软件工程基础

刘钦

Outline

- Software
- Software Engineering
- Knowledge Area
- How to learn SE

What is software?

- Programming Language?
- Program?
- Contrast to hardware?
- ...

Information

- 信息的记录
- 信息的交流
- 信息的存储
- Information => Computing => Digital Computer => Software

Foundation of Digital Computing - 30s

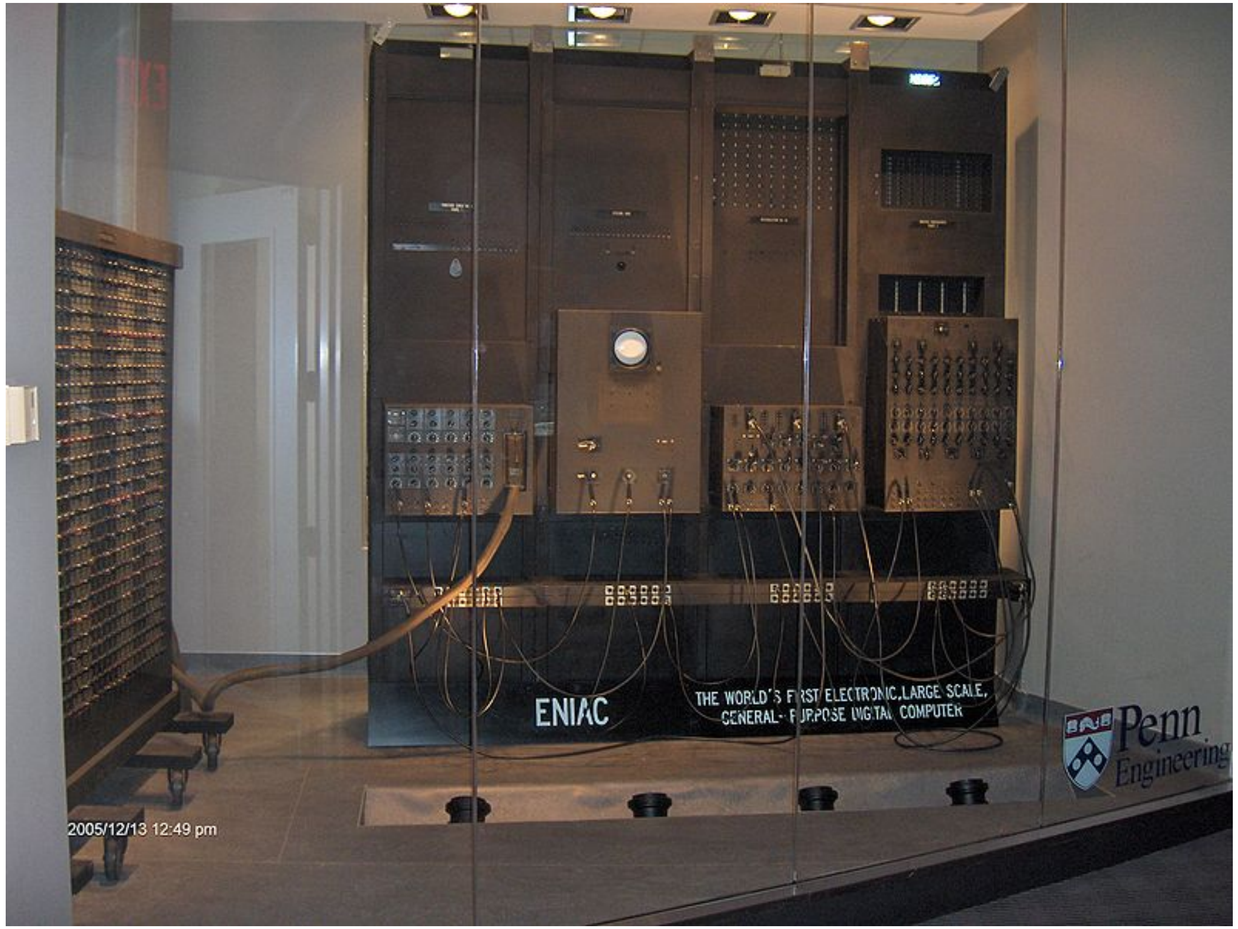
Hardware

- 微分计算器 Vannervar Bush
- 继电器式计算机 Konard Zuse
- ABC原型计算机 John Vincent Atanasoff & Clifford Berry
- Harvard Mark 1 IBM实验室

Theory

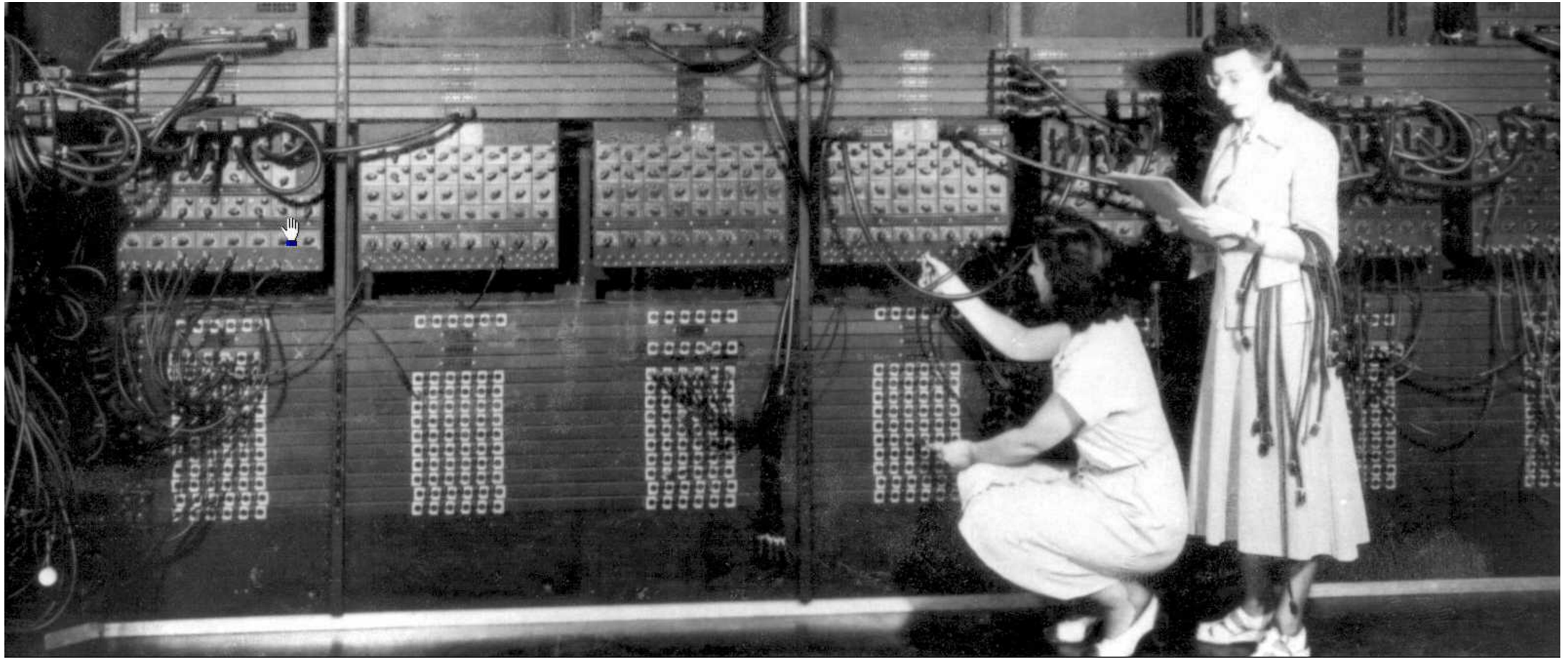
- 《论可计算数及其在判定问题上的应用》Alan Turing
- 电子继电器可以实现布尔符号逻辑　Claude Elwood Shannon

