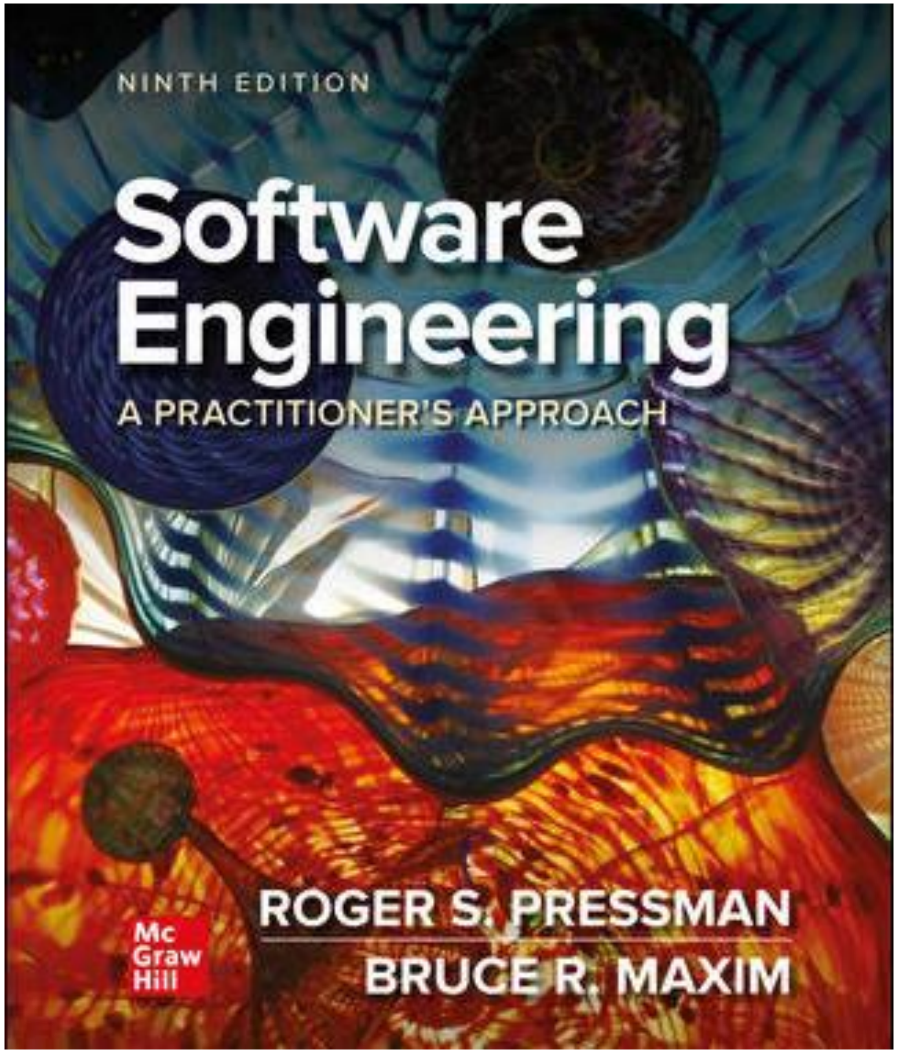
Because learning changes everything.®

Chapter 1

Software and Software Engineering

Introduction



Nature of Software – Defining Software

Software is:

1) *Instructions (computer programs) that when executed provide desired features, function, and performance;*
2) *Data structures that enable the programs to adequately manipulate information.*
3) *Documentation that describes the operation and use of the programs.*

