

# 软件工程与方法

## 第1讲 软件与软件工程

# 本章内容

- 软件工程中常见问题
- 软件工程的基本概念
- 软件生存期模型
- 软件工程知识体系

# 第1讲 软件与软件工程

## 1.1 软件工程中常见问题

# 软件工程的基本问题

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is parts of this more general process.

(续)

Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes, whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What are the main uncertainties in software development?	Environmental uncertainty and software complexity

# 软件的构成

Software is a set of items or objects that form a “configuration” that includes

- programs
- documents
- data ...

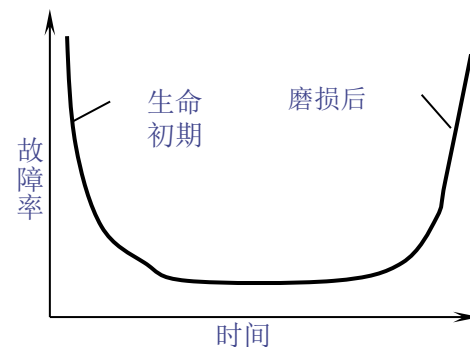
Types of software products:

- Generic products（通用产品）
- Customized products（定制产品）

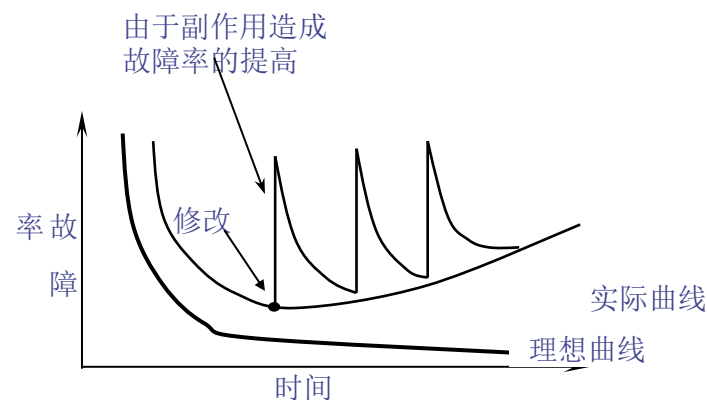


# 软件的特点:

- 1) 软件是一种逻辑实体,而不是具体的物理实体。因而它具有抽象性。
- 2) 软件的生成与硬件不同,在软件的开发过程中没有明显的制造过程。
- 3) 在软件的运行和使用期间,没有硬件那样的机械磨损、老化问题。
- 4) 软件的开发和运行常常受到计算机系统的限制,对计算机系统有着不同程序的依赖性。
- 5) 软件开发至今尚未完全摆脱手工的开发方式。
- 6) 软件的复杂性。
- 7) 软件成本相当昂贵。
- 8) 社会性




硬件的故障率曲线



软件的故障率曲线(实际情况下)

中新网9月8日电综合报道,美国征信机构Equifax称它们的数据库遭到了攻击,将近1.43亿美国人的个人信息可能被泄露,这几乎是全美人口的一半。网络犯罪者已经接触到了包括姓名、社会安全号码、出生日期、地址和驾照编号等在内的敏感信息。

# 软件分类

- 按功能进行划分
- 按规模进行划分 
- 按工作方式划分
- 按服务对象的范围划分
- 按使用的频度进行划分
- 按失效的影响进行划分

软件规模分类

类别	参加人数	研制期限	产品规模(KLOC)
微型	1	1~4周	0.5
小型	1	1~6月	1~2
中型	2~5	1~2年	5~50
大型	5~20	2~3年	50~100
甚大型	100~1000	4~5年	1,000
极大型	2000~5000	5~10年	1,000~10,000



# 第1讲 软件与软件工程

## 1.2 什么是软件工程

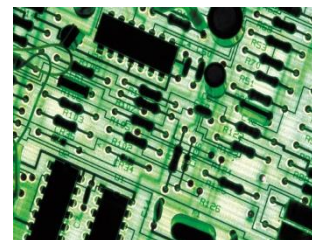
# 什么是软件工程

- 软件工程是一门工程学科，涉及软件生产的各个方面，从最初的系统描述一直到使用后的系统维护，都属于其学科范畴。
- 在软件工程的定义中有两个关键词：
  - 工程学科（Engineering discipline）
  - 软件生产的各个方面（all aspects of software production）

- 工程的特点：
  - 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：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

# 软件工程与一般工程的差别：(1/2)

- 软件是逻辑产品，它不会损坏、磨损、老化，而且可以不断地改进、优化，其可靠性是逻辑性所确定。
- 由于软件是逻辑产品，它的功能只能依赖于硬件和运行环境以及人们对它的操作，才能得以体现。
- 对软件产品的要求比一般有形产品复杂：
  - 用户难以清晰、准确地表达软件产品的功能。
  - 对软件产品的要求，如可靠性、易移植性、易使用性等等是隐含的，也是难以表达的，而且也缺少度量的具体标准，和有形产品的质量检验的精度相距甚远。
  - 软件设计不仅仅涉及到技术复杂性，也不定期涉及到管理复杂性。



# 软件工程与一般工程的差别：(2/2)

- 在软件设计中的复杂性，引起的软件特征包括四方面：
  - 功能多样性（**Functional diversity**）
  - 实现多样性（**Implementation diversity**）
  - 能见度低（**Low visibility**）
  - 软件结构的合理性差（**poor rationality of software structure**）
- 任何一种工程，在其年轻时代总是人工密集的，而到其成熟时期则成为资金密集的。软件工程是智力密集。

Ship Date	Product	Dev Team Size	Test Team Size	Lines of code (LoC)
Jul-93	NT 1.0 (released as 3.1)	200	140	4-5 million
Sep-94	NT 2.0 (released as 3.5)	450	230	7-8 million
May-95	NT 3.0 (released as 3.51)	450	325	9-10 million
Jul-96	NT 4.0 (released as 4.0)	800	700	11-12 million
Dec-99	NT 5.0 (Windows 2000)	1,400	1,700	29+ million
Oct-01	NT 5.1 (Windows XP)	1,800	2,200	40 million
Apr-03	NT 5.2 (Windows Server 2003)	2,000	2,400	50 million

# 软件开发中存在的突出问题

- **成本、进度和质量**仍将是致软件激烈竞争的主要因素。
  - 软件制作能力提高的同时，软件系统的复杂程度也在提高。
  - 计算机与通讯等其它系统的融合，产生的新技术也给软件开发提出了新要求。
  - 开发工作中没有自始至终遵循的工作方法。
- 
- 目标在于使软件开系统向高性价比发展，具体目标：
    - 付出较低的开发成本
    - 达到要求的软件功能
    - 取得较好的软件性能
    - 开发的软件易于移植
    - 需要较低的维护费用
    - 能按时交付使用

# 是否存在银弹？



- Brooks, F. P.. *No Silver Bullet: Essence and Accidents of Software Engineering*. IEEE Computer, 1987, 20(4): 10-19.
- Brooks warn us that certain of the difficulties of building software systems derive from the essential nature of software, hence are unlikely to be overcome by any single breakthrough.

Fred Brooks has famously observed that four properties of software, taken together, differentiate it from other kinds of engineering artifacts (Brooks 1995). These four properties are

1. complexity
2. conformity
3. changeability
4. invisibility

## ● 失败案例

- 1992年，法国伦敦由于救护派遣系统全部崩溃，导致多名病人因为抢救不及时而失去生命。
- 1996年，欧洲航天局首次发射阿丽亚娜5号火箭失败，其直接原因是火箭控制系统的软件故障，导致直接经济损失5亿美元，使耗资80亿美元的开发计划延迟了三年。
- 2003年，在美国电力检测与控制管理系统中，由于分布式计算机系统试图同时访问同一资源引起软件失效，造成美国东北部大面积停电，损失超过60亿美元。

# 研究失败的软件项目的书

Robert L. Glass. *Software Runaways: Lessons Learned from Massive Software Project Failures*. 1998 Prentice Hall PTR.

- 与传统的信息来源相致一的情况：
  - 多数超出控制范围的软件项目都是（或曾经是）大型项目。
  - 多数软件项目之所以失败，原因有多种。可能有、也可能没有一个起主导作用的原因，但是总会有多个问题导致项目的失败。
  - 很多超出控制范围的软件项目在其开发初期被人们赞誉为“有重大进展”，与将要替换掉的系统相比，这些项目具有极大的优势。如果该项目不是被全面展开，人们似乎都看不到有失败的可能。
- 以前没有人提及的特点：
  - 技术与管理一样，也经常成为失败的原因。
  - 存在有两个尤其令人意外的主要技术性问题：
    - ◆ 首先，新技术的使用。
    - ◆ 其次，主要的技术问题是性能。

# 软件工程的各种定义：

- **Fritz Bauer**在1969年NATO会议上的定义
  - 为了经济地获得软件，这个软件是可靠的并且能在实在的计算机上工作，所需要的健全的工程原理（方法）的确立和使用。
- **P. Wegner & B. Boehm**认为：
  - 运用现代科学知识在设计和构造计算机程序，以及开发、运行和维护这些程序所要求的有关文档编制中的实际应用。
- **IEEE**在软件汇编术语中定义：
  - 对软件开发、运作、维护的系统化的、有规范的、可定量的方法之应用，即是对软件的工程化应用。



# 软件工程定义：

**软件工程是指导计算机软件开发和维护的工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它，这就是软件工程。**

## ● 软件工程的特点：

- 工程是将理论和知识应用于实践的科学。
- 软件工程借鉴了传统工程的原则和方法，以求高效地开发高质量软件。
- 软件工程中应用了计算机科学、数学和管理科学。计算机科学和数学用于构造模型与算法，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、质量和成本的管理。

# 软件工程知识构成：

## 软件工程的知識域：

- 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

## 软件工程的三个分支：

- 软件开发技术
- 软件项目管理技术
- 软件质量管理技术

## 软件工程框架：

- 软件工程的目标
- 软件工程原则
- 软件工程活动

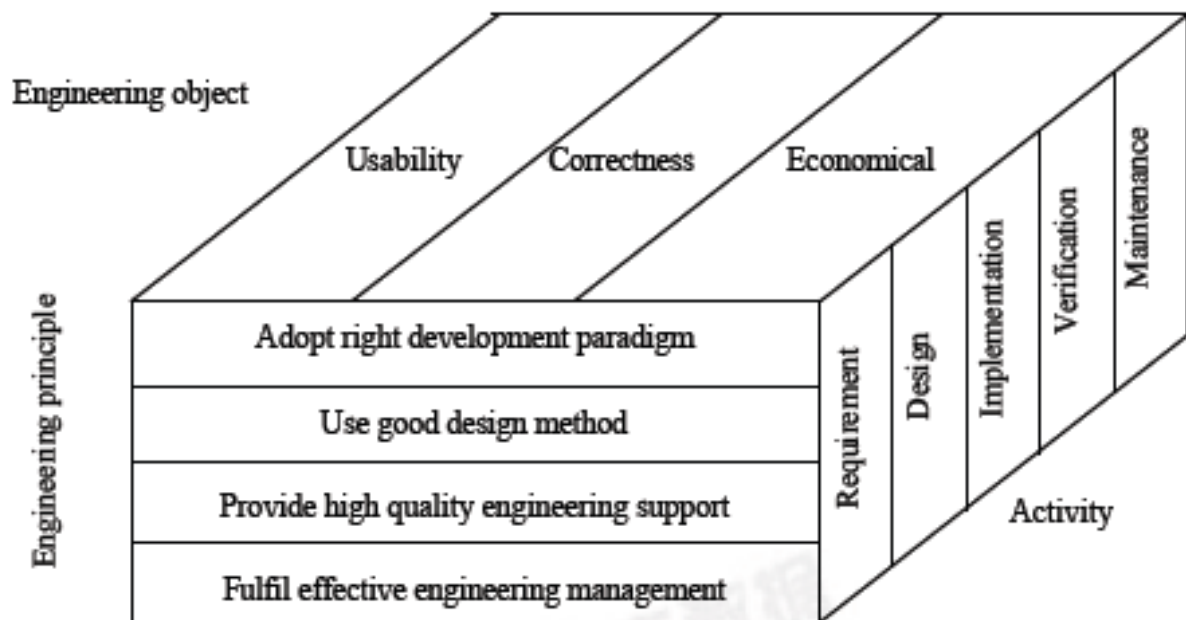


Fig.2 Software engineering framework

# 软件工程框架

软件工程框架：

- 软件工程的目标
- 软件工程原则
- 软件工程活动

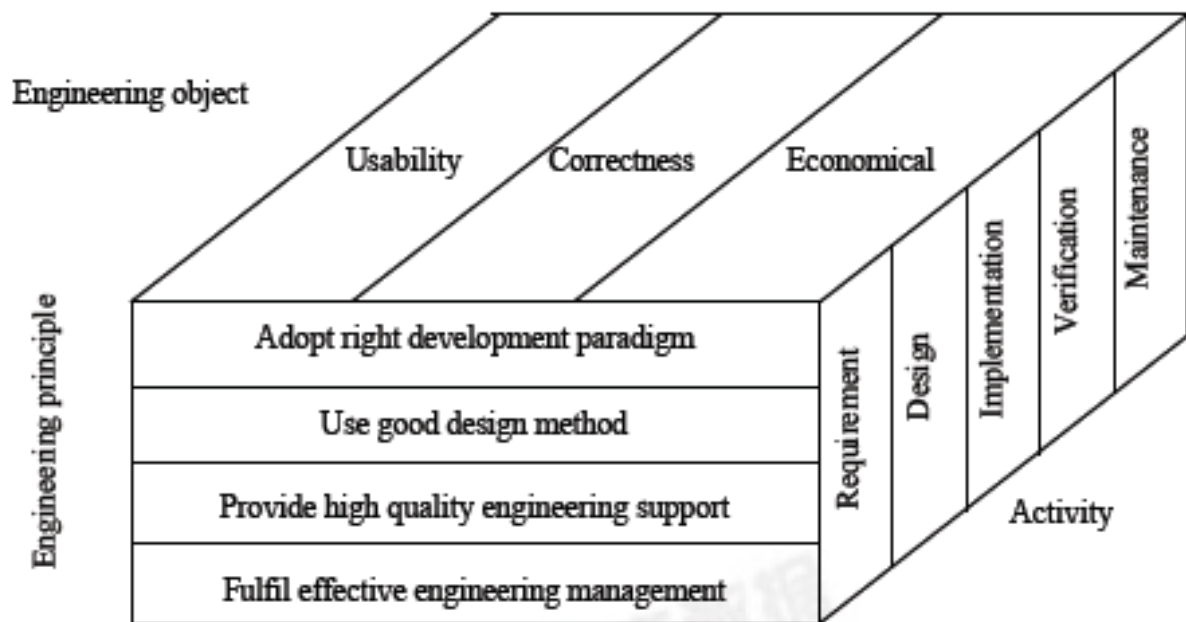
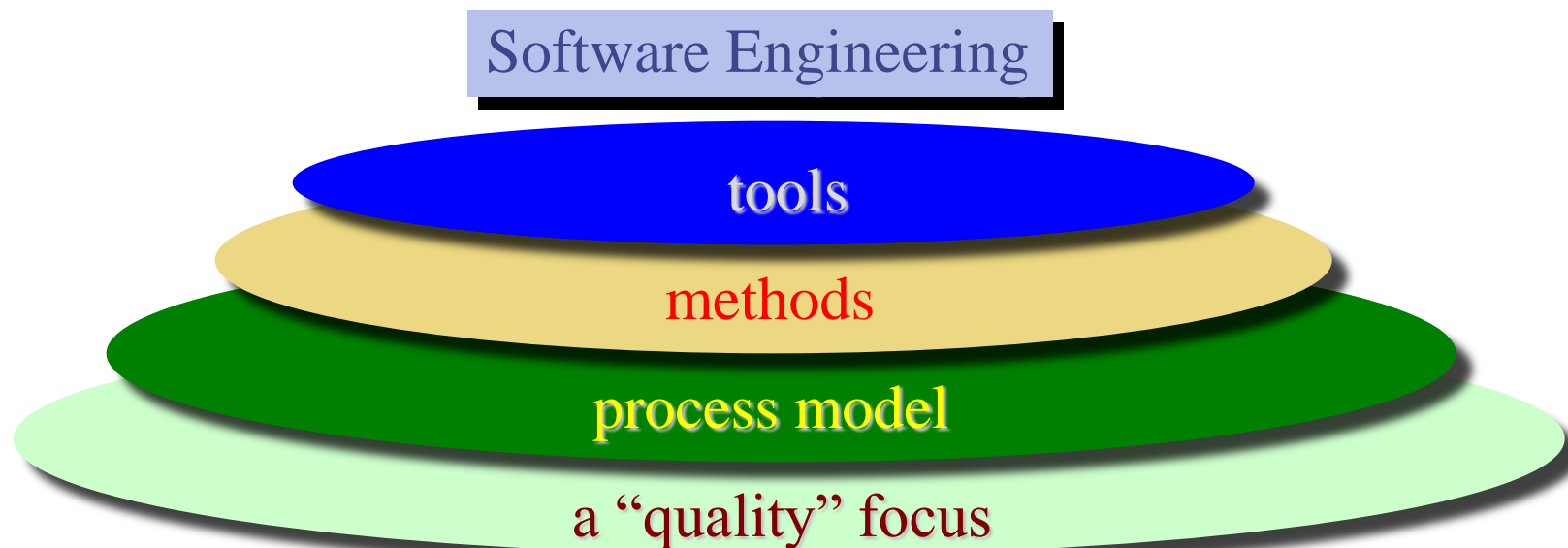


