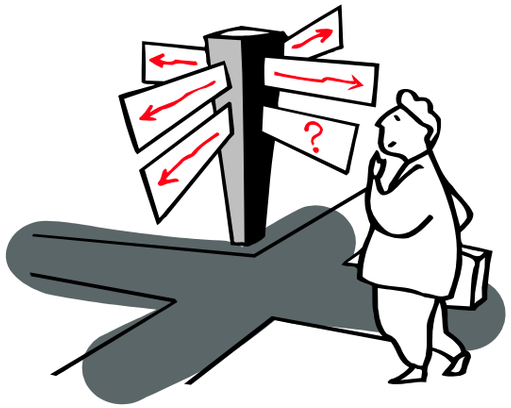
CHAPTER 4 – Project Management



- Introduction
  + When, Why and What?
- Planning & Monitoring
  + PERT charts
  + Gantt charts
  + Uncertainty
    ⇒ Risk to the schedule
  + Dealing with delays
  + Monitoring: earned value analysis
    - Tasks completed, Time sheets
    - Slip Lines, Timelines
  + An afterthought: late projects
    … started late
- Organisation, Staffing, Directing
  + Belbin Roles
  + *Myers Briggs Type Inventory*
  + Team Structures
  + Directing Teams
- Scrum
  + Definition of Done
  + Scaling Scrum
- Conclusion
  + Correctness & Traceability

Literature

+ [Ghez02] In particular, “Management of Software Engineering”
+ [Pres00] In particular, “Software Project Planning” & “Project Scheduling and Tracking”
+ [Somm05] In particular, “Project Planning” & “Managing People”

- Other
  + [Hugh99] Software Project Management, B. Hughes and M. Cotterell, McGraw Hill, 1999.
    * Good practical examples on PERT, Gantt, Time-sheets, ...

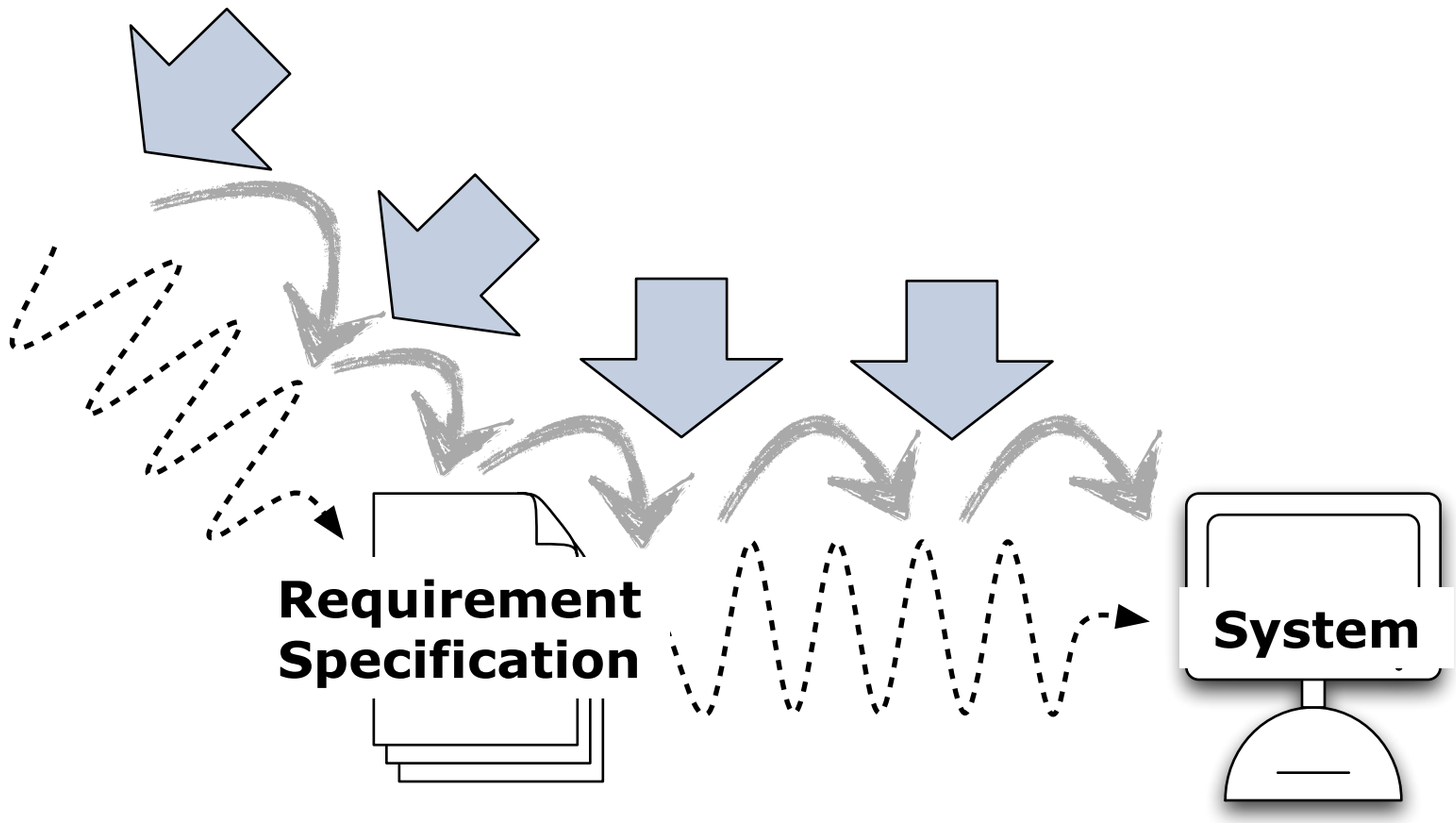
Literature - Papers

- [Henr99] Sallie M. Henry, K. Todd Stevens “Using Belbin's leadership role to improve team effectiveness: An empirical investigation.” ,Journal of Systems and Software, Volume 44, Issue 3, January 1999, Pages 241-250, ISSN 0164-1212.
  + Demonstrating that Belbin roles do make a difference in team efficiency, even for student projects

- [Dema11] Tom De Marco"All Late Projects Are the Same," IEEE Software, pp. 102-103, November/December, 2011
  + All projects that finish late have this one thing in common: they started late.

- [Yoge21] Yogeshwar Shastri, Rashina Hoda, Robert Amor “The role of the project manager in agile software development projects.” Journal of Systems and Software, Volume 173, 2021, 110871. https://doi.org/10.1016/j.jss.2020.110871.
  + Agile projects shouldn’t have project managers … or not?

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)

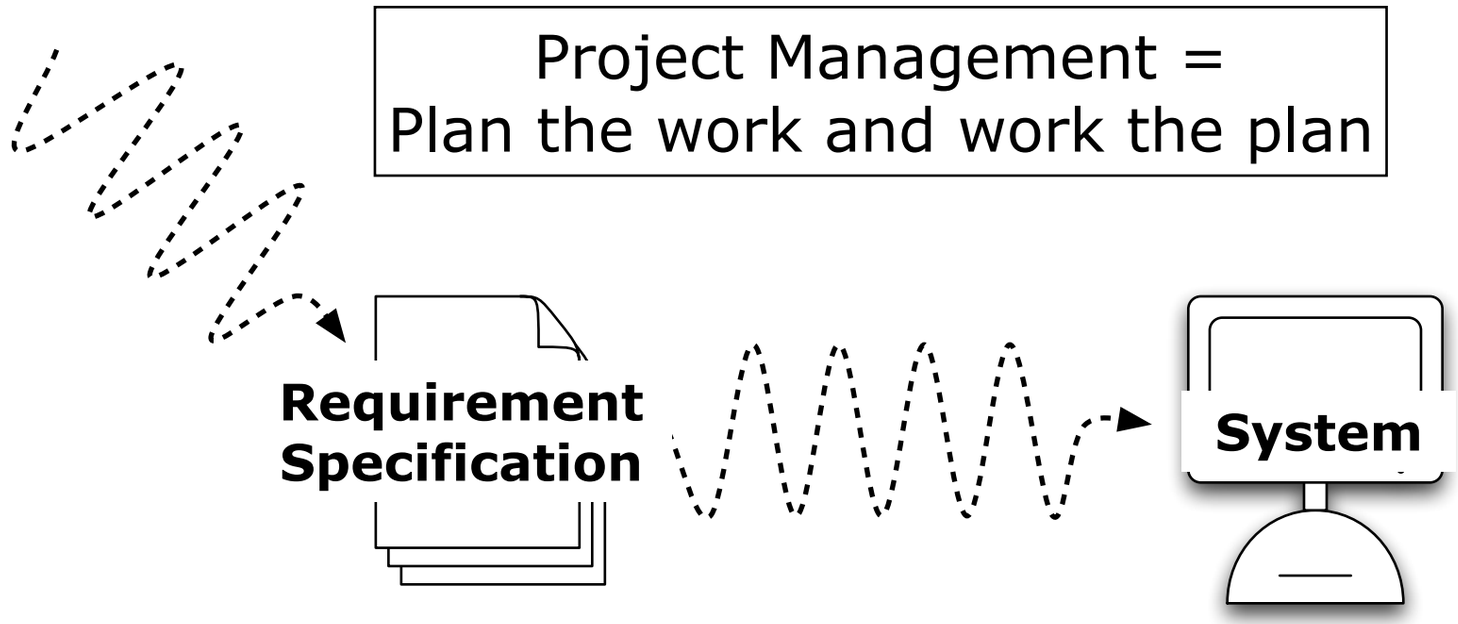
Software Project = <u>Deliver</u> on <u>time</u> and within <u>budget</u>

Achieve interdependent & conflicting goals ...

... with limited resources.

Your project team is a resource!

What is Project Management?



Management Functions

- Planning: 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.)

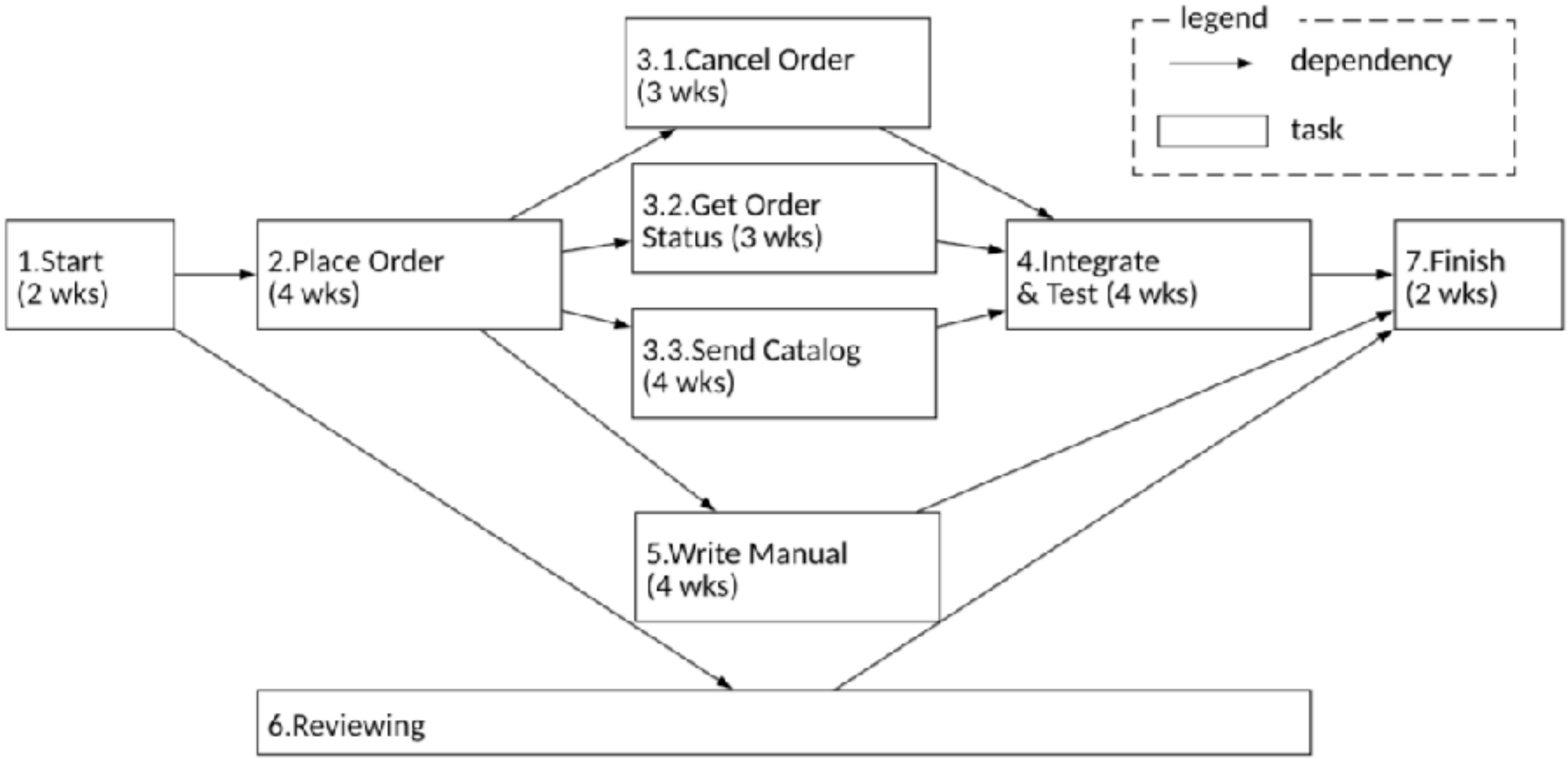
Tasks & Milestones

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 task duration
  - 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

