Software Engineering

HUANG Jie
School of Computer Science and Technology
Tongji University
2025

同济大学
TONGJI UNIVERSITY

Lesson 2 Software and Software Engineering

In lesson 1, we have discussed

- ✓ What is Program ?
- ✓ What is Software ?
- ✓ Why Software Engineering is important ?
- ✓ Challenges in building software products.
- ✓ Some special domain of software:
  - Legacy software
  - WebApps & MobileApps
  - Cloud computing
  - Product line software

Lesson 2 Software and Software Engineering

In these lessons, we will discuss

- ✓ What is the definition of Software Engineering Discipline?
- ✓ What are software engineering realities?
- ✓ What is layered software engineering?
- ✓ What are software activities and elements?
- ✓ General principles of software engineering practice.
- ✓ Some myths of software development.

On Software Engineering

- **Question**

What is the essential difference between programming and software development?

What is the connotation(内涵) and essence of engineering?

What are the similarities and differences between software engineering, civil engineering, and construction engineering?

What are the three elements of software engineering? What issues do each element focus on at the engineering level? What kind of relationship exists between them?

What is the goal of software engineering and what principles does it propose to achieve its objectives ?

What knowledge, abilities, and qualities should excellent software engineers possess?

What is Software Engineering?

- Former definition of Software Engineering
  - The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software. — IEEE
  - An engineering discipline that is concerned with all aspects of software production. — Ian Sommerville
  - The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines. — Fritz Bauer
- **Q**uestion

  What is the definition of Software Engineering discipline ?
  Proposed by Roger S. Pressman?

What is Software Engineering?

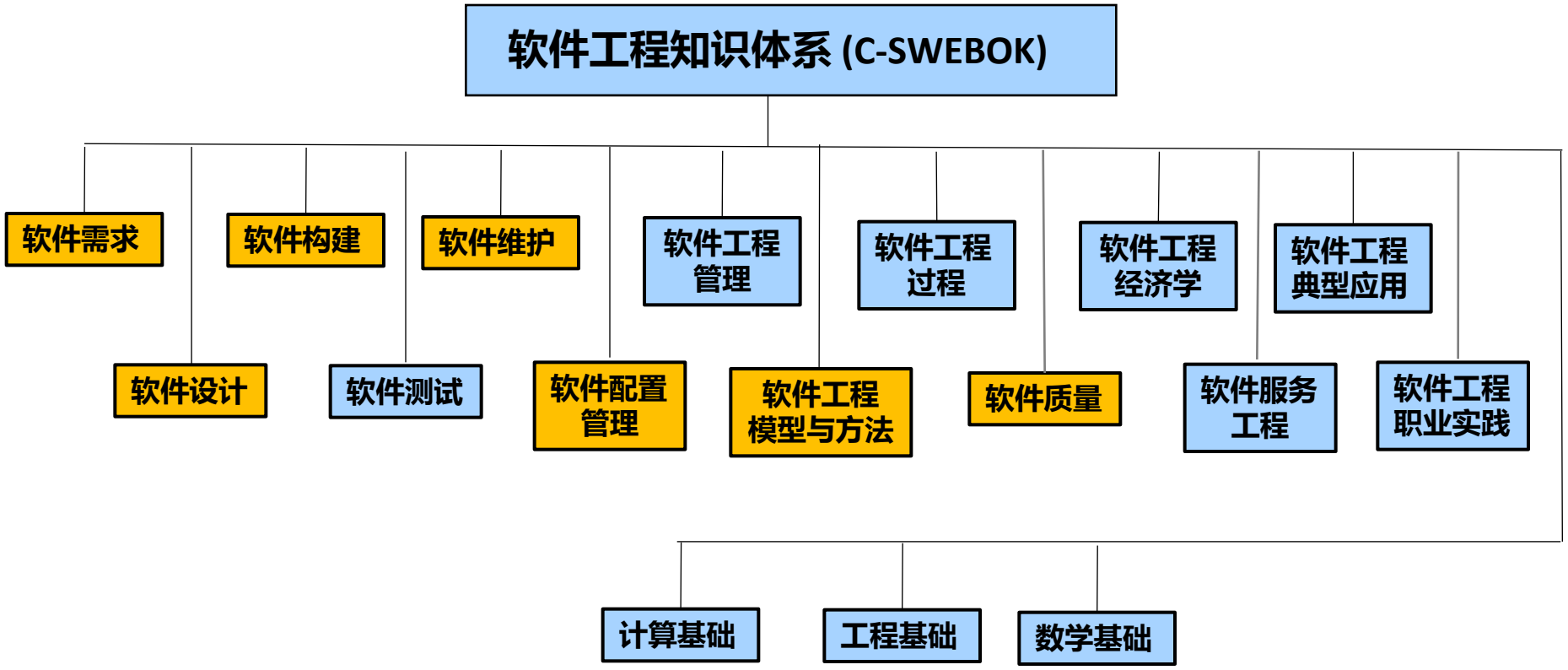
- The IEEE definition
  - (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
  - (2) The study of approaches as in (1).
- Pressman's definition
  - Software engineering encompasses a process, methods for managing and engineering software, and tools.
  - Adaptability and Agility.
  - Software engineering is a layered technology.
  - The bedrock that supports software engineering is a quality focus.

