西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

Software Engineering

Chapter 2
Software Engineering

Reproduced by Ning Li , 2022

Contents

- 2.1 Defining the Discipline
- 2.2 The Software Process
  - 2.2.1 The process framework
  - 2.2.2 Umbrella Activities
  - 2.2.3 Process Adaptation
- 2.3 Software Engineering Practice
  - 2.3.1 The Essence of Practice
  - 2.3.2 General Principles
- 2.4 Software Development Myths
- 2.5 How It All Starts

2.1 Software Engineering

- **Some realities:**
  - *a concerted effort should be made to **understand the problem** before a software solution is developed*
  - *__design__ becomes a key 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.*

2.1 Software Engineering

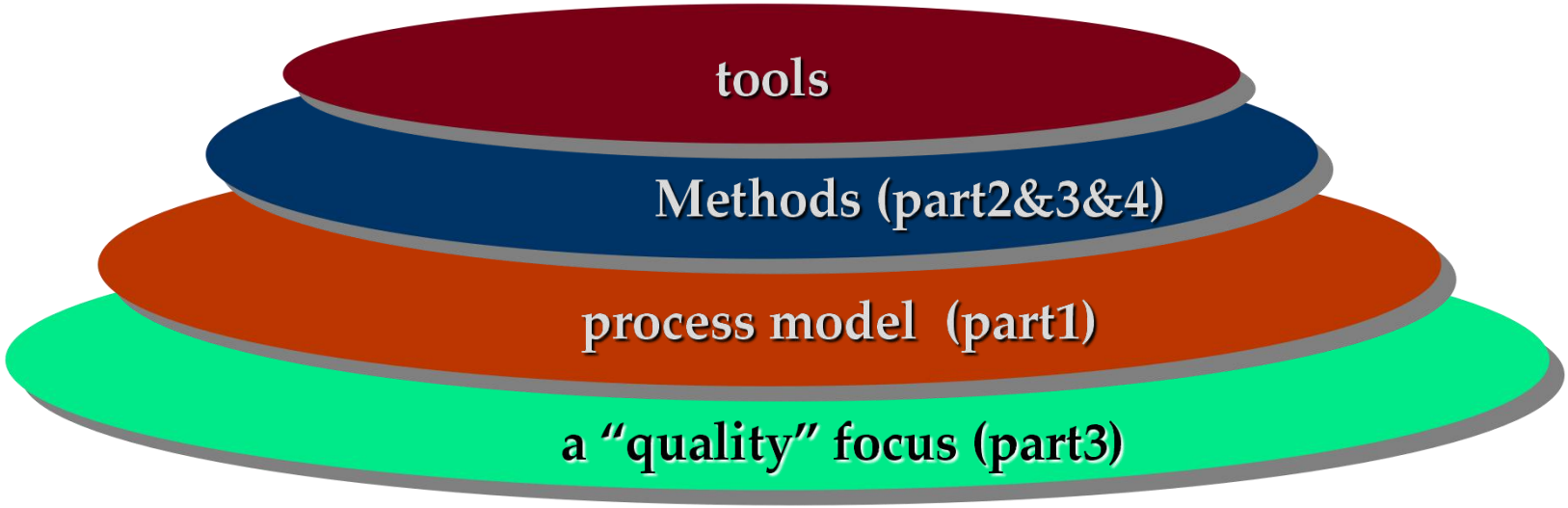
- **The IEEE definition:**

  – *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).

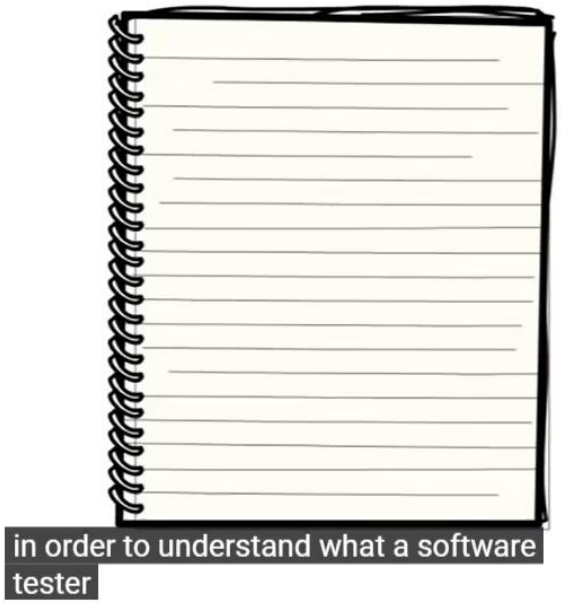
A Layered Technology



Software Engineering

Software Development Life Cycle

Watch a video: SDLC (9 minutes)



