CS 361
Software Engineering I

Schedule

Oregon State University

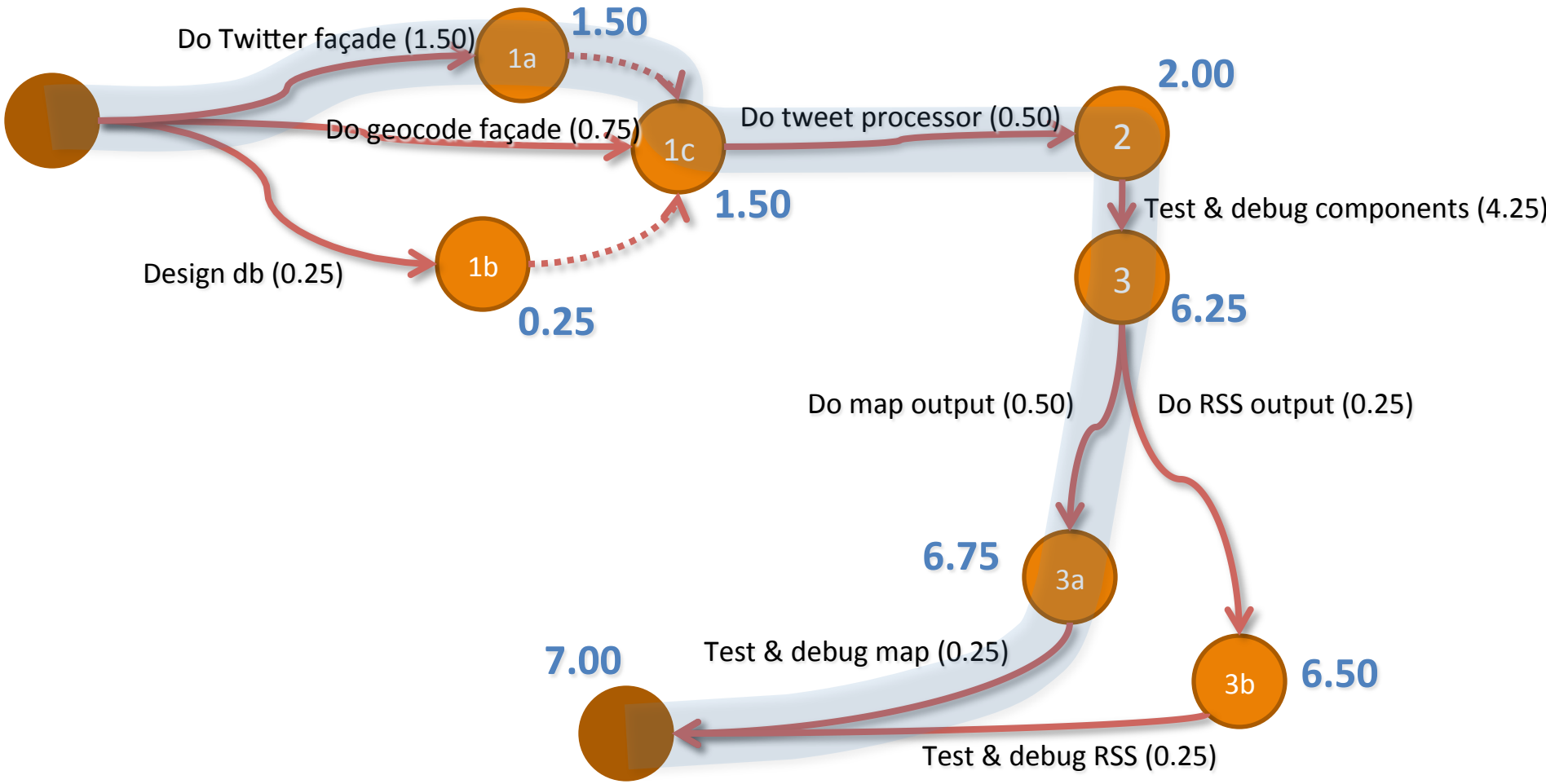
http://www.flickr.com/photos/28481088@N00/315671189/sizes/o/

