Software Engineering

Chapter 5
Practice: A Generic View

Moon kun Lee
Division of Electronics & Information Engineering
Chonbuk National University

What is “Practice”?

- Practice is a broad array of concepts, principles, methods, and tools that you must consider as software is planned and developed.
- It represents the details—the technical considerations and how to’s—that are below the surface of the software process—the things that you’ll need to actually build high-quality computer software.

The Essence of Practice

- George Polya, in a book written in 1945 (!), describes the essence of software engineering practice …
  - *Understand the problem* (communication and analysis).
  - *Plan a solution* (modeling and software design).
  - *Carry out the plan* (code generation).
  - *Examine the result for accuracy* (testing and quality assurance).
- At its core, good practice is common-sense problem solving

Core Software Engineering Principles

- Provide value to the customer and the user
- KIS—keep it simple!
- Maintain the product and project “vision”
- What you produce, others will consume
- Be open to the future
- Plan ahead for reuse
- Think!

Software Engineering Practices

- Consider the generic process framework
  - Communication
  - Planning
  - Modeling
  - Construction
  - Deployment
- Here, we'll identify
  - Underlying principles
  - How to initiate the practice
  - An abbreviated task set

Communication Practices

- Principles
  - Listen
  - Prepare before you communicate
  - Facilitate the communication
  - Face-to-face is best
  - Take notes and document decisions
  - Collaborate with the customer
  - Stay focused
  - Draw pictures when things are unclear
  - Move on …
  - Negotiation works best when both parties win.

Communication Practices

- Initiation
  - The parties should be physically close to one another
  - Make sure communication is interactive
  - Create solid team "ecosystems"
  - Use the right team structure
- An abbreviated task set
  - Identify who it is you need to speak with
  - Define the best mechanism for communication
  - Establish overall goals and objectives and define the scope
  - Get more detailed
    - Have stakeholders define scenarios for usage
    - Extract major functions/features
  - Review the results with all stakeholders

Planning Practices

- Principles
  - Understand the project scope
  - Involve the customer (and other stakeholders)
  - Recognize that planning is iterative
  - Estimate based on what you know
  - Consider risk
  - Be realistic
  - Adjust granularity as you plan
  - Define how quality will be achieved
  - Define how you'll accommodate changes
  - Track what you've planned

Planning Practices

- Initiation
  - Ask Boehm's questions
    - Why is the system begin developed?
    - What will be done?
    - When will it be accomplished?
    - Who is responsible?
    - Where are they located (organizationally)?
    - How will the job be done technically and managerially?
    - How much of each resource is needed?

Planning Practices

- An abbreviated task set
  - Re-assess project scope
  - Assess risks
  - Evaluate functions/features
  - Consider infrastructure functions/features
  - Create a coarse granularity plan
    - Number of software increments
    - Overall schedule
    - Delivery dates for increments
  - Create fine granularity plan for first increment
  - Track progress

Modeling Practices

- We create models to gain a better understanding of the actual entity to be built
- *Analysis models* represent the customer requirements by depicting the software in three different domains: the information domain, the functional domain, and the behavioral domain.
- *Design models* represent characteristics of the software that help practitioners to construct it effectively: the architecture, the user interface, and component-level detail.

Analysis Modeling Practices

- Analysis modeling principles
  - Represent the information domain
  - Represent software functions
  - Represent software behavior
  - Partition these representations
  - Move from essence toward implementation
- Elements of the analysis model (Chapter 8)
  - Data model
  - Flow model
  - Class model
  - Behavior model

Design Modeling Practices

- Principles
  - Design must be traceable to the analysis model
  - Always consider architecture
  - Focus on the design of data
  - Interfaces (both user and internal) must be designed
  - Components should exhibit functional independence
  - Components should be loosely coupled
  - Design representation should be easily understood
  - The design model should be developed iteratively
- Elements of the design model
  - Data design
  - Architectural design
  - Component design
  - Interface design

Construction Practices

- Preparation principles: *Before you write one line of code, be sure you:*
  - Understand of the problem you're trying to solve (see communication and modeling)
  - Understand basic design principles and concepts.
  - Pick a programming language that meets the needs of the software to be built and the environment in which it will operate.
  - Select a programming environment that provides tools that will make your work easier.
  - Create a set of unit tests that will be applied once the component you code is completed.

Construction Practices

- Coding principles: *As you begin writing code, be sure you:*
  - Constrain your algorithms by following structured programming [BOH00] practice.
  - Select data structures that will meet the needs of the design.
  - Understand the software architecture and create interfaces that are consistent with it.
  - Keep conditional logic as simple as possible.
  - Create nested loops in a way that makes them easily testable.
  - Select meaningful variable names and follow other local coding standards.
  - Write code that is self-documenting.
  - Create a visual layout (e.g., indentation and blank lines) that aids understanding.

Construction Practices

- Validation Principles: *After you've completed your first coding pass, be sure you:*
  - Conduct a code walkthrough when appropriate.
  - Perform unit tests and correct errors you've uncovered.
  - Refactor the code.

Construction Practices

- Testing Principles
  - All tests should be traceable to requirements
  - Tests should be planned
  - The Pareto Principle applies to testing
  - Testing begins “in the small” and moves toward “in the large”
  - Exhaustive testing is not possible
    - Pareto principle: 80% of all errors uncovered during testing will likely be traceable to 20% of all program components.

Deployment Practices

- Principles
  - Manage customer expectations for each increment
  - A complete delivery package should be assembled and tested
  - A support regime should be established
  - Instructional materials must be provided to end-users
  - Buggy software should be fixed first, delivered later