软件工程原理与应用

Software Engineering: Principles and Applications

南京航空航天大学计算机学院


软件工程2019
扫一扫二维码，加入群聊。

课程内容

- 软件与软件工程
- 需求分析（获取、分析、管理）
- 软件设计（总体、详细、界面、程序）
- 软件测试（过程、方法、工具）
- 软件维护
- 软件项目管理（度量、过程管理、配置）

参考书

- 构建之法：现代软件工程，邹欣著，人民邮电出版社.
- 人月神话，Frederick P. Brooks，清华大学出版社.
- 人件，Tom DeMarco，机械工业出版社.
- 程序员修炼之道—从小工到专家，Andrew Hunt，电子工业出版社

相关专业课

- 程序设计
- 数据结构
- 操作系统
- 数据库原理
- 软件体系结构
- 现代软件开发技术

----如何在一定的过程和方法中，综合各种专业知识，开发出满足用户需要的软件。

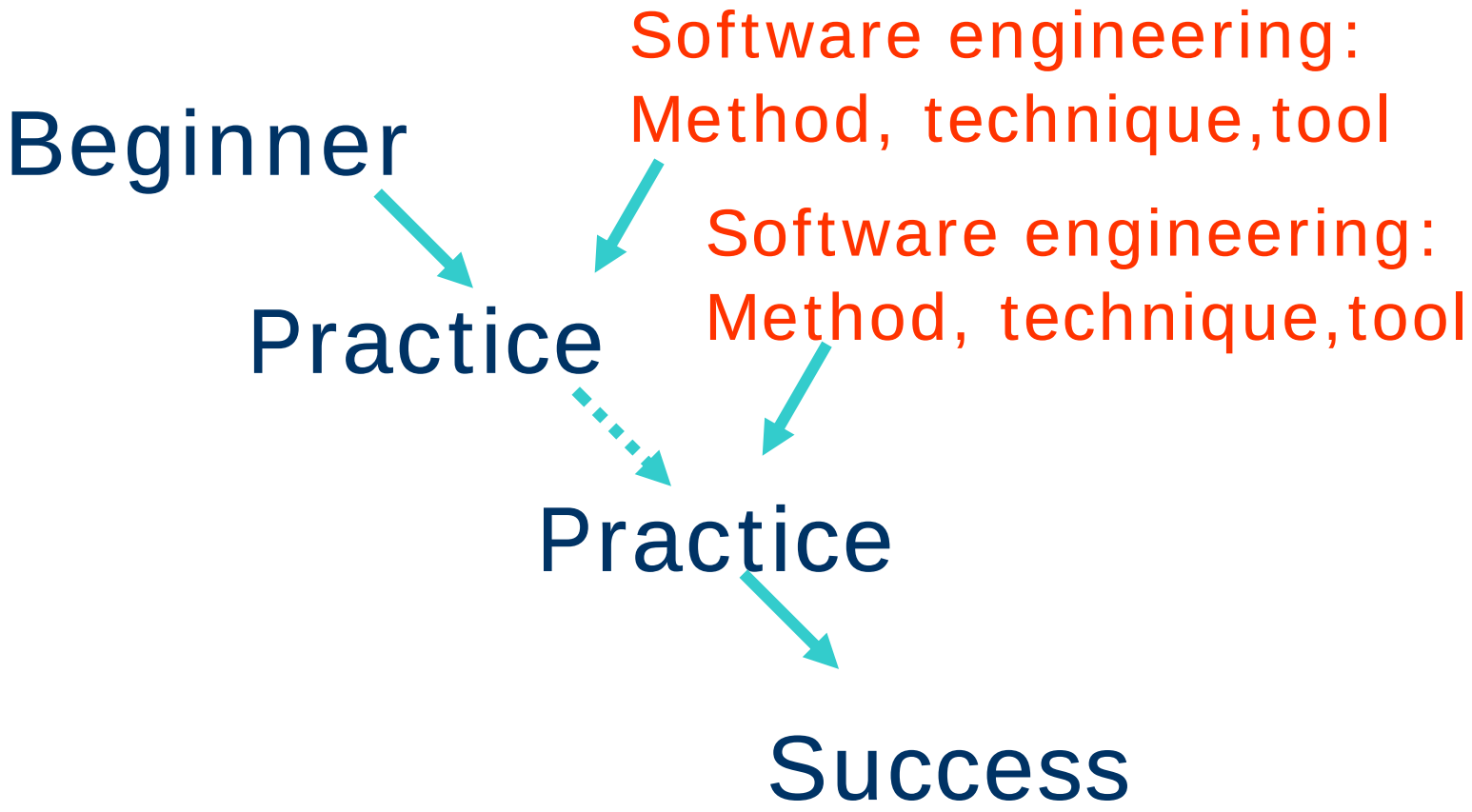
课程目标

- 掌握软件工程的基本原则、方法和技术，为大型软件的研发奠定部分基础
- 为后续课程提供必要的知识准备
  - 软件工程综合课设（7）
  - 毕业设计（8）

培养目标

- 培养自顶向下的抽象思维能力
- 培养独立解决问题的能力
- 培养合作精神
- 在实践中体会软件工程的原则、方法和技术，在实践中提高

