Introduction to Software Engineering Outline

- **Part 1 - Background and Introductory Information**
  - Software Engineering History
  - Software Development Statistics
  - What is Software Engineering?

- **Part 2 - Software Engineering Concepts**
  - Software Engineering Relationships
  - Principles
  - Development Methods & Techniques

Software Engineering History

- Computers were invented in the 1940’s
- Then - computing programming languages were invented
- Eventually - program language training was developed
- However, training was unable to provide sufficient methods & techniques for developing large reliable systems on time & within budget
- By the late 1960’s, digital computers were less than 25 years old and already facing a software crisis
- Software Engineering term first emerged as title of a 1968 NATO conference [1]

Software Development Statistics

- 1979 General Accounting Office Report [2]
  - 50% + of contracts had cost overruns
  - 60% + of contracts had schedule overruns
  - 45% + of software contracted for could not be used
  - 29% + of software was paid for and never delivered
  - 22% + of software contracted for had to be reworked/modified to be used

Software Development Statistics

- **Other Studies**
  - **1982 - Tom DeMarco**
    - **25% of large systems development projects never finished**
  - **1991 Capers Jones Study**
    - **Average Management Information System* project is 1 year late and 100% over budget**
- **Software crisis stubbornly persists**

*** Management Information System - a system for providing information to support organizational activities and management functions.**

Summary

- While there has been some progress, there still are serious problems to overcome in software development
  - schedule and cost estimates are still inaccurate
  - productivity of developers is not keeping up with the demand
  - quality of software is not meeting customer expectations
- Still a need to bring engineering discipline into the software process

Case Studies - Software Crisis

- **1. The Healthcare.gov rollout: In 2013, the launch of the Healthcare.gov website, which was intended to provide a marketplace for individuals to purchase health insurance, was marred by technical issues that made the site largely unusable. The website was built by engaging contractors, and testing was reportedly inadequate. As a result, the site experienced frequent crashes and error messages, making it difficult for users to use/enroll.**

- **The problems were caused by a lack of proper testing and resulted in significant delays and costs, with the final price tag for the troubled rollout estimated at over $2 billion.**

Case Studies - Software Crisis

- **2. The London Stock Exchange outage: In 2019, a software upgrade caused the London Stock Exchange to go offline for several hours, resulting in a loss of approximately £10 billion in trades. The outage was caused by a failure to properly test the software before implementation. The cause of the failure was traced to a flaw in the system's software that was not detected during testing. The outage had significant economic consequences, as many traders were unable to execute trades and some companies saw their stock prices drop.**

Case Studies - Software Crisis

- 3. The Mars Climate Orbiter mission: In 1999, the Mars Climate Orbiter, a spacecraft launched by NASA, was lost due to a software error that caused it to crash into the surface of Mars. The error was caused by a failure to properly test the software, and the mission was a total loss. The error was caused by a discrepancy between the software used to navigate the spacecraft and the software used to calculate the spacecraft's position. The error was not detected during testing and resulted in the loss of the spacecraft.

Case Studies - Software Crisis

- 4. The Therac-25 radiation overdose: In the 1980s, the Therac-25, a medical device used to deliver radiation therapy, caused multiple instances of patient death due to software errors that caused the device to deliver excessive doses of radiation. The errors were caused by a lack of proper testing and resulted in significant harm to patients. The errors were caused by a flaw in the software that was not detected during testing, resulting the device delivering lethal doses of radiation.

- 5. The Airbus A320 crash: In 1988, an Airbus A320 crashed into a mountain in France, killing all passengers and crew on board.

Software Engineering Definition

- **The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. [IEEE Std 610.12-1990]**

- **Institution of engineering discipline in the manufacturing of computer software**