Digital Computers - 40s



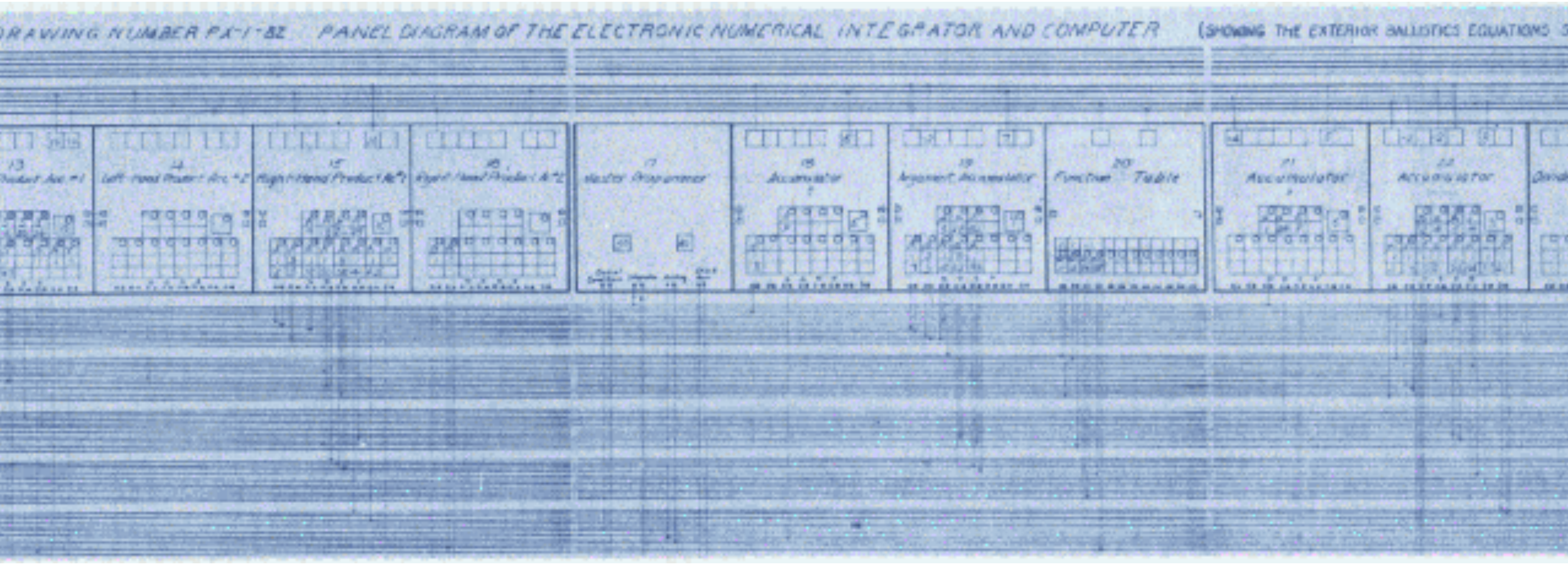
ENIAC

Four ENIAC panels and one of its three function tables, on display at the School of Engineering and Applied Science at the University of Pennsylvania



Programming the ENIAC

Two women set the switches on one of ENIAC's function tables at the Moore School of Electrical Engineering.



Programming chart

ENIAC programming chart representing the wiring to set up an exterior ballistics equation

Steps of programming

- The task of taking a problem and mapping it onto the machine was complex, and usually took weeks.
- After the program was figured out on paper, the process of getting the program "into" ENIAC by manipulating its switches and cables took additional days.
- This was followed by a period of verification and debugging, aided by the ability to "single step" the machine.

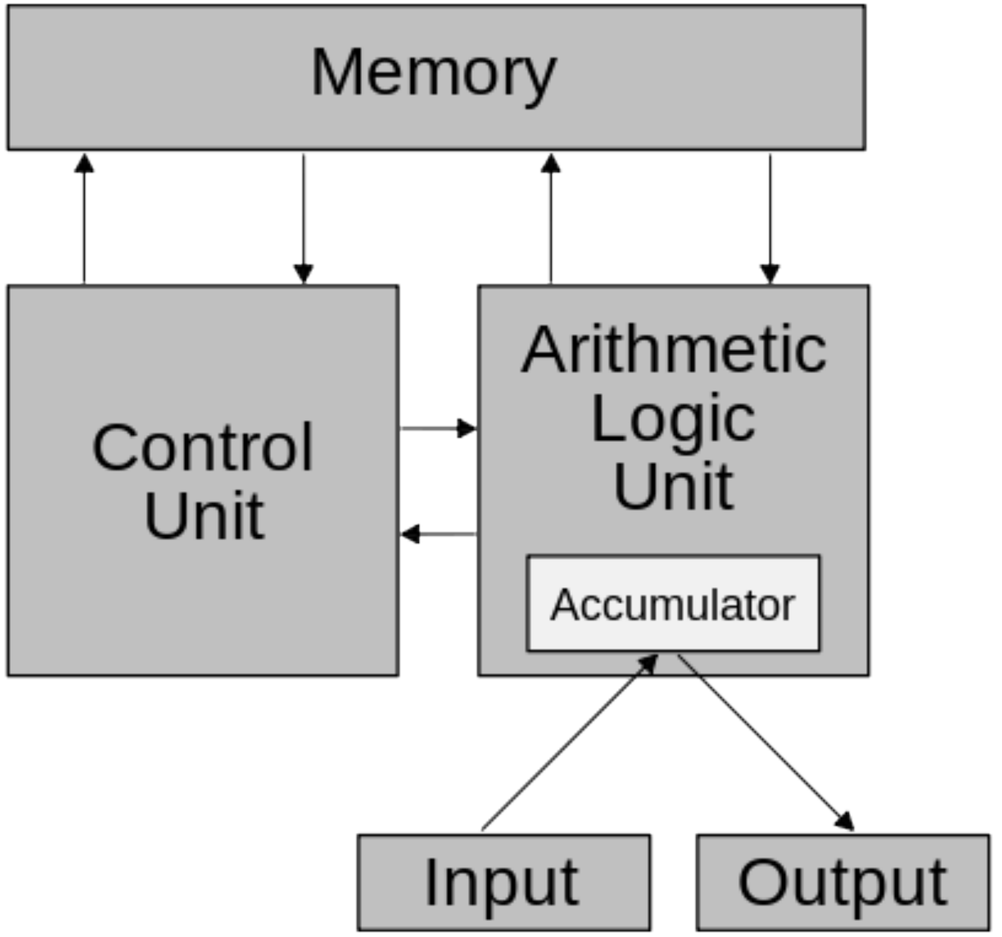
EDVAC

- EDVAC (Electronic Discrete Variable Automatic Computer) was one of the earliest electronic computers. Unlike its predecessor the ENIAC, it was binary rather than decimal, and was a stored program computer.