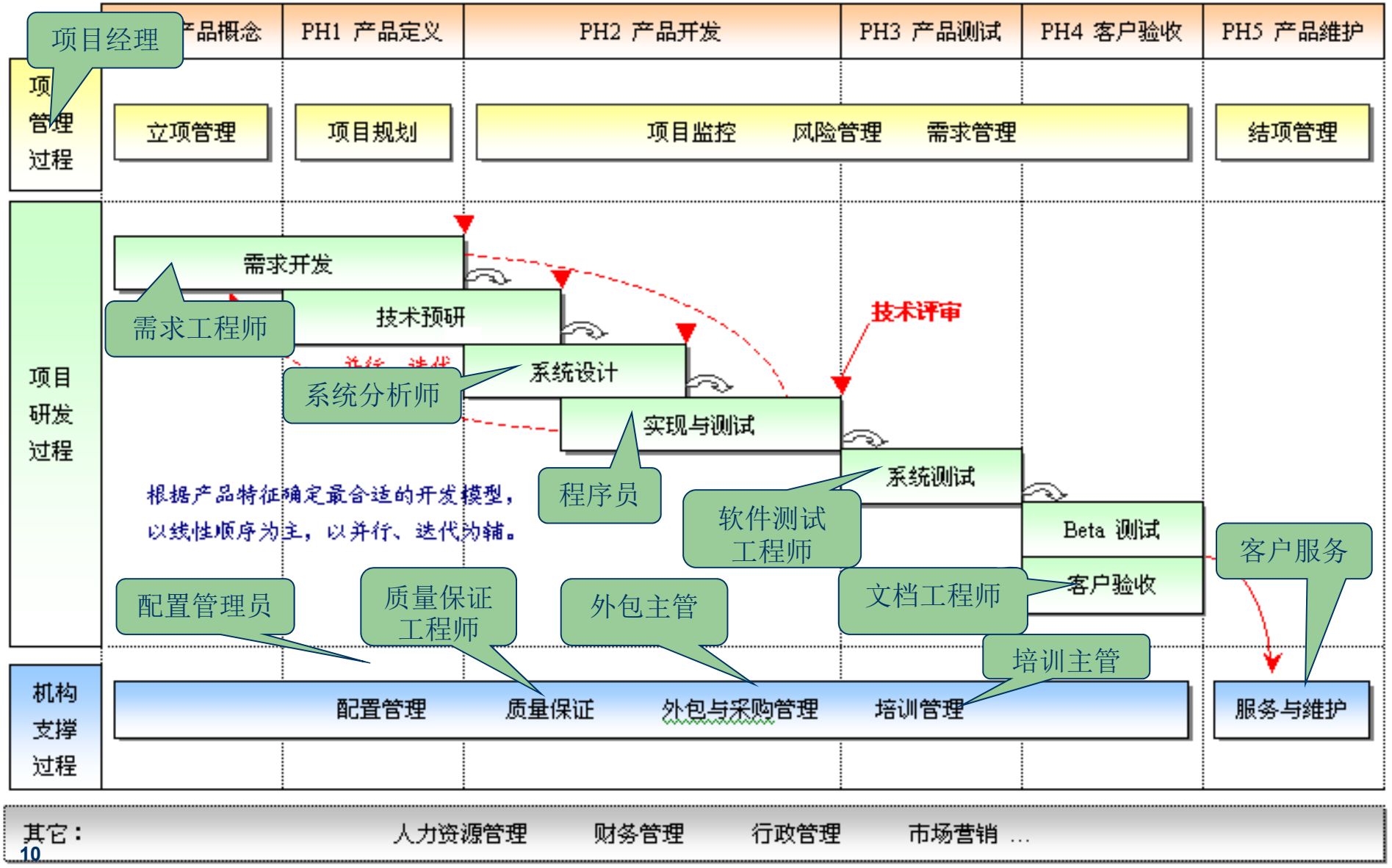
学习方法


Software engineering:
Method, technique,tool
Beginner
Software engineering:
Method, technique,tool
Practice
Practice
Success

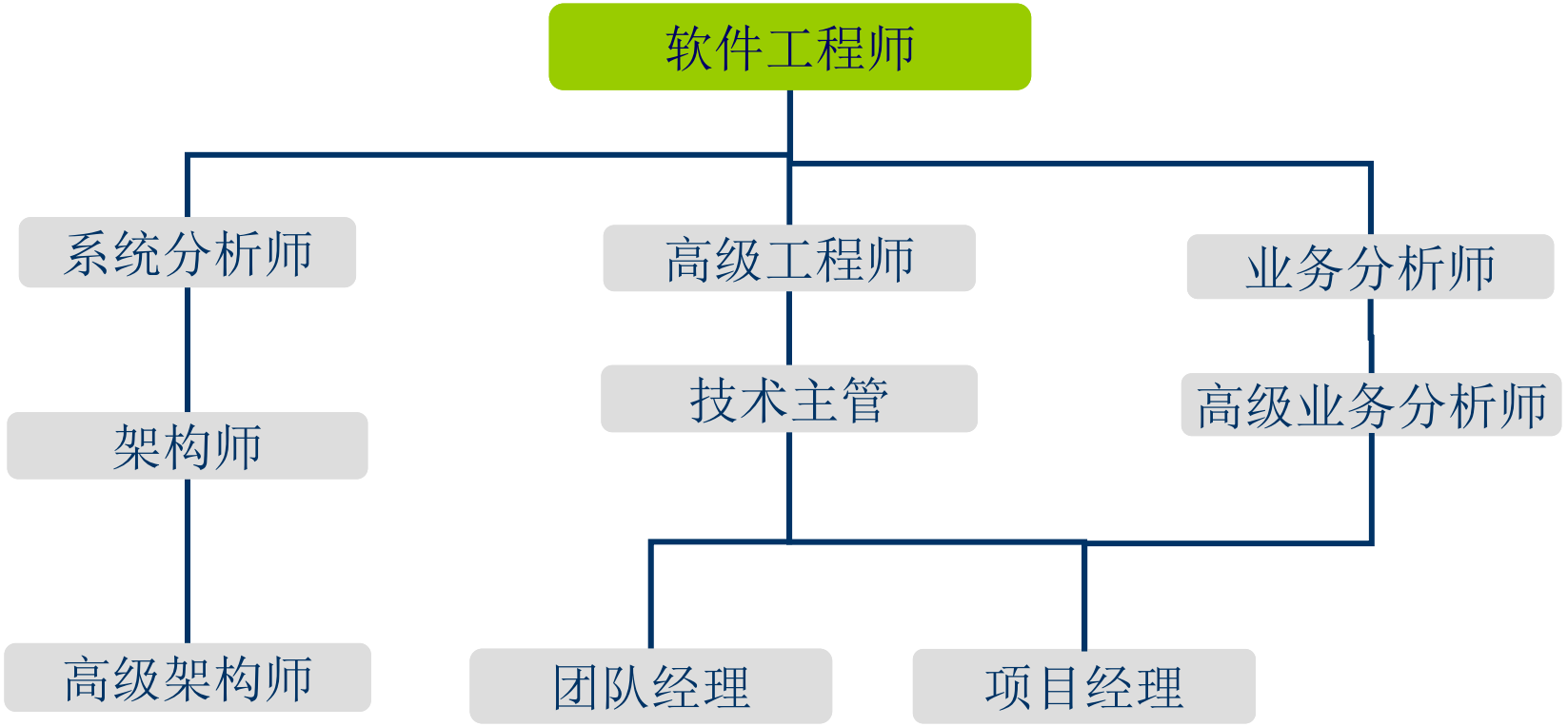
学习困难（历届学生反映）

- 原理和方法的介绍太抽象、不能理解和掌握
  -- 自己做一遍
- 实践过程中要学习的技术太多
  -- 会用一两套就行
- 不知道看什么参考资料
  -- 技术论坛、实例代码