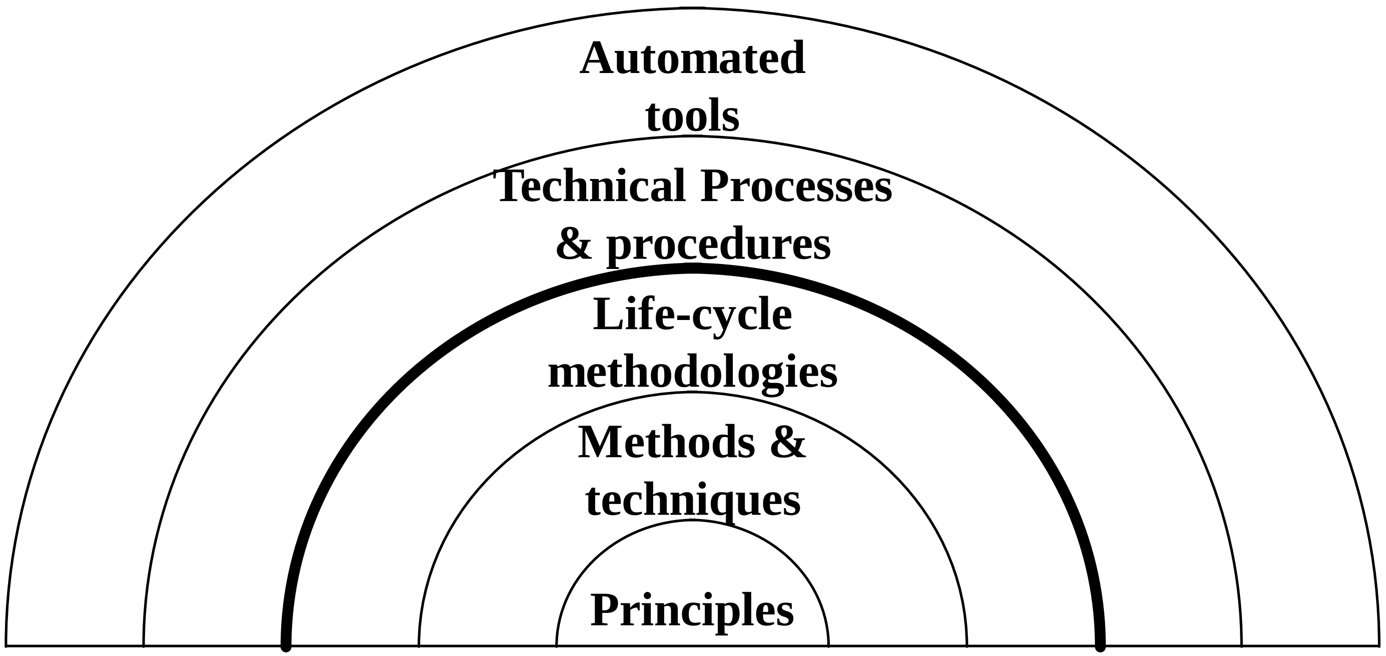
Software Engineering Definition

- **The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. [IEEE Std 610.12-1990]**
- **Institution of engineering discipline in the manufacturing of computer software**

Part 2 - Software Engineering Concepts

- **Part 1 - Background and Introductory Information**
- **Part 2 - Software Engineering Concepts**
  - Software Engineering Relationships
  - Principles
  - Development Methods & Techniques
  - Management Methods & Techniques
  - Life-Cycle Methodologies
  - Software Engineering Processes & Procedures
  - Automated Tools

