Chapter 2

- **Software Engineering**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 7/e

by Roger S. Pressman

Table of Contents

2.1 Software Engineering

- Realities about software development today:
  - 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

. . .

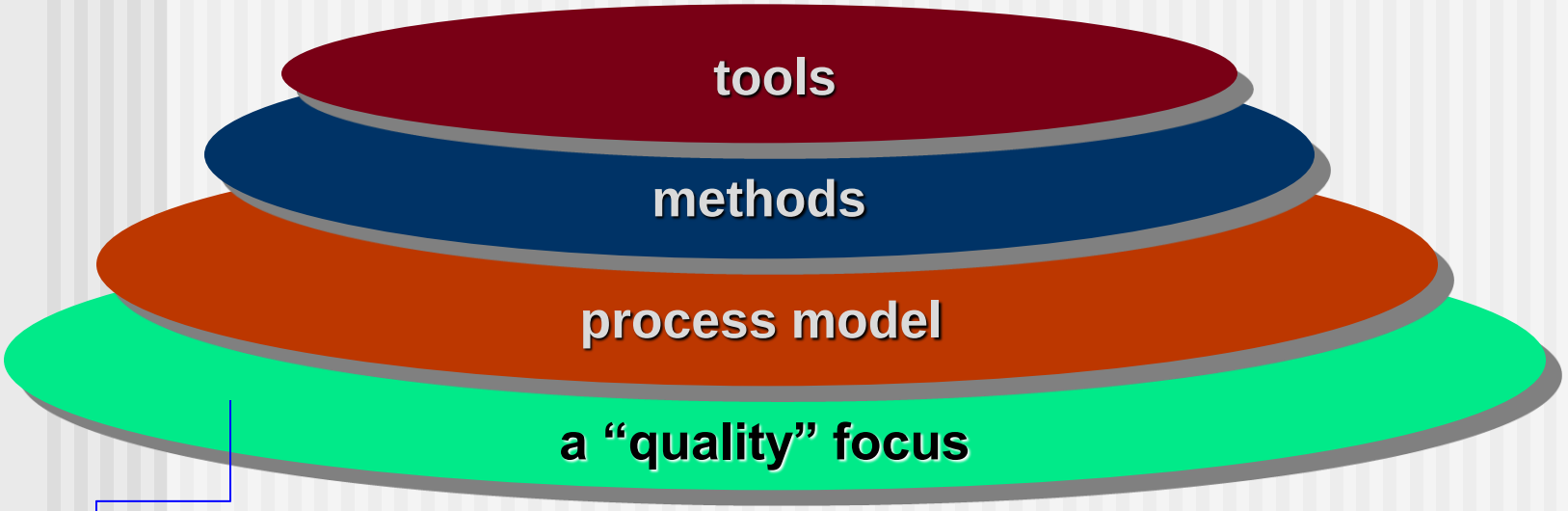
Software Engineering

- Definitions
  - The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.
  - The IEEE definition:

(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 Layered Technology

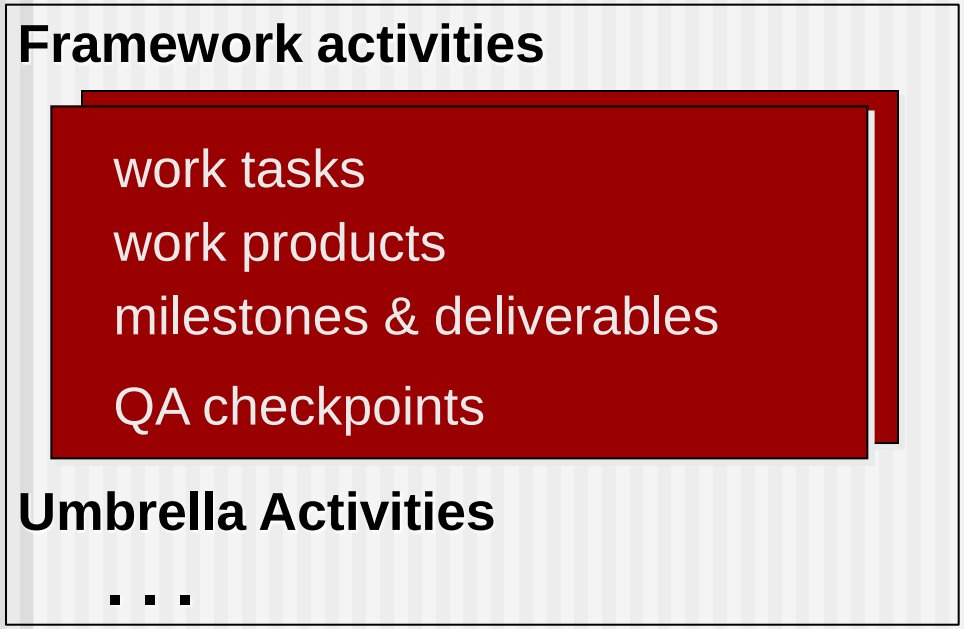


Requires a good design and a use of various management techniques

Software Engineering

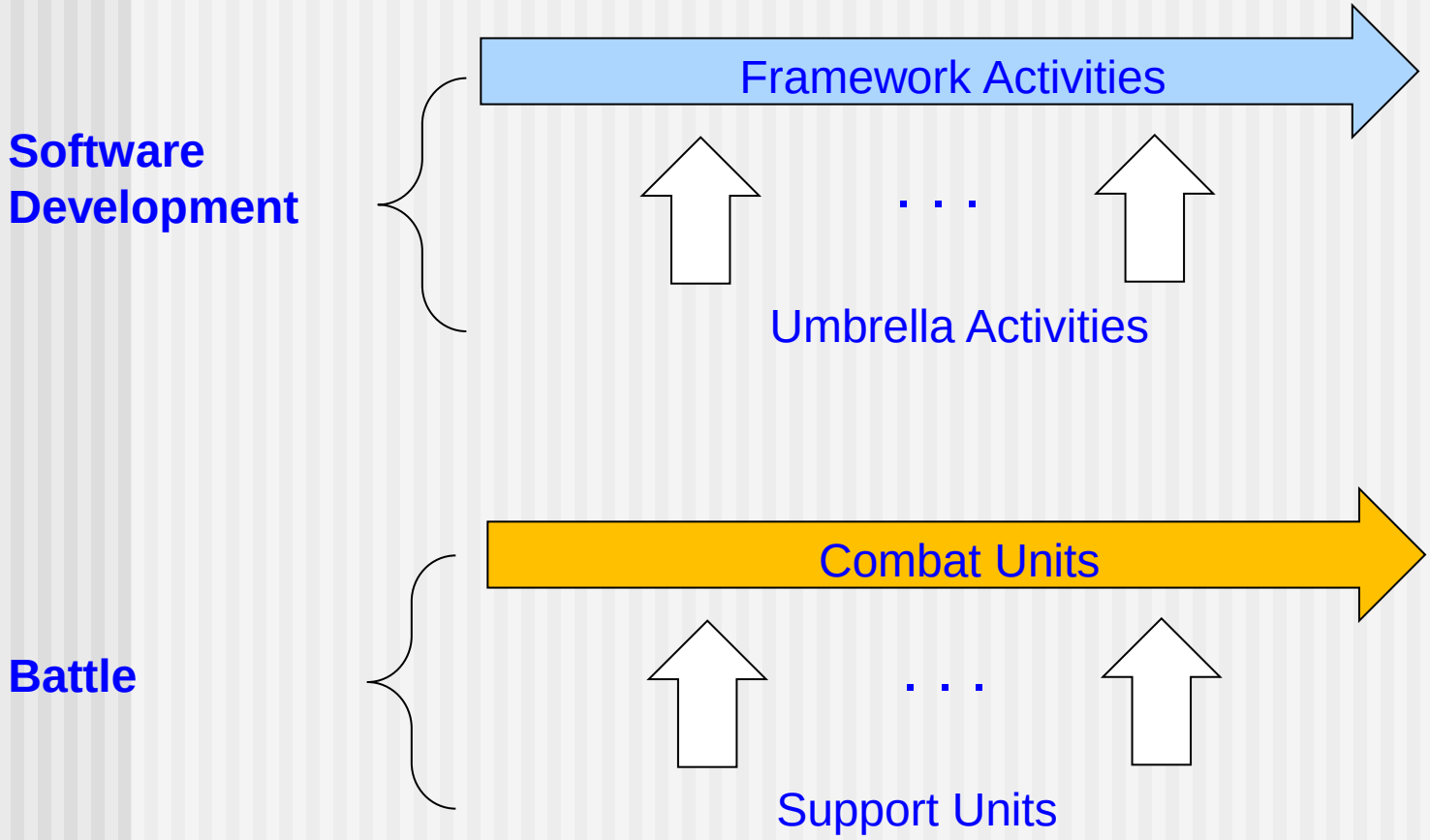
2.2 The Software Process

2.2.1 The Process framework

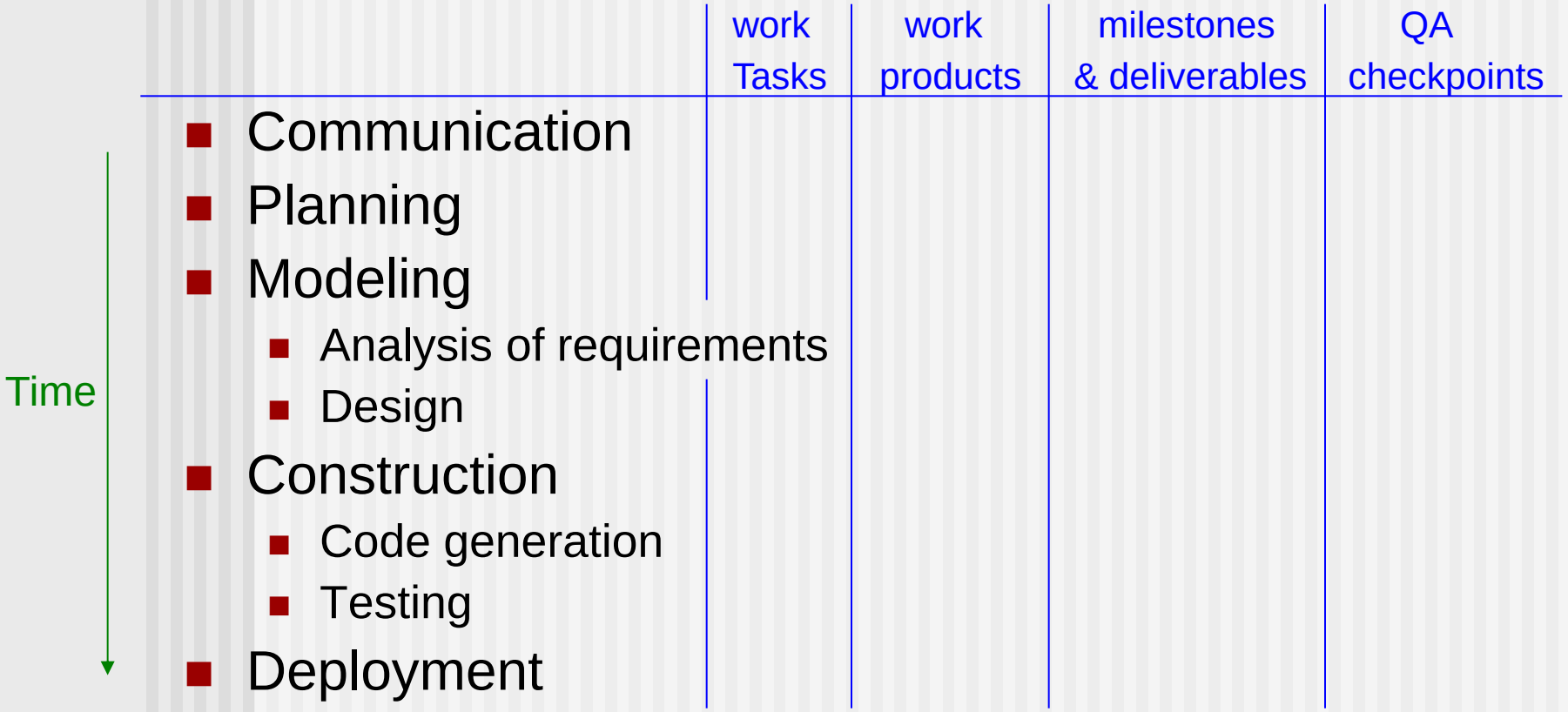


Software Process: "The related set of activities and processes that is involved in developing and evolving a software system. - *Sommerville*

Milestone – Key stages in the project where progress can be assessed



Framework Activities



2.2.2 Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

2.2.3 Adapting a Process Model

Rather than starting from scratch, when appropriate, start with a well-known process and customize it in terms of:

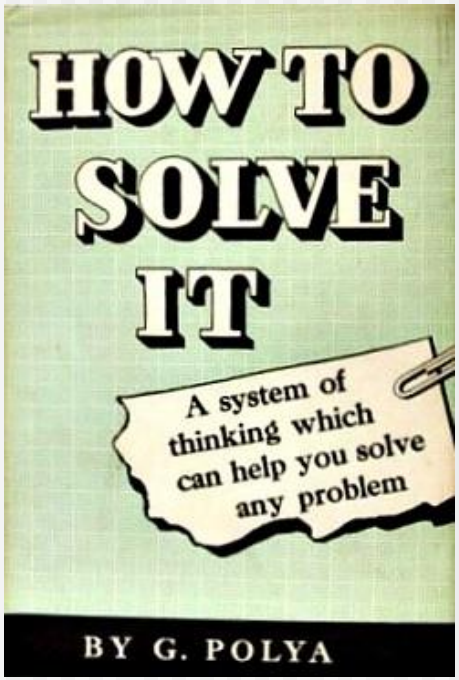
- The overall flow of activities, actions, and tasks and the interdependencies among them
- The degree to which actions and tasks are defined within each framework activity
- The degree to which work products are identified and required
- The manner which quality assurance activities are applied
- The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

2.3 Software Engineering Practice

