哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

软件工程

第一章 软件工程概论

1-1 软件工程概论

徐汉川
xhc@hit.edu.cn

2017年9月4日

软件工程概论

- **1.1 软件的基本概念**
  - 1.1.1 什么是软件
  - 1.1.2 软件的发展
- **1.2 软件工程的基本概念**
  - 1.2.1 软件工程产生的历史根源
  - 1.2.2 软件工程的基本概念
  - 1.2.3 软件工程的知识体系

一个例子

- 移山公司程序员二柱的小孩上了小学二年级，老师让家长每天出30道四则运算题目给小学生做。
- 二柱立马就想到写一个小程序来做这件事。 这个事情可以用很多语言或者工具来实现:
  - C/C++, C#, Unix Shell, Emacs, Java, Perl, Python, …
- 程序员用自己最擅长的工具，很短的时间就可以搞定。

From 《现代软件工程讲义》

一个例子

- 老师看了作业之后，对二柱赞许有加。别的老师闻讯，问二柱能否扩大他的影响力，编个软件给二年级到四年级都用，多了一些小小的要求:
  - 题目避免重复
  - 可定制（数量/打印方式）
  - 可以控制下列参数:
    - 是否有乘除法、是否有括号
    - 数值范围、加减有无负数
    - 除法有无余数、是否支持分数 (真分数, 假分数, …)
    - 是否支持小数 (精确到多少位)、打印中每行的间隔可调整
- 小同学兴高采烈☺地拿着需求回来了，跟老爸说：“老师明天就想要！”
- 二柱：……

From 《现代软件工程讲义》

一个例子

- 假设二柱熬夜做出了这个软件的一个初始版本，交给老师了。
- 过几天老师又问：能否把这个程序放到学校的网站上去，再多一点点小要求：能够支持二元一次方程，能开根号，并且让所有人可以通过网页订制各种类型的四则运算作业。
- 二柱：…… ☹ ！

- 给自己儿子用的：小程序
- 给全校学生用的：软件
- 放在网站上的：服务

规模越来越大，需求越来越复杂；
难度越来越大，所需的时间越来越长，
出错的概率也越来越大。

From 《现代软件工程讲义》

什么是“软件”？

- **软件(Software)：一组对象或项目所形成的一个“配置”，由程序、文档和数据等部分构成。**
  - 程序(program)：可被计算机硬件理解并执行的一组指令，提供期望的功能和性能；
  - 数据(data structure)：程序能正常操纵信息的数据结构；
  - 文档(document)：与程序开发、维护和使用有关的图文材料

- **程序(Program)=软件(Software)?**

独唱→小合唱→合唱→万人大合唱

| | | |

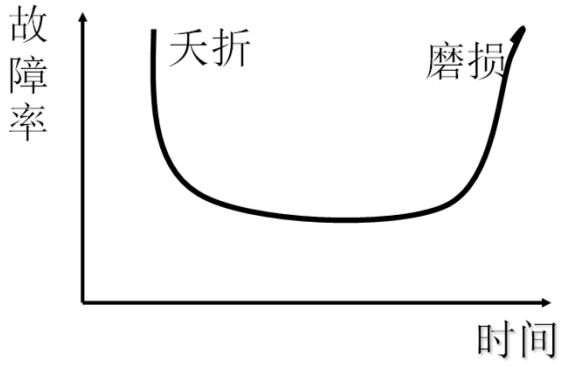
简单程序　较复杂程序　　软件

软件的四大特征

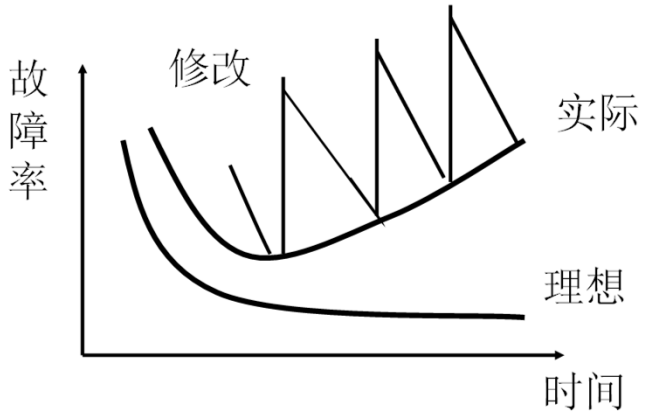
- **复杂性(complexity)**
  - 软件要解决的现实问题通常很复杂，数据、状态、逻辑关系的可能组合导致了软件本身的复杂性；
  - 软件无法以“制造”的方式被生产，只能采用手工开发方式，这是一种人为、抽象化的智能活动(智力密集型)，与人的水平密切相关，人类思维的不确定性导致了开发过程的复杂性；
- **不可见性(invisibility)**
  - 尚未完成的软件是看不见的，无法像产品一样充分呈现其结构，使得人们在沟通上面临极大的困难，难以精确的刻画和度量。
- **易变性(changeability)**
  - 软件所应用的环境由人群、法规、硬件设备、应用领域等因素汇集而成，而这些因素皆会频繁快速的变化。
- **一致性(conformity)**
  - 各子系统的接口必须协同一致，而随着时间和环境的演变，要维持这样的一致性通常十分困难。

软件为何需要不断的变化？

- “变化”是永恒的主题：
  - 软件必须不断的变化以适应新的计算环境或新技术的发展；
  - 软件必须通过不断的功能增强以实现新的业务需求；
  - 软件必须通过扩展以与其他软件系统进行互操作；
  - 软件必须被不断的重构以使其生命周期得以延续；



硬件故障曲线



软件故障曲线

软件不会磨损和老化，但维护困难

软件为何需要不断的变化？

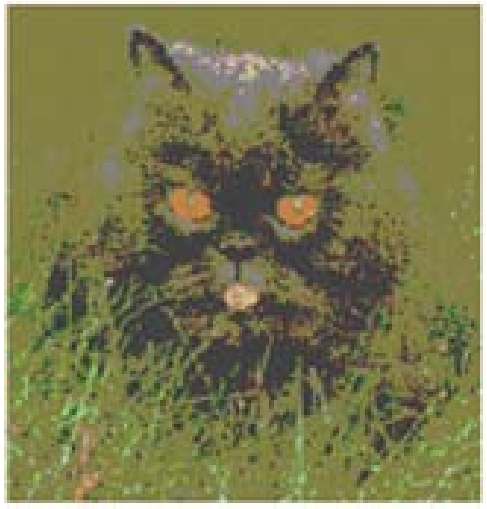
- 遗留系统(Legacy system)：仍在使用中的软件系统，可满足客户需求，但很难以“优雅的”方式对其进行演变以适应新需求或新环境；

- 60%的软件维护费用用于向遗留系统增加新功能，17%用来修正遗留系统中的bug。

软件为何无法被“制造”？

- **土木工程师：**
  - 当一个土木工程师去修建一座跨河大桥来连接河两边的道路时，工程师会非常清楚的知道道路跨河的精确地理坐标位置。行驶的车辆在数年里也不会发生重大的改变。桥梁工程师只需要按照之前已经被上千次的验证过的建筑工艺把河两边的路连接到一起。
- **软件工程师：**
  - 对于软件系统，因为技术或业务发生了变化，在建设过程中(所有需求和设计完成后)，需求需要做重大修改的情况并不罕见。如果把这种情况放到修桥的事情上，相当于当桥的地基打好后，再把桥的搭建位置移动。
- **软件工程领域的很多“奇特”现象：**
  - 程序员的“夜猫子”习惯；
  - 结对编程；
  - 持续交付；
  - …

软件的特性–小幽默



Mechanical Engineering is like looking for a cat in a lighted room.

Chemical Engineering is like looking for a black cat in a dark room.

Software Engineering is like looking for a black cat in a dark room in which there is no cat.

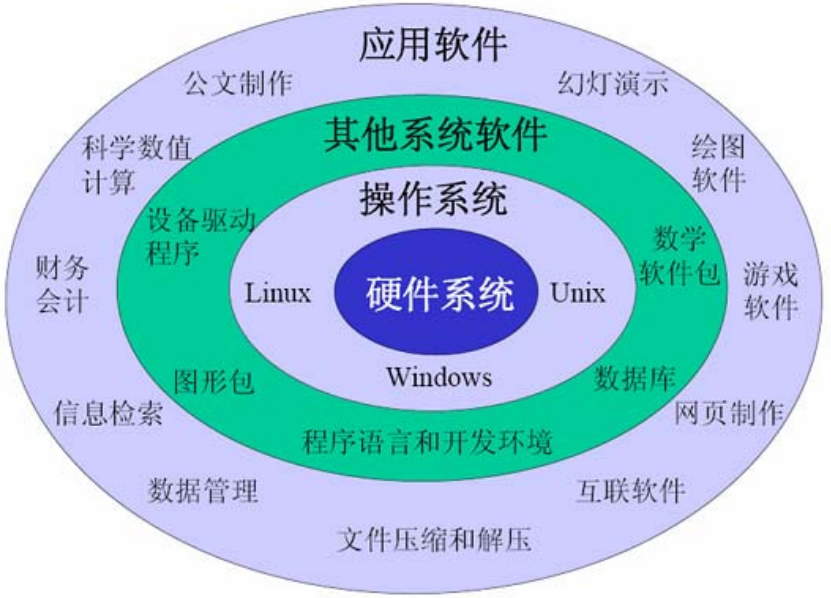
System Engineering is like looking for a black cat in a dark room in which there is no cat and some one yells, “I got it”.

软件的特性

- 不可视性与主观性
- 软件规模与复杂性
- 易变性与不确定性
- 精确性与模糊性

软件的分类

- 系统软件
  - 操作系统
  - 编译器
  - 数据库/DBMS
  - 集成开发环境(IDE)
- 应用软件
  - 商业软件：面向企业/政府
  - 个人软件：面向个人生活
  - 工程和科学计算软件
- 从开发方式上：
  - 商业软件
  - 开源软件
- 从存在形式上：
  - 嵌入式软件/单机软件/分布式软件
  - 移动终端软件
  - 基于Web的软件
  - 服务形态的软件(SaaS)

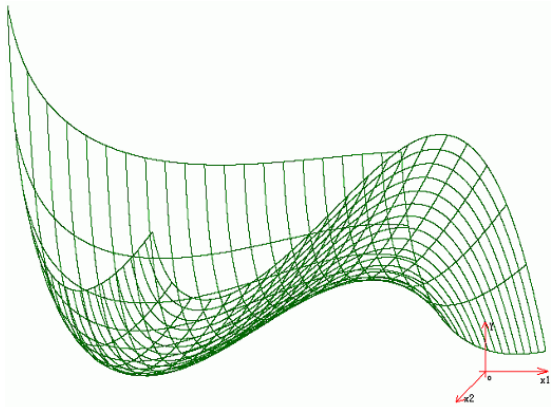


第1章 软件工程概论

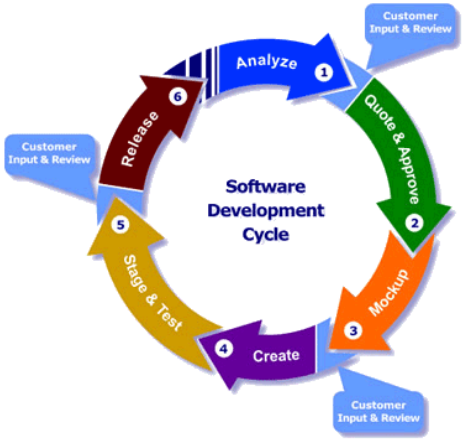
- **1.1 软件的基本概念**
  - 1.1.1 什么是软件
  - 1.1.2 软件的发展
- **1.2 软件工程的基本概念**
  - 1.2.1 软件工程产生的历史根源
  - 1.2.2 软件工程的基本概念
  - 1.2.3 软件工程的知识体系

