复旦大学 计算机学院 研究生专业课程

高级软件工程

软件工程概述

彭鑫

pengxin@fudan.edu.cn

www.se.fudan.edu.cn/pengxin

内容提要

- 软件工程的诞生
- 软件工程基本概念
- 软件开发的根本性困难
- 软件工程知识领域

内容提要

- 软件工程的诞生
- 软件工程基本概念
- 软件开发的根本性困难
- 软件工程知识领域

软件工程的诞生

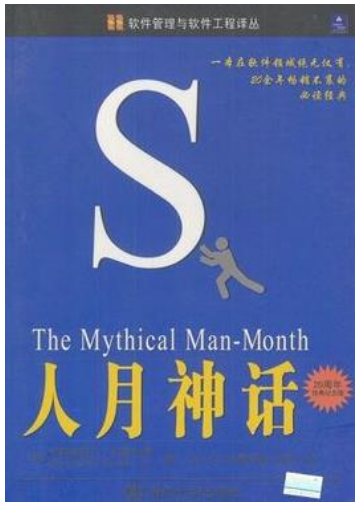
- 60多年前，计算机科学从电子学中脱颖而出
- 为克服以手工作坊方式为主生产软件带来的软件危机，1968年NATO会议上，第一次提出了软件工程的概念
- 从此，软件从硬件中脱颖而出，软件工程从计算机体系结构中脱颖而出

软件危机(Software Crisis)

- 软件开发进度难以预测
- 软件开发成本难以控制
- 用户对产品功能难以满足
- 软件产品质量无法保证
- 软件产品难以维护

IBM360系统的例子

- 开发时间：1963-1966年
- 投入人力：5000人年
- 代码量：100万行
- 耗资数亿美元
- 结局
  - 发布推迟，成本也是估计的好几倍
  - 需要的内存比计划中的要多
  - 第一次发布时并不能很好地运行，直到发布了几次以后(每个后续版本都是找出上千个错误后修订的结果)



FrederickP.Brooks.Jr.

BROOKS的总结

.....正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难。.....程序设计工作正像这样一个泥潭，.....一批批程序员被迫在泥潭中拼命挣扎，.....谁也没有料到问题竟会陷入这样的困境.....

软件工程概念的诞生

Fritz Bauer, NATO, 1968

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

Doug McIlroy on Software Components, 1968

内容提要

- 软件工程的诞生
- 软件工程基本概念
- 软件开发的根本性困难
- 软件工程知识领域

工程(ENGINEERING)

字典中的定义

The application of scientific and mathematical principles to practical ends such as the design, manufacture, and operation of efficient and economical structures, machines, processes, and systems.

工程化生产的特征

- 质量及成功在很大程度上能够得到保障
- 可以通过标准等手段进行明确总结和定义
- 所涉及的原则有相应的理论和实践支撑
- 通常包含一系列定义良好的过程，包括活动、相关的工具、角色和工作指南
- 可重复(可复制成功)、可学习、可培训、可遵循
- 稳定、可控
- 实用性原则

软件工程的定义

- **Fritz Bauer**：为了经济地获得可靠的和能在实际机器上高效运行的软件而建立和使用的好的工程原则
- **IEEE**：(1)将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中(2)(1)中所述方法的研究
- **计算机科学技术百科全书(第2版)**：应用计算机科学理论和技术以及工程管理原则和方法，按预算和进度实现满足用户要求的软件产品的工程，或以此为研究对象的学科

