第一章 软件工程概述

周 瑞

QQ群：235458708

信息与软件工程学院

本章学习目标

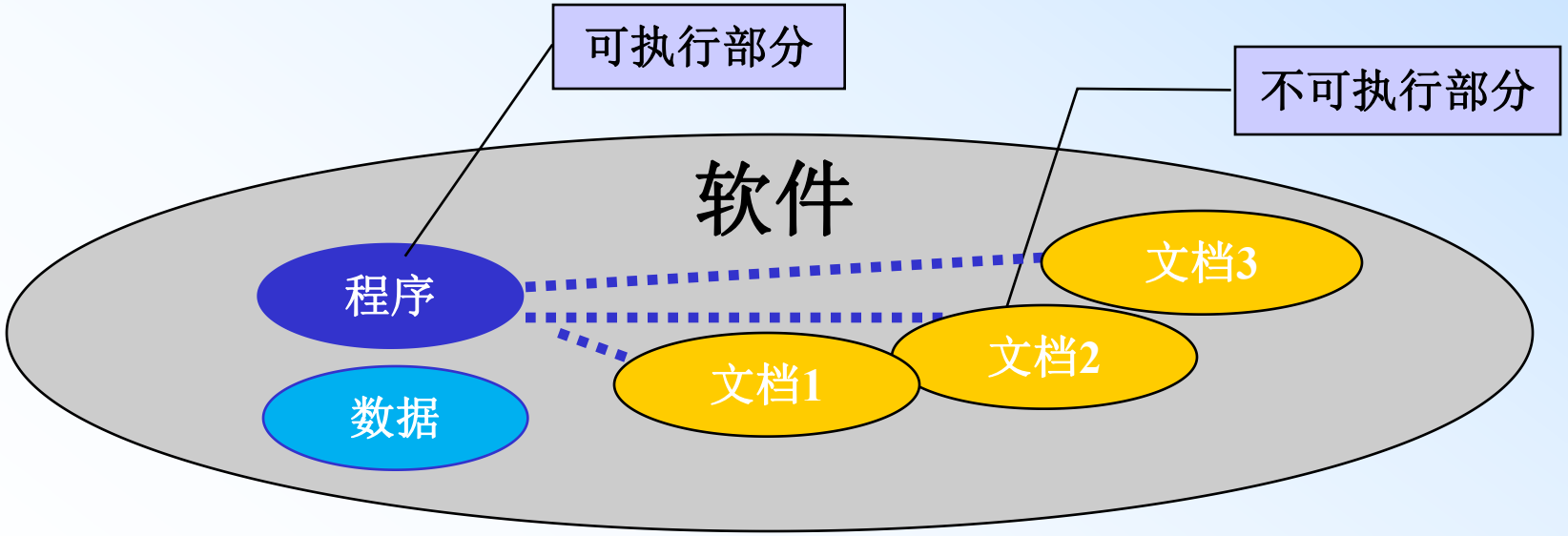
1. 掌握软件的概念和特点
2. 掌握软件工程的概念和基本特征
3. 理解为什么学习软件工程、如何学习

本章内容

1. 软件的概念、特点和分类
2. 软件危机及其原因
3. 软件工程的概念、发展、特征、研究内容和主要活动
4. 软件质量属性
5. 软件工程的基本原则和Wasserman规范
6. 软件工程的主要方法
7. 软件工程知识体系
8. 一些对软件工程的误解

什么是软件

- 软件 = 程序 + 数据 + 文档
  - Software = Program + Data + Document
  - 程序：按事先设计的功能和性能需求执行的指令序列
  - 数据：是程序能正常操纵信息的数据结构
  - 文档：与程序开发、维护和使用有关的图文材料



软件的特点

① 软件是无形的

② 软件是开发的，不是制造的，充满个人行为和个人因素

③ 软件生产是简单的拷贝（盗版问题）

④ 软件需求在软件开发之初往往不明确，会多次修改

⑤ 软件需要维护，软件维护易产生新的问题

⑥ 软件测试非常困难（穷举测试不可能）

⑦ 软件开发和运行都离不开相关计算机系统环境，软件开发环境对产品影响较大

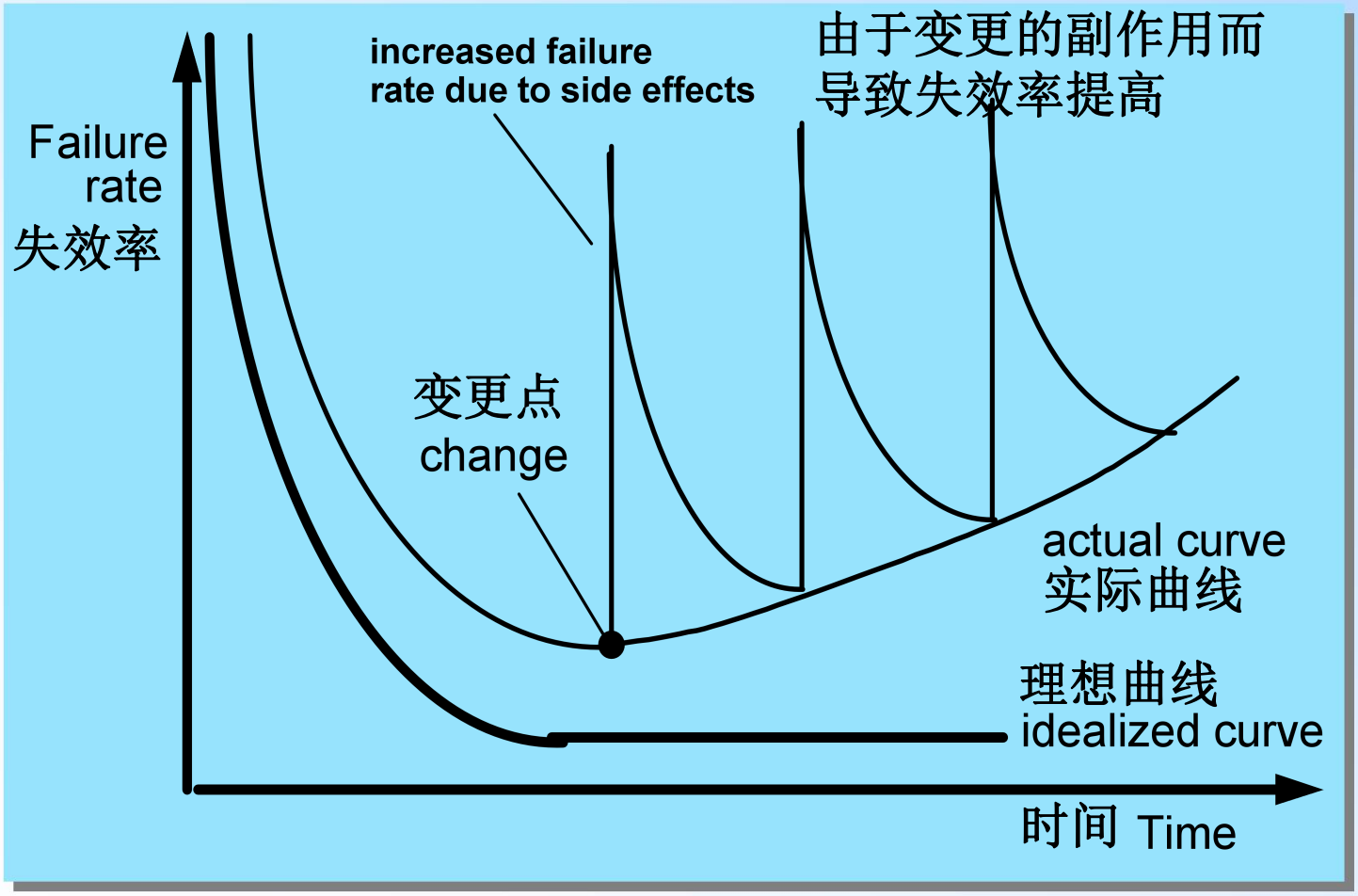
⑧ 软件开发时间和工作量难以估计

⑨ 软件的开发进度几乎没有客观衡量标准

⑩ 虽然整个工业向着基于构件的构造模式发展，然而大多数软件仍是根据实际的顾客需求定制的

⑪ 软件不会磨损和老化，但会退化和废弃（由于软件变更）

软件失效曲线


increased failure
rate due to side effects
由于变更的副作用而
导致失效率提高
Failure
rate
失效率
变更点
change
actual curve
实际曲线
理想曲线
idealized curve
时间 Time