软件及其开发方式的发展

- **第一阶段(1950-1960年代)：**
  - 软件主要用于数值计算的需求；
  - 程序设计被认为是一种任人发挥创造才能的活动，程序的质量完全依赖于程序员的个人才能；
  - 软件从“简单→复杂”的发展，导致“软件危机”的出现；

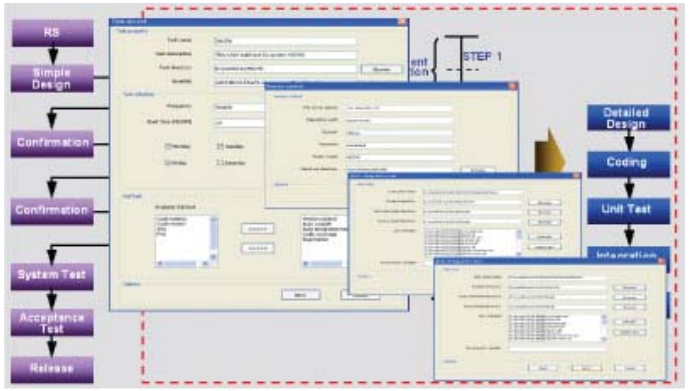


- **第二阶段(1970年代)：**
  - 软件开始向商务领域推广，出现了数据库、结构化编程等技术；
  - 软件不仅是程序，还包括开发、使用、维护等文档，“软件生命周期”的概念开始形成；

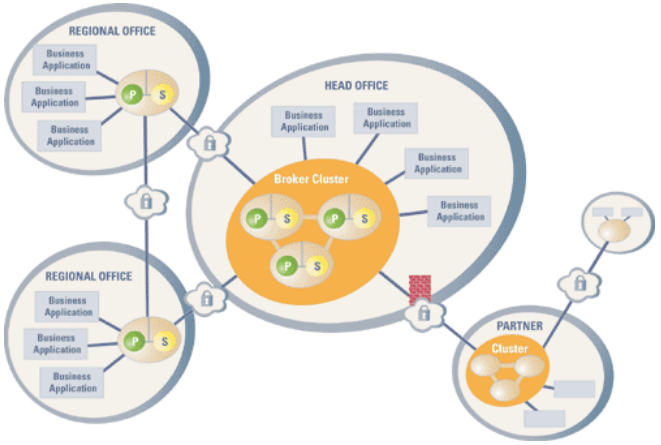


软件及其开发方式的发展

- **第三阶段(1980年代)：**
  - 软件系统的规模、复杂度进一步扩大；
  - 开始关注对软件开发过程的管理及工程性的开发；
  - 出现了CASE工具；
  - 开始关注软件的质量度量和管理；
  - 面向对象思想OO开始出现；

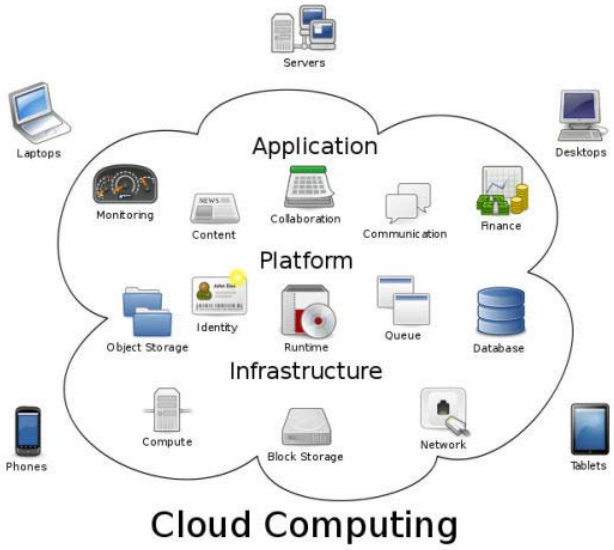


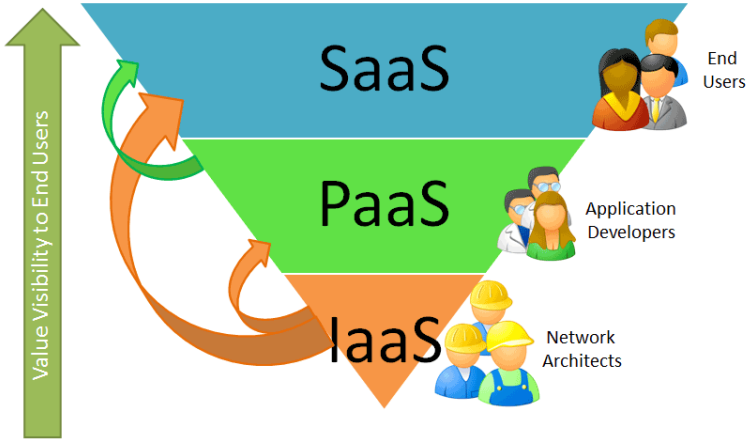
- **第四阶段(1990年代)：**
  - Internet和Web→分布式、异构环境下的软件；
  - 软件复用成为关注点；
  - 软件生命周期的每一个阶段都发展出详细的方法论；
  - 分布式计算、网格技术；



软件及其开发方式的发展

- 第五阶段(2000年代-至今)：
  - 软件的服务化、系统互操作的需求
  - 基于云计算平台的软件体系结构、模型驱动的开发方法MDA、敏捷软件开发方法、软件集成开发环境及工具
  - 面向服务的体系架构SOA方法
  - 基于互联网与云计算的软件开发方法
  - 普适计算、移动计算





软件及其开发方式的发展

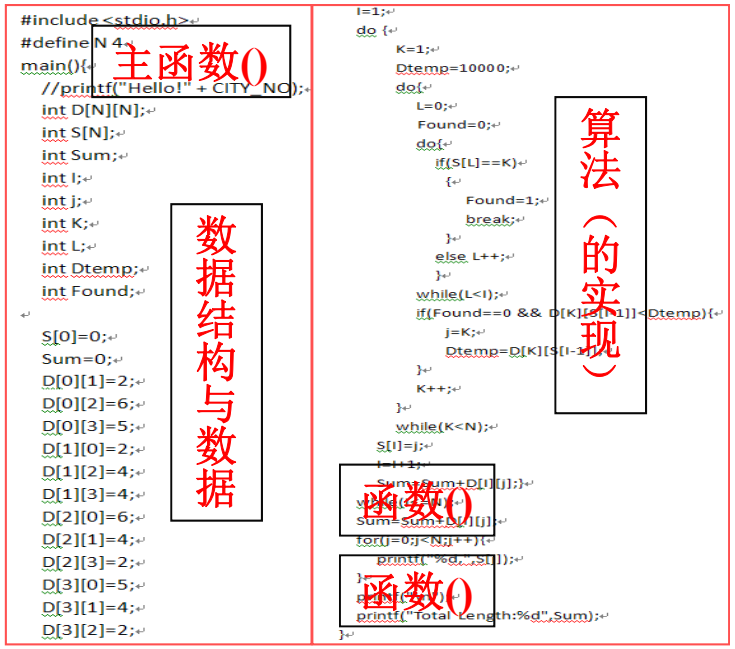
1960's-70's年代: 结构化方法

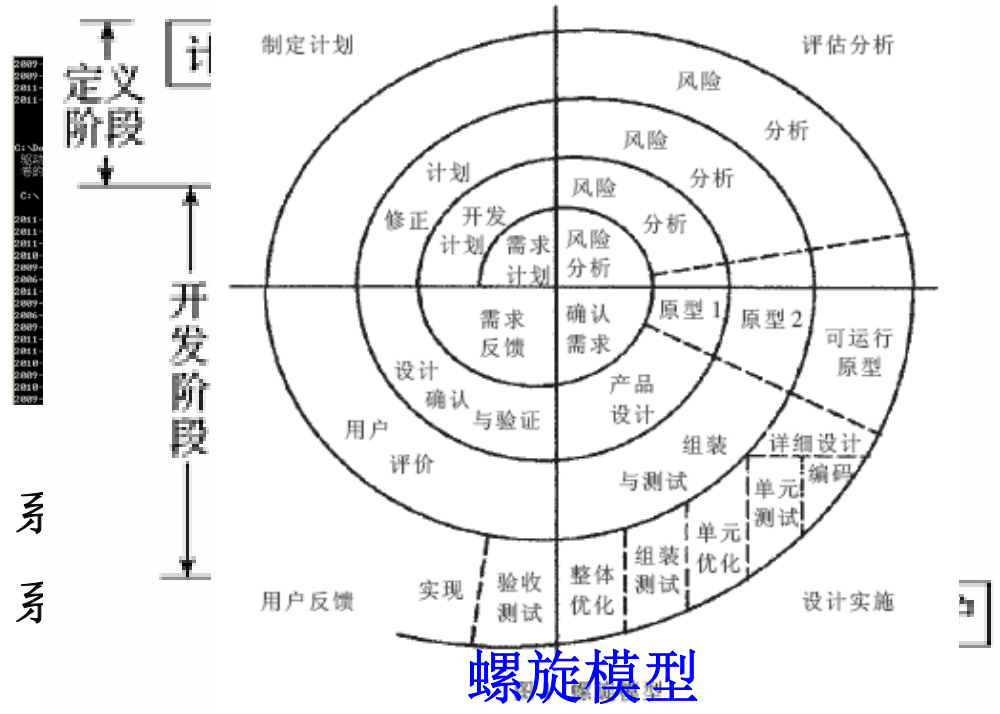
简单的平房

方法：结构化程序设计方法、瀑布模型、螺旋模型等

编程语言：Fortran语言、Pascal语言、C语言

结构化方法好比建平房或用建平房的技术建造复杂建筑





螺旋模型

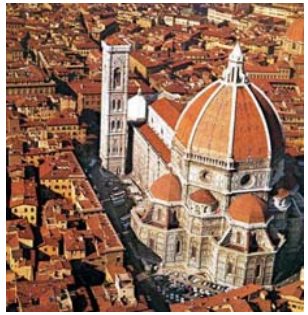
软件及其开发方式的发展

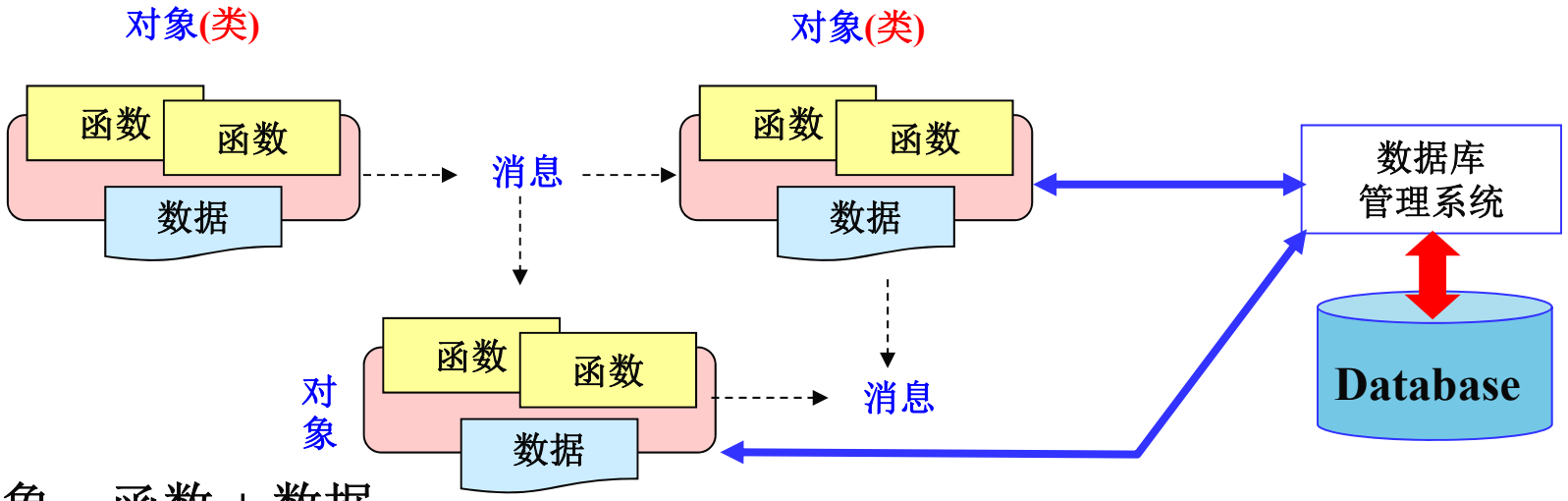
1980's年代: 面向对象的方法

方法：面向对象方法、面向对象模型及建模工具等

编程语言：C++(83)、 Java(95)、Visual 系列语言(90)等

面向对象方法好比建高楼，可以更方便地构建复杂建筑





对象 = 函数 + 数据

系统 = 对象 + 消息 (1980's)