软件工程定义的共同点

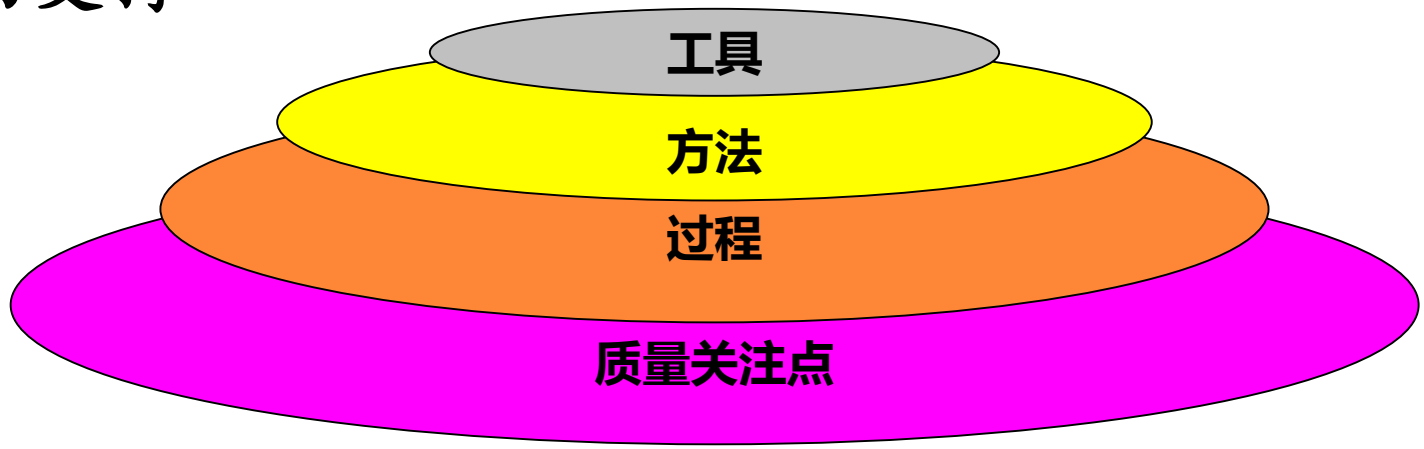
- 目标：以较高的效率、合理的成本开发高质量的软件
- 强调软件开发的工程化：遵循工程的原则，采用系统的、严格约束的、可量化的过程和方法
  - 工程化：可预测、可控制、可重复…
- 包括软件工程实践和方法研究两方面

软件工程的目标

- **生产具有高质量以及开销合理(时间、成本)的产品**
  - 软件质量：软件产品满足预期需求的程度
    - 需求分析和验证—保证问题理解的正确性
    - 软件测试、评审、形式化方法、过程控制—质量保障
  - 开销合理是指软件开发、运行的整个进度和成本满足用户要求的程度
    - 软件项目管理和过程管理—提高可预测性、可控性
    - 好的软件设计、文档和编码风格—可维护性、可扩展性
    - 模块化设计、开发工具支持—提高软件开发效率

软件工程技术体系层次

- 质量关注点(quality focus)是软件工程的根基
- 过程(process)是软件工程的基础
- 方法(methods)为构造软件提供技术上的解决方案
- 工具(tools)为过程和方法提供自动化或半自动化的支持



Roger S. Pressman 著, 郑人杰、马素霞、白晓颖等 译. 软件工程——实践者的研究方法（原书第六版）. 机械工业出版社, 2007年4月.

软件生存周期 (SOFTWARE LIFE CYCLE)

- **软件生存周期：软件孕育、诞生、成长、成熟、衰亡的整个过程**
- **ISO/IEC 12207软件生存周期过程标准将软件生存周期过程分成系统上下文过程和软件特定过程两个大部分**
  - 系统上下文过程：关注系统工程，包括软件产品所处的商业环境、组织管理、需求管理、质量管理和业务运营等环节
  - 软件特定过程关注软件工程
    - 7个软件实施过程
    - 8个软件支持过程
    - 3个软件复用过程

对软件工程的理解

- 面向质量和经济目标的软件开发工程化
- 包括技术和管理
- 对软件整个生存周期的理解
- 软件演化和软件维护
- 针对各种不同类型软件的工程化开发
- 涉及多个层面：软件产业、软件组织、软件项目、软件开发者

内容提要

- 软件工程的诞生
- 软件工程基本概念
- 软件开发的根本性困难
- 软件工程知识领域

软件的噩梦

- **1995年：(美国)只有16.2%的IT项目取得成功，超过31%的项目被取消，花费810亿美元**
- **1996年：欧洲航天局研制的阿里亚娜五型火箭在发射后不到40秒爆炸**
  - 事后调查发现，错误发生于当一个很大的64位浮点数转换为16位带符号整数时出现异常
- **2007年：票务官网开通半小时即瘫痪**
  - 奥组委票务中心主任宣读道歉信
  - 票务官网开通后压力激增，承受了超过设计容量8倍的流量
- **2008年：希思罗机场新航站楼再遇故障**
  - 据新华社电英国伦敦希思罗机场新启用的5号航站楼5日因行李管理系统出现故障再次陷入混乱之中，英国航空公司被迫取消12个短程航班……