Features of Software Engineering

- Software engineering can be defined as the study & application of engineering to the design, development and maintenance of software product.
- Software engineering discusses systematic and cost-effective techniques to software development. These techniques have resulted from innovations as well as lessons learnt from past mistakes.
- Engineering always connotes(意味着) several things:
  - Designing a product in detail.
  - A methodical approach to carrying out the design and production so that the product is produced with the desired qualities.
  - Trade-off in different alternatives.

Contents of Software Engineering Discipline

中国软件工程知识体系 (C-SWEBOK, 2019)



Software Engineering realities

- Some realities should be recognized
  - A concerted effort(一致努力) 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(影响深远的) definition (**Q:** *by whom*?)
  - [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.

A Layered Structure of Technology

Software Engineering Architecture

A Layered Technology: *Process*

- Any engineering approach, including software engineering, must rest on an organizational commitment to quality.
- Foster a continuous process improvement.
- Bedrock that supports software engineering is a quality focus.
- GOAL:
  - Both quality and maintainability are an outgrowth of good design.

A Layered Technology: *Process*

- The foundation for software engineering is the process layer.
- The software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
- Process defines a framework that must be established for effective delivery of software engineering technology.

A Layered Technology: *Process*

- Process forms the basis for management control of software projects and establishes the content in which
  - technical methods are applied,
  - work products (models, documents, data, reports, forms, etc.) are produced,
  - milestones are established,
  - quality is ensured, and
  - change is properly managed.

A Layered Technology: *Methods & Tools*

- Software engineering methods provide the technical how-to's for building software.
- Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing, and support.
- Methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

A Layered Technology: *Methods & Tools*

- Tools provide automated or semi-automated support for the process and the methods.
- **E**xample
  - When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering (CASE), is established.

A Process Framework

- A process is a collection of *activities*, *actions*, and *tasks* that are performed when some work product is to be created.
- An activity strives to achieve a broad objective (e.g., communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which software engineering is to be applied.
- An action (e.g., architectural design) encompasses a set of tasks that produce a major work product (e.g., an architectural model.

A Process Framework

- A task focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a tangible outcome.
- In the context of software engineering, a process is not a rigid prescription for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work (the software team) to pick and choose the appropriate set of work actions and tasks.
- The intent is always to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Framework Activities

- A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity. In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.
- A generic process framework for software engineering encompasses five activities:
  - Communication
  - Planning
  - Modelling
  - Construction
  - Deployment

Framework Activities

- For many software projects, framework activities are applied iteratively as a project progresses. That is, communication, planning, modeling, construction, and deployment are applied repeatedly through a number of project iterations
- Each iteration produces a software increment that provides stakeholders with a subset of overall software features and functionality.
- As each increment is produced, the software becomes more and more complete.

Stakeholders

Who is involved in a Software Engineering Project

- The concept of a software engineering project classifies all of the individuals involved in the development process as Stakeholders. The examples.