- Eckert and Mauchly and the other ENIAC designers were joined by John von Neumann in a consulting role; von Neumann summarized and discussed logical design developments in the 1945 First Draft of a Report on the EDVAC.

Von Neumann architecture

- The stored-program concept



Software is one part of Hardware - 50s

Commercial Computer

- Ferranti Mark I
- UNIVAC I
- LEO I
- IBM 701
- IBM 650

Programming Language

- 1955 FORTRAN
- 1958 LISP
- 1959 COBAL

Naming

- In a 1958 article in American Mathematical Monthly, John W. Tukey became the first person to define the programs on which electronic calculators ran.

Software is not Hardware - 60s

变革

- ASCII美国信息交换标准码
- ATM
- IBM信息管理系统IMS应用于阿波罗航天器
- 软件咨询业务
- IBM S/360
- 信用卡
- DEC PDP-1小型机

软件的特性

- 软件与现实世界关系更加密切，对需求的规格化更加困难
- 软件比硬件容易修改的多，并且不需要昂贵的生产线复制产品
- 软件没有损耗
- 软件不可见

Program = Algorithm + Data Structure - 70s ~ 80s

Program

- Algorithms + Data Structures = Programs is a 1976 book written by Niklaus Wirth covering some of the fundamental topics of computer programming, particularly that algorithms and data structures are inherently related.

- For example, if one has a sorted list one will use a search algorithm optimal for sorted lists.

Software Development is much more complicated than Programming
- 90s ~ now

软件工作量随应用程序规模变化的情况

规模（功能点）	规模（KLOC）	编码（%）	文档编写（%）	缺陷清除（%）	管理和支持（%）
1	0.1	70	5	15	10
10	1	65	7	17	11
100	10	54	15	20	11
1000	100	30	26	30	14
10, 000	1000	18	31	35	16

- 源自[Jones1996]

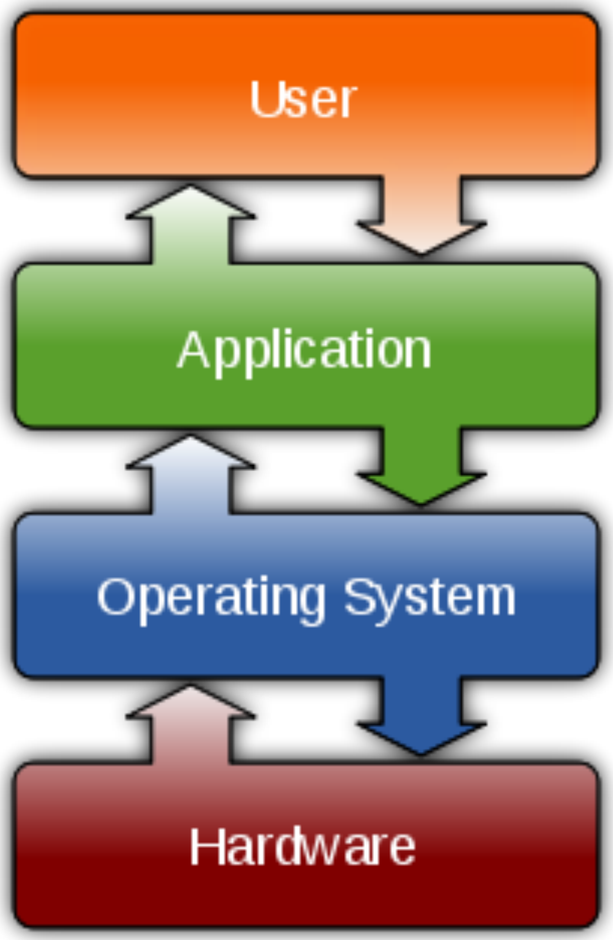
大型软件系统（规模为100万LOC）的成本因素排序

排名	工作内容
1	缺陷清除（审查、测试、发现并修复Bug）
2	纸质文档编写（计划、规格说明、用户手册）
3	会议和交流（顾客、团队成员、经理）
4	编程或编码
5	项目管理
6	变更控制

- 源自[Jones2007]

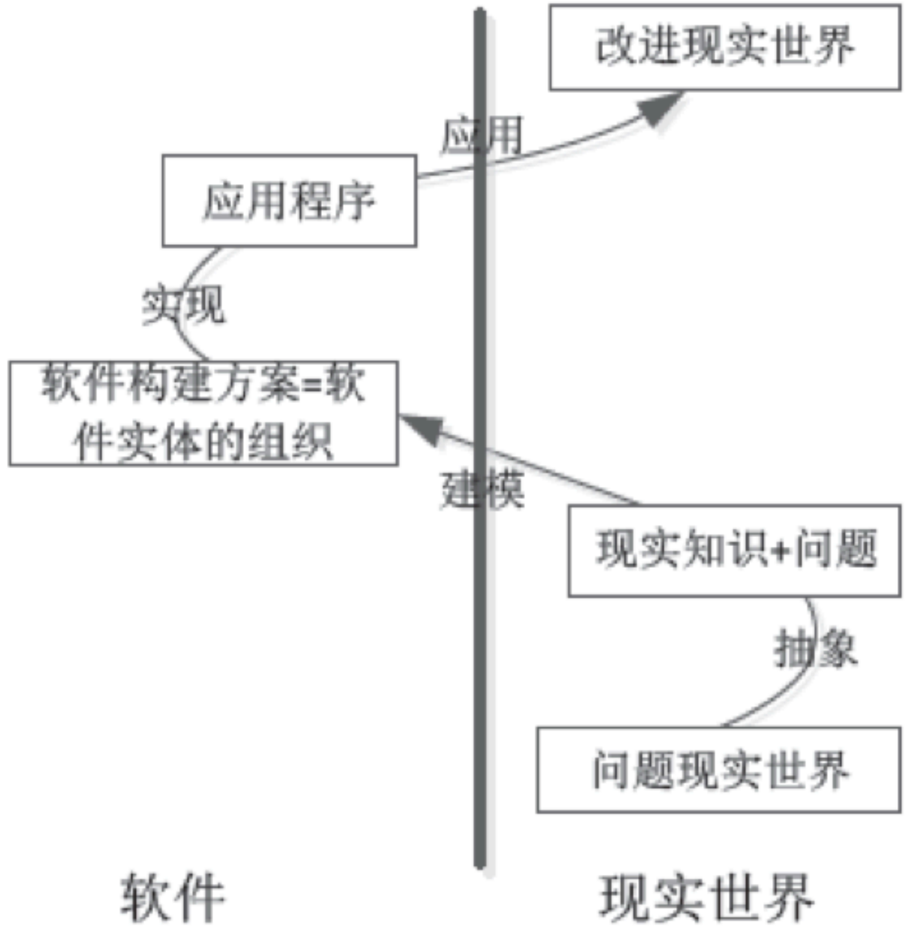
Types of software

- System software
  - 操作系统、数据库、数据仓库、嵌入式设备、安全
- Programming software
  - IDE、测试、持续集成、建模、度量
- Application software
  - 商业、政府、休闲、医疗、教育、国防、个人、专业、科学



