CHAPTER 2 – 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 completes, Time sheets
  + Slip Lines, Timelines
• An afterthought: late projects ... started late

Organisation, Staffing, Directing
• Belkin Roles
• Team Structures
• Directing Teams

Conclusion
• Correctness & Traceability
2. Project Management

**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**

  ➡ “adding people to a late project makes it later” + lots of timeless wisdom
  ➡ Lots of advice on how the sociology in teams affect productivity
  ➡ Good practical examples on PERT, Gantt, Time-sheets, ...
  ➡ Explains how to define your own project management strategy
Literature - Papers

  + Demonstrating that Belbin roles do make a difference in team efficiency, even for student projects
  + All projects that finish late have this one thing in common: they started late.
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 = **Deliver on time and within budget**

Achieve interdependent & conflicting goals ...

... with limited resources.

*Your project team is a resource!*
What is Project Management?

Project Management = Plan the work and work the plan

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!
2. Project Management

**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

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)
2. Project Management

- **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
2. Project Management

**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

1. Start
2. Place Order
3. Extras
   3.1. Cancel Order
   3.2. Get Order Status
   3.3. Send Catalog
4. Integrate & Test
5. Write Manual
6. Reviewing
7. Finish

(task) slack time milestone summary task
For each task, list the required resources.
• Mainly staff (incl. type of skills required)
• ... and special equipment

<table>
<thead>
<tr>
<th>Activity</th>
<th>Resource</th>
<th>Time</th>
<th>Quantity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Senior Programmer</td>
<td>2 wks</td>
<td>2</td>
<td>Initially senior programmers only</td>
</tr>
<tr>
<td>2</td>
<td>Senior Programmer</td>
<td>4 wks</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Senior Programmer</td>
<td>3 wks</td>
<td>1</td>
<td>Implementation: extra junior staff</td>
</tr>
<tr>
<td></td>
<td>Junior Programmer</td>
<td>3 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Senior Programmer</td>
<td>2 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Junior Programmer</td>
<td>2 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Senior Programmer</td>
<td>4 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Junior Programmer</td>
<td>4 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Senior Programmer</td>
<td>4 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Junior Programmer</td>
<td>4 wks</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Senior Programmer</td>
<td>4 wks</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writer</td>
<td>4 wks</td>
<td>1</td>
<td>Manual</td>
</tr>
<tr>
<td>6</td>
<td>Quality Engineer</td>
<td>1 day/wk</td>
<td>1</td>
<td>Assistance from QA department</td>
</tr>
<tr>
<td>7</td>
<td>Senior Programmer</td>
<td>2 wks</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Gantt Chart: Resource Allocation

1. Start
2. Place Order
3. Extras
   3.1. Cancel Order
   3.2. Get Order Status
   3.3. Send Catalog
4. Integrate & Test
5. Write Manual
6. Reviewing
7. Finish

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

1. Start
2. Place Order
3. Extras
   3.1. Cancel Order
   3.2. Get Order Status
   3.3. Send Catalog
4. Integrate & Test
5. Write Manual
6. Reviewing
7. Finish

Senior Programmer

Junior Programmer

Deadline extended ...

... but smooth resource usage
## Gantt Chart: Staff Allocation

<table>
<thead>
<tr>
<th></th>
<th>J</th>
<th>F</th>
<th>M</th>
<th>A</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Darius</td>
<td>1.Start</td>
<td>2.Place Order</td>
<td>3.1 Canc</td>
<td>3.2 ...</td>
<td>4. Test</td>
</tr>
<tr>
<td>Marta</td>
<td>1.Start</td>
<td>2.Place Order</td>
<td>3.3 Send</td>
<td>5. Write Man</td>
<td>7. Finish</td>
</tr>
<tr>
<td>Leo</td>
<td></td>
<td></td>
<td>3.1 Canc</td>
<td>3.2 ...</td>
<td>4. Test</td>
</tr>
<tr>
<td>Ryan</td>
<td>3.3 Send</td>
<td></td>
<td></td>
<td>4. Test</td>
<td></td>
</tr>
<tr>
<td>Sylvia</td>
<td></td>
<td></td>
<td></td>
<td>5. Write Man</td>
<td></td>
</tr>
</tbody>
</table>