Position	Roll
Client	Individual or organization that has commissioned a software project, and for whom the final product is being made. Also referred to as the customer
Project manager	Responsible for overseeing the entire software engineering project. Ensures that the project remains on task, on time and within budget. Manages team leaders
Team leader	Works under project manager and manages teams that are established to handle portions of the software engineering project. Functions as a project manager for smaller 'sub-projects'
System architect	Designs the software system and determines the way in which various parts of the system interact with each other
Programmer/ developer	Responsible for writing the code called for by the system architect. Transforms system model into actual code. Also responsible for creating and maintaining up to date documentation of the code
Tester	Tests software system by using it in the manner prescribed and desired by the client's requirements and specifications
End user	Consumers of the final software product, as intended by the client

Umbrella Activities

- Software engineering process framework activities are complemented by a number of umbrella activities. In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk.
- Typical umbrella activities include:
  - Software project tracking and control
  - Risk management
  - Software quality assurance
  - Technical reviews
  - Measurement，Measure, Metrics
  - Software configuration management
  - Reusability management
  - Work product preparation and production

Umbrella Activities

- A process adopted for one project might be significantly different than a process adopted for another project Among the differences are
  - Overall flow of activities, actions, and tasks and the interdependencies among them.
  - Manner in which quality assurance activities are applied.
  - Overall degree of detail and rigor with which the process is described
  - ……

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).

The Essence of Practice
- 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(细分)? 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?

Decomposition

- The Decomposition Principle advocates decomposing the problem into many small independent parts. The small parts are then taken up one by one and solved separately.
- The idea is that each small part would be easy to grasp and understand and can be easily solved. The full problem is solved when all the parts are solved.
- To aid in this process, the software engineering project in divided into phases that are optimized to achieve unified project progression.
- **GOAL**: The compartmentalization breaks a large project down into more manageable sections, and it can allow for a more effective utilization of resources.

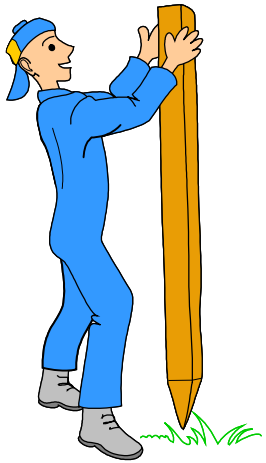
The Essence of Practice
- Plan the Solution

- 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?

The Essence of Practice
- Plan the Solution

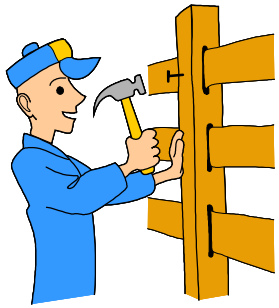
- In order to efficiently and effectively direct this conglomeration (混合物) of goals and resources, it is crucial to establish comprehensive plans of action.
- The creation and use of plans is an activity which occurs throughout the entire development process.
- Plans are continuously analyzed, evaluated and updated to meet changes encountered, such as updated requirements or specifications from the client or an alteration in the system design from the system architect.
- Planning is also invaluable in cost estimation and time-requirements computation, as it sets a schedule from which to work.

Example: Scheduling Fence Construction Tasks

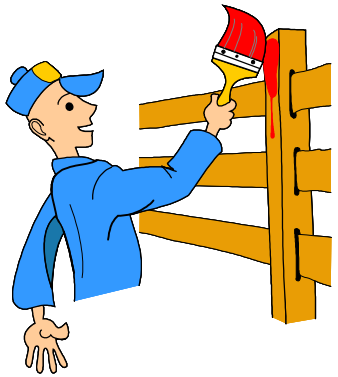


Setting posts
[3 time units]

Cutting wood
[2 time units]



Nailing
[2 time units for unpainted;
3 time units otherwise]



Painting
[5 time units for uncut wood;
4 time units otherwise]

Setting posts $<$ Nailing, Painting

Cutting $<$ Nailing

…shortest possible completion time = ?

[$\Rightarrow$ “simple” problem, but hard to solve without a pen and paper]

The Essence of Practice
- Carry Out the Plan

- Does the solution conform(符合) 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 ?

The Essence of Practice
- Carry Out the Plan

- Problem Solving
  - Software engineering is, in essence, a form of problem solving.

Solution A

- A client might request some intuitive(直观的) application for text entry, storage, reading and editing.
- Solution: a word processor.

