

Chapter 2

■ Software Engineering

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 8/e

by Roger S. Pressman and Bruce R. Maxim

Slides copyright © 1996, 2001, 2005, 2009, 2014 by Roger S. Pressman

For non-profit educational use only

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 8/e*. Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.

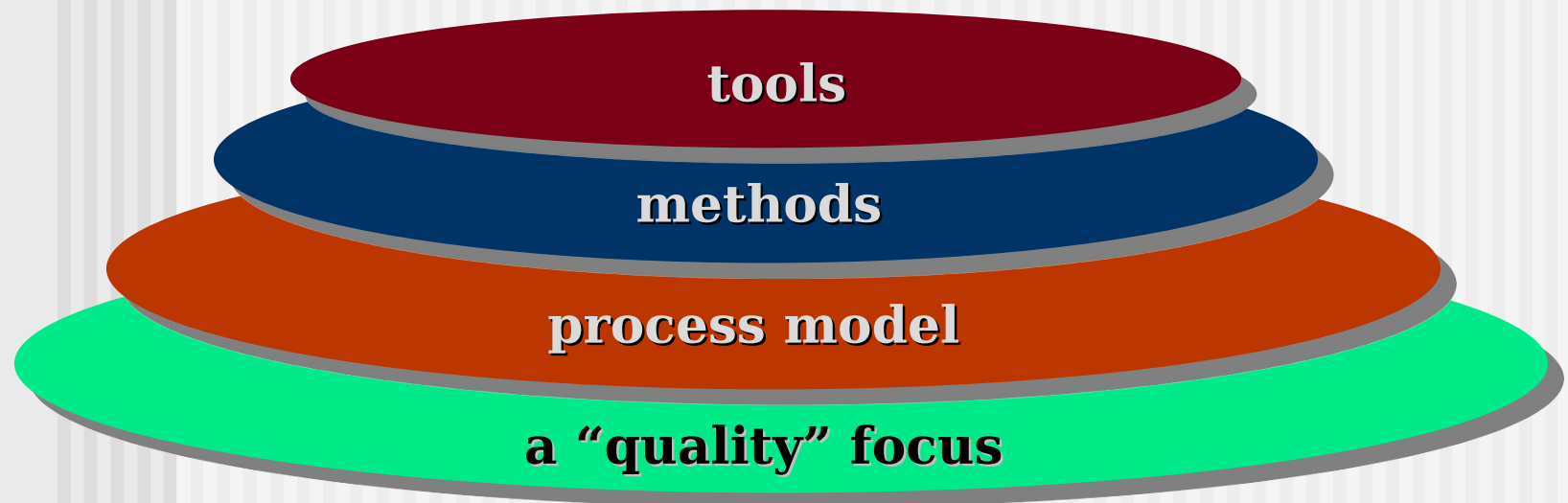
Software Engineering

- 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 Layered Technology



Software Engineering

A Process Framework

Process framework

Framework activities

work tasks

work products

milestones & deliverables

QA checkpoints

Umbrella Activities

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

Adapting a Process Model

- 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

The Essence of 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

- 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

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 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.