软件工程概述

清华大学软件学院

引言

- 软件在当今社会中发挥着重要的作用
  - 社会经济的发展依赖于软件
  - 更多的系统需要软件控制，软件质量和成本成为关键因素
- 软件工程关注于开发成本和软件质量问题
  - 软件工程的概念开始于 1968 年的 NATO 会议
  - 至今尚未解决大型复杂软件开发的问题
- 软件工程是一个正在兴起的年轻学科
  - 工业界形成了 CMM 和 ISO9000 系列标准
  - IEEE 提出了软件工程知识体系
  - IEEE 提出了软件工程本科教程，成为独立学科

一些基本问题

- 什么是软件？
- 如何理解软件的质量特性？
- 什么是软件工程？
- 什么是软件过程？
- 什么是软件过程模型？
- 什么是软件工程方法？
- 什么是 CASE？
- 当前软件工程面临什么挑战？
- 软件工程学科与哪些学科相关？

内容提纲

- 软件
  - 软件的定义与软件危机
  - 软件的本质特性
- 软件工程
  - 定义与发展历史
  - 理解软件质量
  - 过程、方法和工具
- 软件工程学科
  - 软件工程知识体系（SWEBOK）
  - 软件工程职业道德规范

You are here!
你在这儿！

什么是软件

- 软件 = 程序 ?

- 软件的定义

  软件是计算机程序、规程以及运行计算机系统可能需要的相关文档和数据。

- 从软件的内容来说，软件更像是一种嵌入式的数字化知识，其形成是一个通过交互对话和抽象理解而不断演化的过程。

软件的分类

- 通用软件（Generic Software）
  - 通用软件是由软件开发组织开发，面向市场用户公开销售的独立运行系统，有时也被称为套装软件。
  - 举例：操作系统、数据库系统、字处理软件等

- 定制软件（Customized Software）
  - 定制软件是由某个特定客户委托，软件开发组织在合同的约束下开发的软件。
  - 举例：企业 ERP 系统、卫星控制系统、空中交通指挥系统等

软件的应用

- 举例

实时系统:	空中交通控制系统
嵌入式系统:	数码相机，GPS
数据处理系统:	电话帐单，退休金
信息系统:	网站，数字图书
传感系统:	气象数据
系统软件:	操作系统，编译器
通信软件:	路由器，移动电话
办公系统:	文字处理，视频会议
科学计算软件:	仿真模拟，天气预报
图形软件:	电影制作，CAD 设计

软件危机

- 软件危机出现于 20 世纪 60 年代末
- 软件危机是指在计算机软件的开发和维护过程中遇到的一系列严重问题。
  - 软件开发的成本和进度难以准确估计，延迟交付甚至取消项目的现象屡见不鲜
  - 软件存在着错误多、性能低、不可靠、不安全等质量问题
  - 软件维护极其困难，而且很难适应不断变化的用户需求和使用环境

Software systems are like cathedrals; first we build them and then we pray.

软件错误的实例

- ARIANE 5 火箭
  - 1996 年 6 月，耗资 70 亿美元，发射 37 秒后爆炸
  - 发射失败的原因在于软件的错误
- 软件错误
  - 程序中试图将 64 位浮点数转换成 16 位整数时产生溢出
  - 缺少错误处理程序对数据溢出进行管理
  - 备份软件复制而成
- 严格地遵守软件确认过程可以避免这种错误

软件错误的实例

- 爱国者导弹
  - 曾在海湾战争期间对抗伊拉克飞毛腿导弹
  - 1991 年 2 月，一次对抗失利中 28 名美国士兵丧生
  - 问题的症结在于导弹软件包含一个累加计时误差
- 软件错误
  - *target = f (velocity, time)*
  - 计时采用系统时钟（即 1/10 秒）并使用整数表达
  - $(1/10)_2 = 0.0001100110011001100110011001100\ldots$，24 位寄存器存储导致误差 $(0.000000095)_{10}$
  - $0.000000095 \times 100 \text{ hours} \times 60 \times 60 \times 10 = 0.34 \text{ seconds}$

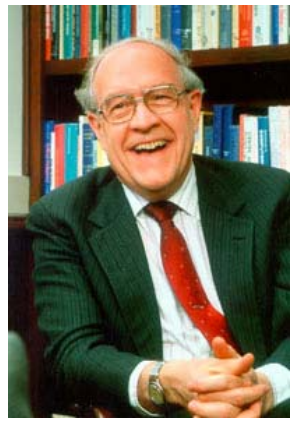
软件错误的实例

- Therac 25 放射医疗仪事故
  - 1986 年由于软件错误导致放射过量，2 人死亡
  - 溢出错误是导致问题的主要原因之一
- 千年虫问题
  - 迫于计算机存储空间的限制，程序员将日期缩减为 2 位数
  - 世界各地更换或升级 2000 年问题软件的花费超过数亿美元
- 其他
  - 电子邮件的病毒
  - 拒绝访问等的网络攻击
  - 网络事务的安全问题

软件的本质特性

- *No silver bullet: essence and accidents of software engineering*
  - Fred Brooks，IBM OS360 项目经理
  - 1999 年图灵奖获得者

“没有任何技术或管理上的进展，能够独立地许诺十年内使生产率、可靠性或简洁性获得数量上的进步。”



- 软件的本质特性
  - 复杂性（*Complexity*）
  - 一致性（*Conformity*）
  - 可变性（*Changeability*）
  - 不可见性（*Invisibility*）

软件的本质特性

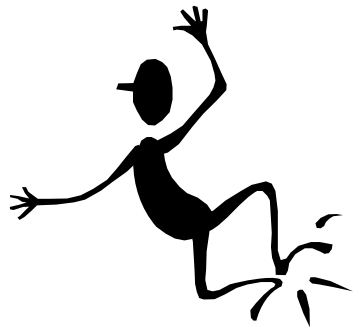
- 复杂性
  - 软件在规模上可能比任何由人类创造的其他实体都要复杂，复杂性是软件的本质特性。
  - 软件的复杂性是必要属性
    - 大量的组合状态
    - 丰富的结构和相互依赖性
    - 良好的接口用以封装内部的复杂性
  - 开发问题也会增加复杂性
    - 高效率的代码通常是复杂的
    - 重用通用化的组件意味着复杂的状态连接
    - 复杂的代码难以维护，导致设计上的更复杂

软件的本质特性

- 一致性
  - 软件必须遵从人为的惯例并适应已有的技术和系统
    - 软件必须遵循各种接口、协议和标准
    - 有些情况下，兼容性是软件开发的目标
  - 软件需要随接口的不同而改变，随时间的推移而变化，而这些变化是不同的人设计的结果。
  - 许多复杂性来自保持与其他接口的一致，对软件的任何再设计，都无法简化这些复杂特性。