The Essence of Practice
- Carry Out the Plan

- Problem Solving
  - Software engineering is, in essence, a form of problem solving.

Solution B

- A client could be seeking a way to market and sell their merchandise(商品) throughout the country without building brick & mortar stores in each province.
- Solution: an online store that provides a way to view the products, a secure method of payment, and a proper tracking for product orders.

The Essence of Practice
- Carry Out the Plan

- Problem Solving

Solution C

- A client might be a videogame firm, looking to take their single-player videogame into the realm(领域) of multiplayer-gaming.
- Solution I: a peer-to-peer based solution

One of the players in a multiplayer session takes on the role of the host, and allows his/her computer to act as a server, collecting, computing and transmitting data to the other players.

- Solution II: a client-server style solution

All of the players in a session connect to a dedicated server whose sole purpose is to handle the computations and client communications for the game.

The Essence of Practice
- Carry Out the Plan

- Problem Solving
  - Software engineers use a set of paradigms to work their way through the phases of the software development process.
  - These phases will be discussed in detail in the next chapters.
  - It is sufficient to say, a software engineer must analyze the initial problem, develop a list of possible solutions and evaluate that list to select the best solution.
  - The software engineer must then design a system to reflect that solution and implement that design into a software product.

The Essence of Practice
- Carry Out the Plan

- The software engineering project is an undertaking that consumes resources in order to produce a software product. The end product described here includes not only the program itself, but all of the documentation, distributions and packaging associated with that program.
- Resources: Time, Money, and People.
- The concept of the software engineering project puts the principals behind software engineering into practice to ensure an efficient and successful development process.

The Essence of Practice
- Carry Out the Plan

- Modeling
  - Between the identification of the problem and the delivery of the solution, there lies a series of steps that must be worked through.
  - One underlying reason for this step-by-step strategy is related to the difficulty of moving directly from a general concept of the solution to the code that makes up the software itself.
  - It is ill advised to write a paper without first creating an outline, it is simply bad practice to code from scratch without a well-defined design to use as a roadmap.

