The Mythical Man-Month
Presented by Annette Tetmeyer
The Tar Pit

The Programming Systems Product

- Programming Product
  - Needs to be run, tested, maintained, and extended by anyone
  - Requires documentation
  - Brooks notes that this costs 3X as much as the original program
  - Now evolve into the *programming system*
The Tar Pit

The Programming Systems Product

- Programming System
  - Add interfaces and system integration
  - Requires more coordination and discipline
  - Additional programming systems components may be needed
  - Again, costs rise an estimated 3X the original programming cost
  - Finally, evolve into the programming systems product
The Programming Systems Product

- Eventually, we end up with the programming systems product
- Costs are estimated to be nine times as much as the original program
- However, now we have a useful product
The Tar Pit

Evolution of the Programming Systems Product

Program

Programming System
(Interfaces, System Integration)
3X cost

Programming Product
(Generalization, Testing, Documentation, Maintenance)
3X cost

Programming Systems Product
9X cost
The Tar Pit

The Joys of the Craft

- Why is programming fun?
  - Sheer joy of making things
  - Pleasure of making things that are useful to other people
  - Fascination with complex problem solving
  - Joy of continuous learning
  - Delight of working in a tractable medium
The Tar Pit

The Woes of the Craft

- Programming is not always joy and fun...
  - We must perform perfectly
  - May not have complete control over objectives, budget, schedule, etc.
  - Most of us do not enjoy debugging
  - Testing is even less fun and takes time
  - Programs may be obsolete when finished...
The Tar Pit

Programming can be a tar pit, but also a very rewarding activity.

We wouldn’t be here if the joys didn’t outweigh the woes.
Additional Topics Discussed

- Effort Estimation, Schedules
- Teams
- Conceptual Integrity
- Redesign and Featuritis
- Communicating Design Decisions
- Communication and Organization
- Estimating
- Size Constraints
Good cooking takes time. If you are made to wait, it is to serve you better, and to please you.
The Mythical Man-Month

Effort Estimation, Schedules

“More software projects have gone awry for lack of calendar time than for all other causes combined”

“Why is this cause of disaster so common?”
Frederick P. Brooks, Jr.
- Joined IBM in 1956
- Worked on the System/360 family computers, OS/360 operating system and Stretch architecture
- Founded the Department of Computer Science at the University of North Carolina at Chapel Hill in 1965
- Continues to research today in areas such as virtual worlds (virtual reality)
The Mythical Man-Month

Points to consider regarding estimating

- Estimating techniques are poorly developed
- Confuse effort with progress (men and months are not interchangeable)
- Uncertainty in estimation does not allow us to stand our ground
- Schedule progress is poorly monitored
- If schedules slip, the natural response is to add manpower
The Mythical Man-Month

Optimism - Estimating techniques are poorly developed

- Natural tendency to assume that “all will go well”
- Don’t adequately prepare for bugs, error checking, etc.
- Programming is a creative process and inherently difficult to estimate
- Small tasks are chained together, reducing the probability that there will not be delays
The Mythical Man-Month

The Man-Month - Confuse effort with progress (men and months are not interchangeable)

- Due to increased lines of communication, there is not a direct correlation between men and months
The Mythical Man-Month

Gutless Estimating - Uncertainty in estimation does not allow us to stand on our ground

- Even if a request is urgent, do not cave in to reducing our estimates
- Should gather and develop data to provide a sound basis for making estimates in the first place
The Mythical Man-Month

Regenerative Schedule Disaster - If schedules slip, the natural response is to add manpower

Follow Brook’s Law: Adding manpower to a late software project makes it later.