应用软件基于现实又高于现实

- 应用软件被开发的目的和意图来源于现实世界的问题。
- 应用软件必须基于现实才能解决问题。
- 软件最终要被用于现实并改进现实。



What is software? - Summary

- Software is independent of hardware
- Software is a tool
- Software = programs + documents + data + knowledge
- Software development is much more complicated than programming
- Application software originate from the reality, and reversely improve the reality

What is software engineering?

Definition

- [IEEE610.12-1990]:
  - (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)

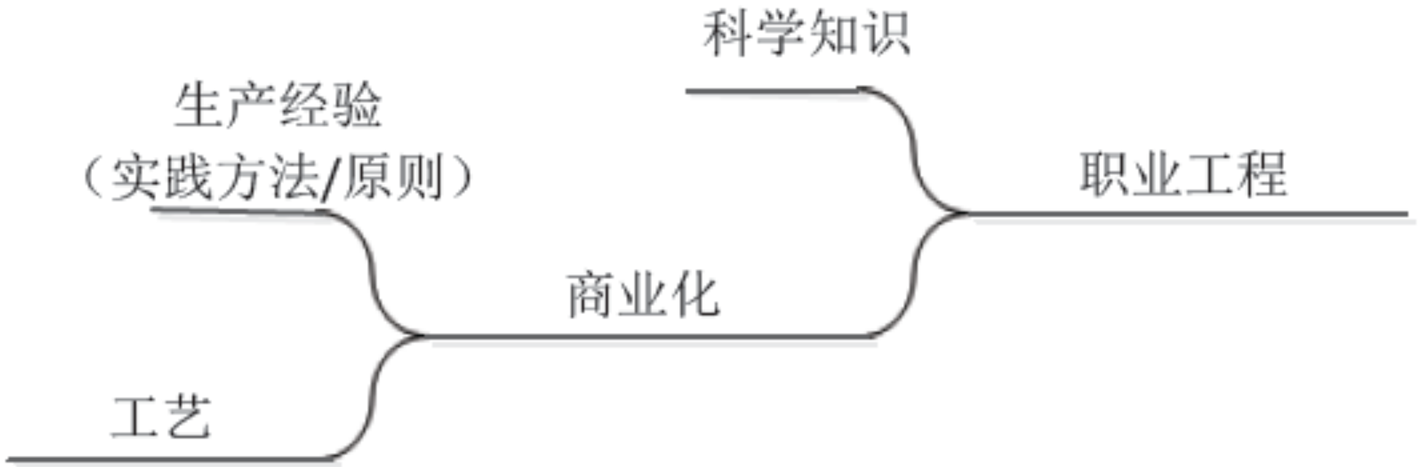
Engineering

- The American Engineers' Council for Professional Development (ECPD, the predecessor of ABET) has defined "engineering" as:

  - The creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation or safety to life and property.

CCSE - About Engineer

- 工程师通过一系列的讨论决策，仔细评估项目的可选活动，并在每个决策点选择一种在当前环境中适合当前任务的方法进行工作。可以通过对成本和收益进行折衷分析调整相应策略。
- 工程师需要对某些对象进行度量，有时需要定量的工作；他们要校准和确认度量方法，并根据经验和实验数据进行估算。
- 软件工程师强调项目设计过程的纪律性，这是团队高效工作的条件。
- 工程师可胜任研究、开发、设计、生产、测试、构造、操作、管理，以及销售、咨询和培训等多种角色。
- 工程师们需要在某些过程中使用工具，选择和使用合适的工具是工程的关键要素。
- 工程师们通过专业协会发展和确认原理、标准和最佳实践方法，并提高个人能力。
- 工程师们能够重用设计和设计制品。



有天赋的业余爱好者
依靠直觉和非理性能力
偶然的进展
无法培训人员
不计成本的使用资源
为自用而制造

熟练工人
依靠程序规范
实用主义改进
经验传授
成本关注
为销售而制造

受过专业教育的职业人员
依靠理论与分析
随科学进展而进步
专业化的职业教育
成本控制
科学知识带来新的应用
市场细分、产品多样化

工程学科的发展

M. Shaw, Prospects for an Engineering Discipline of Software

- 具有解决实际问题的动机：
  - 工程学解决实际问题，而这些问题来源于工程领域之外的人——消费者。
- 应用科学知识指导工程活动：
  - 工程学不依赖于个人的技能，而是强调以科学知识为指导，按照特定方法与技术，进行规律性的设计、分析等活动，实现工程活动的可学习性和可重复性。
- 以成本效益比有效为基本条件：
  - 工程学不单单只是解决问题，它要有效利用所有资源，至少成本要低于效益，即成本效益比有效。
- 构建机器或事物：
  - 工程学强调构建实物工具，例如机器、事物等，并利用实物工具来解决问题。
- 以服务人类为最终目的：
  - 工程学考虑的不是单个客户的需要，而是要运用技术和经验实现全社会的进步。

Factor of Engineering

- Problem: Motivation
- Scientific Knowledge: Instrument
- Solution/Machine: Objective
- Cost-Benefit Effective: Condition

Software Engineering’s Problem

- Real world chaos
    - If the state of the real world do not accord with our expectations , then there is a problem.
- Broad scope
    - All fields that mankind set foot in
    - In others engineering, problems are limited
- Mostly not clear, needed to be discovered
    - In others engineering, problems are specific and clear
- Engineering Idea
    - Objectives (Know what you want to)

Scientific Knowledge

- Computing science as the scientific basis
  - Formal techniques
    - Based on mathematics and logic
- Many aspects have been made systematic
  - Methods/methodologies/techniques
  - Languages
  - Tools
  - Processes

Engineering = Science + Principle + Art

- Science--Computing science knowledge is basis of SE
  - Features of the computer
  - Software entities (program languages)
  - Methods/methodologies/techniques
  - …
- Principle--Practice knowledge should also be effective shared
  - Methods/techniques
  - Problems/solutions
  - …
- Art
  - Creativity: Analysis, Design

Solution/Machine && Product

- Virtual Machine=General machine + Specific solution
- Solution can be mapped into the general machine with the help of program languages
- Not different machines for different problems, like others engineering
- Development of solution is the essential task of SE
- Mapping solution into general machine is accidental task of SE

