

ICS 121 Topic 14:
Software Engineering Myths

Myths and Suggestions
 No Silver Bullet

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.

Topic 14 Software Engineering Myths 2

The Tar Pit

- Program --> Program Product
 - Tested (especially boundary values)
 - Documented (usage and maintenance)
- 3 X Cost of Simple Program
- Programming System --> Programming Systems Product
 - Precisely Defined [Module] Interfaces
 - Follows Prescribed Budget (system and organizational)
 - Tested (especially its integration with other subsystems)
- 9 X Cost of Simple Program

Topic 14 Software Engineering Myths 3

**MMM Propositions:
 Myths and Suggestions**

• The Tar Pit	• Critical Documentation
• Mythical Man-Month	• Plan to Throw One Away
• The Surgical Team	• Plan for Change and Maintenance
• Aristocracy, Democracy, and System Design	• Sharp Tools
• The Second-System Effect	• The Whole and its Parts
• Passing the Word	• Documentation: the other face
• Communication, Documentation, Organization	• No Silver Bullet
• Calling the Shot	
• Ten Pounds in a Five-Pound Sack	

Topic 14 Software Engineering Myths 4

Myths to Consider

- Poor Estimation
- Conceptual Integrity
- Effective Communication
- Developer Productivity
- Pilot Systems
- Designing the bugs out

Topic 14 Software Engineering Myths 5

Myths—Poor Estimation

- Based on assumption that nothing goes wrong
- Large project consists of many smaller tasks
- Probability of no failures diminishes

Topic 14 Software Engineering Myths 6

Myths—Man-Month

- 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

Topic 14 Software Engineering Myths 7

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)

Topic 14 Software Engineering Myths 8

Myths—Gutless Estimating

- Urgency of client causes optimistic estimates
- Regardless of urgency, tasks require same amount of time

Topic 14 Software Engineering Myths 9

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.

Topic 14 Software Engineering Myths 10

The Surgical Team

<ul style="list-style-type: none"> • Surgeon <ul style="list-style-type: none"> – Expert performs design • Copilot <ul style="list-style-type: none"> – Follows design – Knows alternatives • Administrator <ul style="list-style-type: none"> – Manages money, people, space, & machines – Legal and contractual arrangements – Liaison between surgeon and client • Editor <ul style="list-style-type: none"> – Reworks surgeon's documentation for general consumption 	<ul style="list-style-type: none"> • Two Secretaries <ul style="list-style-type: none"> – for administrator and editor • Program Clerk <ul style="list-style-type: none"> – Maintains evolving artifacts (versions etc.) • Toolsmith <ul style="list-style-type: none"> – Expert on supporting software tools • Tester <ul style="list-style-type: none"> – Adversarial role: test cases for functional tests – Assistant role: test cases for debugging • Language Lawyer <ul style="list-style-type: none"> – Special specification and programming language features
--	---

Topic 14 Software Engineering Myths 11

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 14 Software Engineering Myths 12

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

Topic 14

Software Engineering Myths

13

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

Topic 14

Software Engineering Myths

14

Achieving Effective Communication

- Direct
 - Through informal mechanisms (e.G., Telephone)
- Meetings
- Project Workbook
 - All documents and artifacts from design through implementation and testing

Topic 14

Software Engineering Myths

15

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

Topic 14

Software Engineering Myths

16

Pilot Systems

- Plan to throw one away; you will, anyhow.
- Plan for change — mindset
- Plan for change organizationally
 - Training and promotion motivation

Topic 14

Software Engineering Myths

17

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 14

Software Engineering Myths

18

Designing the Bugs Out

- Bug-proofing the definition *aka* 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%
- Control changes via versioning
- Add one component at a time
- Quantize updates
 - large and infrequent or small and frequent

Topic 14

Software Engineering Myths

19

Documentation

- End users
- Acceptance cases
- Modification [flowchart]
- Self-documenting programs

Topic 14

Software Engineering Myths

20

No Silver Bullet

- 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]
- *essence and accidents comes from commentary on Suzuki violin pedagogy*
- Solution: Grow Great Designers

Topic 14

Software Engineering Myths

21