The Essence of Practice
- Carry Out the Plan

- Schematic representation

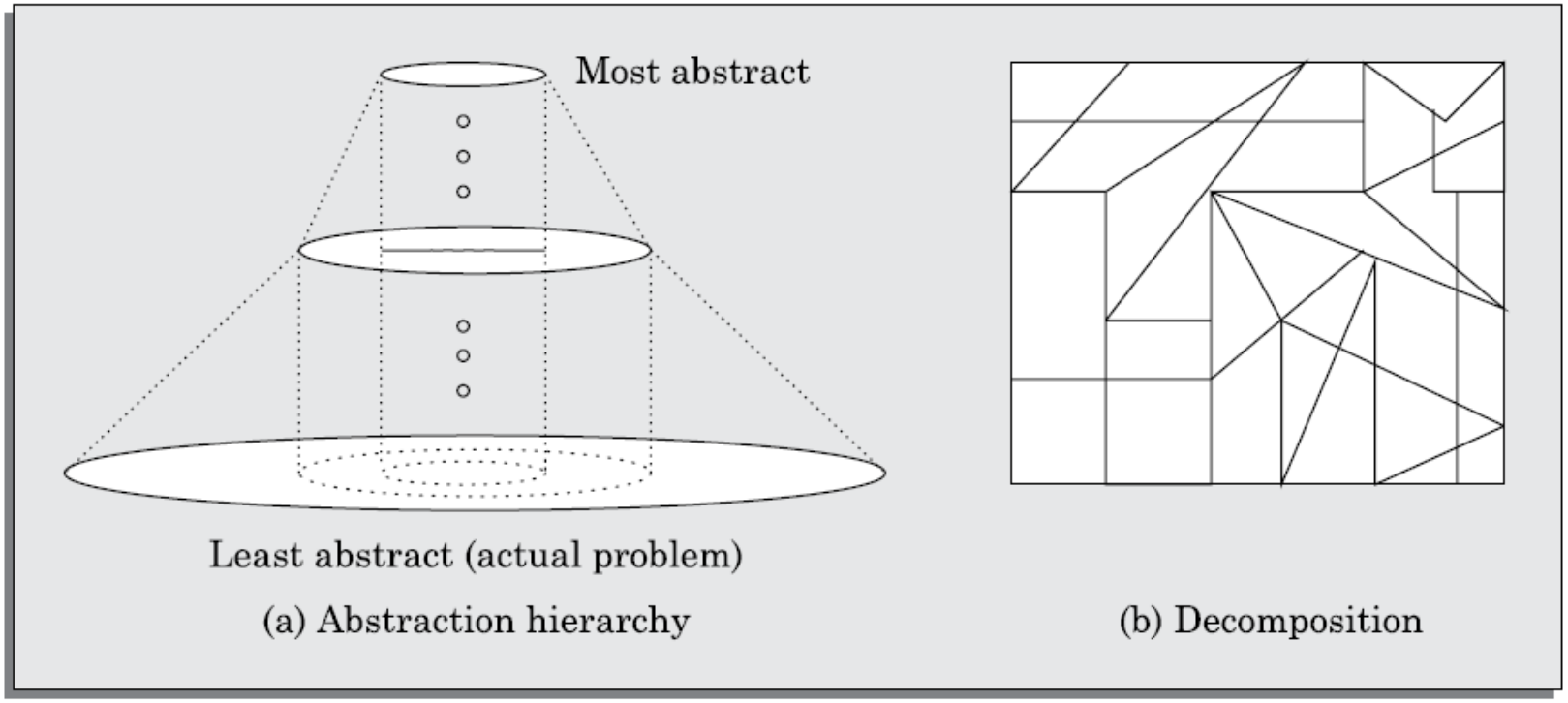
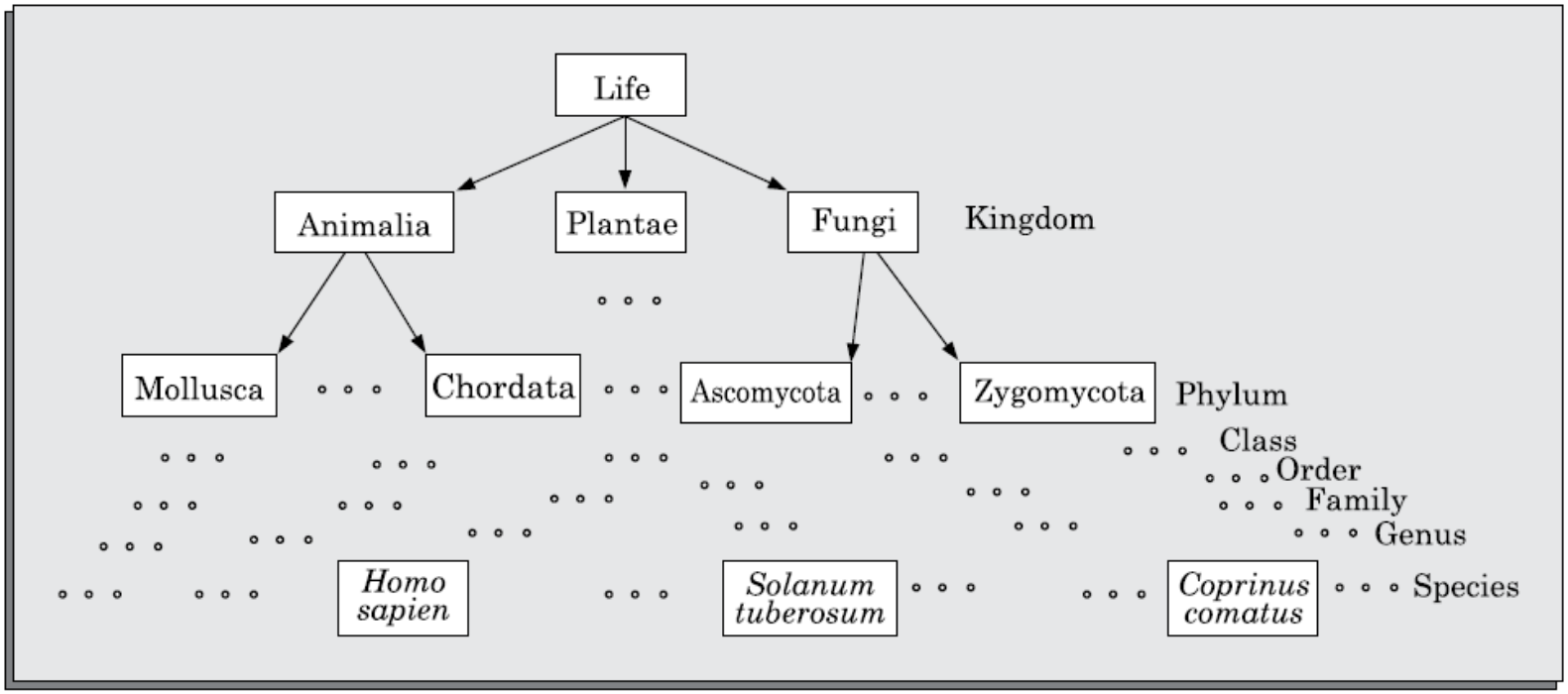