Cost-Benefit Effective

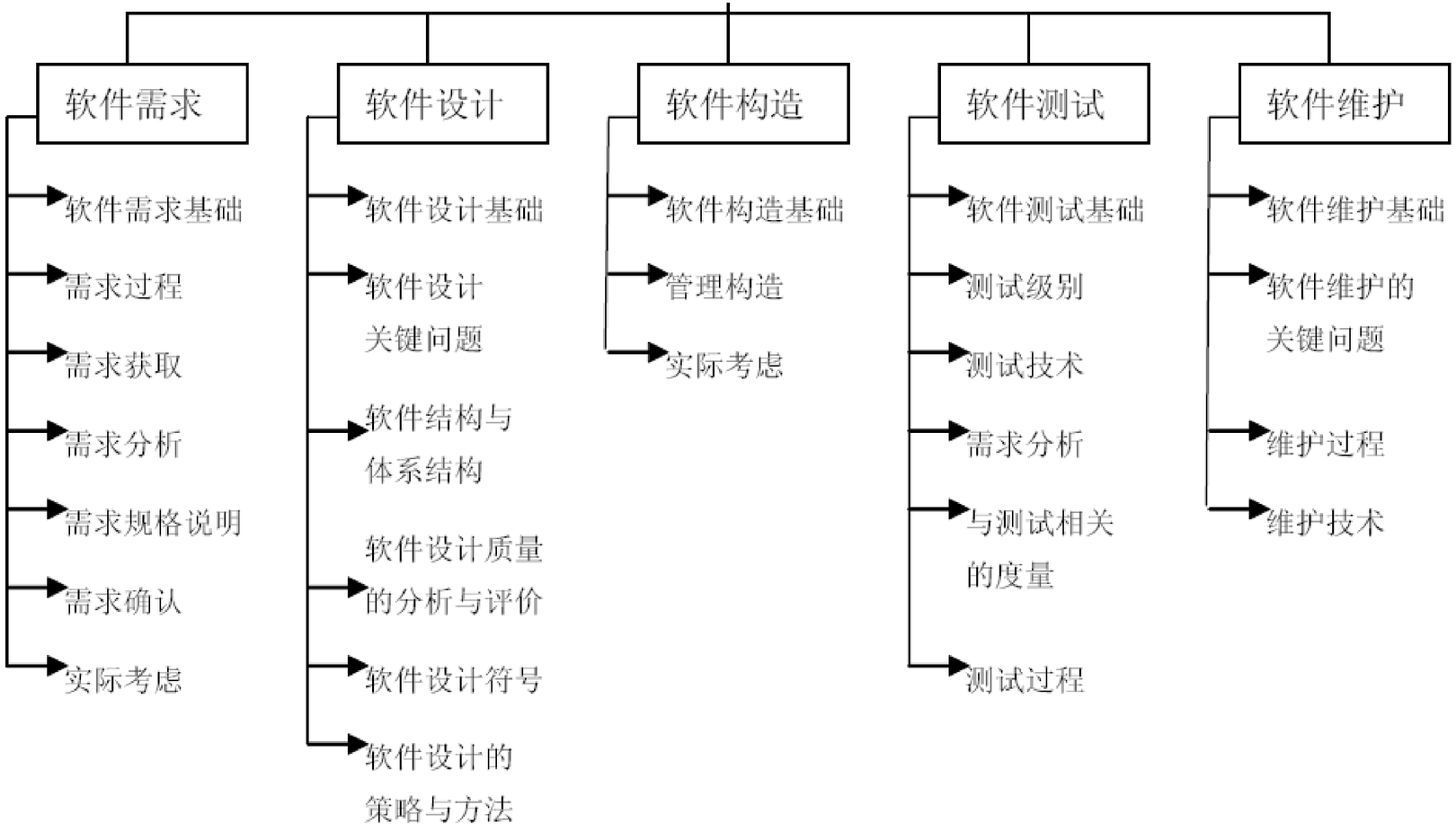
- Feasibility study
    - Benefit is up to clients
    - Cost is up to software engineer
- There are always many ways to a destination, software engineers should choose the most cost-benefit effective one (not the most advanced one).
- When cost-benefit is inessential, then SE may disappears
    - Cost is small
        - Small program
    - Benefit is implicit
        - Exploration study
    - Cost is inessential
        - Computer science study
- Engineering Idea
    - Cost Benefit Effective
- Monitor and Control

Understanding of Software Engineering

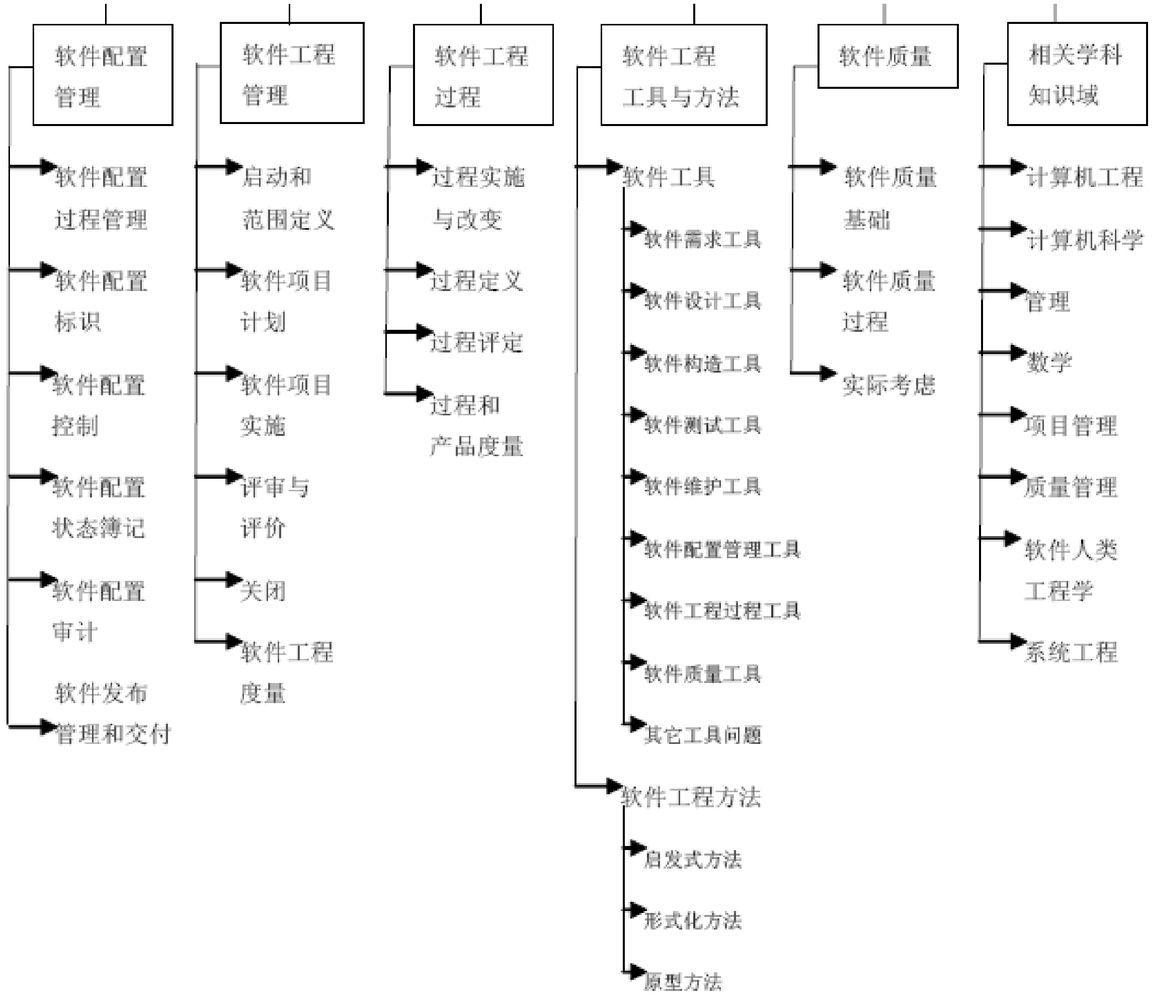
- 软件工程是一种工程活动
- 软件工程的动机是解决实际问题
- 软件工程是科学性、实践性和工艺性并重的
- 软件工程追求足够好，不是最好
- 软件工程真正的产品是基于虚拟计算机的软件方案
- 软件工程的最终目的是要促进整个社会的进步