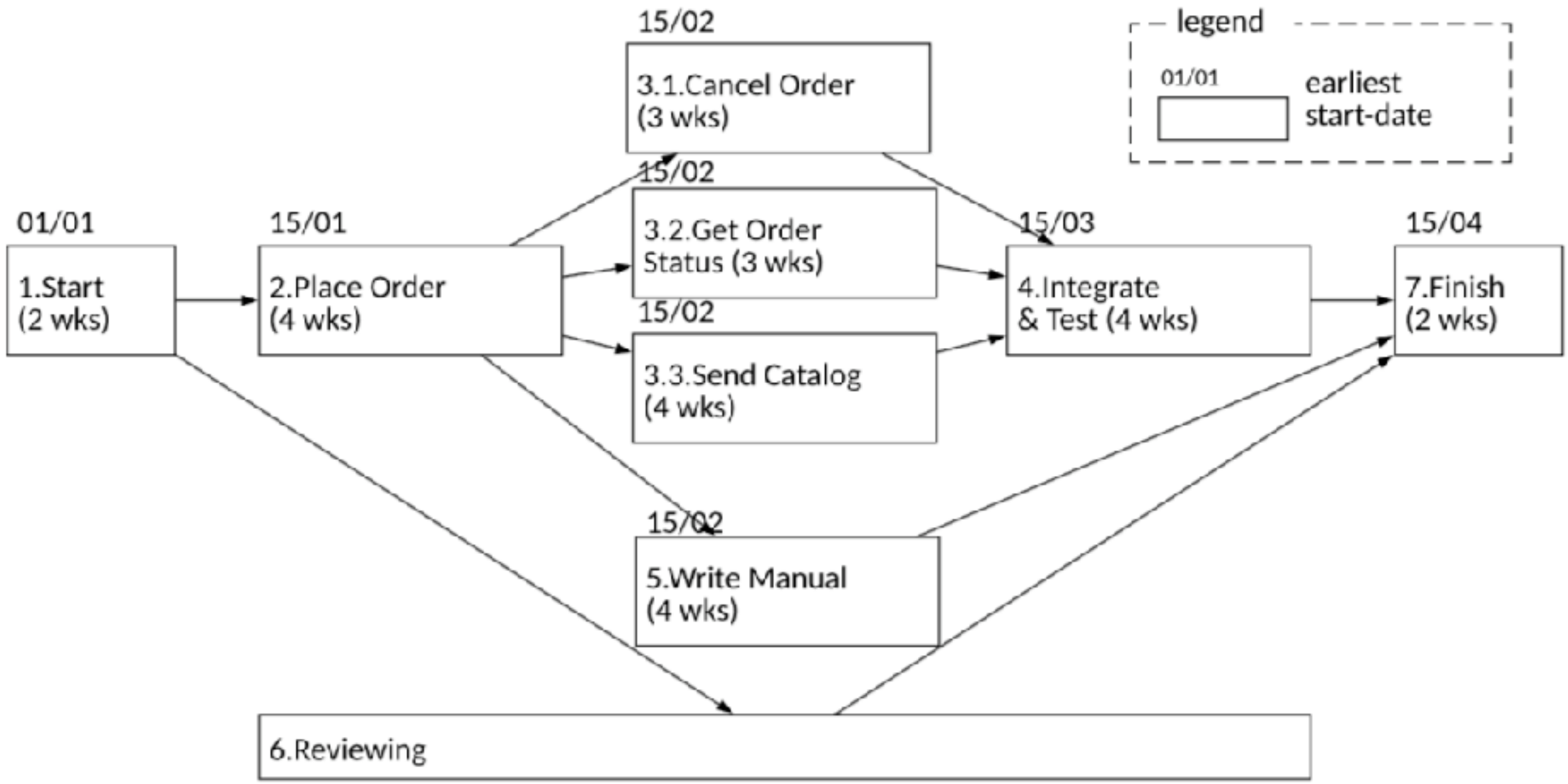
Remember: small tasks & milestones verifiable by customer!

Finding the Critical Path

Revised
(Replaced node with task)

- Forward Pass: compute "earliest start-date" (ESD)
  > ESD (start-task) := start-date project
  + Breadth-first enumeration (use node numbering)
  + For each task: compute earliest start-date
    = Latest of all incoming paths
    > ESD (task) := latest of (
      ESD (preceding task) + estimated task duration (preceding task))
- Backward Pass: compute "latest end-date" (LED)
  > LED (end-task) := ESD (end-task) + estimated task duration
  + Breadth-first enumeration (node numbering in reverse order)
  + For each task: compute latest end-date
    = Earliest of all outgoing paths
    > LED (task):= earliest of (
      LED (subsequent task) - estimated task duration (subsequent task))
- Critical Path
  + = path where delay in one task will cause a delay for the whole project
  + path where for each task:
    > ESD(task) + estimated time (task)= LED(task)

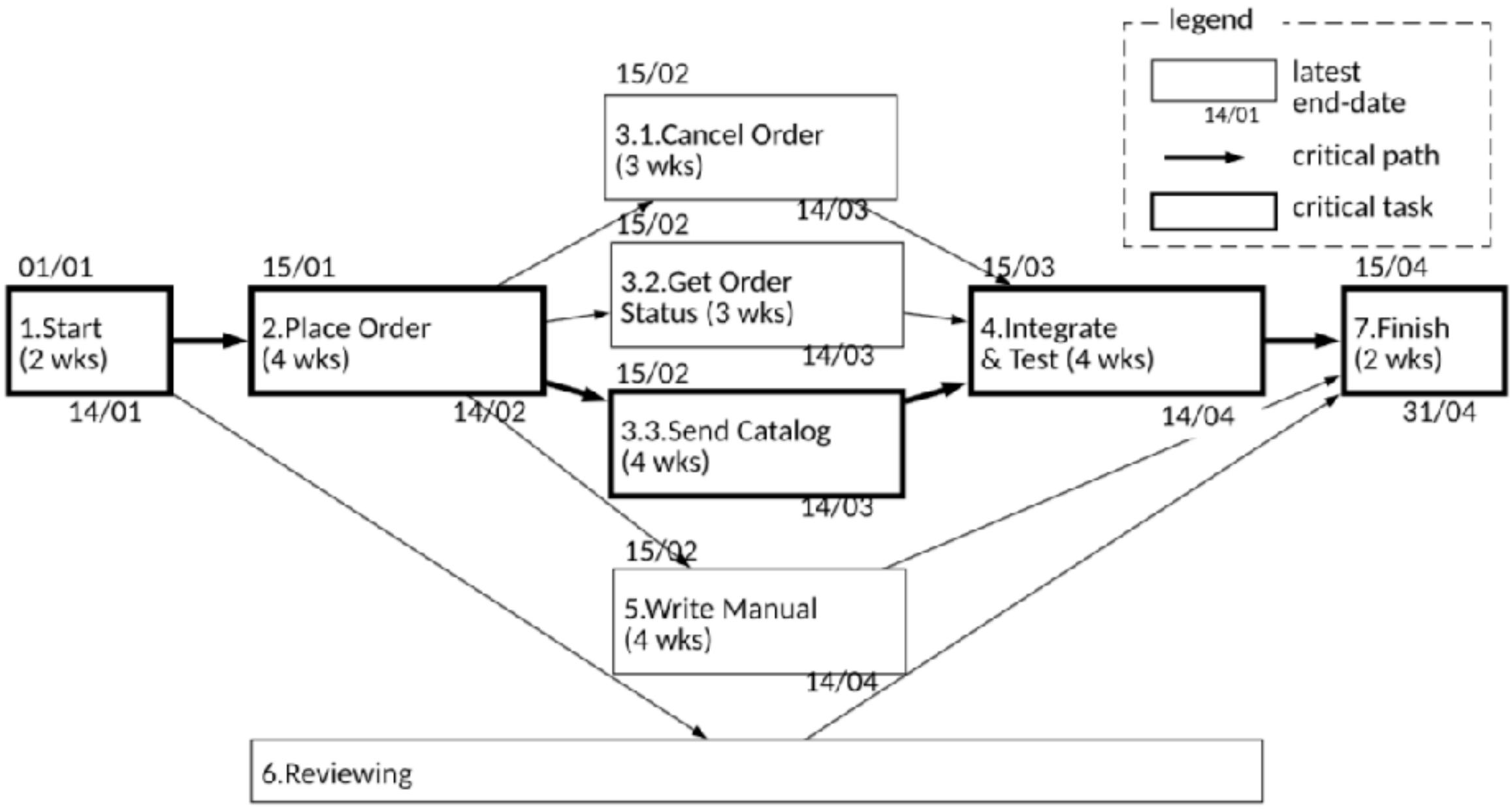
PERT Chart: Forward pass



This is a schedule with coarse grained granularity: 1 month is 4 weeks of 7 days (week 1 = 1-7; week 2 = 8-15; …)

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) - 3 wks, LED(3.3) - 4 wks) := 14/02

When to use PERT Charts?

- Good for: Task interdependencies
  + shows tasks with estimated task duration
  + 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

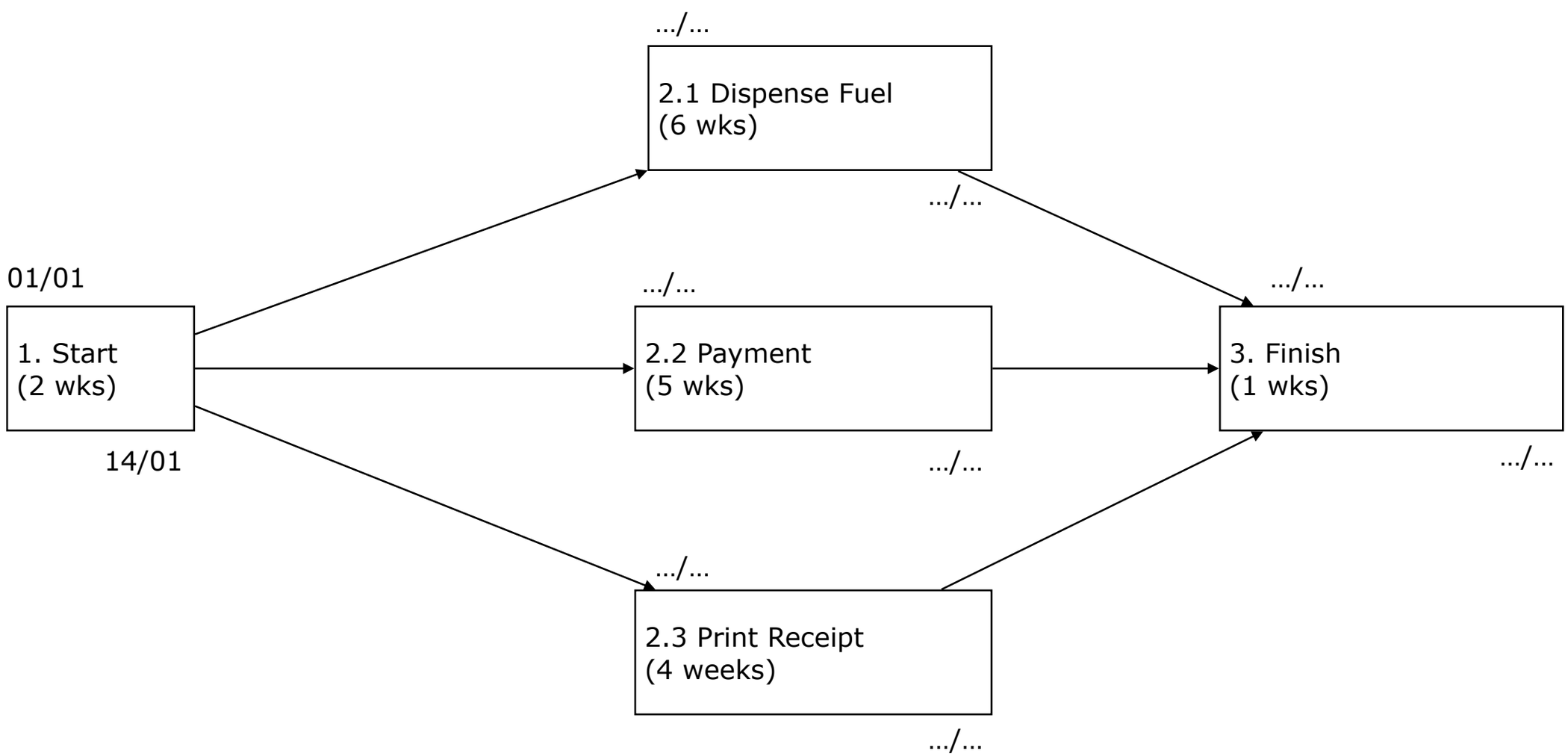
- Not for: Time management

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

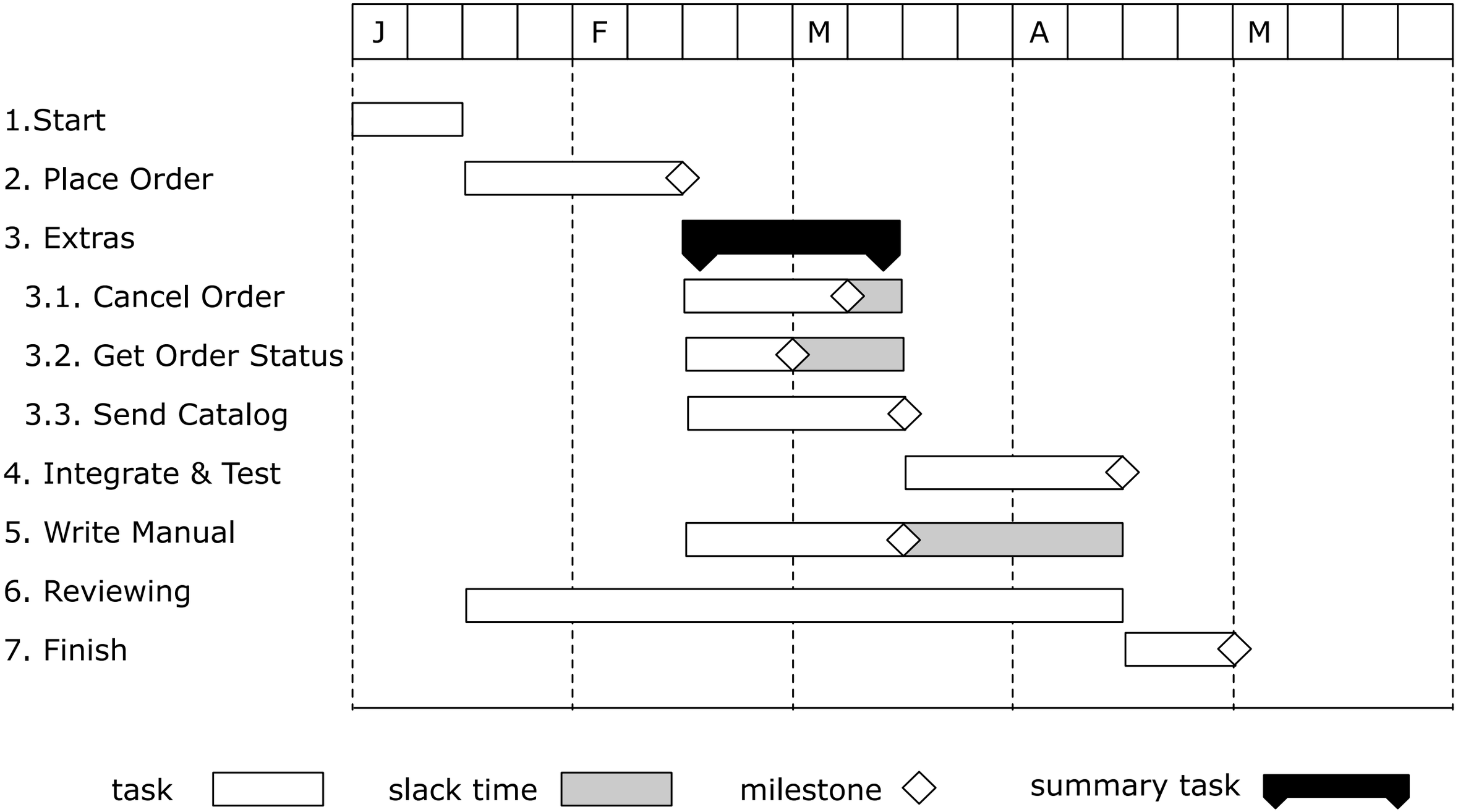
Critical Path (exercise)