软件工程的基本问题

- 软件开发就是将问题空间(所面临的特定问题域)转化为解空间(代码、文档)的过程
- 主要困难
  - 不可见的逻辑产品(对比：物理学和化学对于制造业、建筑业等成熟领域的工程技术支撑)
  - 问题空间和解空间之间的巨大鸿沟
  - 面临的领域纷繁复杂、分布广泛，不同软件系统之间的差异性很大
  - 软件系统的复杂度越来越高(远高于计算机硬件)

软件项目的特点

- 兼具创造性和工程的特点
- 需求来源广泛，涉及经济社会的各个领域
- 开发技术以及用户技术需求发展迅速
  - 内部办公系统—>Web信息发布系统—>Web服务系统—>手机APP
- 导致：项目重复性低，难以凝练、借鉴其它项目经验
- 质量难以量化度量和控制
- 很大程度上依赖于个人能力和管理水平

软件项目的特点-续

- 软件工程过程本身将不可避免地产生新的或变更的客户需求的要求
- 其后果是，软件通常是以迭代的过程开发的，而不是一个相继展开的线形任务序列
- 软件工程有必要综合创造性和规范性两个方面，在二者之间维持平衡常常很困难
- 软件的创新性和复杂性通常很高
- 基础技术的变化速度很快

—IEEE软件工程知识体系(SWEBOK)

软件开发管理的特殊困难

- **根本原因**
  - 软件所特有的兼具创造性和工程性的双重性
  - 许多客户并没有充分认识软件产品内在的复杂性和重要价值
- **表现**
  - 需求变动频繁且得不到客户的足够理解
  - 人员个体差异大，“可替代性”差，团队士气对项目的影响大
  - 沟通与交流至关重要，但往往被忽视

20多年前的软件项目管理(1984)……

- **No CMM/CMMI (1987), No ISO 9001 (1987)**
- **IEEE Transactions on Software Engineering, Special issue on software engineering project management, January 1984**
  - 软件配置管理
  - 软件工程经济学
  - 项目管理计划
  - 挣值技术
  - 软件质量保障
  - 软件评审、走查和内审
  - ……

20多年后的今天

- **制造、建筑、电子设备乃至大型飞机复杂项目的工程化技术逐渐成熟**
  - 项目的可控性、可管理性越来越高
- **软件项目：项目管理问题依然严重**
  - the more things change (TSE January 1984)
    - ISO 9001、CMM/CMMI、PMP …
    - 外包、软件复用/构件化、开源软件…
  - the more they stay the same
    - 失败率仍然很高，结果难以把握
    - 很多项目仍然处于手工作坊阶段，人的个体作用依然很大

没有银弹

- **1986年Brooks的《没有银弹》首次发表 No Silver Bullet – Essence and Accident in Software Engineering**
- 基本论断：没有任何一种单纯的技术或管理上的进步能够独立地承诺在十年内大幅度提高软件生产率、可靠性和简洁性
- 对比：计算机硬件工业中的微电子器件、晶体管、大规模集成对于生产率、可靠性和简洁程度的巨大提高

软件开发的根本困难

- **软件特性中固有的困难**

  软件开发中困难的部分是规格化、设计和测试这些概念上的结构，而不是对概念进行表达和对实现逼真程度进行验证

  - 软件的复杂度是必要属性，不是次要因素：不仅导致技术上的困难，还引发了很多管理上的问题
  - 与各种接口(软硬件、人…)的一致性
  - 变化性：软件产品扎根于庞大的母体中(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，并强迫着软件随之变化
  - 软件的客观存在不具有空间的形体特征(不可见性)：无法构造具有强大功能的可视化概念工具，限制了个人的设计过程，严重地阻碍了相互之间的交流

软件开发的次要困难

- **目前或曾经存在，但并非那些与生俱来的固有困难**
- **在这些方面已经取得了许多重要的进展**
  - 高级语言：开发生产率至少提高了五倍，同时可靠性、简洁性和理解程度也大为提高
  - 分时：保证了及时性，从而使我们能维持对复杂程度的一个总体把握
    - **批处理时代：必须停下编程才能调用编译程序或者执行程序，思维上的中断使我们不得不重新进行思考**
  - 统一的集成开发环境

“可能”的银弹

- Ada和其他高级编程语言
  - 抽象数据类型、层次结构的模块化
- 面向对象编程
- 人工智能
- 专家系统
  - 具体应用的复杂性与程序本身相分离
- “自动”编程
- 图形化编程
- 程序验证
- 环境和工具