软件的双重作用

- 一方面是一种产品
  - 提供计算能力
  - 产生、管理、获取、修改、显示或传输信息
- 另一方面是开发其它软件产品的工具
  - 控制其它程序（如操作系统）
  - 帮助开发其它软件（如软件开发工具）
  - 支持或直接提供系统所需的功能（如数据库管理系统）
  - 改善通信（如网络软件）
  - 其它功能……

软件的分类

- 按软件的功能进行划分：
  - 系统软件
    - 操作系统
    - 数据库管理系统
    - 设备驱动程序
    - 通信处理程序等
  - 支撑软件
  - 应用软件

软件的分类

- 按软件的功能进行划分：
  - 系统软件
  - 支撑软件
  - 应用软件

- 文本编辑程序
- 文件格式化程序
- 磁盘的数据传输程序
- 程序库系统
- 支持需求分析、设计、实现、测试和支持管理的软件

软件的分类

- 按软件的功能进行划分：
  - 系统软件
  - 支撑软件
  - 应用软件
    - 商业数据处理软件
    - 工程与科学计算软件
    - 计算机辅助设计／制造软件
    - 系统仿真软件
    - 智能产品嵌入软件
    - 医疗、制药软件
    - 事务管理、办公自动化软件
    - 计算机辅助教学软件

软件分类-按规模划分

类别	参加人员数	研制时间	代码行数
微型	1	1~4周	0.5k
小型	1	1~6月	1k～2k
中型	2~5	1～2年	5k～50k
大型	5~20	2～3年	50k～100k
甚大型	100~1000	4～5年	~1M
极大型	2000~5000	5～10年	1M～10M

早期软件开发

- 20世纪60年代中期以前
- 软件是为每个具体应用专门编写
- 软件规模较小
- 个体化软件开发方法
- 没有文档，只有代码

- 计算机应用日益普及
- 软件数量和规模急剧膨胀
- 程序错误必须改正
- 有新的需求必须修改程序
- 硬件和操作系统的更新需要软件去适应

20世纪60年代 → **软件危机 Software crisis**

什么是软件危机

- 软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题：
  - 项目大大超出预算
  - 项目超过计划完成时间
  - 软件不符合客户要求
  - 软件质量差
  - 软件运行效率很低
  - 项目难以管理并且代码难以维护
  - 软件不能交付
  - …

软件危机的原因

- 软件本身特点
  - 软件是逻辑部件，缺乏可见性
  - 维护困难
  - 软件越来越庞大和复杂，需要分工协作
- 开发和维护方法不正确
  - 需求分析不充分或存在错误
  - 开发过程不规范，错误认为：软件开发=编程序
  - 不注重文档工作，软件难以维护
  - 缺少软件评测手段

解决软件危机

- 1968年10月北大西洋公约组织(NATO)召开的计算机科学会议上，Fritz Bauer首次提出： 软件工程！！！
- 对计算机软件有一个正确的认识：

  (软件≠程序，软件开发≠编程序)
- 认识到软件开发不是某种个体劳动的神秘技巧，而是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目
- 推广使用在实践中总结出来的开发软件的成功技术和方法
- 开发和使用更好的软件工具

软件工程的发展

- 第一代软件工程：传统的软件工程
- 第二代软件工程：对象工程
- 第三代软件工程：过程工程
- 第四代软件工程：构件工程

软件工程的发展

- 第一代软件工程—传统软件工程（结构化方法学）
  - 60年代末到70年代
  - 为克服软件危机提出
  - 将软件开发纳入工程化的轨道，基本形成软件工程的概念、框架、技术和方法。
- 第二代软件工程—对象工程（面向对象方法学）
  - 80年代中到90年代
  - 面向对象的方法与技术得到发展，研究的重点转移到面向对象的分析与设计，演化为一种完整的软件开发方法和系统的技术体系。

软件工程的发展

- 第三代软件工程 — 过程工程
  - 80年代中开始
  - 人们在软件开发的实践过程中认识到：提高软件生产率、保证软件质量的关键是“软件过程”，是软件开发和维护中的管理和支持能力，逐步形成软件过程工程。
- 第四代软件工程 — 构件工程
  - 90年代起
  - 基于构件（Component）的开发方法取得重要进展，软件系统的开发可通过使用现成的可复用构件组装完成，而无需从头开始构造，以此达到提高效率和质量，降低成本的目的。

软件危机现状

- 软件危机大大改善，仍然没有摆脱软件危机的困扰
- 软件本身特点
  - 软件缺乏可见性
  - 软件越来越庞大和复杂，需要分工协作
- 开发和维护方法问题
  - 需求分析不充分或存在错误
  - 缺少有效的软件评测手段

软件危机现状

- 对软件的需求远远超出现有的生产能力
  - 信息技术的爆炸性增长促进了对新软件的空前需求
  - 软件系统的规模和复杂性不断增长
  - 航天飞机的代码有千万行、空间站的代码行数以亿计、Windows系统的代码行数也是千万级，Linux系统的代码行数也超千万
- 构造安全可靠软件的技术能力不足
  - 过去40年，硬件性能和质量飞速发展，但开发软件的能力未能与硬件提供的机会保持同步

软件工程的定义

定义

1993年IEEE计算机协会将软件工程定义为：

（1）应用系统的、规范的、可度量的方法，来开发、运行和维护软件，即把工程应用到软件。

（2）对（1）中各种方法的研究。

目标

软件工程的目标是在给定的时间和预算内，按照用户的需求，开发易修改、高效、可靠、可维护、适应力强、可移动、可重用的软件。

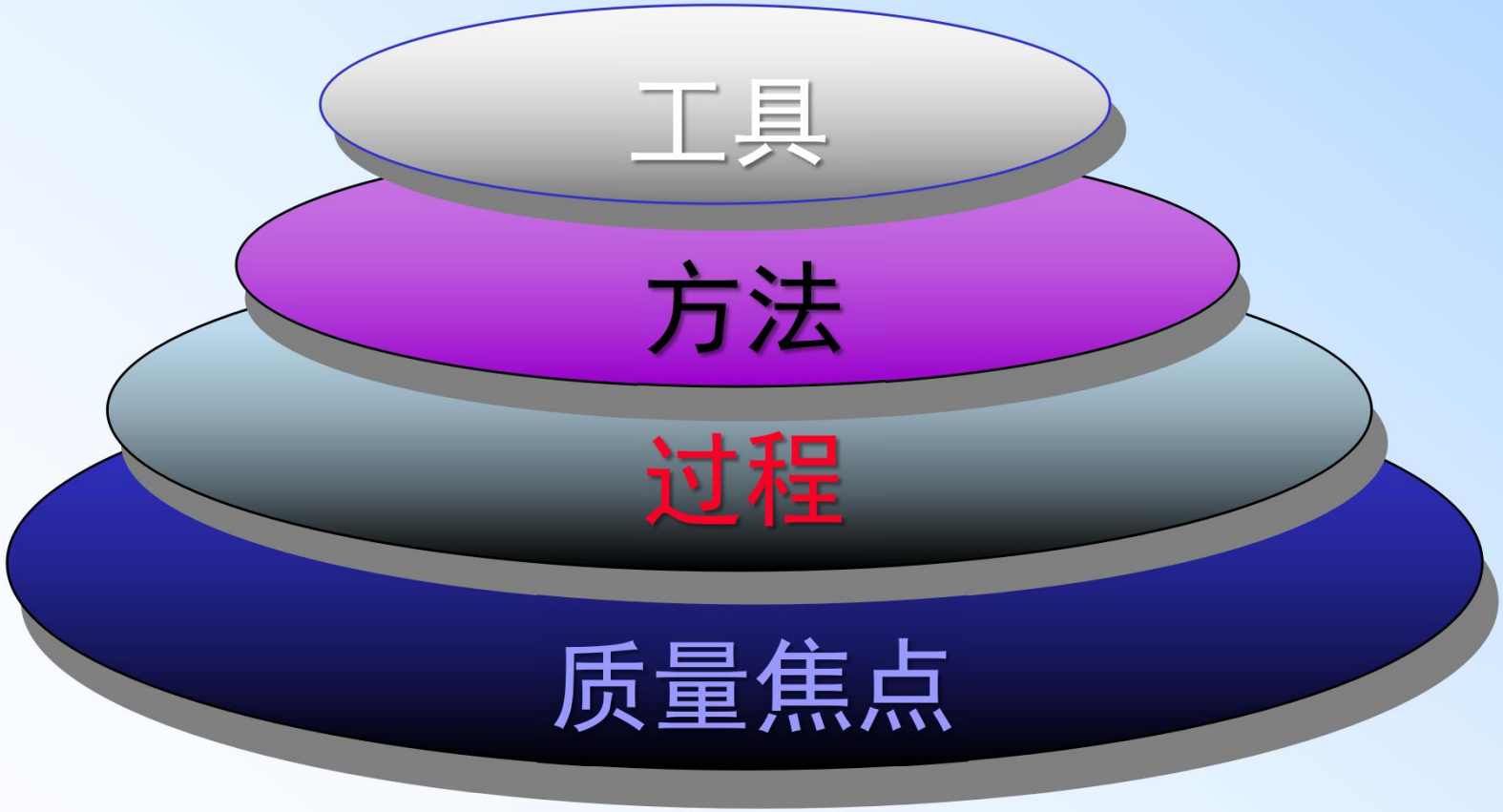
软件工程的研究内容

- 管理
  - 通过计划、组织、控制等一系列活动，合理配置和使用资源，达到既定目标。

- 技术：软件工程方法学（Methodology/Paradigm）
  - 方法（Methods）：怎么做
  - 工具（Tools）：软件工程支撑环境
  - 过程（Process）：工作步骤

