The Mythical Man-Month

Frederick P. Brooks, Jr.

Tony Priddy

April 21, 2009

Agenda [1]

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

Agenda [2]

- Calling the Shot
- Ten Pounds in a Five-Pound Sack
- Optimization
- The Documentary Hypothesis
- Plan to Throw One Away
- Sharp Tools
- Conclusion
- Questions

Introduction

- Brooks
  - Professor of Computer Science UNC at Chapel Hill
  - Development and design PM of the OS/360 project
  - Served on National Science and Defense Science Boards
- The Mythical Man-Month
  - Originally published in 1975, republished in 1995

Overview

- Premise
  - Division of labor creates different problems in large programming projects than in small projects
- Conceptual integrity is vital to the process
- Determination
  - Integrity is achieved through exceptional design
  - Implementation is achieved through well-managed effort

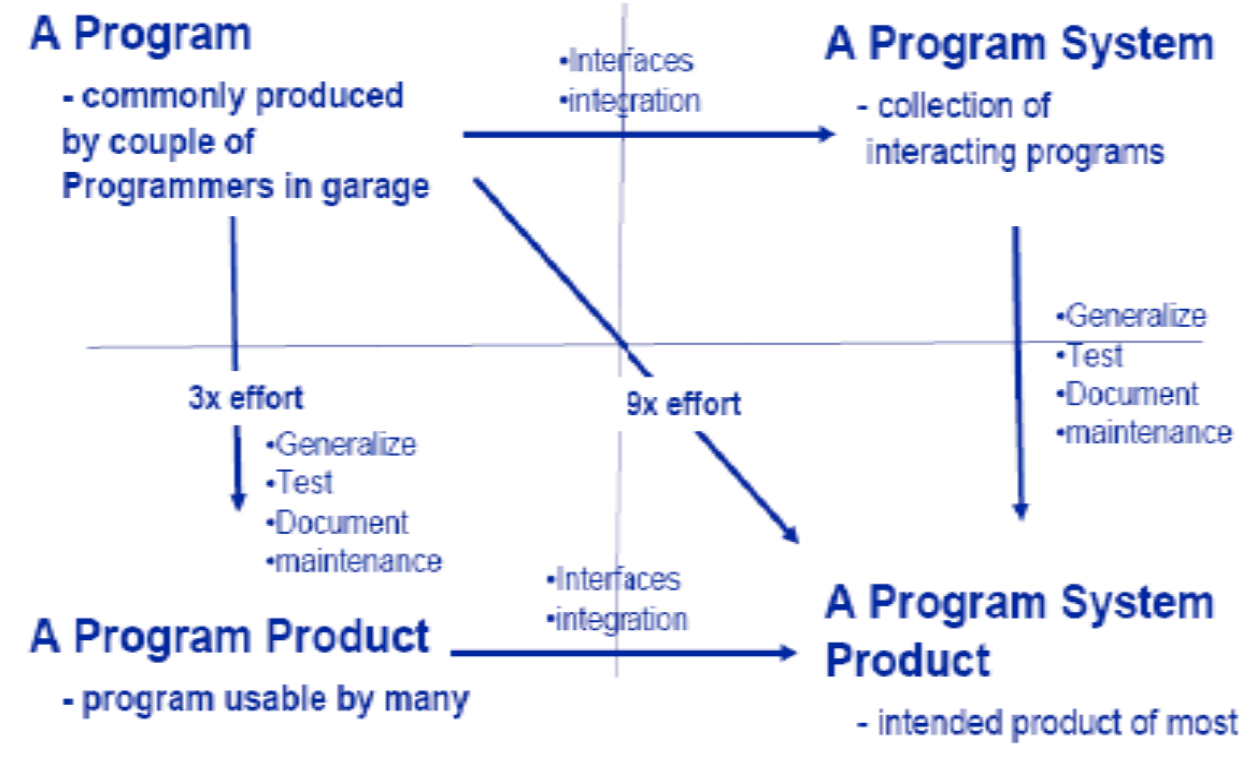
Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

The Tar Pit

- The more you struggle, the more entangled you become
- System programming mimics this struggle
- No one factor appears to be the issue
- Three elements to provide explanation
  - Programming Systems Product
  - Joys of the Craft
  - Woes of the Craft

The Tar Pit
Programming Systems Product



The Tar Pit
Joys of the Craft

- Why do we do this?
  - Joy of making things
  - Making things that are useful to others
  - Fascination with making complex puzzle-like objects
  - Joy of always learning
  - Delight in working in such a tractable medium

