Project Management

Software Architecture, Process, and Management

This lecture will look at traditional project management for large projects. This is mostly to give context to what follows. What can we learn from traditional engineering projects? How are software engineering projects different? How can we best exploit that?

Project Scope Statement

- In order to make any decisions about the project, it needs to be made concrete.
- This is done by writing a scope statement, such as:
  - We will build a garden shed capable of holding 2 bicycles, a lawn mower, and a small workbench
  - Planning permission will not be needed
  - We will not connect it to electric or water mains
  - It cannot cost more than 1000 pounds or take longer than 1 month (may be a constraint rather than scope)
- Note that the scope statement includes not only what it will do, but what will **not** be included.

Example Project

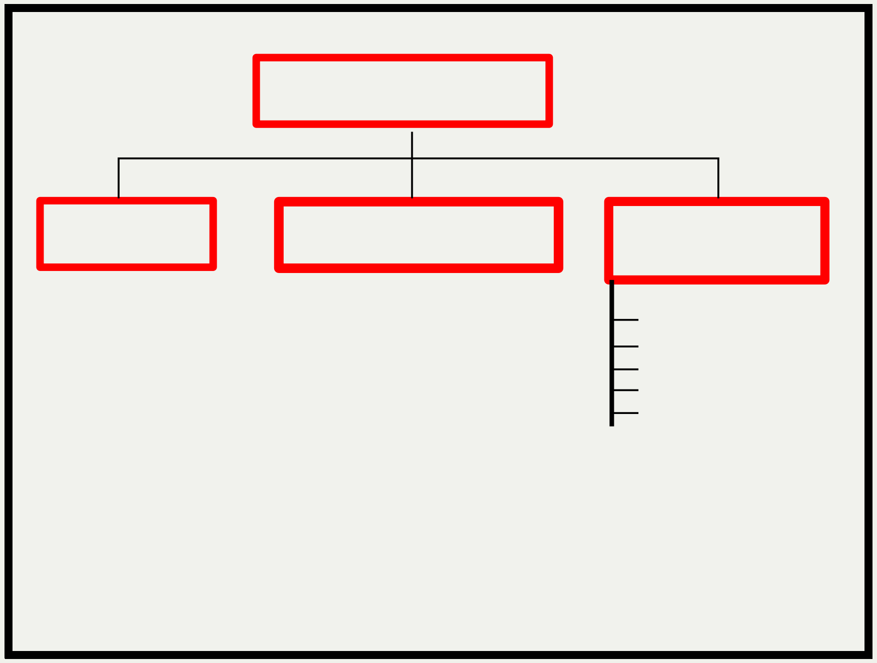
- A garden shed project may involve:
  1. Designing the shed
  2. Figuring out what materials are needed
  3. Ordering or purchasing the materials
  4. Putting together the various parts
- Some of these tasks depend on the others, some must be scheduled, some take labour, etc.
- Note that the 3rd item is an exit item:
  - You could cancel the project before that at minimal cost
  - You would not like to complete only a part of the 3rd item

Project Management Tasks

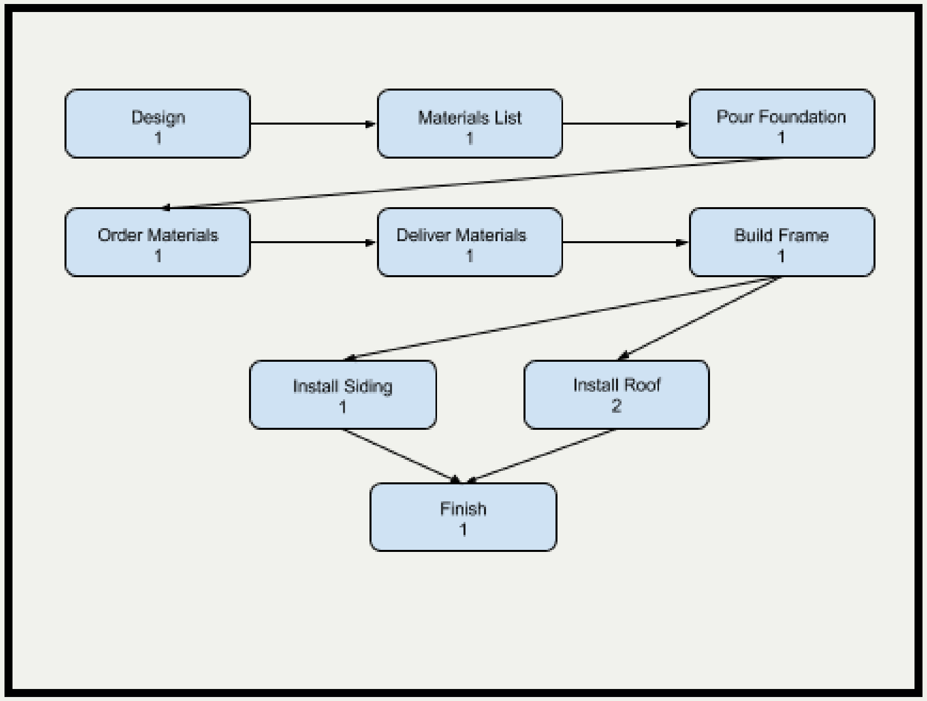
- Probably before work starts, need to figure out:
  - What needs to be done
  - What order they can be done in
  - How long each will take
  - How long the whole project must take in principle
  - How long the whole project is expected to take, given finite resources