Identify the critical path

- Forward pass: earliest start data
- Backward pass: latest end date



Gantt Chart: Time Management



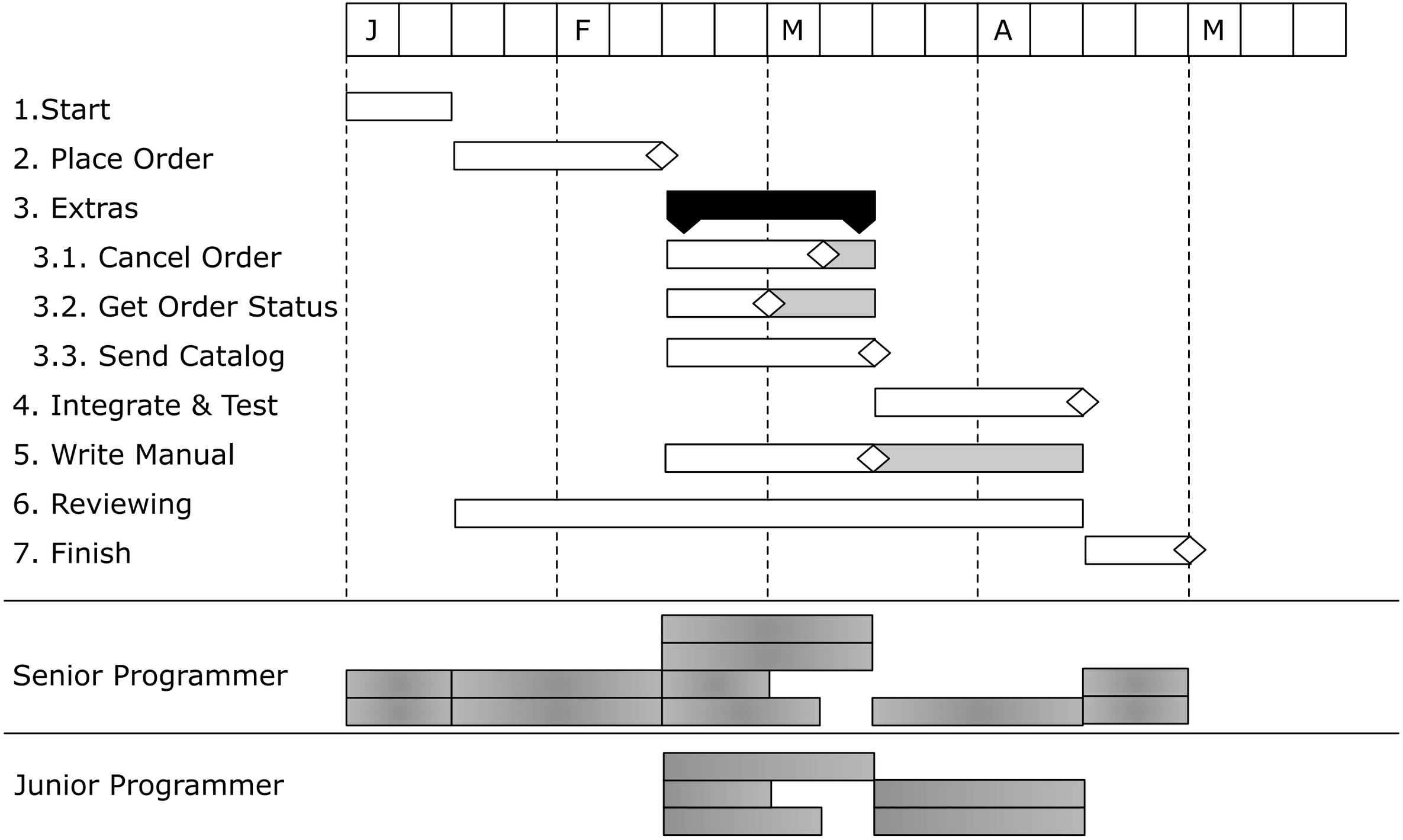
Resource Allocation

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	
3.1	Senior Programmer	3 wks	1	
	Junior Programmer	3 wks	1	Implementation: extra junior staff
3.2	Senior Programmer	2 wks	1	
	Junior Programmer	2 wks	1	
3.3	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

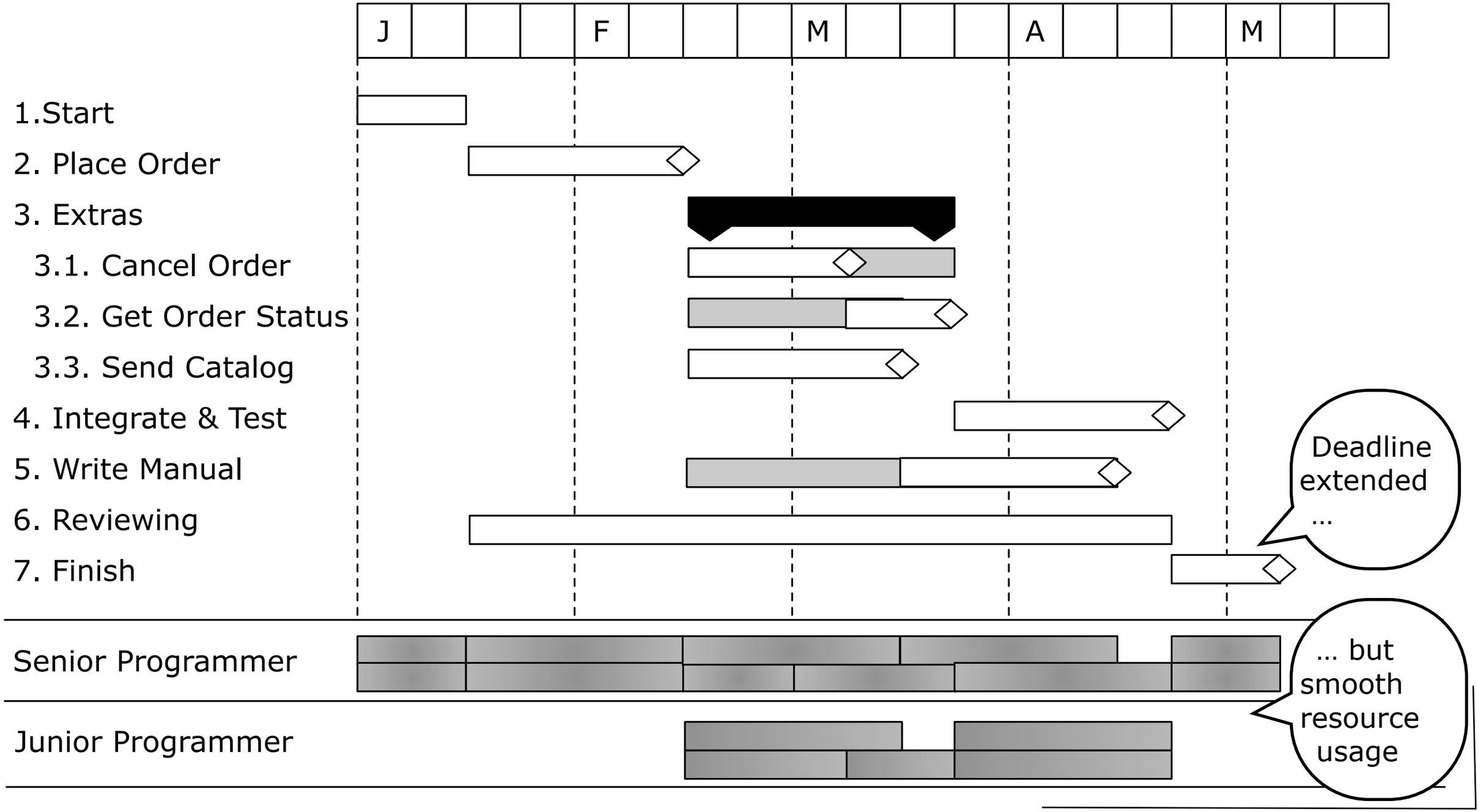


Scheduling tasks at earliest start dates typically gives uneven resource distribution!

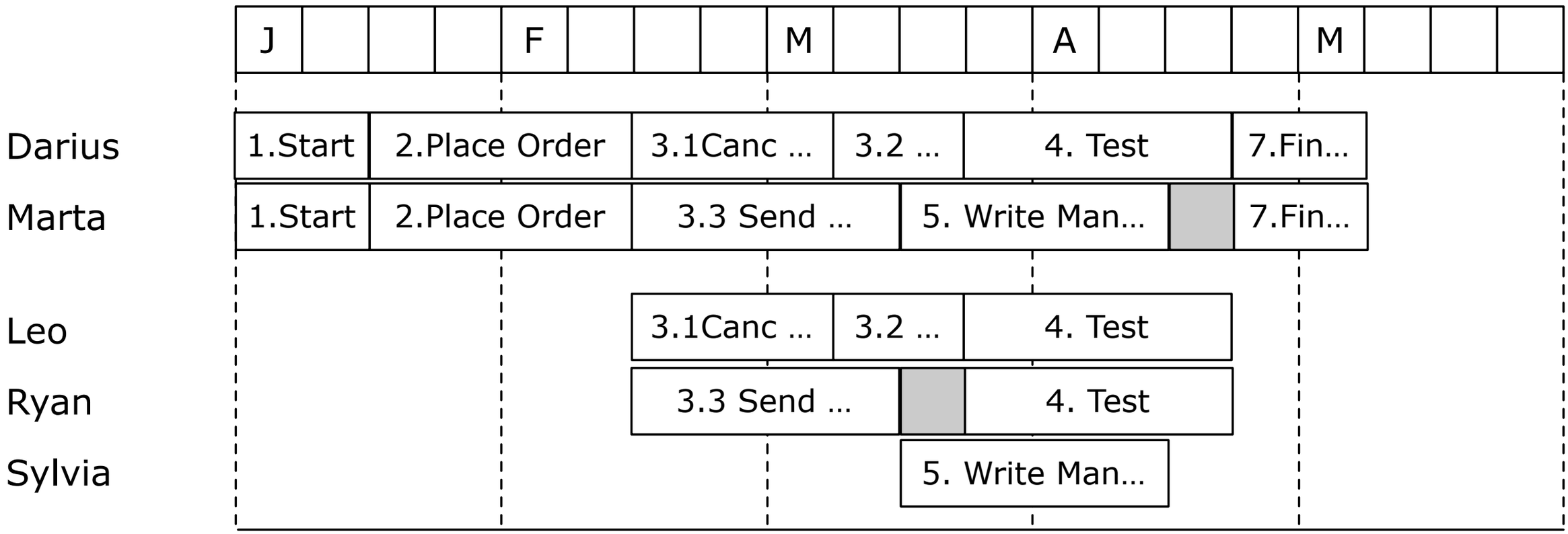
Gantt Chart: Optimized Resources

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)

