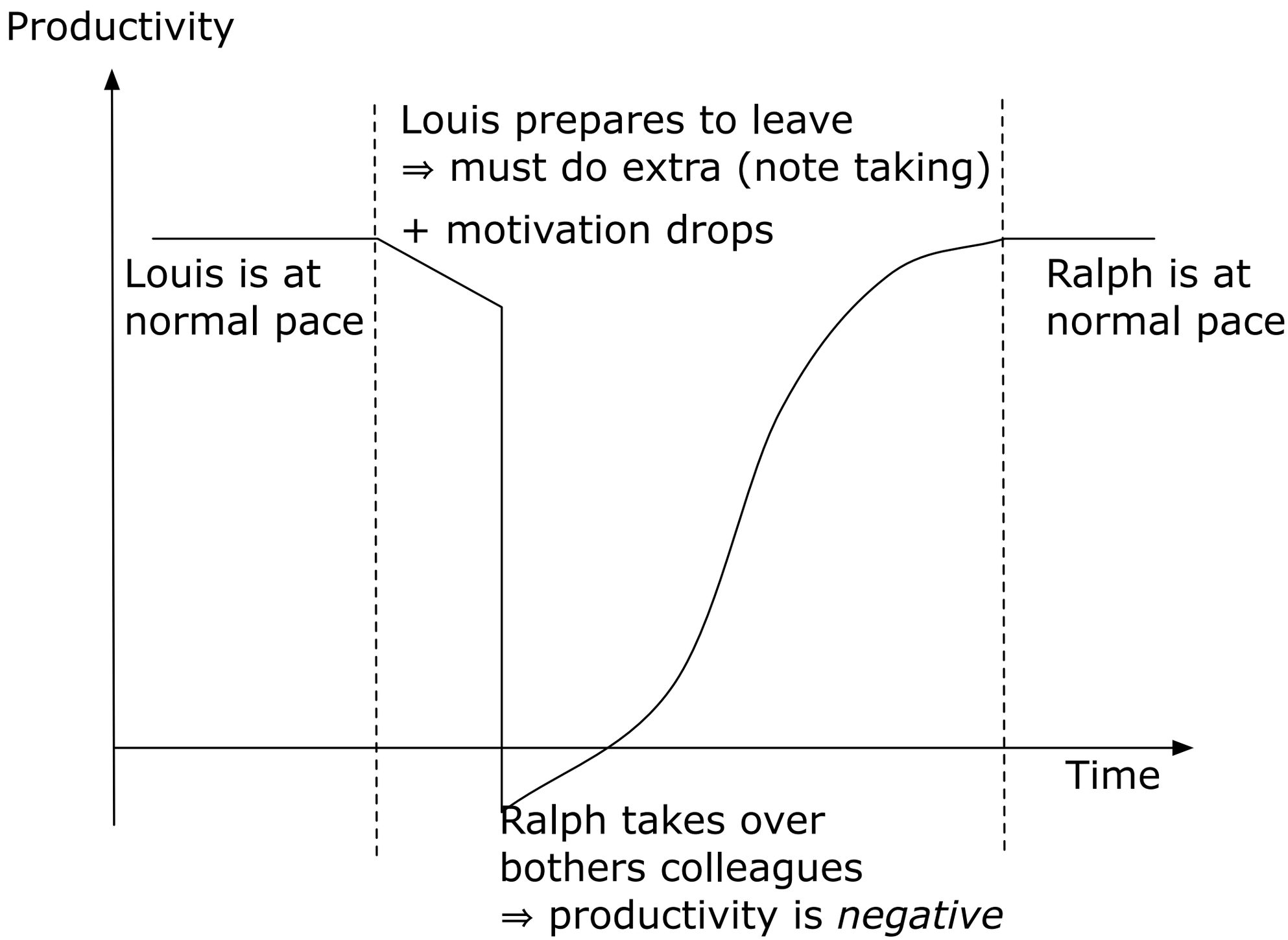
- Adding more people typically slows a project down.

Scheduling Issues

- Estimating the difficulty of problems and the cost of developing a solution is hard
- The unexpected always happens. Always allow contingency in planning
- Productivity is not proportional to the number of people working on a task
  - ➡ Productivity does not depend on raw man-power but on intellectual power
  - ➡ Adding people to a late project makes it later due to communication overhead.
- Cutting back in testing and reviewing is a recipe for disaster
- Working overnight? Only short term benefits …

Cost of Replacing a Person

(See [Dema98a], chapter 13. The Human Capital)



Dealing with Delays

Spot potential delays as soon as possible

- ... then you have more time to recover

How to spot?

