## Deeper

- What can we do if the critical path takes too long to complete?
  - *perhaps some of the work on the critical path is not actually required (for subsequent tasks on the critical path) and can be broken out into separate sub-tasks (off the critical path).*
  - *perhaps some of the task inter-dependencies can be resolved in other ways, permitting more tasks to proceed in parallel.*
  - *perhaps we can start this work sooner or apply more resources to it.*



## Deeper

- This would seem to contradict Scrum model. Can they be reconciled?

*Scrum probably doesn't disagree with the relative amount of work in each activity.*

*– requirements and design are performed in parallel with development*

*– development and testing are performed in parallel*

*– the tasks are small enough for all phases to be done in a single sprint*



## Deeper

- Why is there a minimum time, below which it is impossible to deliver the project, no matter how much money we spend?

*Adding resources can accelerate work when they improve productivity.*

*Adding people can accelerate work when there are multiple tasks to perform.*

*When a one-person task is performed as fast as possible, adding more people won't help.*

*Nine women cannot have a baby in one month.*



## Deeper

- Why does the cost of the project rise so quickly as we try to accelerate it by adding more people?

*The costs of training, communication and management all rise with the staff size.*

*Having many people work on a single logical task may be very inefficient.*

*Haste makes waste.*



## Deeper

- Why does the cost/effort remain relatively flat in the linear range of the curve?

*This is the range where there are enough independent tasks to reasonably employ more people … so that the work could reasonably be done by 2 people in 4 weeks, or by 4 people in 2 weeks.*

*This is the range in which project planning tools can reasonably operate.*



## Deeper

- If this is true, how can a 4-7 person Scrum team be right for every project?

*(a) Because a great deal of work went into breaking the work into tasks appropriate for a Scrum team.*

*(b) Because Scrum is not a huge project methodology, and would probably fall in the critically understaffed range of such projects.*

## Deeper

- What might we do if someone had encountered, and solved a problem?
  - *See if it had an impact on the schedule, or if it suggests a change to our estimates and or plans.*
- What might we do if someone had encountered a problem, and not gotten passed it?
  - *Get them some help.*
  - *Attempt to assess the impact to our schedule.*
  - *See if it suggests a change to our plans.*



## Deeper

- Is the engineering being dishonest? Is there any reasonable sense in which the work might actually be ¾ done?
  - *It is quite possible that ¾ of the code is working, or that we are ¾ of the way through the task list.*
  - *The problem is that ¾ of the code does not represent ¾ of the work. This is the misunderstanding that needs to be resolved.*



## Deeper

- Is there a second, completely independent problem in this scenario?
  - *Did the manager finally just notice that no progress has been made in six weeks?*
  - *If the work is proving to be harder than expected, the issues should have been raised and new plans devised six weeks ago.*



## Deeper

- Give an example of non-specific or non-measurable milestones?
  - *non-specific:* *system testing*
  - *non-measurable:* *satisfactory throughput*
- Why are these characteristics important?
  - *If we are not sure exactly what a milestone means, we may not be sure whether or not we have achieved it.*
  - *If we cannot measure a milestone, we may not have an objective basis to determine whether or not we have achieved it.*



## Deeper

- What is the difference between work and progress?
  - *Work is related effort expended. Progress is related to goals achieved. All effort does not result in progress.*
- Why is it important that milestones be tied to progress, rather than work?
  - *Because our goal is not to do work, but rather to achieve goals.*



## Deeper

- Why are earning rules used? Why not just accrue value at task completion?
  - *If the task granularity is too coarse, earning events may be too infrequent to enable fine grained progress tracking.*
  - *Earning rules attempt to recognize that there is starting and progressing through a task does represent an increase in earned value.*

## Deeper

- Suggest some reasonable progressive earning rules for a single component?
  - *% of planned test cases passed*
  - *% of planned routines implemented*
  - *all planned modules designed, reviewed, coded, unit-tested, integrated*



## Deeper

- What is the danger of progressive earning rules?
  - *EVA is supposed to produce objective measures of progress. Completion of a task is (hopefully) measurable. Anything less is subjective, and the value of the progress is speculative. We could earn progress points and still never finish.*
- Is there an alternative?
  - *Further sub-divide the defined tasks into smaller tasks, with objectively ascertainable completions.*