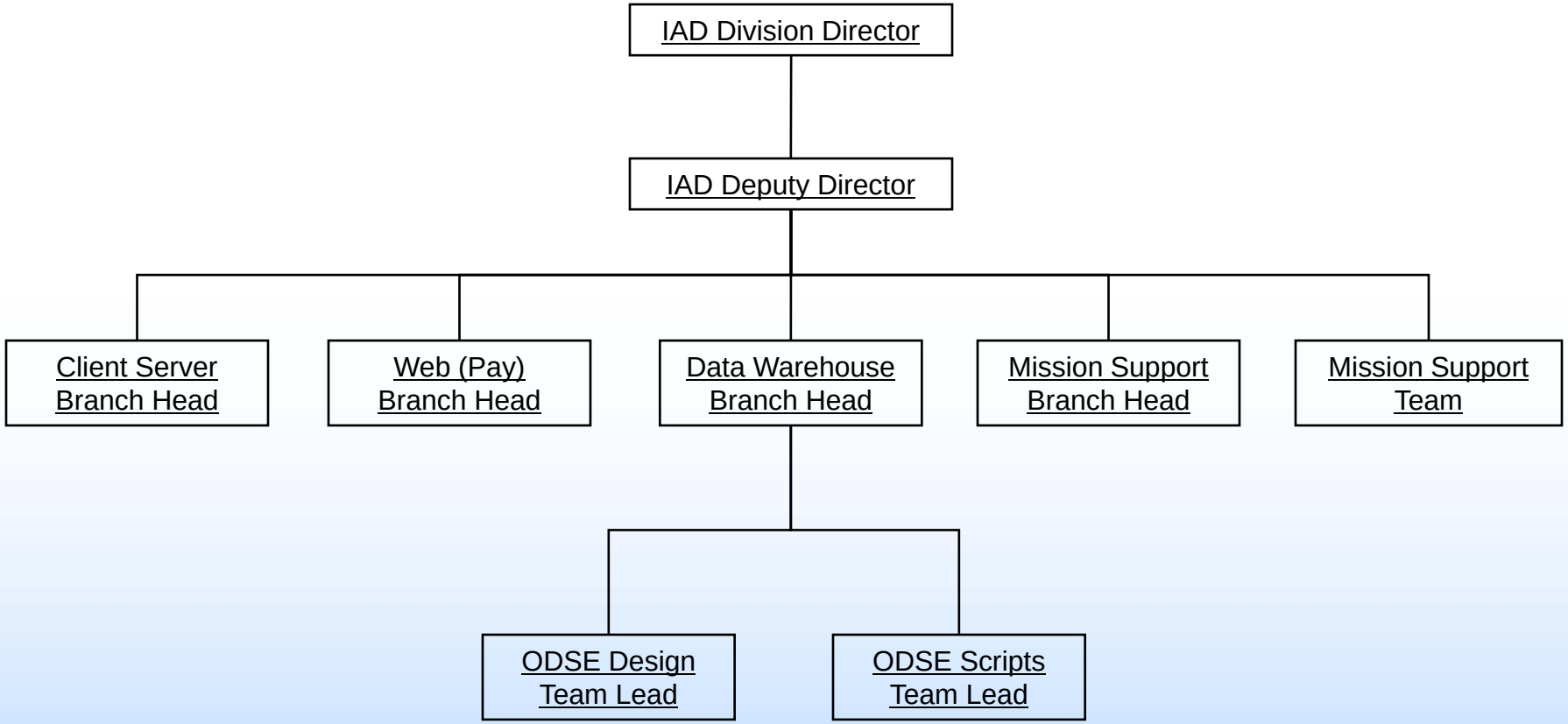
The Tar Pit
Woes of the Craft

- Why should we quit?
  - Must perform perfectly
  - Loss of control
  - Tedious extermination process
  - Monotonous testing process
  - Fruits of your labor may become obsolete *during* process

Technology Services Organization-Kansas City (TSO-KC)

- Mission
  - To develop, deliver, and maintain information systems and information technology solutions that satisfy the requirements of our customers
- Vision
  - To become the premier software development organization within the DoD and government agencies by providing innovative information systems and information technology solutions for our customers

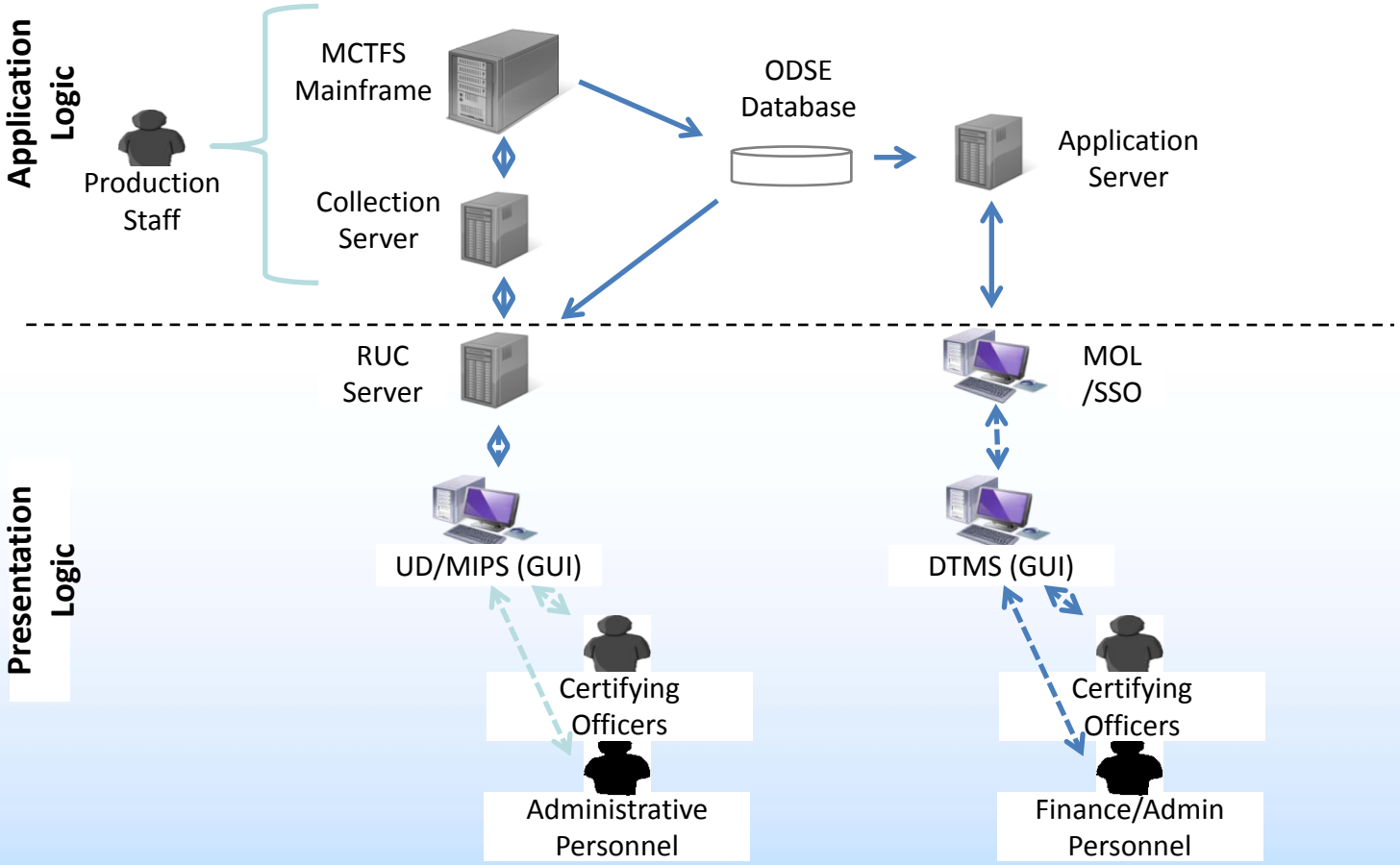
Integrated Applications Division (IAD)



