

## "Pre-Cambrian" Software Developments

- Stored-program concept (Turing/ACE, von Neumann/EDVAC)
- Very primitive control and data structures
- Software tools: "initial orders", program notations, post-mortem dump, libraries, "open" and "closed" subroutines (e.g. Wilkes/EDSAC)
- Interpretive schemes (e.g. Bennett/EDSAC, Hill/CSIRAC)

1

## "Cambrian" Software Developments

- "Autocoding"—embryonic programming languages
- List processing (e.g. IPL)
- First true programming languages, e.g. FORTRAN, LISP, ALGOL 60, COBOL (and compilers)
- Beginnings of operating systems

2

## The Software Crisis

- More ambitious projects, and more resounding failures (e.g. Hoare, "The Emperor's Old Clothes", F. Brooks)
  - Structured Programming (Dijkstra, Hoare, Wirth...)
  - Software Engineering (Brooks, Boehm, Parnas, Mills...)
- Lead to

3

## The Mythical Man-Month: The Book

- Fred Brooks, leader of IBM's OS/360 project, 1964–1965, later University of North Carolina, Chapel Hill
- Classic collection of essays (1972), drawing from this experience

## The Mythical Man-Month: The Essay

Projects go awry for lack of calendar time—Why?

- Poor estimating
- Confusing effort with progress
- Lack of stubbornness
- Poor progress monitoring
- Slippage: "Add manpower"

*Like dousing a fire with gasoline*

4

## The Mythical Man-Month: The Essay

- Optimism
- The Man-Month
- Systems Test
- Gutless Estimating
- Regenerative Schedule Disaster

6

## Optimism

- All programmers are optimists
- All will go well
- Each task will take only as long as it "ought" to take
- Three stages of creativity: idea, implementation, interaction
- Software "an exceedingly tractable medium" (or seems to be)
- Vanishingly small probability that everything will go well

7

## The Man-Month

- The "man-month" measures cost, not progress
- Applies only when task is partitionable with no communication (Fig. 2.1)
- "The bearing of a child takes nine months, no matter how many women are assigned"—sequential tasks, e.g. debugging (Fig. 2.2)
- Communication costs, up to  $O(n^2)$ , or worse (Figs 2.3, 2.4)

8

## Systems Test

- Sequential and unpredictable (optimism strikes again)
- Brooks's Recipe
  - 1/3 planning
  - 1/6 coding
  - 1/4 component test
  - 1/4 system test
- Severe effects of slippage during testing: staffing, delivery, psychology...

9

## Regenerative Schedule Disaster

- Example project, with milestones (Fig. 2.5)
- Slippage on Part A (Fig. 2.6)
- Options?

1. Keep deadline, add staff (optimistic)
  2. Keep deadline, add staff (pessimistic)
  3. Reschedule—"Take no small slips"
  4. Trim, intentional or unintentional
- Cost of training and interaction

10

11

## Outless Estimating

- Turning up the heat on an omelette
- Difficult position of software managers
- Sharing of data and improved estimating techniques

## Brooks's Law

*Adding manpower to a late software project makes it later.*  
(oversimplifying outrageously)

12