软件的本质特性

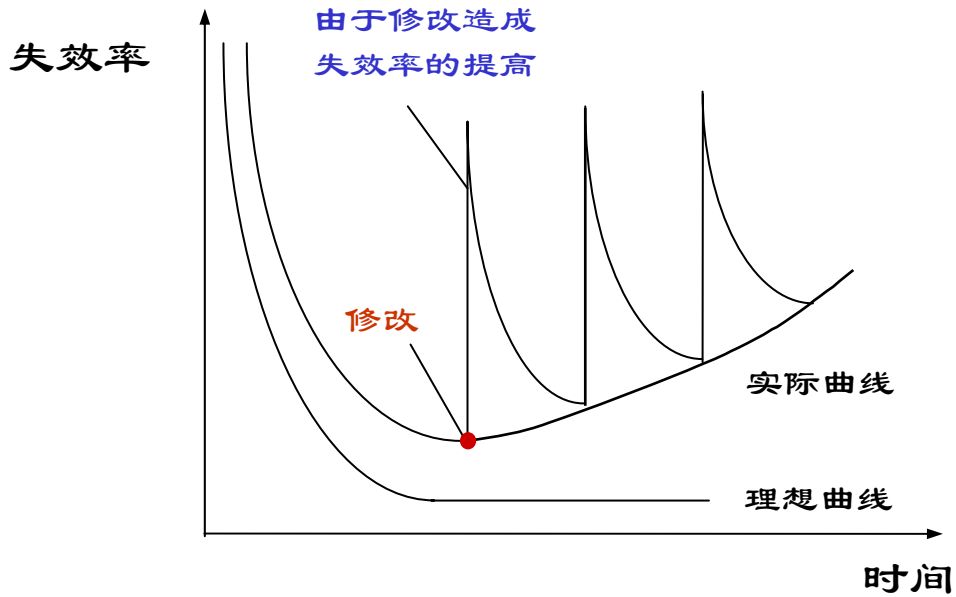
- 可变性
  - 软件产品扎根于文化的母体中，如各种应用、用户、自然及社会规律、计算机硬件等，后者持续不断地变化着，这些变化无情地强迫着软件随之变化。
  - 所有成功的软件都会发生变更！
    - 当人们发现软件很有用时，会在原有应用范围的边界，或者在超越边界的情况下使用软件；
    - 功能扩展的压力主要来自那些喜欢基本功能，又对软件提出了很多新用法的用户们。



软件的本质特性

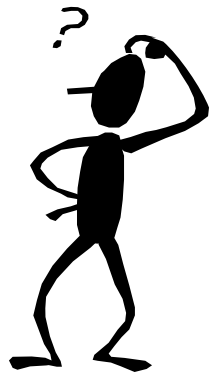
- 可变性（续）
  - 人们总是认为软件是容易修改的，但忽视了修改所带来的副作用，不断的修改最终导致软件的退化。

软件的失效率曲线



软件的本质特性

- 不可见性
  - 软件是不可见的和无法可视化的
    - 软件的客观存在不具有空间的形体特征
    - 定义“需要做什么”成为软件开发的根本问题
  - 人们一直试图使用不同的技术进行软件可视化
    - 控制流程、数据流、依赖关系、UML、……
    - 这些技术仍然无法给出准确的、完整的描述
  - 软件仍然保持着无法可视化的固有特性，从而剥夺了一些具有强大功能的概念工具的构造思路。这种缺憾不仅限制了个人的设计过程，也严重地阻碍了相互之间的交流。



内容提纲

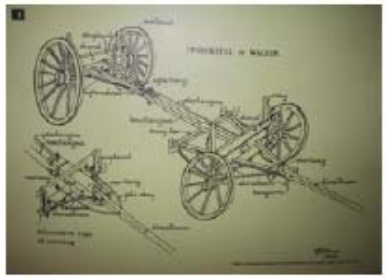
- 软件
  - 软件的定义与软件危机
  - 软件的本质特性
- 软件工程
  - 定义与发展历史
  - 理解软件质量
  - 过程、方法和工具
- 软件工程学科
  - 软件工程知识体系（SWEBOK）
  - 软件工程职业道德规范

You are here!

你在这儿！

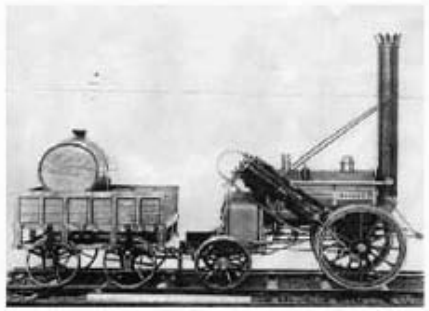
工程的含义

- 工程是将理论和所学的知识应用于实践的科学，以便经济有效地解决实际问题。



- craft
  - personal skill
  - experience
  - flexible materials

- engineering
  - tables and tools
  - recorded knowledge
  - controlled materials



工程的含义

- 规模上的差异
  - 花园小道 vs. *汽车高速公路*
  - 树上小屋 vs. *摩天大楼*
  - 加法程序 vs. *医院档案系统*
- 手工（*Craft*）：小规模的设计与建造
  - 简单问题与单一目标
  - 个人控制与个人技能
- 工程（*Engineering*）：大规模的设计与建造
  - 复杂问题与目标分解
  - 多人参与，需要考虑运营、管理、成本、质量控制、安全等

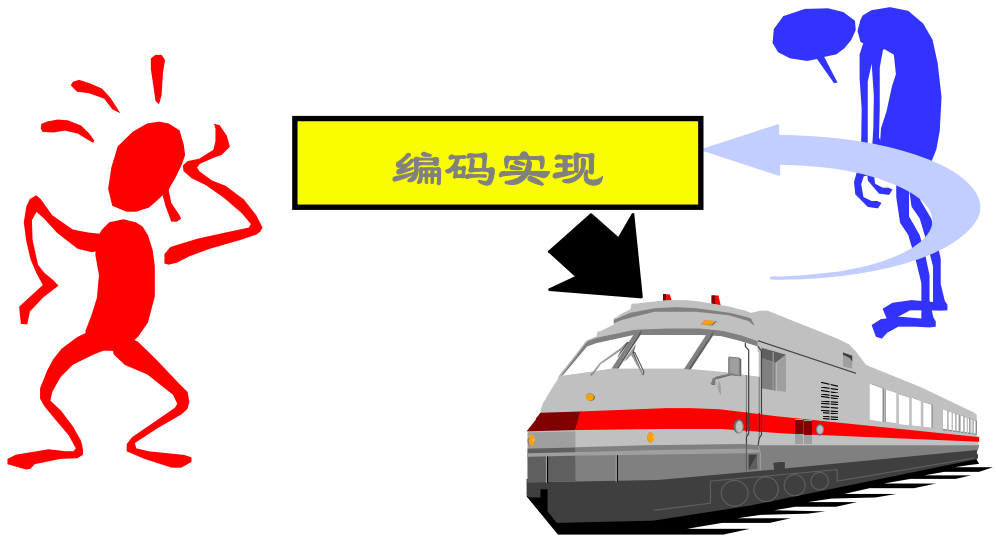
工程的特征

- 平衡与决策
  - 需要进行一系列决策和认真评价，并在每一个决策点做出适当选择，适当与否可以通过平衡成本和利益的分析来判断。
- 度量与验证
  - 应该度量事物，在适当的时候定量工作；需要校正并验证度量，并在经验和实验数据的基础上进行近似。
- 运用工具
  - 工程师需要将工具系统地应用在过程中，因此选用适当的工具是工程的关键。