Knowledge Area

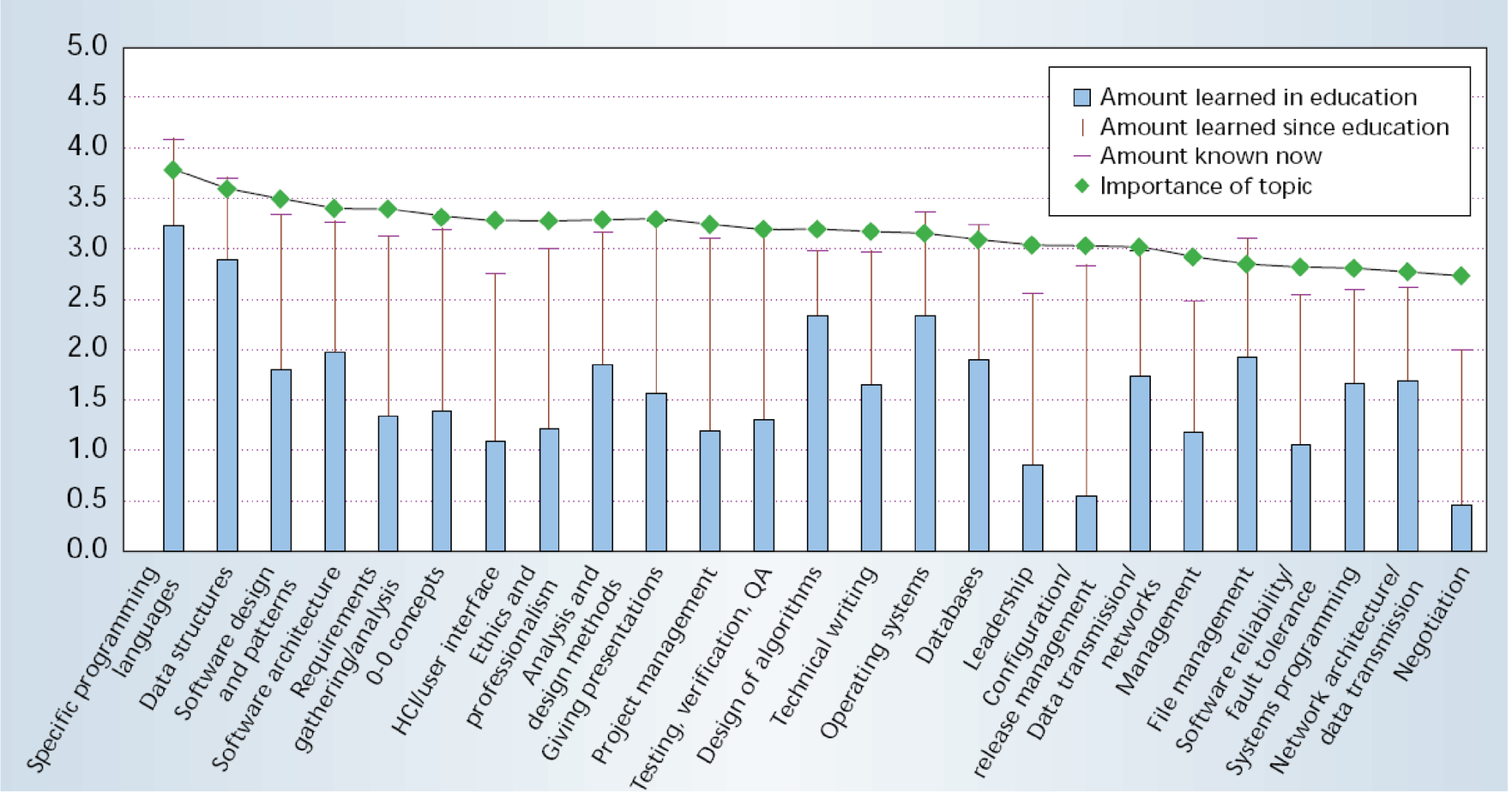
- SWEBOK
  - The Software Engineering Body of Knowledge (SWEBOK) is a product of the Software Engineering Coordinating Committee sponsored by the IEEE Computer Society