软件及其开发方式的发展

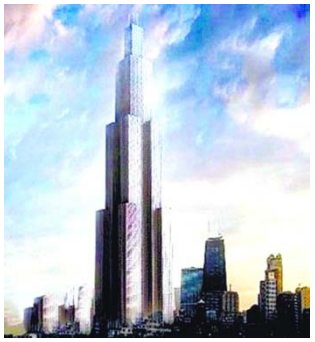
1990's年代: 构件化方法

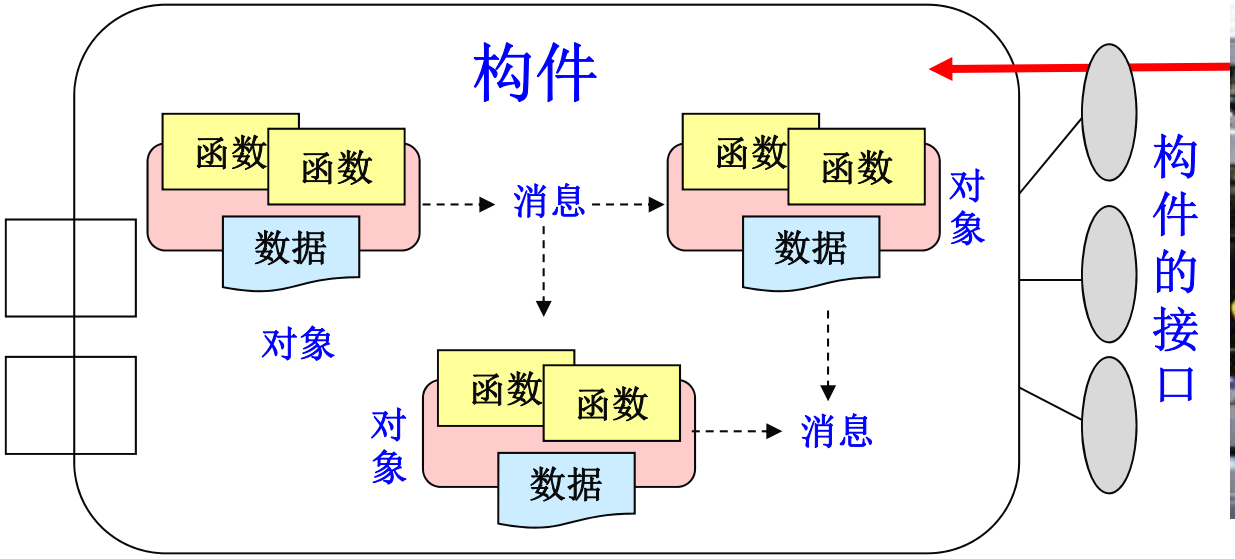
方法：软构件方法、Web Services、软件复用方法等

编程语言：Visual系列语言、Windows操作系统等

构件化方法好比堆积木、造预制件等，可以批量地、快速地构建更为复杂的建筑。

现代化复杂的高楼





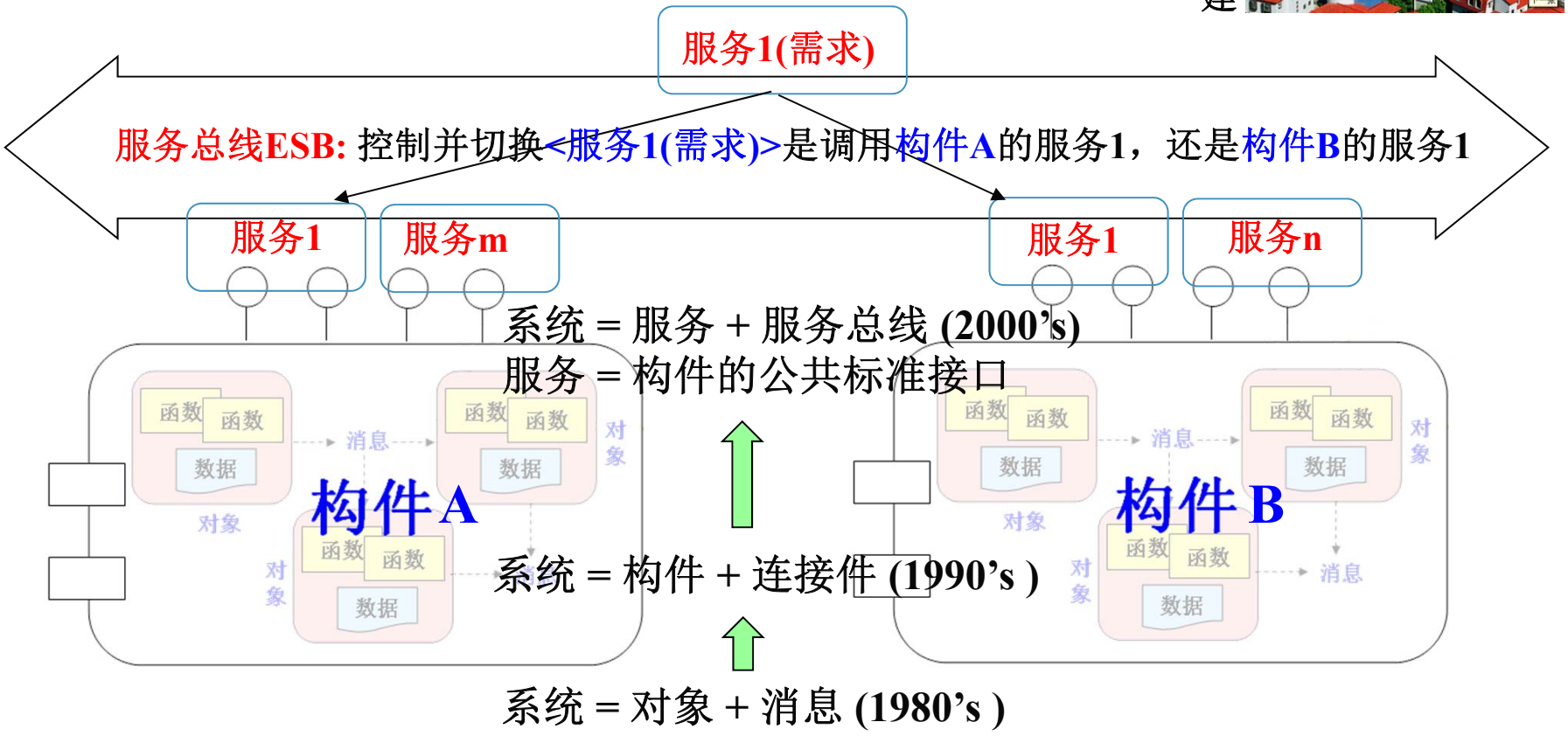
构件 = 对象 + 消息　or　构件 = 实体 + 接口

系统 = 构件 + 连接件 (1990's)

软件及其开发方式的发展

2000's年代：面向服务的体系结构SOA 方法 ➔
基于Internet与云计算的软件开发方法

城镇与城市的构建



软件开发技术的发展过程

- **1950-1960年代：**

  软件=程序(Program)

  面向过程的软件 =算法(Algorithm)+数据结构(Data Structure)

- **1970年代：**

  软件=程序(Program)+文档(Document)

  软件=程序(Program)+文档(Document)+数据(Data)

- **1980-1990年代：**

  面向对象的软件=对象(Object)+消息(Message)

- **1990年代-至今：**

  面向构件的软件=构件(Component)+框架(Framework)

  面向服务的软件=服务(Service)+消息(Message)+总线(Bus)

从另一个角度看软件产业的发展

- 阶段1：独立的程序(Independent Programming Service)
- 阶段2：软件产品(Software Product)
- 阶段3：企业解决方案(Enterprise Solution)
- 阶段4：支持大众应用的软件包(Packaged Software for the Mass)
- 阶段5：网络软件与服务(Internet-Based Software and Service)

软件产业的发展线索


IBM
科学计算软件@Mainframe
SAP
ORACLE
面向商业的软件@PC&Server
YAHOO!
Google
服务化软件@Internet
Microsoft
面向个人的软件@PC
twitter
facebook
Social, Local, Mobile(SoLoMo)@Internet

软件工程概论

- 1.1 软件的基本概念
  - 1.1.1 什么是软件
  - 1.1.2 软件的发展
- **1.2 软件工程的基本概念**
  - 1.2.1 软件工程产生的历史根源
  - 1.2.2 软件工程的基本概念
  - 1.2.3 软件工程的知识体系

软件开发失败的例子

- **1996年6月，阿丽亚娜5号火箭发射失败，内部计算机在发射40秒后失灵，导致它胡乱转向而偏离了航向，自我破坏机构随即将其炸成碎片。**

- **原因：Ariane 4的代码被复用到Ariane 5上，但后者引擎更快，导致了一个将64位浮点数转化为16位有符号整数的计算错误：后者系统中的64位浮点数要大于前者，从而导致溢出，使计算机停止工作。相应的备份机随后也停机。**

阅读：历史上最糟糕的软件bug

http://www.wired.com/software/coolapps/news/2005/11/69355

软件开发失败的例子

- **1963-66年，IBM开发OS/360操作系统，投资几千万美元，工作量5000多人年，拖延几年才交付使用，而且每年要发现近100个错误。**
- **这个项目的负责人F. Brooks事后总结了他在组织开发过程中的沉痛教训时说：**
  - 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，谁也没有料到竟会陷入这样的困境。

我来到软件公司上班后，发现公司以前同事写的程序真是垃圾，根本无法维护。我要推翻重写！

后来一个老员工笑嘻嘻地告诉我，我们现在看到的程序，就是去年的新员工愤怒地推翻重写之后的结果，大家反映还没有以前的版本好用呢。

—— From《现代软件工程讲义》

No Silver Bullet（没有银弹） by F. Brooks

- 复杂的软件工程问题无法靠简单的答案来解决。
- 所有的软件研发都包括了**essential task**和**accidental task**：
  - 前者是去创造出一种由抽象的软件实体所组成的复杂概念结构；
  - 后者则是用编程语言来表现这些抽象的实体，并在某些空间和速度的限制之下，将程序对应至机器语言。
- 软件项目平常看似单纯而率直，但很可能一转眼就变成一只时程延误、预算超支、产品充满瑕疵的怪兽；
- 人们渴望有一种银弹(**silver bullet**)，能够有效解决软件研发中的两大困难：
  - 软件本身在概念建构上具有先天的困难，即如何从抽象性问题发展出具体概念上的解决方案。
  - 将概念构思施行于计算机上所遭遇到的困难。

关于“No Silver Bullet”的系列学术争论

- **Brooks的初始论文：“没有银弹”**
  - Brooks, “No silver bullet — essence and accidents of software engineering,” Computer 20, 4 (April, 1987), pp. 10-19. 《人月神话》第16章
- **Cox的反驳：“银弹存在”**
  - Cox, "There Is a Silver Bullet", Byte (October, 1990), pp. 209-218. http://www.drdobbs.com/there-is-a-silver-bullet/184407534/
- **Harel的细致分析：“紧咬银弹”**
  - Harel, D.,“Biting the silver bullet: Toward a brighter future for system development,”Computer (January, 1992), pp.8-20. http://www.inf.ed.ac.uk/teaching/courses/seoc/2005_2006/resources/bullet10.pdf
- **Brooks继续发表观点：**
  - Brooks, "No Silver Bullet" Refired, 1995. 《人月神话》第17章
- **Cox继续阐述：**
  - Cox, "No Silver Bullet" Revisited, American Programmer Journal. November, 1995. http://virtualschool.edu/cox/pub/NoSilverBulletRevisted/.