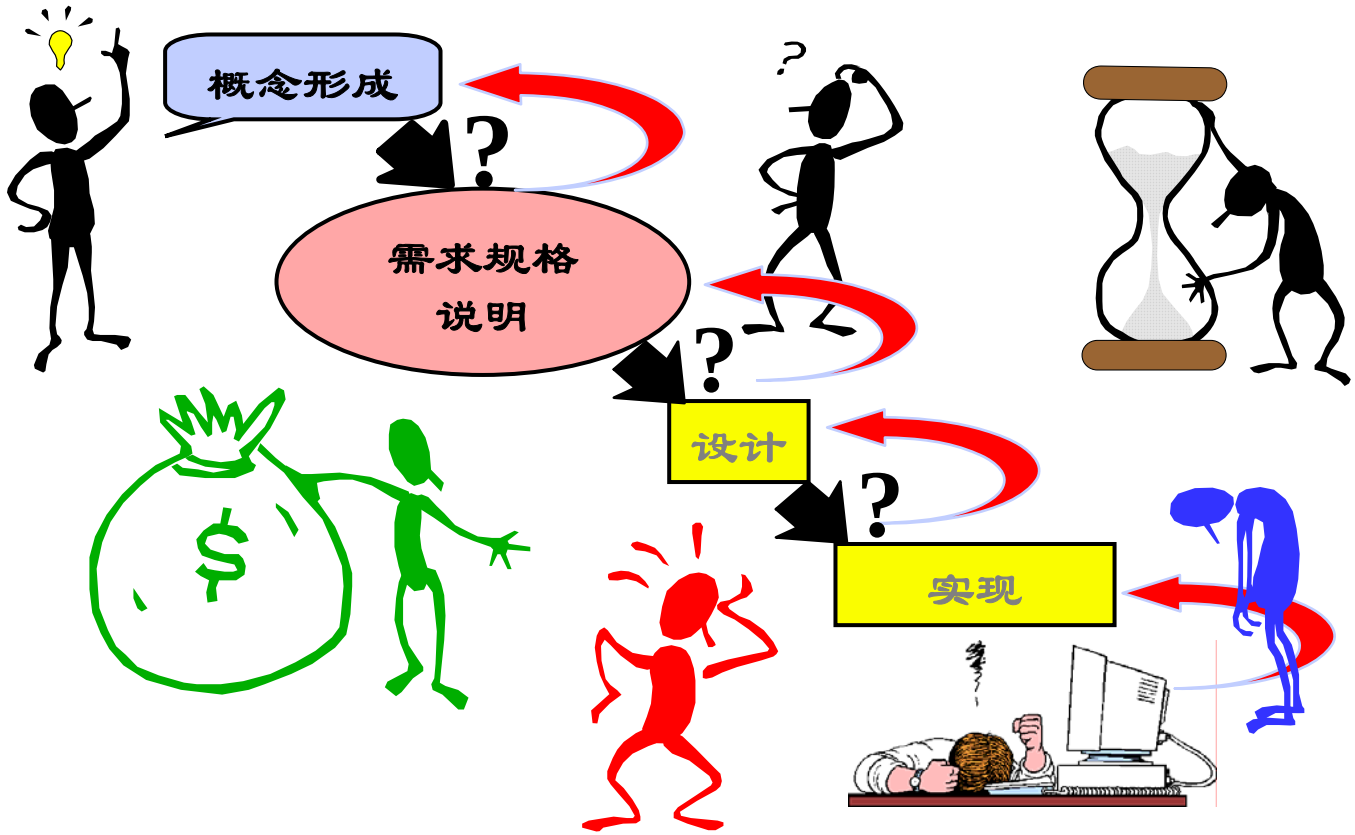
工程的特征

- 团队协同工作
  - 注重训练有素，并以团队的形式进行有效的工作。
- 角色分工
  - 多重角色：研究、开发、设计、生产、测试、构造、实施、管理以及其他诸如销售、咨询和教学等。
- 最佳实践
  - 通过专业团体不断地开发和确认工程原则、标准和实践。
- 强调重用
  - 工程师应该重用设计和设计制品。

只有编码的开发过程



工程化的软件开发



什么是软件工程

- 软件工程的定义

[Bauer, 1972]

软件工程是为了经济地获得能够在实际机器上高效运行的可靠软件而建立和使用的一系列好的工程化原则。

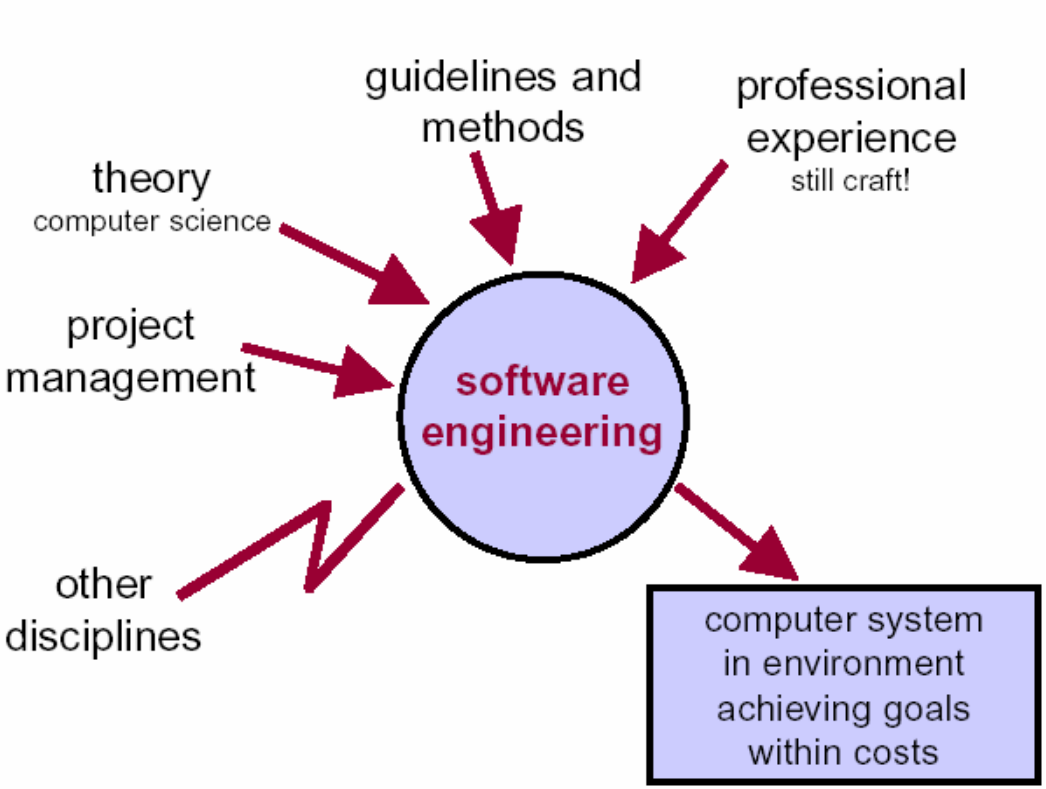
[CMU, 1990]