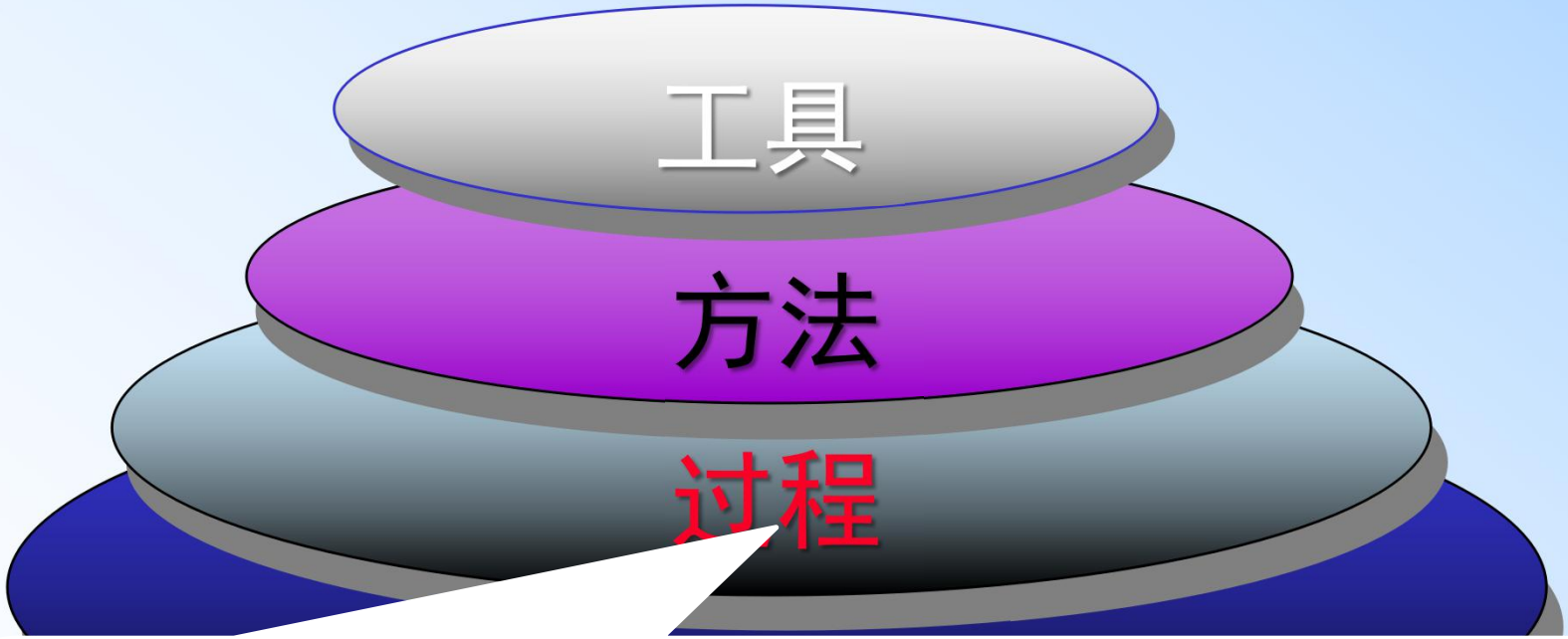
软件工程三要素

软件工程：一种层次化技术



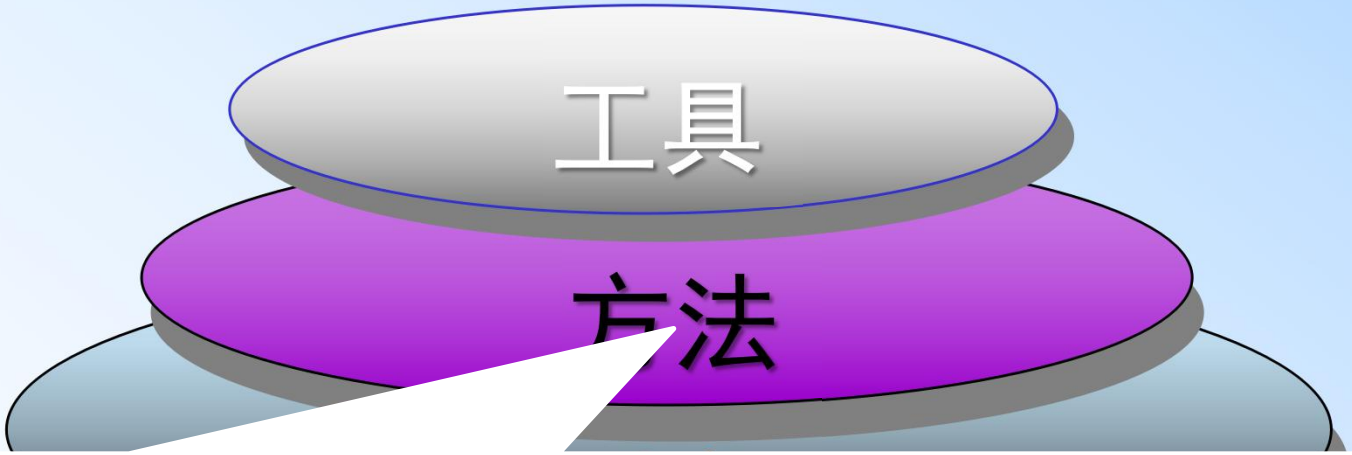
软件工程层次图

软件工程：一种层次化技术



过程贯穿软件开发的各个环节，在各环节之间建立里程碑； 管理者在软件工程过程中对软件开发的质量、进度、成本进行评估、管理和控制； 技术人员采用相应的方法和工具生成软件工程产品（模型、文档、数据、报告、表格等）。