Critical path: longest route through the activity graph

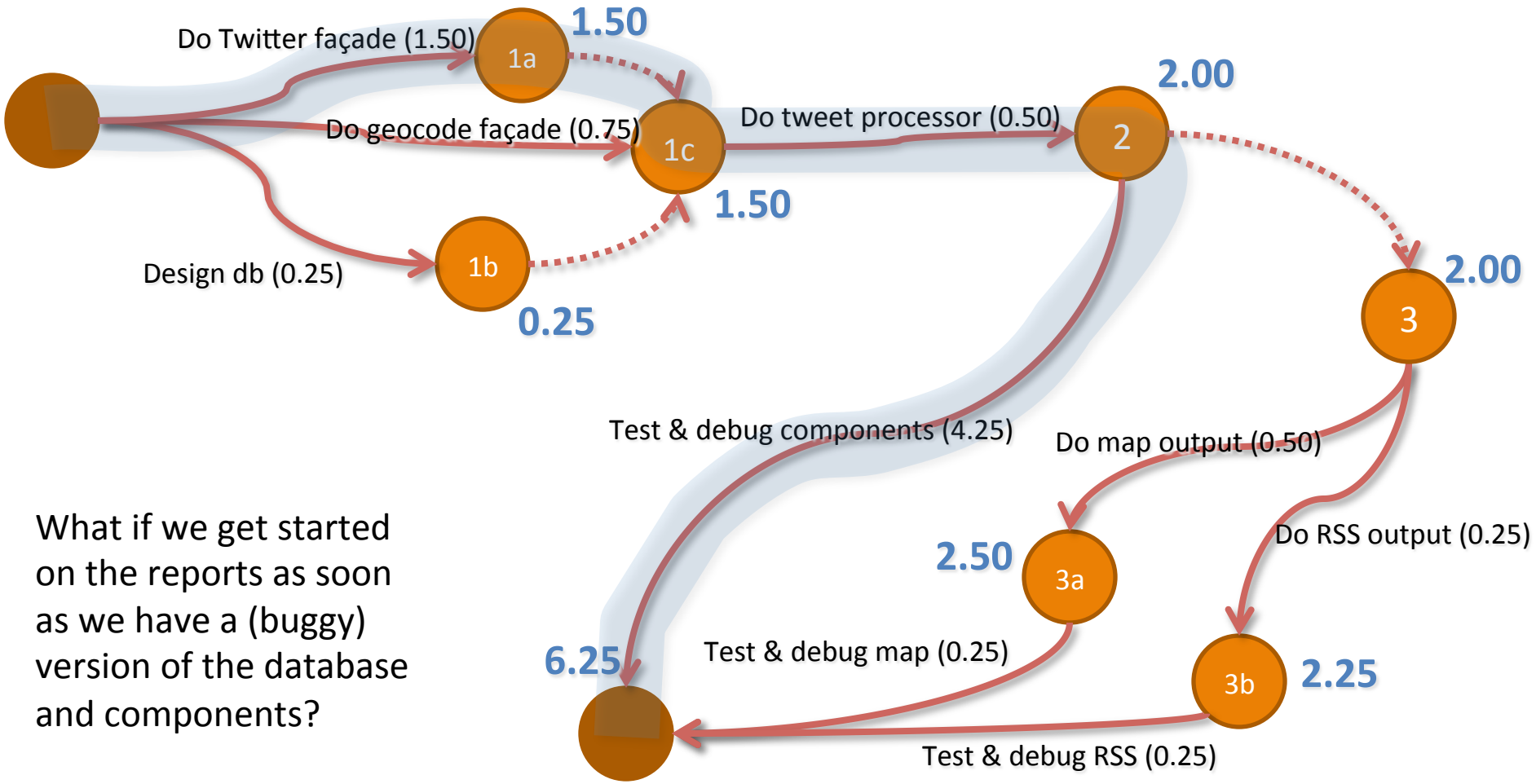
- Sort all the milestones in "topological order"
  - i.e.: sort milestones in terms of dependencies

- For each milestone (in order), compute the earliest that the milestone can be reached from its immediate dependencies

Example: computing critical path



Example: tightening the critical path



What if we get started on the reports as soon as we have a (buggy) version of the database and components?