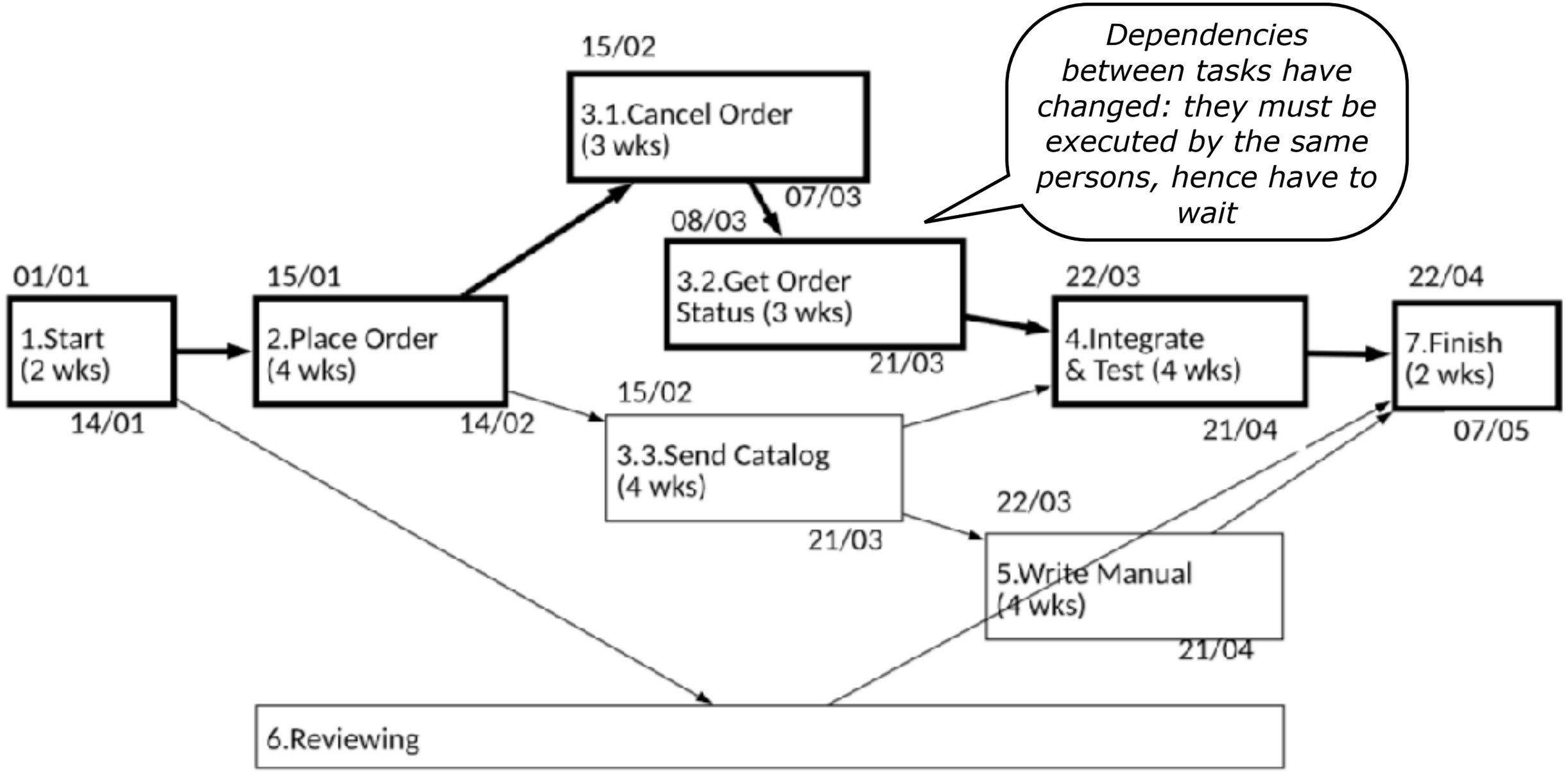
- Not for: Task Interdependencies

(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”? In a Ham and Egg Breakfast...the chicken is involved and the pig is committed!

- Build confidence
  + within the team
  + with the customer
    > ... re-planning will not be considered harmful

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. (See [Gold95])

Knowns & Unknowns

[This is terminology used for planning military campaigns.]
Phillip G. Armour, "The Five Orders of Ignorance", COMMUNICATIONS OF THE ACM October 2000

Slide Repeated from Intro

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

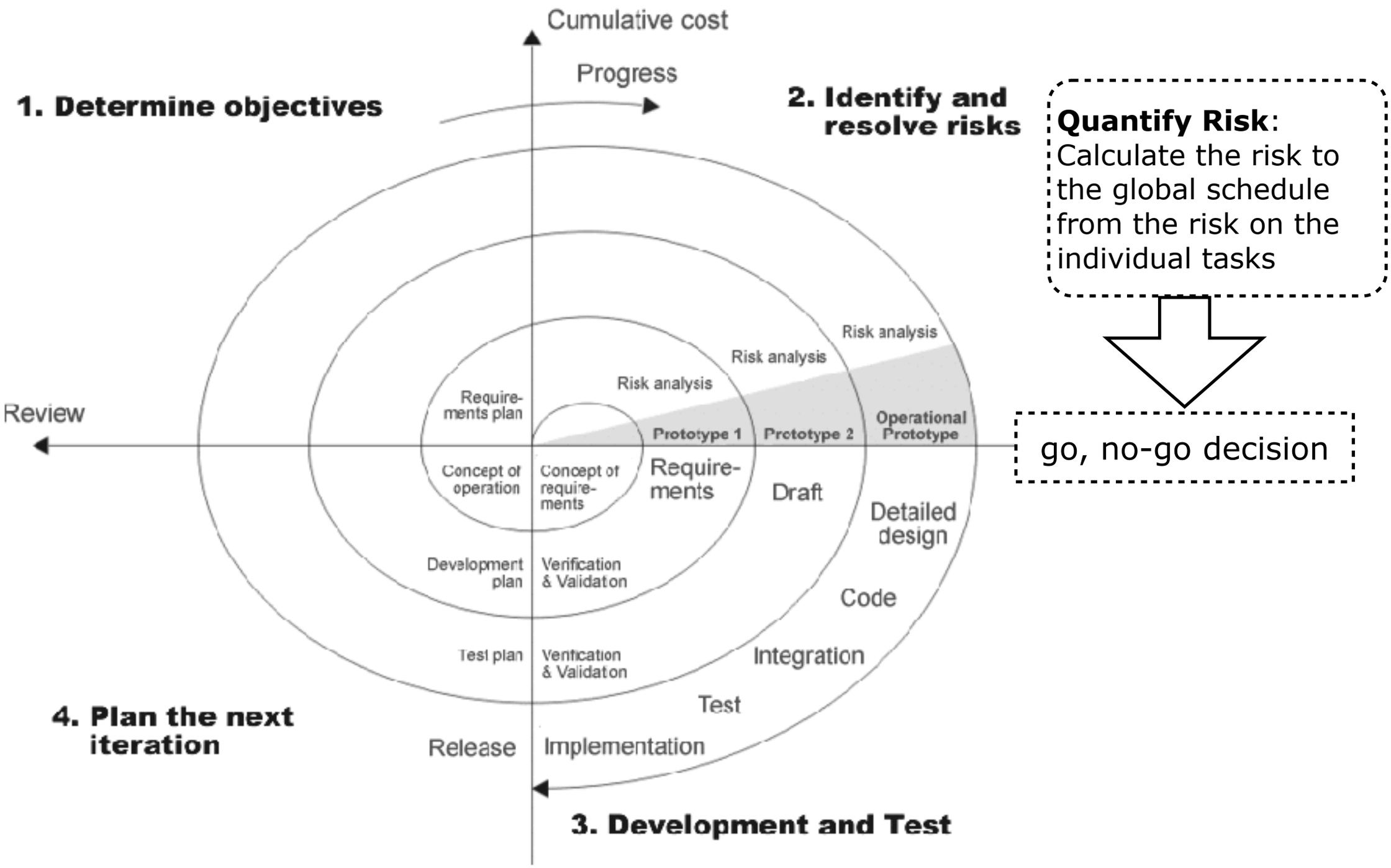
Inception: Risk Factors

Repeated from Requirements

- During inception you must identify the project's *risk factors*
  + you do not have control over the system's context and it will change
  + projects never go according to plan
    > identify potential problems early (… including wild success)
- Example

Context	Risk Factors	Impact	Likely	Urgency
Competitors	Time to market (too late/too early)			
Market trends	More internet at home			
Potential disasters	Suppliers don't deliver on time			
	System is down			
Expected users	Too many/few users			
Schedule	Project is delivered too early/too late	<< Risky Path (Project Management)		
Technology	Dependence on changing technology			
	Inexperienced team			
	Interface with legacy systems			

Risk Analysis: Quantify Risks for Delays



© Image adapted from Boehm, B. (1988) A Spiral Model of Software Development and Enhancement. IEEE Computer, 21 (5), 62-72.

Calculating Risky Path (1/2)

** Revised **
(Improved Formulas)

- (This calculation is an advanced but crucially important part of PERT)
- Estimate Task Time
  + For each task, estimate
    - likely time LT(task), optimistic time OT(task), pessimistic time PT(task)
    - deduce estimated time (= weighted average)

$$ET(task) = \frac{OT(task) + 4 \cdot LT(task) + PT(task)}{6}$$

    - Redo the critical path analysis with the estimated time ET
- Calculate Standard Deviation *per Task*
  + For each task, calculate the degree of uncertainty for the task time

$$S(task) = \frac{PT(task) - OT(task)}{6}$$

Example: Calculating Risk (1/2)

** Revised **
(Improved Formulas)

- Optimistic Time, Likely Time and Pessimistic Time is given
- deduce estimated time ET(task)
  + Redo the critical path analysis with ET

$$ET(task) = \frac{OT(task) + 4 \cdot LT(task) + PT(task)}{6}$$

- calculate standard deviation S(task)

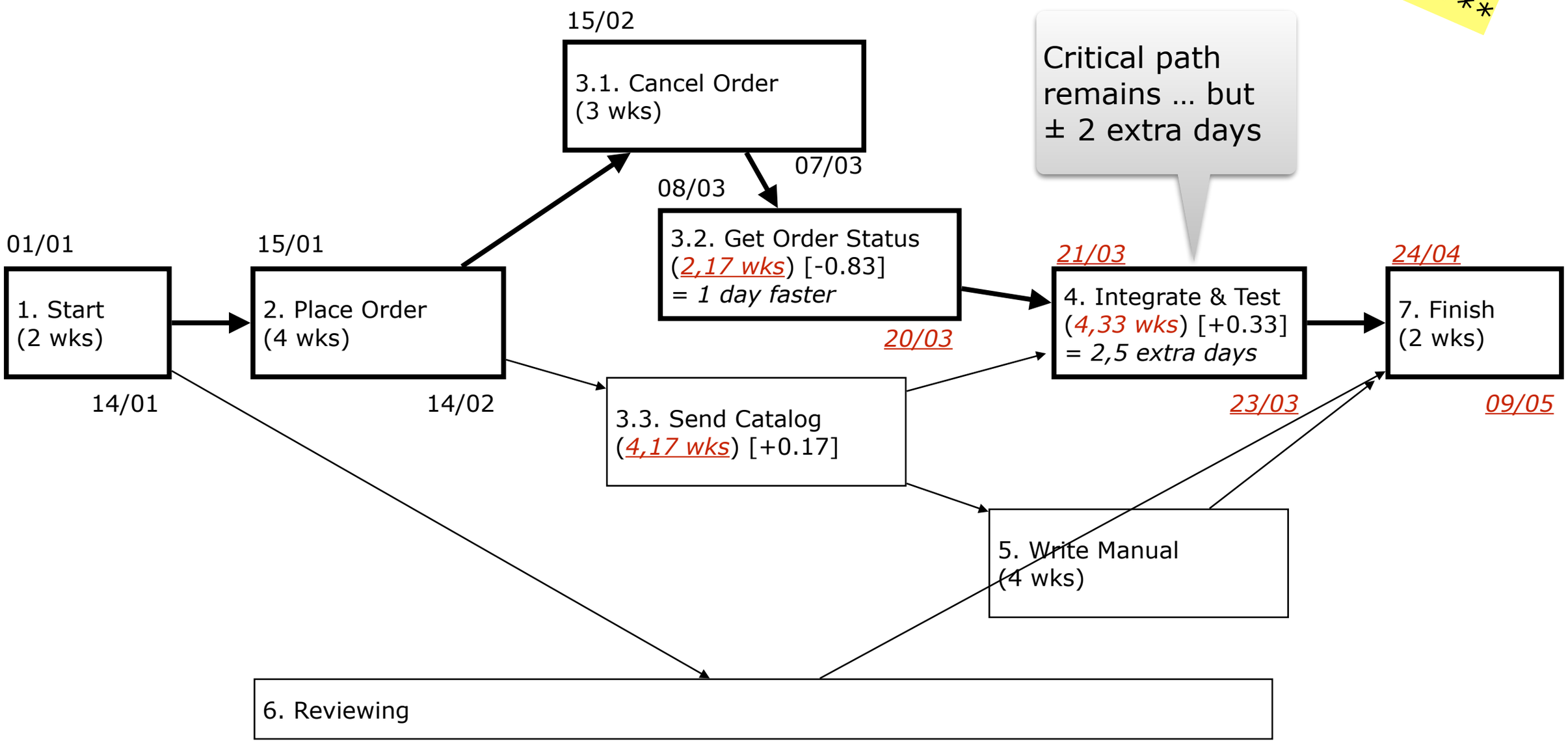
$$S(task) = \frac{PT(task) - OT(task)}{6}$$

	OT	LT	PT	ET	S
1.Start	2	2	2	2	0
2.Place Order	3	4	5	4	0,33
3.1.Cancel	2	3	4	3	0,33
3.2.Get Order	2	2	3	2,17	0,17
3.3.Send Catalogue	3	4	6	4,17	0,50
4.Test	4	4	6	4,33	0,33
5.Manual	3	4	5	4	0,33
7.Finish	2	2	2	2	0

Task 3.3 is riskiest task
(interface with legacy database)

Example: Redo Critical Path with ET

*** New ***



Calculating Risky Path (2/2)

** Revised **
(Improved Formulas)

- Forward Pass: Calculate Standard Deviation *per Path*
  + For each possible *path up until a given task n*
    - calculate the degree of uncertainty for the path execution time
      * Paths with a high deviation are likely to slip.

$$S(path) = \sqrt{\sum_{task \in path} S(task)^2}$$

  + For each task: compute standard deviation per path leading into the task
      * Degree to which a given task may end later than planned
      * = Maximum of all standard deviations for incoming paths

$$SP(task) = \max_{path \in incoming} S(path)$$

Results of Risky path Analysis

** Revised **
(Improved Formulas)

- Riskiest Task = the node with the highest risk for delay
  > Maximum for all S(task)

$$S(task) = \frac{PT(task) - OT(task)}{6}$$

- Risky Path = start-to-end path(s) with the highest standard deviation
  > Risky path applies to the whole PERT chart!
  > SP (end) := maximum of all incoming paths for end node

- Worst Case Delay: Applies to the risky path(s) only
  > = worst case impact the risky path may have on the end date

$$WorstCaseDelay(path) = \sum_{task \in path} PT(task) - LT(task)$$

Example: Calculating Risk (2/2)

** Revised **
(Improved Formulas)

- For each task n: compute standard deviation per path
  > = Maximum of all standard deviations for incoming paths

