Lecture 27
20th Anniversary Reflections

Kenneth M. Anderson

Foundations of Software Engineering

CSCI 5828 - Spring Semester, 2000

Today's Lecture

- Discuss Brook's reflections on the Mythical Man-Month, 20 years after he wrote it.

Why a new version?

- The book was still being read
  - 20 years later Brooks was still receiving comments and questions from readers
  - In the Reflections chapter, he consider what is still right, what is wrong, and then examines current trends in Software Engineering
    - Brooks himself has moved on; his most recent research is in virtual environments

Conceptual Integrity

- "Today I am more convinced than ever. Conceptual integrity is central to product quality."
  - He cites having a single architect the most important step towards achieving this goal
  - He mentions that in his software engineering class with teams of 4 people, he insists that each team select a manager and an architect

The Architect

- He still believes in having one person in charge of the design goals and architecture for a system
  - The architect is the user's champion
  - He discusses "recursion" with architects
    - For large systems, additional architects should be assigned to work on subsystems. The chief architect is still in charge of the overall system

The User

- Brooks focuses on the user
  - microcomputer revolution has spawned large, hard-to-characterize user sets for applications
    - Unlike the contract software systems of the 70's
    - It's much harder to design a general-purpose tool
    - Featuritis is a problem (Word 6 for the Mac)
  - Brooks argues for the need to define the user set
    - He says "its better to be explicit and wrong than ambiguous"
    - He recommends using probability distributions about how many users will have a particular characteristic; then base design decisions on these probability distributions

The Second System Effect Gotcha

- The Mythical Man-Month recommended
  - avoiding architects on their second systems
    - an architect must be extra disciplined on his/her second system
  - and throwing out your first system
- Well, which is it?!?
  - With respect to the former
    - Brooks was speaking of the second "fielded" system
  - With respect to the latter
    - Brooks was speaking of rapid prototypes

Graphical User Interfaces

- Brooks considers the WIMP interface to be a "triumph" (but ultimately sides with speech)
  - He uses the Macintosh as an example
    - Although he correctly cites the work of Doug Englebart from Stanford and Bob Taylor at Xerox PARC as the originators
  - Conceptual Integrity achieved in the metaphor
  - Seamless support for novice and skilled users
  - Two Mice (Failed to mention Doug Englebart's chording device)
  - GUI in ROM (Direct Incorporation)

Critiquing the Waterfall

- Brooks says "The Waterfall is Wrong!"
  - Actually he points out that the many variations that had sprung up in response to the original waterfall showed that it was considered wrong back in the 70's
  - It needs
    - User involvement and Feedback
    - Incremental Development

The Man-Month Revisited

- Brooks sites a study by Boehm
  - of 63 software engineering projects
- that confirms the fallacy of the Man-Month as a productivity measure
  - T = 2.5(MM)^1/3 (Time to ship)
  - "Hardly any projects succeed in less than 3/4 of the calculated optimum schedule, regardless of the number of people applied!
- Brook's Law has also stood the test of time

Other topics

- Power of People
  - Team Fusion
    - Moving projects causes them to start over
- Power of Giving up Power
  - Delegate Power down the org. chain
    - Empowers teams; improves morale
- Millions of computers
- Shrink-wrapped Software

Buy *and* Build

- Components raise the level of abstraction
  - MetaProgramming
    - 4th Generation Languages; Scripting Languages
- Attacks the Essence
  - Components provide richness of function, shorter development time, tested components, better documentation, and lower cost
- Four types of Users
  - as-is, single-application metaprogrammer, external function, metaprogrammer developer