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

Dependencies between tasks have changed: they must be executed by the same persons, hence have to wait.
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
2. Project Management

Risk Analysis: Quantify Risks for Delays

1. Determine objectives
2. Identify and resolve risks
3. Development and Test
4. Plan the next iteration

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 $ET(n) := (OT(n) + 4 \times LT(n) + PT(n)) / 6$

**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
  ➡ $SP (n) := \text{maximum of } \left( \sqrt{\sum S(mi)^2} \right)$
  [where $mi$ is a node in the path to $n$]
  $$\sqrt{\sum_{i=1}^{n} S(m_i)^2}$$
### Example: Calculating Risk

<table>
<thead>
<tr>
<th></th>
<th>OT</th>
<th>LT</th>
<th>PT</th>
<th>ET</th>
<th>S</th>
<th>SP is maximum of columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>path 1</td>
<td></td>
</tr>
<tr>
<td>2. Place O.</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>0,33</td>
<td>0,33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>path 1,2</td>
<td></td>
</tr>
<tr>
<td>3.1. Cancel</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0,33</td>
<td>0,47</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>path 1,2,3.1</td>
<td></td>
</tr>
<tr>
<td>3.2. Get O.</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2,17</td>
<td>0,17</td>
<td>0,5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>2,17</td>
<td></td>
<td>path 1,2,3.1,3.2</td>
</tr>
<tr>
<td>3.3. Send C.</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4,17</td>
<td>0,5</td>
<td>0,6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,17</td>
<td></td>
<td>path 1,2,3.3</td>
</tr>
<tr>
<td>4. Test</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4,33</td>
<td>0,33</td>
<td>0,6, 0,69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,33</td>
<td></td>
<td>path 1,2,3.1,3.2,4, 1,2,3.3,4</td>
</tr>
<tr>
<td>5. Manual</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>0,33</td>
<td>0,69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>path 1,2,3.3,5</td>
</tr>
<tr>
<td>7. Finish</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,6 0,69 0,69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>path 1,2,3.1,3.2,4,7 1,2,3.3,4,7 1,2,3.3,5,7</td>
<td></td>
</tr>
</tbody>
</table>

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!
2. Project Management

**PERT Chart: Risky Path**

- 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

<table>
<thead>
<tr>
<th>Early with big risk for delay</th>
<th>Later with small risk for delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>delivery of project within 4 (four) months ... but can be 1 month early ... or 4 months late!</td>
<td>delivery of project within 5 (five) months ... at maximum 1 week late ... or 1 week early.</td>
</tr>
</tbody>
</table>

- 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)

Productivity

<table>
<thead>
<tr>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ralph is at normal pace</td>
</tr>
<tr>
<td>Louis prepares to leave ⇒ must do extra (note taking) + motivation drops</td>
</tr>
<tr>
<td>Ralph takes over bothers colleagues ⇒ productivity is negative</td>
</tr>
<tr>
<td>Louis is at normal pace</td>
</tr>
</tbody>
</table>

(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 (\(\text{=}\) Time sheets)

Organizations usually require staff to maintain time sheets
\(\text{=}\) bookkeeping of time spent by an individual for a particular task in a project

Opportunity to monitor team occupation
- Compare time spent (\(\text{=}\) 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

<table>
<thead>
<tr>
<th>J</th>
<th>F</th>
<th>M</th>
<th>A</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Place Order</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Extras</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1. Cancel Order</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2. Get Order Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3. Send Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Integrate &amp; Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Write Manual</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Reviewing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Finish</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 end 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.
[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


**IMPLICATIONS**

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

---

**Diagram Description**

- **Working alone** (30%)
- **Interaction with other people** (50%)
- **Non-productive (travel and training)** (20%)
Belbin Team Roles

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

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

An effective team has members that cover nine classic team roles.
Overlap is possible!
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 than 8 (eight) persons in total!
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 known”, 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 Gantt chart, incl. allocating and optimizing of resources
- draw a slip line and a timeline
Can you answer the following questions?

- Name the various activities covered by project management. 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 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?