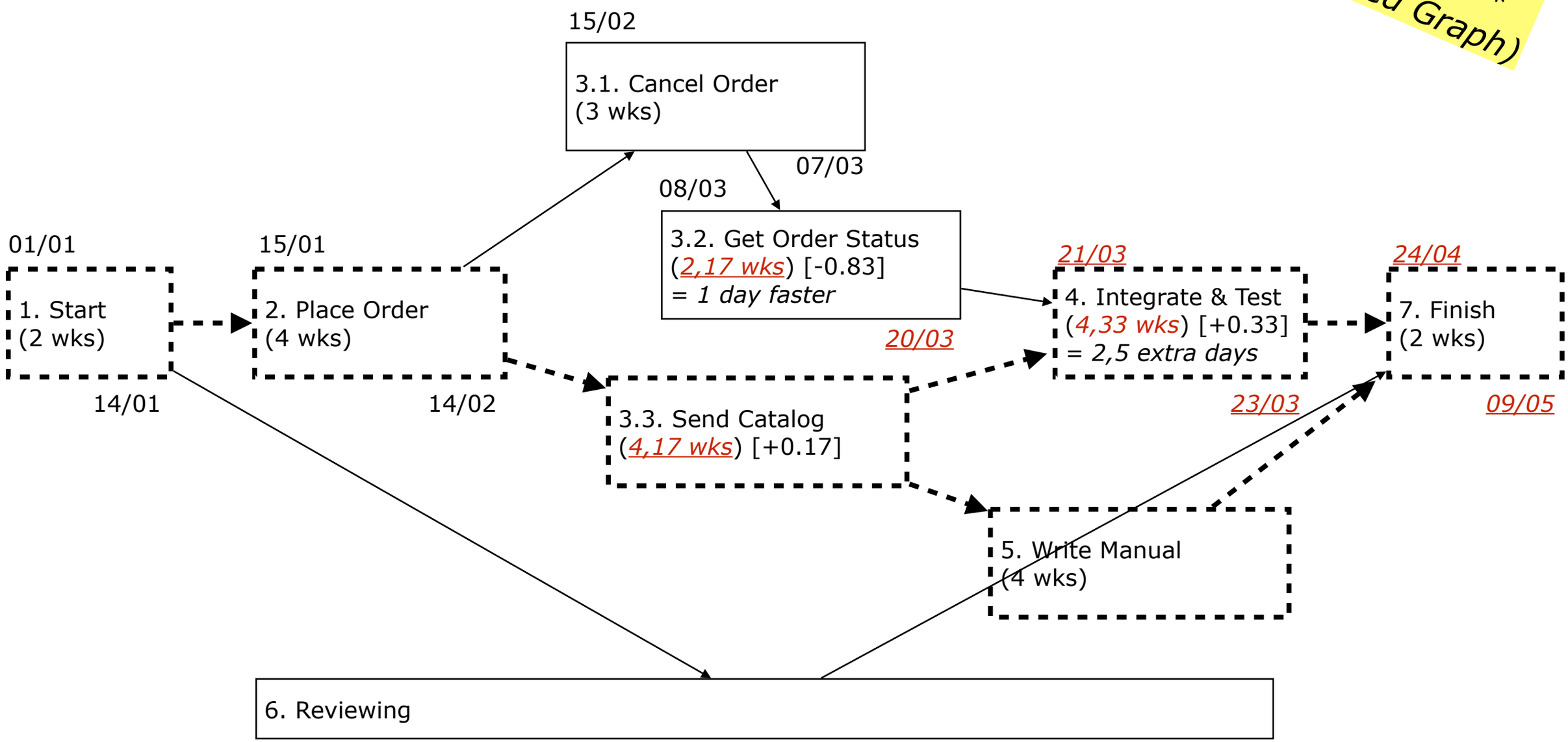
$$S(path) = \sqrt{\sum_{task \in path} S(task)^2}$$

End Node	path	$S(m_1)$	$S(m_2)$	$S(m_3)$	$S(m_4)$	$S(m_5)$	$S(m_6)$	$\sqrt{(\Sigma S(m_i)^2)}$	
1.Start	1	0						0	
2.Place O.	1,2	0	0,33					0,33	
3.1.Cancel	1,2,3.1	0	0,33	0,33				0,4667	
3.2.Get O.	1,2,3.1,3.2	0	0,33	0,33	0,17			0,4967	
3.3.Send C.	1,2,3.3	0	0,33	0,5				0,5991	
4.Test	1,2,3.1,3.2,4	0	0,33	0,33	0,17	0,33		0,5963	
	1,2,3.3,4	0	0,33	0,5	0,33			*0,684*	<< max
5.Manual	1,2,3.3,5	0	0,33	0,5	0,33			*0,684*	
7.Finish	1,2,3.1,3.2,4,7	0	0,33	0,33	0,17	0,33	0	0,5963	
	1,2,3.3,4,7	0	0,33	0,5	0,33		0	*0,684*	<< max
	1,2,3.3,5,7	0	0,33	0,5	0,33		0	*0,684*	<< max

⇒ Paths 1,2,3.3,4,7 and 1,2,3.3,5,7 represent largest risk!

Example: Risky Path

** Revised **
(Improved Graph)



- Worst case delay ("pessimistic time" minus "likely time" for all tasks on risky path)
  + 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

$$WorstCaseDelay(path) = \sum_{task \in path} PT(task) - LT(task)$$

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

Calculating Risk: exercise

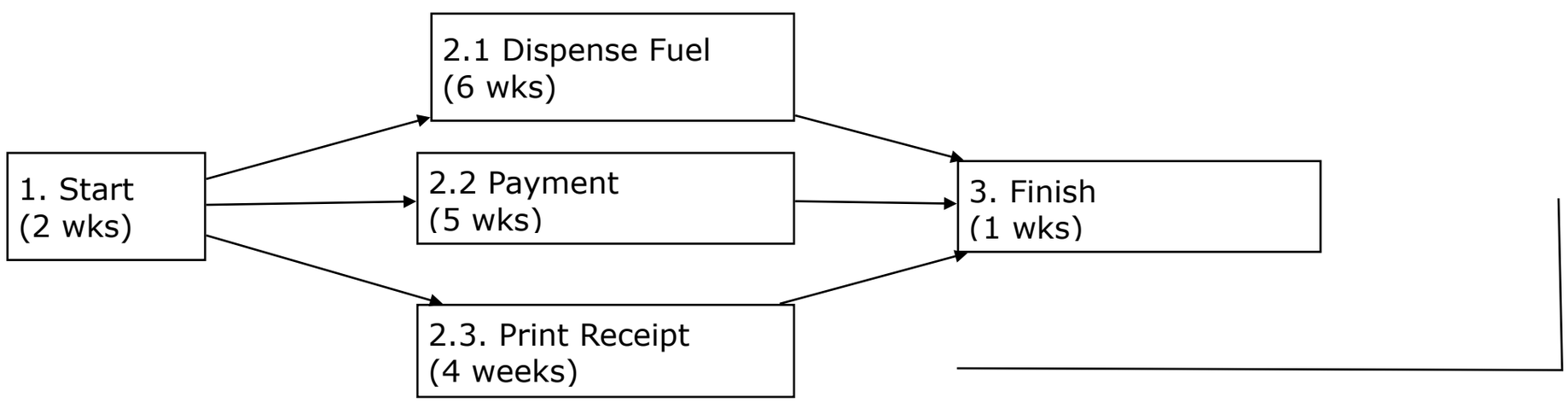
** New **

- What is the riskiest task?
- What is riskiest path?
- What is the worst case delay?

$$S(task) = \frac{PT(task) - OT(task)}{6}$$

$$S(path) = \sqrt{\sum_{task \in path} S(task)^2}$$

	OT	LT	PT	S	path	S(path)
1.Start	2	2	2		1	
2.1 Dispense Fuel	5	6	8		1,2.1	
2.2 Payment	4	5	8		1,2.2	
2.3 Print Receipt	3	4	5		1,2.3	
3. Finish	1	1	1		1,2.1, 3	
					1,2.2, 3	
					1,2.3, 3	



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!

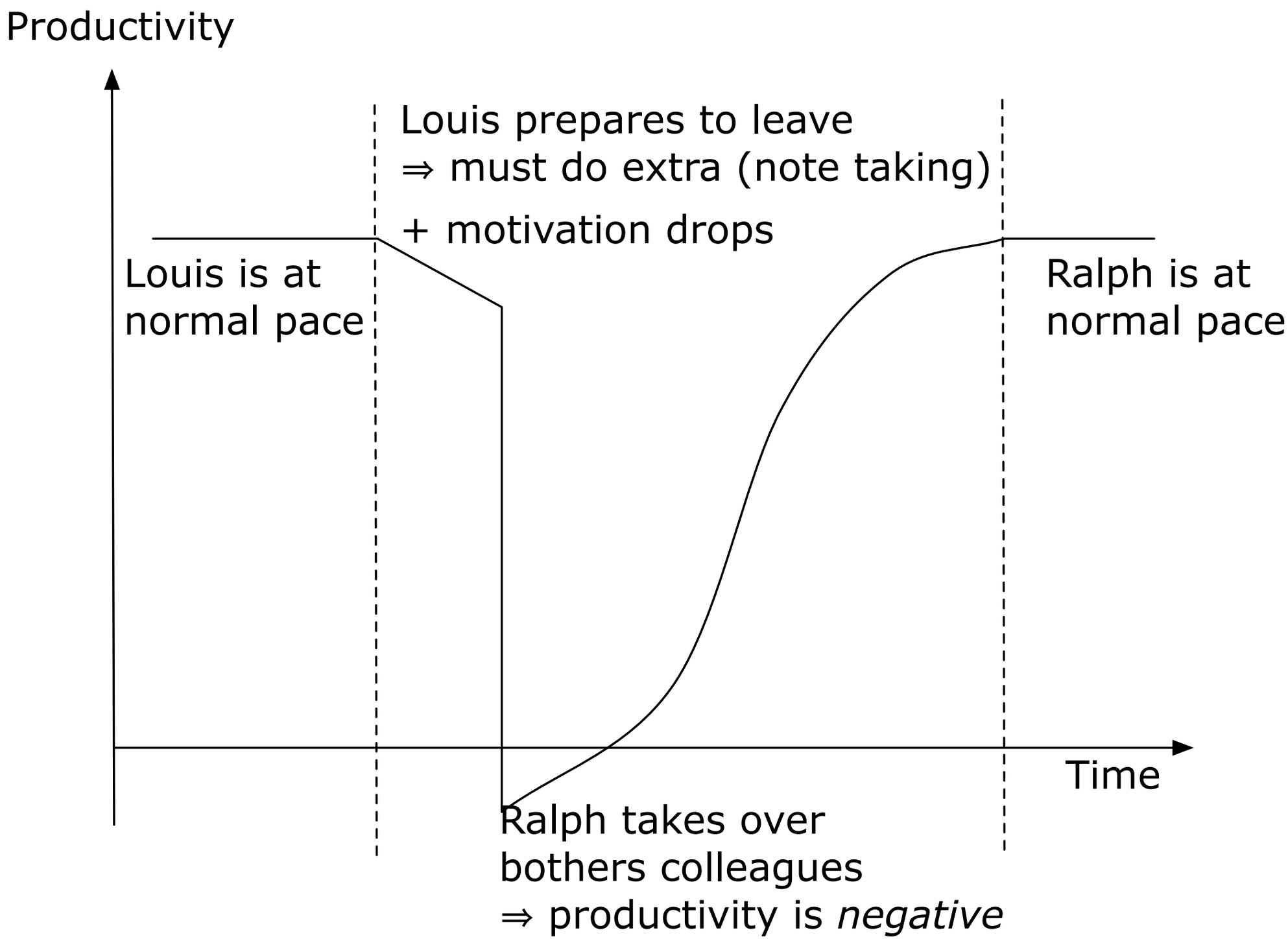
Delays

- 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 [Dema98], 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