针对根本问题有希望的解决途径-1

- **购买和自行开发：构建软件最可能的彻底解决方案是不开发任何软件**
  - 对于软件消费者和开发者
    - 对于消费者：面向大众的软件销售和使用—已经实现
    - 对于开发者：大量使用商用第三方构件(COTS)
  - 蕴含着复用和软件构件的思想
  - 根本问题并未改变：需求过于专业，能够购买复用的仅仅是少量通用软件包
  - 重大的变化在于计算机硬件/软件成本比例
    - 在1960年，200万美元机器的购买者觉得他可以为定制的薪资系统支付25万美元
    - 现在，对5万美元的办公室机器购买者而言，很难想象能为定制薪资系统再支付如此高的费用

针对根本问题有希望的解决途径-2

- **需求精化和快速原型**
  - 软件开发最困难的部分：精确地确定搭建什么样的系统
  - 谬误：事先明确地阐述系统，竞标然后进行实际开发，最后安装
  - 快速原型化系统的方法和工具：解决了软件的根本而非次要问题
- **增量开发—增长，而非一次性搭建**
  - 使逆向跟踪更方便，并非常容易进行原型开发
  - 对士气的推动是令人震惊的
- **卓越的设计人员**

敏捷和精益

Agile - Able to move quickly and easily

可以迅速和容易地移动

-- Longman English Dictionary

Lean - Thin in a healthy and attractive way; No extra fat

瘦，但是健康、强壮；没有多余的脂肪。

-- Longman English Dictionary

敏捷和精益

- 敏捷：通过灵活、快速地响应市场的变化，获取竞争优势
- 精益：通过快速持续的交付价值，获取竞争优势

敏捷宣言

我们正在探索更好的软件开发方法。
通过亲身的、或者协助他人进行软件开发实践，
我们建立了如下的价值观：

个体和交互 重于 过程和工具
工作的软件 重于 详尽的文档
客户合作 重于 合同谈判
响应变化 重于 遵循计划

也就是说，尽管右项有其价值，
我们更重视左项的价值。

内容提要

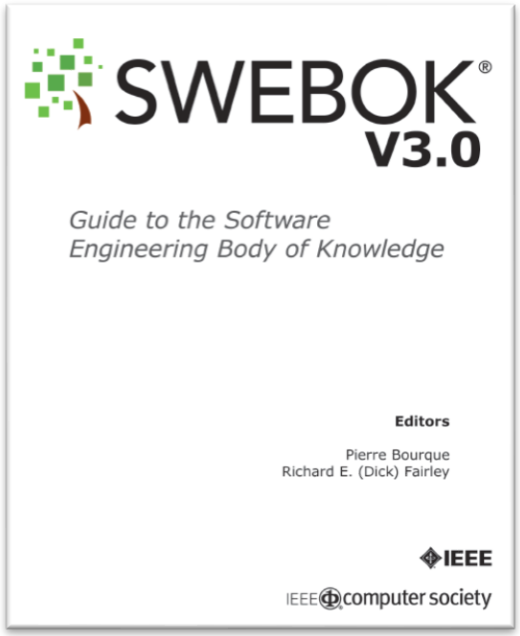
- 软件工程的诞生
- 软件工程基本概念
- 软件开发的根本性困难
- 软件工程知识领域

知识体系(Body of Knowledge)

- 任何一个职业都是建立在相应的知识体系基础上
  - 制定职业培训计划
  - 专家资格认证
  - 职业许可认证

SWEBOK GUIDE

SoftWare Engineering Body Of Knowledge



The 335-page summary of what every software engineer should know about software engineering.

Guide does not purport to define the body of knowledge but rather to serve as a compendium and guide to the body of knowledge that has been developing and evolving over the past four decades.

The *Guide* must, necessarily, develop and evolve as software engineering matures.