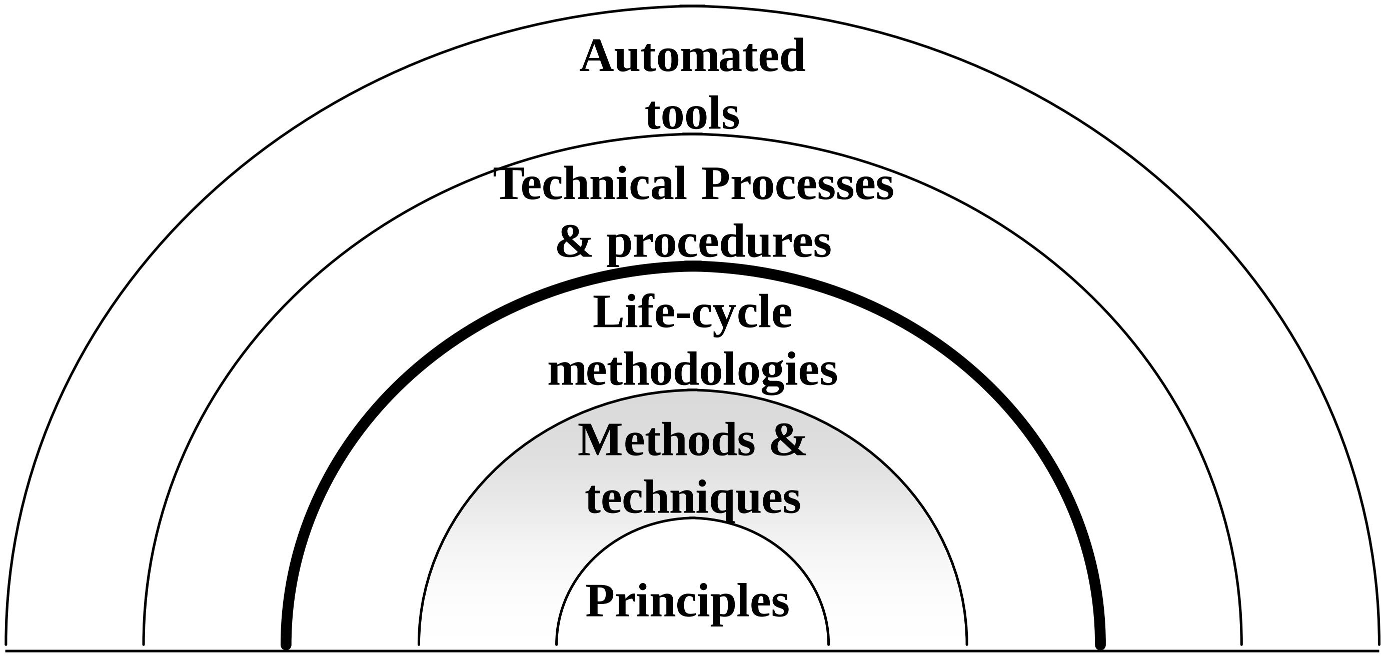
Software Engineering Relationships



Selected Software Engineering Principles

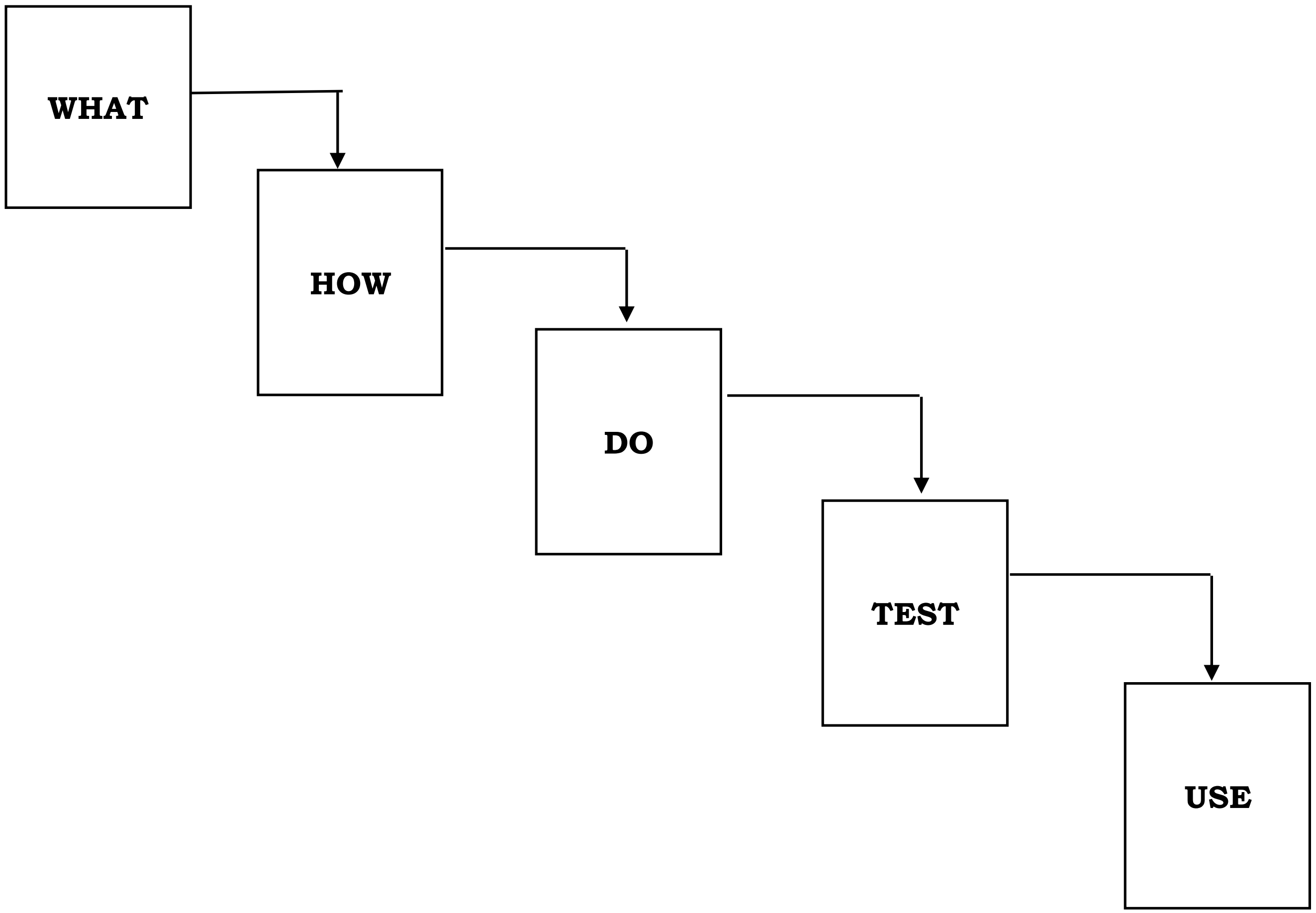
- Rigor and Formality
- Modularity
- Abstraction
- Anticipation of Change
- Generality
- Incrementality