软件工程职业规划


项目经理
PH1 产品定义
PH2 产品开发
PH3 产品测试
PH4 客户验收
PH5 产品维护
项目管理过程
立项管理
项目规划
项目监控
风险管理
需求管理
结项管理
项目研发过程
需求开发
需求工程师
技术预研
系统设计
系统分析师
技术评审
实现与测试
程序员
系统测试
软件测试工程师
Beta 测试
客户验收
客户服务
文档工程师
根据产品特征确定最合适的开发模型，
以线性顺序为主，以并行、迭代为辅。
配置管理员
质量保证工程师
外包主管
培训主管
机构支撑过程
配置管理
质量保证
外包与采购管理
培训管理
服务与维护
其它：
人力资源管理
财务管理
行政管理
市场营销 …

技术方向发展之路



企业对软件工程师的要求

- 业务技能
  - 能独立应用掌握的技术解决业务问题
- 敬业精神
  - 良好的工作态度，专心致志于工作，认真负责
- 学习能力
  - 不断学习掌握新技术、新经验，推陈出新
- 团队心态
  - 一人为大家，大家为一人
- 专业素质
  - 行为、待人处事遵循公司规范

软件工程职业的乐趣

- 《人月神话》：
  - 一种创建事物的纯粹快乐；
  - 开发出来的东西对别人有用；
  - 将相互啮合的零部件组装在一起，看到它们精妙地运行，并得到预先所希望的结果；
  - 不断学习新方法新技术的乐趣；