2.3.1 The Essence of (SW Development) 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).

☛ General Problem Solving Steps



(1) Understand the Problem

- *Who has a stake (= interest) 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?

(2) 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?

(3) 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?

(4) 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?

2.3.2 Hooker's General Principles for software development

1: *The Reason It All Exists*

A software system exists "to provide value to its users" .

2: *KISS (Keep It Simple, Stupid!)*

Typically it takes thought and work over multiple iterations to simplify.

3: *Maintain the Vision*

Without vision, conflicting ideas will coexist in a system.

4: *What You Produce, Others Will Consume*

5: *Be Open to the Future*

6: *Plan Ahead for Reuse*

7: *Think!*

"Think before act" almost always produces better results.

David Hooker
Builder of Software
Software Architect

2.4 Software Development Myths

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth,

but …

- Invariably lead to bad decisions,

therefore …

- Insist on reality as you navigate your way through software engineering

Management myths

- We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?
- If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde":
  An armed mob with no internal organization which blindly attacks whatever is nearby).
- If I decide to outsource the software project to a third party,
  I can just relax and let that firm build it.

Customer myths

- A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.
- Software requirements continually change, but change can be easily accommodated because software is flexible.

Practitioner's myths

- Once we write the program and get it to work, our job is done.
- Until I get the program "running" I have no way of assessing its quality.
- The only deliverable work product for a successful project is the working program.
- Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

2.5 How It all Starts

- *SafeHome:*
  - Every software project is precipitated by some **business need—**
    - the need to correct a defect in an existing application;
    - 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.

The *SafeHome* project will be used throughout this book to illustrate the inner workings of a project team as it builds a software product.