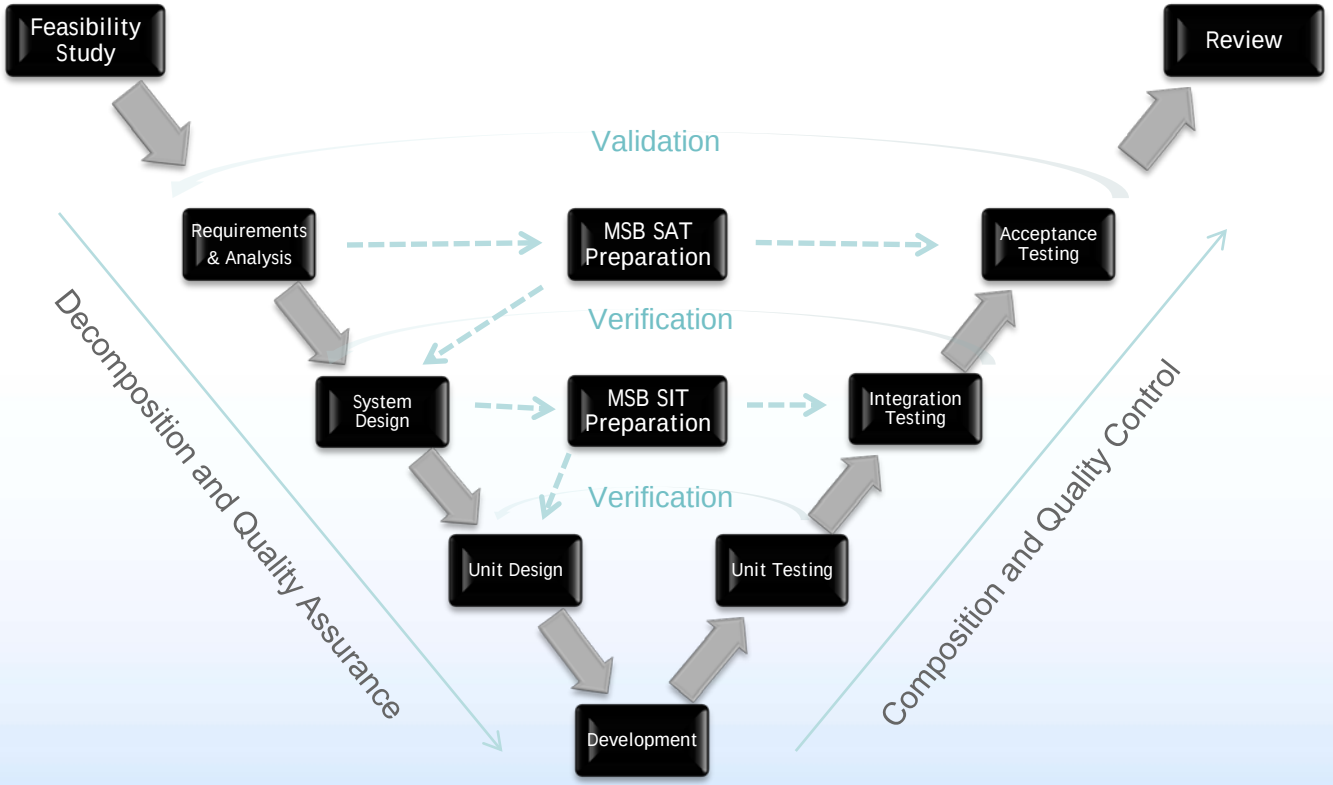
Integrated Applications Division (IAD)

- Seven distinct systems
  - Discharge Accounting Sheet (DAS)
  - Document Tracking and Management System (DTMS)
  - Operational Data Store Enterprise (ODSE)
  - Remote Access Pay Transaction and Reporting System (RAPTRS)
  - Unit Diary/Marine Integrated Personnel System (UD/MIPS)
  - W2/W2 Correction (W2/W2C)
  - Management Reports (MGNTRPTS)

Integrated Applications Division (IAD)



Integrated Applications Division (IAD)



IAD V-Model Draft

Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

The Mythical Man-Month

- Time causes more problems than all other problems combined
- Causes for disaster
  - Optimism
  - The Man-Month
  - Systems Test
  - Gutless Estimating
  - Regenerative Schedule Disaster

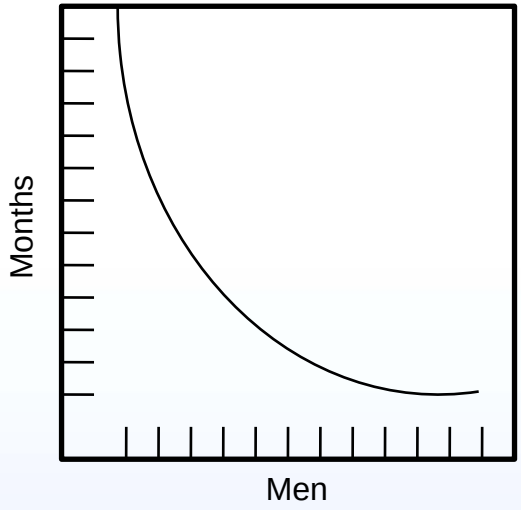
The Mythical Man-Month
Optimism

- Programmers are intrinsically optimistic
  - "This time it will surely run"
  - "I just found the last bug"
- Swamp Gas
  - Programmers work with increasingly tractable medium
  - Inclination is to expect few difficulties in implementation
  - Ideas are faulty
    - Ultimately, optimism is unjustified