SWEBOK
IEEE computer society

SWEBOK Guide的目的

- **定义了广泛认同的软件工程知识体系，为课程、培训和认证提供了基础**
  - 以主题的方式刻画了软件工程知识体系的内容
  - 提升了对于软件工程的统一认识
  - 澄清了软件工程与其他学科（计算机科学、项目管理、计算机工程、数学等）的关系和边界
  - 为开发相关课程、能力认证和职业许可材料提供了基础

SWEBOK GUIDE总体结构

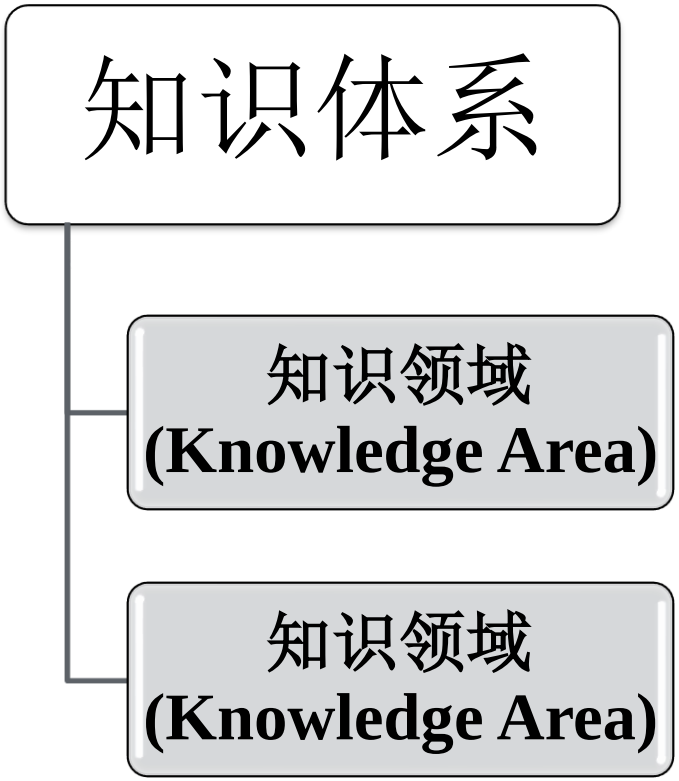


Table I.1. The 15 SWEBOK KAs
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