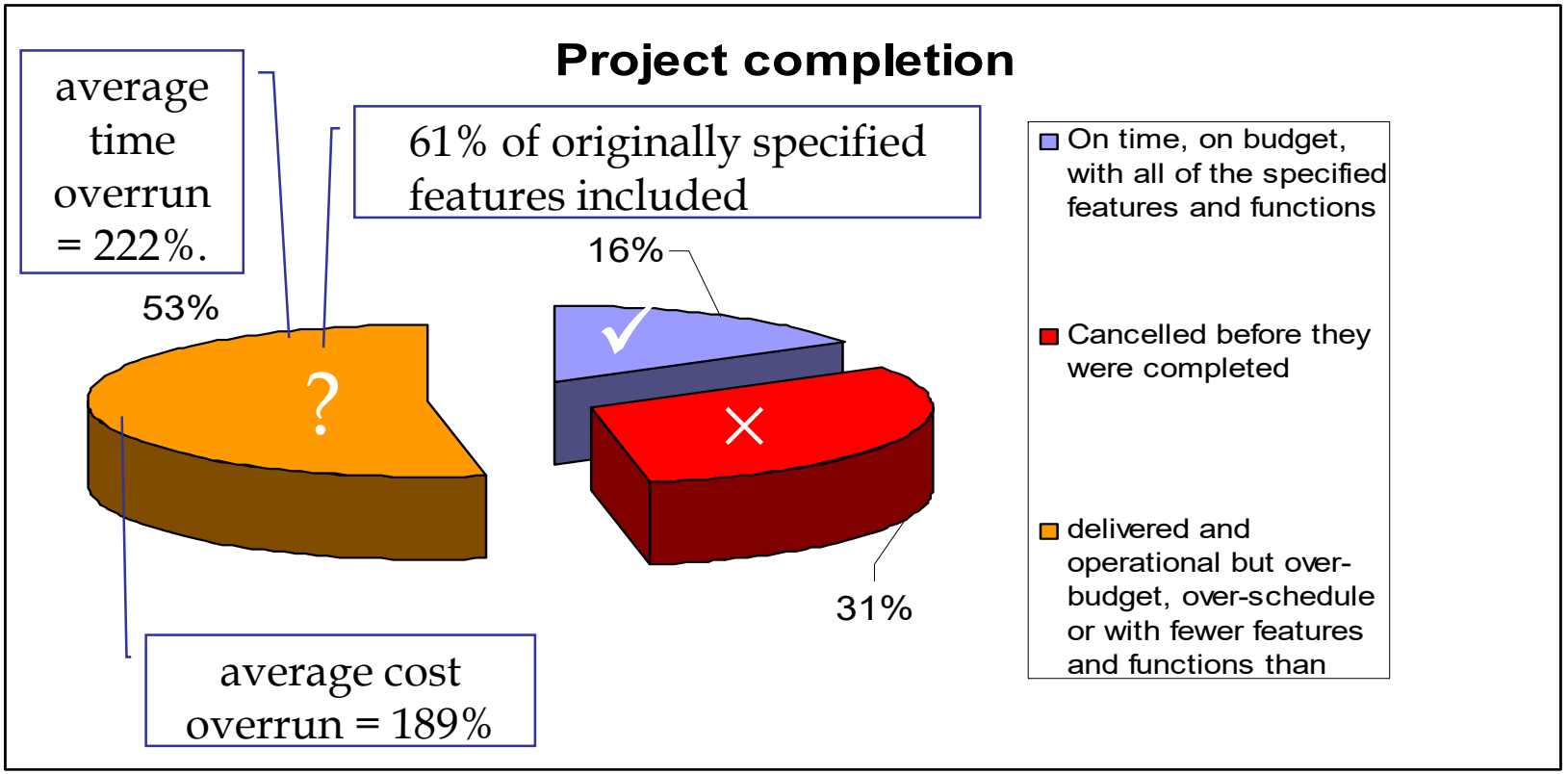
软件危机

- **软件危机(Software Crisis)：计算机软件的开发和维护过程所遇到的一系列严重问题；**
- **软件危机的表现：**
  - 对软件开发成本和进度的估算很不准确，甚至严重拖期和超出预算；
  - 无法满足用户需求，导致用户很不满意；
  - 质量很不可靠，经常失效；
  - 难以更改、调试和增强；
  - 没有适当的文档；
  - 软件成本比重上升；
  - 软件开发生产率跟不上计算机应用迅速深入的趋势。

先看统计数字：混乱的软件开发

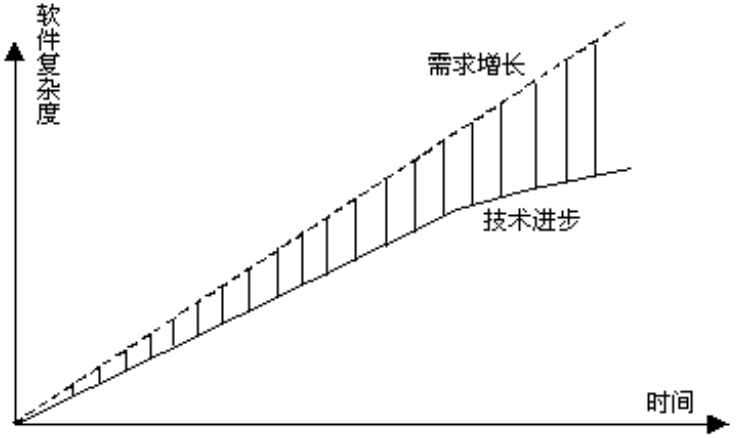
Standish Group – 2004
调查对象：来自美国不同行业的365个公司的CIO；8,380个不同的项目；

调查的结果
- **预算：189%**
- **时间：222%**
- **功能：61%**

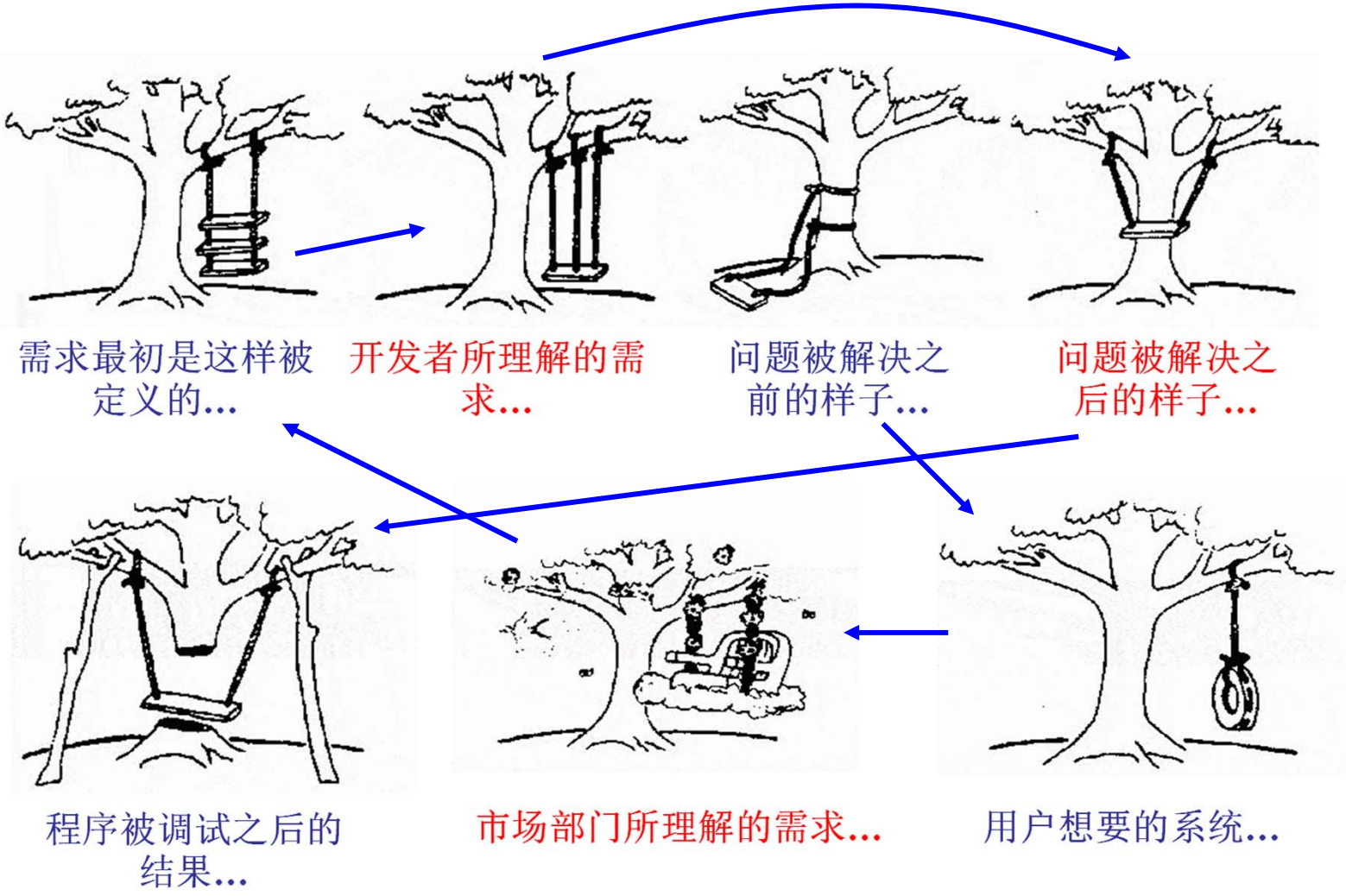


为何出现这种情况？

- **客观上：**
  - 随着软件规模的急速软件产品开发的复杂度和难度随软件规模呈指数级别增长增长，传统的软件开发方法已经不可用了
- **主观上：软件开发人员缺乏工程性的、系统性的方法论**
  - 程序员具有编程的能力，但对软件开发这一过程性较强的任务却缺乏足够的工程化思维；
  - 对软件开发的一些认识的误区：软件神话(Software myths)；
  - 没有将“软件产品研发”与“程序编码”区分清楚；
  - 忽视需求分析、轻视软件维护；