https://www.youtube.com/watch?v=i-QyW8D3ei0

2.2.1 A Process Framework

Process framework

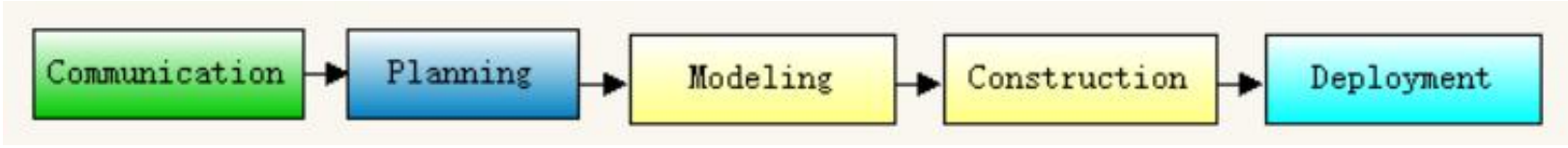
- Framework activities

work tasks
work products
milestones & deliverables
QA checkpoints

- Umbrella activities

Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment



主观题 10分

设置

Would you like to share your experience or lesson in your past development? (e.g. your database course project)?

正常使用主观题需2.0以上版本雨课堂

作答

2.2.2 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

2.2.3 Process Adaptation

How to do the process adaptation?

- 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

2.2.3 Process Adaptation

- 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

2.2.3 Process Adaptation

Process not a rigid prescription that must be followed dogmatically by a software team.

it should be

- ✓ agile
- ✓ adaptable

Contents

- 2.1 Defining the Discipline
- 2.2 The Software Process
  - 2.2.1 The process framework
  - 2.2.2 Umbrella Activities
  - 2.2.3 Process Adaptation
- 2.3 Software Engineering Practice
  - 2.3.1 The Essence of Practice
  - 2.3.2 General Principles
- 2.4 Software Development Myths
- 2.5 How It All Starts

2.3.1 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).

1. 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?

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 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*

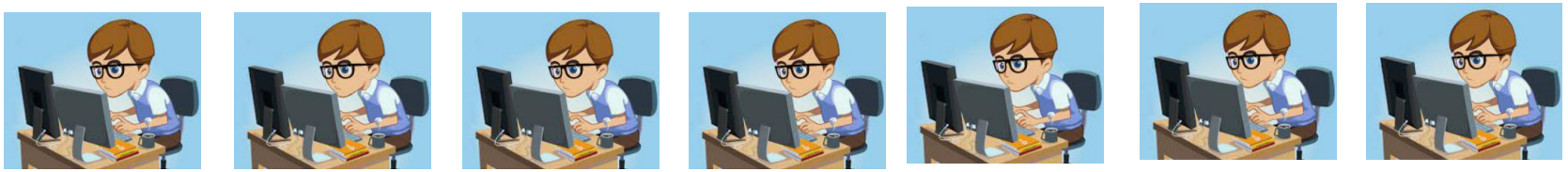
7: *Think!*

2.4 Software Myths

- Managers

Myth: *If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).*

Reality: Software development is not a mechanistic process like manufacturing. In the words of Brooks [Bro95]: "adding people to a late software project makes it later." At first, this statement may seem counterintuitive. However, as new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well-coordinated manner.



2.4 Software Myths

- Customers

Myth: *Software requirements continually change, but change can be easily accommodated because software is flexible.*

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small.[16] However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources and major design modification.

2.4 Software Myths

- Practitioners

Myth: *Once we write the program and get it to work, our job is done.*

Reality: Someone once said that "the sooner you begin 'writing code,' the longer it'll take you to get done." Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.

Myth: *Until I get the program "running" I have no way of assessing its quality.*

Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project—*the technical review.* Software reviews (described in Chapter 15) are a "quality filter" that have been found to be more effective than testing for finding certain classes of software defects.

technical review

2.4 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

2.5 How It all Starts

- 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.