Fig.2 Software engineering framework

# 软件工程3要素



软件工程3要素：

- ① 方法：为软件开发提供“如何做”的技术。
- ② 工具：为软件工程方法提供自动或半自动的软件支撑环境。
- ③ 过程：将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。

# 软件工程的四条基本原则：

- 必须认识软件需求的变动性，并采取措施来保证结果产品能忠实地满足用户需求。
- 采用合适的设计方法。在软件设计中，通常要考虑软件的模块化、抽象与信息隐蔽、局部化、一致性以及适应性等特征。合适的设计方法有助于这些特征的实现，以达到软件工程的目标。
- 软件工程项目的质量与经济开销直接取决于对它所提供的支撑的质量和效用。
- 有效的软件工程只有在对软件过程进行有效管理的情况下才能实现。

# 软件工程的基本原理（Barry. Boehm）

- ① 用分阶段的生命周期计划严格管理
- ② 坚持进行阶段评审
- ③ 实行严格的产品控制
- ④ 采用现代程序设计技术
- ⑤ 结果应能清楚地审查
- ⑥ 开发小组的人员应该少而精
- ⑦ 承认不断改进软件工程实践的必要性



# 对软件的新认识

- 程序从个人按自己意图创造的“艺术品”转变为广大用户接受的工程化产品。
- 软件的需求是软件发展的动力。
- 软件工作的范围从只考虑程序的编写到涉及整个软件生存期。
- 软件是服务。
- 好的软件应具有用户所需的功能与性能，而且应该可维护、可靠和可用。
- 软件工程面临的主要挑战是要面临遗留系统、不断增长的多样性以及减少交付次数等问题。

# 与软件工程有关的问题

- 软件工程与计算机科学的区别：
  - 计算机科学侧重于理论和基础。
  - 软件工程侧重于开发和交付的实际活动。
- 软件工程与系統工程的区别：
  - 系统工程侧重基于计算机系统开发的所有方面，包括硬件、软件和处理工程。
  - 软件工程只是它的一部分。
- 软件工程的方法
  - 软件开发的结构化研究方法，包括：系统模型、标记法、规则、设计忠告和过程指南。
- **CASE**（计算机辅助软件工程）
  - 旨在使软件过程活动自动化的软件系统。
  - **CASE**常用作方法支持



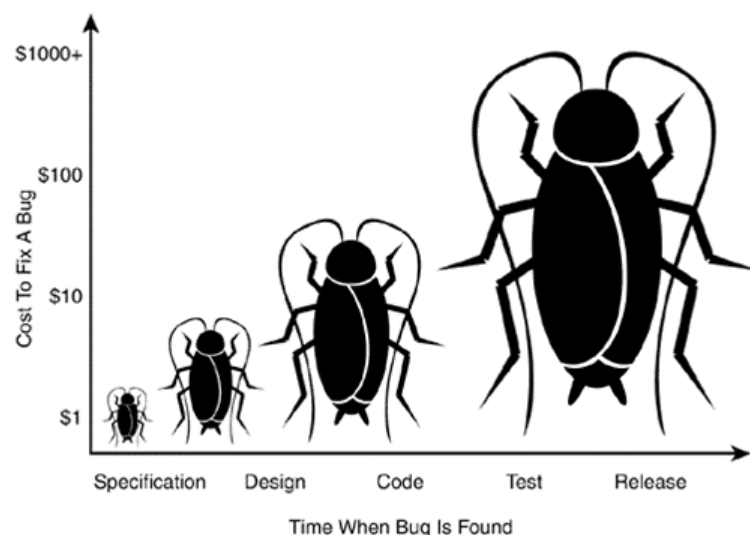
# CeBASE整理的软件缺陷问题

- 软件发布后发现和修复缺陷的成本大幅增加；
- 软件项目返工工作量比例居高不下，但随着过程成熟度的提高而减少；
- 软件工程中返工工作量与缺陷、软件失效与缺陷，以及缺陷在软件模块中的分布都是符合2.8原则的；
- 同行评审、评审准备工作等有助于发现更多的缺陷；
- 有经验的开发人员可以显著减少缺陷的引入率。

CeBASE (Center for Empirically-Based Software Engineering)

[<http://www.CeBASE.org>]

Shull F, Basili V, Boehm B, et al. What We Have Learned About Fighting Defects[J]. 2003:249-258.



# 第1讲 软件与软件工程

## 1.3 软件生存期模型

## App死亡潮：400万应用僵尸超八成周期仅10月

艾媒咨询分析称，目前我国主要应用商店的应用规模已累计超过400万个。但App的生命周期平均只有十个月，85%的用户会在1个月内将其下载的应用程序从手机中删除，而到了5个月后，这些应用程序的留存率仅有5%。

2015年04月06日09:12 理财周报

<http://tech.sina.com.cn/i/2015-04-06/doc-iavxeafs4632525.shtml>

- ① 软件的生命周期是什么？
- ② 软件生命周期包括哪些主要的阶段？

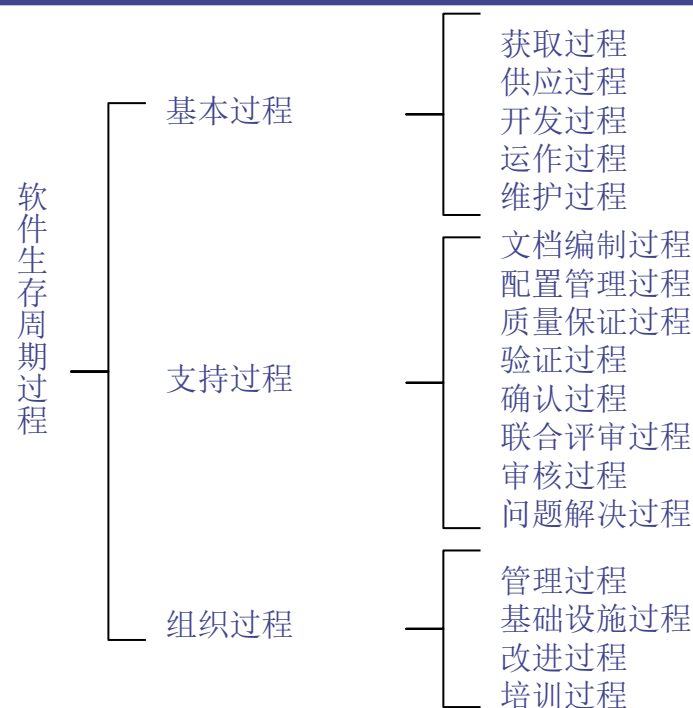
# 软件生存周期（Software Life Cycle）

GB/T 11457-2006 《信息技术——软件工程术语》

- 软件产品从构思开始至软件不再可用结束时的时间周期。
- 软件生存周期典型的地包括：需要阶段、设计阶段、实现阶段、测试阶段、安装和验收阶段、操作和维护阶段，有时还包括退役阶段。

ISO/IEC 12207:1995 《信息技术-软件生存周期过程》把软件生存周期的各个过程分成三类：

- 基本生存周期
- 支持生存周期
- 组织的生存周期



# 主要的软件生存期模型(life cycle model)

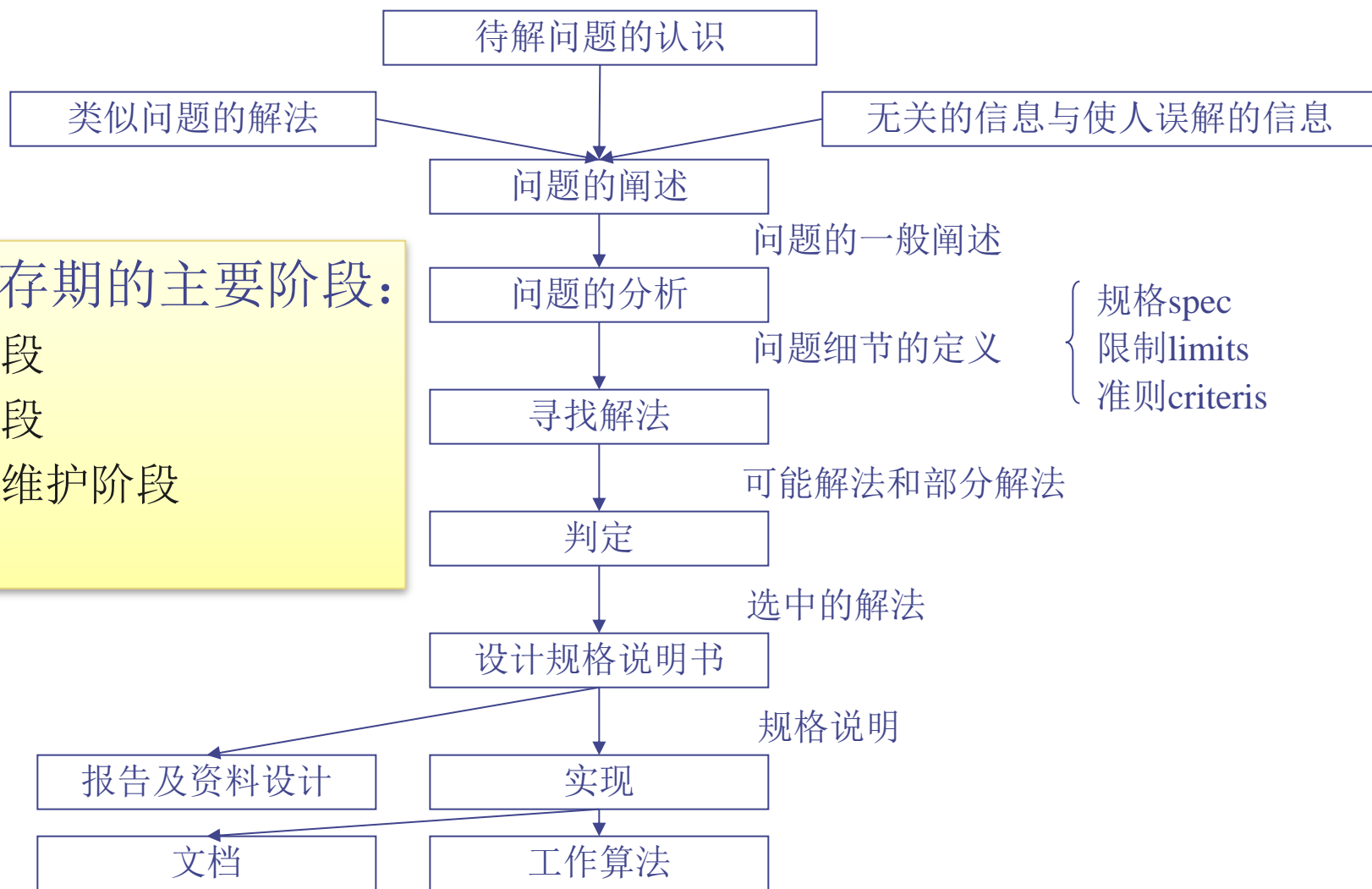
- 瀑布模型
- 原型模型
- 快速应用开发（RAD）模型
- 螺旋模型
- 增量模型和迭代模型
- 构件组装模型
- 并发模型
- 测试驱动（TDD）型的软件开发
- Rational的RUP和UML
- 形式化描述技术FDT
- 敏捷方法（XP）



软件生存期模型意义：

- 软件开发模型是软件开发全部过程、活动和任务的结构框架。
- 软件开发模型能清晰、直观地表达软件开发全过程，明确规定了要完成的主要活动和任务，用来作为软件项目开发的基础。

# 一般问题解决过程



软件生存期的主要阶段:

- 1.准备阶段
- 2.开发阶段
- 3.使用和维护阶段

# 软件生存期主要任务： (1/2)

## 1. 问题定义

- 这个阶段必须回答的关键问题是：“要解决的问题是什么”。

## 2. 可行性研究

- 这个阶段要回答的关键问题是：“上一个阶段所确定的问题是否有行得通的解决办法”。

## 3. 需求分析

- 这个阶段的任务仍然不是具体地解决客户的问题，而是准确地回答“目标系统必须做什么”这个问题。
- 这个阶段的另外一项重要任务，是用正式文档准确地记录对目标系统的需求，这份文档通常称为规格说明(specification)。

## 4. 概要设计

- 这个阶段的基本任务是，概括地回答“怎样实现目标系统?”这个问题。
  - ◆ 首先，应该设计出实现目标系统的几种可能的方案。
  - ◆ 概要设计的另一项主要任务就是设计程序的体系结构，也就是确定程序由哪些模块组成以及模块间的关系。

## 5. 详细设计

- 这个阶段的任务是设计出程序的详细规格说明。就是把解法具体化，也就是回答“应该怎样具体地实现这个系统”这个关键问题。

# 软件生存期主要任务： (2/2)

## 6. 编码和单元测试

- 这个阶段的关键任务是写出正确的容易理解、容易维护的程序模块。

## 7. 综合测试

- 这个阶段的关键任务是通过各种类型的测试（及相应的调试）使软件达到预定的要求。

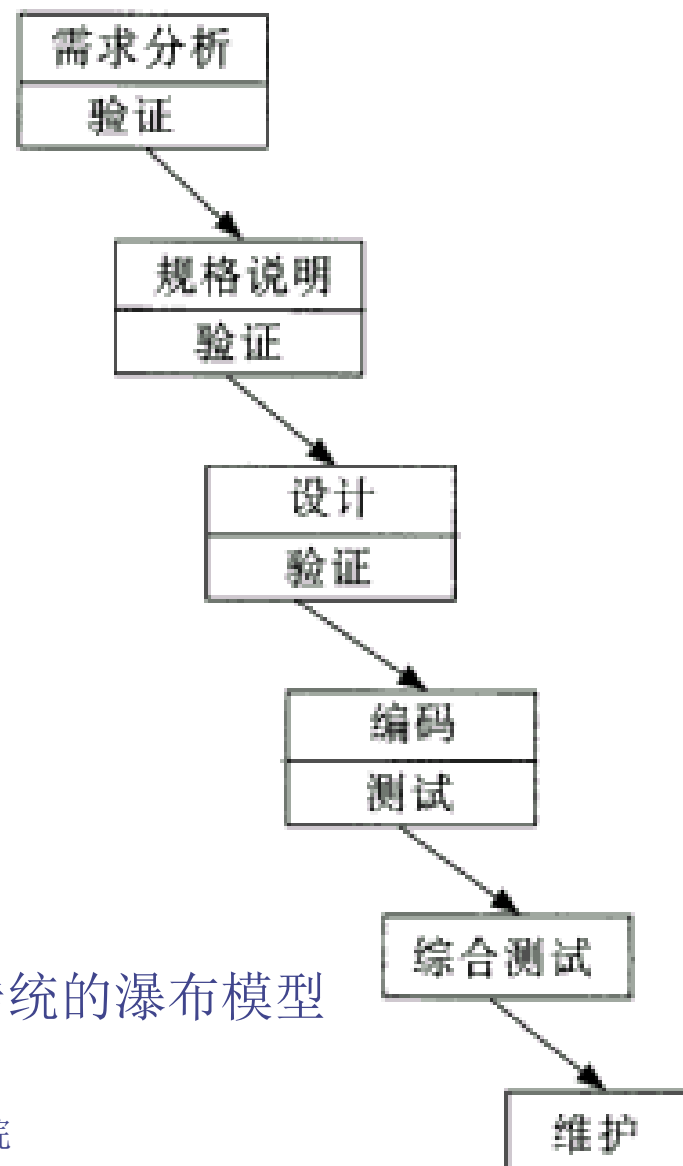
## 8. 软件维护

- 维护阶段的关键任务是，通过各种必要的维护活动使系统持久地满足用户的需要。
- 通常有四类维护活动：
  - ◆ 改正性维护，也就是诊断和改正在使用过程中发现的软件错误；
  - ◆ 适应性维护，即修改软件以适应环境的变化；
  - ◆ 完善性维护，即根据用户的要求改进或扩充软件使它更完善；
  - ◆ 预防性维护，即修改软件为将来的维护活动预先做准备。