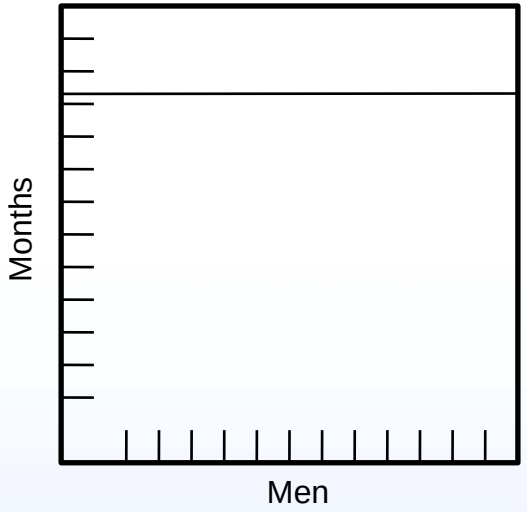
The Mythical Man-Month
The Man-Month

- Cost varies as the product of the number of men and months
- Progress does not
- Man-Month as a unit of measurement is a deceptive myth
  - Confuse effort with progress (men and months are not interchangeable)

The Mythical Man-Month
The Man-Month (continued)

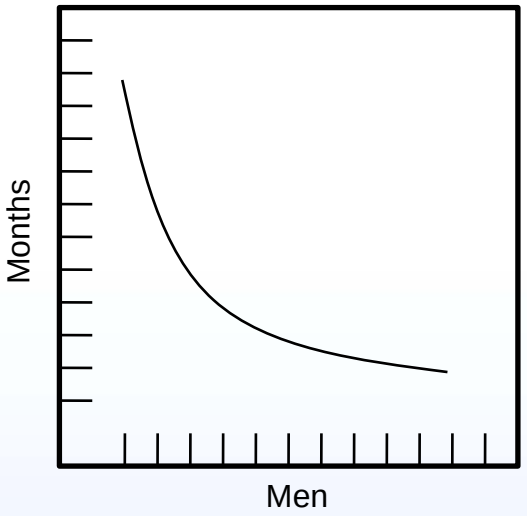


Time versus number of workers – perfectly partitionable task

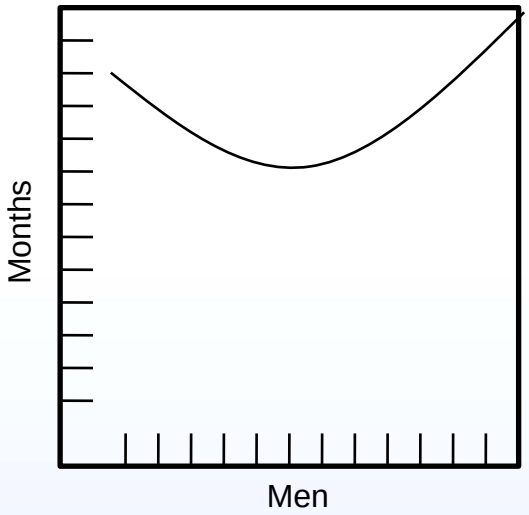


Time versus number of workers – unpartitionable task

The Mythical Man-Month
The Man-Month (continued)



Time versus number of workers – partitionable task requiring communication



Time versus number of workers – task with complex interrelationship

The Mythical Man-Month
The Man-Month (continued)

- When communication is required effort must be added to total amount of work
- Burden of communication
  - Training
    - Cannot be partitioned
    - Added effort varies linearly with number of workers
  - Intercommunication
    - n(n-1)/2
- Adding more men lengthens the schedule

The Mythical Man-Month
Systems Test

- Time required depends on number/subtlety of errors
  - Theoretically number should be zero
  - Due to inherent optimism expect less bugs
- For this reason, testing is usually the most mis-scheduled part of programming

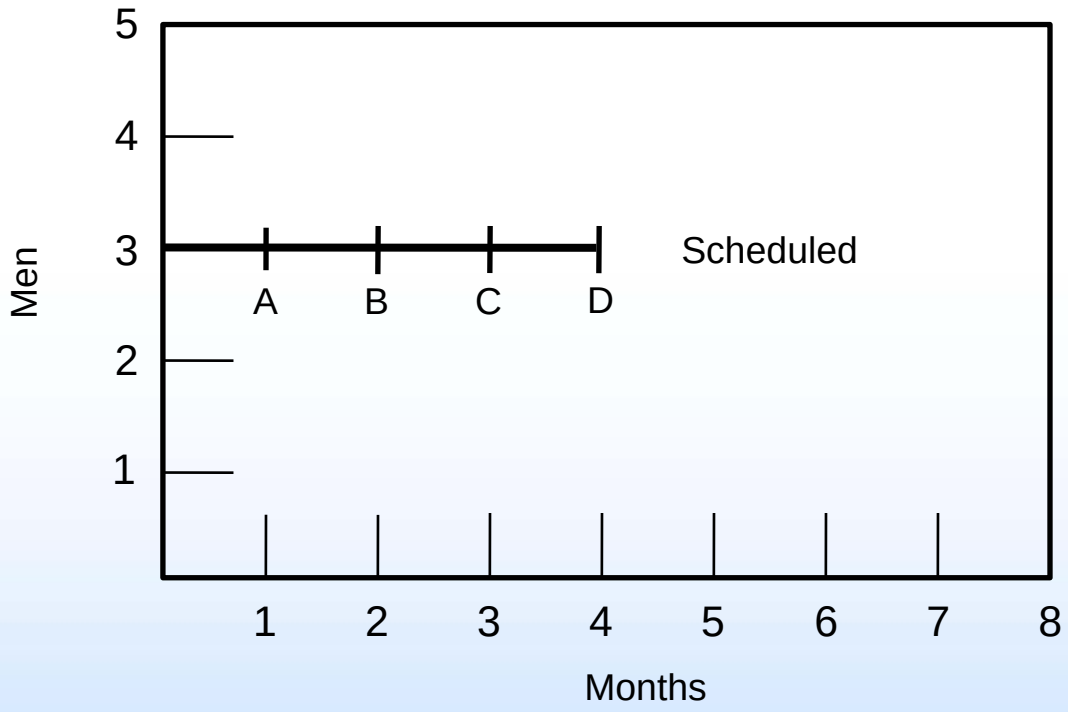
The Mythical Man-Month
Systems Test (continued)