- Earned value analysis
  - ➡ planned time is the project budget
  - ➡ time of a completed task is credited to the project budget

How to recover?

- A combination of following 3 actions
  - + Adding senior staff for well-specified tasks
    - ➡ outside critical path to avoid communication overhead
  - + Prioritize requirements and deliver incrementally
    - ➡ deliver most important functionality on time
    - ➡ testing remains a priority (even if customer disagrees)
  - + Extend the deadline

Calculating Earned Value (= Tasks Completed)

The 0/100 Technique

- earned value := 0% when task not completed
- earned value := 100% when task completed
  - ➡ tasks should be rather small
  - ➡ gives a pessimistic impression

The 50/50 Technique

- earned value := 50% when task started
- earned value := 100% when task completed
  - ➡ tasks should be rather large
  - ➡ may give an optimistic impression
  - ➡ variant with 20/80 gives a more realistic impression

The Milestone Technique

- earned value := number of milestones completed / total number of milestones
  - ➡ tasks are large but contain lots of intermediate milestones
  - ➡ Good for summary views on large schedules (otherwise consider to split task in several subtasks and fall back on 0/100)

Calculating Earned Value (= Time sheets)

Organizations usually require staff to maintain time sheets
= bookkeeping of time spent by an individual for a particular task in a project

Time Sheet

Name: Laura Palmer_____ Week ending: March, 3rd 2000_

Rechargeable hours

Project	Task	Activity	Description	Hours	Delay?
C34	5	5.3	Chapter 3	25	-
C34	5	5.4	Chapter 4	5	+
C34	6	6.0	Reviewing 4		

Non-rechargeable hours

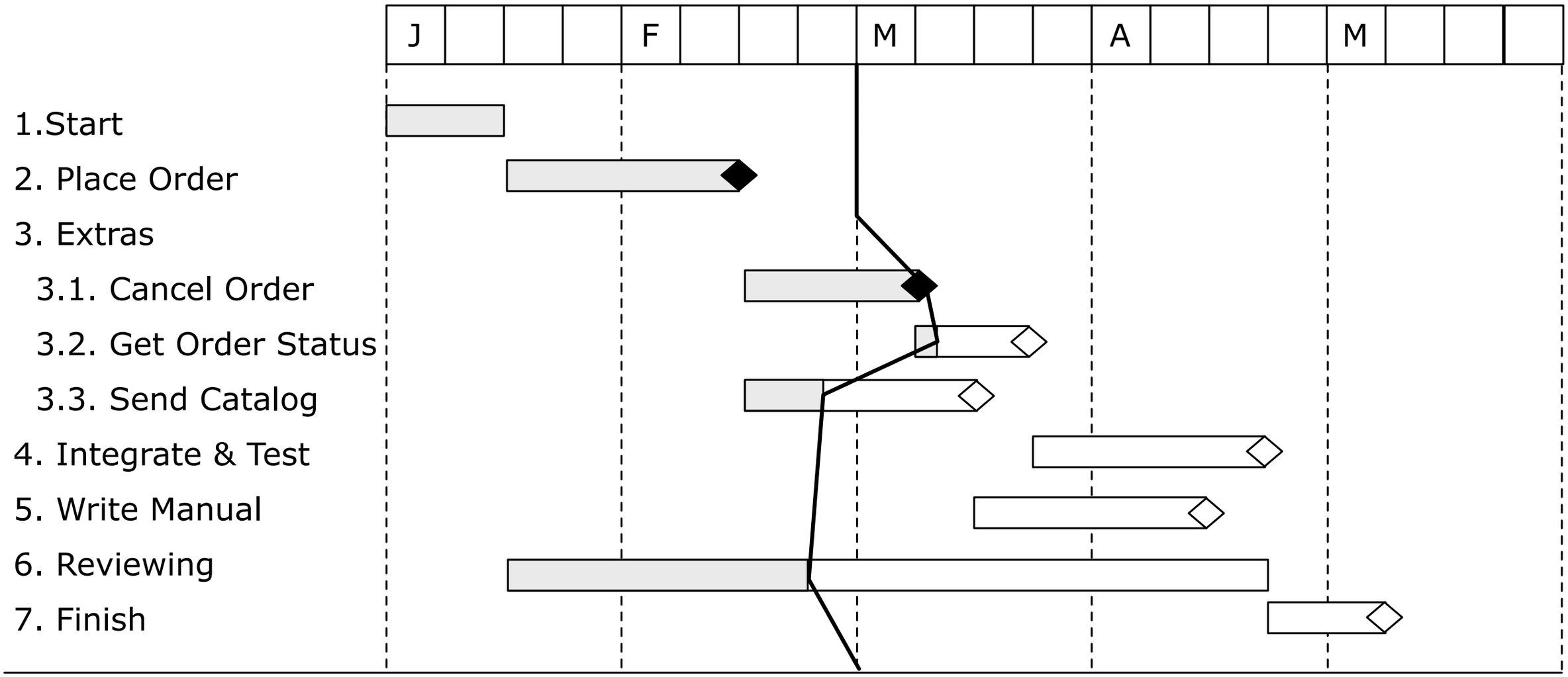
Hour	Description	Authorized
8	Use-case training	J.F. Kennedy