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

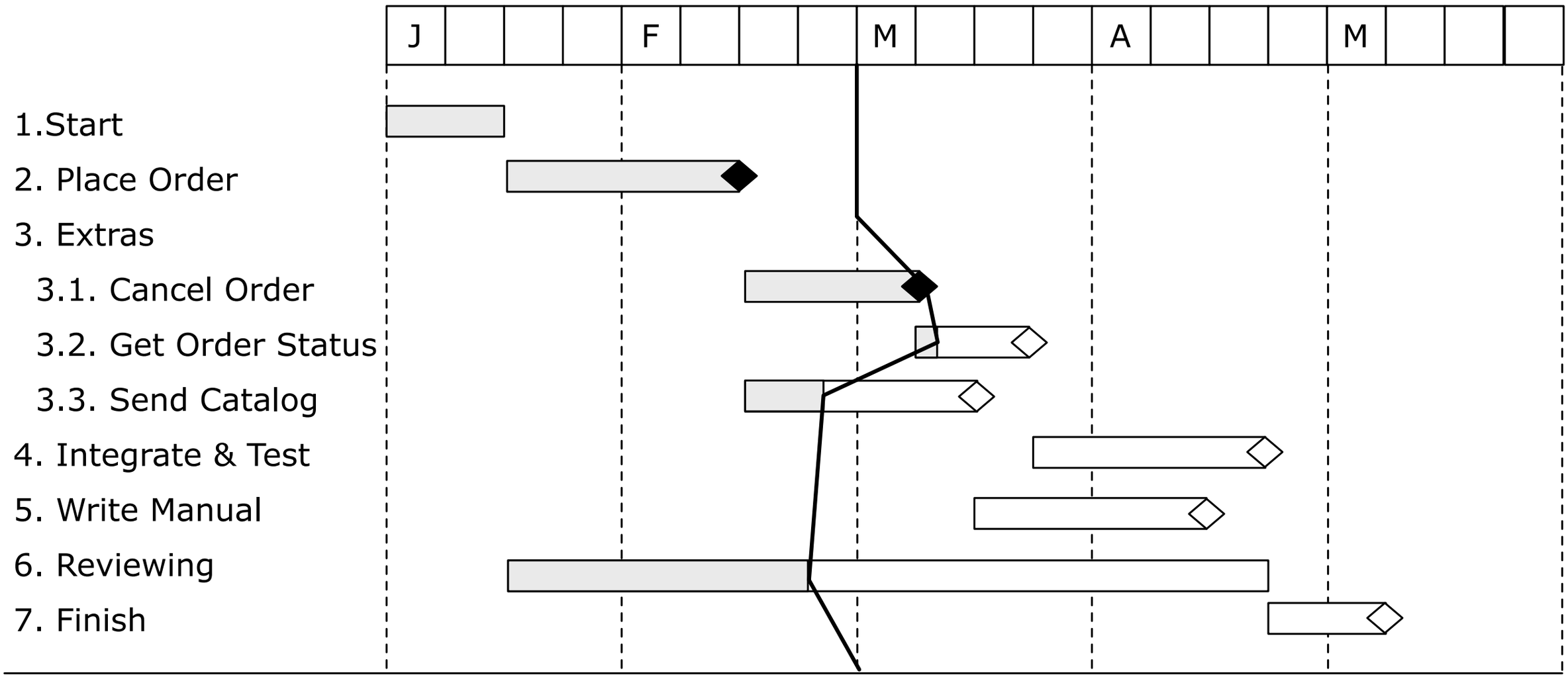
Opportunity to monitor team occupation

- 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



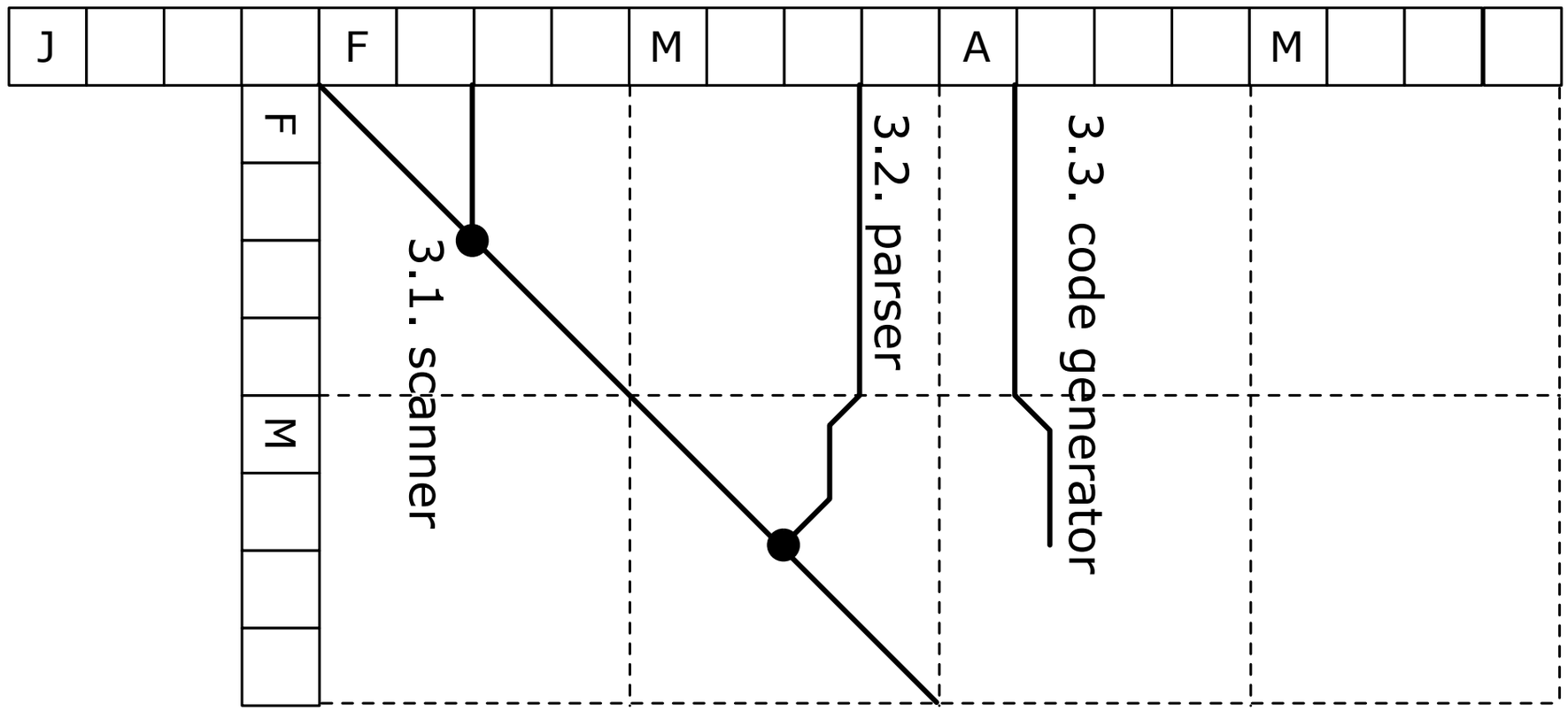
Interpretation

- 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

An afterthought ...

All projects that finish late have this one thing in common: they started late.

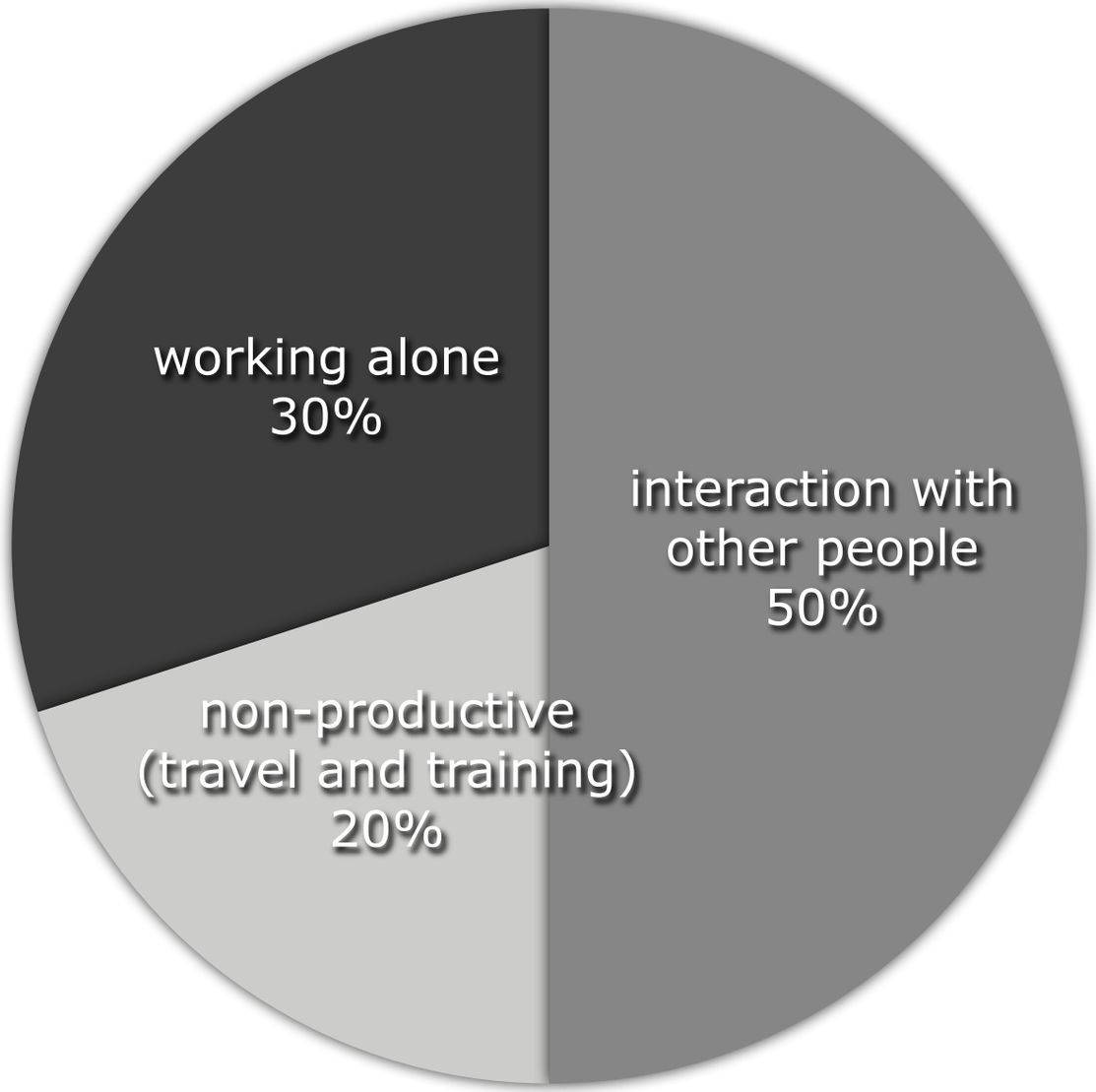
- [Dema11] Tom De Marco"All Late Projects Are the Same," IEEE Software, pp. 102-103, November/December, 2011

- 1. Nobody had the guts to kick off the project until the competition proved it doable and desirable; by then, the project was in catch-up mode and had to be finished lickety-split.
  ⇒ Business failure: blame marketing
- 2. If the project were started long enough before its due date to finish on time, all involved would have had to face up to the fact from the beginning that it was going to cost a lot more than anyone was willing to pay.
  + On the surface: poor risk analysis and cost estimation
  + What if gains would be orders of magnitude larger than the cost?
  + Who decides to start an expensive project with marginal gains?
  ⇒ Management failure: blame decision makers
- 3.No one knew that the project needed to be done until the window of opportunity was already closing.
  ⇒ Business failure + Management failure

Individuals work in Teams

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

- [McCu78] G M McCue, "IBM's Santa Teresa Laboratory — Architectural Design for Program Development," IBM Systems Journal, 17, 1, pp. 4-25, 1978]



IMPLICATIONS?

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

Belbin Team Roles

"Do you want a collection of brilliant minds or a brilliant collection of minds?"
[Dr. Raymond Meredith Belbin (1926)]

Action Oriented Roles	Shaper	Challenges the team to improve
	Implementer	Puts ideas into action
	Completer Finisher	Ensures thorough, timely completion
People Oriented Roles	Coordinator	Acts as a chairperson
	Team Worker	Encourages cooperation
	Resource Investigator	Explores outside opportunities
Thought Oriented Roles	Plant	Presents new ideas and approaches
	Monitor-Evaluator	Analyzes the options
	Specialist	Provides specialized skills

An *effective* team has members that cover nine classic team roles.

Overlap is possible!

Myers Briggs Type Inventory (MBTI)

** New **

Use the questions on the outside of the chart to determine the four letters of your Myers-Briggs type.
For each pair of letters, choose the side that seems most natural to you, even if you don't agree with every description.

1. Are you outwardly or inwardly focused? If you:

- Could be described as talkative, outgoing
- Like to be in a fast-paced environment
- Tend to work out ideas with others, think out loud
- Enjoy being the center of attention

then you prefer **E** Extraversion

- Could be described as reserved, private
- Prefer a slower pace with time for contemplation
- Tend to think things through inside your head
- Would rather observe than be the center of attention

then you prefer **I** Introversion

2. How do you prefer to take in information? If you:

- Focus on the reality of how things are
- Pay attention to concrete facts and details
- Prefer ideas that have practical applications
- Like to describe things in a specific, literal way

then you prefer **S** Sensing

- Imagine the possibilities of how things could be
- Notice the big picture, see how everything connects
- Enjoy ideas and concepts for their own sake
- Like to describe things in a figurative, poetic way

then you prefer **N** Intuition

ISTJ Responsible, sincere, analytical, reserved, realistic, systematic. Hardworking and trustworthy with sound practical judgment.

ISFJ Warm, considerate, gentle, responsible, pragmatic, thorough. Devoted caretakers who enjoy being helpful to others.

INFJ Idealistic, organized, insightful, dependable, compassionate, gentle. Seek harmony and cooperation, enjoy intellectual stimulation.

INTJ Innovative, independent, strategic, logical, reserved, insightful. Driven by their own original ideas to achieve improvements.

ISTP Action-oriented, logical, analytical, spontaneous, reserved, independent. Enjoy adventure, skilled at understanding how mechanical things work.

ISFP Gentle, sensitive, nurturing, helpful, flexible, realistic. Seek to create a personal environment that is both beautiful and practical.

INFP Sensitive, creative, idealistic, perceptive, caring, loyal. Value inner harmony and personal growth, focus on dreams and possibilities.

INTP Intellectual, logical, precise, reserved, flexible, imaginative. Original thinkers who enjoy speculation and creative problem solving.

ESTP Outgoing, realistic, action-oriented, curious, versatile, spontaneous. Pragmatic problem solvers and skilful negotiators.

ESFP Playful, enthusiastic, friendly, spontaneous, tactful, flexible. Have strong common sense, enjoy helping people in tangible ways.

ENFP Enthusiastic, creative, spontaneous, optimistic, supportive, playful. Value inspiration, enjoy starting new projects, see potential in others.

ENTP Inventive, enthusiastic, strategic, enterprising, inquisitive, versatile. Enjoy new ideas and challenges, value inspiration.

ESTJ Efficient, outgoing, analytical, systematic, dependable, realistic. Like to run the show and get things done in an orderly fashion.

ESFJ Friendly, outgoing, reliable, conscientious, organized, practical. Seek to be helpful and please others, enjoy being active and productive.

ENFJ Caring, enthusiastic, idealistic, organized, diplomatic, responsible. Skilled communicators who value connection with people.

ENTJ Strategic, logical, efficient, outgoing, ambitious, independent. Effective organizers of people and long-range planners.

