Mythical Man-Month - Chapter 3 - The Surgical Team

The Problem:

- Acknowledges significant productivity variations between good and poor programmers.
- Measured differences are surprisingly large: ratios of 10:1 on productivity and 5:1 on program speed and space.
- No correlation observed between experience and performance.
- Argues that coordinating many minds increases costs due to communication and debugging.

Mill's Proposal:

- Harlan Mills proposes a creative solution: divide a large project into segments handled by a team organized like a surgical team.
- The "small sharp team" is often ideal but too slow for large systems.
- The solution is to have a specialized team with a "surgeon" (chief programmer), a "copilot," an "administrator," an "editor," two secretaries, a "program clerk," a "toolsmith," a "tester," and a "language lawyer."
- Each has a specific role to support the "surgeon" and make the process more efficient.

How it works:

- The surgical team structure allows for better efficiency with simpler communication.
- The "surgeon" and "copilot" have a complete understanding of design and code, ensuring conceptual integrity.
- The superior-subordinate relationship in the surgical team eliminates judgment differences, simplifying communication.
- The specialization of roles for the rest of the team is the key to its efficiency.

In summary, Mills's proposal for team composition, based on the surgical model, offers an innovative approach to solving the challenge of efficiently building large systems. Each member has a specific role, contributing to both specialization and collaboration, making the process faster and more cohesive.

Mythical Man-Month - Chapter 4 - Aristocracy, Democracy, and System Design

Conceptual Integrity in Architecture:

- European cathedrals often lack architectural unity due to changes by different builders over generations.
- Reims Cathedral stands out for its conceptual integrity achieved by self-abnegation of builders over eight generations.
- Programming systems often lack conceptual integrity due to the separation of design tasks among many individuals.

Achieving Conceptual Integrity:

- Conceptual integrity is crucial in system design.
- It's better to omit certain features than compromise conceptual integrity.
- Key questions: How to achieve conceptual integrity? Does it imply an elite of architects and subordinate implementers?

Purpose of a Programming System:

- Programming systems aim to make computers easy to use.
- They provide languages and facilities, but the external description is much larger than the computer system itself.
- The ratio of function to conceptual complexity is the ultimate test of system design.

Aristocracy and Democracy in System Design:

- Conceptual integrity suggests a system should be designed by one mind or a few agreeing minds.
- Schedule pressures require involving many hands.
- Two approaches: careful division of labor between architecture and implementation, and structuring implementation teams differently.

Architects vs. Implementers:

- Architecture defines the user interface; implementation describes how it's made to happen.
- Creativity exists in both architecture and implementation.
- Discipline enhances creativity, and constraints can lead to better designs.

Challenges During Implementation:

- Multimillion-dollar mistakes can arise from decisions favoring implementation over architecture.
- Siren song of involving more implementers can lead to schedule issues and decreased conceptual integrity.

Overlap of Specification and Building:

- Overlapping architecture, implementation, and realization is possible.
- Three phases—architecture, implementation, and realization—can proceed in parallel, reducing communication complexities and improving conceptual integrity.
- Conceptual integrity doesn't imply a longer building time; experience shows the opposite.