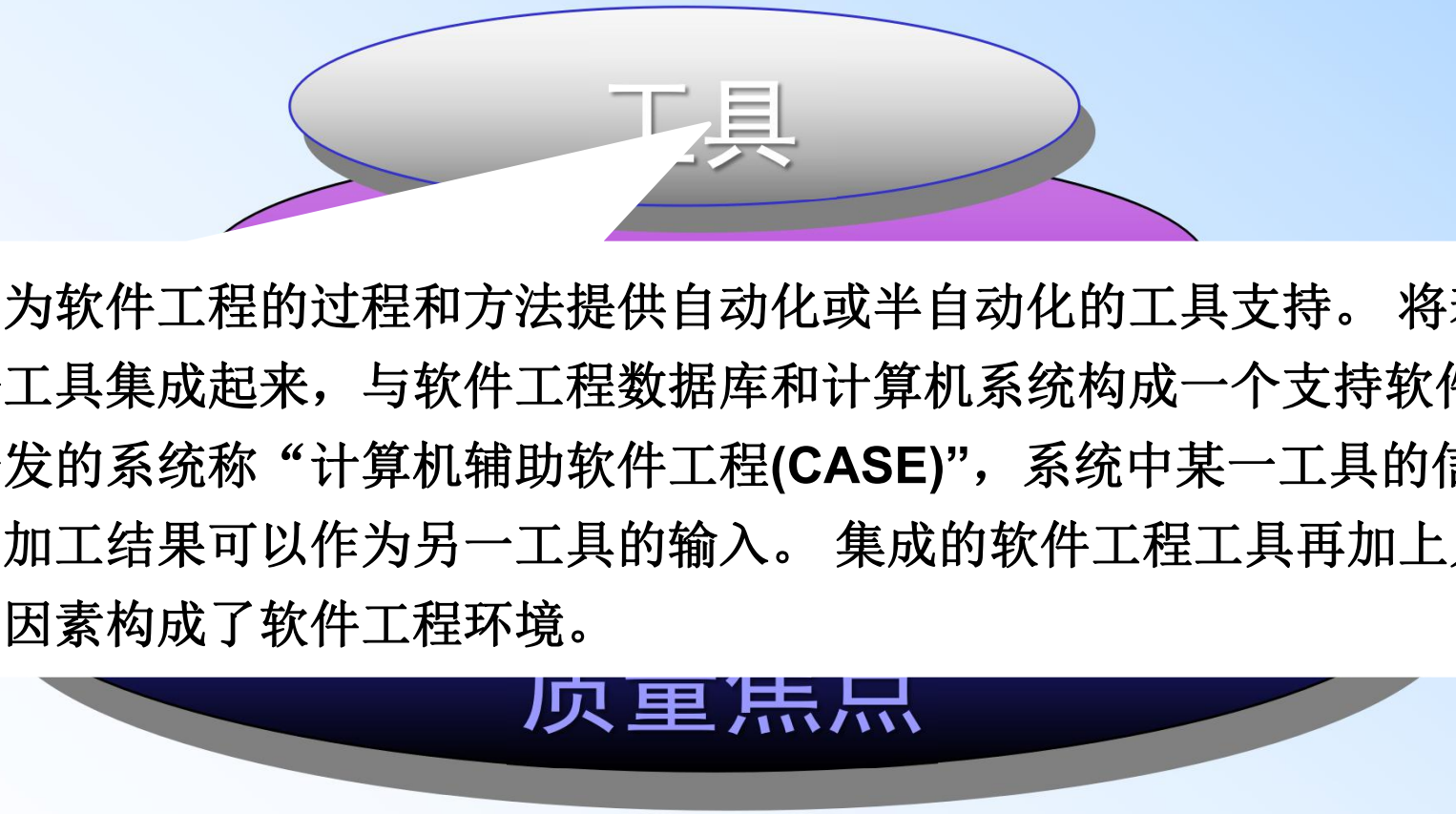
软件工程：一种层次化技术



软件工程方法是完成软件工程项目的技术手段。它支持项目计划和估算、系统和软件需求分析、设计、编程、测试和维护。软件工程方法依赖一组原则，它贯穿软件工程的各个环节。软件工程方法分两类：结构化方法和面向对象方法。

软件工程层次图

软件工程：一种层次化技术



它为软件工程的过程和方法提供自动化或半自动化的工具支持。 将若干工具集成起来，与软件工程数据库和计算机系统构成一个支持软件开发的系统称“计算机辅助软件工程(CASE)”，系统中某一工具的信息加工结果可以作为另一工具的输入。 集成的软件工程工具再加上人的因素构成了软件工程环境。

软件工程层次图

软件的质量属性

- 正确性/Correctness：满足需求规格和用户目标的程度
- 可靠性/Reliability：一定时间内无故障运行的能力
- 有效性/Effectiveness：用户完成特定任务和达到特定目标时所具有的正确和完整程度
- 可用性/Availability：用户能否用软件完成他的任务，效率如何，主观感受怎样
- 可维护性/Maintainability ：为修改错误、增加功能、提高质量而诊断并修改软件的难易程度
- 可测试性/Testability ：对软件进行测试的难易程度
- 灵活性/Flexibility：反映软件适应变化的能力，修改或改进一个已投入运行的软件所需的工作量

软件的质量属性

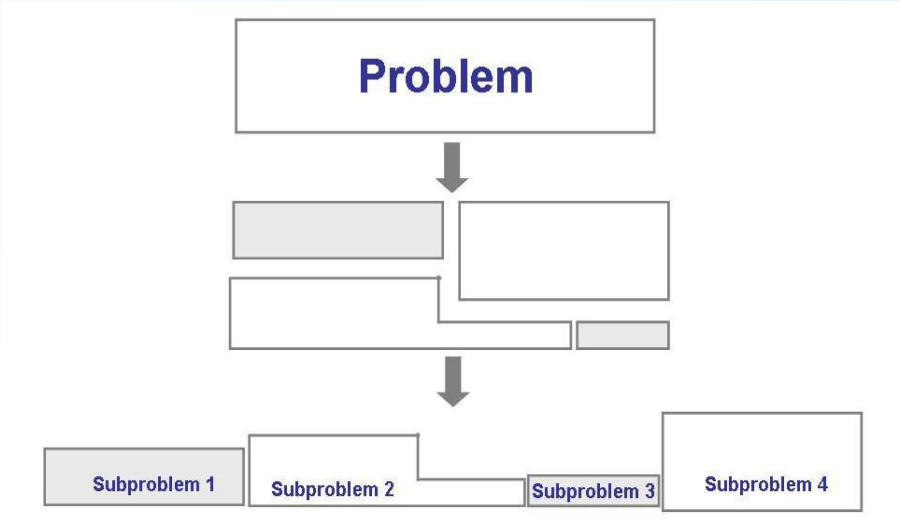
- 可移植性/Portability ：软件不经修改或稍加修改就可以运行于不同软硬件环境的难易程度
- 可重用性/Reusability：重用软件或构件的难易程度
- 互操作性/Interoperability ：本软件与其它系统交换数据和相互调用服务用以协同运作的难易程度
- 安全性/Security：向合法用户提供服务，阻止非授权使用服务
- 安全性/防危性/Safety：不发生重大事故的能力
- 健壮性/Robustness：异常情况下软件能够正常运行的能力
- 易用性/Usability：用户使用软件的容易程度