Fig. 2-1 Schematic representation

The Essence of Practice
- Carry Out the Plan

- Abstraction

Simplifying a problem by omitting unnecessary details is known as the principle of abstraction. (界、门、纲、目、科、属、种)



The Essence of Practice
- Carry Out the Plan

- Modeling
  - Software engineers use abstraction to describe the intricacies(复杂) of a software system in an understandable way.
  - Then they translate these concepts into visual representations by engaging in modeling.
  - Modeling is not limited to just the software product itself, models also can be used for the initial client problems and the system's operating environment as well.
  - UML provides a common framework which is intuitive to use and can be shared among different engineers.

The Essence of Practice
- Carry Out the Plan

- Communication
  - Without a means for communication, it would be nearly impossible for the successful completion of a software development project.
  - Example
    - The client must communicate to the project manager exactly what he or she wants.
    - The project manager must communicate with the team leaders to properly explain team rolls.
    - The developers must communicate with upper management in order to fully understand what the client wants.
  - To properly monitor progress, participants might communicate and report their status to team leaders or the project leader during regular meetings.

The Essence of Practice
- Carry Out the Plan

- Using and managing resources
  - The materials dedicated to software engineering are nearly boundless.
  - A resource is anything that will be utilized or consumed during a software engineering project.
  - Common resources include time and money, also include all the people working on the project and the machines on which the project is developed.
  - Improper allocation and management of resources can lead to the failure of a project, while appropriate use can help make the software development process a success.

The Essence of Practice
- 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?

The Essence of Practice
- Examine the Result

- Reaching milestones and producing deliverables
  - A software engineering project begins with the elicitation of requirements from the client and ends with the delivery of a software product.
  - As a means of maintaining communication to ensure adequate progress the client and the software engineers agree on milestones to be met along the way. Milestones serve as benchmarks, and allow the client to evaluate the work being done.

The Essence of Practice
- Examine the Result

- Reaching milestones and producing deliverables
  - A software engineering project is continually producing deliverables during the process. A deliverable is any artifact created during the software development process, such as documentation, files, or the code itself.
  - The deliverables and milestones are a form of communication between the client and a software engineering development team.

The Essence of Practice
- Examine the Result

- Maintaining a product
  - A firm's success is not based on the production of a perfect product, it is based on effective product maintenance.
  - The process of maintenance of a software product involves three basic steps:
    - First, the existing product is cleaned, meaning that the bugs are fixed.
    - Second, the product is stabilized to ensure that the updated product functions as intended.
    - Third, any enhancements or additions are added to the product to adjust functionality or improve performance as required by the client.
  - These steps are repeated throughout the life of the software product.

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

- Management myths.
- Customer myths.
- Practitioner's myths.
- **E**xample

If we get behind schedule, we can add more programmers and catch up.

Adapted from 《The Mythical Man-Month》

by Frederick P. Brooks.

Summary

Software is the key element in the evolution of computer-based systems and products and one of the most important technologies on the world stage.