软件通常是按这样的方式构造出来的☹



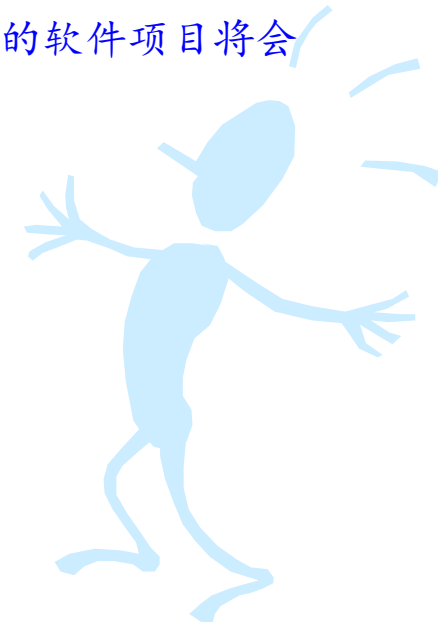
软件神话

- 软件神话(software myths)：关于软件及其开发过程被人盲目相信的一些说法
  - 影响到几乎所有的角色：管理者、顾客、其他非技术性的角色、具体的技术人员；
  - 看起来是事实的合理描述(有时的确包含真实的成分)、符合直觉，并经常被拿来做宣传；
  - 实际上误导了管理者和技术人员对软件开发的态度，从而引发了严重的问题；

经典书籍《人月神话》

软件神话的几个例子

- 客户神话：
  - 有了对项目目标的大概了解，便足以开始编写程序，我们可以在之后的开发过程中逐步补充新的需求；
  - 如果我将一个软件外包给一家软件公司，我可以完全放手不管；
- 软件公司的管理者：
  - 我们已经有了一本写满软件开发标准和规程的“宝典”，它无所不包，囊括了我们可能遇到的任何问题；
  - 如果我们未能按时完成计划，我们可以通过增加程序员人数而赶上进度；
  - 只要我们使用了最先进的软件开发语言、开发工具和开发环境，我们的软件项目将会一帆风顺。
- 开发者：
  - 客户对他们的需求非常了解，他们要什么我就做什么好了；
  - 当我完成程序并将其交付使用之后，我的任务便结束了；
  - 直到程序开始运行，才能评估其质量的好坏；
  - 对一个成功的软件项目，可执行程序是唯一可交付的成果；
  - 软件工程将导致我们产生大量的无用文档，并因此降低工作效率；



“软件工程”的提出

计算机应用发展 → 软件数量多规模大 → 软件成本高质量低

个体化软件开发方法 → 软件维护困难

→ 软件危机 → 软件工程

软件工程概论

- 1.1 软件的基本概念
  - 1.1.1 什么是软件
  - 1.1.2 软件的发展
- **1.2 软件工程的基本概念**
  - 1.2.1 软件工程产生的历史根源
  - 1.2.2 软件工程的基本概念
  - 1.2.3 软件工程的知识体系

“软件工程”术语的产生

- **1968年NATO的科学委员会在德国召开的一次计算机软件国际会议上，对软件开发的方法、 技术进行了广泛的讨论，首次提出了“软件工程”的概念。**
  - 软件工程是为了经济的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工程化原则；
- **(IEEE) 软件工程是：**
  - 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，即将工程化应用到软件上；
- **(国务院学位委员会)软件工程是：**
  - 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定义、开发、发布和维护的工程，或以之为研究对象的学科。

什么是“软件工程”？

- **软件工程**：用来开发、管理和维护软件产品的理论、方法和工具； (I. Sommerville)
- **软件工程**：开发高质量软件的生产过程，力争按期交付、不超过预算成本、尽可能的满足用户需求；(S.R. Schach)
- **软件工程**：将科学知识实际应用于计算机程序及其开发、操作和维护相关的文档的设计与构造过程；(B.W. Boehm)
- **软件工程**：技术与管理型的方法，以在预定时间与成本约束条件下系统化的生产、开发、维护和修改软件产品； (R. Fairley)
- **软件工程**：将相关工具与技术系统化的应用于软件开发的过程当中；(S. Conger)
- **软件工程**：设计与开发高质量的软件系统的技术；(S.L. Pfleeger)

什么是软件工程？

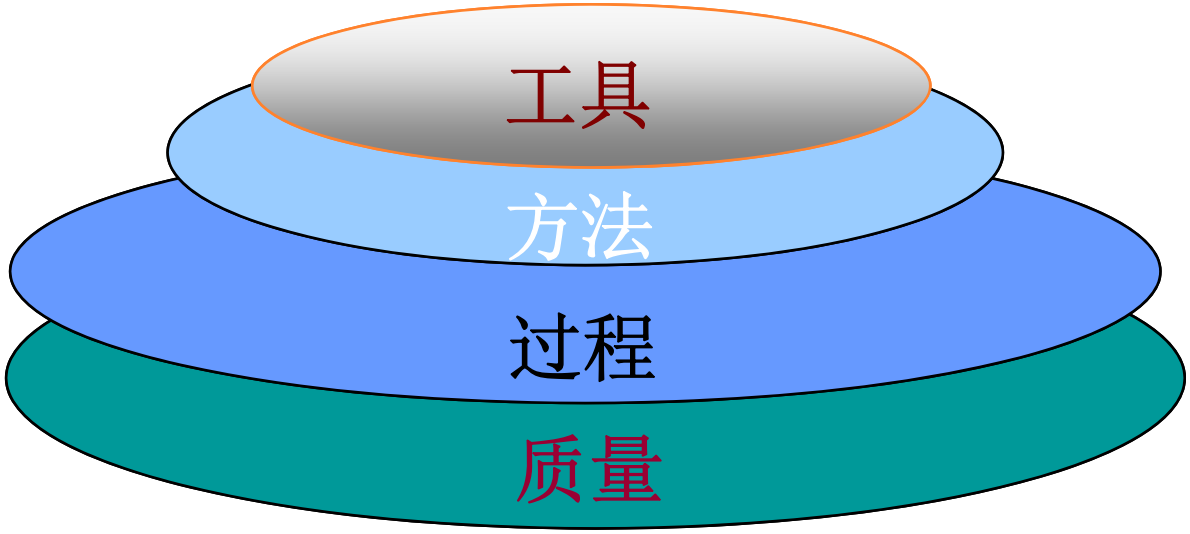
- 软件工程重要的不是技术而是如何开发软件项目的思想

- 软件工程是一种建模活动
- 软件工程是一种解决问题的活动
- 软件工程是一种知识获取的活动
- 软件工程是一种受软件工程原理指导的活动

归纳起来，“软件工程”是…

- 范围：
  - 软件开发过程(设计、开发、运行、维护)
  - 软件开发中应遵循的原则和管理技术
  - 软件开发中所采用的技术和工具

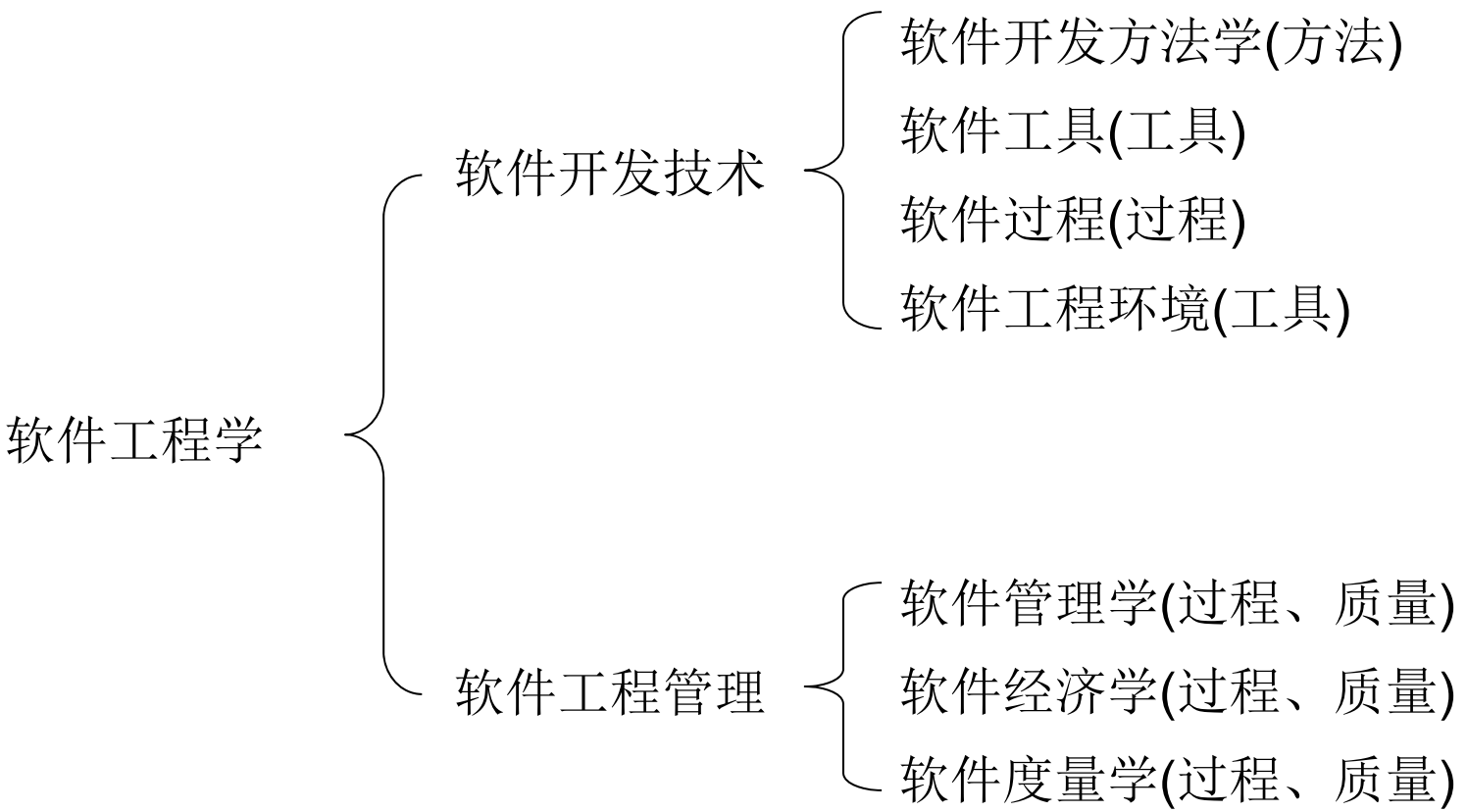
- 目标：
  - 高质量
  - 按时交付
  - 控制成本
  - 满足用户需求



软件工程概论

- 1.1 软件的基本概念
  - 1.1.1 什么是软件
  - 1.1.2 软件的发展
- **1.2 软件工程的基本概念**
  - 1.2.1 软件工程产生的历史根源
  - 1.2.2 软件工程的基本概念
  - 1.2.3 软件工程的知识体系

软件工程的范畴


软件工程学
软件开发技术
软件开发方法学(方法)
软件工具(工具)
软件过程(过程)
软件工程环境(工具)
软件工程管理
软件管理学(过程、质量)
软件经济学(过程、质量)
软件度量学(过程、质量)