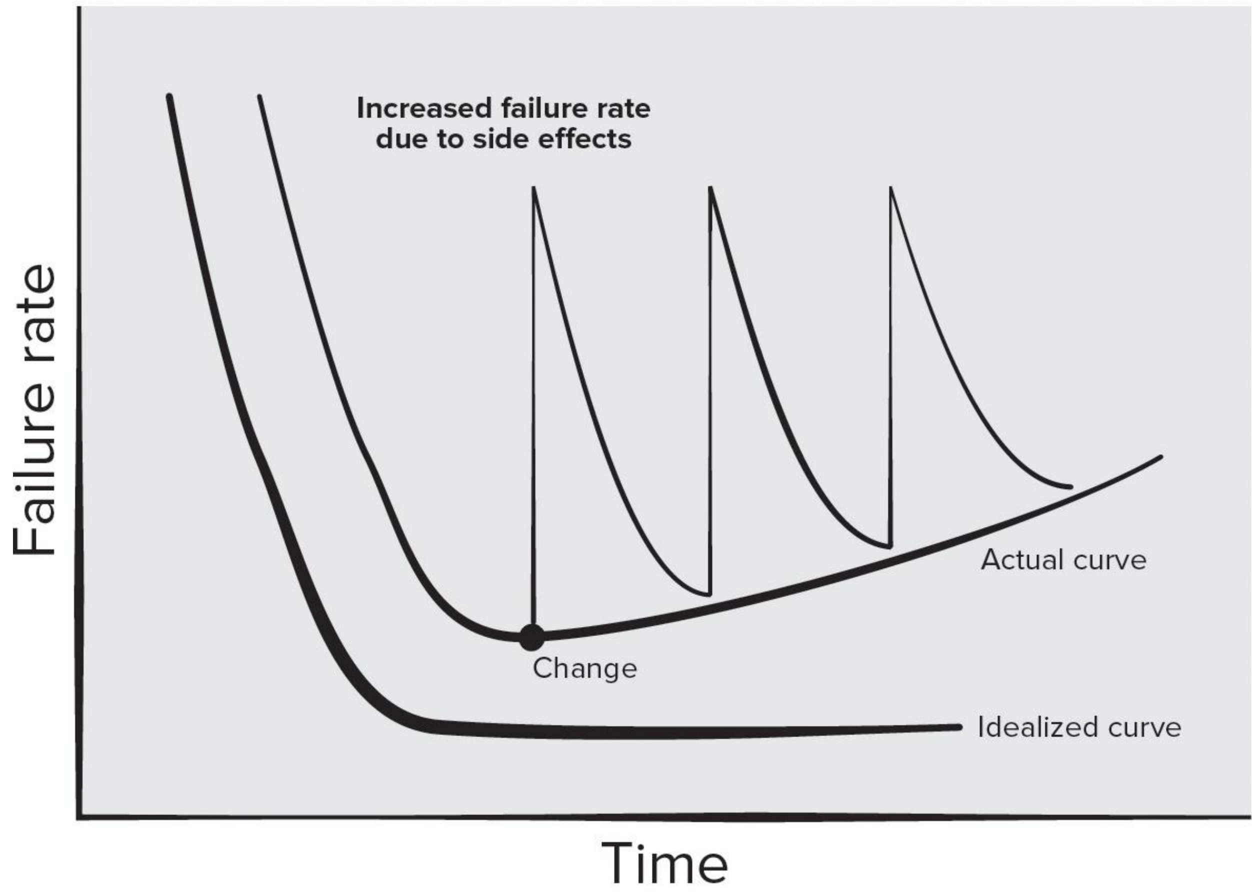
What is Software?

- *Software is developed or engineered it is not manufactured in the classical sense.*
- *Software doesn't "wear out“ but is does deteriorate.*
- *Although the industry is moving toward component-based construction, most software continues to be custom-built.*

Software Application Domains

- System software.
- Application software.
- Engineering/Scientific software.
- Embedded software.
- Product-line software.
- Web/Mobile applications.
- AI software (robotics, neural nets, game playing).

Wear versus Deterioration



Access the text alternative for slide images.

Legacy Software

Why must software change?

- Software must be *adapted* to meet the needs of new computing environments or technology.
- Software must be *enhanced* to implement new business requirements.
- Software must be *extended* to make it interoperable with other more modern systems or databases.
- Software must be *re-architected* to make it viable within a network environment.