Opportunity to monitor team occupation
- Compare time spent (= earned value) vs. time planned
- Ask staff member if delay for this task is expected

Monitoring Delays – Slip Line (Gantt chart)

Visualise percentage of task completed via shading

- draw a slip line at current date, connecting endpoints of the shaded areas
- bending to the right = ahead of schedule, to the left = behind schedule



Interpretation

Today is 1rst of March

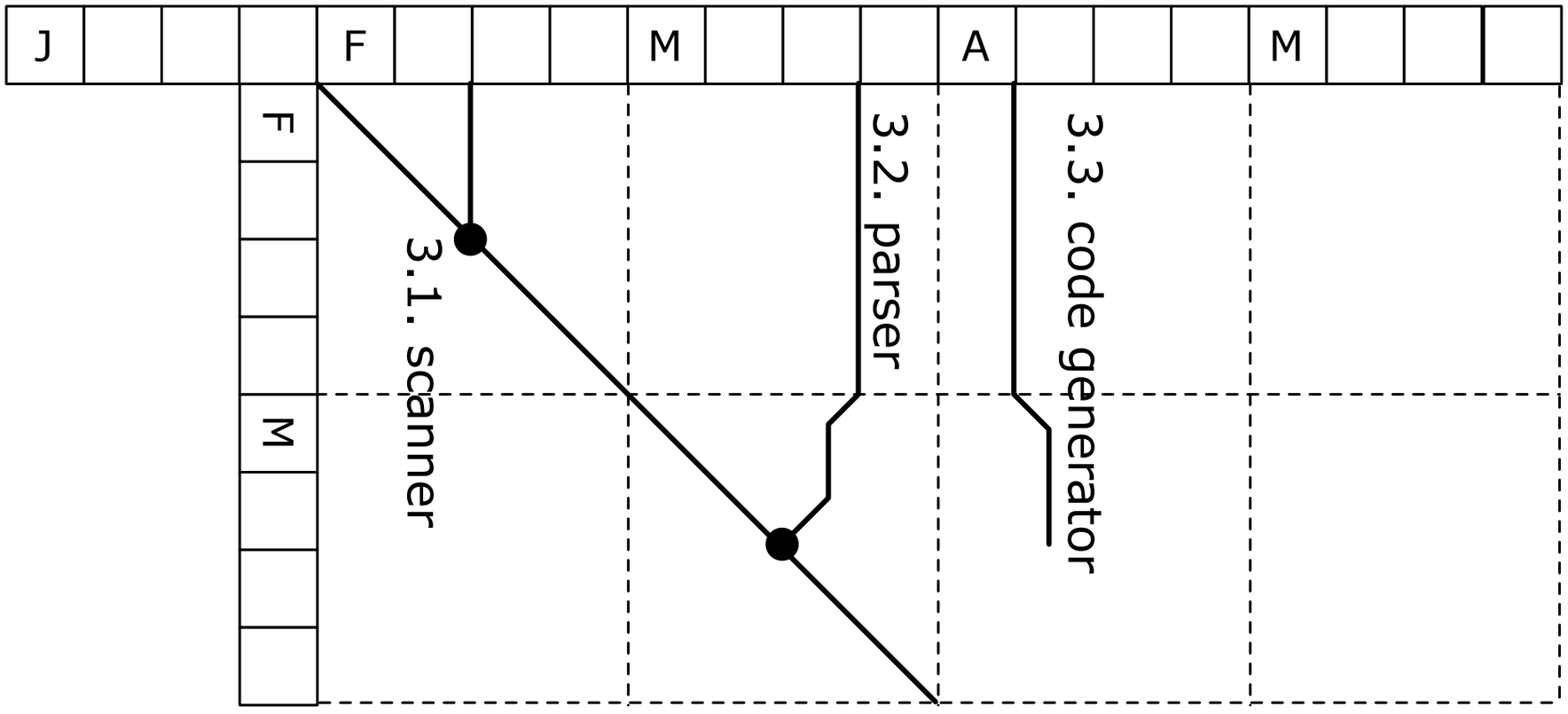
Task 3.1 is finished ahead of schedule and task 3.2 is started ahead of schedule