## Deeper

- What does it mean to say these are a linear measure of progress?
  - *Completed points are a linear function of progress towards completion.*
- What would have to be true to make these points a linear measure of progress?
  - *Our estimates average out to being equally good/bad over all estimated tasks.*



## Deeper

- How does a waterfall project try to manage poor or unstable requirements?
  - *by elicitation, analysis, validation, and review*
- How does an agile project manage poor or unstable requirements?
  - *by involving the product owner in the process and regular delivery of successive implementations to users for feedback.*



## Deeper

- How does a waterfall project try to prevent poor estimates?
  - *By a great deal of design and review, estimation at the lowest possible level, competing estimates, … but it is hard.*
- How does agile try to prevent poor estimates?
  - *By only estimating small, well understood tasks.*



## Deeper

- How does a waterfall project attempt to manage scope creep?
  - *The product owner must tightly manage the requirements.*
- How does an agile project attempt to manage scope creep?
  - *(a) requirements are allowed to evolve*
  - *(b) the product owner is responsible for managing the requirements*

## Deeper

- How does a waterfall project try to prevent unpleasant surprises during construction?
  *by design reviews and prototyping of risky issues*
- How does an agile project try to prevent unpleasant surprises during construction?
  *by reviewing designs during the sprint planning and prototyping alternative approaches*



## Deeper

- How does a waterfall project try to ensure code quality and correctness?
  *by design reviews and well planned testing.*
- How does an agile project try to ensure code quality and correctness?
  *by design reviews and well planned testing*



## Deeper

- How can a waterfall project avoid unpleasant surprises during the integration phase?
  *early or continuous integration*
- How does an agile project attempt to avoid unpleasant surprises during the integration phase?
  *continuous integration*



## Deeper

- How does a waterfall project attempt to prevent problems w/external dependencies?
  *by analyzing the risks and developing management and contingency plans.*
- How does an agile project attempt to prevent problems w/external dependencies?
  *by not starting a task until all external dependencies have been satisfied*



## Deeper

- How does a waterfall project attempt to manage unplanned interruptions?
  *detect them during status review, and when they come up, develop a new plan (or slip)*
- How does an agile project attempt to manage unplanned interruptions?
  *detect them during stand-up, deus ex machina may do something about them, and our velocity goes down (revising extrapolated end dates)*



## For Next Lecture

- McConnell, section 28.4, 34.8
  - general thoughts on measurement & improvement
- Spolsky: Stealth Process Improvement
- Wikipedia
  - Best Practices Benchmarking
  - Process Improvement
  - Capability Maturity Models
  - Six Sigma
  - ISO9000
  - Software Metrics
- Kaner – Software Engineering Metrics
  - 12pp, good analysis of problems w/software metrics

## Deeper $A_1$

- What is the advantage of breaking a project down into tasks and sub-tasks?
  *Tasks are natural units of work, estimation, and assignment.*
- What is the advantage of breaking down goals, over breaking down tasks?
  *We might not yet have defined enough tasks to achieve the goal. Breaking down goals is less likely to make this mistake.*



## Deeper $A_2$

- Give examples of a process-centric work breakdown?
  *inception, requirements, design, coding, unit testing, system testing, support, …*
- Give examples of a product-centric work breakdown?
  *web front-end, applets, database, help, installation, management, …*



## Deeper $A_3$

- What is the problem with top-down evolution of a hierarchical decomposition?
  *As with all top-down design, there may be details (discovered in lower levels) that change requirements/estimates and create new dependencies for other tasks.*

  *A schedule is a living document!*



## Deeper $B_1$

- Is there a better alternative to saying that design can start one month after architecture?
  *Enumerate sub-tasks within the architecture task, and predicate the design activities on the associated architectural sub-tasks.*



## Deeper $B_2$

- Why do we call this the critical path?
  *This is the longest path in the project, and the one that will most greatly limit the speed with which the project can be accomplished.*

  *This is also sometimes referred to as "the long pole in the tent". If you can't get this one up, the others don't matter.*



## Deeper $G_1$

- How does flat (vs. natural) staffing affect these schedule?
  *Early resources up front can't be used.*
  *Insufficient resources at the peak stretch out the schedule.*
  *Extra resources at the end, help make up for the work that couldn't be done earlier, but may turn into waste soon thereafter.*
  *Plus … the extra resource may be the wrong ones.*