软件工程是以工程的形式应用计算机科学和数学原理，从而经济有效地解决软件问题。

[IEEE, 1993]

软件工程是①将系统性的、规范化的、可定量的方法应用于软件的开发、运行和维护，即工程化应用到软件上；②对①中所述方法的研究。

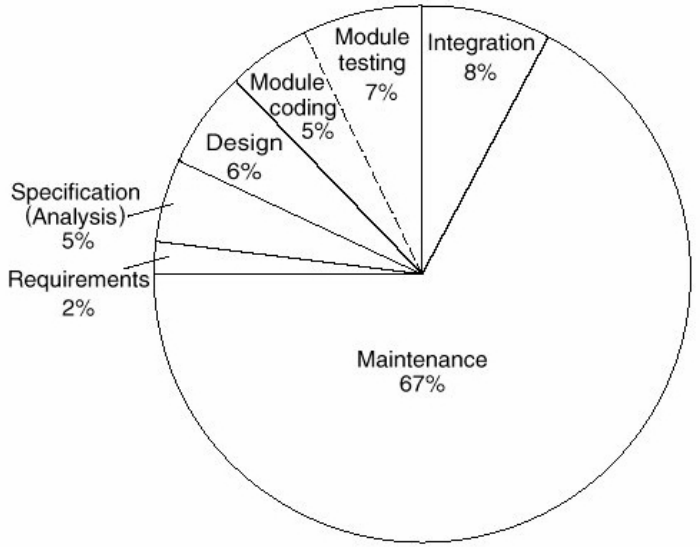
什么是软件工程


guidelines and
methods
professional
experience
still craft!
theory
computer science
project
management
software
engineering
other
disciplines
computer system
in environment
achieving goals
within costs

软件工程的关注焦点

- 软件质量（*Software Quality*）
  - 软件质量是软件产品与明确的和隐含的需求相一致的程度
  - 软件质量通常采用一系列质量特性来描述

- 软件成本（*Software Cost*）
  - 软件开发成本是指软件开发过程中所花费的费用
  - 软件维护成本是指软件投入运行后软件变更所需的费用



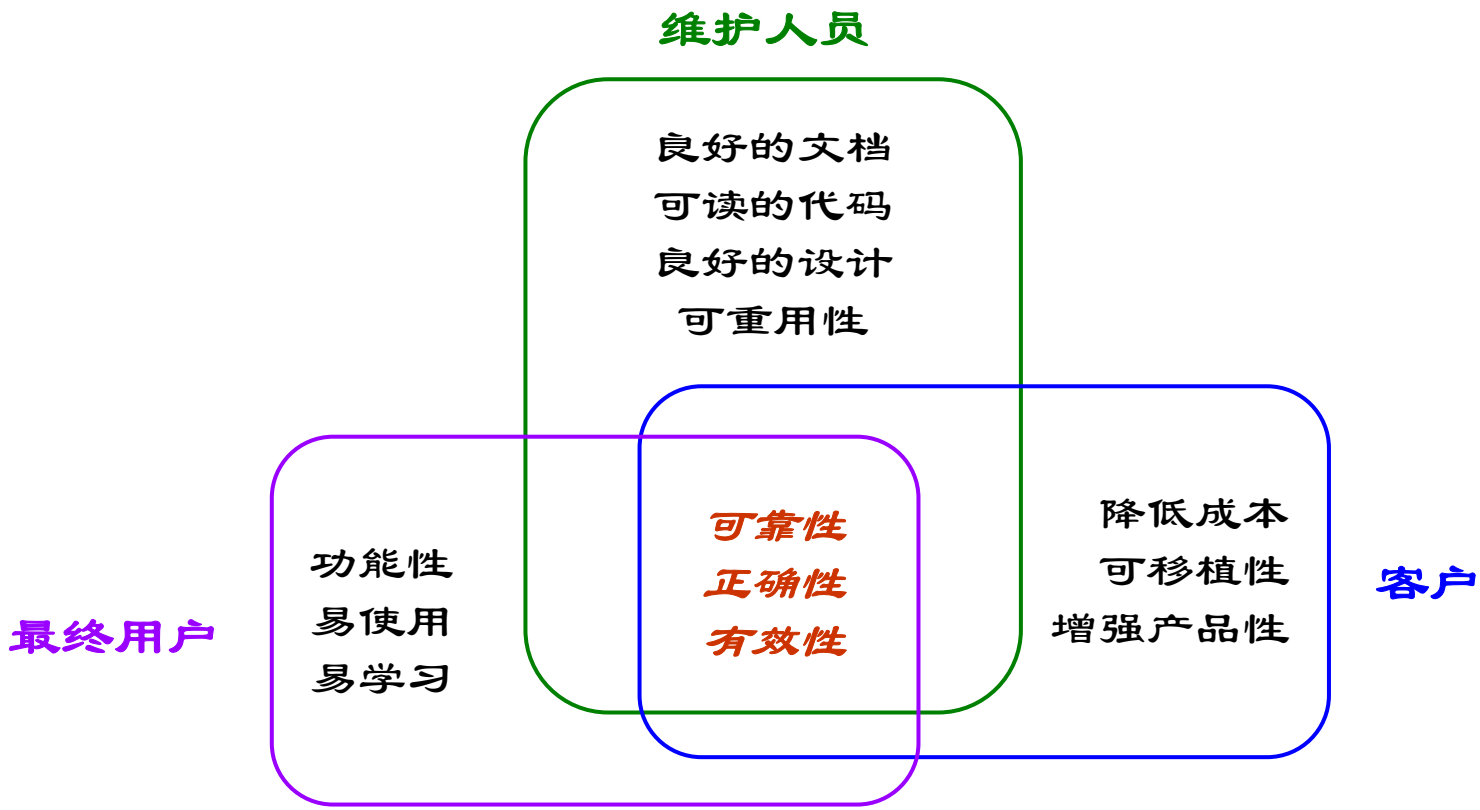
理解软件质量

- 你同意以下说法吗？为什么？

  “运行正确的软件就是高质量的软件。”

- 软件除了提供用户所需的功能以外，还应该具有一系列反映质量的属性，包括可维护性、可依赖性、有效性和可用性等。
  - 可维护性：软件必须能够不断进化以满足客户的需求变化
  - 可依赖性：软件必须是可靠的、保密的、安全的
  - 有效性：软件不应该浪费内存和处理器等系统资源
  - 可用性：软件必须是可用的，用户可以很方便地使用

理解软件质量


维护人员
良好的文档
可读的代码
良好的设计
可重用性
最终用户
功能性
易使用
易学习
可靠性
正确性
有效性
降低成本
可移植性
增强产品性
客户

软件工程的三要素

- 软件工程以关注软件质量为目标，包括过程、方法和工具三个要素。
- *过程*
  - 支持软件生命周期的所有活动
- *方法*
  - 为软件开发过程提供“如何做”的技术
- *工具*
  - 为软件开发方法提供自动的或半自动的软件支撑环境