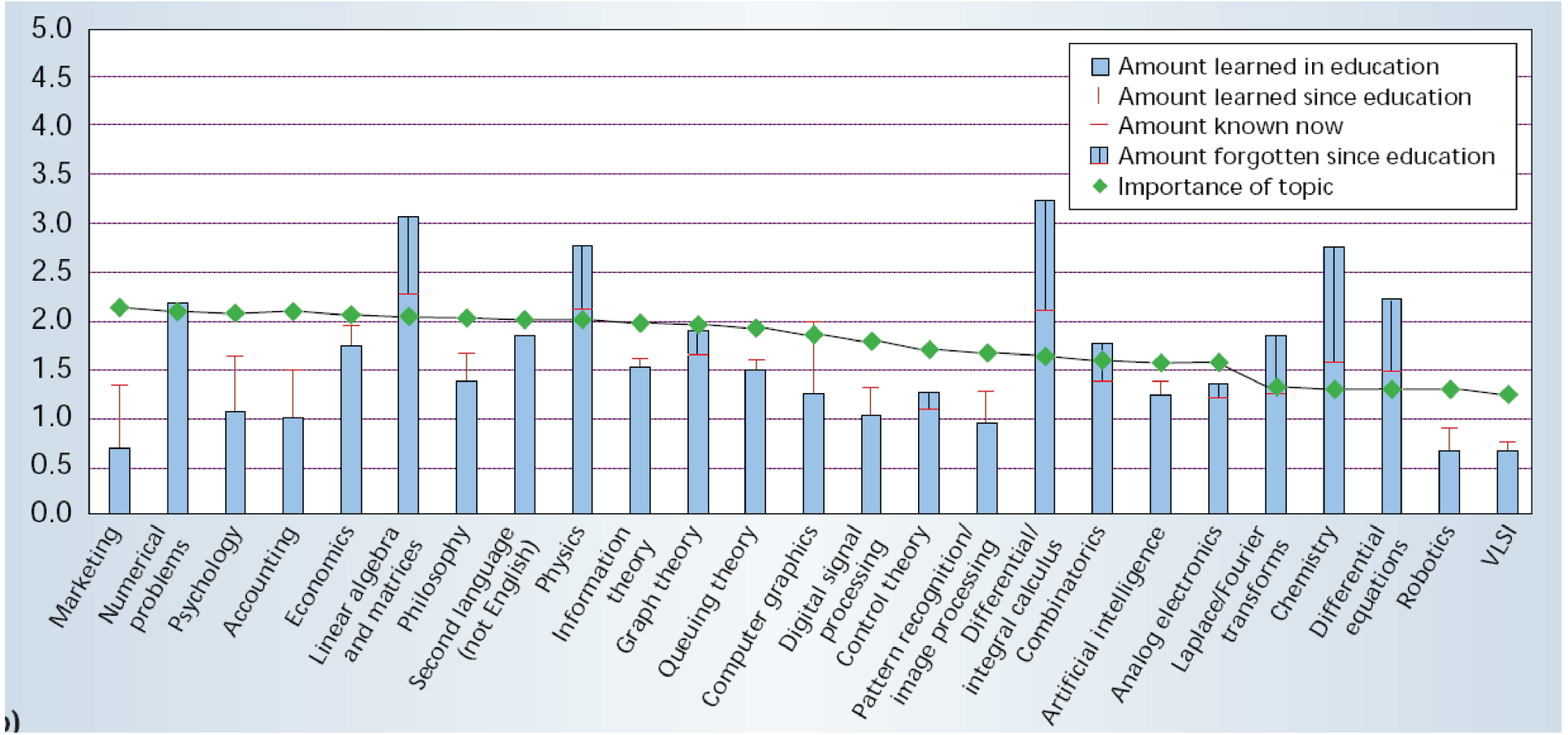
SWEBOK软件工程技术知识域



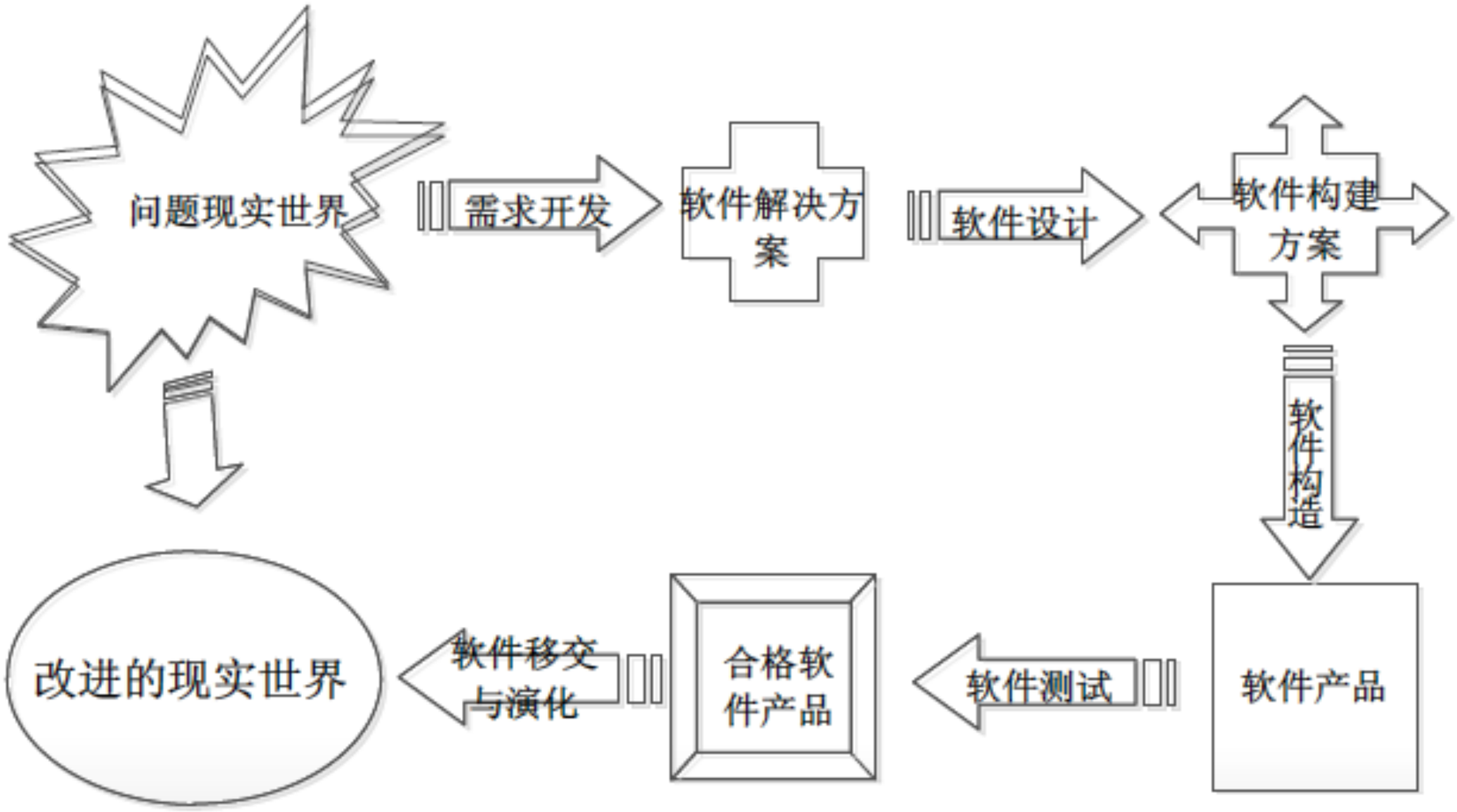
SWEBOK软件工程管理知识域



职业软件工程师的软件工程知识重要性评价 I



职业软件工程师的软件工程知识重要性评价 II


问题现实世界
需求开发
软件解决方案
软件设计
软件构建方案
软件构造
软件产品
软件测试
合格软件产品
软件移交与演化
改进的现实世界

软件开发活动

Software Development Activity

	Target	Artifact
Software Requirement	What	SRS
Software Design	How	SDD
Software Construction	Build	Code and executable file
Software Testing	Are you building the “right” thing? Are you building it “right”?	Test Report
Software Deliver	Install	User Document and System Document
Software Maintenance	Revolution	New version software