Tasks 3.3 and 6 seem to be behind schedule (i.e., less completed than planned)

Monitoring Delays – Timeline Chart

Visualise slippage evolution

- downward lines represent planned completion time as they vary in current time
- bullets at the end of a line represent completed tasks



Interpretation (end of October)

Task 3.1 is completed as planned.
Task 3.2 is rescheduled 1/2 wk earlier end of February and finished 1 wk ahead of time.
Tasks 3.3 rescheduled with one week delay at the and of February

Slip Line vs. Timeline

Slip Line

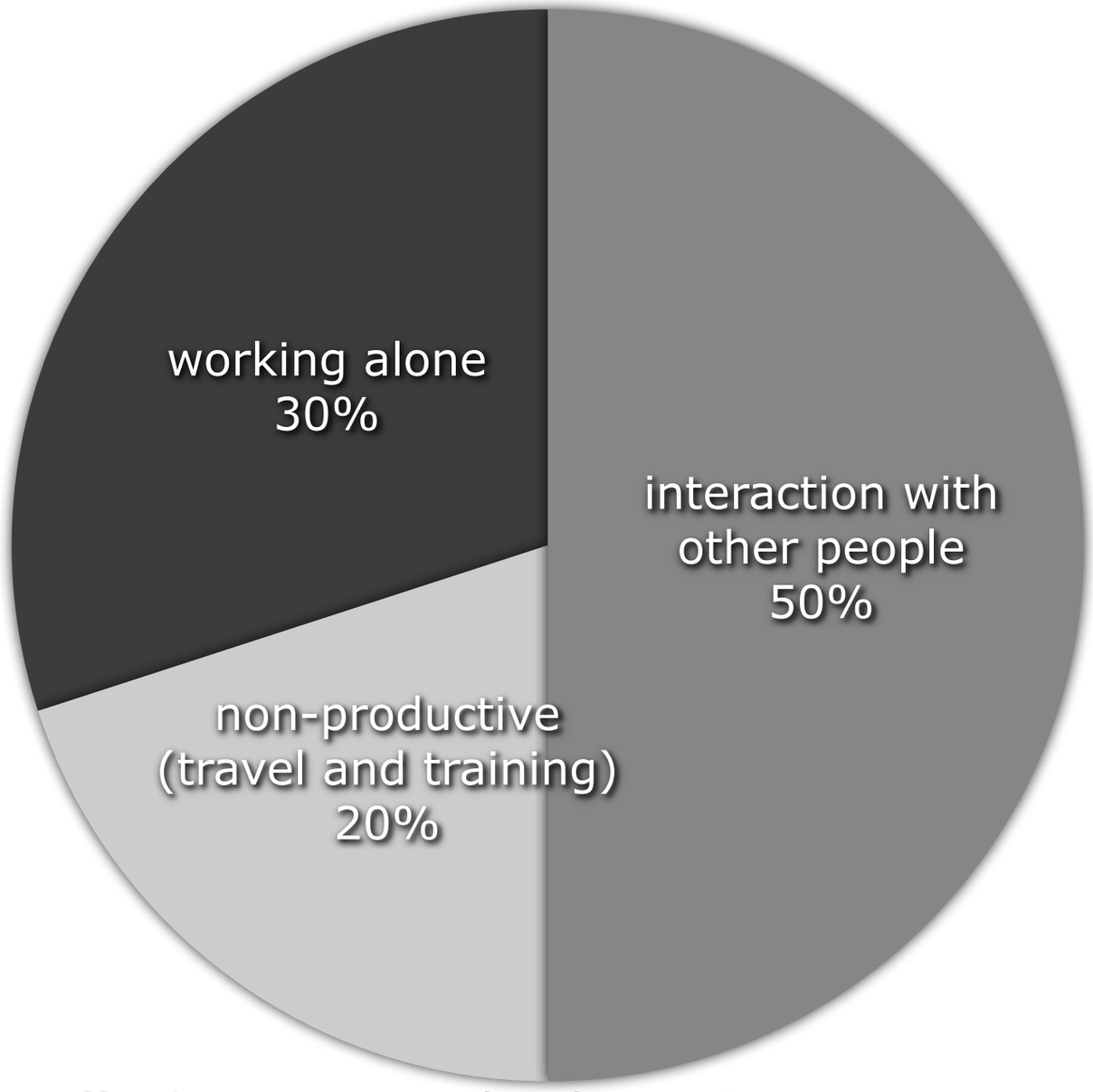
- Monitors current slip status of project tasks
  - \+ many tasks
  - \+ only for 1 point in time
    - \- include a few slip lines from the past to illustrate evolution