- Brooks' rule for software task scheduling
  - 1/3 planning
  - 1/6 coding
  - 1/4 component test and early system test
  - 1/4 system test, all components in hand
- Differs from conventional scheduling
  - Larger than normal planning number
  - Half of schedule is debugging
  - Easy part to estimate (coding) only gets 1/6

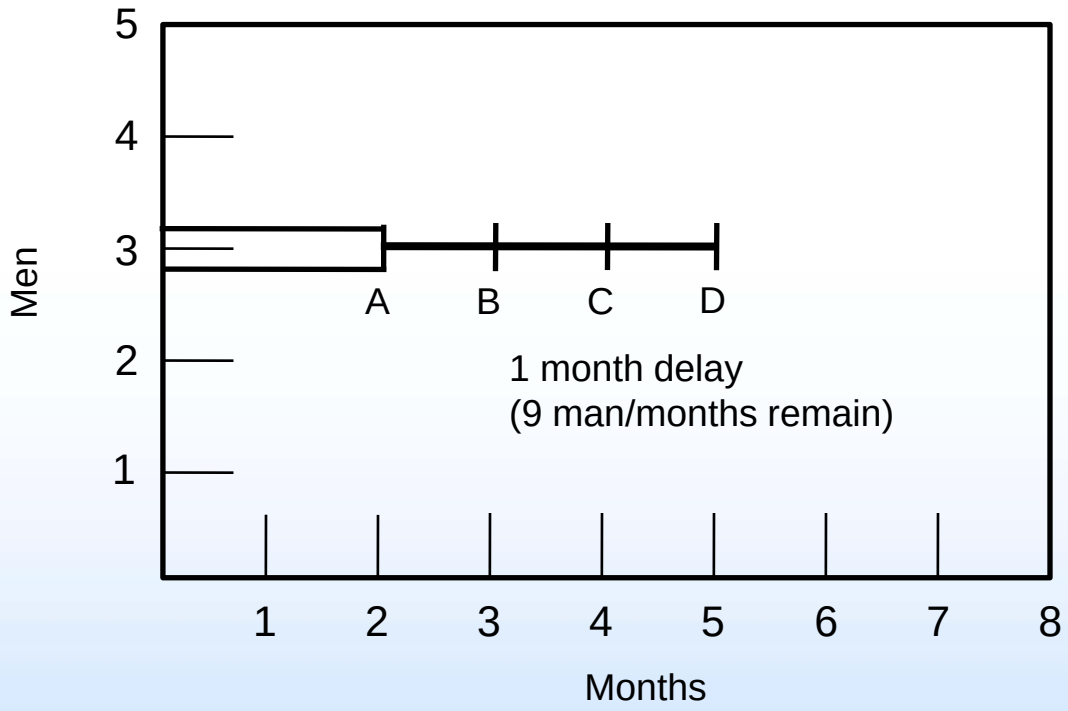
The Mythical Man-Month
Gutless Estimating

- False scheduling
  - Estimating to meet the customer's desires
  - Hard to defend estimate based on:
    - No quantitative method for deriving estimate
    - Little data
    - Manager's hunches
- Solution
  - Develop & Publicize data
    - Productivity, bug incidence and source, estimating rules, etc.

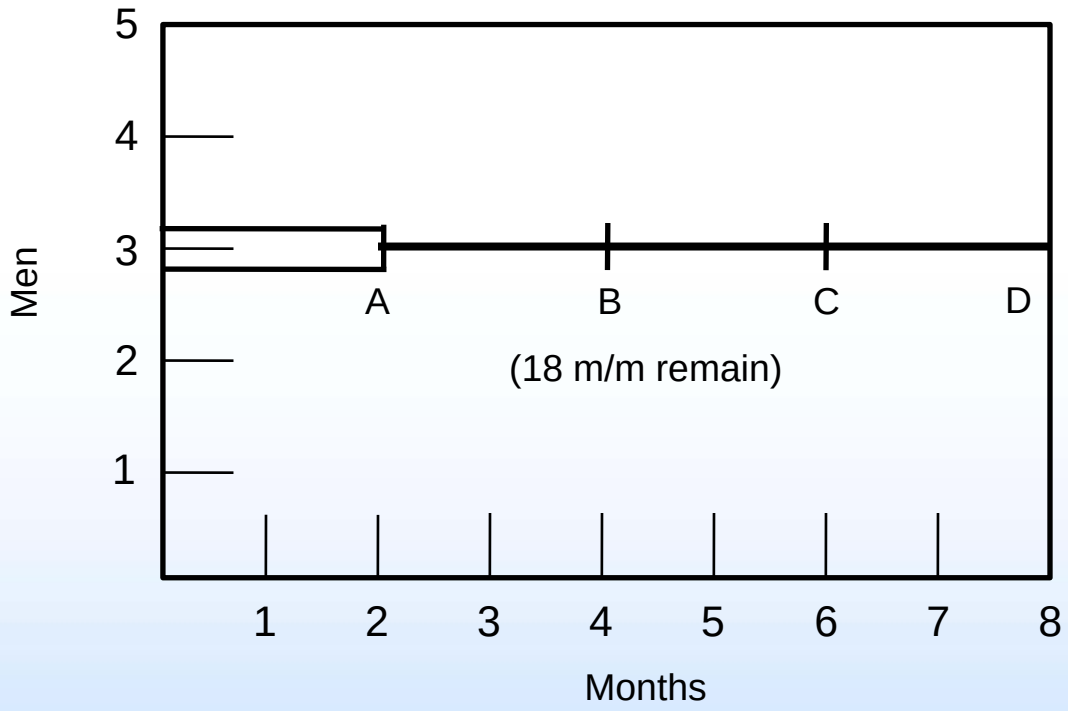
The Mythical Man-Month
Regenerative Schedule Disaster