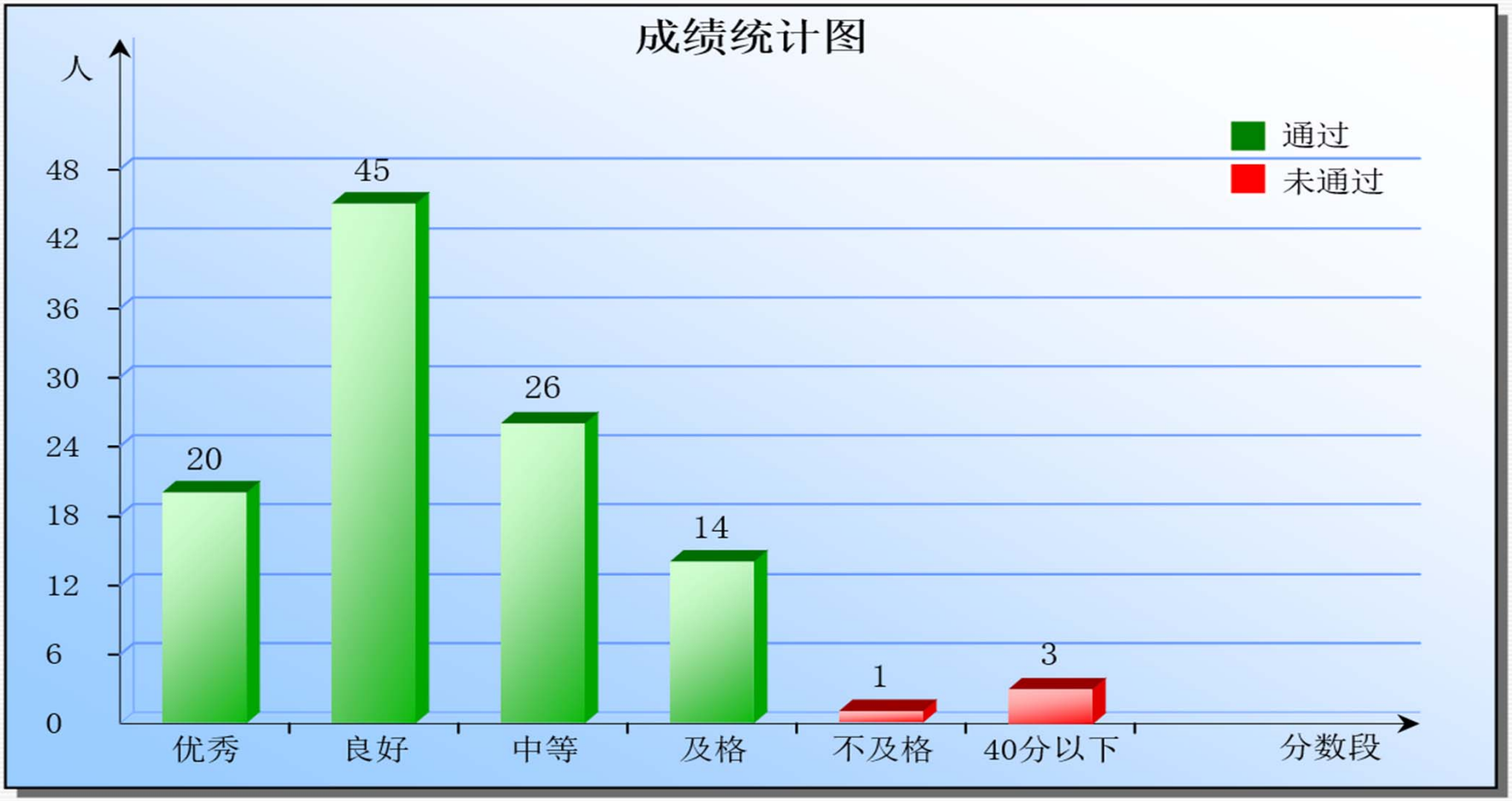
职业的苦恼

- 《人月神话》：
  - 是由他人来设定目标，供给资源，提供信息；
  - 概念性设计是有趣的，但寻找琐碎的 bug 却只是一项重复性的活动；
- 加班、封闭开发
- 没有一劳永逸的解决方案
- 编程的活儿能干到多少岁？

考核方法

- Attendance --- 10%
- Project --- 20%
- Test --- 70%

软件工程原理与应用Ⅱ.16103630



总人数	优秀	良好	中等	及格	不及格	40分以下	无成绩
109	20	45	26	14	1	3	0
最高分	最低分	平均分	标准差	偏度	峰度	区分度	
95	30	78.83	0.88			0.289	

Project

- 2-3人/组
- 中小型系统开发，包括至少3个功能
- 各组选题不能重复
- 完成5个报告（开题、需求、分析、设计、测试），20分/报告，课间拷报告模版
- 两周内完成分组、选题并报至班长处，班长用Excel列表分组、组员、题目Email 给我

关于选题

- 图书管理系统
  - 借书+还书+图书信息管理+读者信息管理+…
- 学生信息管理系统
  - 基本信息管理+选课管理+成绩管理+…
- 人事信息系统
  - 基本信息管理+工资管理+奖惩信息管理+…

除非有创新的功能设计，否则以上题目不要选了

选题来源

- 创新基金
- 现有课题
  - 如有参与导师课题，得到允许，适当改变
- 其他课程的大作业
  - 图形学

题目列举

- Wap邮件系统
- 二手商品交易系统
- 博客系统
- 网络性格测评系统
- 车棚车辆管理系统
- 医务管理系统
- 论坛
- 仓库管理系统

- 客房管理系统
- 订餐系统
- 物业管理系统
- 网上书城
- 电子超市
- 教材管理系统
- 航空票务系统
- 。。。

TIOBE 8月编程语言排行 1-20

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	↑	Python	10.020%	+3.03%
4	3	↓	C++	6.057%	-1.41%
5	6	↑	C#	3.842%	+0.30%
6	5	↓	Visual Basic .NET	3.695%	-1.07%
7	8	↑	JavaScript	2.258%	-0.15%
8	7	↓	PHP	2.075%	-0.85%
9	14	⇈	Objective-C	1.690%	+0.33%
10	9	↓	SQL	1.625%	-0.69%
11	15	⇈	Ruby	1.316%	+0.13%
12	13	↑	MATLAB	1.274%	-0.09%

