Discussion Slides

Deeper

- What are “new technology” problems?
  New technologies are, by definition, not mature. Their capabilities may not be well understood, and the tools may be flaky.
- Can “programmers” solve them?
  Not in the short term. The risks must be assessed, through research and prototyping, and plans must be developed to mitigate those risks. These are design, planning, and management problems.

Deeper

- “technological illiteracy” problems?
  When people who do not understand the capabilities, implications, or limitations of a technology attempt to base products on it.
- Can this be solved by “programming”?
  The problems are architectural, because the product (as specified) cannot work. These problems call for architectural solutions.

Deeper

- What does “technological incompetence” encompass?
  Architecture, Design, Coding, Debugging, Testing, and Support.
  Developer productivity is a function of experience, tools, and the problem.
  Often people claim that their programmers are not productive enough, when the real problems are poor design and planning.

Deeper

- How does Brooks define “accidental complexity”?
  Difficulties and complications that arise from not having the right tools for the job?
- Examples?
  - *domain appropriate programming language*
  - *more powerful debugging tools*
  - *automated testing frameworks*

Deeper

- What does Brooks consider to be the “essential complexities”?
  Complexity that cannot safely be abstracted away.
  The specification, design, and testing things involving the interactions of algorithms, function invocations, and related data items.
  The problem is not “representing” these things. Tools and languages can handle that. The problem is ensuring the relationships are correct, and that the specifications properly capture the essential relationships.

Deeper

- What is the fallacy in:
  We've already got enough policies

There are indeed many policies: some good, some bad, some not used often enough, some used inappropriately.

The mere fact that you are tired of going to school does not mean that you have learned everything you need to learn.

Deeper

- What is the fallacy in:
  Process takes more time than it saves.

A process that prevents mistakes can easily save time. Processes that add overhead without saving time (or otherwise improving the outcome) are inappropriate processes.

Deeper

- What is the fallacy in:
  You'll know how long it will be when I finish

If you don't have any idea how long it will take to finish a job, it is probably because you haven't yet figured out what work you have to do. Doing a complicated job without a plan usually ends badly.

Deeper

- What is the fallacy in:
  If it's late, we can add more people.

That trick never works:
Can you find people with the right skills?
How long will it take to get them on-board?
Training them slows down the work?
9 women cannot have a baby in one month!

Deeper

- What is the fallacy in:
  It works for me, ship that sucker.

If software has complex functionality, we need a plan to correctly ascertain whether or not it is, in fact, working.

Deeper

- What is the fallacy in:
  Once it's done, we can see how good it is.
  If it has bugs, we'll find and fix them.

It is much easier and faster to do it right than to do it wrong and fix it later.

If you don't have a plan for success, you have a plan for failure.