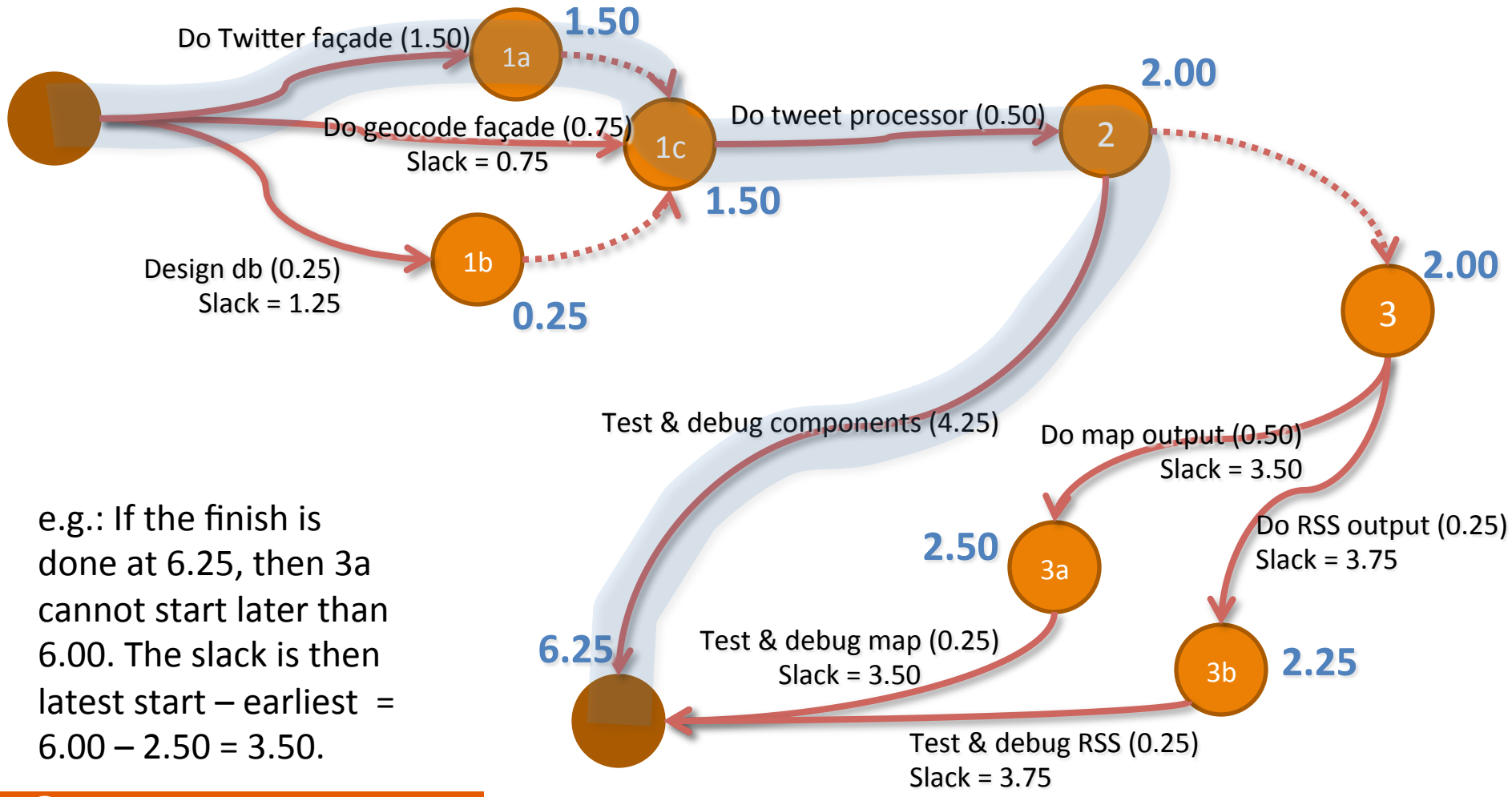
Slack time

- **Activity slack** =
  latest possible start time –
  earliest possible start time

- Indicates how "spare time" that activity has (in case something goes wrong)