软件工程的特征

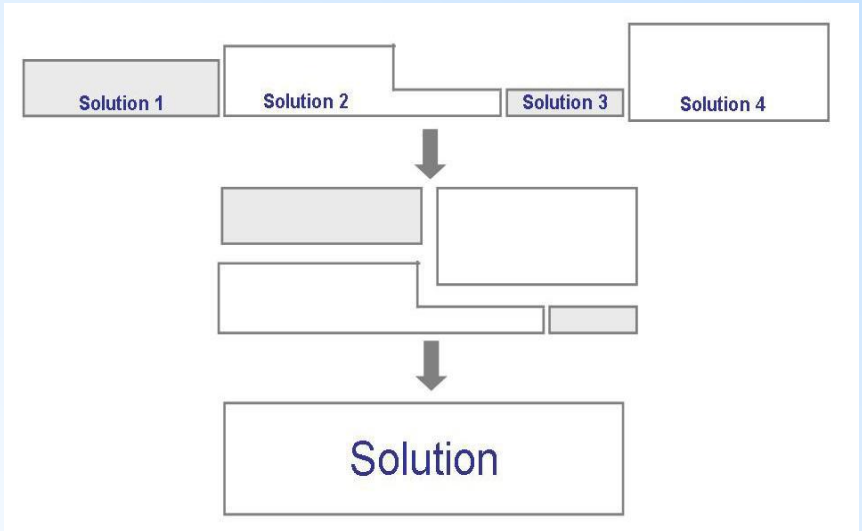
① 软件工程关注于较大型软件的构造

② 软件工程的中心课题是控制复杂性（分而治之）

分析



合成



软件工程的特征

③ 软件经常变化（可维护）

④ 开发软件的效率非常重要

⑤ 和谐的合作是开发软件的关键（人员管理）

⑥ 软件必须有效地支持它的用户

- 功能要求、性能要求、质量要求
- 用户手册、培训

⑦ 在软件工程领域中，通常由具有一种文化背景的人替具有另一种文化背景的人创造产品

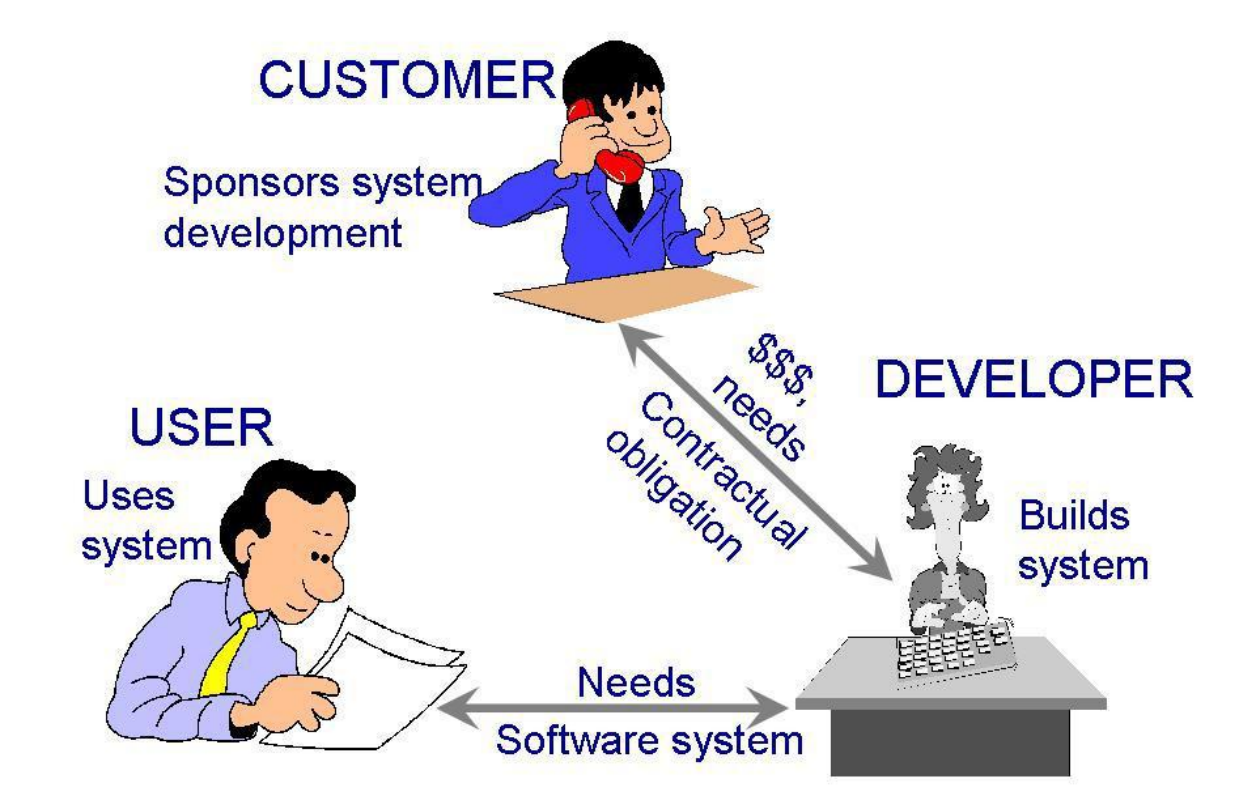
软件开发包含的活动

- 需求分析
- 总体设计
- 详细设计
- 程序实现
- 单元测试
- 集成测试
- 系统测试
- 系统交付
- 维护

软件工程涉及的人员

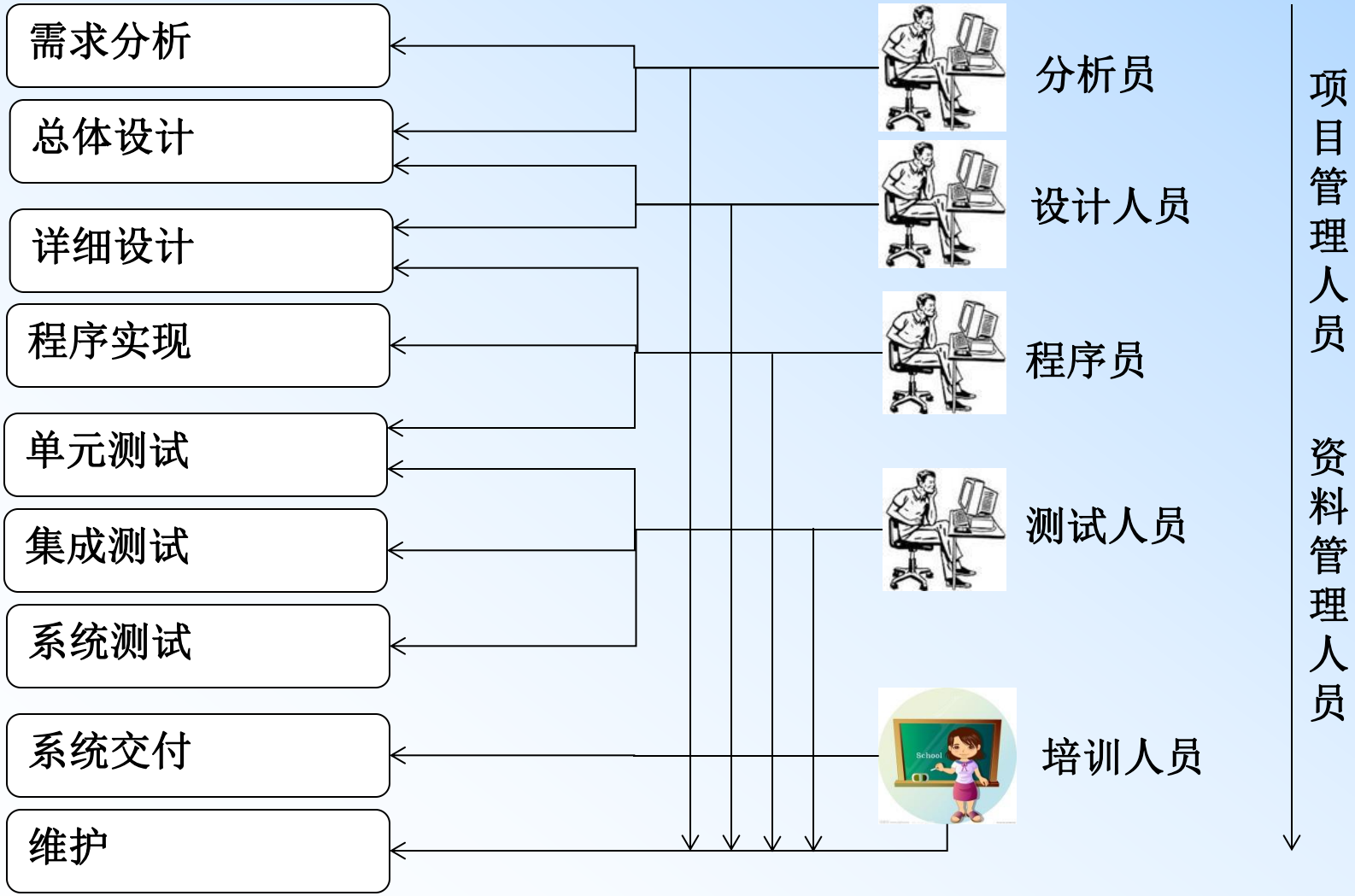
客户

用户



开发人员

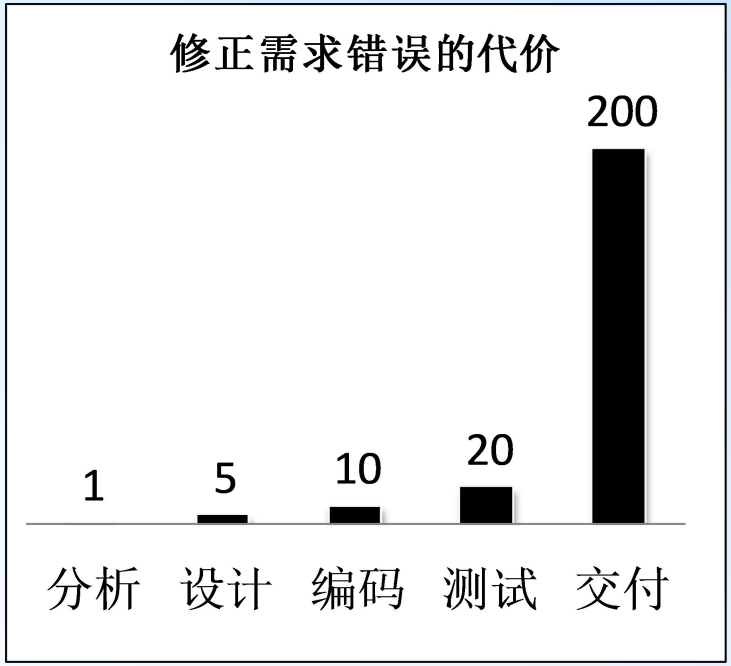
软件工程涉及的开发人员



软件工程的基本原则

- B. W. Boehm于1983年总结了开发软件的经验，提出了软件工程的7条基本原则
- 被认为是确保软件产品质量和开发效率原则的最小集合，又相互独立、缺一不可、相当完备