(1) 软件开发方法学

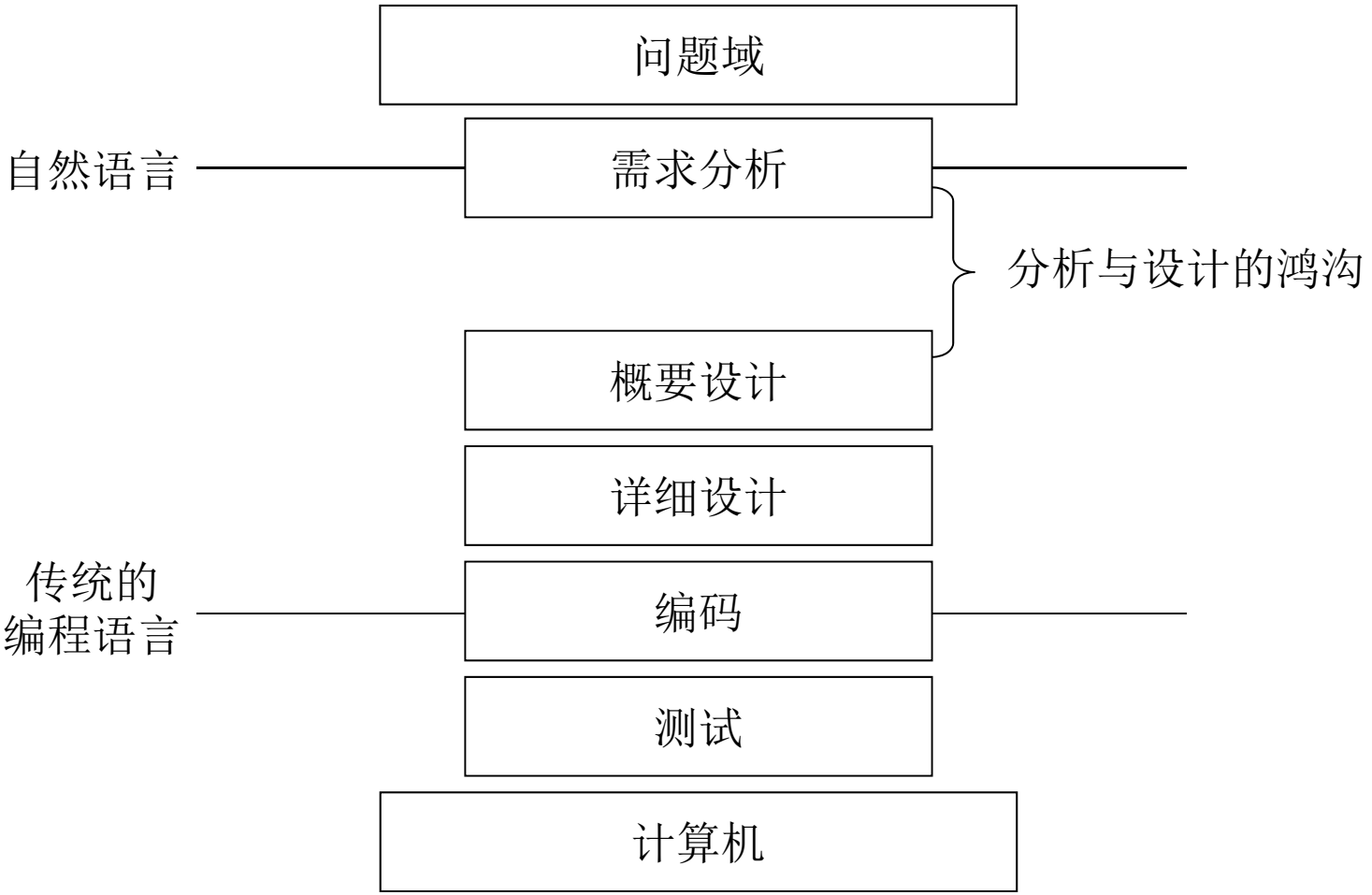
- 软件开发方法：
  - “如何做”，How to do；
  - 使用预先定义好的一组模型表示方法、良好的设计技术与原则、质量保证标准等方面来组织软件开发的过程；
- 分类
  - 结构化开发方法(Structured Analysis and Design Technique, SADT)
  - 面向对象开发方法(Object Oriented Analysis and Design, OOAD)

(1) 软件开发方法学

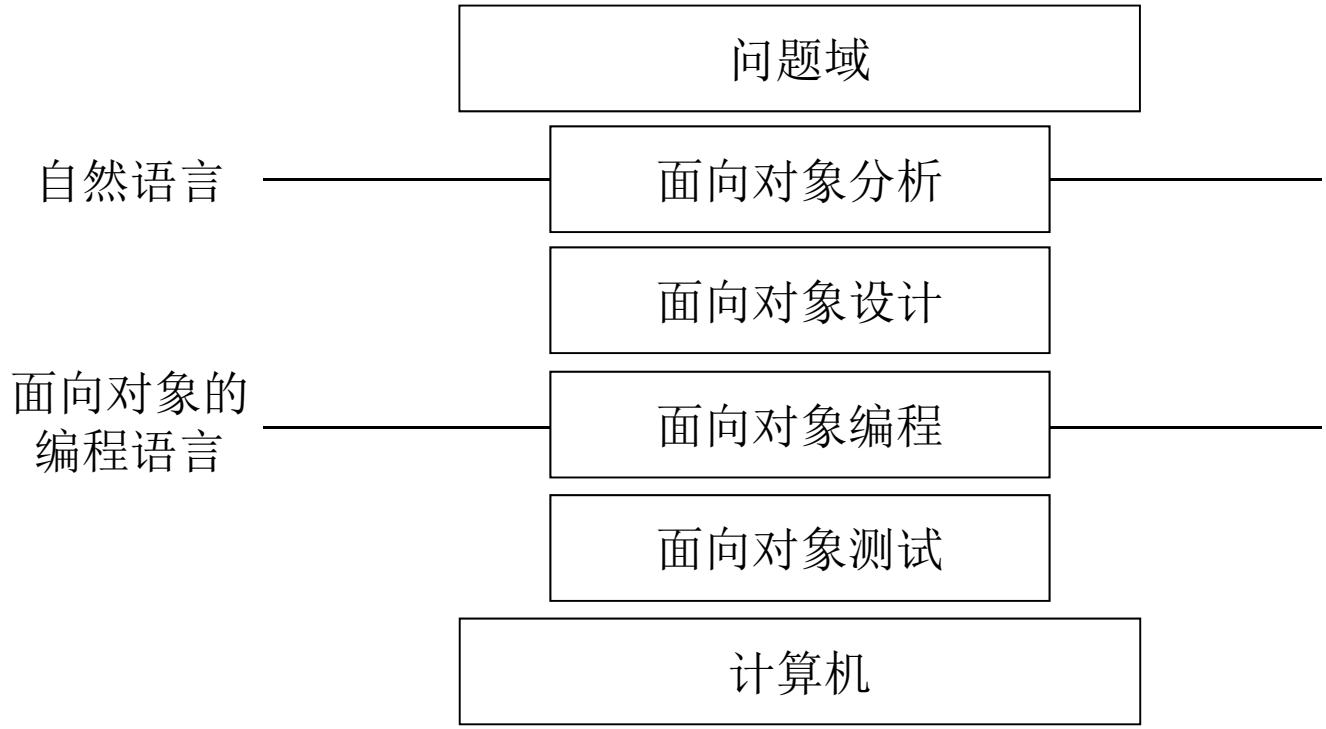
- 传统软件工程
  - 以结构化程序设计为基础
  - 程序=数据结构+算法
  - 自顶向下：结构化需求分析→结构化设计(概要设计、详细设计)→面向过程的编码→测试

- 面向对象软件工程
  - 以面向对象程序设计为基础
  - 程序 = 对象 + 消息
  - 软件分析与对象抽取→对象详细设计→面向对象的编码→面向对象的测试

传统软件工程方法：结构化方法



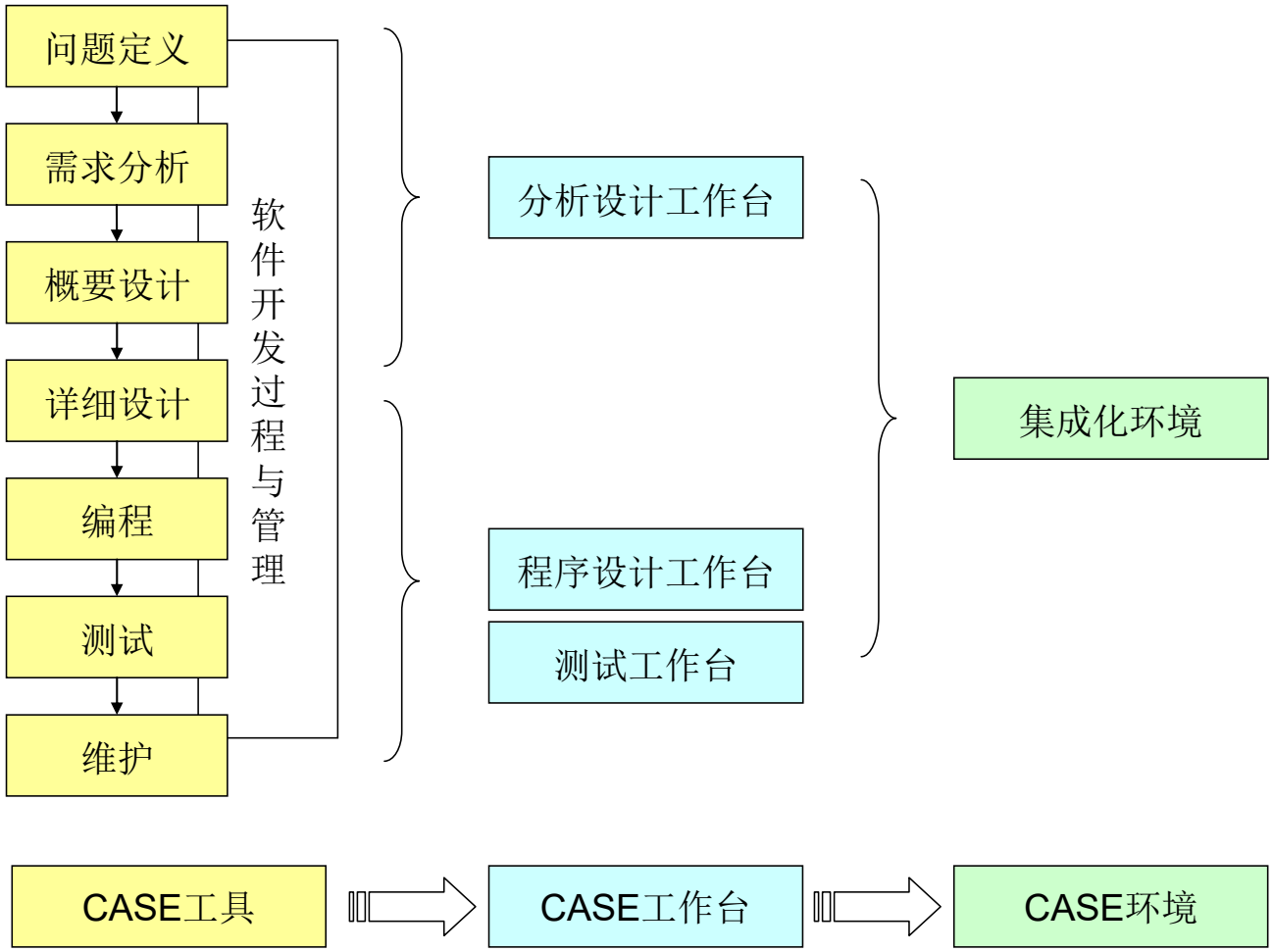
面向对象方法



(2) 软件工具与软件工程环境

- **工具：自动或半自动的软件支撑环境，辅助软件开发任务的完成，提高开发效率和软件质量、降低开发成本**
  - 项目管理工具
  - 需求管理工具
  - 设计建模工具
  - 编程与调试工具
  - 测试与维护工具
- **多个工具集成在一起，形成了软件工程开发环境CASE (Computer Aided Software Engineering)，全面支持软件开发的全过程。**