- Activities on the critical path always have **zero** slack time

Example: computing slack time



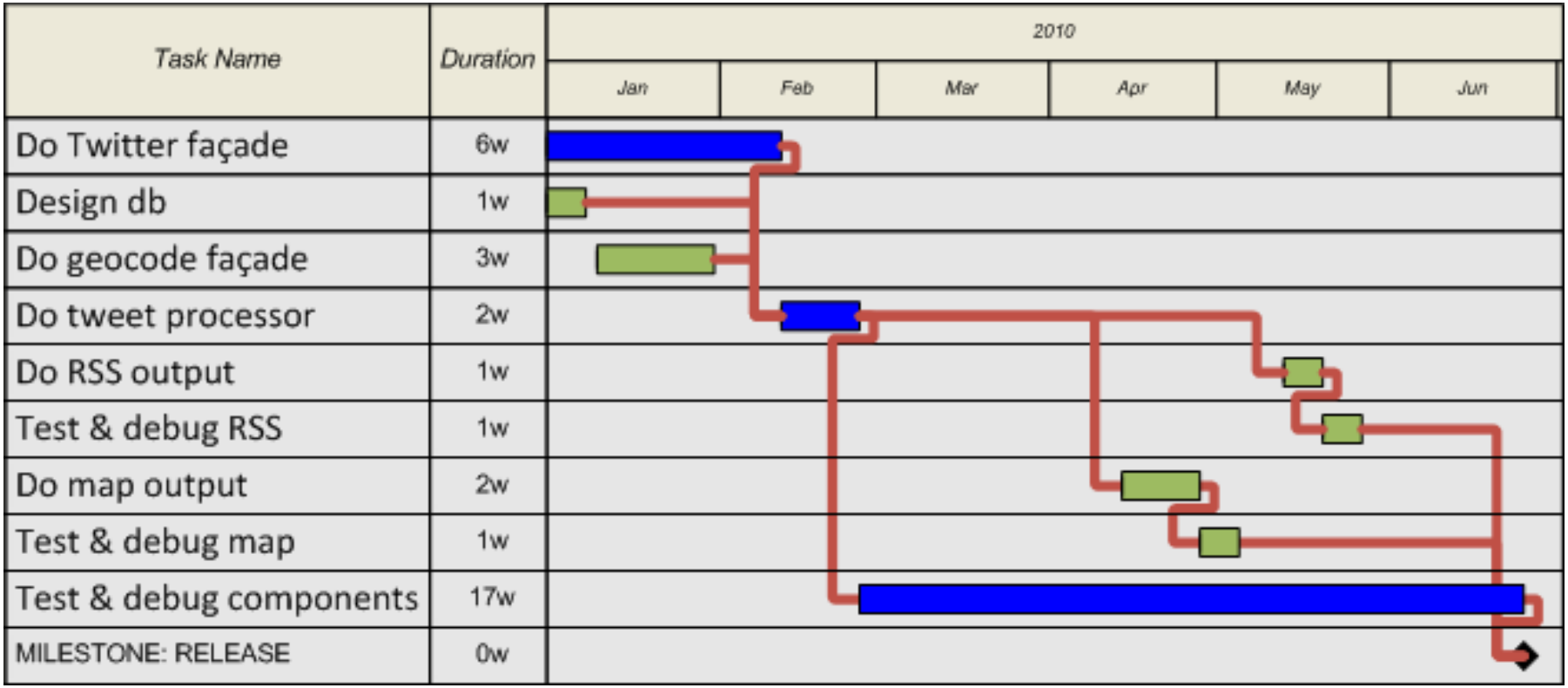
e.g.: If the finish is done at 6.25, then 3a cannot start later than 6.00. The slack is then latest start – earliest = 6.00 – 2.50 = 3.50.

Gantt Chart

- Shows activities on a calendar
  - Useful for visualizing ordering of tasks & slack
  - Useful for deciding how many people to hire

- One bar per activity
- Arrows show dependencies between activities
- Milestones appear as diamonds

Example Gantt chart



Gantt chart quickly reveals that we only need to hire two people (blue & green)

Adding people to the same activity

- To speed up an activity, you can add people
- Be aware of diminishing returns
  - Can 100 people complete a task 100 times faster than a single person?
  - Of course not!
- Decreasing schedule time usually leads to increasing overall project cost
  - Rushing can also harm quality

Compare this lecture to your textbook

- Did you notice that this lecture *started* with a set of requirements and an architecture?

- In contrast, your textbook assumes that you are scheduling *before* you have requirements and an architecture.

- What are the pros and cons of each approach?