# 瀑布模型（Waterfall Model）

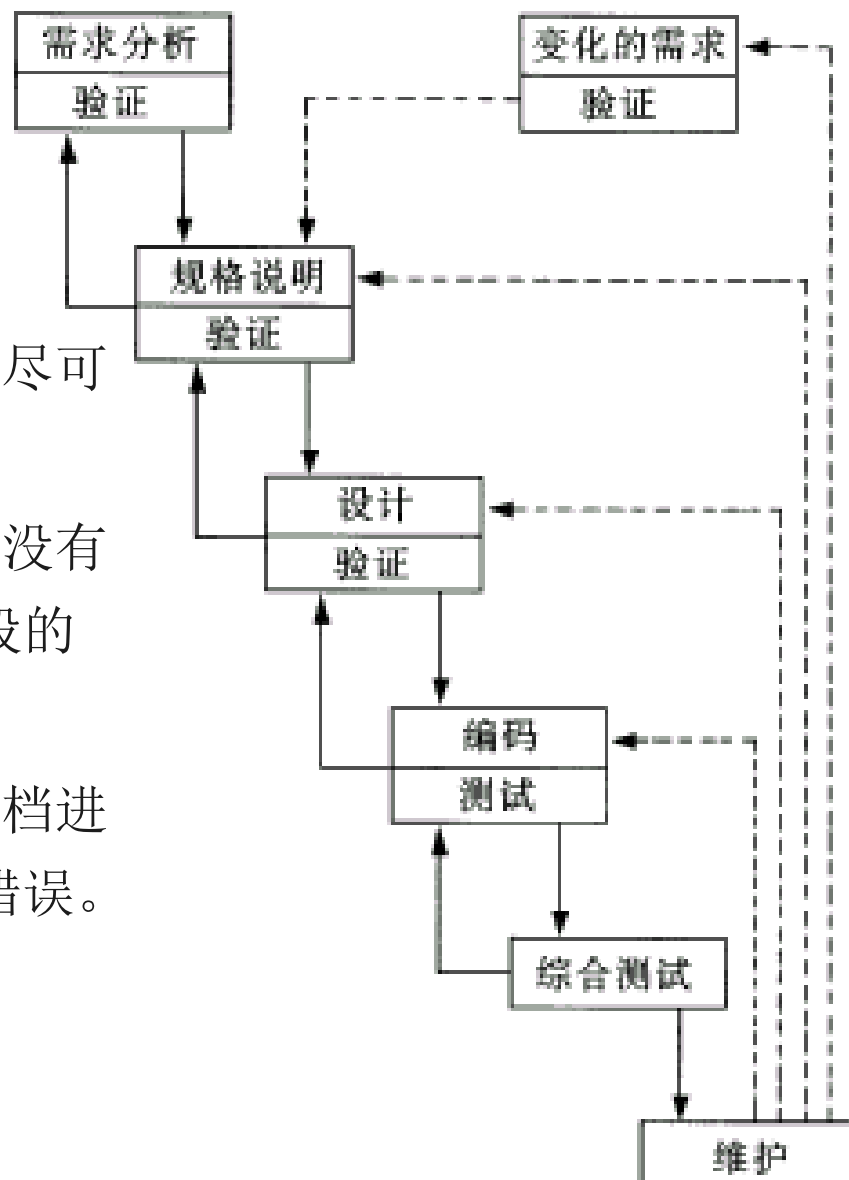
- 1970年，Winston Royce提出软件开发的瀑布模型。
- 在20世纪80年代之前，瀑布模型一直是唯一被广泛采用的生命周期模型，现在它仍然是软件工程中应用最广泛的过程模型之一。
- 按照传统的瀑布模型来开发软件，有如下几个特点：
  - (1) 阶段间具有顺序性和依赖性
  - (2) 推迟实现的观点
  - (3) 质量保证的观点



# 实际的瀑布模型

## 瀑布型的特点：

- 清楚地区分逻辑设计与物理设计，尽可能推迟程序的物理实。
- 每个阶段都必须完成规定的文档，没有交出合格的文档就是没有完成该阶段的任务。
- 每个阶段结束前都要对所完成的文档进行评审，以便尽早发现问题，改正错误。

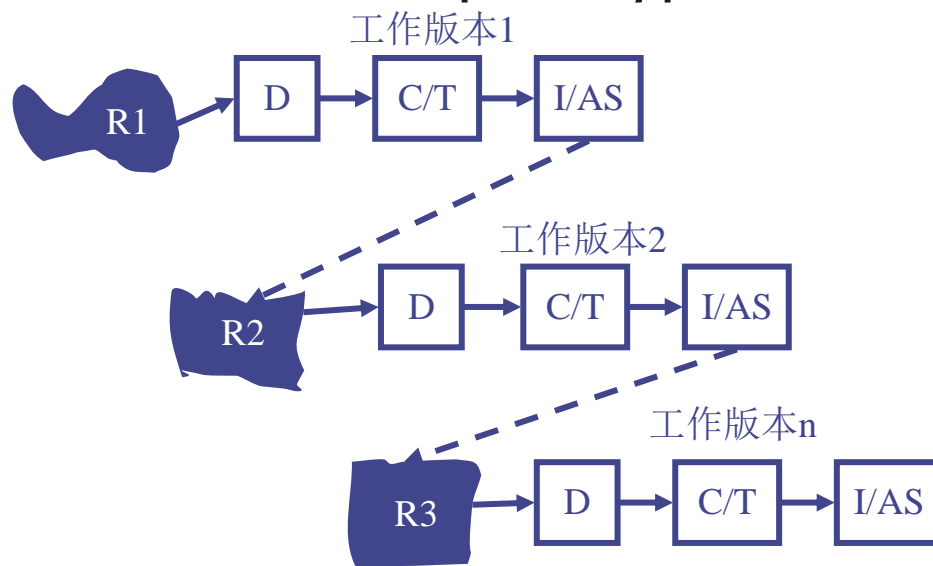


# 演化模型（evolutionary model）

- 演化模型主要针对事先不能完整定义需求的软件项目，此模型可以减少风险。
- 为了探索可行性和弄清需求，作第一次试验开发，以取得有效的反馈信息，来支持软件的最终设计和实现。
- 第一次试验开发出来的软件称为原型（prototype）。

- 演化模型的多种形式：

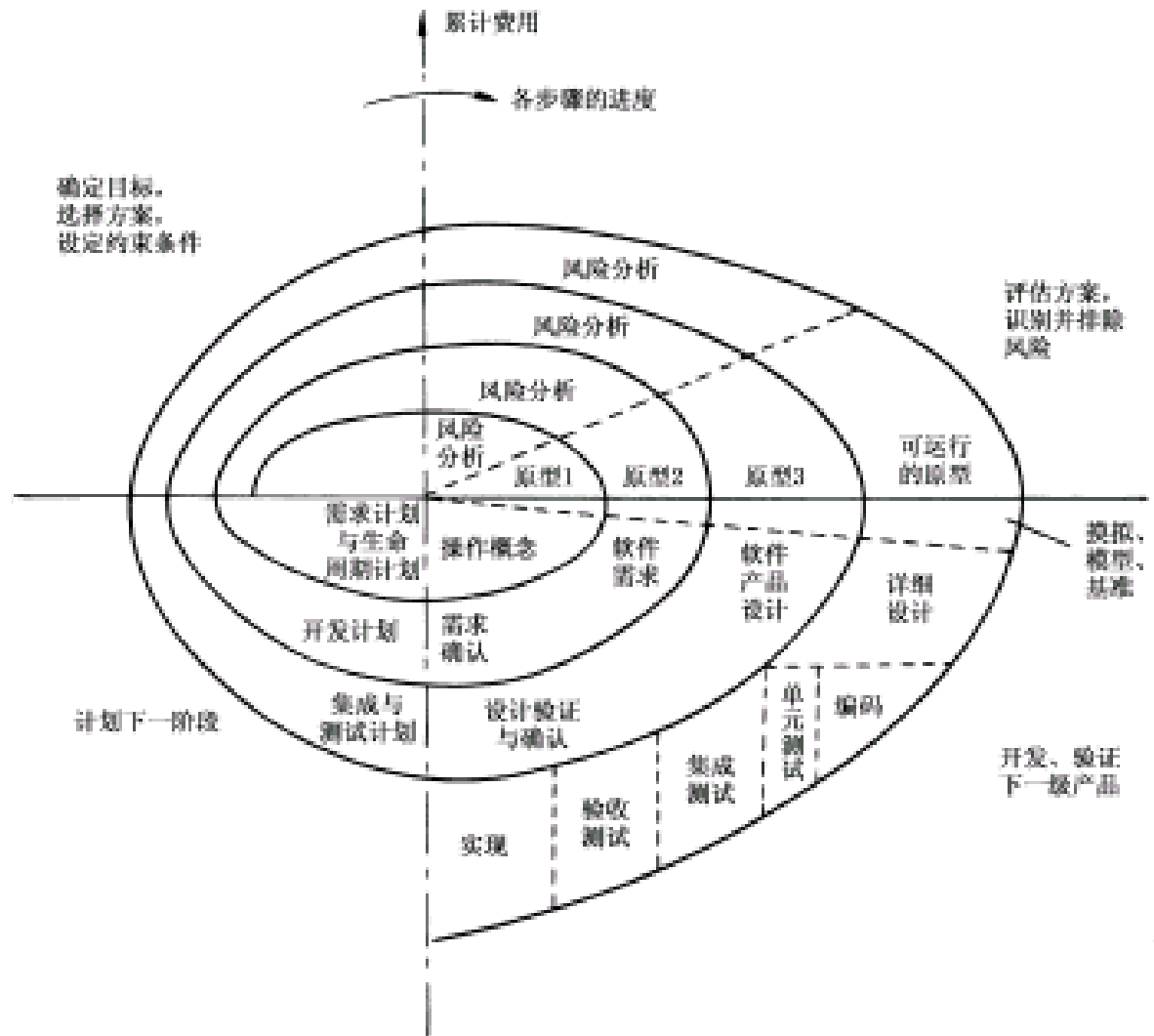
- 丢弃型
- 样品型
- 渐增式演化型



R: 需求 C/T: 编码/测试 D: 设计 I/AS: 安装和验收支持

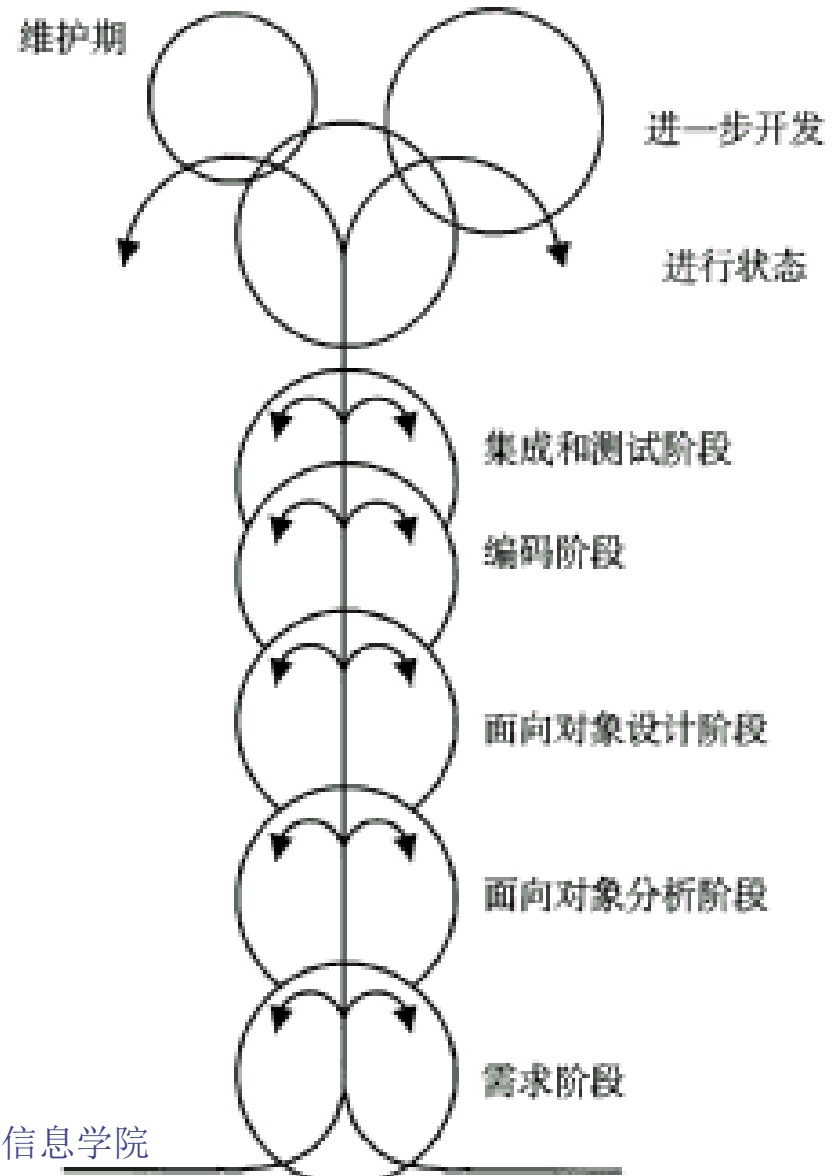
# 螺旋模型（Spiral Model）

- 螺旋模型是瀑布模型与演化模型结合，并增加两者所忽略的风险分析。
- 螺旋模型通常用来指导大型软件项目开发，它将开发划分为：
  - 制定计划
  - 风险分析
  - 实施开发
  - 客户评估



# 喷泉模型（Water Fountain Model）

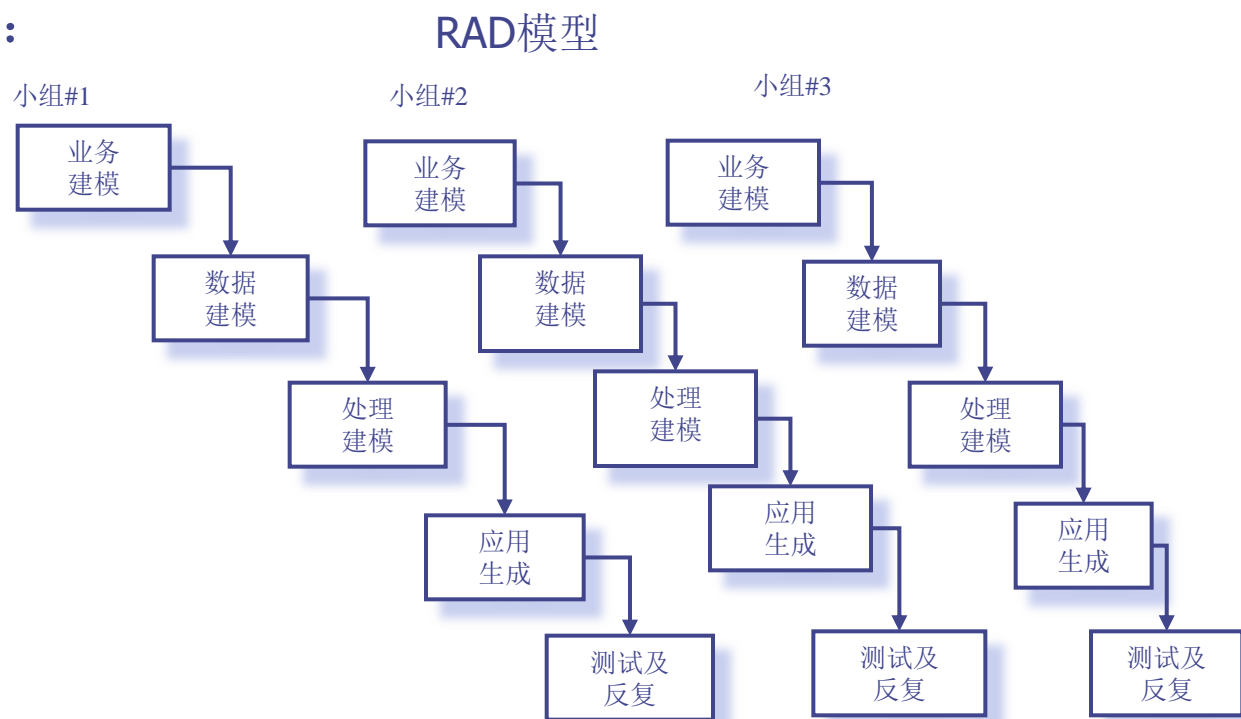
- 喷泉模型主要用于采用对象技术的软件开发项目，“喷泉”这个词体现了面向对象软件开发过程迭代和无缝的特性。
- 为避免使用喷泉模型开发软件时开发过程过分无序，应该把一个线性过程（例如，快速原型模型或图中的中心垂线）作为总目标。但是，同时也应该记住，面向对象范型本身要求经常对开发活动进行迭代或求精。



# RAD模型

- RAD（Rapid Application Development）模型使用基于构件的建造方法，强调极短的开发周期。
- 如果需求理解得很好，且约束了项目的范围，RAD过程使得一个开发组能够在很短时间内创建出功能完善的系统。
- RAD模型的主要阶段：