什么是软件过程

- 软件过程是指开发软件产品的一组活动及其结果。
- 软件过程的四个基本活动
  - *规格说明（Specification）*

    定义软件功能以及对其使用的限制
  - *软件开发（Development）*

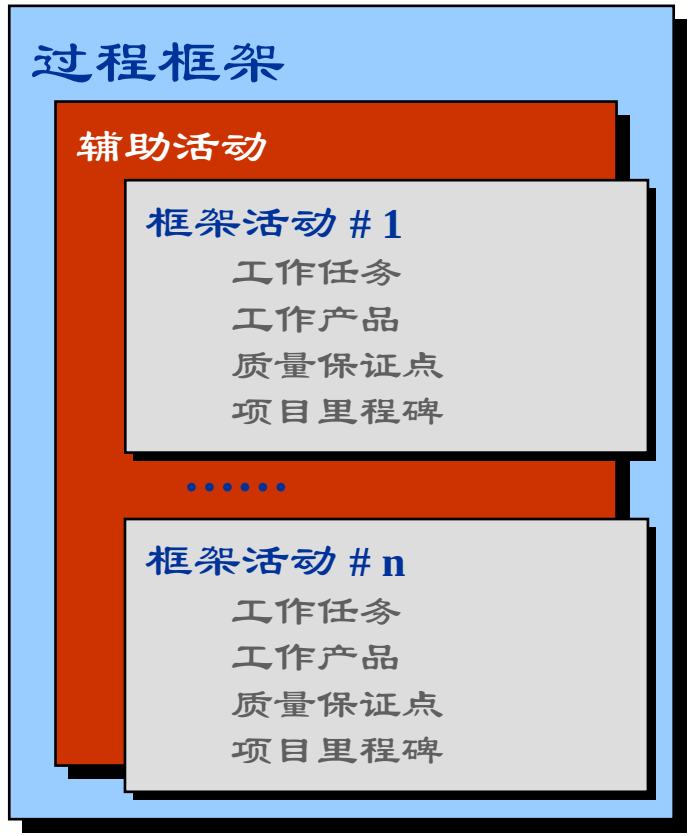
    设计和实现满足规格说明的软件
  - *软件确认（Validation）*

    验证软件以保证能够满足客户的要求
  - *软件演化（Evolution）*

    改进软件以适应不断变化的需求

软件过程框架

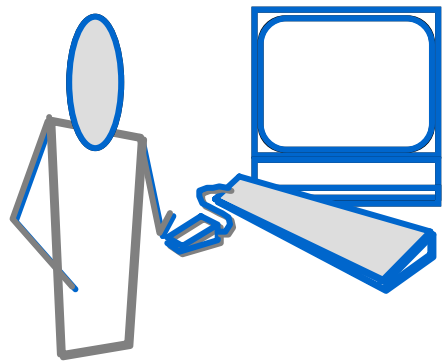
- 软件过程框架
  - 框架活动
    - 涉及软件开发的基本活动
    - 每个任务集合由工作任务、工作产品、质量保证点、项目里程碑组成
  - 辅助活动
    - 包括软件质量保证、软件配置管理、软件项目管理、文档生成与管理、软件度量、风险管理等



什么是软件过程模型

- 软件过程模型是从特定角度呈现的对软件过程的简化描述。
  - 软件过程模型是对实际过程的抽象描述
  - 包括软件过程的活动、软件产品以及参与人员的不同角色

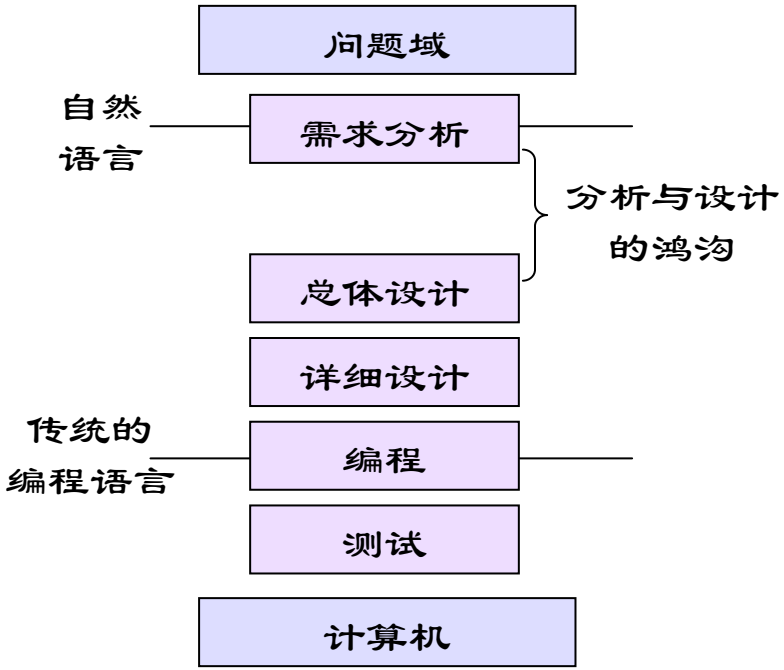
- 常见的软件过程模型
  - 瀑布模型
  - 进化式开发模型
  - 形式化模型
  - 组件式开发模型

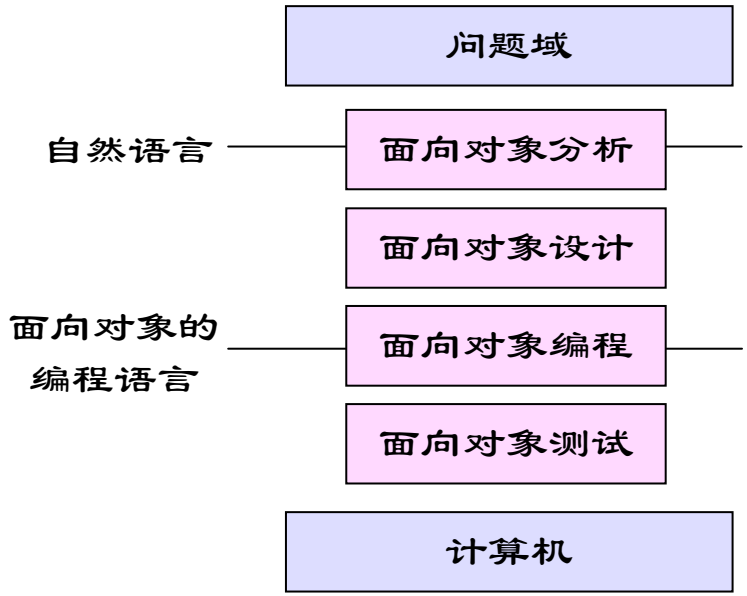


什么是软件工程方法

- 软件工程方法是软件开发的结构化方法，包括模型描述、规则、设计建议和过程指南等
  - 模型描述
    - 对所开发的系统建立图形化的模型描述
    - 例如：对象模型、数据流模型、状态机模型
  - 规则
    - 应用于系统模型的约束
  - 设计建议
    - 有关良好设计实践的建议
  - 过程指南
    - 软件开发所遵循的活动以及这些活动的组织结构