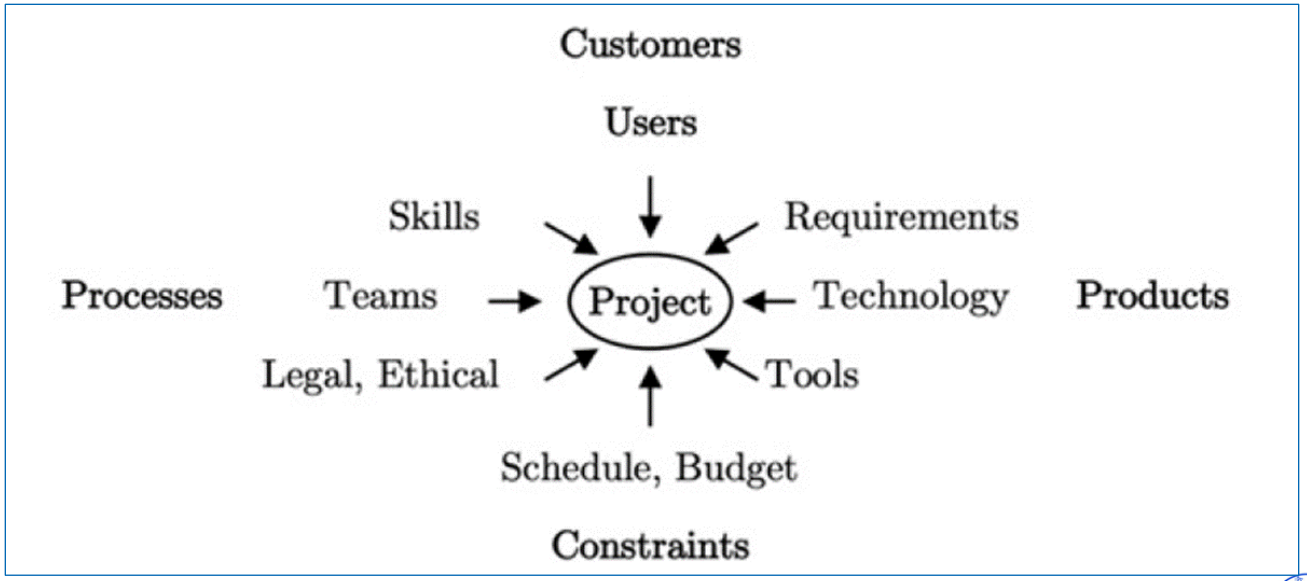
Over the past 60 years, software has evolved from a specialized problem-solving and information analysis tool to an industry in itself. Yet we still have trouble developing high-quality software on time and within budget.

Software - programs, data, and descriptive information - addresses a wide array of technology and application areas. Legacy software continues to present special challenges to those who must maintain it.

Summary

Software engineering encompasses process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality.

The software process incorporates five framework activities - communication, planning, modeling, construction and deployment - that are applicable to all software projects.



Assignments

Proposal of Course Project

After completing the team building, please discuss and determine the course project for your team. After clarifying the course project of your team, please fill out and submit the project proposal according to the template (on Canvas).

Constraints:

1. The course project should be an application development project. If it is an algorithm project of your mentor, the application scenario of this algorithm should be searched for, and based on this scenario, application development should be carried out.

2. The quantity of source codes for each course project (excluding automatically generated codes) should not be less than 5000 lines.

3. The proposal should be submitted by the team leader, before **23:59** on **12th** October 2025 by replying to the assignment on canvas.

4. The naming rule of the proposal: Teamleader std's ID + His/Her Name + Project Name.

5. If a group is unable to find a suitable course project, they can choose a candidate project recommended by the teaching teacher and use it as a course project to carry out research and development work. The candidate projects will be released on the Canvas platform on **29th** September 2025.

Assignments

- Chapter 1. Problems to ponder
  - 1.4 According to Pressman

    Many modern applications change frequently—before they are presented to the end user and then after the first version has been put into use. Suggest a few ways to build software to stop deterioration due to change.

- Questions
  - Read Case studies from 课外阅读资料

    Chapter 1 of Engineering Software Products

    by Ian Sommerville, Page31～Page38.
  - All of these cases would be discussed in the next lesson.

Assignments

- Reading

  The Mythical Man-Month Essay on Software Engineering

  written by Frederick P.Brooks

  Chapter 2. The Mythical Man-Month

- Preview

  Software Engineering (8th edition) written by Roger.S.Pressman et al.

  Chapter 3. Software Process Structure

  Chapter 4. Process Models

  Chapter 5. Agile Development

huangjie@tongji.edu.cn