专题： 软件与软件工程

Topic ：Software and Software Engineering

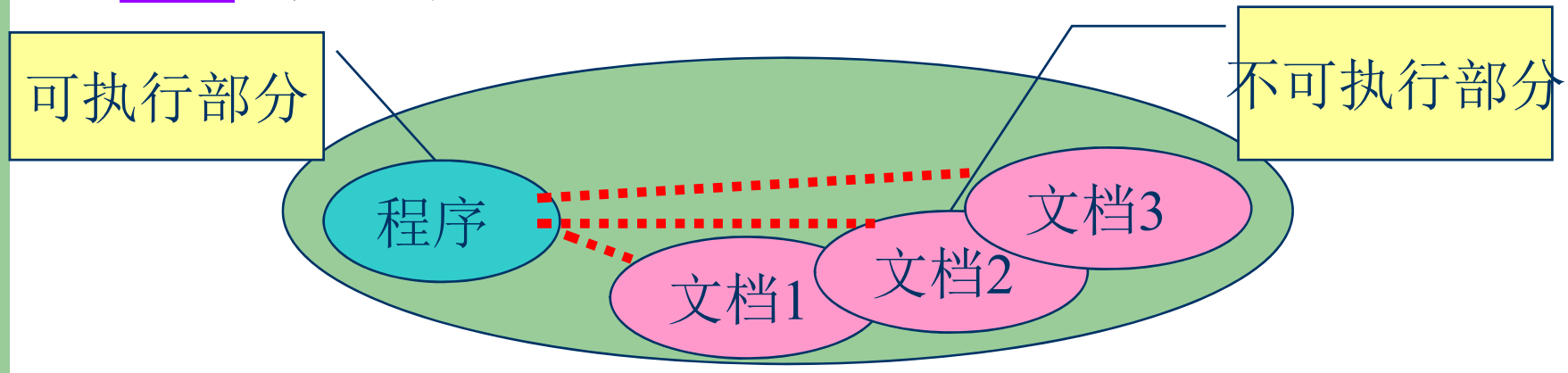
用工程化的方法开发软件系统

本专题内容

1. 软件的概念
2. 软件工程的概念
3. 软件过程模型
4. 敏捷开发
5. 开发工具和环境

1.1软件和软件的组成

软件 与计算机系统操作有关的程序、数据以及相关文档的完整集合；



程序 按设计功能和性能要求执行的指令序列

数据 使 程序能正常操纵信息的数据结构

文档 是与程序开发、维护和使用有关的图文材料。

1.1软件和软件的组成

- 程序：由程序设计语言所描述的、能为计算机所识别、理解和处理的语句序列
- 程序设计语言具有严格的语法和语义
- 程序设计语言的类型：
  - 面向机器: 如汇编语言、机器语言等
  - 面向过程: 如Fortran, Pascal, C等等
  - 面向对象: 如Java，Python，C++等等
  - 面向问题: 如结构化查询语言SQL等等

1.1 软件的概念

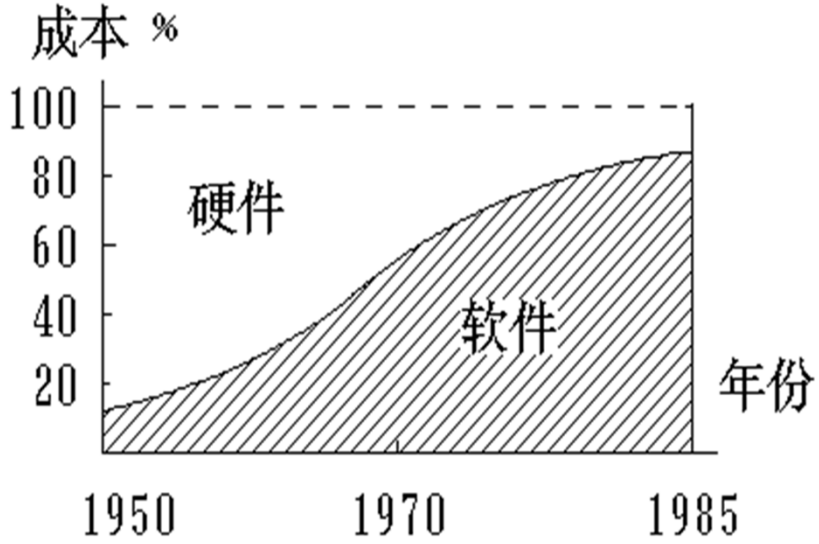
- 文档: 记录软件开发活动和阶段性成果，为理解软件所必需的阐述性资料
  - 需求分析文档
  - 软件设计文档
  - ……
- 编写文档的目的
  - 促进对软件的开发、管理和维护；
  - 便于各种人员(用户,开发人员等)的交流