Timeline

- Monitors how the slip status of project tasks evolves
  - \+ few tasks
    - \- crossing lines quickly clutter the figure
    - \- colors can be used to show more tasks
  - \+ complete time scale

Individuals work in Teams

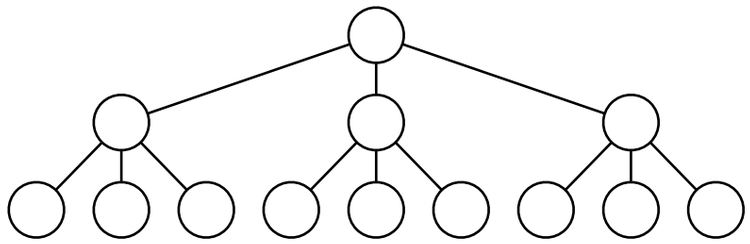
Distribution of a software engineer's time, as logged within IBM [McCu78a]



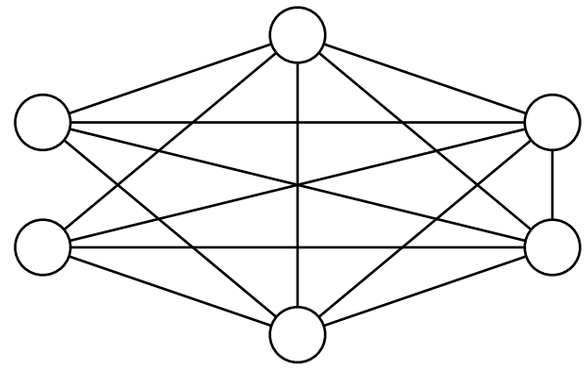
- You cannot afford too many solo-players in a team
- Complementary personalities are as important as technical skills
- More women are necessary

Typical Team Structures

Hierarchical (Centralized)
e.g. Chief Programmer
- For well-understood problems
- Predictable, fast development
- Large groups



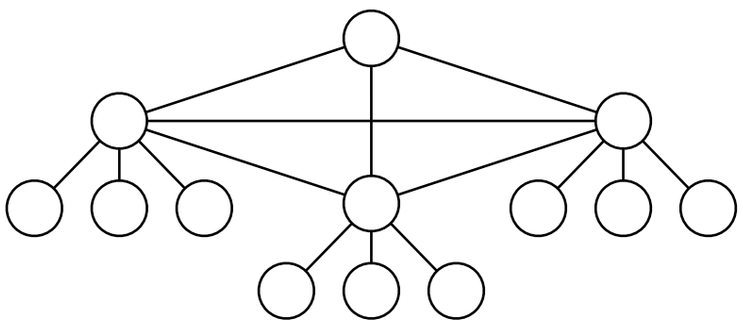
Consensus (Decentralized)
e.g. Egoless Programming Team
- For exploratory projects
- Fast knowledge transfer
- Small groups