Work Breakdown Structure

- A WBS is a diagram showing the major subtasks of the project:
- Rule of thumb: break things down as far as necessary to estimate and schedule them, and no further.

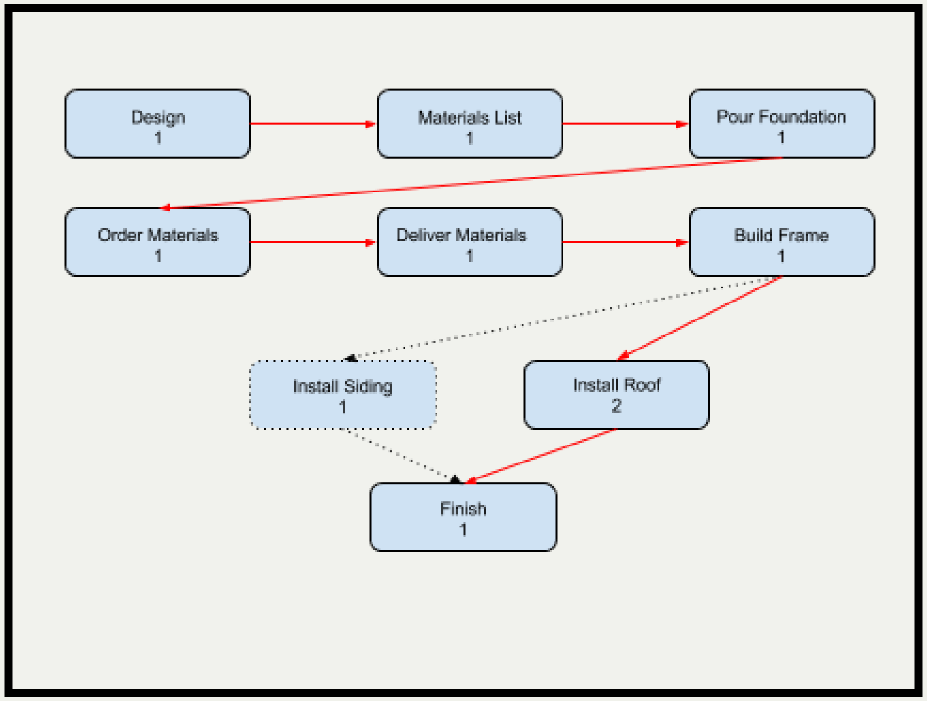


Network Diagram



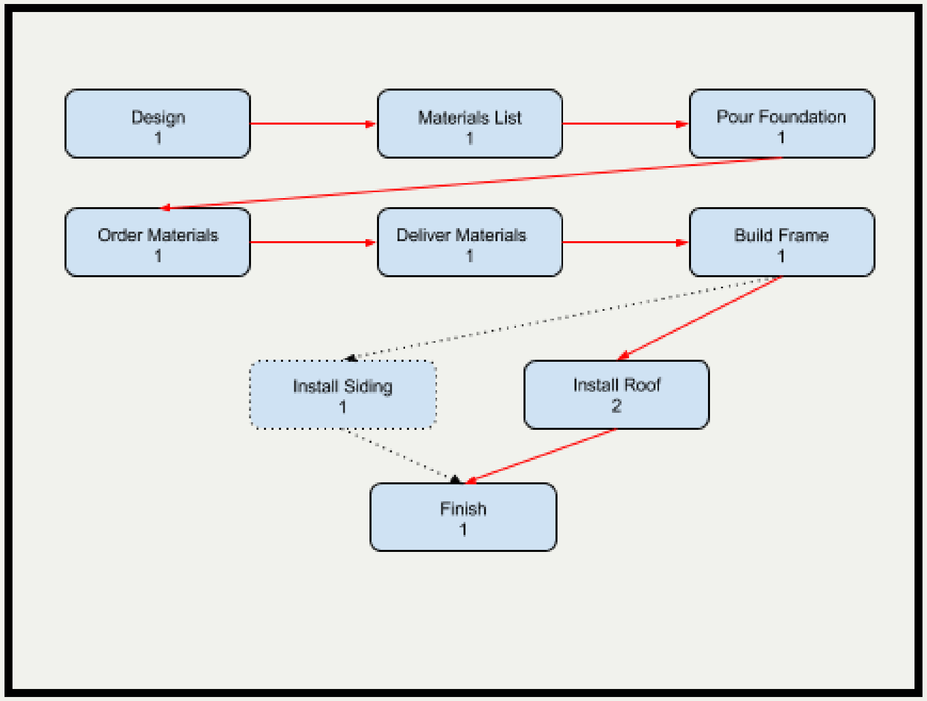
- Network diagrams can be constructed from the WBS, adding dependencies and estimated durations.
- Note that you can have different network diagrams for the same project, depending on your assumptions and approach.

Critical Path



- The critical path is the longest path through the network
- It is the minimum duration of the project assuming infinite resources and accurate estimates
- Here, everything but putting up the siding is on the critical path, and must happen in the order specified.