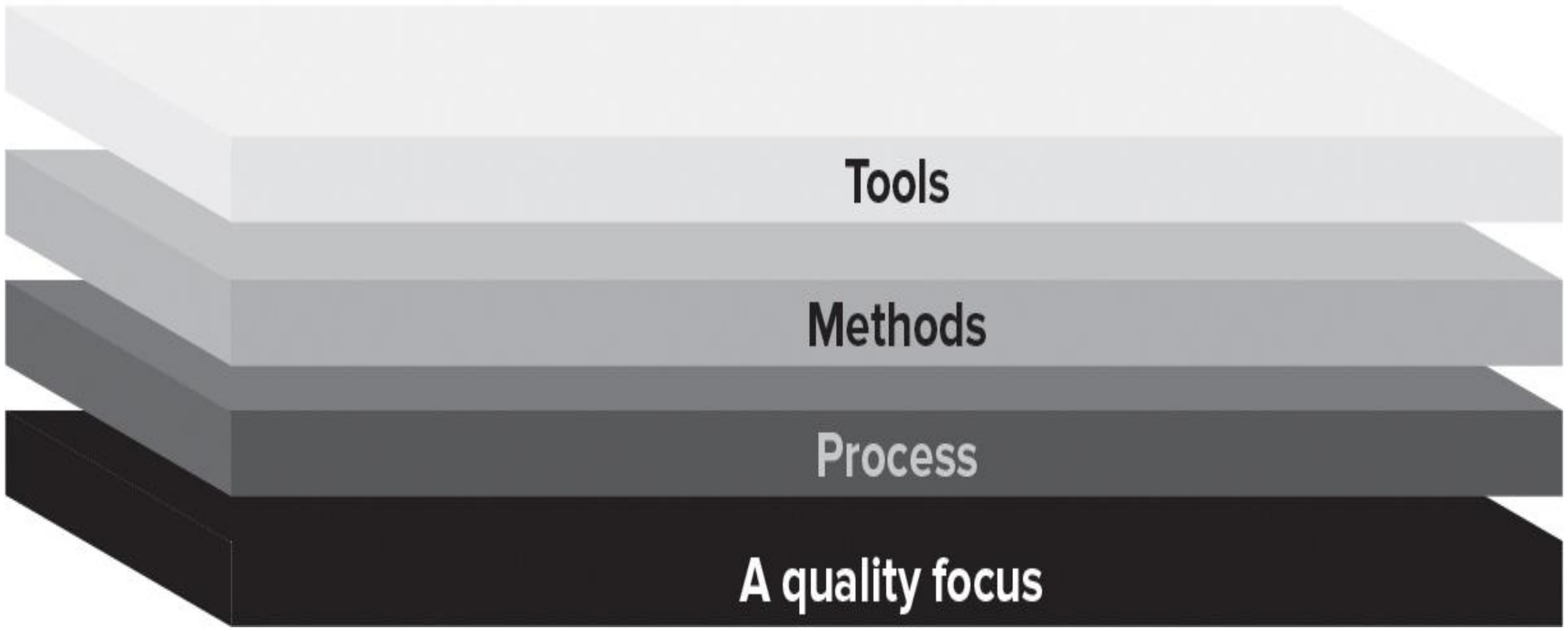
Defining the Discipline

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).*

Software Engineering Layers



Process Framework Activities

Communication.

Planning.

Modeling.

- Analysis of requirements.
- Design.

Construction:

- Code generation.
- Testing.

Deployment.

Umbrella Activities

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

Process Difference Requiring Adaptation

- Overall flow of activities, actions, and tasks and the interdependencies among them.
- Degree to which actions and tasks are defined within each framework activity.
- Degree to which work products are identified and required.
- Manner which quality assurance activities are applied.
- Manner in which project tracking and control activities are applied.
- Overall degree of detail and rigor with which the process is described.
- Degree to which the customer and other stakeholders are involved with the project.
- Level of autonomy given to the software team.
- Degree to which team organization and roles are prescribed.

Essence of 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 result for accuracy* (testing & 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 a 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?

Carryout 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* – provide value to users.
2. *KISS (Keep It Simple, Stupid!)* – design simple as it can be.
3. *Maintain the Vision* – clear vision is essential.
4. *What You Produce, Others Will Consume.*
5. *Be Open to the Future* - do not design yourself into a corner.
6. *Plan Ahead for Reuse* – reduces cost and increases value.
7. *Think!* – placing thought before action produce results.

How it all Starts – SafeHome Begins

Every software project is precipitated by some business need—

- The need to correct a defect in an existing application;
- The need to the need to adapt a ‘legacy system’ to a changing business environment;
- The need to extend the functions and features of an existing application, or
- The need to create a new product, service, or system.

McGraw Hill Education

Because learning changes everything.®

www.mheducation.com