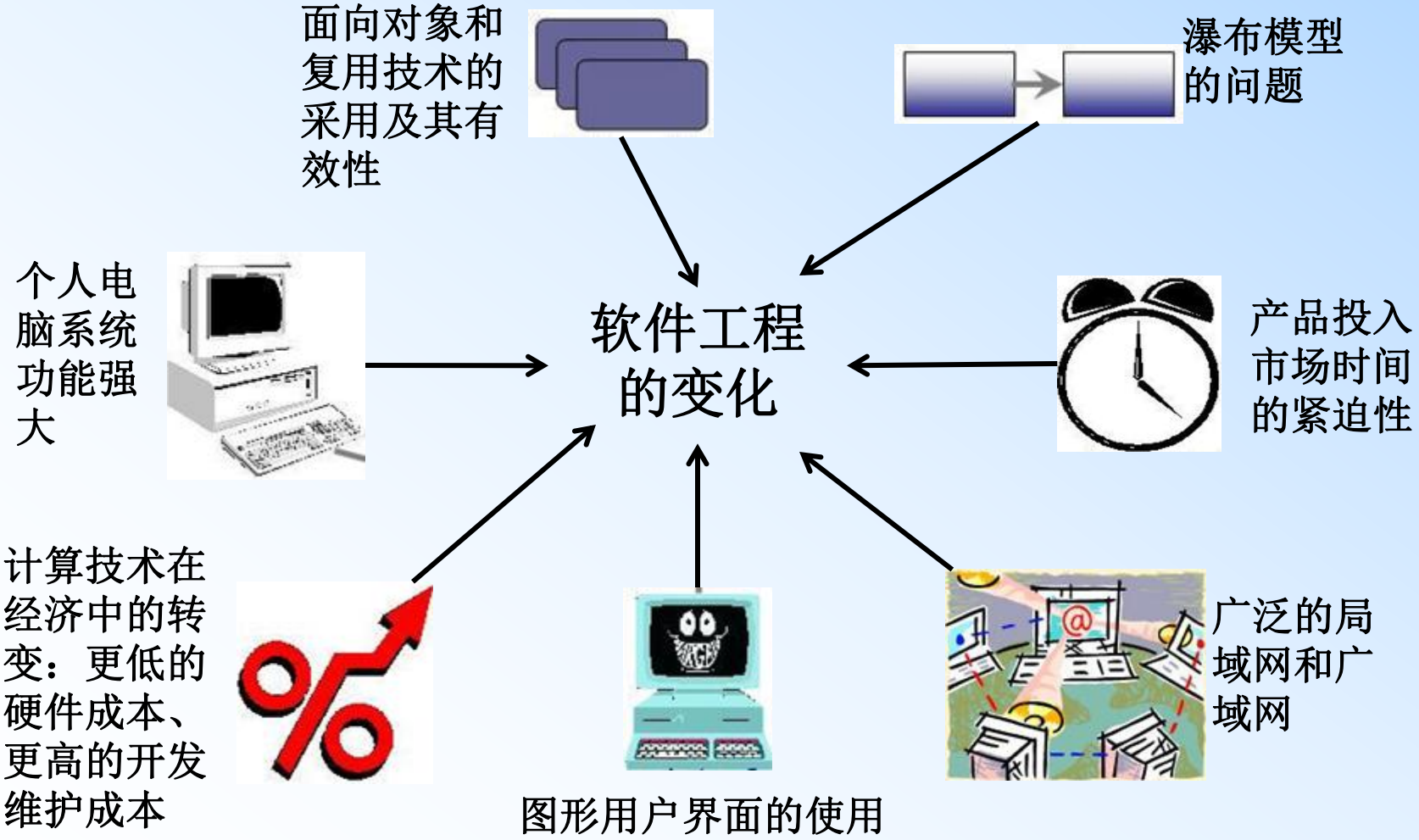
1. 按软件生存周期分阶段制定计划并认真实施
2. 坚持进行阶段审查



软件工程基本原则

3. 实行严格的产品控制
   - 识别影响软件质量的因素，并加以控制
   - 如需求变更
4. 采用现代程序设计技术
   - 提高软件生产率和质量
5. 明确责任，结果应能清楚的审查
6. 开发小组的人员应该少而精
7. 不断改进软件工程实践

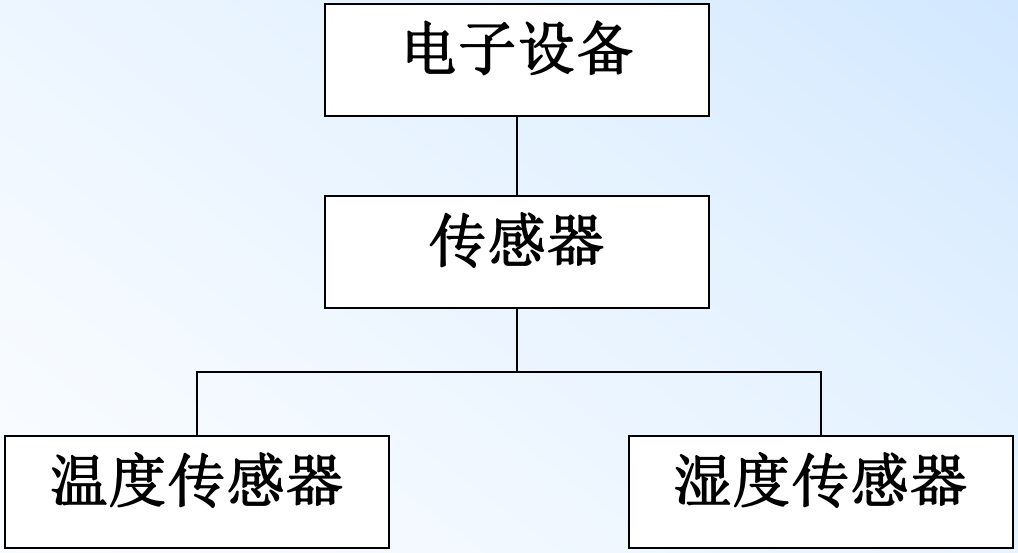
改变现代软件开发的关键因素



应对现代软件工程变化的Wasserman规范

1. 抽象（Abstraction）

在某种概括层次上对问题的描述，只关注问题的关键部分，而忽略细节。



应对现代软件工程变化的Wasserman规范

2. 分析和设计方法以及表示法
   - Analysis and design methods and notations
   - 便于审查、交流、复用
3. 用户界面原型化
   - User interface prototyping
   - 原型（Prototype）：具有部分功能的小系统
   - 帮助用户和客户标识和明确系统的关键需求
   - 证明设计或算法的可行性
   - 迭代过程