Methods & Techniques

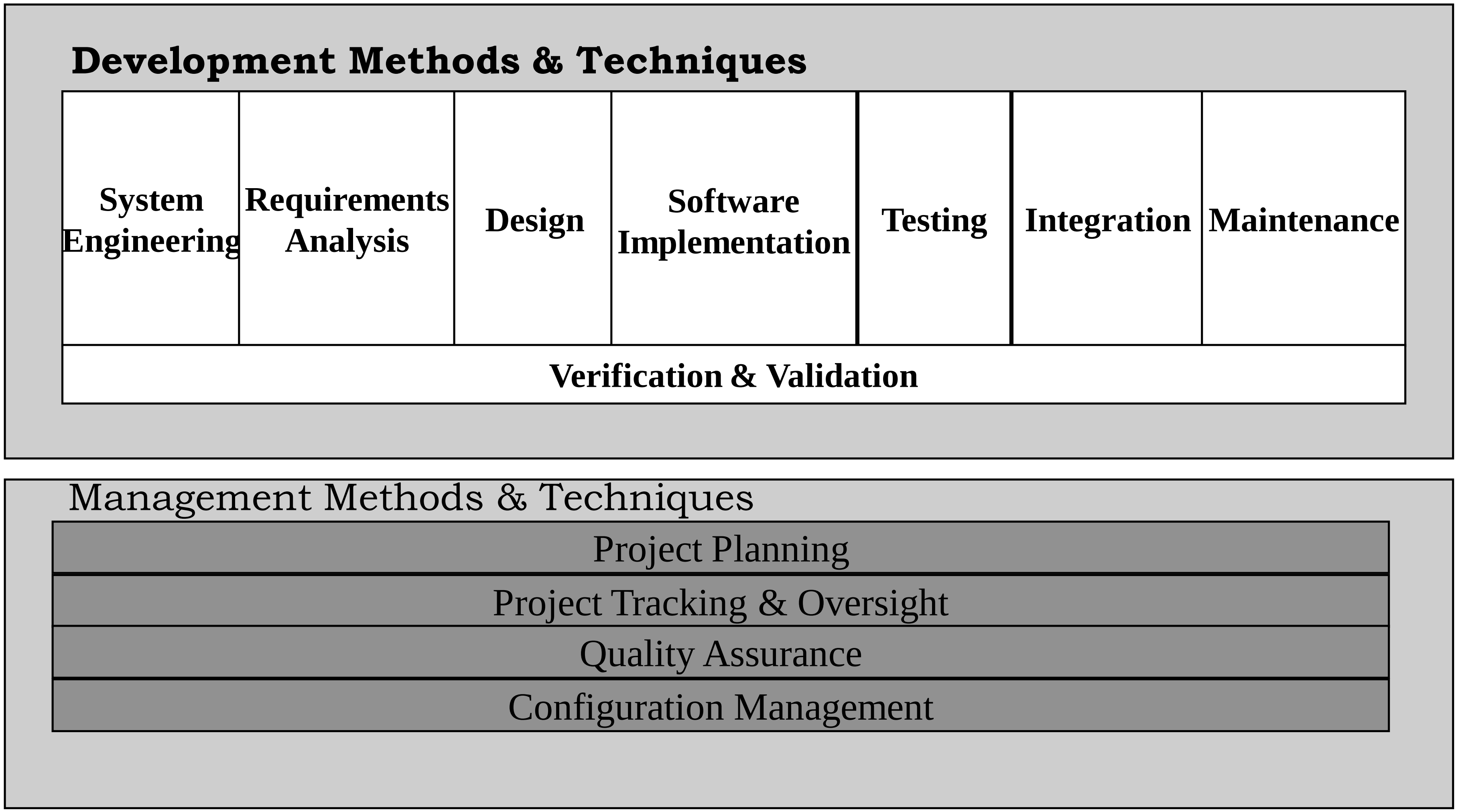


Basic Problem Solving Flow

SDLC Methods:



Development Methods & Techniques


Development Methods & Techniques
System Engineering
Requirements Analysis
Design
Software Implementation
Testing
Integration
Maintenance
Verification & Validation
Management Methods & Techniques
Project Planning
Project Tracking & Oversight
Quality Assurance
Configuration Management