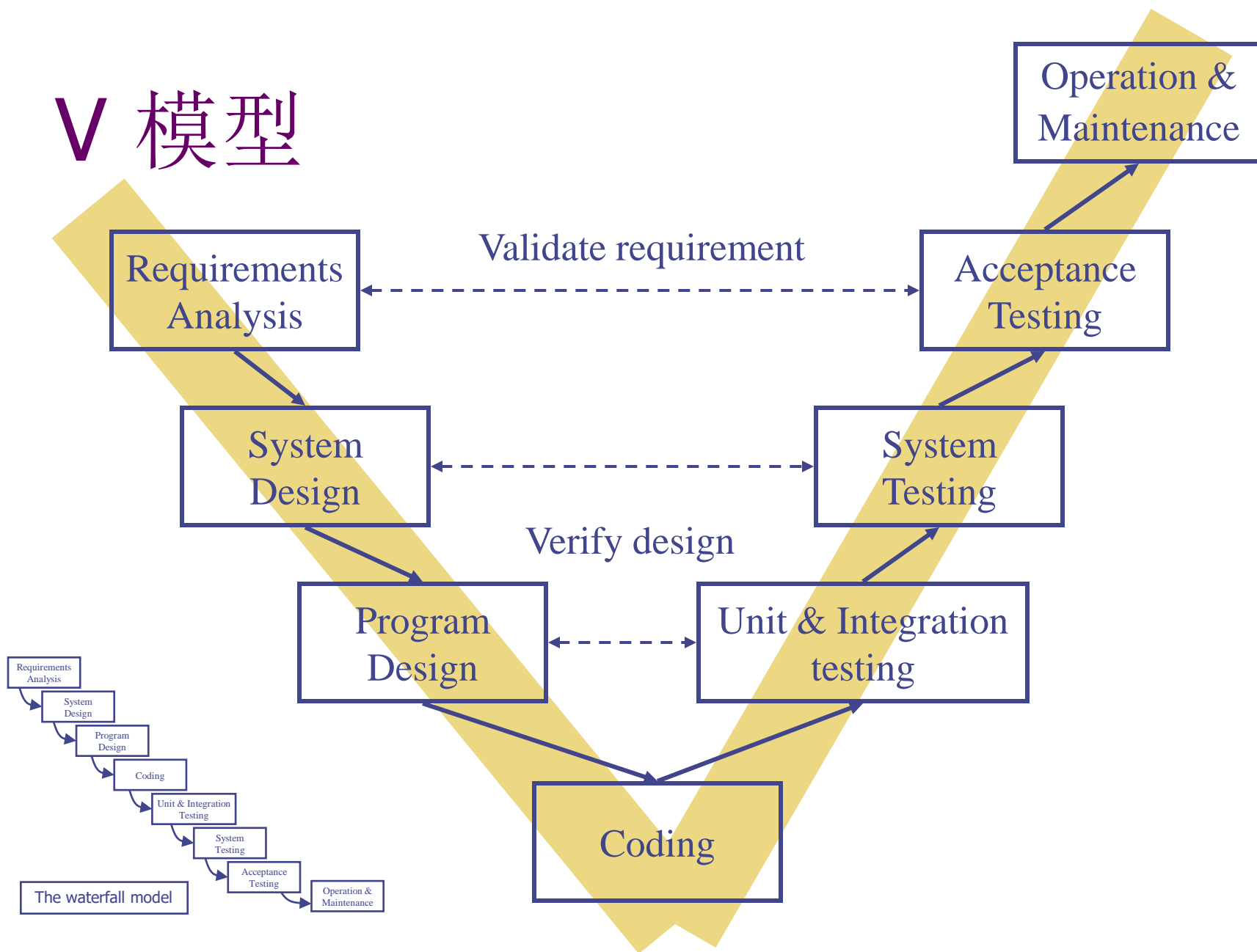
- 业务建模
- 数据建模
- 处理建模
- 应用生成
- 测试及反复



# RAD不适用项目：

- 如果系统难以被适当地模块化，那么建造RAD所需的构件就会有问题。
- 如果高性能是一个指标，且该指标必须通过调整接口使用其适应系统构件才能赢得，RAD方法就可能失败。
- RAD不适合技术风险很高的情况，当一个新应用要采用很多新技术，或当新构件要求已有计算机程序有高可靠性时，RAD就可能失败。
- RAD的缺陷：
  - 对于大型项目，RAD需要足够的人力资源以创建足够的RAD组。
  - RAD要求承担必要的快速活动的开发者和用户在一个很短的时间框架下完成一个系统。如果任何一方没有完成约定，都会导致RAD项目失败。

# V 模型





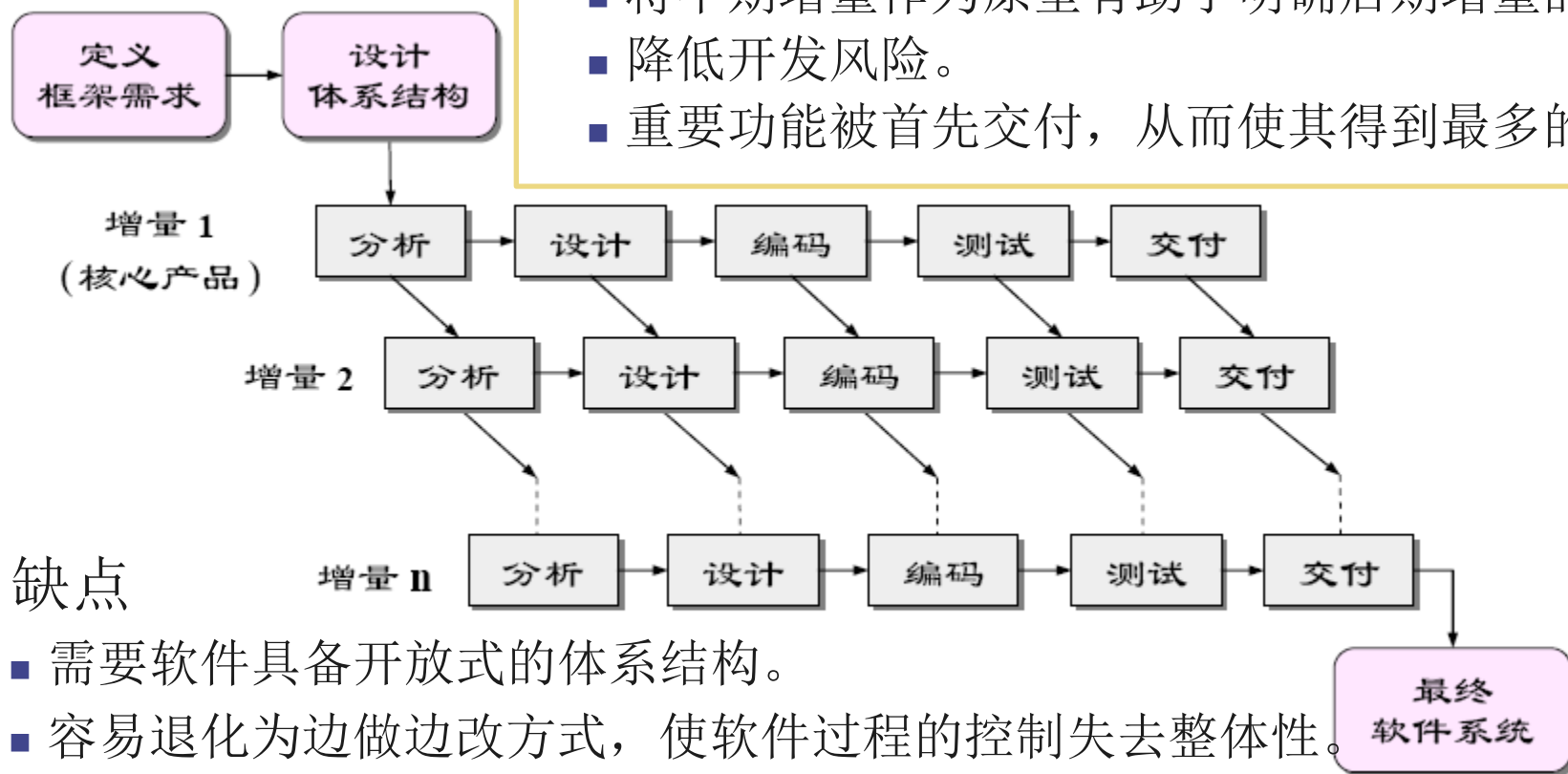
# V模型的特点

- The V model is a variation of the waterfall model that demonstrates how the testing activities are related to analysis and design.
- Coding forms the point of the V, Analysis and design on the left, testing and maintenance on the right.
- V model key:
  - The model's linkage of the left side with the right side of the V implies that if problem are found during verification and validation, the the left side of the V can be reexecuted to fix and improve the requirements, design, and code before the testing steps on the right side are reenacted.
  - The model makes more explicit some of the iteration and rework that are hidden in the waterfall depiction.

# 增量模型

## ● 优点

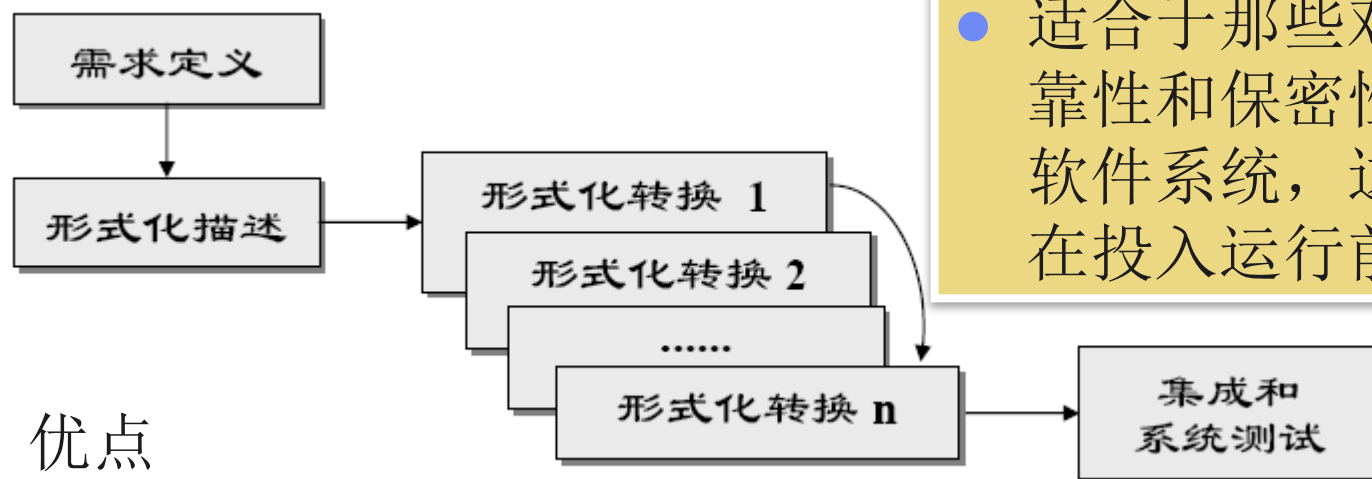
- 整个产品被分解成若干个构件逐步交付，用户可以不断地看到所开发软件的可运行中间版本。
- 将早期增量作为原型有助于明确后期增量的需求。
- 降低开发风险。
- 重要功能被首先交付，从而使其得到最多的测试。



## ● 缺点

- 需要软件具备开放式的体系结构。
- 容易退化为边做边改方式，使软件过程的控制失去整体性。
- 需求难以在增量实现之前详细定义，因此增量与需求的准确映射以及所有增量的有效集成可能会比较困难。

# 形式化方法模型



- 形式化方法模型是采用形式化的数学方法将系统描述转换成可执行的程序。
- 适合于那些对安全性、可靠性和保密性要求极高的软件系统，这些系统需要在投入运行前进行验证。

## ● 优点

- 由于数学方法具有严密性和准确性，形式化方法开发过程所交付的软件系统具有较少的缺陷和较高的安全性

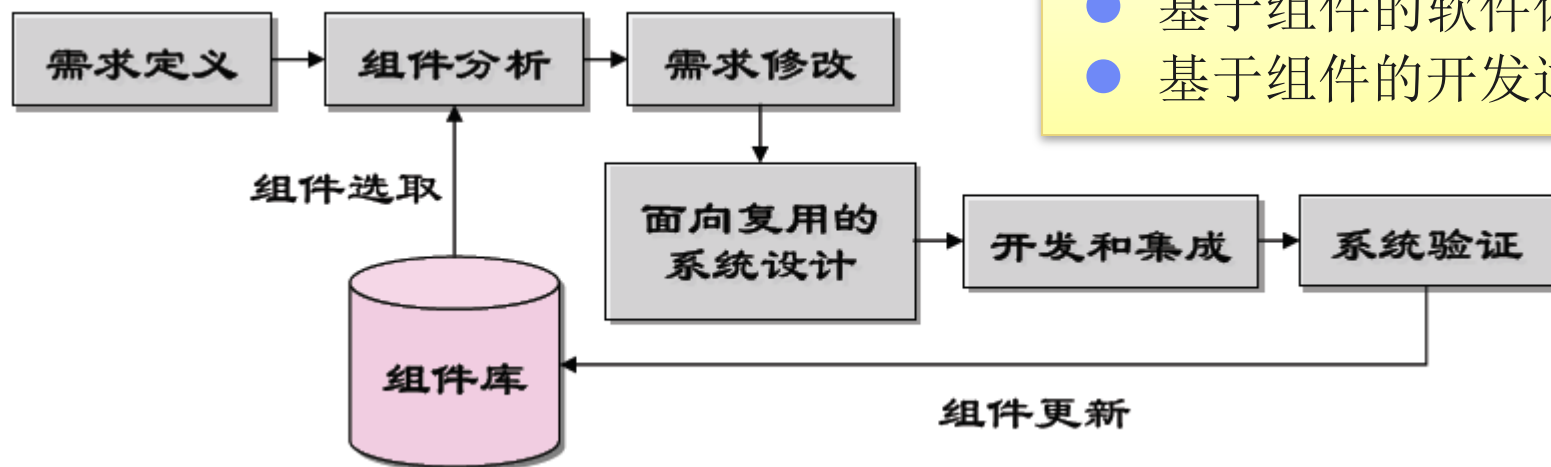
## ● 缺点

- 开发人员需要具备一定的技能并经过特殊训练
- 形式化描述和转换是一项费时费力的工作
- 现实应用的系统大多数是交互性强的软件，但是这些系统难以用形式化方法进行描述

# 基于组件的开发模型

组件开发技术的两个重要因素：

- 基于组件的软件体系结构
- 基于组件的开发过程



## 优点

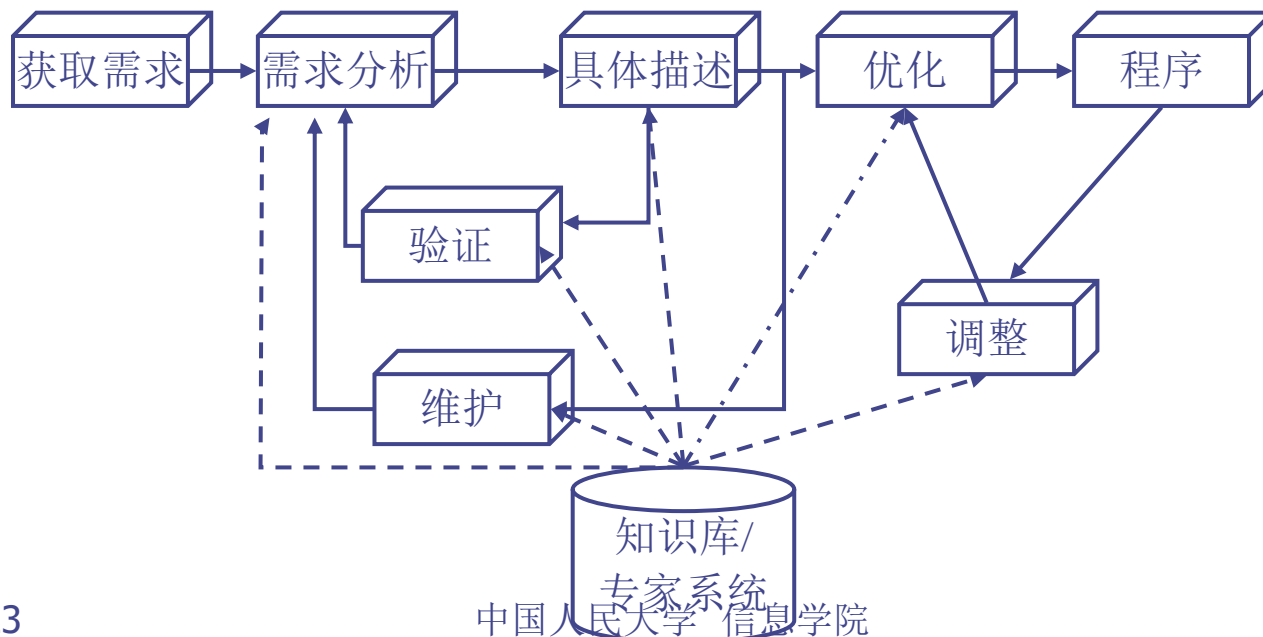
- 充分体现软件复用的思想
- 实现快速交付软件

## 缺点

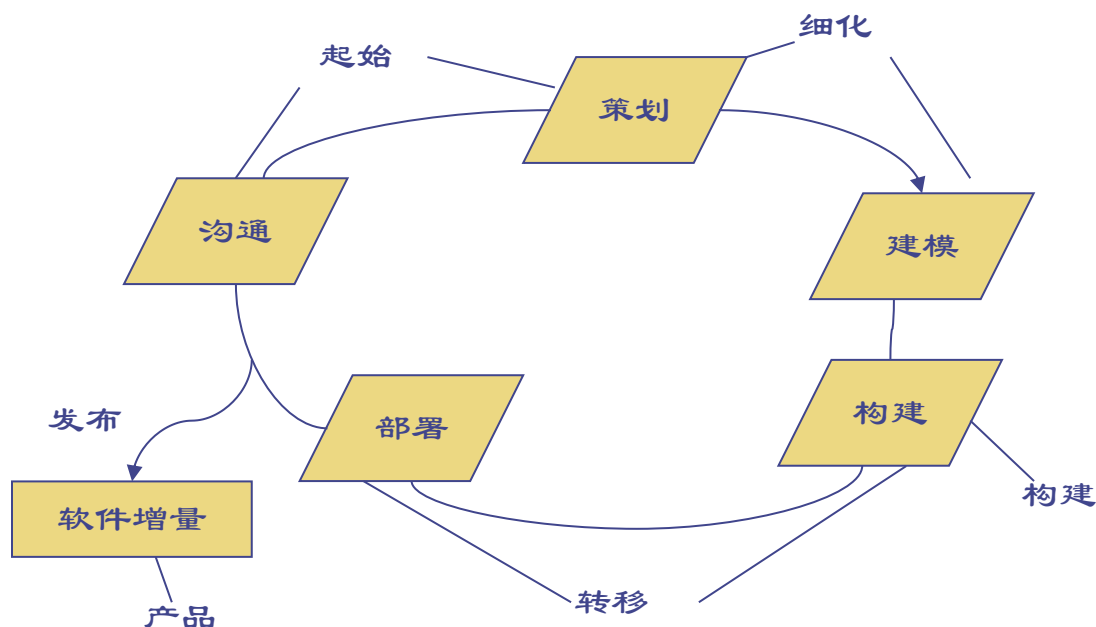
- 商业组件的修改受到限制，影响系统的演化

# 智能模型（Intelligent Model）

- 智能模型也称为基于知识的软件开发模型，它是知识工程与软件工程在开发模型上结合的产物。
- 智能模型与其它模型不同点是它的维护不在程序一级上进行，而是在功能规约一级上进行，从而可以把精力更加集中于具体描述的表达上，这样把问题的复杂性大为降低。