结构化方法 VS. 面向对象方法

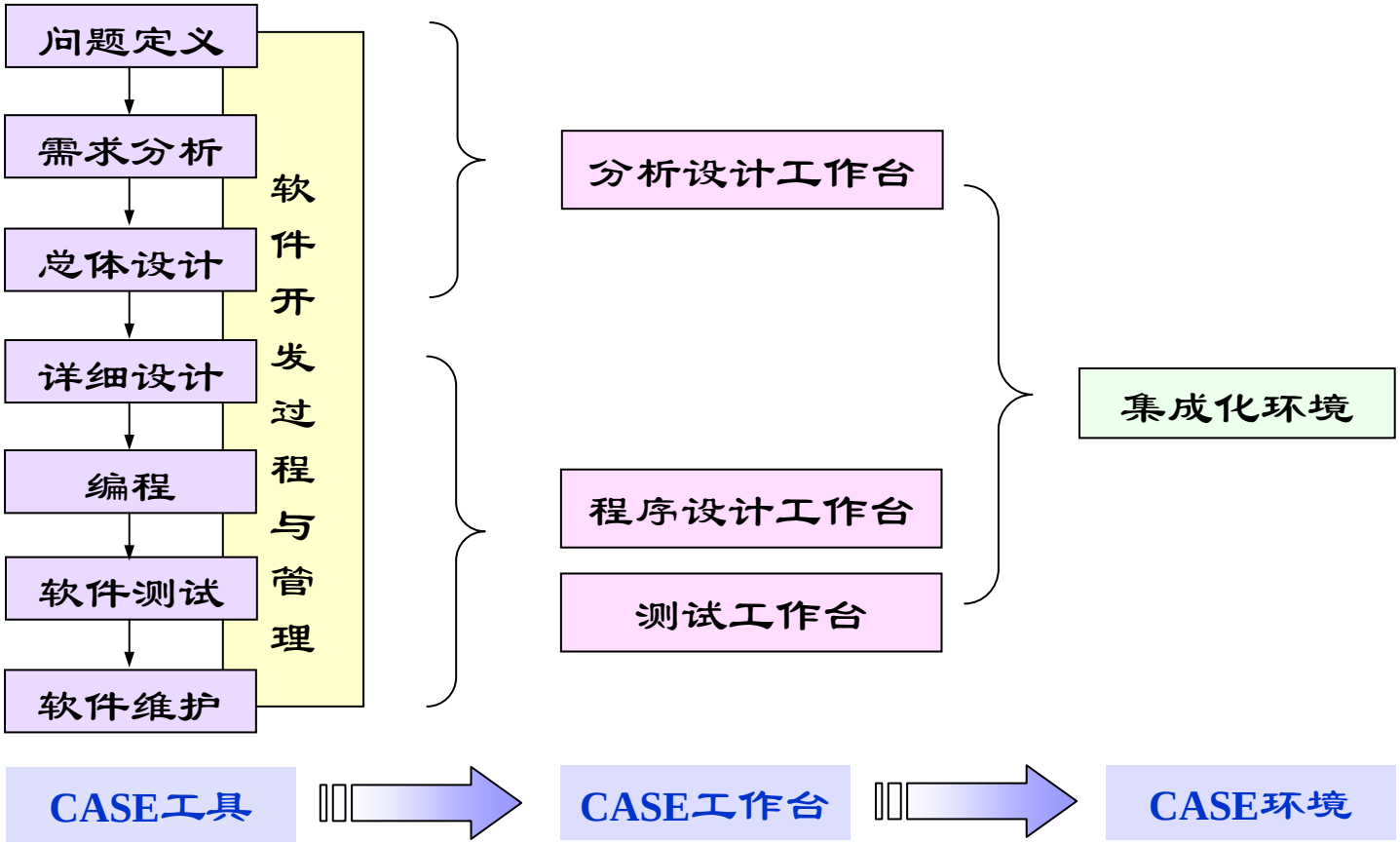




什么是 CASE

- CASE（*Computer Aided Software Engineering*）
  - 计算机辅助软件工程是一组工具和方法的集合，用于辅助软件开发、维护、管理过程中的各项活动，促进软件过程的工程化和自动化。
  - 所有的软件工程方法都需要 CASE 的相应技术来支持
    - 用于系统模型的图形编辑器
    - 管理设计实体的数据字典
    - 生成用户界面的 GUI 软件
    - 辅助生成系统文档的报告生成器
    - 支持程序纠错的调试器
    - 代码生成器
    - ……

CASE 的层次

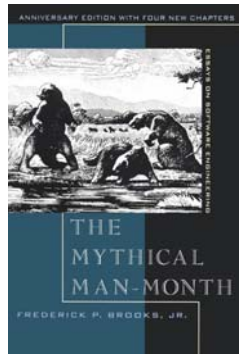

问题定义
需求分析
总体设计
详细设计
编程
软件测试
软件维护
软件开发过程与管理
分析设计工作台
程序设计工作台
测试工作台
集成化环境
CASE工具
CASE工作台
CASE环境

CASE 工具

- IBM Rational 公司产品（*http://www.rational.com/*）
  - 开发过程管理：*RUP*
  - 需求管理：*RequisitePro*
  - 可视化建模：*Rose*
  - 自动测试：*Robot, Test Realtime, TestManager, XDE Tester*
  - 项目管理：*ProjectConsole*
  - 配置管理：*ClearCase, ClearQuest*
- 开源 CASE 工具（*http://sourceforge.net/*）
  - *CVS:* 应用广泛的版本管理工具
  - *UML Modeler:* UML 模型图形编辑工具
  - *UML2EJB:* 将 XML 表示的 UML 模型转换成 EJB 代码的转换器

软件工程的发展

- 软件工程的诞生
  - 1968 年 10 月，NATO 科学委员会在德国的加尔密斯举行会议讨论大型软件项目的若干问题
  - 提出了“*软件工程*”和“*软件危机*”的术语
- 一本经典的著作“*人月神话（Mythical Man Month）*”
  - Brooks 在 1975 年出版
  - 描写了大型软件开发中的许多关键问题
  - Brooks 法则：向进度落后的项目中增加人手，只会使进度更加落后。



people ≠ time

软件工程的发展

- 控制机器（1956－1967）
  - 软件工程的史前时代
  - 批处理系统
  - 汇编语言，FORTRAN语言，COBOL语言
  - 已经认识到软件开发不仅是编码
- 控制过程（1968－1982）
  - 软件工程成为一个研究领域
  - 出现软件产品的定价和独立的软件产业
  - 结构化开发技术
  - Pascal语言，C语言，面向对象编程语言
  - 形成软件生命周期的概念，以瀑布模型为典型

软件工程的发展

- 控制复杂性（1983－1995）
  - PC 时代的到来
  - CASE 工具的开发
  - 原型化开发技术， CMM，软件过程改进活动
  - 面向对象开发技术
  - Objective C， C++， JAVA 语言
- 1996－现在
  - 分布式计算
  - 应用领域的扩展
  - 网络环境的软件工程
  - 面向对象技术的进一步发展（设计模式、组件、中间件）

软件工程面临的挑战

- 遗留系统的问题
  - 遗留系统是指那些过时或存在问题的计算机系统，通常是许多年以前开发的
  - 挑战：既要以合理的成本维护和更新系统，又要能够继承系统中重要的商业信息和服务

