The Mythical Man-Month
Frederick P. Brooks, Jr.

Published 1975, Republished 1995
Experience managing the development of OS/360 in 1964-65

Central Argument
- Large programming projects suffer management problems different in kind than small ones, due to division of labor.
- Critical need is the preservation of the conceptual integrity of the product itself.

Central Conclusion
- Integrity achieved through exceptional designer.
- Implementation achieved through well-managed effort.

The Tar Pit

Program --> Program Product
- Tested (esp. boundary values)

3 X Cost of Simple Program

Programming System --> Programming Systems Product
- Precisely Defined (Module) Interfaces
- Follows Prescribed Budget (system and organizational)
- Tested (esp. its integration with other subsystems)

9 X Cost of Simple Program

Myths

- Poor Estimation
- Man-Month Myth
- Not Planning for Testing
- Gutless Estimating
- Regenerative Schedule Disaster

Myths—Poor Estimation

- Based on Assumption that Nothing Goes Wrong
- Large Project Consists of Many Smaller Tasks
- Probability of No Failures Diminishes

Myths—Man-Month Myth

- True: Project Cost is Proportional to Number of Personnel
- False: Progress is Proportional to Number of Personnel
- Fallacy is in an Assumption of Subtasks Requiring No Communication

Myths—Not Planning for Testing

- Many Projects on Schedule until Testing
- Bias toward No Failure
- Suggested Schedule
  - 33% for Planning
  - 17% for Implementation
  - 50% of time to Testing (half for component and half for integration)
Myths—Gutless Estimating

- Urgency of Client causes Optimistic Estimates
- Irregardless of Urgency, Tasks require same amount of time

Myths—Regenerative Schedule Disaster

- Adding Personnel Requires Retraining
- Retraining is not in the Planned Schedule
- Project Falls Further Behind
- Cycle Regenerates Itself

Adding manpower to a late software project makes it later.

The Surgical Team (1)

- Surgeon
  - Expert Performs Design
- Copilot
  - Follows Design
  - Knows Alternatives

The Surgical Team (2)

- Administrator
  - Manages Money, People, Space, & Machines
  - Legal and Contractual Arrangements
  - Liaison between Surgeon and Client
- Editor
  - Reworks Surgeon’s Documentation for General Consumption

The Surgical Team (3)

- Two Secretaries
  - for Administrator and Editor
- Program Clerk
  - Maintains Evolving Artifacts (versions etc.)
- Toolsmith
  - Expert on Supporting Software Tools
- Tester
  - Adversarial Role: Test Cases for Functional Tests
  - Assistant Role: Test Cases for Debugging
- Language Lawyer
  - Special Specification and Programming Language Features

The Surgical Team — Notes

- Based on 10:1 Ratio
- Scales up through Hierarchical Division of Problems
- Single Surgeon on Problem (Subproblem) maintains Conceptual Integrity of Design
- Requires Good Communication among Surgeons

Topic 11
Mythical Man-Month
Conceptual Integrity

- Conceptual Integrity = consistency and accuracy of model
- Conceptual Integrity implies Ease of Use
- Achieved More Easily with Fewer Designers (a surgeon/architect-based approach)
- Achieved More Easily with Fewer Functions
- Ratio of Productivity Gain to Cost [of System and Training] usually Decreases with Increased Functionality

The Second System Effect

- Systems Evolve to Include Esoteric Features
- In so doing, they fail to anticipate paradigm shifts.
  - e.g., overlay systems dying before virtual memory systems
- Solution: Experienced System Architect Sensitive to the Second System Effect

Achieving Effective Communication

- Direct
  - Through Informal Mechanisms (e.g., telephone)
- Meetings
  - Project Workbook
    - All Documents and Artifacts from Design through Implementation and Testing

Developer Productivity

- Not All Working Hours are Devoted to a Project
  - Interruptions include meetings, high-frequency, unrelated tasks.
- Productivity is constant in the units.
- Higher Level tools Imply Higher productivity

Pilot Systems

- Plan to throw one away; you will, anyhow.
- Plan for Change — Mindset
- Plan for Change Organizationally
  - Training and Promotion Motivation

Pilot Systems Reconsidered

- Prototypes considered harmful!
- Plan to throw one away?
- Or ...
  - Follow an incremental build strategy.
  - E.g., Microsoft Daily Build
  - E.g., Mockups and Scenarios
  - Not Prototypes

Topic 11
Mythical Man-Month
Designing the Bugs Out (1)

- Bug-proofing the Definition a.k.a. Conceptual Integrity
- Testing the Specification
- Top-down Design
  - Allows Design by Single or Small Number of Architects
- Structured Programming
- Component Debugging and Reuse
- Interactive Debugging
- Use Debugging Scaffolding
  - as much as 50%

Designing the Bugs Out (2)

- Control Changes
  - Versioning
- Add One Component at a Time
- Quantize Updates
  - Large and Infrequent
  - or Small and Frequent

Documentation

- End Users
- Acceptance Cases
- Modification [Flowchart]
- Self-Documenting Programs

No Silver Bullet (NSB)

- Essence—the difficulties inherent in the nature of the software [complexity]
- Accidents—those difficulties that today attend its production but that are not inherent [production]
- Solution: Grow Great Designers

p. vii, essence and accidents comes from commentary on Suzuki violin pedagogy