Software Engineering

- Some realities:
  - a concerted effort should be made to understand the problem before a software solution is developed
  - design becomes a pivotal activity
  - software should exhibit high quality
  - software should be maintainable

- The seminal definition:
  - [Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.
Software Engineering

- The IEEE definition:
  - **Software Engineering:**
  - (1) *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*
  - (2) *The study of approaches as in (1).*
A Process Framework for SE

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment
Umbrella Activities for SE Projects

- Software project tracking and control
- Risk management
- Software quality assurance
- Technical reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production
Software Engineering Practice

- Polya suggests:
  1. Understand the problem (communication and analysis).
  2. Plan a solution (modeling and software design).
  3. Carry out the plan (code generation).
  4. Examine the result for accuracy (testing and quality assurance).
Understand the Problem

- **Who has a stake in the solution to the problem?** That is, who are the stakeholders?
- **What are the unknowns?** What data, functions, and features are required to properly solve the problem?
- **Can the problem be compartmentalized?** Is it possible to represent smaller problems that may be easier to understand?
- **Can the problem be represented graphically?** Can an analysis model be created?
Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?
Carry Out the Plan

- Does the solution conform to the plan? Is source code traceable to the design model?
- Is each component part of the solution provably correct? Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?
Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?
Hooker’s General Principles

1: The Reason It All Exists
2: KISS (Keep It Simple, Stupid!)
3: Maintain the Vision
4: What You Produce, Others Will Consume
5: Be Open to the Future
6: Plan Ahead for Reuse
7: Think!
Software Myths

- If we get behind schedule, we can add more programmers and catch up

In “The Mythical Man-Month” Fred Brooks observed that “adding people to a late software project makes it later”

Why? As new people are added, people who were working must spend time educating new comers thereby reducing the amount of time spent on productive development efforts.