May have to consider rescheduling rather than adding fuel to the fire.
The Surgical Team

Teams

- How do we create teams to work on systems development projects?
- If a system is very large, what are the implications?
The Surgical Team

- We all wish to staff projects with the ideal team:
  - Small in size
  - Sharp, intelligent
  - Excellent communication skills
- How is this possible for very large system development?
- Balance the need for efficiency and conceptual integrity with the ability to produce a viable system before it is obsolete.
The Surgical Team

- The surgical team metaphor is applied to the systems development team
- Each segment of a large project should have a surgical team
- Balances support and effective communication to achieve better productivity
- Some of the analogies may be slightly dated
- Proposed by Harlan Mills
The Surgical Team

Surgical team functions with:

- Clearly defined responsibilities and decision making roles
- Skill sets to support responsibilities and roles
- Clear channels (lines) of communication
The Mythical Man-Month was written in 1975

Series of essays on mistakes made and lessons learned from the experiences at IBM

Chapters open with a metaphor related to the topic
The Surgical Team

Team Players
- Chief Programmer (surgeon)
- Copilot
- Administrator
- Editor (edits documentation)
- Secretaries
- Program clerk
- Toolsmith (provides basic tools for the team)
- Tester
- Language lawyer (expert in a particular area)
The Surgical Team

- Using this basic team structure, scale up the number of teams depending on the project size
- Each team is responsible for a piece of the program, thereby maintaining conceptual integrity
- Chief programmers on each team communicate with each other (reducing the total lines of communication)
This great church is an incomparable work of art. There is neither aridity nor confusion in the tenets it sets forth. . . . It is the zenith of a style, the work of artists who had understood and assimilated all their predecessors’ successes, in complete possession of the techniques of their times, but using them without indiscreet display nor gratuitous feats of skill.

It was Jean d’Orbais who undoubtedly conceived the general plan of the building, a plan which was respected, at least in its essential elements, by his successors. This is one of the reasons for the extreme coherence and unity of the edifice.

REIMS CATHEDRAL GUIDEBOOK
Aristocracy, Democracy, and System Design

"Conceptual integrity is the most important consideration in system design."

Prefer a system with fewer features that sticks with a set design concept than a system with many good features that are uncoordinated and independent.
Aristocracy, Democracy, and System Design

Achieving Conceptual Integrity
- The system should reflect a single philosophy
- Quality and schedule should be improved
- Overall, the system will better reflect the user’s needs
Aristocracy, Democracy, and System Design

Aristocracy and Democracy

- Architect must decide what the system will do based on the user’s needs
- Others may have great ideas, but the architect has to ensure conceptual integrity
- System development must be an aristocracy (architect has the final word), not a democracy
What Does the Implementer Do While Waiting?

- Brooks recounts the multi-million dollar mistake
- Choice was between allowing 150 implementers to sit idle while the architect did the job
The Second-System Effect

Redesign and “Featuritis”

- Tendency of the first system is to be clean and spare
- Architect may think of new features that should be added
- Second system tends to show less restraint and may suffer from “featuritis”
- Future systems reign in the tendency of over-design
The Second-System Effect

How do we avoid the Second-System Effect?

- Self-discipline
- Continue to maintain conceptual integrity
- Avoid functional ornamentation
- Determine the trade-off between time, money and functionality
- Good project management
- Using experienced architects
Silver Anniversary Edition

- Released in 1995
- Main essays are unchanged
- Additional comments have been added in later chapters
- While some of the references to technology are dated, the topics addressed are current
- Considered to be a classic on software project management
Passing the Word

Communicating Design Decisions

- How do we communicate?
  - Written specifications – the manual
  - Formal definitions
  - Direct incorporation
  - Conferences and courts
  - Multiple implementations
  - Telephone log
  - Product tests

*some of these concepts may be dated or surpassed by current technology*
Passing the Word

Written specifications – the manual
- An external specification of the product
- Should be precise, full and accurately detailed
- Describes what the user sees and refrains from describing what the user does not see
- 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 this 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 for ongoing project decisions and coordination?
  - When is a design review needed?
Passing the Word

Conferences and Courts
- Follow good meeting techniques
  - Agendas
  - Roles and Responsibilities
  - Communication Rules
  - Documentation
Passing the Word

Telephone Logs
- New technology may surpass the need for telephone logs, but the concept still applies

- Document communication
  - What was done
  - Why it was done
  - Ensure that the team can access the communication
  - Avoids guessing and misinterpretation
Passing the Word

Product Test

- Independent testers are surrogate customers
- If the word did not get passed (communication failed), testers find the breakdown in communication
Why Did the Tower of Babel Fail?