## Deeper $G_3$

- Does this curve tell us anything about the first two weeks of Project 2?
  - *We had lots of people available, but couldn't make good use of their time because we didn't have detailed plans for what everyone was supposed to do.*



## Deeper $G_4$

- Why does worker efficiency go down on the right of the linear region?
  - *The project may be under-staffed:*
    - *team may not have experts in required areas*
    - *team members cannot pace or help one another*
    - *activities may be under-managed*
  - *The project may be over-staffed:*
    - *communications overhead reduces efficiency*
    - *inter-dependencies introduce delays*
    - *more people mean more problems*



## Deeper $H_1$

- Why do we need to ask what tasks people are working on?
  - *The work-breakdown and sub-task definitions, are (like estimates) imprecise.*
  - *They may be working on things that had not been included the plan … in which case the plan may need updating.*
  - *They may have been distracted by an interruption … in which case our priorities must be reassessed or reemphasized.*



## Deeper $H_2$

- Why do we ask for peoples' opinions on when they expect to be done?
  - *So that we can project an expected completion date for these activities, and determine whether or not we are still on schedule.*
  - *So that we can identify problems and ensure that they are effectively addressed as soon as possible.*



## Deeper $D_3$

- What does it mean for milestones to be "evenly spaced measures of work"?
  - *Milestones should be achieved on a regular basis (e.g. daily or weekly).*
- Why does this matter?
  - *If it is a long way between milestones, it might take us a long time to realize that we are not making progress.*



## Deeper $D_4$

- What are "inch-pebbles", how do they differ from "mile-stones", and why are they
  - *Inch-pebbles are much smaller than mile-stones. They are much finer grained measures of progress.*
  - *Finer grained tracking enables problems to be identified and responded to quickly.*
  - *If trackable events are a month apart, we could be a month behind by the time we get our first warning of a problem.*

## Deeper

- What Scrum and EVA have in common
  - *linear measures of progress*
  - *enable plan vs. actual comparison*
  - *based on specific, measurable milestones*
  - *earning rules based on initial estimates*
- How Scrum and EVA differ
  - *developers can't estimate dollars*
  - *points are self-calibrating*
  - *Scrum does progressive estimation*



## Deeper

- Give examples of "classes of people with specific skills".
  - *U/I designers, C programmers, COM+ programmers, usability testers*
- Are members of these classes really interchangeable?
  - *No, but we can often plan as if they were.*
  - *If it makes a difference, call them distinct resources, and schedule them individually.*



## Deeper

- How would you define a critical resource?
  - *One that forces the serialization of otherwise parallel tasks, and lengthens the critical path.*
- How would we fix such a problem?
  - *Get more of that resource.*
  - *Re-plan the work so we need less of it.*
  - *Schedule (and multiplex) that resource for maximum effectiveness, and watch it very carefully.*



## Deeper

- What happens to earned value progress measurement if our planned value chart greatly over-estimated our productivity?
  - *Earned Value remains a valid measure of progress (what fraction of the work is complete) but it will show us to be both behind schedule and over budget.*



## Deeper

- Can earned value greatly mis-estimate our progress towards completion?
  - *Only if we greatly mis-estimated the relative difficulties of tasks.*
  - *As long as we get the relative value (ratios) right, earned value will reasonably track our progress towards completion, and merely show us to be over or under (estimated) cost of construction.*



## Deeper

- What happens to earned value if the scope (and hence value) of the project increases?
  - *Our percentage complete falls correspondingly.*
  - *Our rate of work completion measures are unaffected.*
  - *Our tracking vs. budget is also unaffected (assuming the budget is increased to handle the added work).*

## Deeper

- Is there any fundamental reason that Earned Value must be computed in dollars?
  - *Dollars are a natural unit, in that both work and products can usually be given a dollar value.*
  - *The methodology, however, is not tied to particular units. Any consistent unit that can be used to describe resource expenditures can be used (e.g. staff-months, lab-months, quatlu's, etc).*



## Deeper

- Why can automated scheduling tools only be used within the linear range of the PNR effort/time curve?
  - *The increased overheads in the over-staffed and under-staffed ranges result in reduced productivity, which would necessitate adjustments to the task size estimates.*