1. Important Distinctions
   1. Estimate: honest, realistic assessment of how much resources (time, money,…) securing objective will requirement
   2. Target
      1. Describes descramble goal
      2. Sometimes given, sometimes choose
         1. Sometimes this is mistaken for an estimate
   3. Plan
      1. Describes recipe for how will achieve objective
2. Estimation: why
   1. Need ongoing estimates of time and money
   2. Rough estimation key for
      1. Scoping and go/no-go decision
         1. Very useful for rapid iteration of feature plan
      2. Assessing level of risk involved in project
         1. Risky project requires either re-evaluation or very careful execution
      3. Planning staged deliverables
3. Key considerations
   1. We are very poor at estimating, particularly for large and less concretely described pieces of work
   2. Estimates have strong optimistic bias
   3. Early estimates are used to judge project success
   4. Key issue: when and by whom estimation is conducted
   5. Don’t allow pressure to change estimates
      1. Don’t allow targets to become estimates
      2. Later in class
   6. Try to avoid letting estimates be turned into commitments
4. Activities Commonly Omitted From Estimates
   1. Software-development activities commonly missing from software estimate
      1. Ramp-up time for new team member
      2. Mentoring of new team member
      3. Management coordination/manager meeting
      4. Cutover/deployment
      5. Data conversion
      6. Installation
      7. Customization
      8. Requirements clarifications
      9. Maintaining the revision control system
      10. Supporting the build
      11. Maintaining the automated smoke test
      12. Used in conjunction with daily build
      13. Installation of test builds at user
      14. …
5. Motivation
   1. Breaking down an estimate into pieces fosters better estimate
      1. Clearer conceptualization of each element of work involved
      2. Law of large numbers: relative independence of error in estimates means many errors cancel
6. Decomposition-based technique
   1. Systematic identification of quantities of work and equipment required
      1. Key features sought
         1. Exhaustive
         2. Mutually exclusive
      2. Used to calculate several factors
         1. Cost flow
         2. Resource requirements
         3. Staffing needs
      3. When a work breakdown schedule (WBS) exists, breakdown often uses its taxonomy
7. Decomposition: Gotchas
   1. Optimism
      1. Too optimistic by 20-30
         1. Tendency is for developers to give best case estimate rather than mean/median
      2. Client pressure, pride push towards smaller #s
      3. One way to deal with: provide range estimates (e.g. 90% confidence intervals)
   2. Difficulties estimating 90% confidence intervals
      1. Almost always too narrow
      2. Professional pride? Sense that narrow estimate is better?
8. Avoid Point Decomposition Estimates
   1. Point estimates hide uncertainty
   2. Most developers will tend to give something close to best case estimate
9. Range Estimates: what and why
   1. Range estimates make explicit
      1. Optimistic assumptions
      2. Pessimistic assumptions
      3. Expected case
   2. Explicitly distinguishing these types of estimates
      1. Makes it less likely that developers will accidentally substitute a best-case for expected case
      2. Gives client important information regarding uncertainty in estimates
      3. Decreases risk that an estimate will be mistaken for a commitment
   3. Statistical care needed in transitioning from sub piece range estimates to project ranges
   4. Generally far underestimate width of interval
   5. Sometimes clients react emotionally to wide range of uncertainties
      1. Silver lining: use to motivate investigation to reduce uncertainty
   6. Danger if don’t specify assumption regarding interpretation of best and worst case estimates
10. Going from range for pieces to ranges for entire schedule
    1. Not appropriate: Assuming that best case for entire schedule = all best case for components
       1. Or for worst case
       2. Fundamental reason: vanishingly unlikely that will get best/worst outcome for each and every piece at the same time
          1. More likely, will be off in one way for some estimates, and in another way for other estimates

Scheduling

1. Linkage to Estimation
   1. Estimate often give us a sense of task duration/effort – but not project length
      1. Naively turning estimate into calendar duration assumes that all activities can be done in parallel or must be done in serial