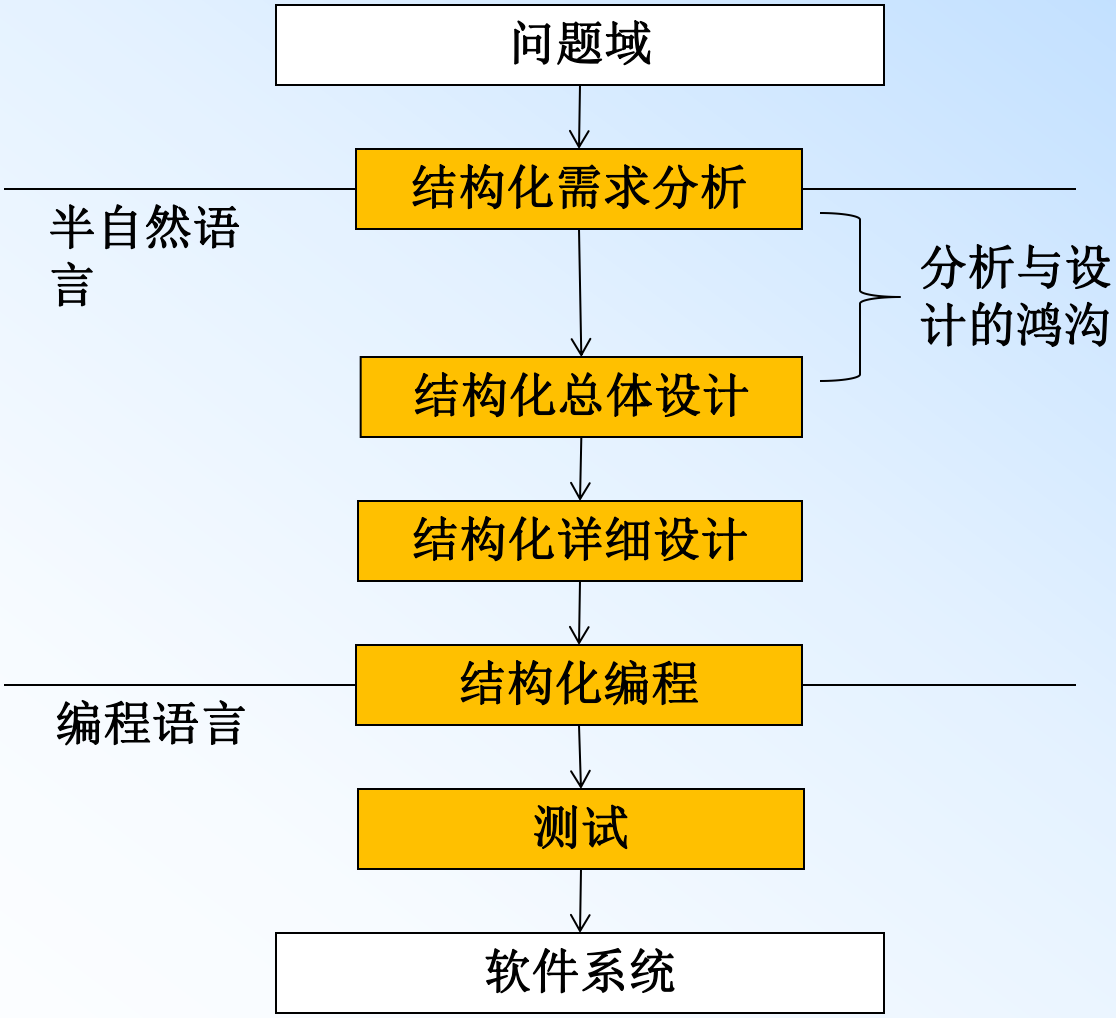
应对现代软件工程变化的Wasserman规范

4. 软件体系结构（Software architecture）
   - 系统的模块/组件，以及它们之间的关系
   - 关系到实现、测试、维护的方便性以及系统质量
5. 软件过程（Process）：不同软件需要不同过程
6. 复用（Reuse）：复用以前的代码、设计、测试等
7. 测度（Measurement)：量化的指标
   - 工作进展、成本、工作量、代码质量、测试覆盖率
8. 工具和集成环境
   - CASE：Computer Aided Software Engineering