3. How do you prefer to make decisions? If you:

- Make decisions in an impersonal way, using logical reasoning
- Value justice, fairness
- Enjoy finding the flaws in an argument
- Could be described as reasonable, level-headed

then you prefer **T** Thinking

- Base your decisions on personal values and how your actions affect others
- Value harmony, forgiveness
- Like to please others and point out the best in people
- Could be described as warm, empathetic

then you prefer **F** Feeling

4. How do you prefer to live your outer life? If you:

- Prefer to have matters settled
- Think rules and deadlines should be respected
- Prefer to have detailed, step-by-step instructions
- Make plans, want to know what you're getting into

then you prefer **J** Judging

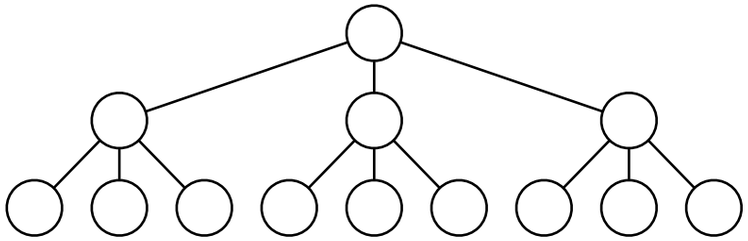
- Prefer to leave your options open
- See rules and deadlines as flexible
- Like to improvise and make things up as you go
- Are spontaneous, enjoy surprises and new situations

then you prefer **P** Perceiving

Typical Team Structures

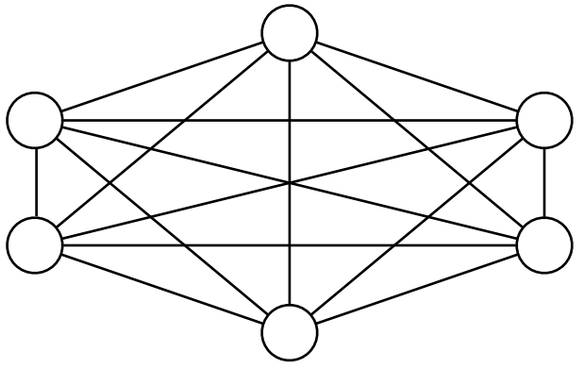
Hierarchical (Centralized)
e.g. Chief Programmer

- For well-understood problems
- Predictable, fast development
- Large groups



Consensus (Decentralized)
e.g. Egoless Programming Team

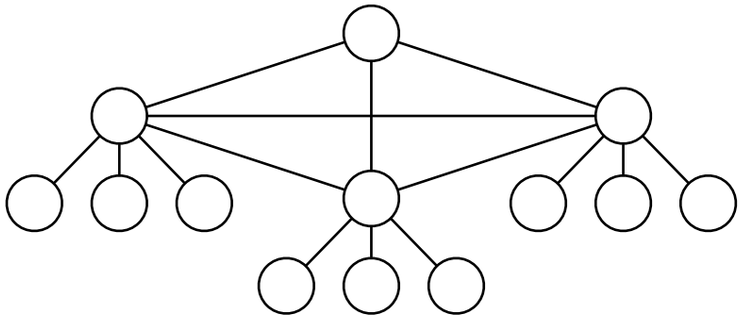
- For exploratory projects
- Fast knowledge transfer
- Small groups



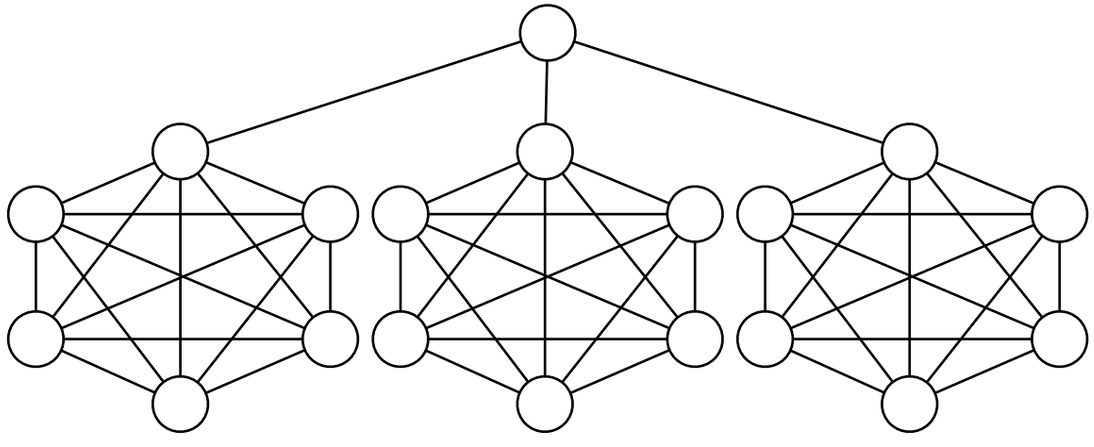
There is no "one size fits all" team structure!

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



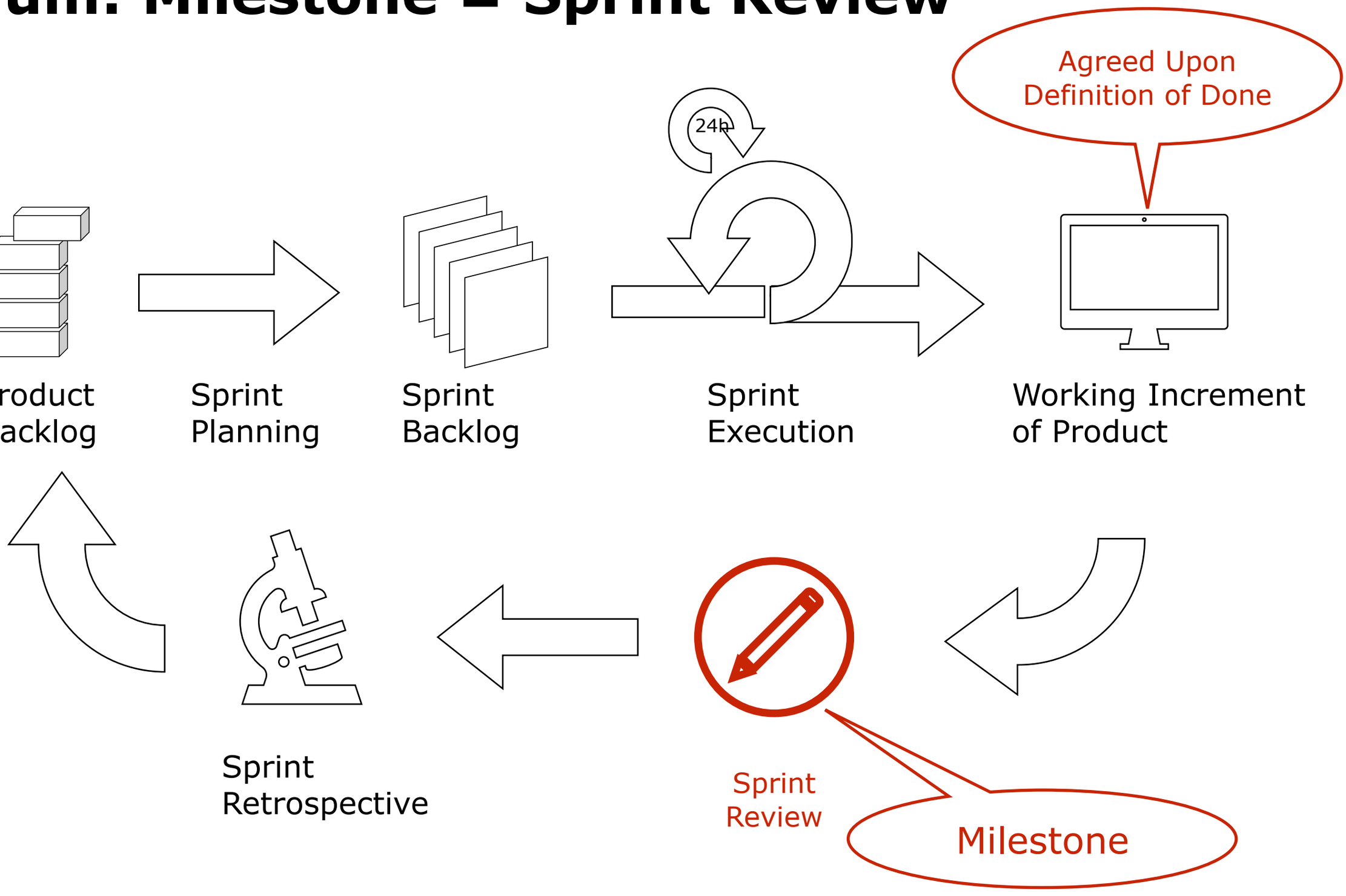
Directing Teams

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

Scrum: Milestone = Sprint Review



Definition of Done

definition of done = a checklist of the types of work that the team is expected to successfully complete before it can declare its work to be potentially shippable.

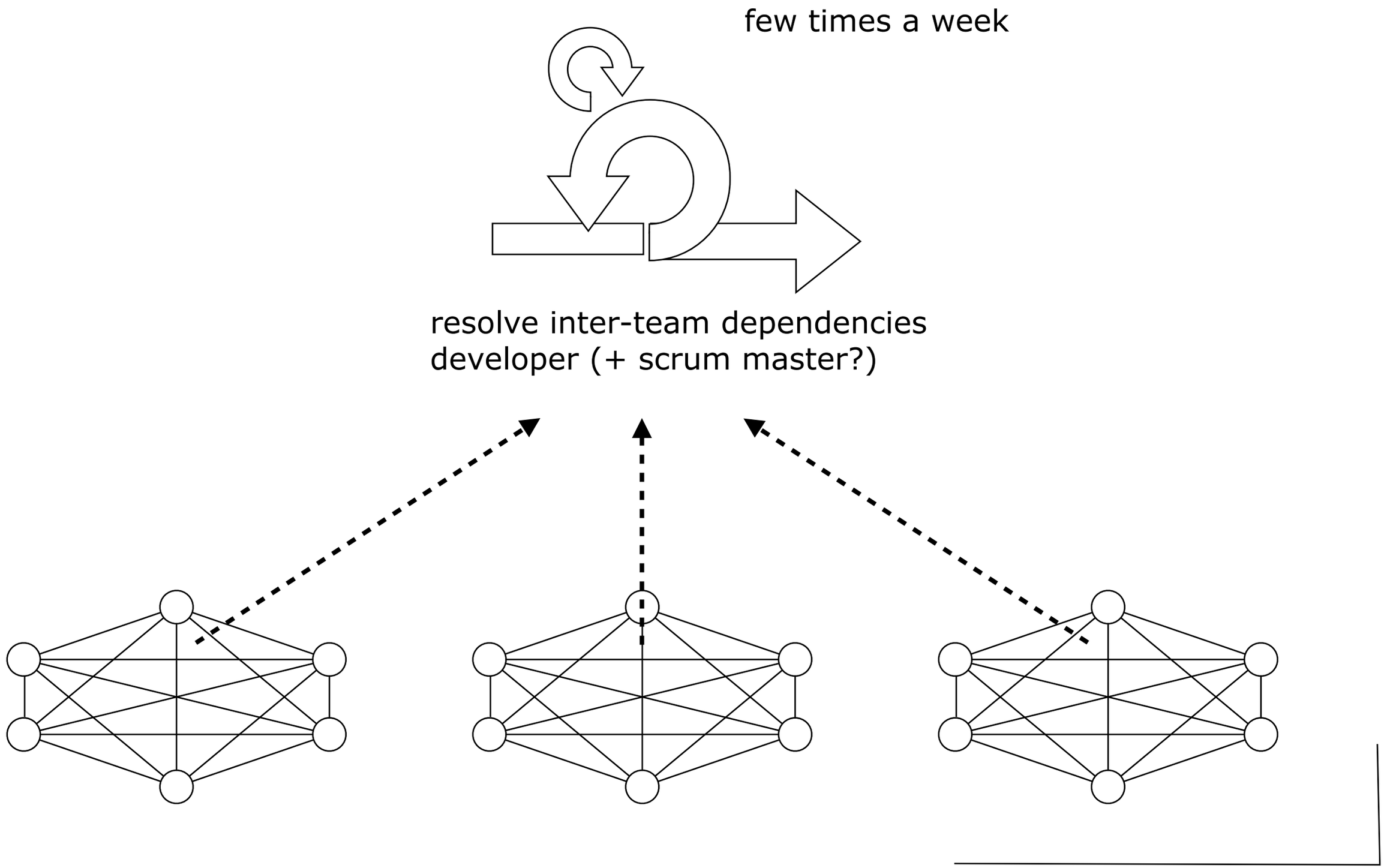
Different levels of "doneness":

- Task level
- User story level
  + (e.g. completed FIT acceptance tests with customer)
- Iteration level
  + (e.g. all stories developed, all bugs closed)
- Release level
  + (e.g. installation package created, stress testing completed)