SWEBOK V3

SWEBOK GUIDE结构示例

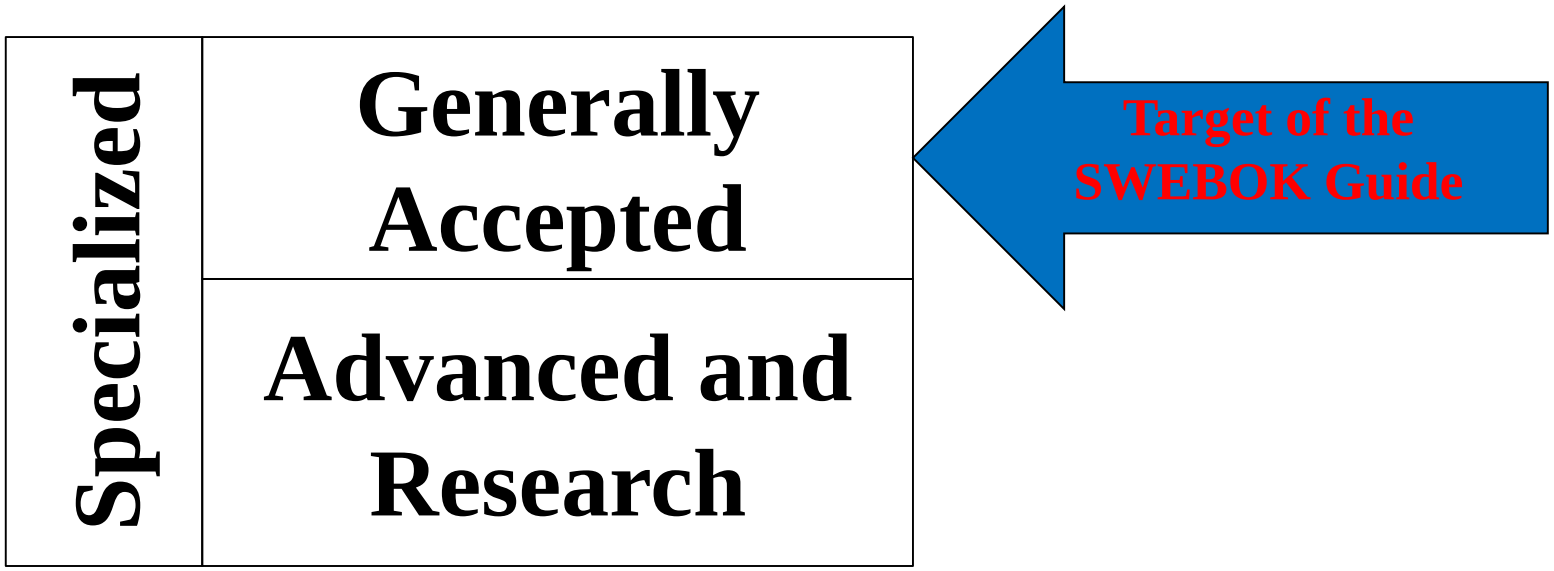
Chapter 3: Software Construction

1. Software Construction Fundamentals
   - *1.1. Minimizing Complexity*
   - *1.2. Anticipating Change*
   - *1.3. Constructing for Verification*
   - *1.4. Reuse*
   - *1.5. Standards in Construction*
2. Managing Construction
   - *2.1. Construction in Life Cycle Models*
   - *2.2. Construction Planning*
   - *2.3. Construction Measurement*
3. Practical Considerations
   - *3.1. Construction Design*
   - *3.2. Construction Languages*
   - *3.3. Coding*
   - *3.4. Construction Testing*
   - *3.5. Construction for Reuse*
   - *3.6. Construction with Reuse*
   - *3.7. Construction Quality*
   - *3.8. Integration*
4. Construction Technologies
   - *4.1. API Design and Use*
   - *4.2. Object-Oriented Runtime Issues*
   - *4.3. Parameterization and Generics*
   - *4.4. Assertions, Design by Contract, and Defensive Programming*
   - *4.5. Error Handling, Exception Handling, and Fault Tolerance*
   - *4.6. Executable Models*
   - *4.7. State-Based and Table-Driven Construction Techniques*
   - *4.8. Runtime Configuration and Internationalization*
   - *4.9. Grammar-Based Input Processing*
   - *4.10. Concurrency Primitives*
   - *4.11. Middleware*
   - *4.12. Construction Methods for Distributed Software*
   - *4.13. Constructing Heterogeneous Systems*
   - *4.14. Performance Analysis and Tuning*
   - *4.15. Platform Standards*
   - *4.16. Test-First Programming*
5. Software Construction Tools
   - *5.1. Development Environments*
   - *5.2. GUI Builders*
   - *5.3. Unit Testing Tools*
   - *5.4. Profiling, Performance Analysis, and Slicing Tools*

Matrix of Topics vs. Reference Material

SWEBOK V3

SWEBOK中的知识选取



Generally Recognized: Applicable to most projects, most of the time, and widespread consensus about their value and usefulness.

Project Management Institute - PMI

成果：多方面共识

- 知识域列表
- 每个知识域的主题列表及相关引用材料
- 相关学科列表

SWEBOK V2 (2004)

知识域(10)	相关学科(8)
软件需求	计算机工程
软件设计	计算机科学
软件构造	管理
软件测试	数学
软件维护	项目管理
软件配置管理	质量管理
软件工程管理	软件人类工程学
软件工程过程	系统工程
软件工程工具和方法	
软件质量	

SWEBOK V3修订

- **修订目的**
  - 反映近些年软件工程研究与实践的新成果
  - 将SWEBOK和CSDA、CSDP、SE2004（软件工程本科课程大纲）、GSwE2009（软件工程硕士课程大纲）、SEVOCAB（软件工程术语）等标准进行统一
  - 合并和更新以上各标准的参考文献，遴选最重要的文献，减少文献数量，以利于读者的学习
  - 建立每三年版本更新的模型和计划
- **计划周期：2008~2010**
- **实际：2014年2月发布**

SWEBOK V3 (2014)

知识域(15)		相关学科(7)
软件需求	软件工程实践知识域（11个）	计算机工程
软件设计		计算机科学
软件构造		管理
软件测试		数学
软件维护		项目管理
软件配置管理		质量管理
软件工程管理		~~软件人类工程学~~
软件工程过程		系统工程
软件工程模型和方法		
软件质量		
NEW! 软件工程职业实践		
NEW! 软件工程经济学	软件工程教育基础知识域（4个）	
NEW! 计算基础		
NEW! 数学基础		
NEW! 工程基础		