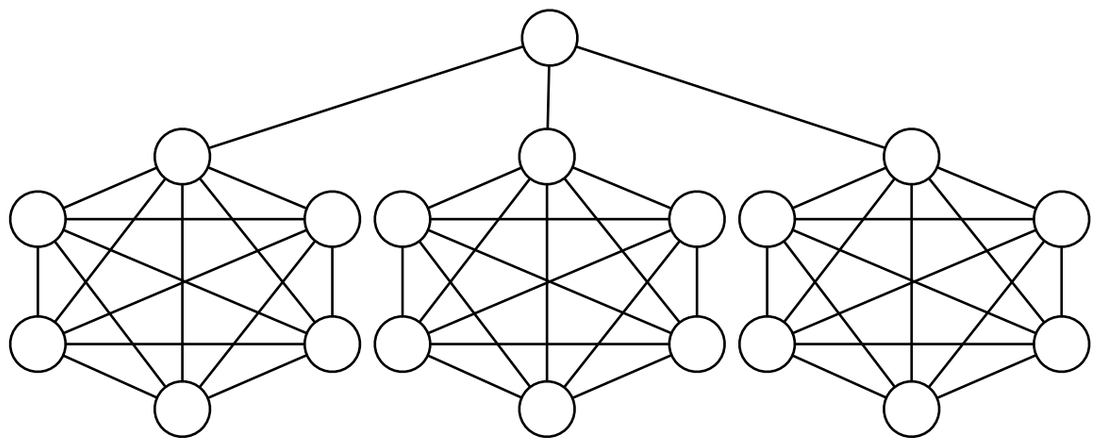
There is no “one size fits all” team structure!

Organize so that no one person has to talk to more then 8 (eight) persons in total !

Decentralized upper management
+ Centralized teams



Centralized upper management
+ Decentralized teams



Directing Teams

Directing a team = the whole becomes more then the sum of its parts

Managers serve their team

- Managers ensure that team has the necessary information and resources
  - ➡ incl. pizza!
- Responsibility demands authority
  - + Managers must delegate
    - ➡ Trust your own people and they will trust you.
- Managers manage
  - + Managers cannot perform tasks on the critical path
    - ➡ Especially difficult for technical managers

- Developers control deadlines
  - + A manager cannot meet a deadline to which the developers have not agreed

Conclusion: Correctness & Traceability

Correctness

- The purpose of a plan is not correctness.
  - + The purpose is to detect deviations as soon as possible
    - ... and take appropriate actions
      - ➡ Adding people to a late project makes it later

- Are we building the system right?
  - + Deliver what's required
    - ➡ ... on time within budget

Traceability

- Plan ⇔ Requirements & System?
  - + Only when done well
    - ➡ small tasks
    - ➡ milestones verifiable by customer

Summary (i)

You should know the answers to these questions

- Why is it necessary to define tasks small?
- What is a milestone? What can you use them for?
- What is a critical path? Why is it important to know the critical path?
- What can you do to recover from delays on the critical path?
- How can you use Gantt-charts to optimize the allocation of resources to a project?
- What is a “Known kown”, and “Unknown known” and an “Unknown Unknown” ?
- How do you use PERT to calculate the risk of delays to a project?
- Why is it necessary to apply earned value analysis during project management?
- Why does replacing a person imply a negative productivity?
- What’s the difference between the 0/100; the 50/50 and the milestone technique for calculating the earned value ?
- Why shouldn’t managers take on tasks in the critical path?
- How can you ensure traceability between the plan and the requirements/system ?

You should be able to complete the following tasks

- draw a PERT Chart, incl. calculating the critical path and the risk of delays
- draw a Gant chart, incl. allocating and optimizing of resources
- draw a slip line and a timeline

Summary (ii)

Can you answer the following questions?

- Name the various activities covered by project managament. Which ones do you consider most important? Why?
- Compare PERT-charts with Gantt charts for project planning and monitoring.
- How can you deal with “Unknown Unknows” during project planning ?
- Choose between managing a project that is expected to deliver soon but with a large risk for delays, or managing a project with the same result delivered late but with allmost no risk for delays. Can you argue your choice ?
- Describe how earned-value analysis can help you for project monitoring.
- Would you consider bending slip lines as a good sign or a bad sign? Why?
- You’re a project leader and one of your best team members announces that she is pregnant. You’re going to your boss, asking for a replacement and for an extension of the project deadline. How would you argue the latter request?
- You have to manage a project team of 5 persons for building a C++ compiler. Which team structure and member roles would you choose? Why?