Slack



- Tasks not on the critical path have *slack*
- The duration by which they can be late without making the project later than the critical path duration

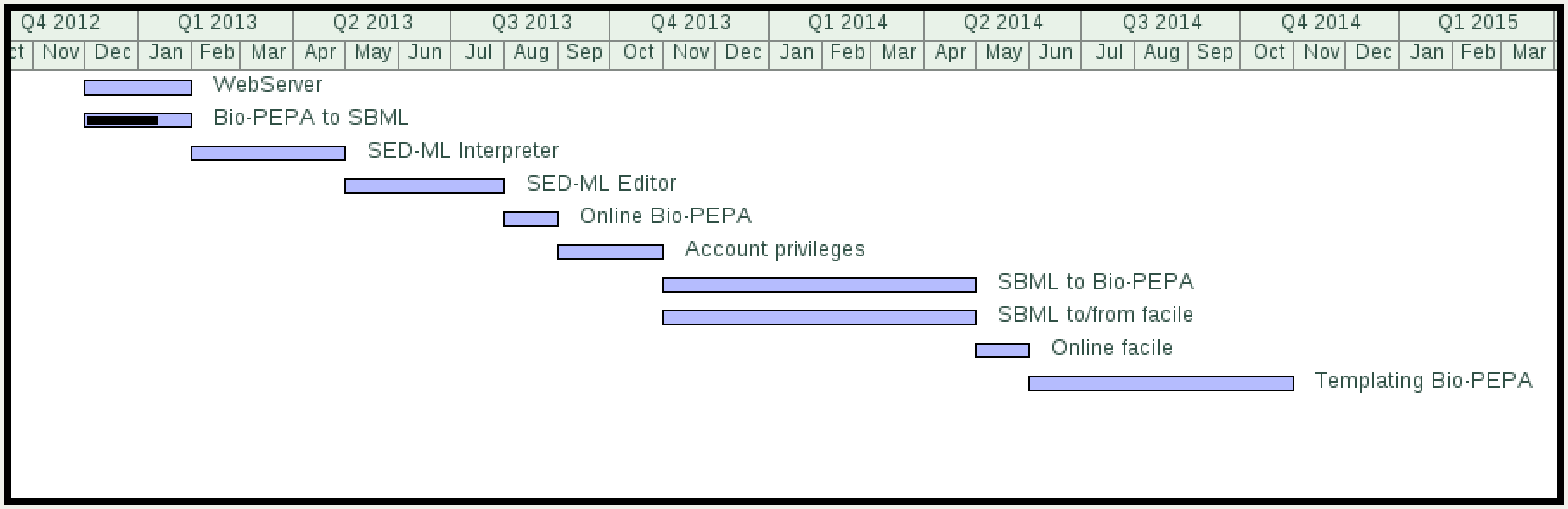
Crashing/Fast Tracking

- If the critical path is still not fast enough, it’s possible to shorten the duration by changing some assumptions.
- **Crashing**: Change the duration of some critical task, e.g., if it is possible to parallelise it by assigning more people to it.
- **Fast tracking**: Allow tasks to be done in parallel by changing the logic in the network diagram.
- If tasks can be parallelised (as in building construction) these allow total duration to be greatly reduced.