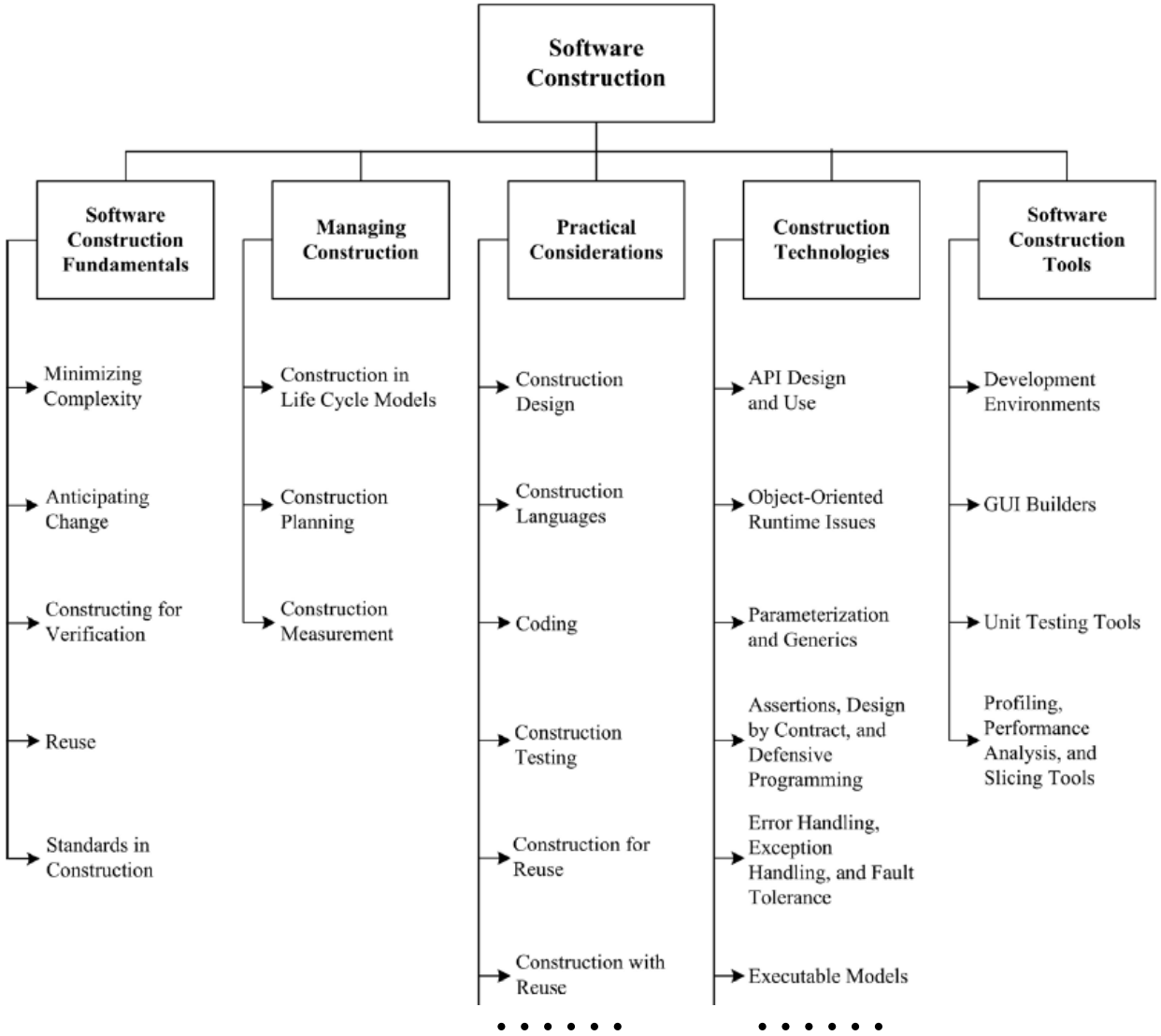
软件工程管理与经济学

- **软件开发不可避免地会涉及到商业因素以及对于软件制品和人的工作的规划、协调和管理**
  - 软件配置管理和软件工程管理：软件配置管理、软件项目管理、软件度量等
  - 软件质量：软件质量和质量管理，包括软件质量的基本概念、软件质量管理过程、管理技术和工具等
  - 软件工程经济学：商业环境下的软件工程决策，包括风险分析、成本效益分析等

计算、数学与工程基础

- **与软件开发、演化和运行相关的**
  - 计算基础：程序设计原理、算法、并行与分布式计算、数据库、网络等
  - 数学基础：集合、关系、函数、逻辑、证明、图和树、有限状态机、语法等
  - 工程基础：经验方法和实验技术、统计分析、工程化设计、建模与模拟、标准等

SWEBOK V3知识域示例：软件构造



高级软件工程

软件工程概述

The End