Now the entire earth was of one language and uniform words. And it came to pass when they traveled from the east, that they found a valley in the land of Shinar and settled there. And they said to one another, "Come, let us make bricks and fire them thoroughly"; so the bricks were to them for stones, and the clay was to them for mortar. And they said, "Come, let us build ourselves a city and a tower with its top in the heavens, and let us make ourselves a name, lest we be scattered upon the face of the entire earth". And the Lord descended to see the city and the tower that the sons of man had built. And The LORD said, "Lo! [they are] one people, and they all have one language, and this is what they have commenced to do. Now, will it not be withheld from them, all that they have planned to do? Come, let us descend and confuse their language, so that one will not understand the language of his companion". And the Lord scattered them from there upon the face of the entire earth, and they ceased building the city. Therefore, He named it Babel, for there the Lord confused the language of the entire earth, and from there the Lord scattered them upon the face of the entire earth.
Chapter 1-9 Synopsis

- 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?
- Calling the Shot
- Ten Pounds in a Five-Pound Sack
Why Did the Tower of Babel Fail?

Communication and Organization
- Without communication, we cannot coordinate
- How do teams communicate with each other?
  - Informally
  - Meetings
  - workbook
Why Did the Tower of Babel Fail?

- Informal communication
  - Phone
  - Email
  - Non-meeting communication
- Meetings
  - Follows up informal communication
  - Starts to close the communication gap
Why Did the Tower of Babel Fail?

Workbook

"A project workbook is not so much a separate document as it is a structure imposed on the documents that the project will be producing anyway."
Why Did the Tower of Babel Fail?

- Workbook
  - Essential for maintaining communication and therefore, organization
  - Needs to be written and available to all members of the team
  - Includes all documentation:
    - Objectives
    - Specifications
    - Standards
    - Etc.
Why Did the Tower of Babel Fail?

- Workbook
  - Should always be up to date
  - Logically organized to reduce lines of communication. Consider organizing by:
    - Division of labor
    - Specialization of function
  - Use current technology to manage
  - Searchable
Why Did the Tower of Babel Fail?

- Workbook
- Should all information be available to all parties? (Is information shielding a good strategy?)
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 into estimation
Portman’s Data

- Even with careful estimation using expert data, estimates were still about one-half of the actual time.
- Analyzing the actual time uncovered non-estimated time for other tasks.
  - Machine downtime, meetings, paperwork, high-priority unrelated jobs, company business, 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 a key in 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 increase exponentially by a power of 1.5
Calling the Shot

What does this mean?

- To “call the shot”, keep the following in mind:
  - Use expert estimation and experience (past data)
  - Complex and large projects appear to increase actual effort
  - Keep lines of communication in mind when estimating
  - Consider overhead in estimation
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 (similar to our textbook)
  - Algorithms
  - Control structures
  - Data structures
Summary

- Use the essays to make the connection with good software engineering and project management practices
  - Effort Estimation, Schedules
  - Teams
  - Conceptual Integrity
  - Redesign
  - Communication
Questions?

Annette Tetmeyer
913-897-8578
tetmeyer@ku.edu
The Tar Pit

- Tar pits can trap any type of beast: large or small, quick or slow, single or in groups
- Once trapped, the fierce struggle against the tar pit leads to greater entanglement and ultimate demise
- The entanglements of the tar pit can catch us off guard, drawing us in before we know what is happening
The Tar Pit

- System programming can have a “sticky” nature, much like the tar pits.
- To better understand, look at three elements:
  - The Programming Systems Product
  - Joys of the Craft
  - Woes of the Craft
The Tar Pit

The Programming Systems Product

- Program
  - Garage programming
  - Small team (1-2 people)
  - May produce a fantastic program in record time
  - Author may need to be involved to run the program
  - Must evolve into the *programming product*