PERT/CPM Charts

- Network diagrams come in a variety of flavors with different names:
  - A Program Evaluation and Review Technique (PERT) chart shows dependencies and time estimates, using 3-point estimates.
  - The Critical Path Method (CPM) chart is a related alternative, using single estimates.
- Both show similar information, but use different methods for calculating the critical path and slack.

Gantt Chart



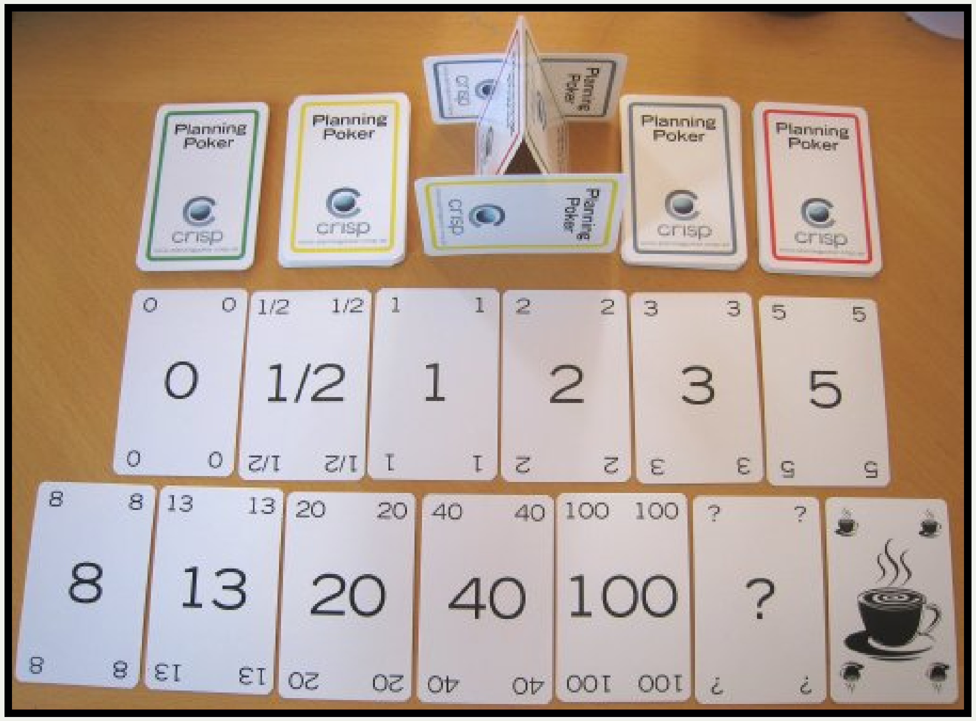
- A Gantt chart shows the tasks and their durations graphically, in calendar form, with one bar per activity.
- The bar shows the earliest start date and expected duration.

Resources

- All these charts assume infinite resources
  - things that can logically happen in parallel, are assumed to happen in parallel.
- In reality, there are limited resources, so there are often many dependencies not shown in the charts.
- E.g. if only one person is available to work on the shed project, the siding and roofing tasks must be done in series, due to the resource constraint.

How do we decide how long each task will take?

- *"Planning Poker"* is a game doing this rounds in development circles
- **There are cards for it:**
- **And a web-based implementation**



Use Project Management for your Software Projects?

- **Why?**
  - Software is just a project like any other, with tasks, interdependencies, resources, etc. Right?
- **Why not?**
  - The vast majority of software development tasks cannot be estimated up front
  - doing so requires knowing the requirements and the design, plus how long debugging will take
  - after those three tasks there's very little work left.

Use Project Management for your Software Projects?

- Clear win for Project Management:
  - Software projects embedded into larger projects.
  - Here, the software must be delivered on time for the rest of the project to succeed, and depends on the other parts. In such cases, project management is very useful.
  - Within the software-only portion, perhaps not.