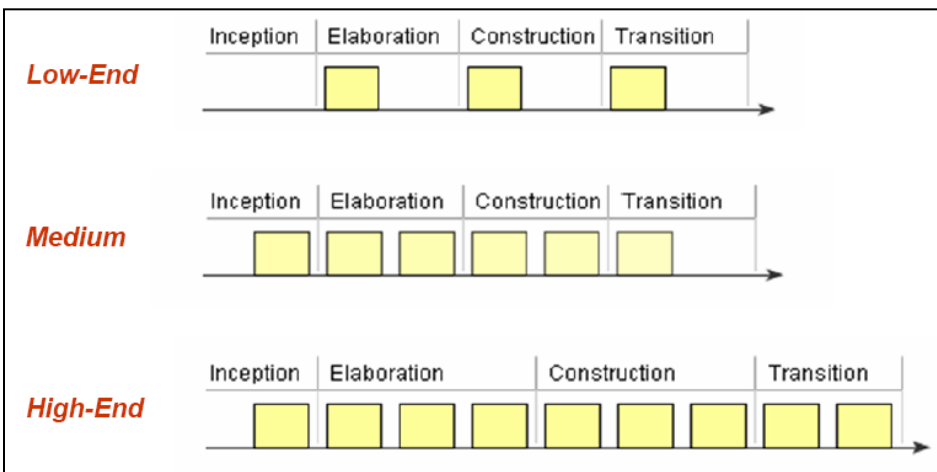
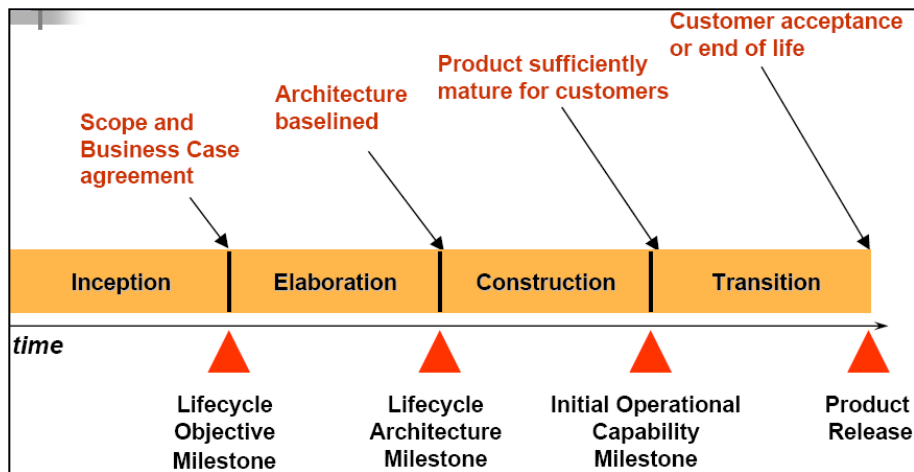
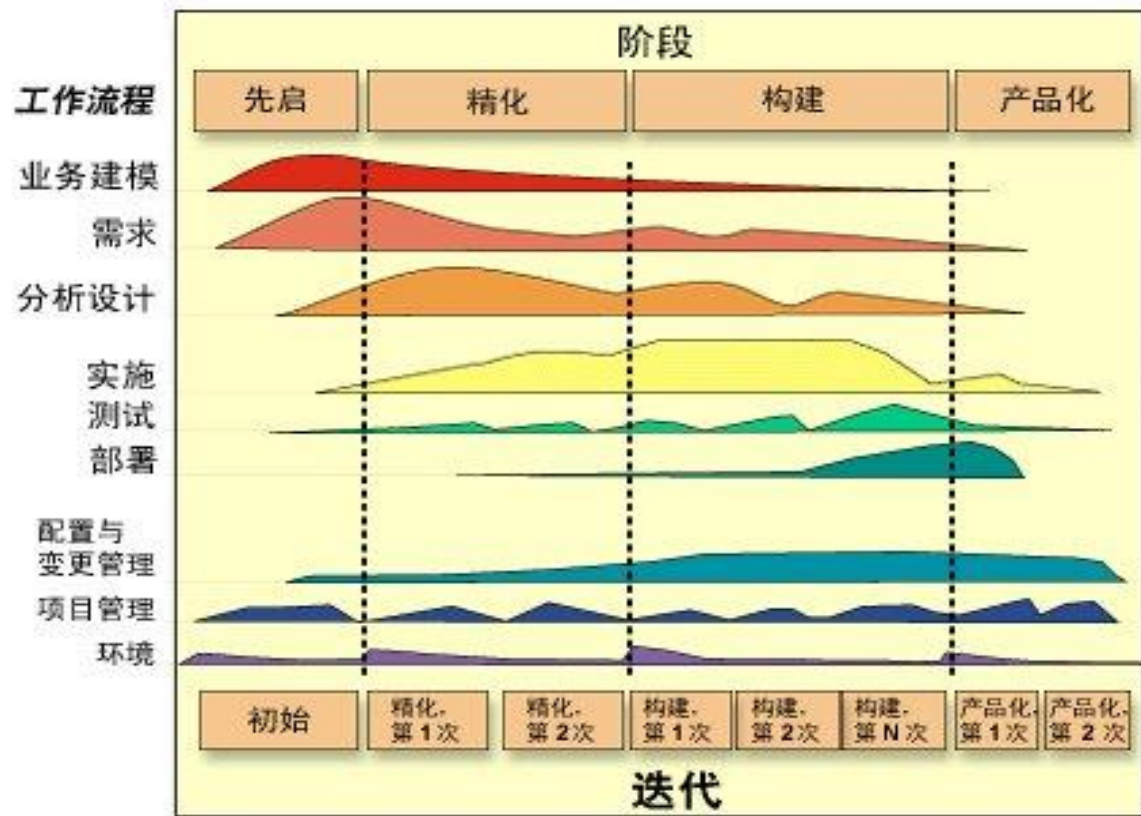


# 统一过程（RUP）



- ① **初始阶段**：建立一个业务案例或理由，确定核心需求，初步风险评估，原型
- ② **细化阶段**：建立架构，建立综合计划，交付的是系统行为模型（系统环境、场景、领域模型），以及基于领域模型的基线产品构想、开发计划、评估标准产品发布描述，其他还有用户手册、测试计划等
- ③ **构造阶段**：编程，产品包括可执行代码、系统和用户手册、部署计划、质量保证结果等工作产品
- ④ **发布阶段**：已完成的所有制品

# RUP



# RUP的4+1视图

- **逻辑视图**（Logical View）：设计的对象模型（使用面向对象的设计方法时）。
- **进程视图**（Process View）：捕捉设计的并发和同步特征。
- **物理视图**（Physical View）：描述了软件到硬件的映射，反映了分布式特性。
- **开发视图**（Development View）：描述了在开发环境中软件的静态组织结构。
- 架构的描述，即所做的各种决定，可以围绕着这四个视图来组织，然后由一些用例（use cases）或场景(scenarios)来说明，从而形成了第五个视图，即**用例视图**（use-case view）。



# 敏捷开发（agile development）

- 敏捷开发方法的宗旨是：**沟通、简化、反馈、激励**。
- 强调人的作用，构建起具有合作精神的、自组织的、有凝聚力量的团队。
- 敏捷开发是一种开发过程，而不是软件生命周期模型。



# 极限编程（eXtreme Programming）

- (1) 客户作为团队成员
- (2) 用户素材
- (3) 短交付周期
- (4) 验收测试
- (5) 结对编程
- (6) 测试驱动开发
- (7) 集体所有权
- (8) 持续集成
- (9) 可持续的开发速度
- (10) 开放的工作空间
- (11) 计划博奕
- (12) 重构
- (13) 隐喻

# 软件过程建模（UML）

- 软件过程建模模型
  - 基于UML的过程建模
  - 基于IDEF3的过程建模
  - 基于Agent的自适应软件过程模型
  - 基于SOA的软件过程模型

# 微软公司过程（MSF）

微软公司的开发管理原则

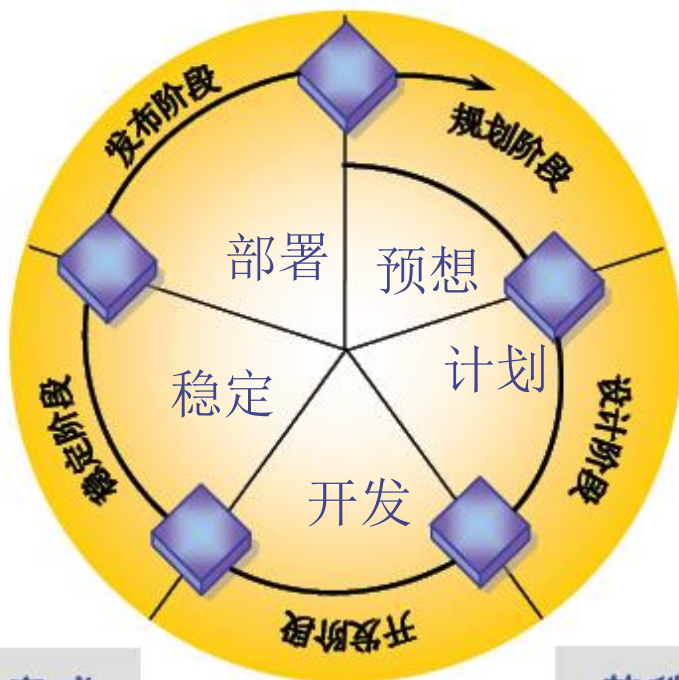
- 以目标驱动的开发过程
- 具有外部可见的里程碑
- 基于多版本的产品发布
- 并行协作的小型化团队
- 经常性的同步和稳定

MSF

最终产品发布

可发布版本  
准备就绪

项目的目标  
得到认可



代码开发完成

软件设计完成

# MSF的软件开发过程

- 规划阶段
  - 开展市场调查研究，结合公司战略形成产品的远景目标。
  - 愿景/范围说明书、风险评估说明书、项目组织结构说明书
- 设计阶段
  - 根据产品的远景目标，完成软件的功能规格说明和总体设计，并确定产品开发的主要进度。
  - 目标和功能说明书、风险管理计划、总体计划（进度）表、规范
- 开发阶段
  - 整个开发任务划分成若干个递进的阶段，并设置成M1、M2、……、Mn等内部里程碑，在每个里程碑都提交阶段性的工作成果。
  - 源代码、配置信息、功能说明书、使用支持、测试说明书（用例）
- 稳定阶段
  - 实行全面的内部和外部测试，最终形成可发布的RTM版本。
  - 版本、注释、使用支持、测试结果、源代码、项目文档、评审记录
- 发布阶段
  - 在确认产品质量符合发布标准之后，发布产品及其相关的消息。
  - 运营和支持IS、程序和过程、报告和日志、文档、总结、使用报告、下步计划

# MSF

- *M1*（1/3功能）
  - 开发（设计、编码）
  - 可用性实验 - 内部发布测试
  - 系统构建
  - 程序调试
  - 集成
  - 代码稳定
  - 缓冲时间
- *M2*（1/3功能）
  - 开发
  - 可用性实验
  - 内部发布测试
  - 系统构建
  - 程序调试
  - 集成
  - 代码稳定
  - 缓冲时间
- *M3*（1/3功能）
  - 开发
  - 可用性实验
  - 内部发布测试
  - 系统构建
  - 程序调试
  - 集成
  - 代码稳定
  - 缓冲时间
  - “零缺陷”发布
  - 最终产品发布

# MSF的软件开发过程

## ■ 递进式的开发策略

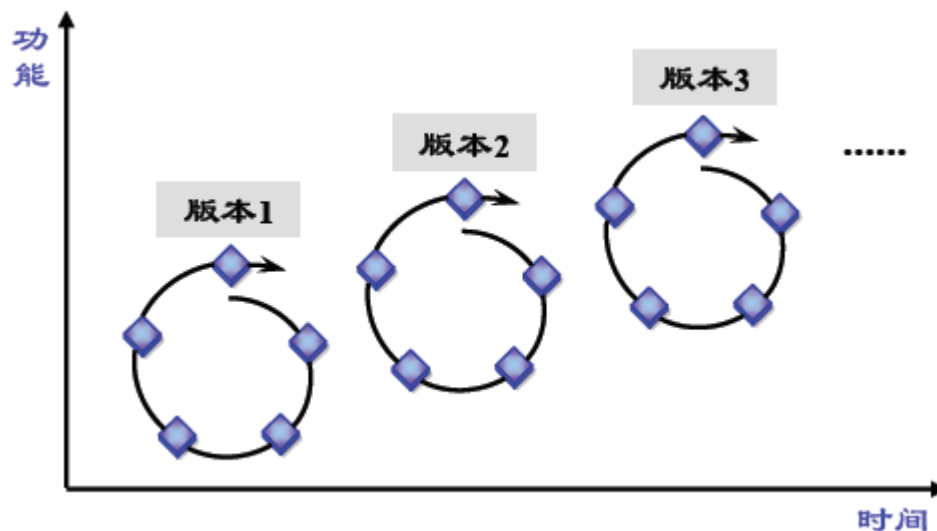
- 解决问题的及时性
- 不确定和变更因素的可控性
- 缩短产品上市周期

## ■ 缓冲时间

- 在每一个递进阶段，缓冲时间有利于开发人员应对变更、开发难题和时间延迟等问题

## ■ 并行与同步机制

- 小规模团队并行开发、每天的系统构建



# 实例：微软Window2000项目

- 人员5000
- 分四组：程序管理员、程序开发人员、测试员和合作伙伴，客户
- 时间：下午3：00～8：00构建产品，8：00～第二天9：00测试，9：00由程序管理人员写下测试报告以供下午开发人员参考。



# 软件生命周期模型选择问题

经过软件工程多年的经验积累，可以通过已知的项目特点来选择适合的生命周期模型，以此减少项目已知的风险，生命周期模型选择的原则如下：

系统环境特征	级别	适用模型
项目风险级	高	Spiral
需求理解程度	低	Spiral
需求更改频率	高	Iterative/Incremental
产品定义程度	高	Waterfall, RAD
用户参与程度	高	Iterative, RAD
发布日期压力	高	Iterative/Incremental
员工经验	低	Waterfall, Iterative
平台移植	经常	Iterative/Incremental

项目特点	瀑布模型	增量模型	原型+瀑布	迭代
需求规模	中、小	大、中	中、小	大、中、小
需求清晰度	很清晰	较清晰	不清晰	不清晰
客户信息化能力	高	中	中	中、低
客户界面要求	低	中	高	高
行业特点	行业规则简单或已掌握的行业	新行业的应用型项目或已掌握的行业	新行业的应用型项目	新行业的研发型或应用型项目
架构设计能力	有较成熟的架构设计能力	采用了新的架构设计	采用新的架构设计	不能确定的架构设计
技术要求	采用成熟的技术	采用较少的新技术	采用较少的新技术	采用较多的新技术
管理要求	工作产品的控制基线，需要有较高的可见度和可靠性。	每个增量阶段需要发布可操作产品。	原型阶段，需要控制原型制作的成本。	需要有较高的项目控制能力。每个迭代阶段需要有明确的目标。