注意

- 软件≠计算机程序
- 软件应包括：
  - 程序
  - 配置文件
  - 系统文档
  - 产品信息

1.2 软件的特点（1/3）

- 逻辑实体、智力产品
- 制造即拷贝



1.2 软件的特点（2/3）

- 无磨损和老化，不遵循“浴盆曲线”，但存在退化问题

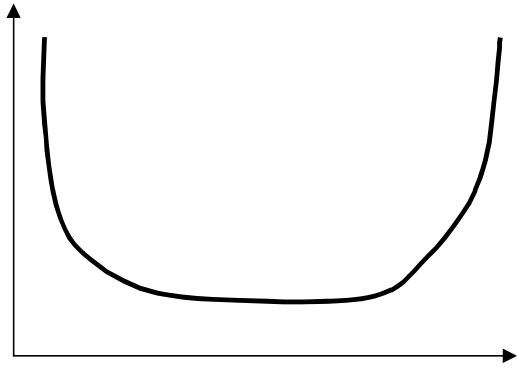


图 U型曲线(即浴盆曲线)

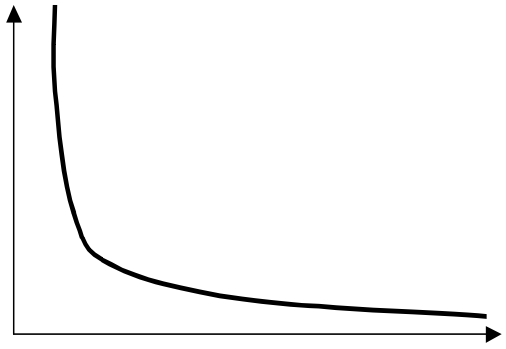


图 软件的故障率曲线（理想）

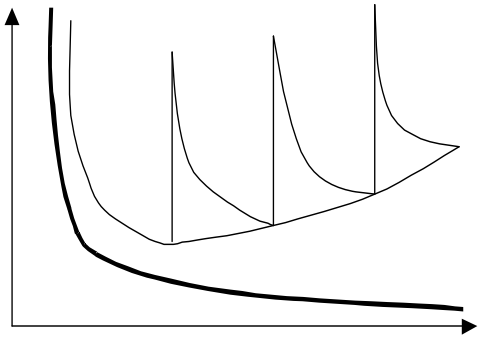


图 软件的故障率曲线（实际）

1.2 软件的特点（3/3）

- 尚未摆脱手工方式
- 软件移植的需要
- 复杂（问题复杂性 / 程序结构复杂性）
- 软件开发的性质如成本、进度、质量等难以估计控制
- 维护困难，可复用性

1.3 软件的分类

- 按功能

  系统软件 / 支撑软件 / 应用软件

- 按工作方式

  实时处理 / 分时 / 交互 / 批处理

- 按服务对象

  项目 / 产品（定制 / 通用）

- 按失效影响

  关键软件 / 非关键软件

1.3 软件的分类

- 规模

分类	子程序数量	参加人员	开发期限	程序规模（行）
微型	10-20	1人	1-4周	500以下
小型	25-50	1人	1-6月	1K-2K
中型	250-1000	2-5人	1-2年	5K-50K
大型		5-20人	2-3年	50K-100K
甚大型		100-1000人	4-5年	1M
极大型		2000-5000人	5-10年	1M-10M

2 软件工程的概念

2.1 软件的发展：（发展阶段）

程序设计 → 程序系统→ 软件工程 → OO软件工程

软件体系结构/设计模式/框架/构件化/重用/生产线

2.1 软件的发展

- 程序设计阶段 — 50至60年代
  - 自定义软件，面向批处理
  - 程序规模小，编写者与使用者往往是同一人
  - 没有系统的方法可循、没有设计、文档资料
  - 个体手工方式

2.1 软件的发展

- 程序系统阶段 — 60至80年代
  - 程序规模大，多道程序设计、多用户系统、数据库管理系统
  - 软件产品大量销售，软件数量急增
  - 生产方式：由个体发展为软件车间
  - 软件维护耗资巨增，软件产品不可维护，导致软件危机

