Discussion Slides

Deeper

- How do we know that we are ready to move from the definition to the planning phase?

In general, we can start the planning process as soon as our requirements seem to be credible and stable (converged).

In novel projects, there may be many iterations of definition, requirements, and preliminary planning before real commitments are made.

Deeper

- How do we know that we are ready to move from the planning to the construction phase?

In general, we can start the construction process when we believe the plans to be adequate (to meet the requirements) and complete.

A good test for completeness is whether or not the people who will do the work understand what is to be done and believe that they how to do it.

Deeper

- Are there situations that would seem to demand parallel development?

Parallel development is the first choice when the schedule requires work to be completed faster than would be possible with sequential scheduling.

If there are multiple independent teams available to work on the project, this too would seem to be begging for parallel development.

Deeper

- Why might smaller teams be more efficient?

Better communication and coordination, fewer misunderstandings and conflicts.

- Why might smaller tasks be lower risk?

A smaller task is more easily understood, and problems are more easily anticipated.

- How might this improve efficiency?

Lower overhead for communication, and fewer mistakes.

Deeper

- What would make parallel development (for nominally independent tasks) difficult?

When multiple tasks require the same resources (typically people, but possibly labs or special equipment). This doesn't preclude parallel development, but necessitates careful scheduling.

Deeper

- Why does parallel development often lead to problems when the pieces are combined?
  - *The requirements and interface definitions for the independently developed components may have been ambiguous, and differently interpreted by the different teams.*
  - *Problems encountered in the implementation of one component may require changes to the interfaces of another component to fix.*

Deeper

- How can design enable or preclude parallel development?
  - *If the interfaces between components are simple and clean, it may be easy to develop them independently.*
  - *If the interfaces and interactions between components are complex, it may be necessary to develop them together.*

Deeper

- Give an example where open requirements would not preclude the start of planning?
  - *All requirements don't affect all components. We don't need complete requirements for A before we can start B.*
  - *Some requirements have few design implications. We may consciously choose to delay the fixing of those requirements (to get started sooner, or to give us more time to develop the right requirements).*

Deeper

- How do you know if it is safe to start phase n+1 before phase n is complete?
  - *You must be able to "put a fence around" the missing information.*
  - *This means you need a complete understanding of what information is missing, and some estimate of the range within which it will fall.*
  - *These become the assumptions upon which the phase n+1 work is predicated. They must be monitored carefully.*

Deeper

- How might an action in the design of component A invalidate work done in the implementation of component B?
  - *We allowed B's implementation to move forward because we believed it to be independent of the design of A. In the process of designing A, we found a problem, that could only be fixed by changing the specifications of component B.*
  - *We were wrong when we said that the design of B was independent of the design of A.*

Deeper

- Is it possible to thoroughly test one component, if components with which it interacts are still incomplete?
  - *It is often possible to simulate missing components in order to test the existing ones. Such simulations can actually provide better testing than the real components would.*
  - *This does not happen by accident, but must be anticipated when the high level architecture is created.*

Deeper

- What kinds of things could you reasonable defer to a later release?
  - *advanced features that require more time*
  - *quality or performance enhancements*
  - *support for new protocols or technologies*

  It boils down to a question of whether or not you can satisfy user demands with a subset of the proposed work.

  A release doesn't have to do everything, but it does have to be useful.

Deeper

- Examples of useful subsets
  - a home music server could be a very successful product, even if advanced features like "go find me other music I will probably like" are not available in the first release.
- Examples of useless subsets
  - a home music player that doesn't support mp3 or whose web interface doesn't work with Internet Explorer is a non-starter.

Deeper

- If you can choose what subset you want to build, doesn't that guarantee that your product will be buildable?

  There may be a hundred different features from which you can choose ... but some of those features may be absolute requirements for success. If only one of those requirements is something you can't build, then you cannot build a viable product.

  This could happen if there are already products in the market, and you can only enter the market if you provide a capability that the existing products lack.

Deeper

- How can we fund successive incremental approximations towards our ultimate goal?

  If we are developing a commercial product, we may be able to sell release n*, and generate a revenue stream that justifies investment in release* n+1*.*

  If we are doing research, we may be able to use results obtained from version n *to bolster our claim that we can obtain even greater results from version* n+1*.*

Deeper

- How could succesful release n create support for release n+1?

  By attracting customers and proving the demand for the envisioned product.

- How could successful release n erode support for release n+1?

  It may turn out that release n *solves the customer's needs, and there is no demand for the enhanced features planned for* n+1*.*

Deeper

- What if first release is not well received?

  Perhaps we chose the wrong feature subset.

  Perhaps we implemented it poorly.

  Perhaps we misunderstood the goal or market.

  Perhaps this is evolution in action, and we should give up on this product and move on to something more productive.

  We learned this lesson sooner, and at a lower cost than we would have, had we tried to build the whole thing.

Deeper

- Why would anybody ever choose option A?
  - *The situation doesn't allow enough time to do option B.*
  - *They don't understand the unknowns well enough to plan attacks on them.*
  - *They lack the experience and courage to "explain things" to people higher up the food chain.*

Deeper

- Why might we have to use a spiral of product simulations to clarify user requirements?
  - *We are creating a new service or appliance. We think we know what users will want to do, but we don't know how proposed interfaces will work out, or what operations they will need that we haven't yet thought of.*

Deeper

- Why might we have to use a spiral prototyping cycles to reduce technical risk?
  - *We are designing something complex and novel. We have a lot of ideas, but we honestly haven't got a clue how they are going to work. We should prototype some of our proposals to see how difficult they are and how well they work before committing to a particular design.*

Deeper

- The advantages of agile/iterative?
  - *Getting more frequent feedback results in fewer mistakes being made, and enables mistakes to be found and fixed sooner. It also aborts wrong directions sooner, saving considerable effort … and allowing that effort to be spent on features that will have value.*

Deeper

- The advantages of agile over iterative?
  - *They are not huge because agile methods are iterative. But the key characteristic of agile methods (that might not be present in other iterative approaches) is a high degree of customer involvement … and more feedback is likely to yield a better result.*

Deeper

- What might happen if management was promised an incremental release model, but the project was actually a spiral?
  - *This happens all the time. Engineering is going through prototyping cycles (whether they know it or not) and management is expecting a product to be released soon.*
  - *Such projects usually fail when management cuts off the funding for them.*

Deeper

- What could happen if management planned on concurrent development, but did not bother to tell the designers about this?
  - *Concurrent development assumes a high degree of independence between the components. If the designers were not told of the importance of such independence, they might adopt a design that gained other advantages at the expense of reduced independence.*
  - *This would preclude successful concurrent development efforts.*