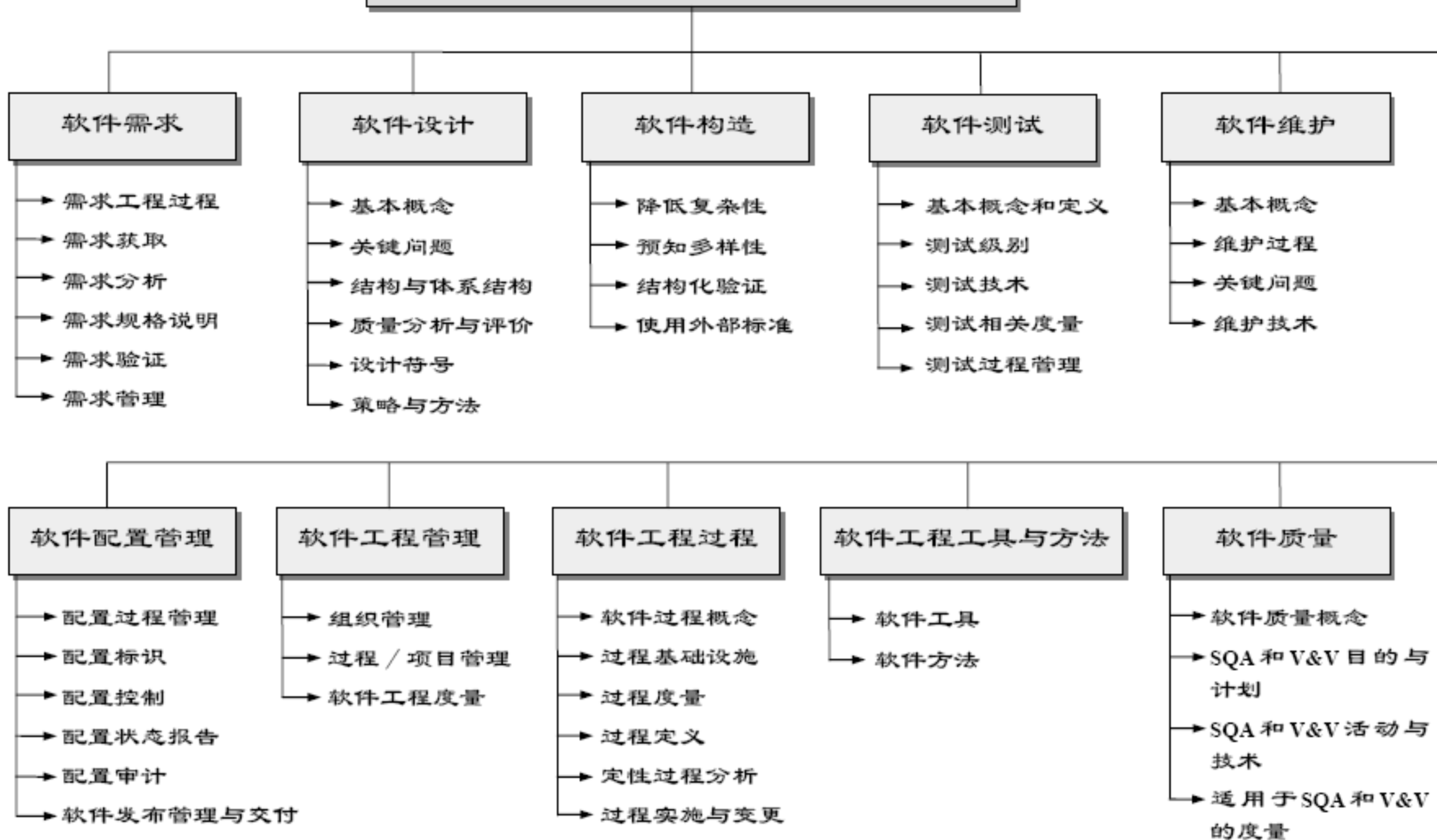
# 第1讲 软件与软件工程

## 1.4 软件工程知识体系

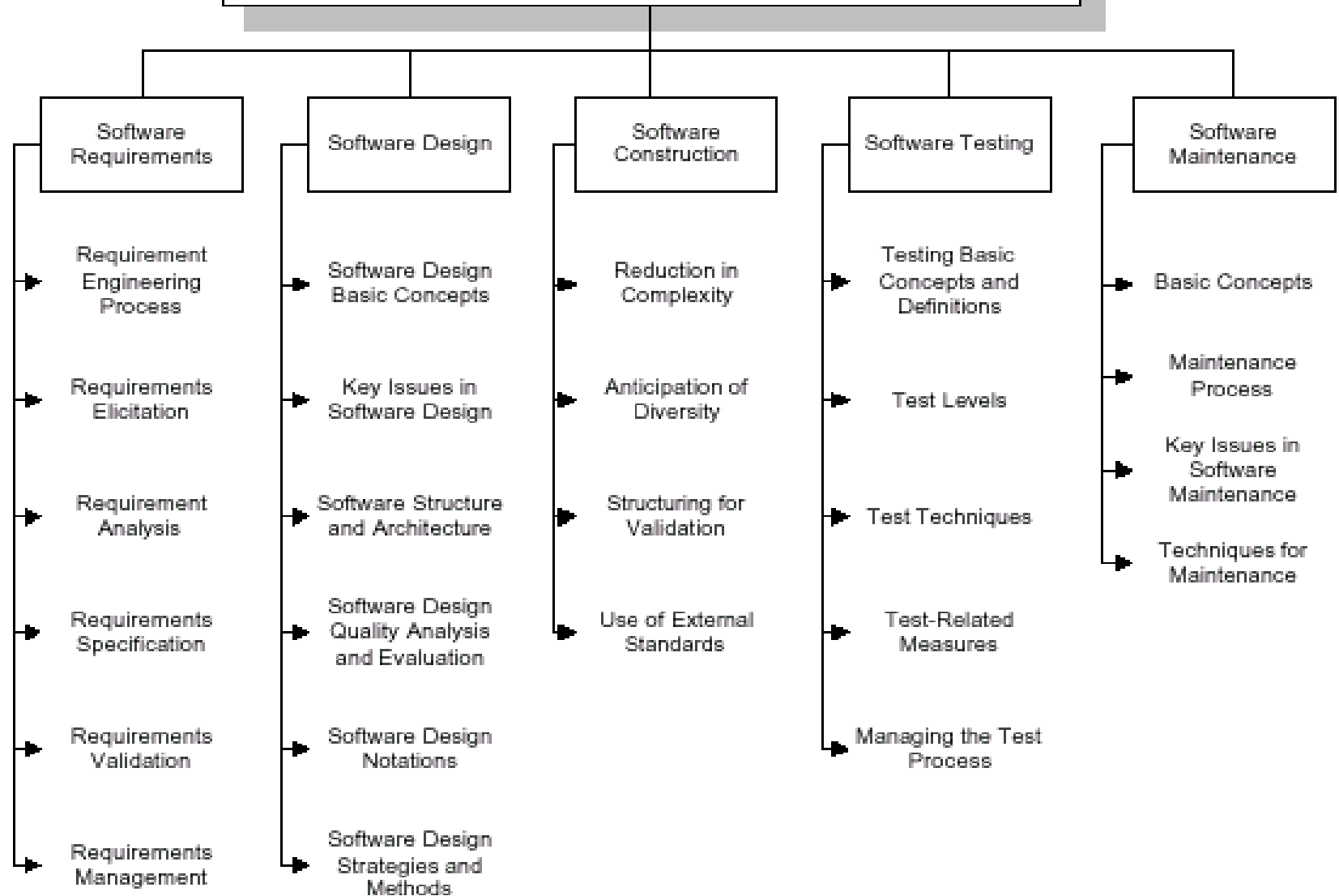
# **SWEBOK** (Software Engineering Body Of Knowledge)

- Software Requirements
- Software Design
- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration Management
- Software Engineering Management
- Software Engineering Process
- Software Engineering Tools and Methods
- Software Quality

## 软件工程知识体系 (SWEBOK)



# Guide to the Software Engineering Body of Knowledge (Trial Version)



(a)

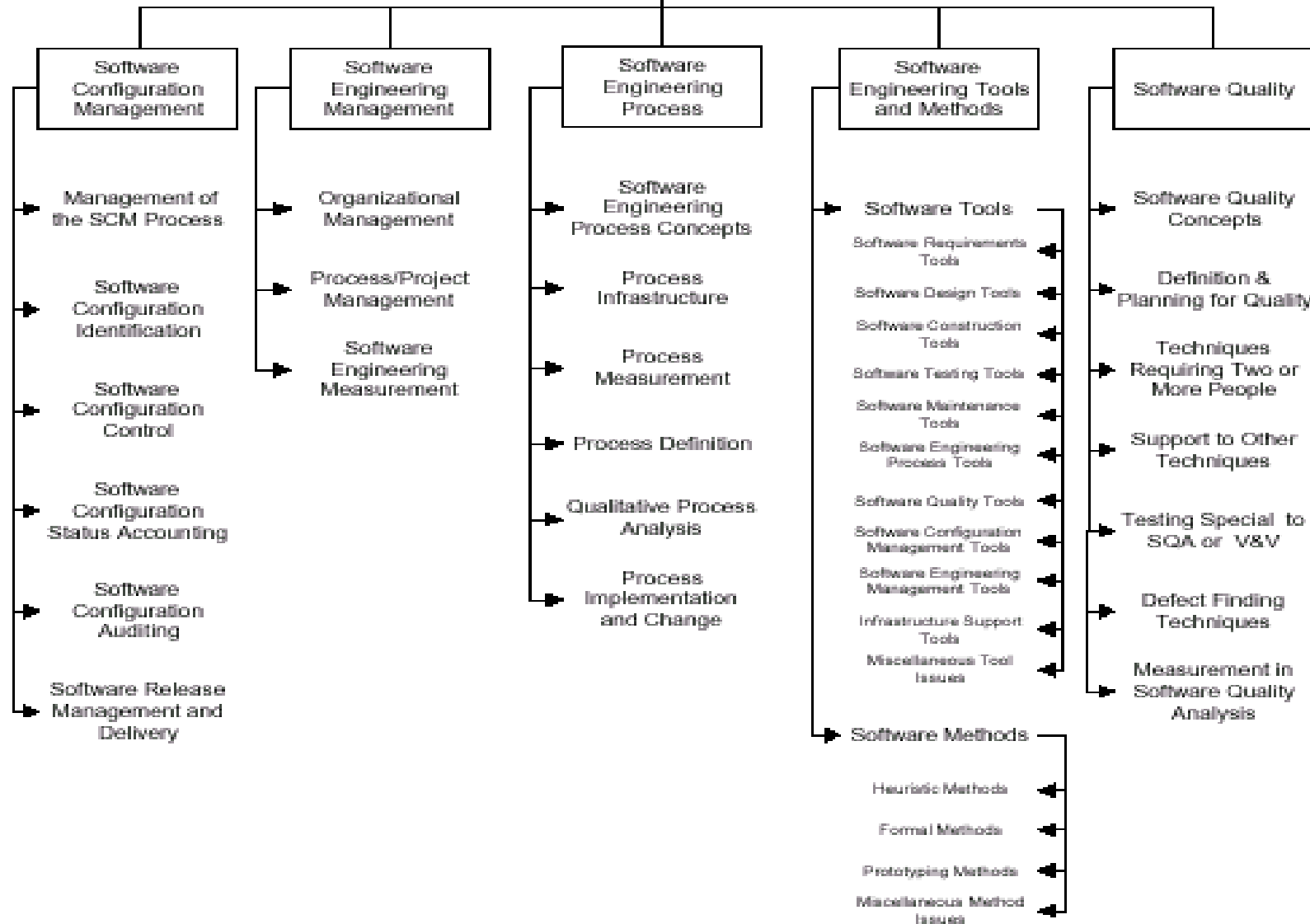
(b)

(c)

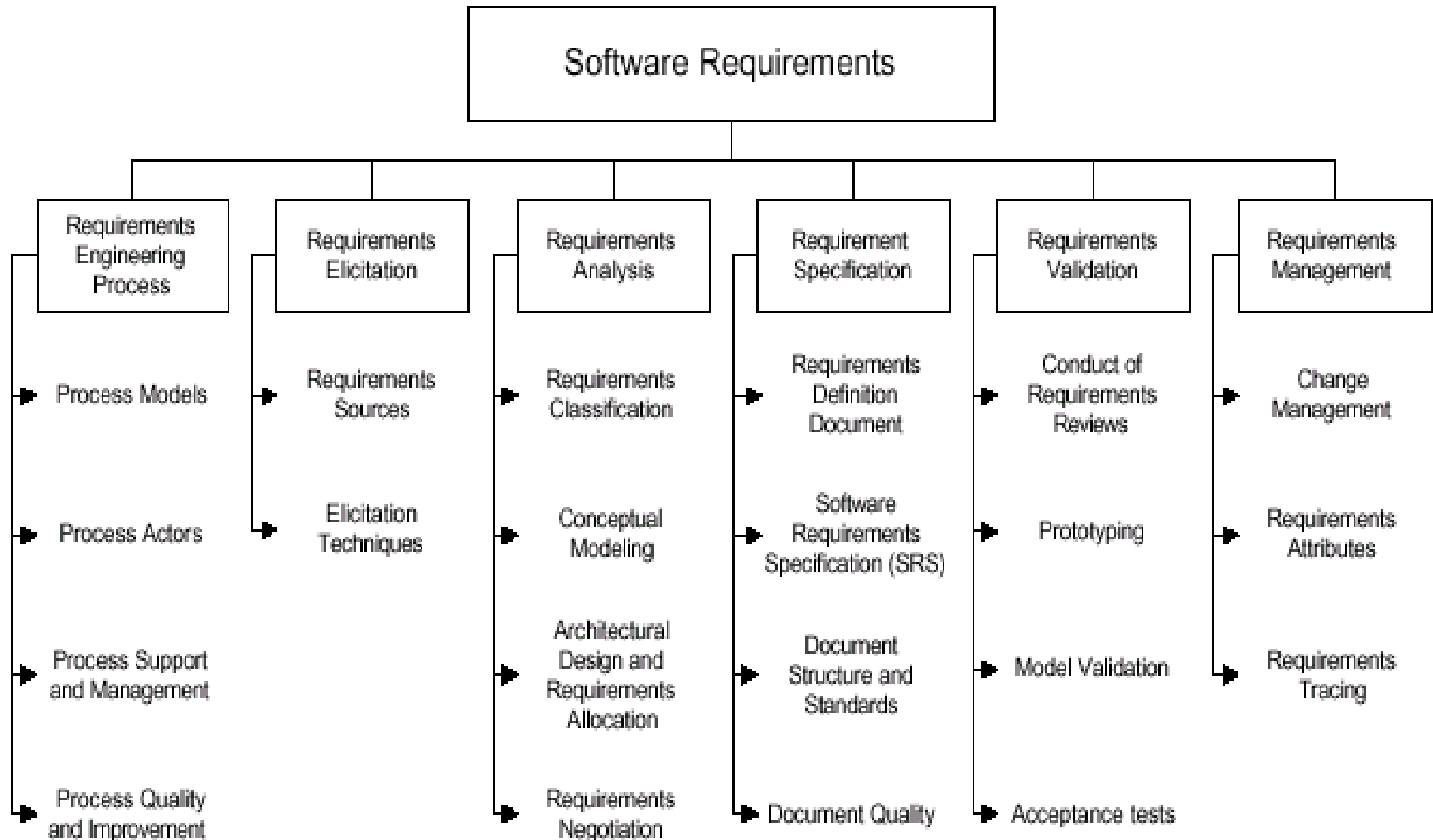
(d)

(e)