- 异构系统的问题
  - 网络环境下包含不同的硬件平台和软件系统
  - 挑战：需要提出新的开发技术，能够使所开发的软件系统运行在不同的硬件平台和系统环境下

软件工程面临的挑战

- 高可信软件开发的要求
  - 软件的重要作用要求正确性、可靠性、安全性等可信性质
  - 挑战：如何在软件的开发和运行中保证其具有高可信的性质
- 软件开发方式的变化
  - 网络时代带来的冲击
    - 开源软件开发技术
    - Web 工程
  - 挑战：研究分布式的软件体系结构和开发模式，探索与之相适应的软件工程策略

内容提纲

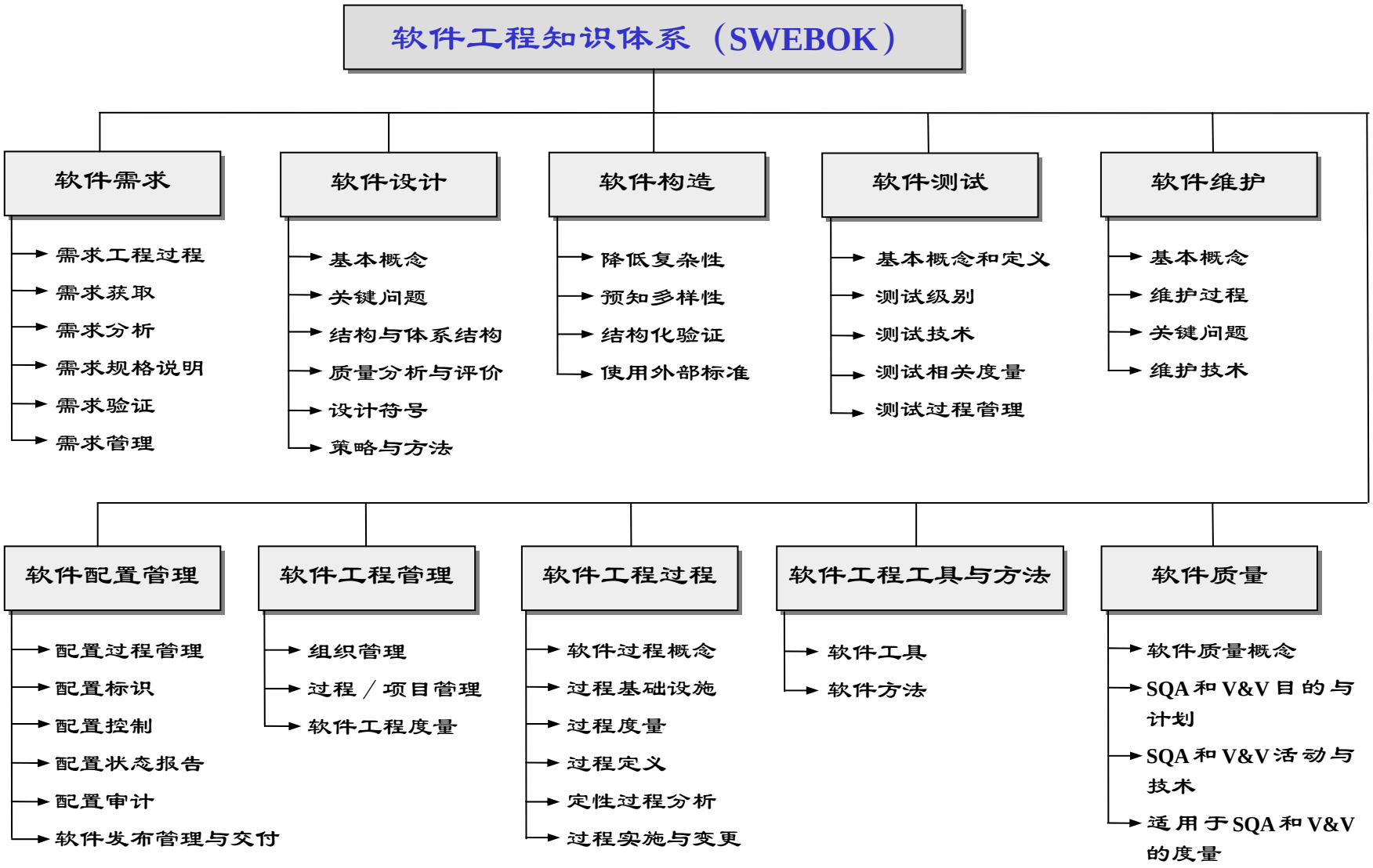
- 软件
  - 软件的定义与软件危机
  - 软件的本质特性
- 软件工程
  - 定义与发展历史
  - 理解软件质量
  - 过程、方法和工具
- 软件工程学科
  - 软件工程知识体系（SWEBOK）
  - 软件工程职业道德规范

You are here!
你在这儿！

软件工程知识体系

- 软件工程知识体系（SWEBOK）
  - IEEE 计算机学会发起研究，从而促进软件工程发展成为独立的专业学科
  - 2001 年 5 月完成，最新发布“ SWEBOK 指南 V1.00（试用版）”

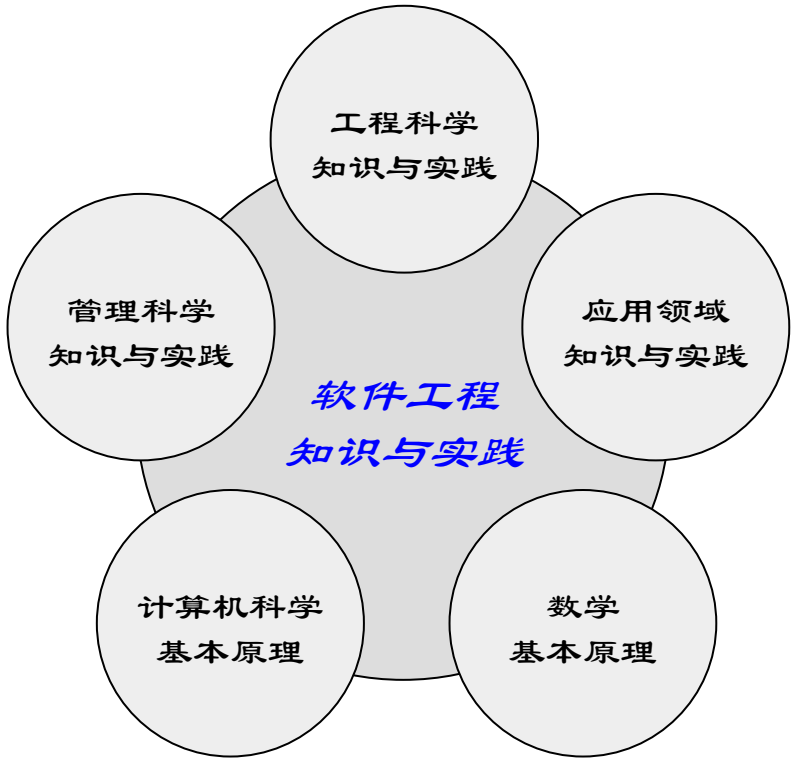
- SWEBOK 的组成
  - 将软件工程知识分解成若干知识域，形成层次化的组成结构
  - 10 个知识域
    - 软件需求、软件设计、软件构造、软件测试、软件维护
    - 软件配置管理、软件工程管理、软件工程过程、软件工程工具与方法、软件质量