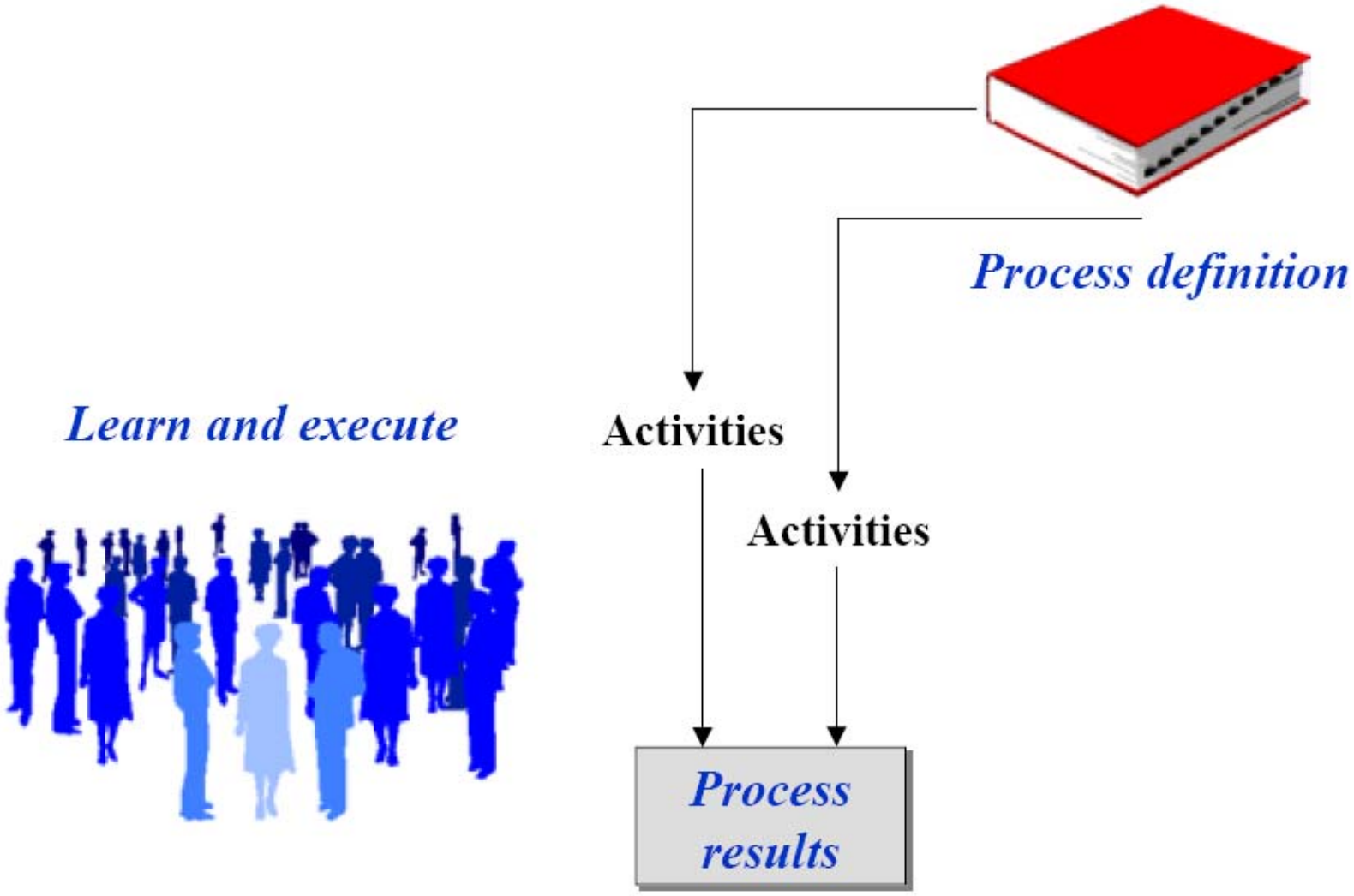
计算机辅助软件工程CASE



(3) 软件过程

- **软件工程的过程：**
  - 管理和控制产品质量的关键；
  - 由一系列活动与步骤组成，如需求分析与设计、开发、验证与测试、演化与维护等；
  - 定义了技术方法的采用、工程产品(模型、文档、数据、报告、表格等)的产生、里程碑的建立、质量的保证和变更的管理；
  - 将人员、技术、组织与管理有机的结合在一起，实现在规定的时间和预算内开发高质量软件的目标。

(3) 软件过程

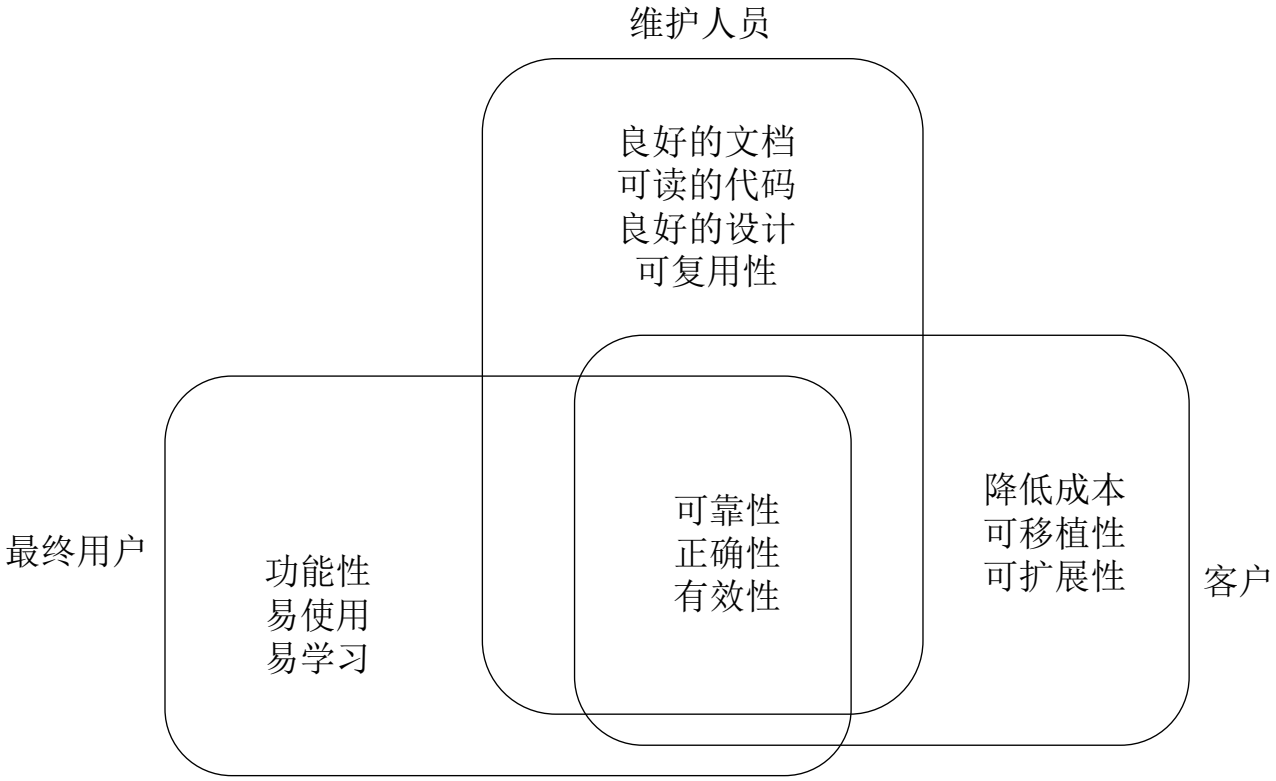

Process definition
Learn and execute
Activities
Activities
Process results

(4) 软件工程管理

- 目的
  - 为了按照进度和预算完成软件开发计划
- 内容
  - 成本估算
  - 进度安排
  - 人员组织
  - 质量保证
  - …

(5) 软件质量特性

- **软件质量：**
  - 软件产品与需求相一致的程度，由一系列质量特性来描述；
  - 并不取决于开发人员自身，通常与客户、用户、维护人员等提出的要求密切相关；