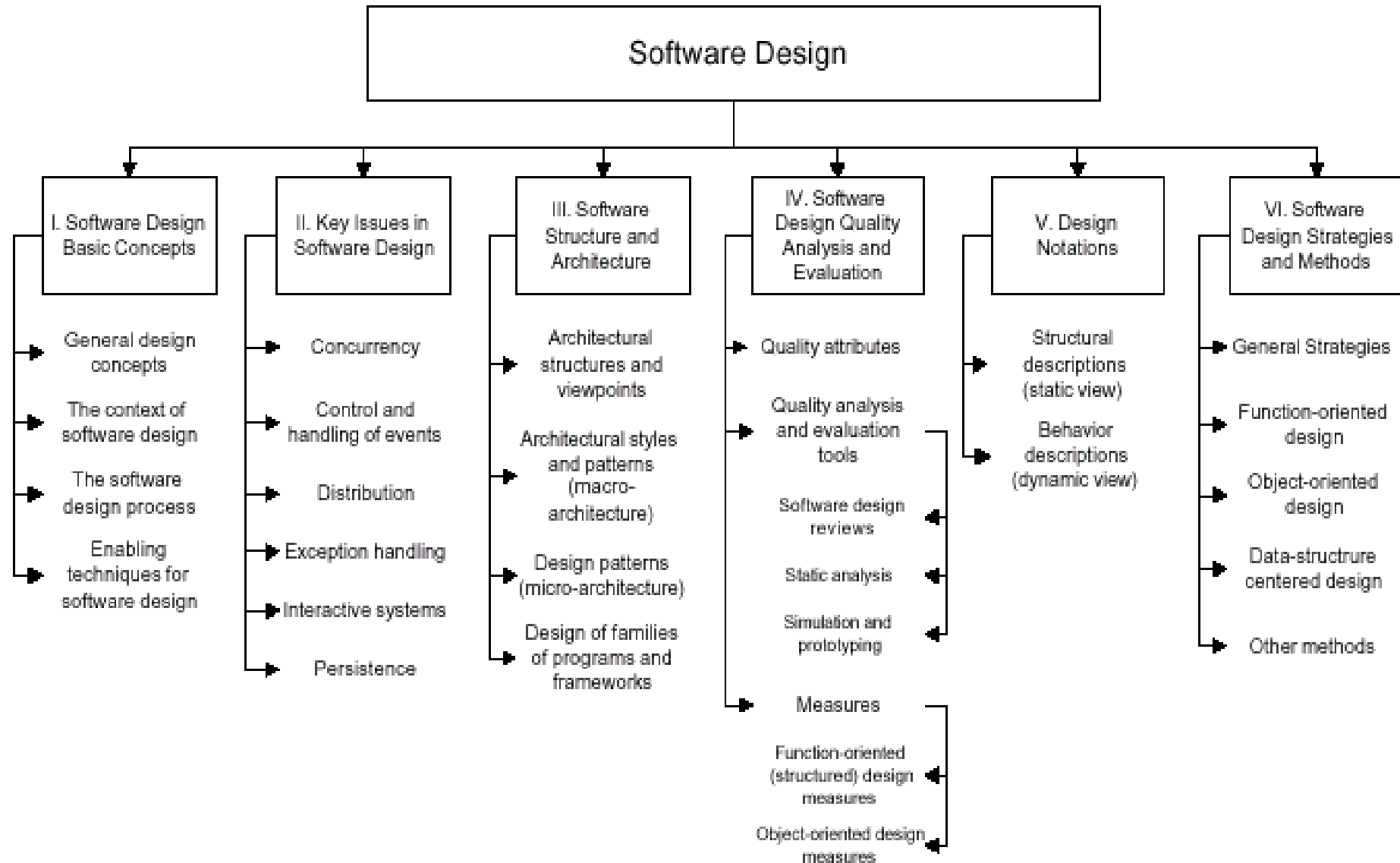
# Guide to the Software Engineering Body of Knowledge (Trial Version)