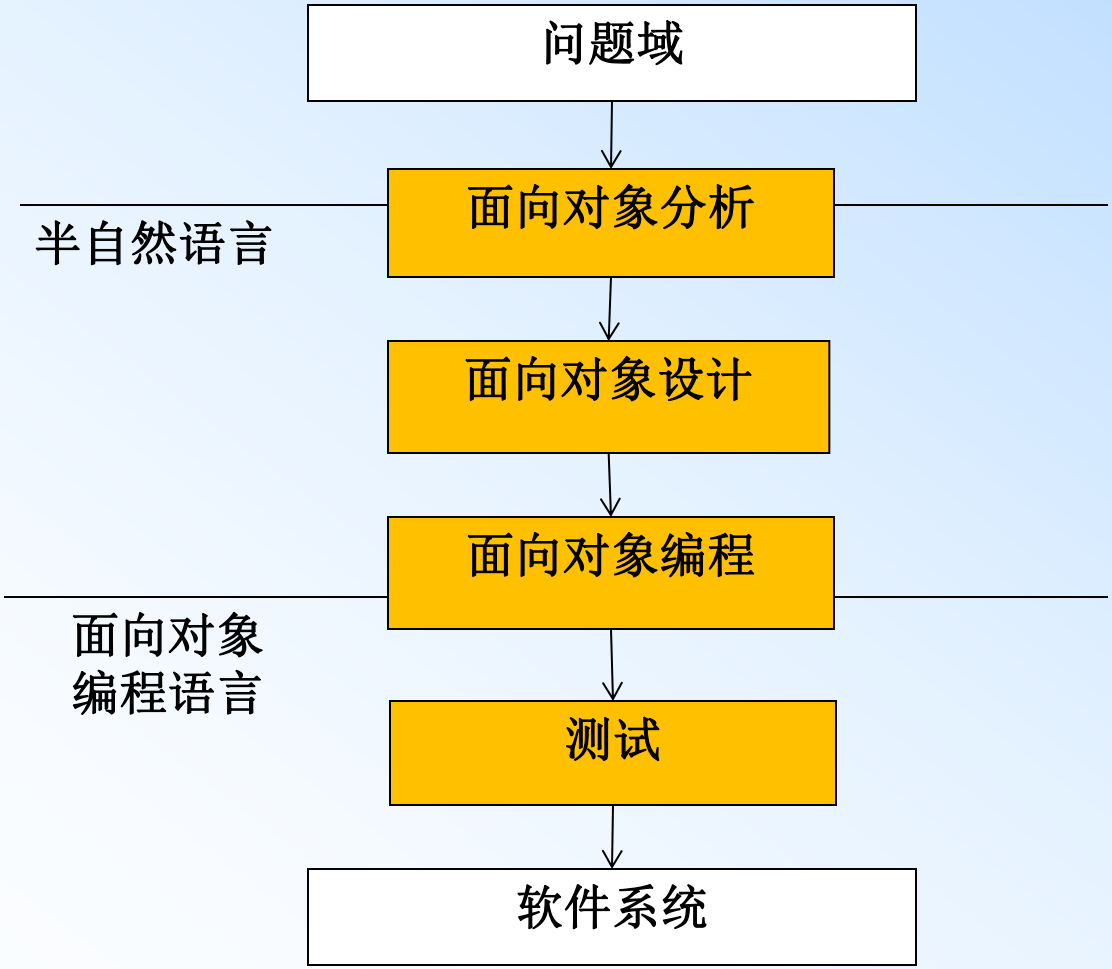
软件工程方法学

- 结构化方法学/传统方法学
- 面向对象方法学

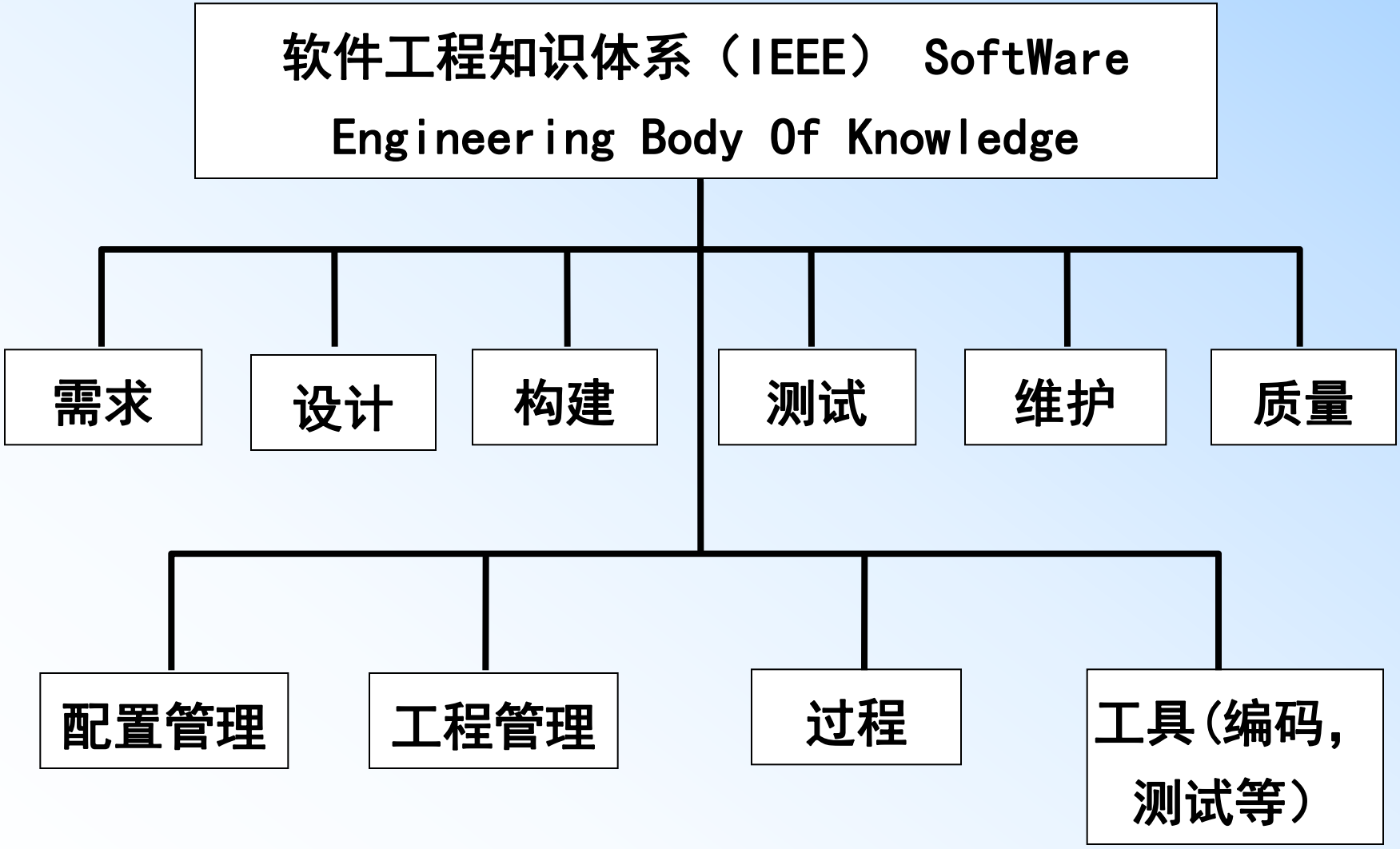
结构化方法学/传统方法学



面向对象方法学



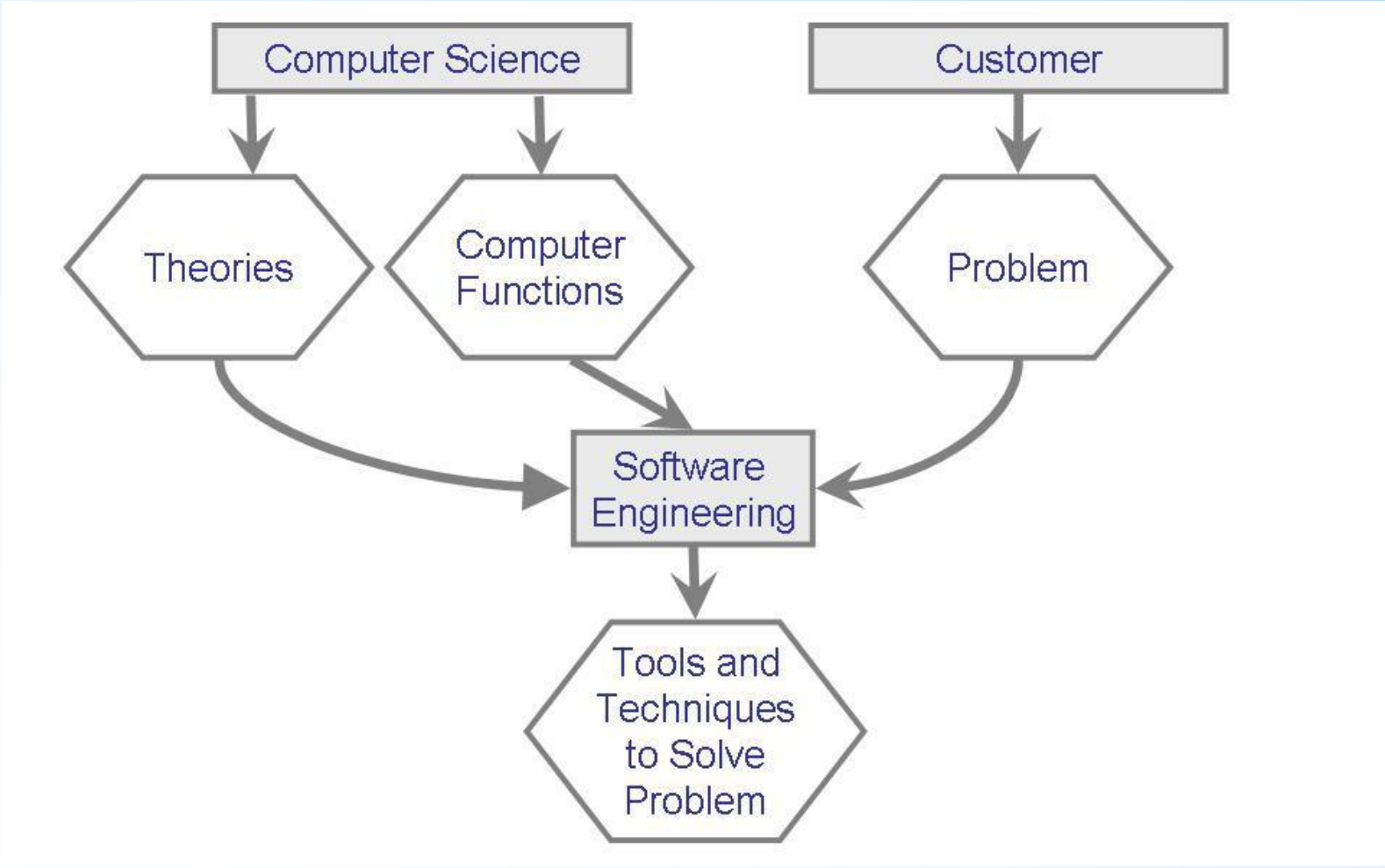
软件工程知识体系


软件工程知识体系（IEEE） SoftWare Engineering Body Of Knowledge
需求
设计
构建
测试
维护
质量
配置管理
工程管理
过程
工具(编码，测试等)

软件工程与计算机科学的差别

	软件工程	计算机科学
目标	在时间、资源、人员这3个主要限制条件下构建满足用户需求的软件系统	探索正确的计算和建模方法，从而改进计算方法本身
产品	软件（比如办公包和编译器）	算法（比如希尔排序法）和抽象的问题（比如哲学家进餐问题）
进度与时间表	软件项目都有特定的进度与时间表	研究项目一般不具有设置的进度与时间表
关注点	软件工程关注如何为用户实现价值	软件理论关注的是软件本身运行的原理，比如时间复杂度、空间复杂度、算法的正确性
变化程度	随着技术和用户需求的不断变化，软件开发人员必须时刻调整自己的开发以适应当前的需求。同时软件工程本身也处于不断的发展中	对于某一种特定问题的正确解决方法将永远不会改变
需要的其他知识	相关领域的知识	数学

软件工程与计算机科学的差别



一些对软件工程的误解（1）

- M代表误解，R代表现实
- 管理方面的误解
  - M1：如果我们项目进度落后了，可以加入更多的程序员来赶进度。
  - R1：软件开发的机制和手工作业不一样。在一个延迟了的软件项目中加入新的开发人员只会让它延迟更多。
  - M2：如果我们将软件项目外包给第三方，我们就轻松了，让那个公司去完成它吧。
  - R2：如果组织管理方不懂得如何从内部管理和控制软件项目，即使将项目外包也无济于事。

一些对软件工程的误解（2）

- 客户方面的误解
  - M1：对目标的一般陈述就足以开始编程，我们可以今后再补充细节。
  - R1：前期糟糕的项目需求定义，是导致软件失败的主要原因。
  - M2：虽然软件需求不断变更，但是因为软件是弹性的，因此可以很容易的适应变更
  - R2：项目需求的确在不断变化，但变化所产生的影响是根据变化提出的时间不同而不同的。

一些对软件工程的误解（3）

- 开发者方面的误解
  - M1：一旦我们编程完毕并成功运行，我们的工作就结束了。
  - R1：工业数据显示，软件开发60%-80%的精力将耗费在软件首次提交给用户以后。
  - M2：当我的程序运行之前，我没有办法评估它的质量。
  - R2：一个最有效的软件质量保证机制—技术评审，应当在项目的正式开始启动时就开始实行。

一些对软件工程的误解（4）

- 开发者方面的误解
  - M1：唯一可交付的工作成果是一个成功运行的项目程序
  - R1：一个可运行的程序只是软件结构的一部分，它还包含了许多其它因素。
  - M2：软件工程将会让我们去创建大量不必要的文档，并且总是使我们的进度放慢。软件工程仅仅是文档而已。
  - R2：软件工程并不是创建文档，而是创建质量。更好的质量减少返工的概率。更少返工会让项目更早交付。所有的文档都是提高团队沟通和质量所必须的。

软件工程职业道德和责任规范

- **诚信**：工程师们应当对自己的雇主和顾客时刻保持诚信而无论之前是否达成了关于诚信的协议。
- **能力**：工程师们不应该虚夸自己的能力水平，不应该故意接受一份超出自己能力范围的工作。
- **知识产权**：工程师们应该了解当地的知识产权法律法规，如专利权、版权等，应该小心确保雇主和客户的知识产权受到了保护。
- **滥用计算机**：软件工程师不以他们的工作职责为由滥用别人的电脑。滥用计算机的范围很广，从极小（在雇主的机器上玩游戏）到极其严重的（传播病毒）。

本章复习要点

1. 软件的概念、特点和分类
2. 软件危机及其原因
3. 软件工程的概念、发展、特征、研究内容和主要活动
4. 软件工程的7项基本原则
5. 应对现代软件工程变化的Wasserman规范：8个概念
6. 软件工程的主要方法
   - 结构化方法
   - 面向对象方法