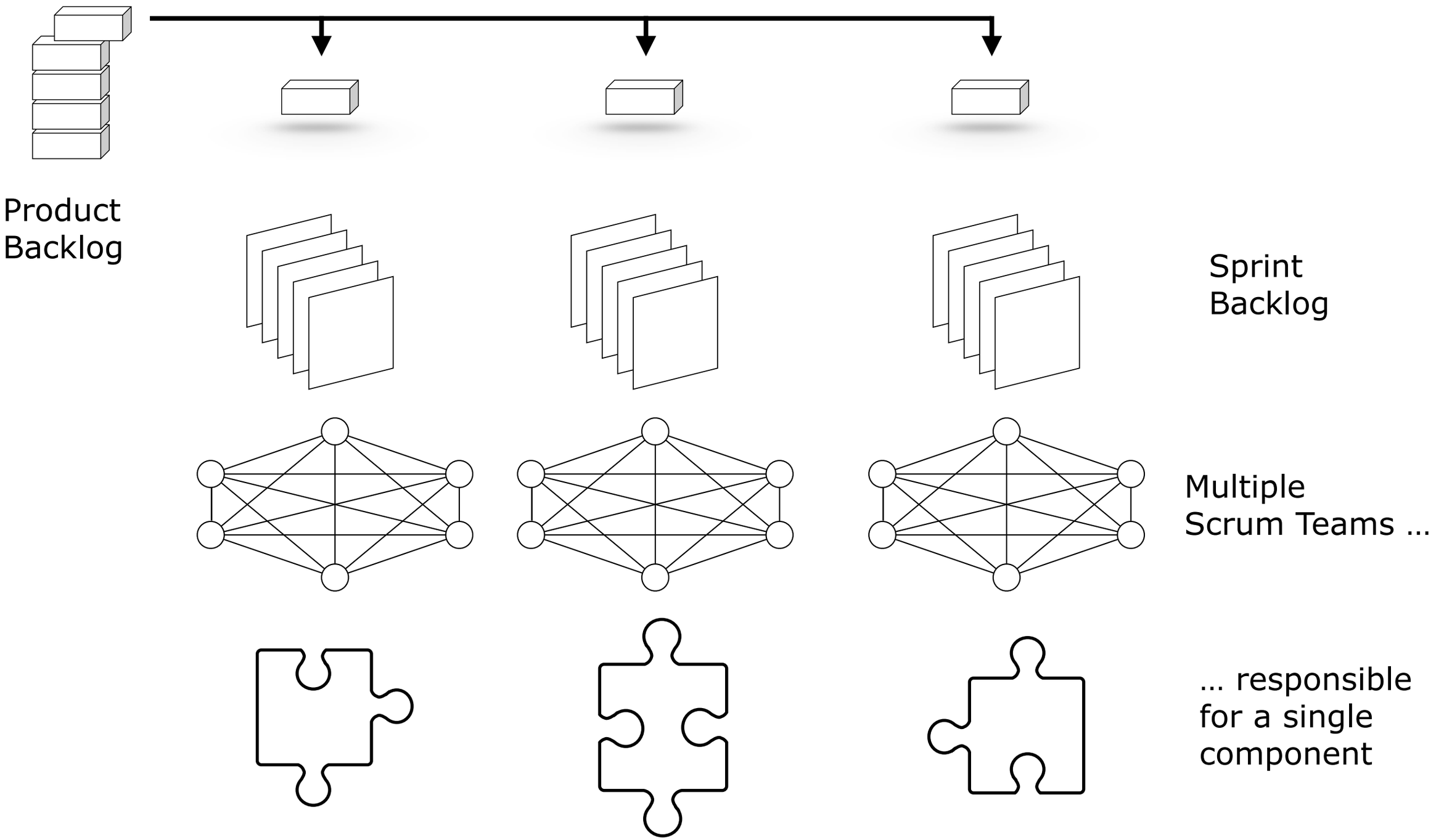
✔	Design reviewed
✔	Code completed
✔	Code refactored
✔	Code in standard format
✔	Code is commented
✔	Code checked in
✔	Code inspected
✔	End-user documentation
✔	Tested
✔	Unit tested
✔	Integration tested
✔	Regression tested
✔	Platform tested
✔	Language tested
✔	Zero known defects
✔	Acceptance tested
✔	Live on production servers

Scaling Scrum: Scrum of Scrum

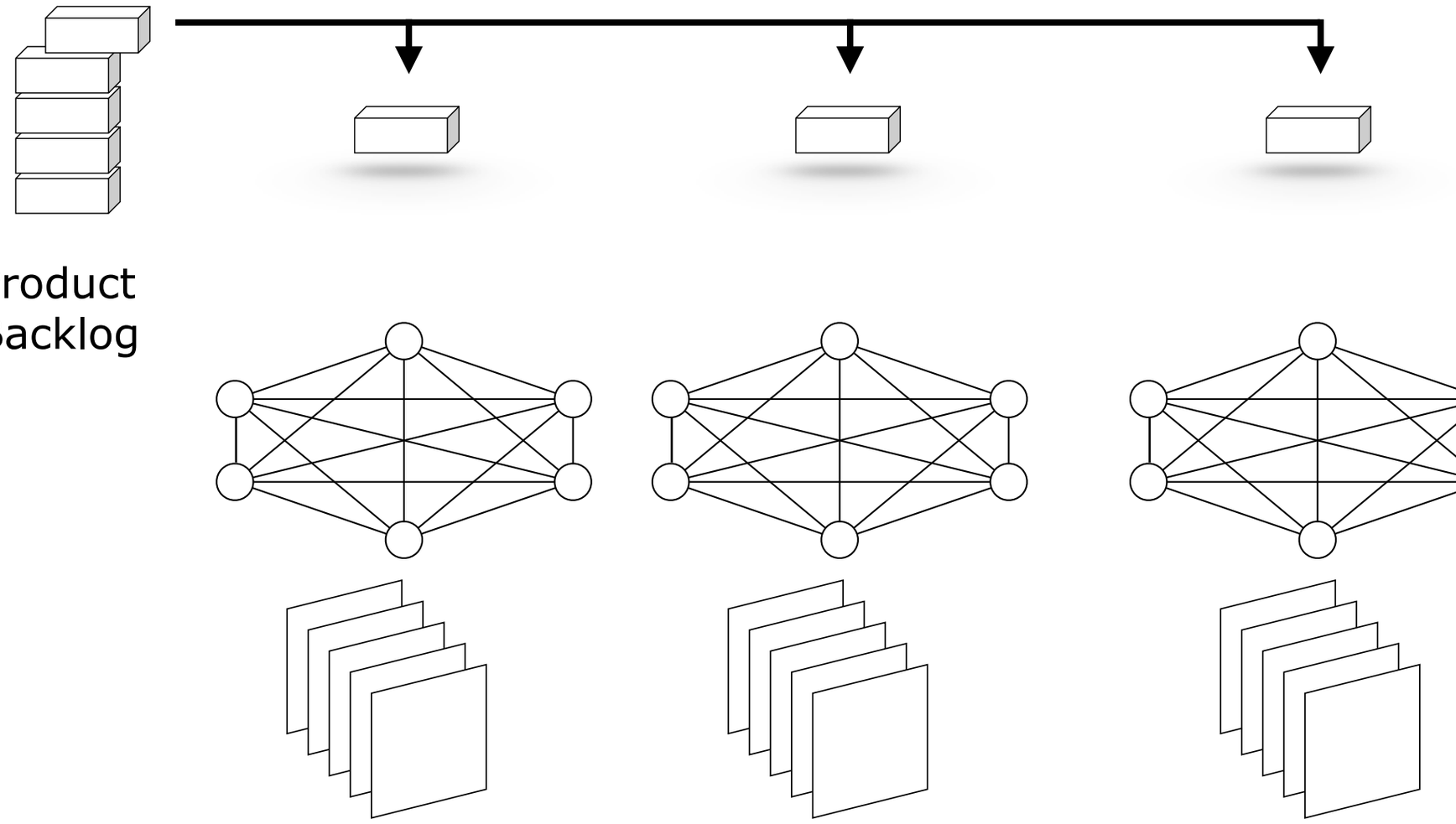
Synchronisation of work via “scrum of scrums”



Scaling Scrum: Component Team

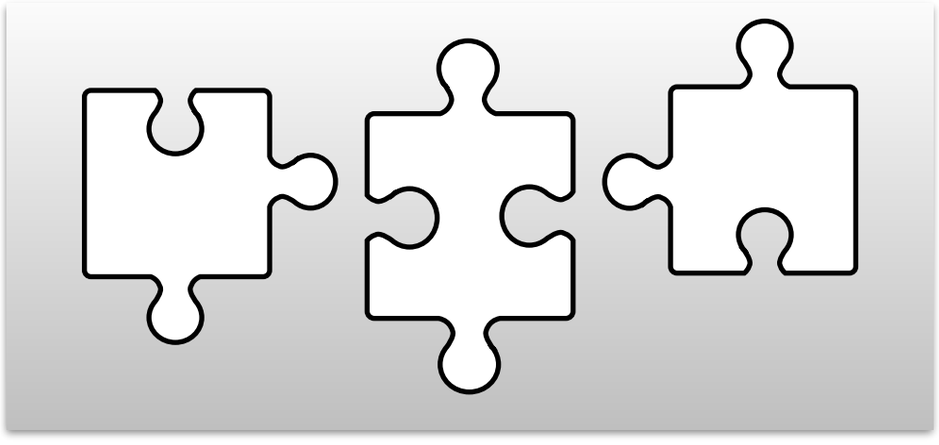


Scaling Scrum: Feature Team



Multiple
Scrum Teams …

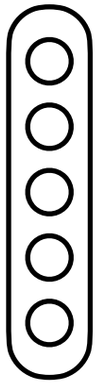
… responsible for
a single feature



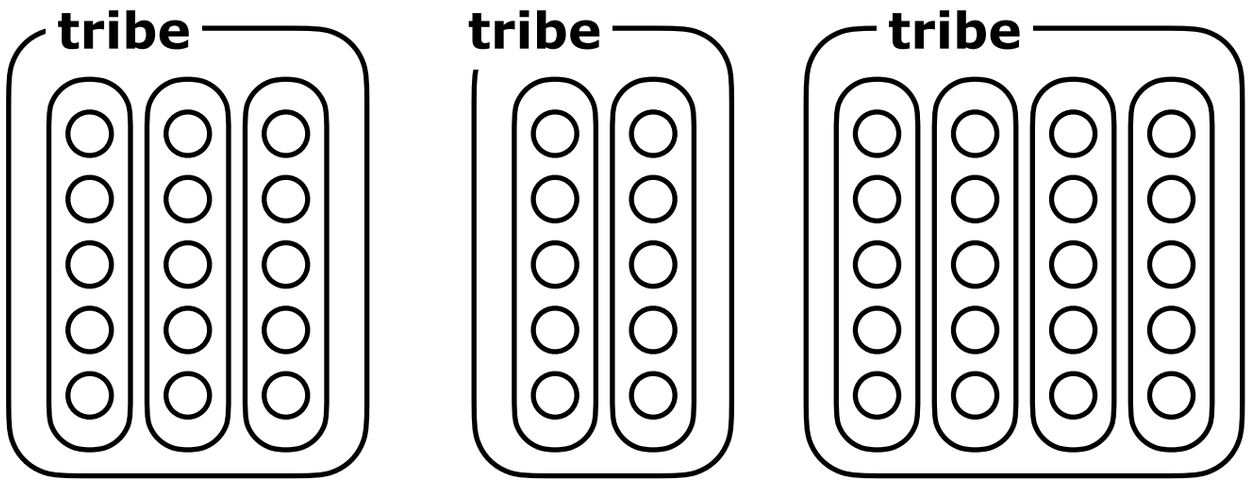
Joint code
ownership

Spotify Scrum Model

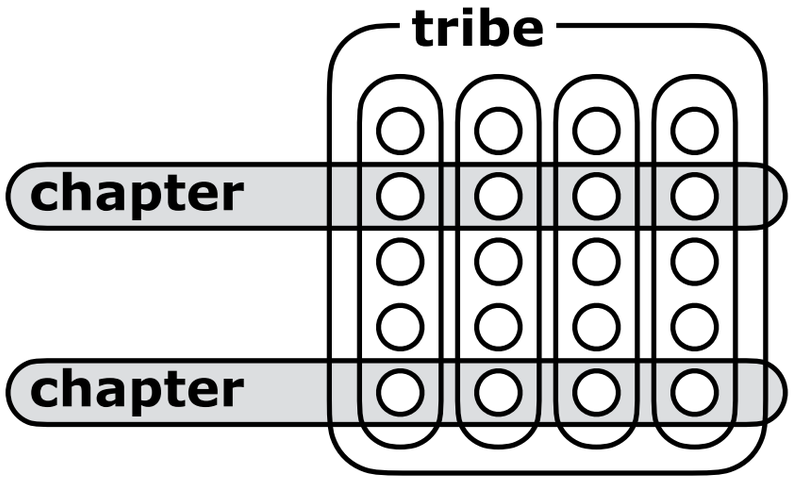
Squad =
Scrum Team



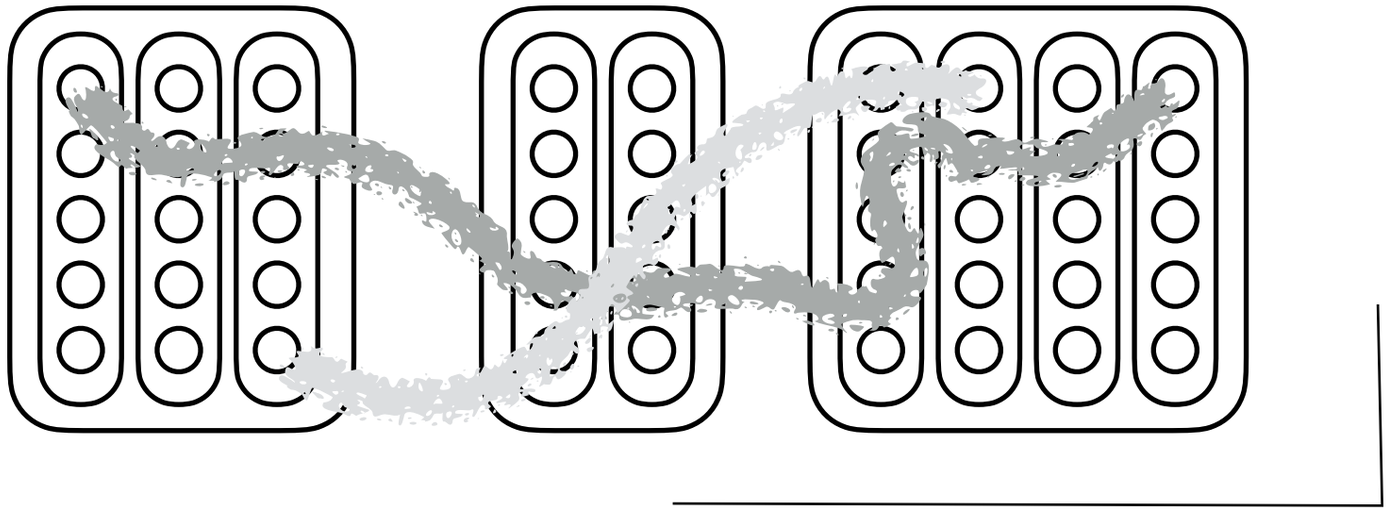
Tribe =
Loosely coupled Scrum Teams working
on related features/components



Chapter =
Team members with similar
expertise within a tribe.



Guild =
Team members with similar
interests across tribes.



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
  + Name the five activities covered by project management.
  + 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 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?
  + What is the “definition of done” in a Scrum project?
  + Give a definition for a Squad, Tribe, Chapter and Guild in the spotify scrum model.

- 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 management. Which ones do you consider most important? Why?
  + How can you ensure traceability between the plan and the requirements/system?
  + Compare PERT-charts with Gantt charts for project planning and monitoring.
  + How can you deal with “Unknown Unknowns” 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 almost 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?
  + Can you give some advantages and disadvantages of scrum component teams and scrum feature teams.