# Software Requirements

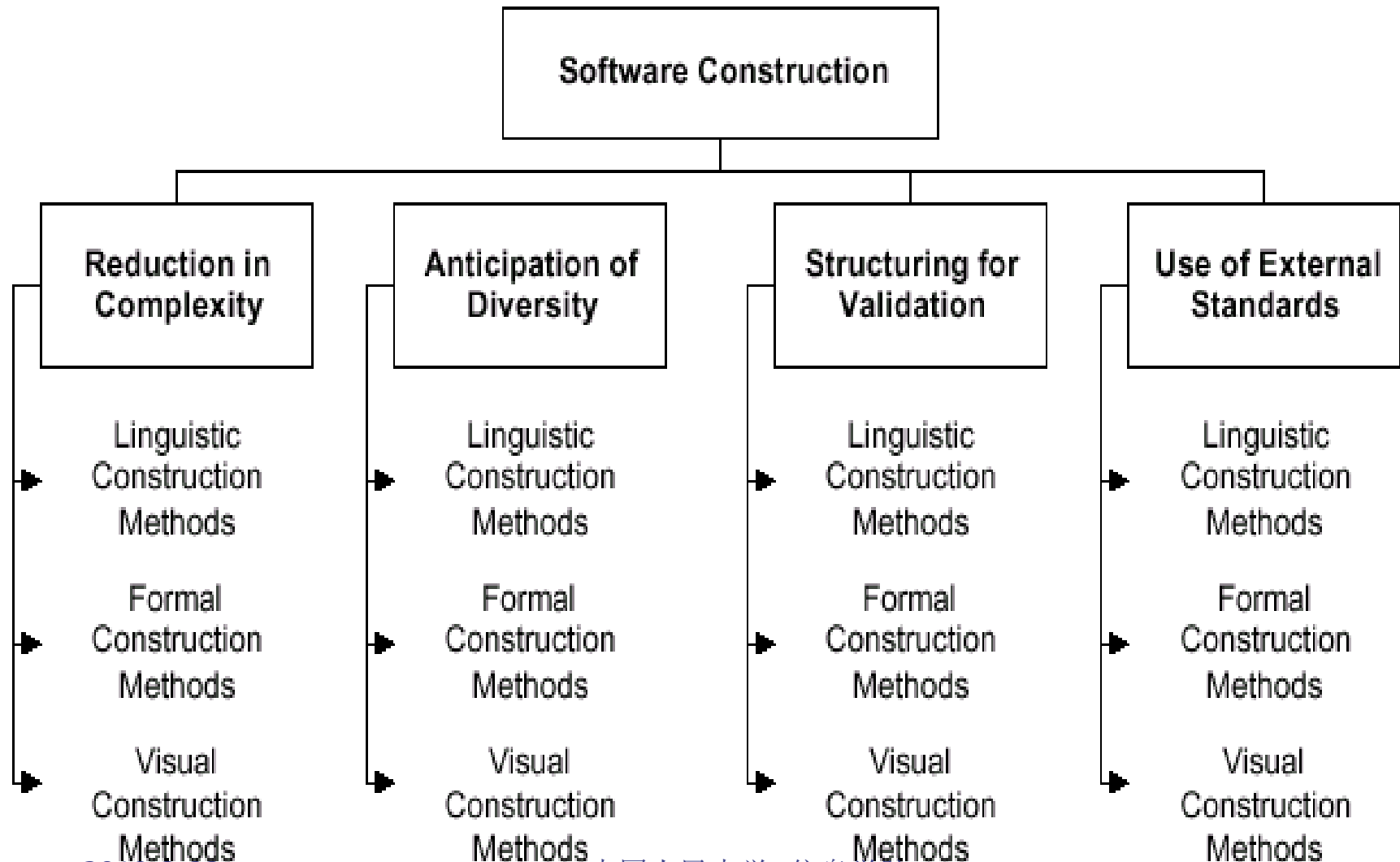


# Software Design

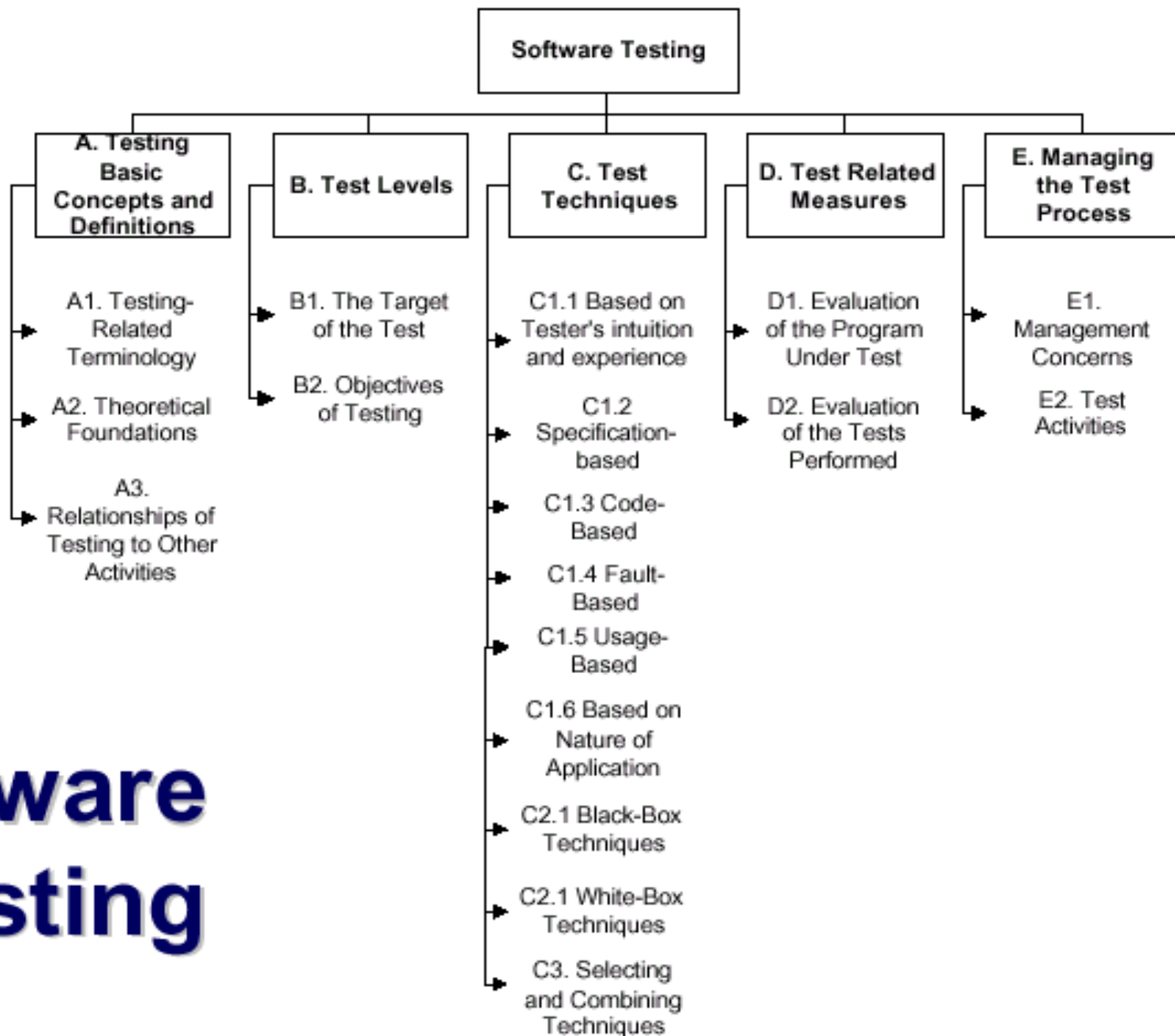




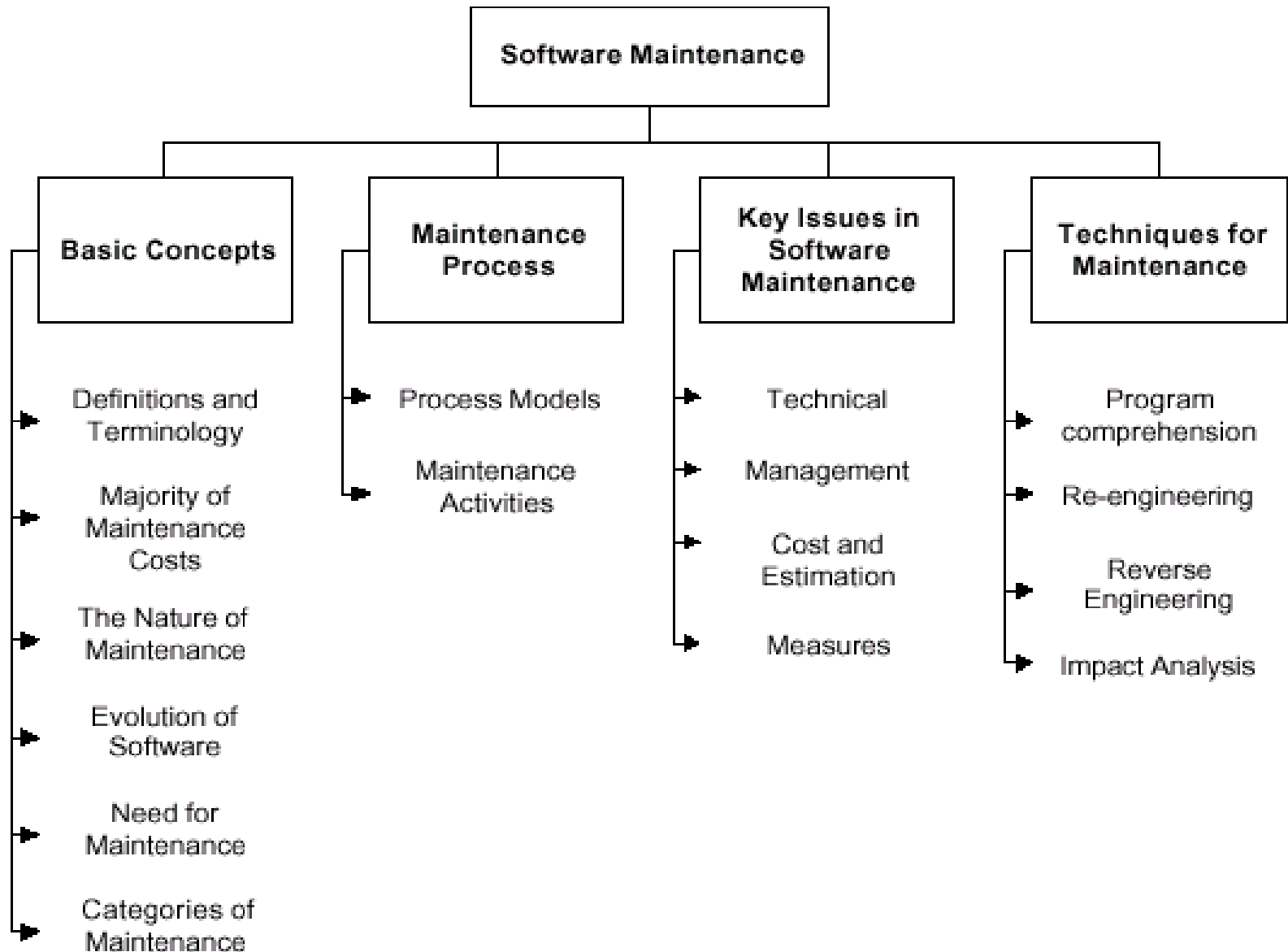
# Software Construction



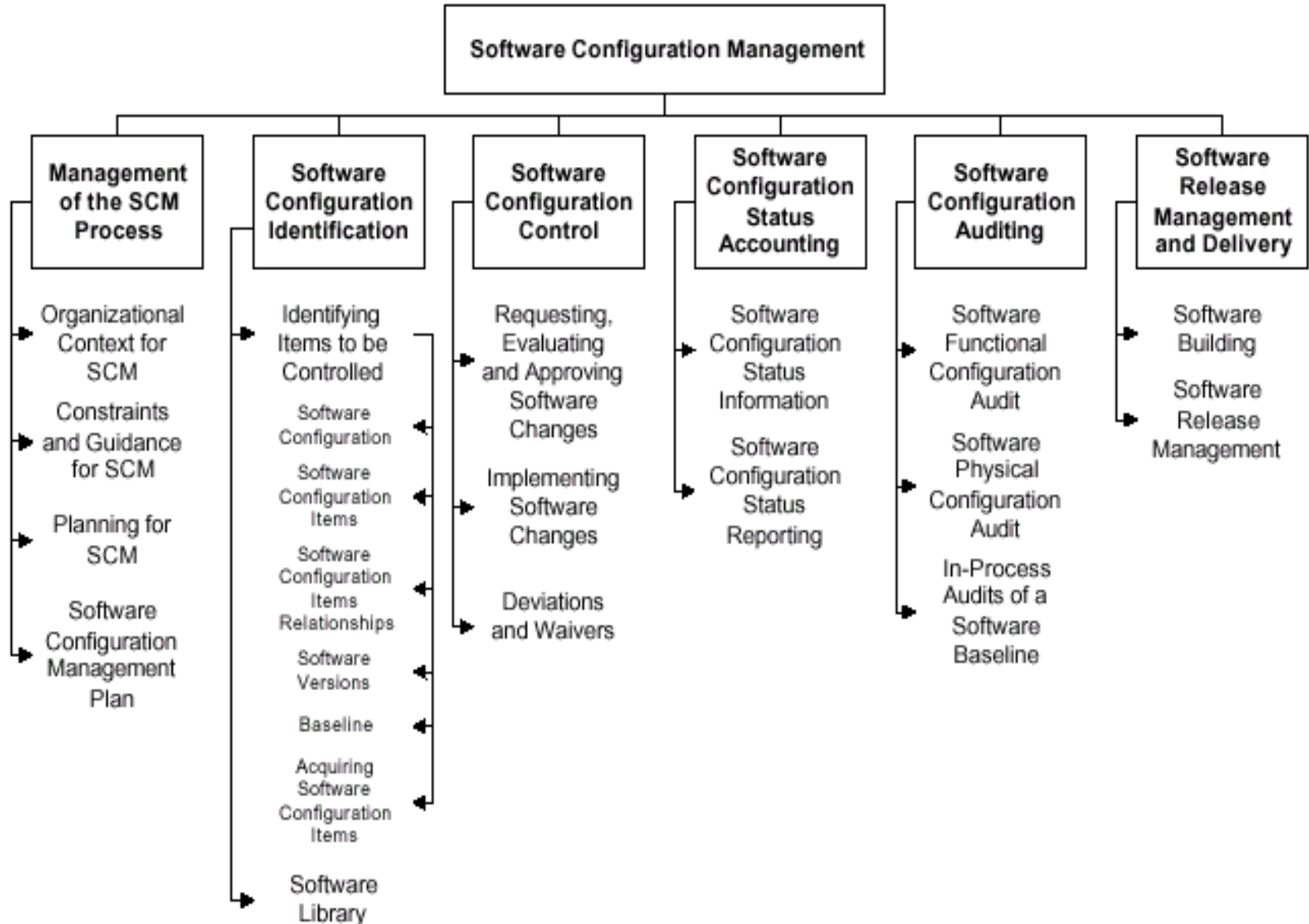
# Software Testing



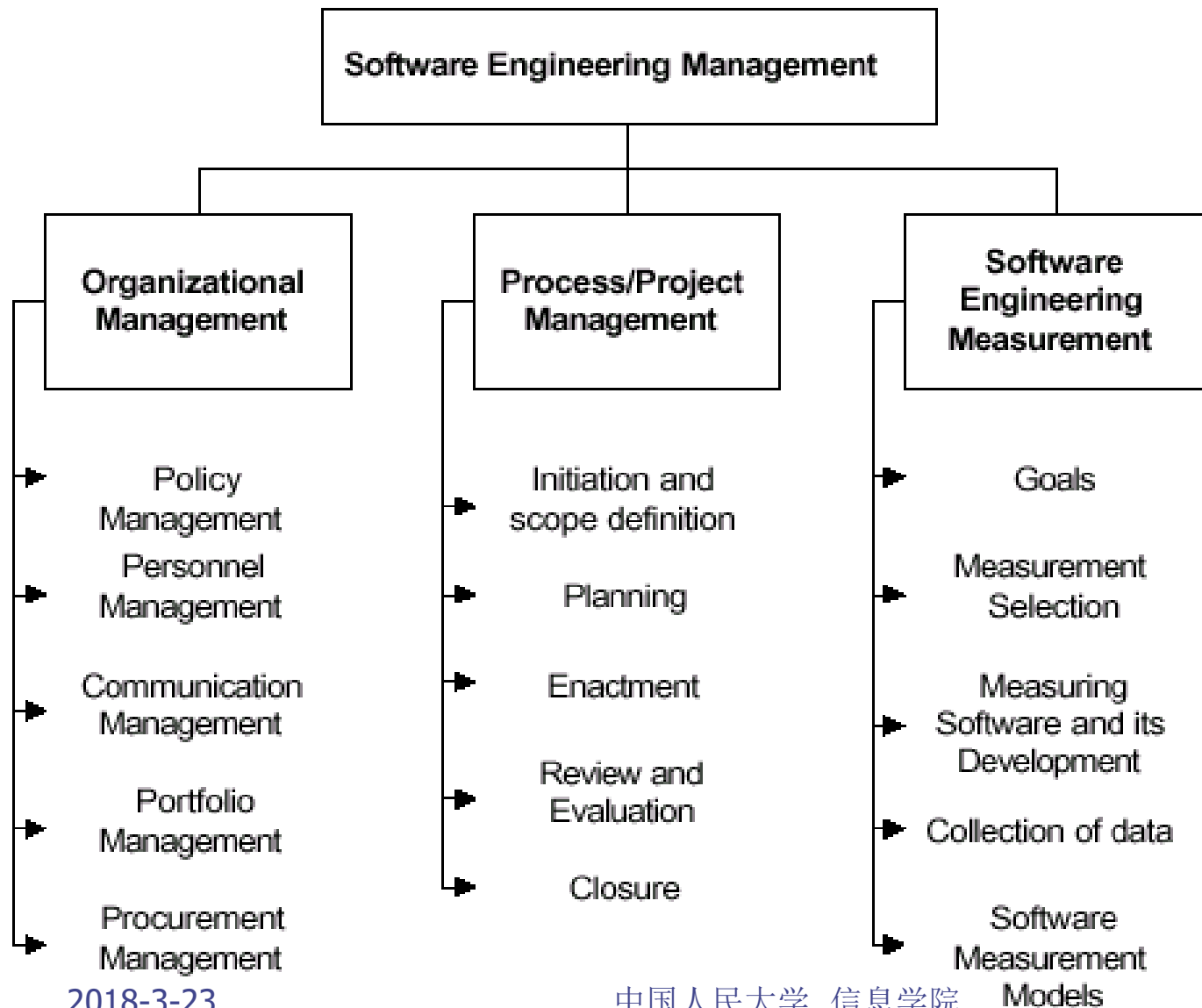
# Software Maintenance



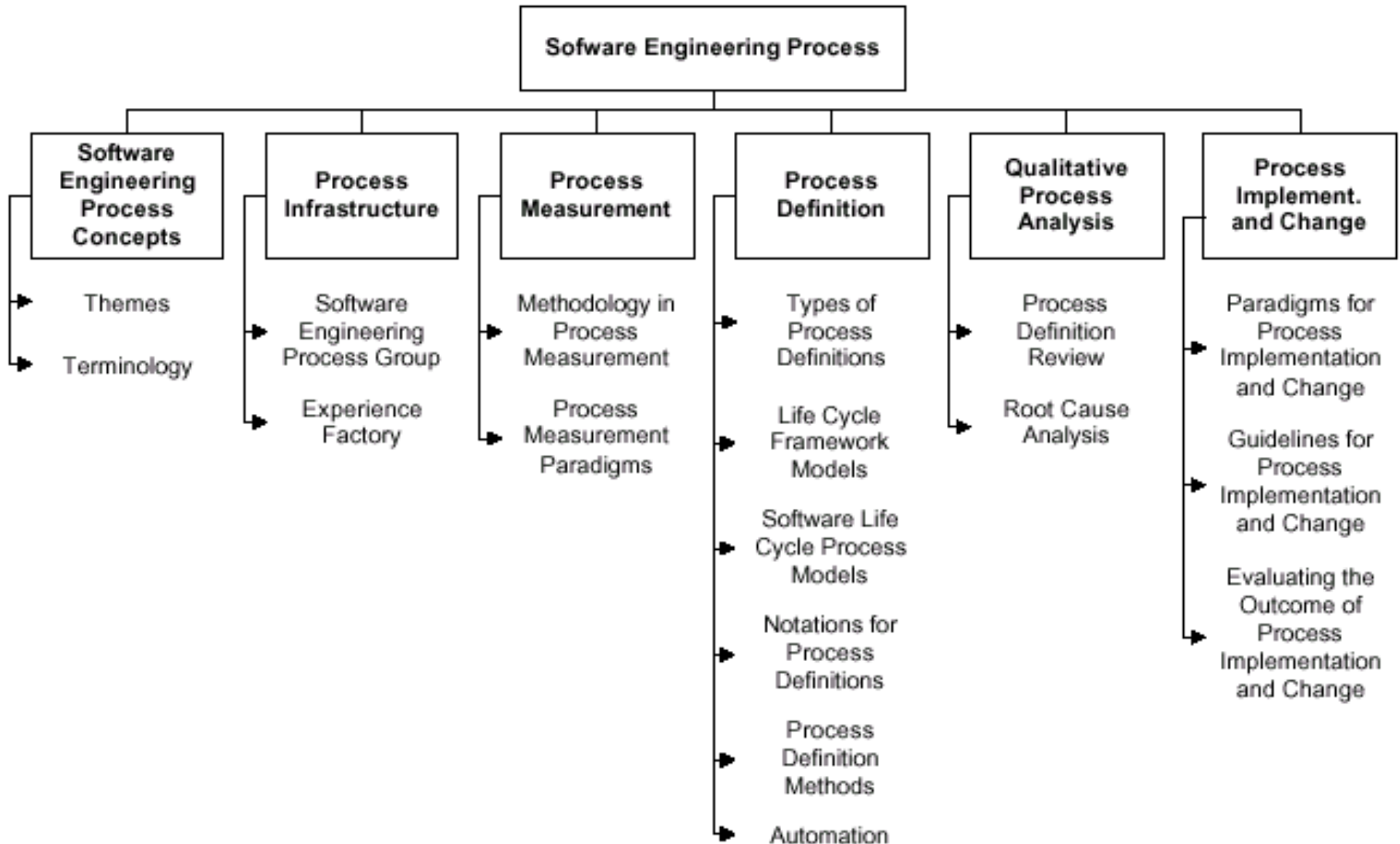
# Software Configuration Management



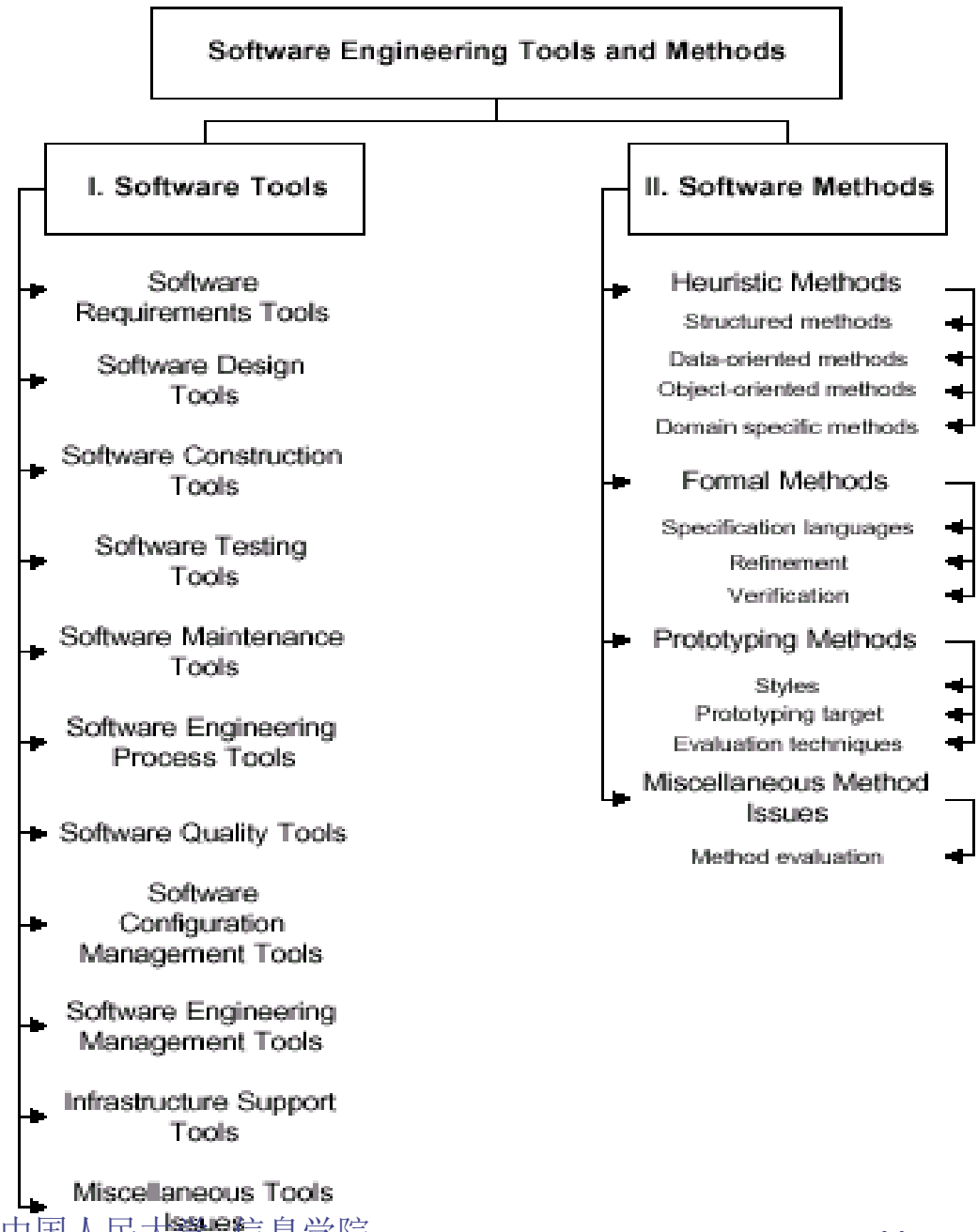
# Software Engineering Management



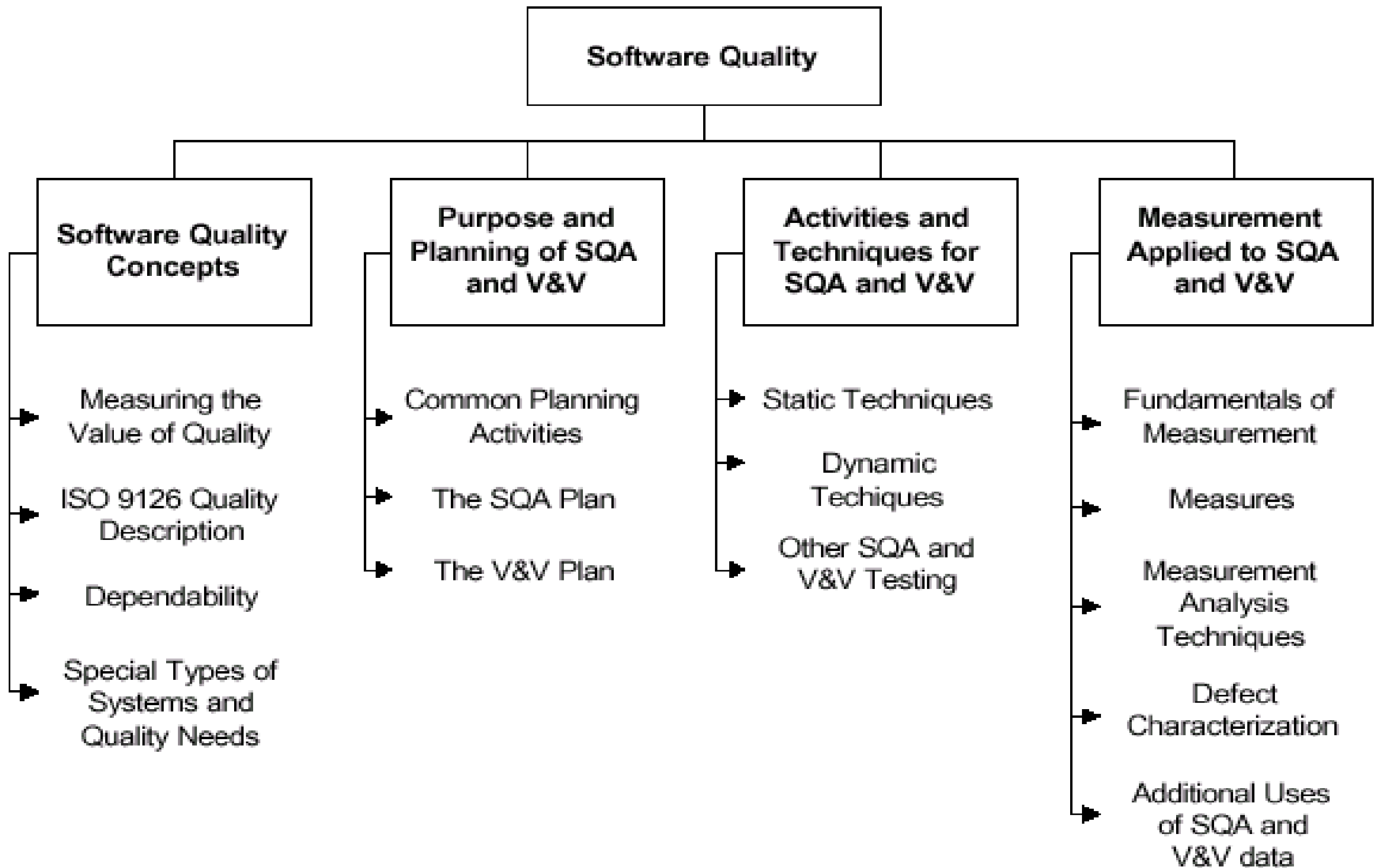
# Software Engineering Process



# Software Engineering Tools and Methods



# Software Quality





# 第1讲 软件与软件工程

## 1.5 软件工程师的职业特点

# Issues of professional responsibility (职业道德)

- *Confidentiality* (机密)

Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

- *Competence* (工作竞争力)

Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

- *Intellectual property rights* (知识产权)

Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

- *Computer misuse* (计算机滥用)

Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

# Code of ethics – principles（代码道德）

- PUBLIC（公众感）  
Software engineers shall act consistently with the public interest.
- CLIENT AND EMPLOYER（客户和顾主）  
Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- PRODUCT（产品）  
Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- JUDGMENT（判断力）  
Software engineers shall maintain integrity and independence in their professional judgment.
- MANAGEMENT（管理能力）  
Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- PROFESSION（职业感）  
Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
- COLLEAGUES（对待同事）  
Software engineers shall be fair to and supportive of their colleagues.
- SELF（自我要求）  
Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

# 讨论题

1. 软件生存期模型的作用是什么？
2. 选择软件生存期模型时，需要考虑哪些因素？
3. 本单位是否需要引入新的软件开发方法
4. 职业工程人员是否应该和医生或律师一样要颁布资格证书？讨论一下。

# 课堂练习题

1. 分析案例A中的项目，为它们选择恰当的软件生命周期模型，并说明理由。
2. 除了遗留系统、多样性和快速交付方面的挑战，说出软件工程可能面临的其它问题和挑战。

# 习题

1. 软件工程是如何克服软件危机的？
2. 软件的缺陷为什么在软件开发和维护过程中会扩大？
3. 原型开发的目的是、特点和包括那几类？
4. 简述模型在软件开发中的作用。
5. 论述过程、方法和工具在软件工程实践中的关系。
6. 分析比较各种软件开发模型的特征和优、缺点？