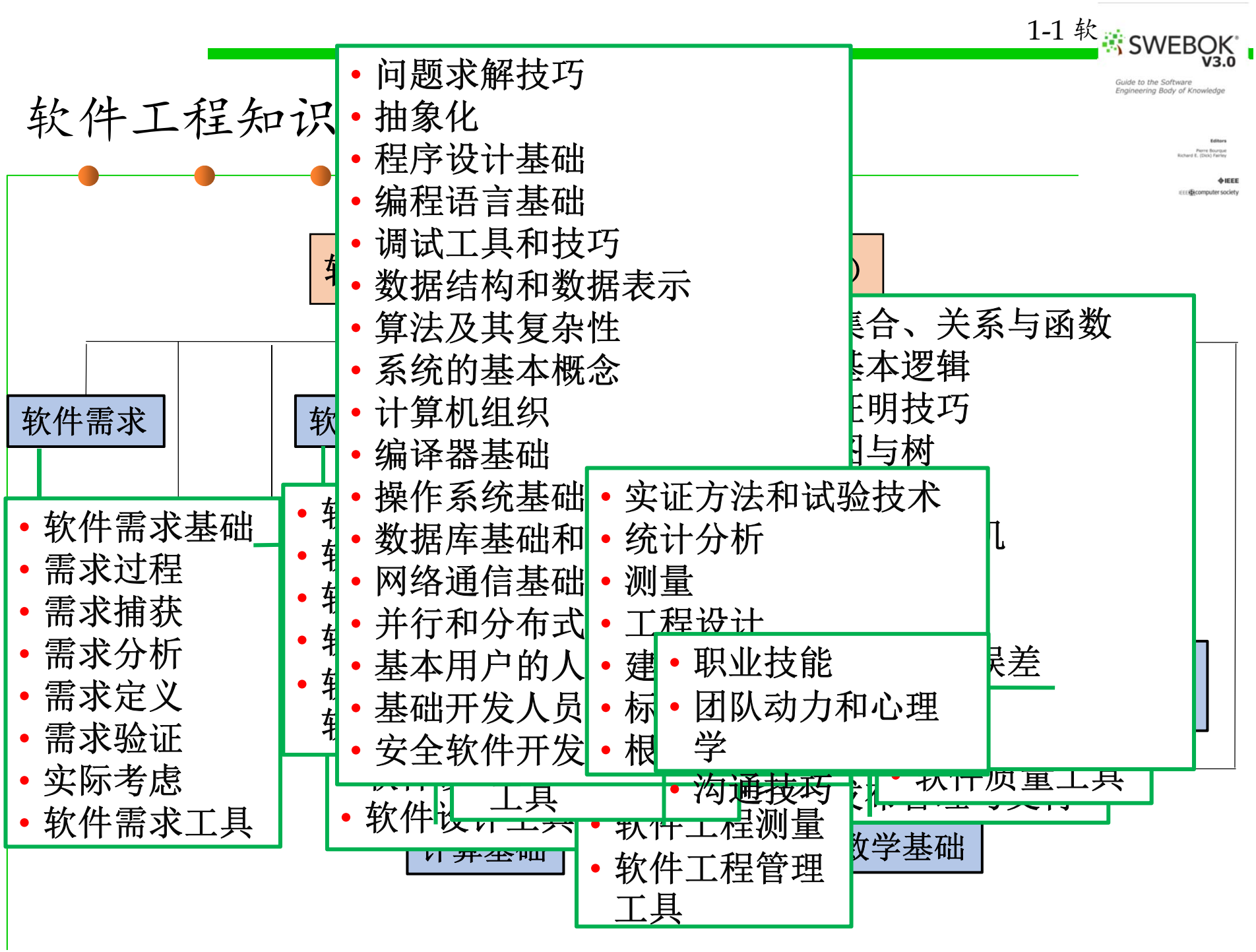
(5) 软件质量特性

- **软件有“bug”(缺陷)，软件团队的很多人都整天和 bug 打交道，bug 的多少可以直接衡量一个软件的开发效率、用户满意度、可靠性、可维护性**
  - 软件的开发效率 – 开发过程中bug 太多了，导致软件无法按时交付；
  - 用户满意度 – 用户使用时报告了很多bug，纷纷表示对生活影响很大；
  - 可靠性 – 这个软件经常会崩溃，这个操作系统会死机；
  - 可维护性 – 这个软件太难维护了，按下葫芦起了瓢，修复了一个问题, 另一个问题又出来了。也没有足够的文档，维护人员纷纷表示要把原作者找出来打一顿。
- **Bug=软件的行为和用户的期望值不一样；**
- **微软的观点：完美的软件在世界上是不存在的，软件工程的一个重要任务就是要决定一个软件在什么时候能“足够好”，没有严重的“bug”，可以发布。**

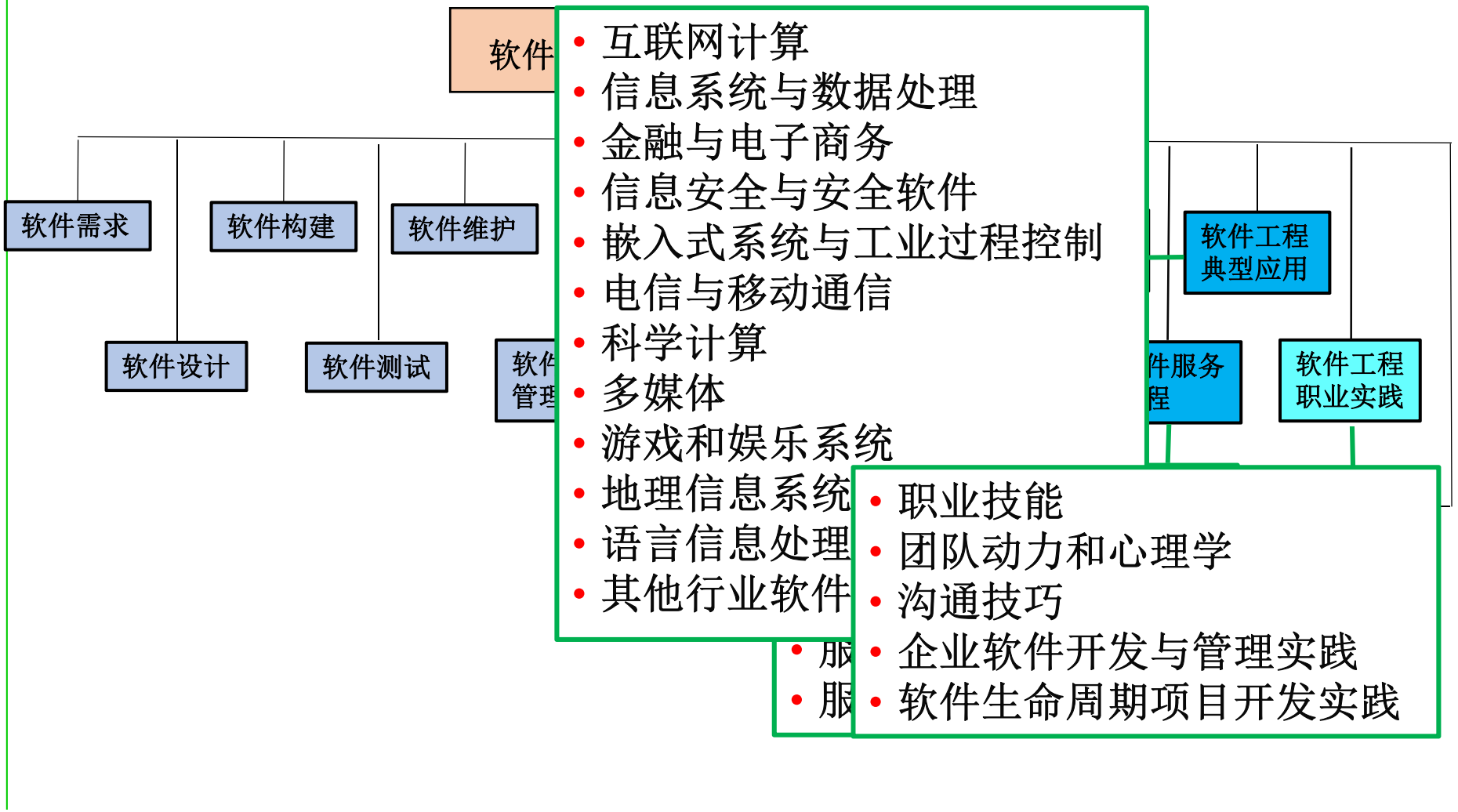
From 《现代软件工程讲义》

软件工程知识体系SWEBOK

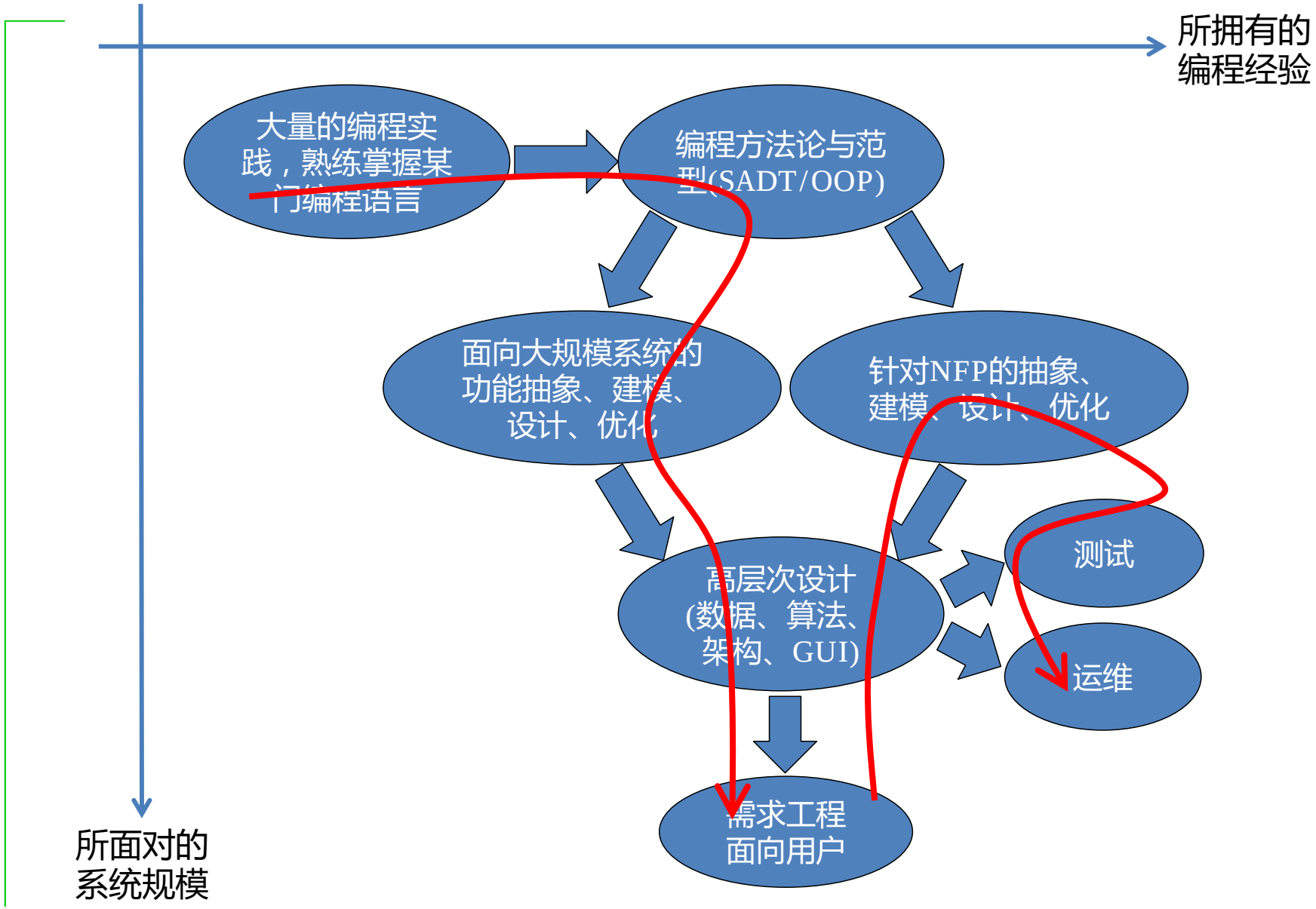
- IEEE CS 软件工程专业知识体系SWEBOK，软件工程知识体系 **(Software Engineering Body of Knowledge, SWEBOK)**
- 美国电子电气工程师学会IEEE CS与美国计算机联合会ACM成立了软件工程协调委员会， 于1994年开始研究软件工程知识体系（SWEBOK）；历经草人阶段、石人阶段和铁人阶段；IEEE CS于2001、2004年先后发布SWEBOK 1.0、2.0版。2014年3月，IEEE CS正式发布SWEBOK 3.0版，成为软件工程知识体系的样板。
  - 为软件工程学科定义清晰的边界；
  - 描述软件工程学科的内容、特征，以及与其他相关学科的关系；
  - 为软件工程课程计划的开发和职业资格认证提供依据；
  - 最新版本由IEEE - CS于2014年发布。


SWEBOK V3.0
Guide to the Software Engineering Body of Knowledge
软件工程知识
软件需求
软件需求基础
需求过程
需求捕获
需求分析
需求定义
需求验证
实际考虑
软件需求工具
问题求解技巧
抽象化
程序设计基础
编程语言基础
调试工具和技巧
数据结构和数据表示
算法及其复杂性
系统的基本概念
计算机组织
编译器基础
操作系统基础
数据库基础和
网络通信基础
并行和分布式
基本用户的人
基础开发人员
安全软件开发
实证方法和试验技术
统计分析
测量
工程设计
职业技能
团队动力和心理学
沟通技巧
软件工程测量
软件工程管理工具
集合、关系与函数
基本逻辑
证明技巧
图与树
误差
软件质量工具
数学基础
工具
软件设计工具

软件工程知识体系：C-SWEBOK（中国软件工程教指委）



小结：软件工程的知识与技术范畴



本章阅读材料（已上传）

- 《人月神话》
- 《Fifty Years of Progress in Software Engineering》
- 《A View of 20th and 21st Century Software Engineering》
- 《what knowledge is important to a software professional》

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束

2017年9月4日