角色分工

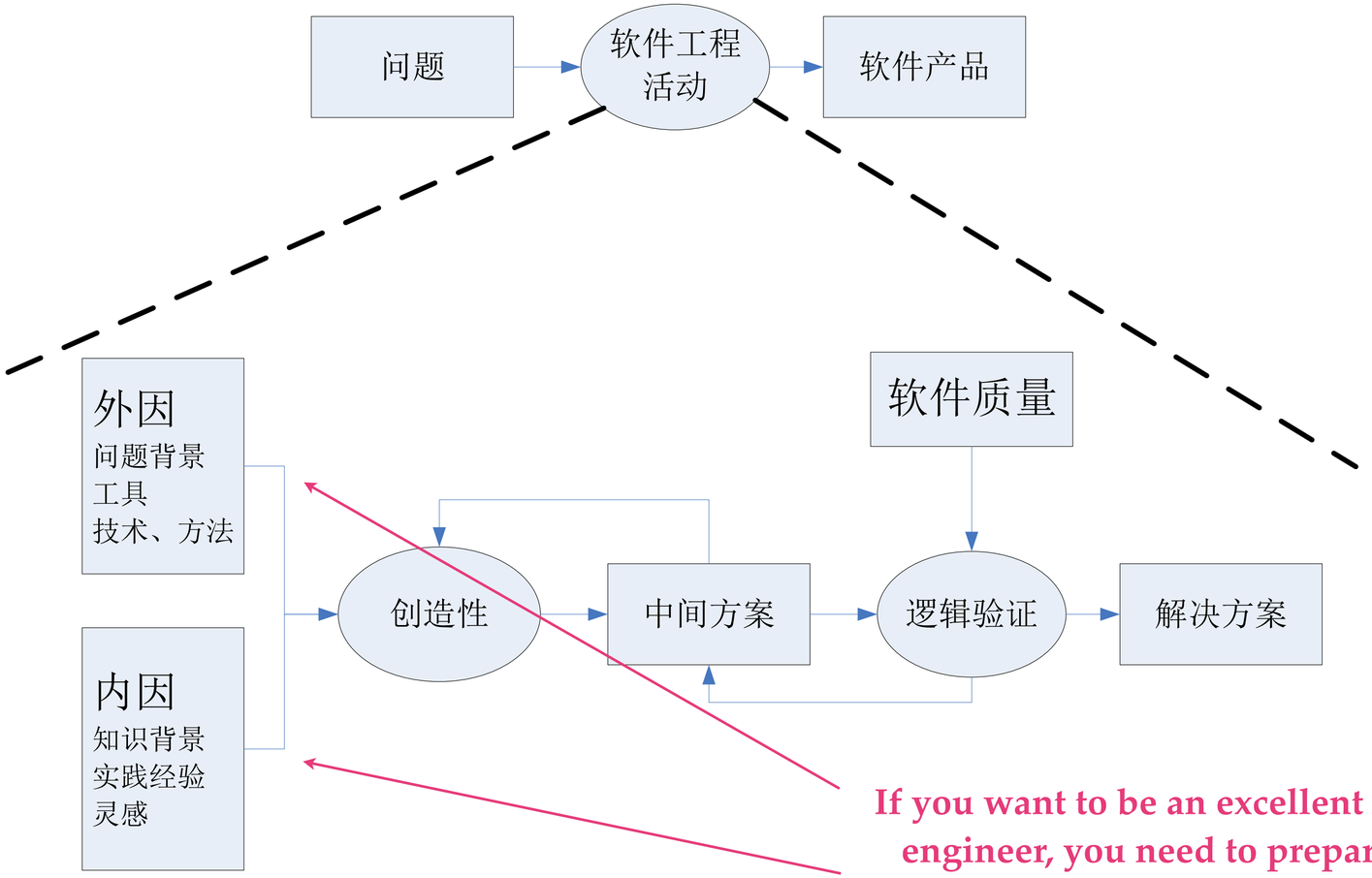
- 需求工程师，又被称为需求分析师：
  - 承担需求开发任务。软件产品的需求开发工作通常由多个需求工程师来完成，他们共同组成一个需求工程师小组，在首席需求工程师的领导下开展工作。通常一个团队只有一个需求工程师小组。
- 软件体系结构师：
  - 承担软件体系结构设计任务。通常也是由多人组成一个小组，并在首席软件体系结构师的领导下开展工作。通常一个团队只有一个软件体系结构师小组。
- 软件设计师：
  - 承担详细设计任务。在软件体系结构设计完成之后，可以将其部件分配给不同的开发小组。开发小组中负责所分配部件详细设计工作的人员就是软件设计师。一个团队可能有一个或多个开发小组。一个小组可能有一个或多个软件设计师。

角色分工

- 程序员：
  - 承担软件构造任务。程序员与软件设计师通常是同一批人，也是根据其所分配到的任务开展工作。
- 人机交互设计师：
  - 承担人机交互设计任务。人机交互设计师与软件设计师可以是同一批人，也可以是不同人员。在有多个小组的软件工程团队中，可以有一个单独的人机交互设计师小组，也可以将人机交互设计师分配到各个小组。
- 软件测试人员：
  - 承担软件测试任务。软件测试人员通常需要独立于其他的开发人员角色。一个团队可能有一个或多个测试小组。一个小组可能有一个或多个软件测试人员。

角色分工

- 项目管理人员：
  - 负责计划、组织、领导、协调和控制软件开发的各项工作。相比于传统意义上的管理者，他们不完全是监控者和控制者，更多得是协调者。通常一个团队只有一个项目管理人员。
- 软件配置管理人员：
  - 管理软件开发中产生的各种制品，具体工作是对重要制品进行标识、变更控制、状态报告等。通常一个团队只有一个软件配置管理人员。
- 质量保障人员：
  - 在生产过程中监督和控制软件产品质量的人员。通常一个团队有一个质量保障小组，由一个或多个人员组成。
- 培训和支持人员：
  - 负责软件移交与维护任务。他们可以是其他开发人员的一部分，也可以是独立的人员。
- 文档编写人员：
  - 专门负责写作软件开发各种文档的人员。他们的存在是为了充分利用部分宝贵的人力资源（例如需求工程师和软件体系结构师），让这些人力资源从繁杂的文档化工作中解放出来。



If you want to be an excellent software engineer, you need to prepare these knowledge.

How to learn SE

外因

- 问题背景
  - 职业决定
  - 学习特殊领域的特点：嵌入式、信息系统、网络...
- 工具
  - 掌握常用工具的使用: 编程工具、管理工具...
  - 实践
- 技术、方法（重要）
  - 课堂学习： 结构化、面向对象...
  - 实践巩固

内因

- 哲学与解决问题的能力
- 知识背景（重要）
    - 课堂学习基础
    - 课外大量阅读
- 实践经验（重要）
    - 多动手实践
    - 多参与交流（浏览专业论坛）
- 灵感
    - 智力 （晶体与流体）
    - Up to you!

- “你纵然学得了剑法，倘若使出时剑法不纯，毕竟还是敌不了当世高手，此刻你已经得到了门径，要想多胜少败，再苦练二十年，便可和天下英雄一较长短了”

— 风清扬