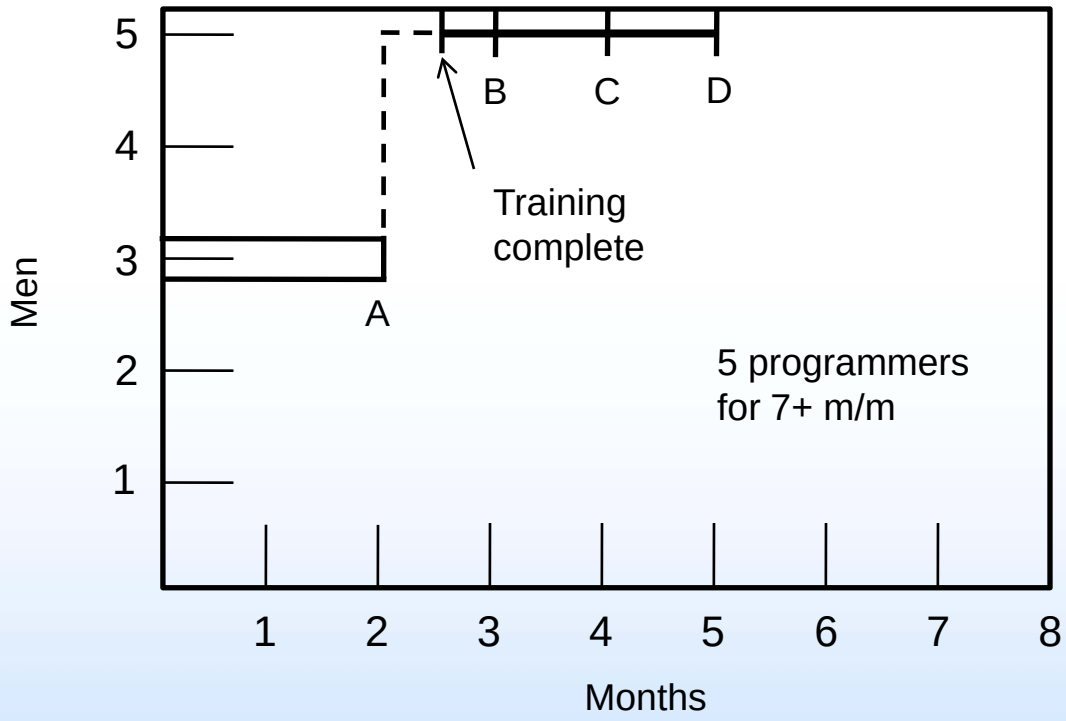
The Mythical Man-Month
Regenerative Schedule Disaster (contd)



The Mythical Man-Month
Regenerative Schedule Disaster (contd)



The Mythical Man-Month
Regenerative Schedule Disaster (contd)



The Mythical Man-Month
Regenerative Schedule Disaster

- Brooks' Law
  - "Adding manpower to a late software project makes it later"
- Adding people to a software project:
  - Increases communication requirement
  - Requires education
  - Requires repartitioning of work

Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

The Surgical Team

- How should the development team be arranged?
- The problem
  - Good programmers are much better than poor
- Programmers
  - Typically 10 times better in productivity
  - Typically 5 times better in terms of program elegance

The Surgical Team
Example

- Consider the following example
  - 200-person project with 25 experienced managers
  - Previous slide argues for firing the 175 workers and use the 25 managers as the team
    - However, this is still bigger than "the ideal" small team size of 10 people (general consensus)
  - However, the original team was too small to tackle large systems
    - OS/360 had over 1000 people working on it; consumed 5000 man-years of design, construction, and documentation

The Surgical Team
Solution

- For efficiency and conceptual integrity
  - A small team is preferred
- To tackle large systems
  - Considerable resources are needed
- One solution
  - Harlan Mill's Surgical Team approach
    - One person performs the work
    - All others perform support tasks