2.5 How It all Starts

SafeHome[9]

How a Project Starts

The scene: Meeting room at CPI Corporation, a (fictional) company that makes consumer products for home and commercial use.

The players: Mal Golden, senior manager, product development; Lisa Perez, marketing manager; Lee Warren, engineering manager; Joe Camalleri, executive vice president, business development

The conversation:

Joe: Okay, Lee, what's this I hear about your folks developing a what? A generic universal wireless box?

Lee: It's pretty cool . . . about the size of a small matchbook . . . we can attach it to sensors of all kinds, a digital camera, just about anything. Using the 802.11n wireless protocol. It allows us to access the device's output without wires. We think it'll lead to a whole new generation of products.

Joe: You agree, Mal?

Mal: I do. In fact, with sales as flat as they've been this year, we need something new. Lisa and I have been doing a little market research, and we think we've got a line of products that could be big.

Joe: How big . . . bottom line big?

Mal (avoiding a direct commitment): Tell him about our idea, Lisa.

Lisa: It's a whole new generation of what we call "home management products." We call 'em *SafeHome*. They use the new wireless interface, provide homeowners or small-businesspeople with a system that's controlled by their PC—home security, home surveillance, appliance and device control—you know, turn down the home air conditioner while you're driving home, that sort of thing.

Lee (jumping in): Engineering's done a technical feasibility study of this idea, Joe. It's doable at low manufacturing cost. Most hardware is off the shelf. Software is an issue, but it's nothing that we can't do.

Joe: Interesting. Now, I asked about the bottom line.

Mal: PCs and tablets have penetrated over 70 percent of all households in the USA. If we could price this thing right, it could be a killer app. Nobody else has our wireless box . . . it's proprietary. We'll have a 2-year jump on the competition. Revenue? Maybe as much as $30 to $40 million in the second year.

Joe (smiling): Let's take this to the next level. I'm interested.

Review

- What is software

  Definition: Instruction + data struct + documentation

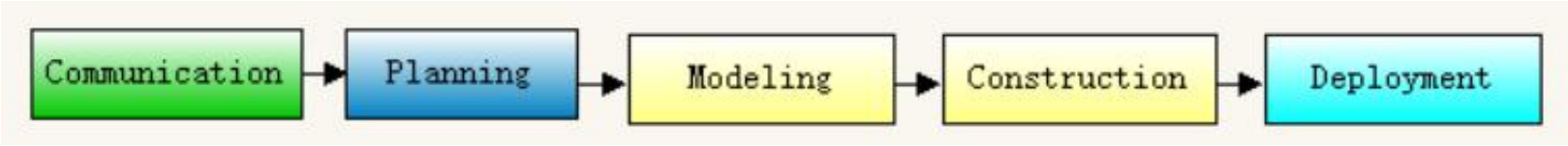
  Nature: changing!

- What is software engineering

  Application of engineering to software.

  It encompasses process, methods, and tools.

  Five activities:



Assignment 1 - 1

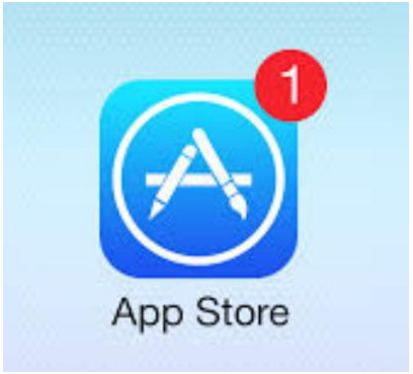
- 1. Many modern applications change (upgrade) frequently before they are presented to the end user and then after the first version has been put into use.

  Question:

  Suggest a few ways to build software to stop deterioration due to change.

EMAIL to us:
李宁 lining@nwpu.edu.cn
杨雨晨(TA) 480755534@qq.com

(Deadline: 17 Jan)



Assignment 1 - 2

- **2. Search and read software failure event (select one)**

  - Therac-25
  - Ariane-5 rocket
  - NASA Mars Phobos 1
  - .....

Question:

1) event description and related materials (news, videos, etc.)

2) failure reason and your own thoughts/analysis

3) process improvement solution

Assignment 1 - 3

- 3. As software becomes more pervasive, risks to the public (due to faulty programs) become an increasingly significant concern.

Qestion:

Develop a doomsday but realistic scenario in which the failure of a computer software could do great harm, either economic or human. Then tell us how to avoid it?

THE END