   2. Scheduling allows turning many task durations into an estimate of project length
   3. Scheduling allows understanding of cash flow over time
   4. Given time value of money, scheduling critical to understanding present value of estimate
2. Why schedule?
   1. Key: both lower chance of delay and assists in recovering from delay, resolving responsibility
      1. Delays often result simply from poor planning
   2. Assistance in reasoning about huge number of details
      1. Identify critical task
   3. Resources can be difficult to manage
   4. Formalization necessary but not sufficient for managing
3. Ubiquitous Role of Schedule I
   1. Design (preliminary schedule)
      1. Establish finish, milestone times for choreographing activities
      2. Procurement time, subcontractor sched., rollout
   2. Identify critical path (activities that can’t be delayed without delaying project
   3. Framework for monitoring
   4. Allow understanding of cash flow over time
   5. Importance for thinking through issues
   6. Communication tool between parties
   7. Link to resources
      1. Payments
      2. Resource usage
   8. Role in control
      1. Assessing impacts of changes
         1. Allows demonstration of indirect cost
      2. Legal importance – assigning responsibility for a delay (particularly common in construction)
4. Scheduling considerations
   1. No silver bullet – also essential to
      1. Control requirements
      2. Revise scheduling as progress continues
      3. Get right people involved in estimate
      4. Buy-in by all stakeholders
   2. Want shared schedule
   3. Small projects may not need-but collection does
   4. Include things done outside company
5. Gantt /” Bar Charts” (LOOP)
   1. WWI origin (Systematized earlier work)
   2. Very effective communication tool
   3. Very popular for representation of simpler schedules
      1. Can be cumbersome when have > 50 activities
   4. Key shortcoming: No dependencies captured
      1. No information on how changing start/delivery data of one activity affects other activities
   5. Most effective as reporting format rather than as primary representation for reasoning
6. Critical Path method (CMP) (NON-LOOP)
   1. Origin at Dupont Corporation (1956)
   2. Sometimes narrow term, sometime more general
   3. Directed acyclic graph
   4. Drawn (topologically sorted) left to right
   5. Specify activities and associated information (e.g. duration) and run scheduling algorithm to yield scheduling recommendations/constraints
7. Network methods: basic steps
   1. Define activities from task list
   2. Estimate duration, money, resources for each activity
   3. Define dependencies between activates
   4. Iterate
      1. Perform CPM scheduling
      2. Estimate time, cost, resource usage over project
      3. If acceptable, terminate
      4. If not acceptable, impose dependencies or added/reduced resources
8. Precedence Considerations
   1. Unless impose constraints, assuming that activities may be performed in parallel (but don’t have to be)
   2. Relationship between activities reflect constraints
      1. Resource/financial (e.g. only so many developers to do task)
      2. Managerial (e.g. process flow rule)
      3. Regulatory/contractual (e.g. must finish design before handling it to outsourced contractor)
      4. Physical (e.g. only one occupant of usability lab at a time)
9. CPM Algorithm
   1. Input: Task name, durations, dependencies
   2. Derives early, late finish/start for nodes
   3. Can run on AOA or AON diagrams
   4. That(n) [Linear time] is task count
   5. NB:
      1. ES, EF are earliest could start/finish given other tasks that must do first
      2. LS, LF are latest could start/finish without delaying whole project
         1. These are not the latest that could start having “kept busy” with other activities
10. Passes
    1. Set early start for (each) starting node
    2. Forward pass – determines early start and finish
       1. Because all preceding activities must finish before a successor, early start of a given node is maximum of early finishes of preceding node
    3. Overall project duration (late finish) is defined as maximum of early finishes for node
    4. Backward pass – determines late start & late finish
       1. Because preceding activity must finish before any following activity, late finish of a given activity is minimum of late starts of following activities
11. Float/Slack Fundamentals
    1. Intuitively, measures leeway in scheduling
       1. Degree of freedom in timing for performing task
    2. Difference between when we have to finish activity and earliest activity can be done
       1. LF-EF or LS-ES
       2. Types of float differ in how define have to
       3. Key understanding here: this activity might not be the bottleneck (might now be on critical path)
    3. NB: while it may be possible to schedule an activity at many different points, some points may be far preferable to others
12. Critical Path
    1. Definition: Longest 0-float path
       1. For algorithm as described, at least 1
       2. Must be completed on time or entire project delayed
    2. Determines minimum time for project
       1. Want to consider near-critical activities too
    3. Typically evolves over time, as activity duration unfold
    4. No flexibility to shift for
       1. E.g. resource leveling
    5. Contingency buffering practice common