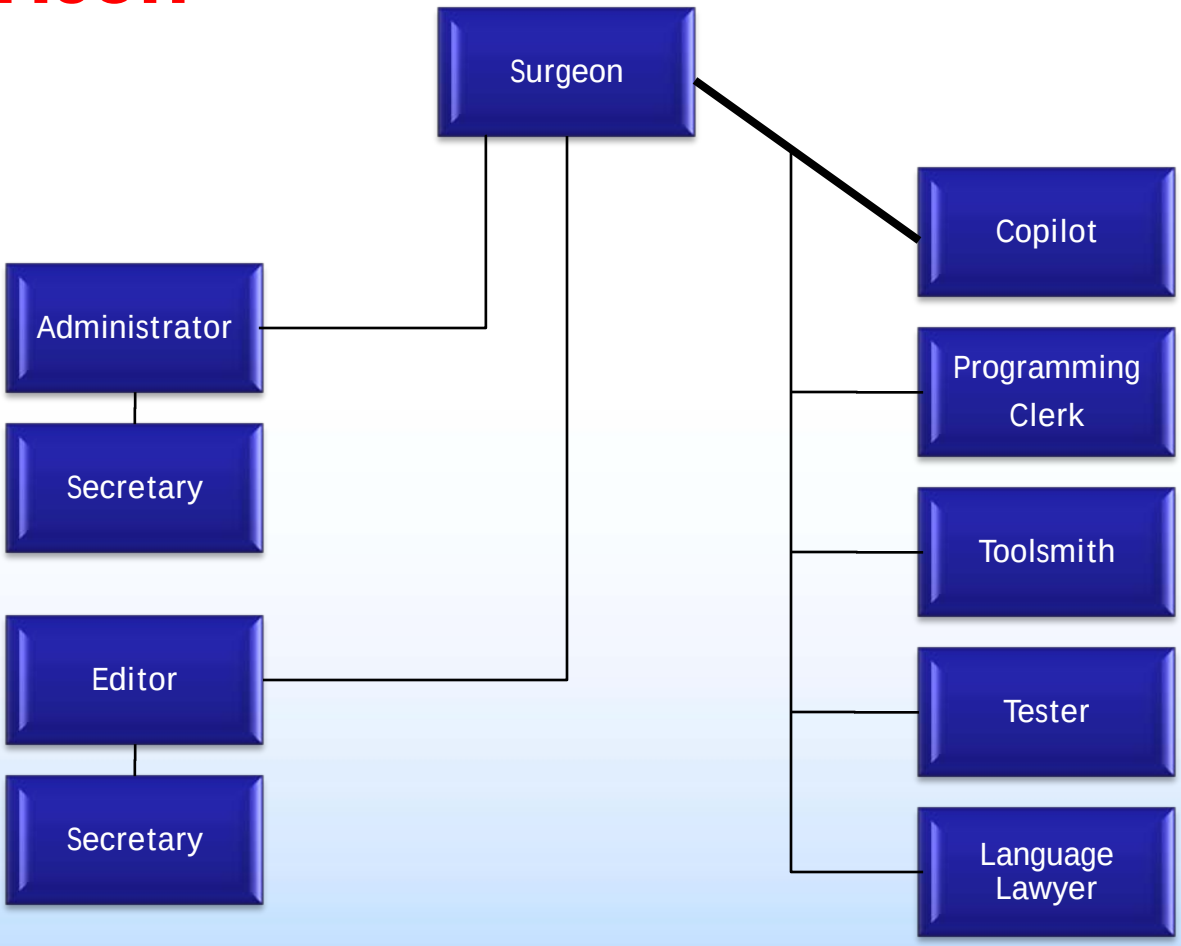
The Surgical Team
Composition

- The surgeon
  - The chief programmer
- The co-pilot
  - Like the surgeon but less experienced
- The administrator
  - Relieves the surgeon of administrative tasks
- The editor
  - Proof-edits documentation
- Two secretaries
  - Support admin and editor
- The program clerk
  - Probably obsolete today
- The toolsmith
  - Supports the work of the surgeon
- The tester
- The language lawyer
  - Master of the programming language

The Surgical Team
Comparison

- Conventionally, work is divided equally
  - Now only surgeon and co-pilot divide the work
- Conventionally, each person has equal say
  - The surgeon is the absolute authority
- Note communication paths are reduced
  - Normally 10 people → 45 paths
  - Surgical Team → at most 21 paths

The Surgical Team
Comparison



The Surgical Team
Composition (continued)

- Based on 10:1 ratio
- Scales up through hierarchical division of problems
- Single surgeon on problem (sub-problem) maintains conceptual integrity of design
- Requires good communication among surgeons

The Surgical Team
Composition (continued)

- Reconsider the 200 person team
  - Communication paths → 19,900
- Create 20, ten-person surgical teams
- Now, only 20 surgeons must work together
  - 20 people → 190 paths
- Two orders of magnitude less
- Key problem is ensuring conceptual integrity of the design

The Surgical Team
Measure of Success

- The key test to a system's design is the ratio of functionality to conceptual complexity
  - Ease-of-use is enhanced only if the functionality provides more power than it takes to learn (and remember) how to use it in the first place!
  - Neither function or simplicity alone is good enough

Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

Aristocracy, Democracy and System Design

- Conceptual integrity
  - System has same look and feel throughout
  - How do we keep things consistent same look and feel when different teams work on different parts?
- Solution
  - Hire an architect
  - Record the vision
  - Enforce constraints

Aristocracy, Democracy and System Design

- Brooks example → Cathedrals
  - Many cathedrals consist of contrasting design ideas
  - Reims Cathedral
- With respect to software
  - Design by too many people results in conceptual disunity
  - Makes the program hard to understand and use

Aristocracy, Democracy and System Design

- Brooks considers integrity the most important consideration in system design
  - Better to leave functionality out of a system if it causes the conceptual integrity of the design to break
- Conceptual Integrity
  - Consistency and accuracy of model
  - Implies ease of use

Aristocracy, Democracy and System Design

Achieving Conceptual Integrity

- Achieved more easily
  - With fewer designers/functions
  - System reflects single philosophy
- Improvement of quality and schedule
- System will better reflect the needs of the user
- Ratio of productivity gain to cost [of system and training] usually decreases with increased functionality

Aristocracy, Democracy and System Design

Aristocracy and Democracy

- Architect must decide direction based on user's needs
- System development must be aristocratic
  - Not democratic
  - Architect has final word
- Architect absorbs the team's ideas while ensuring conceptual integrity

Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

The Second-System Effect

- The first system is minimal and solid great ideas are saved for the second one
- All the great ideas are stuffed into the second one
- Result:
  - Small percentage increase in functionality and quality
  - Large percentage increase in size and complexity

The Second-System Effect Avoidance

- Self-discipline
- Maintain conceptual integrity
- Avoid functional ornamentation
- Determine trade-off between functionality, time, and money
- Adhere to principles of project management
- Experienced architects

Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

Passing the Word

- Communication is the root of most problems
- How do we communicate?
  - Written specifications
  - Formal definitions
  - Direct incorporation
  - Conferences and courts
  - Multiple implementations
  - Telephone log
  - Product tests

Passing the Word
The Manual

- An external specification of the product
- Should be precise, full, and accurately detailed
- Describes only what the user sees
- Includes descriptions of user interfaces

Passing the Word
Formal Definitions

- Use standards to develop formal definitions
- Keep style and prose consistent
- May use a limited number of writers to achieve consistency
- Definition writing should reflect audience
  - Requirements definition vs. requirements specification

Passing the Word
Conferences and Courts

- Meetings are a necessity
- Set a meeting schedule to reflect your project
  - Weekly, monthly, mid-project?
  - Are more frequent meetings needed?
  - When is a design review needed?
- Follow good techniques
  - Agendas
  - Roles and responsibilities
  - Communication rules
  - Documentation

Passing the Word
Telephone Logs

- New technology may surpass the need for telephone logs, but concept still applies
- Document communication
  - What was done?
  - Why it was done?
  - Ensure team has access to communication
  - Avoids guessing and misinterpretation

Passing the Word
Product Test

- Independent testers are surrogate customers
- When communication fails, testers identify the breakdown

Agenda

- Introduction
- Overview
- The Tar Pit
- The Mythical Man-Month
- The Surgical Team
- Aristocracy, Democracy and System Design
- The Second-System Effect
- Passing the Word
- Why Did the Tower of Babel Fail?