软件工程知识体系（SWEBOK）
软件需求
需求工程过程
需求获取
需求分析
需求规格说明
需求验证
需求管理
软件设计
基本概念
关键问题
结构与体系结构
质量分析与评价
设计符号
策略与方法
软件构造
降低复杂性
预知多样性
结构化验证
使用外部标准
软件测试
基本概念和定义
测试级别
测试技术
测试相关度量
测试过程管理
软件维护
基本概念
维护过程
关键问题
维护技术
软件配置管理
配置过程管理
配置标识
配置控制
配置状态报告
配置审计
软件发布管理与交付
软件工程管理
组织管理
过程／项目管理
软件工程度量
软件工程过程
软件过程概念
过程基础设施
过程度量
过程定义
定性过程分析
过程实施与变更
软件工程工具与方法
软件工具
软件方法
软件质量
软件质量概念
SQA和V&V目的与计划
SQA和V&V活动与技术
适用于SQA和V&V的度量

软件工程及其相关学科

- 软件工程是应用计算机科学、数学与管理科学等基本原理，开发软件的工程。它借鉴传统工程的原则和方法，以提高质量，降低成本为目的。

- 计算机科学和数学用于构造软件的模型与算法；工程科学用于制定规范、设计范型、评估成本及确定权衡；管理科学用于计划、资源、质量、成本等管理。



软件工程与计算机科学

- 软件工程与计算机科学的区别
  - 计算机科学
    - 研究构成计算机和软件系统基础的有关理论和方法
    - 举例：数据结构、离散数学、算法分析等
  - 软件工程
    - 研究开发和发布软件的实际问题
    - 举例：飞行控制软件

- 软件工程的研究与实践包括两个方面，一是根植于计算机科学，二是表现为一种工程学科。

软件工程职业道德规范

- 软件工程不仅是技术的应用，还包括许多责任。
  - 软件工程人员应当遵循本行业的职业道德规范，否则无法在这个行业中长久立足。

- 分析下面的一些行为
  - 将公司的软件代码随意复制给其他人
  - 为了个人的利益作出不切实际的承诺
  - 使用盗版软件
  - 迫于时间压力交付缺乏严格测试的代码
  - 未经允许就在别人的机器上玩游戏和上网
  - ……

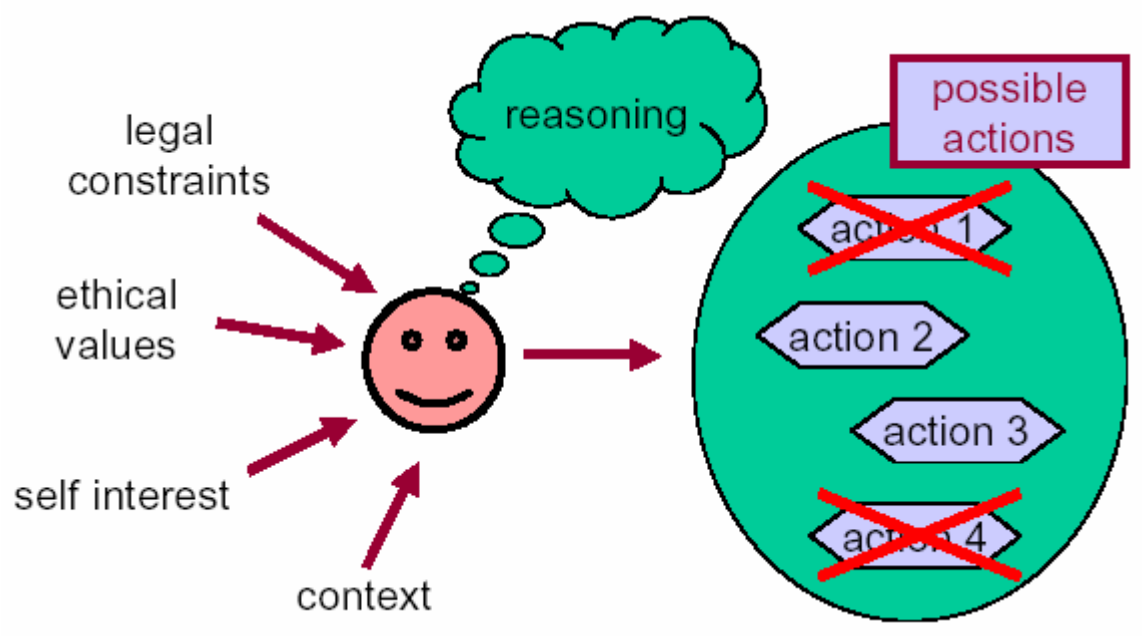
IEEE / ACM 职业道德准则

1. 公众
   - 软件工程人员应始终与公众利益保持一致。
2. 客户和雇主
   - 在与公众利益保持一致的原则下，软件工程人员应满足客户和雇主的最大利益。
3. 产品
   - 软件工程人员应当确保他们的产品及其改进符合尽可能高的专业标准。
4. 判断
   - 软件工程人员应当具备公正和独立的职业判断力。

IEEE / ACM 职业道德准则

5. 管理
   - 软件工程管理者和领导者应拥护和倡导合乎道德的有关软件开发和维护的管理方法。
6. 职业
   - 在与公众利益一致的原则下，软件工程人员应当提高职业的信誉。
7. 同行
   - 软件工程人员对其同行应持平等和支持的态度。
8. 自我
   - 软件工程人员应当终身学习专业知识，促进合乎道德的职业实践方法。

道德的判断


reasoning
possible actions
legal constraints
ethical values
self interest
context
action 2
action 3

有影响的软件工程期刊

- *Transactions on Software Engineering*（IEEE）
- *Software*（IEEE）
- *Software Engineering Notes*（ACM Special Interest Group）
- *Transactions on Software Engineering and Methodology*（ACM）
- *The Journal of Systems and Software*（Elsevier）
- *Proceedings of the International Conference on Software Engineering*（ACM／IEEE）
- *Proceedings of the International Conference on Software Maintenance*（IEEE）
- *Software Maintenance: Research and Practice*（Wiley）

参考文献

1. F.P. Brooks Jr., “No silver bullet: essence and accidents of software engineering”, *IEEE Computer*, vol. 20, No. 4, pp10-19, 1987.

2. Brooks, F., *The Mythical Man-Month*, Addison-Wesley, 1975

3. Bourque, Pierre, et al., “The Guide to the Software Engineering Body of Knowledge”, *IEEE Software*, Nov./Dec., pp 35-44, 1999

4. Abran, A. and J. Moore, *SWEBOK: Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society Press, 2002, can be download at http://www.swebok.org/

5. IEEE Computer Society/ACM Joint Task Force on Software Engin-eering Ethics and Professional Practices. *Software engineering code of ethics and professional practice*, available at http://computer.org/tab/seprof/code.htm