The Tower of Babel
Why Did Project Fail?

- Lack of communication and organization
- Can't communicate without coordination
- How do teams communicate?
  - Informally
  - Meetings
  - Workbook

The Tower of Babel
Informal

- Communication
  - Phone
  - Email
  - Non-meeting communication
- Meetings
  - Follows up informal communication
  - Starts to close the communication gap

The Tower of Babel
Workbook

- Essential for maintaining communication and organization
- Needs to be written and available to all members of the team
- Includes all documentation:
  - Objectives
  - Specifications
  - Standards
  - Etc.

The Tower of Babel
Workbook

- Should always be up to date
- Logically organized to reduce lines of communication
  - Division of labor
  - Function
- Use current technology to manage
- Searchable
- Should all information be available to all parties?

Agenda

- Calling the Shot
- Ten Pounds in a Five-Pound Sack
- Optimization
- The Documentary Hypothesis
- Plan to Throw One Away
- Sharp Tools
- Conclusion
- Questions

Calling the Shot

- Estimating
  - System development estimates include:
    - Planning time
    - Coding
    - Testing
    - System integration
    - Training times
    - Documentation
    - Etc.

Calling the Shot

- How do we estimate all of these activities?
- Direct extrapolation is not always reliable and can lead to gross inaccuracies
- Essay discusses several insights
  - Portman's Data
  - Aron's Data

Calling the Shot
Portman's Data

- Even with careful estimation using expert data, estimates still about one-half actual time
- Analyzing actual time uncovered non-estimated time for other tasks
  - Machine downtime
  - Meetings
  - Paperwork
  - Higher priority unrelated tasks
  - Sickness
  - Etc.

Calling the Shot
Aron's Data

- Studied large systems for design and programming tasks
- Categorized systems based on very few, some, and many interactions between programmers
- Interactions appear to play key productivity
  - Very few: 10,000 instructions per man-year
  - Some: 5,000 instructions per man-year
  - Many: 1,000 instructions per man-year

Calling the Shot
Additional Data

- Program size and complexity increase the effort required
- One study shows effort increases exponentially by a power of 1.5

Calling the Shot

- Keep the following in mind
  - Use expert estimation and experience
  - Complex and large projects appear to increase actual effort
  - Keep lines of communication in mind while estimating
  - Consider overhead

Agenda

- Calling the Shot
- Ten Pounds in a Five-Pound Sack
- Optimization
- The Documentary Hypothesis
- Plan to Throw One Away
- Sharp Tools
- Conclusion
- Questions

Ten Pounds in a Five-Pound Sack

- Size Constraints
  - Much of the essay is devoted to space cost, controlling size, etc.
  - Essence of programming discusses good programming guidelines
    - Algorithms
    - Control structures
    - Data structures

Agenda

- Calling the Shot
- Ten Pounds in a Five-Pound Sack
- The Documentary Hypothesis
- Plan to Throw One Away
- Sharp Tools
- Conclusion
- Questions

The Documentary Hypothesis

- Seems to be core set of documents required of any project type
  - Objectives
  - Specifications
  - Schedule
  - Budget
  - Space allocation
  - Organization chart
  - Estimate, forecast, prices

Agenda

- Calling the Shot
- Ten Pounds in a Five-Pound Sack
- The Documentary Hypothesis
- Plan to Throw One Away
- Sharp Tools
- Conclusion
- Questions

Plan to Throw One Away

- Plan to throw one away; you will, anyhow
  - Save your customers frustration
  - It will save you embarrassment
- Changes in objectives are inevitable
  - Changes in development strategy and technique are also inevitable
- Use of configuration/change management is essential
- Reducing redundancy and automated support for documentation helps

Plan to Throw One Away (continued)

- Too much acceptance of change is poor management
- Threatening organizational structure encourages lack of documentation
- Flexibility in assignments is a must
  - Have the best person on the job at all times
  - Keep management and technical people changeable "Member of the Technical Staff"
  - Suggest overcompensating for move from managerial to technical position to overcome perceived hierarchy

Plan to Throw One Away
Calculate Change

- Uncalculated change
  - As opposed to hardware, software maintenance is usually unplanned change
    - No cleaning, no lubrication
    - Usually means fixing mistakes and adding functionality
  - Change in architecture means change visible to user

Plan to Throw One Away
Survival of the Fittest

- What Brooks is referring to is what is now called *Software Evolution*
  - It is inevitable that people desires grow with their knowledge of what is possible
  - Their ability to find bugs grows with their willingness to experiment with features
  - System designers constantly work to keep the users happy
  - Systems evolve as they are used

Plan to Throw One Away
Infestation

- Fixing a bug usually breeds more bugs
  - Most changes (bug fixes) have far-reaching effects
    - Lack of documentation
    - Lack of modularity
    - Lack of cohesion on programming team
    - Chief programmer got hit by a bus
- Call for regression testing techniques
- Programs become fragile with time
  - Design gets lost
  - Modularity decreases

Agenda

- Calling the Shot
- Ten Pounds in a Five-Pound Sack
- Optimization
- The Documentary Hypothesis
- Plan to Throw One Away
- Sharp Tools
- Conclusion
- Questions

Sharp Tools

- Hording of tools is natural but foolish
  - Essential problem is communication
  - Tool lifetime is short
- Target and vehicle machines
  - Not having target facility leads to unexpected bugs
    - Memory size
    - CPU speed
    - Bus speed
    - Etc.

Conclusion

- Don’t fight it
- Time causes more problems than all other problems combined
- Men and months are not interchangeable
- Ensuring conceptual integrity of the design shapes team composition
- Communication is the root of most problems
- Remain flexible

Questions