2.1 软件的发展

- 软件工程阶段 — 80年代90年代末
  - 分布式系统
  - 软件工程化的设计原则、方法和标准
  - 软件产业兴起软件产品化、系列化、工程化和标准化
  - 生产方式是：由软件车间联成软件工厂、公司
  - 软件标准的规范不完善，软件危机仍然存在

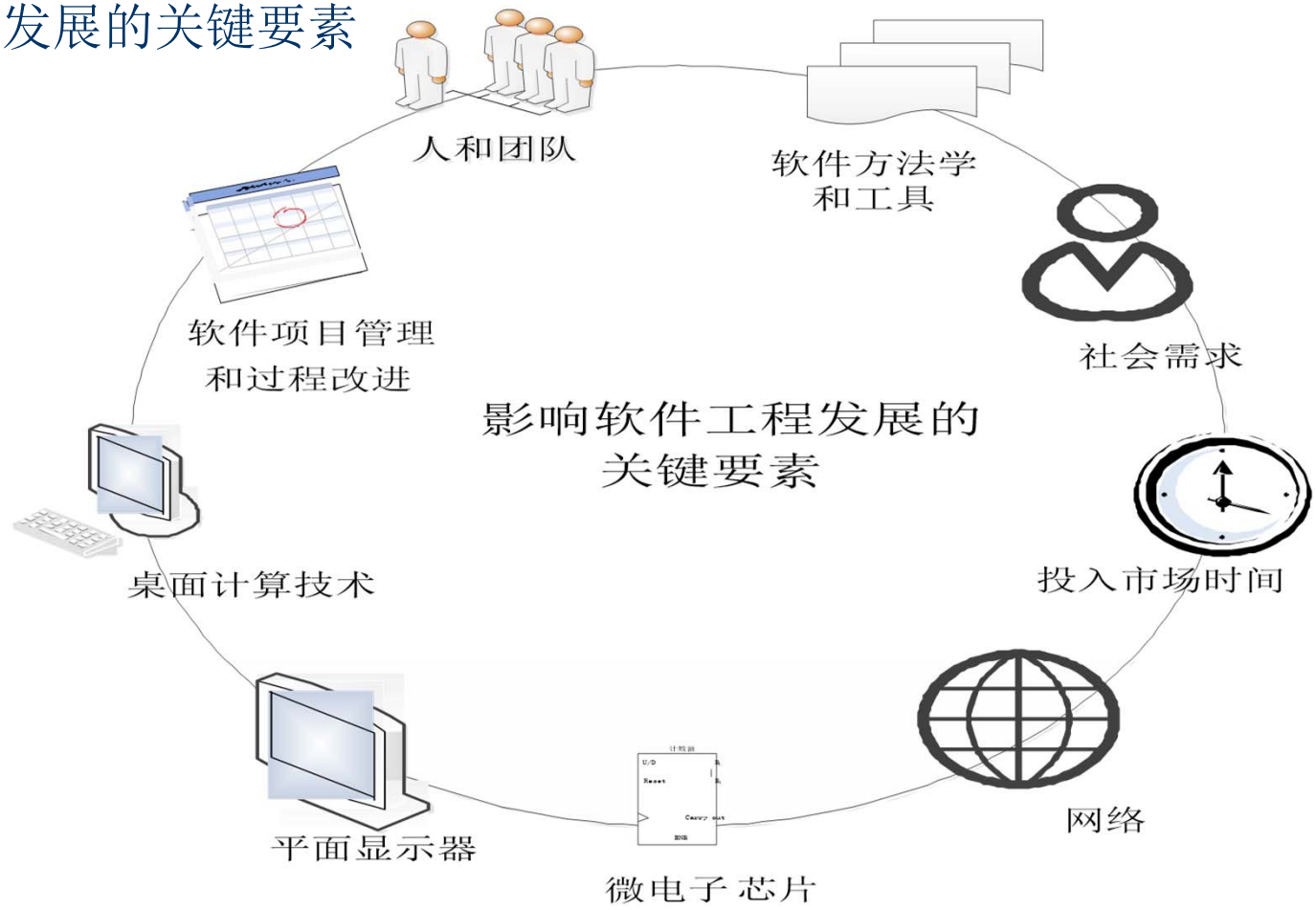
2.1软件的发展

- OO软件工程阶段 —90年代后
  - 强大的桌面系统
  - 面向对象技术
  - 专家系统
  - 并行计算
  - 网络计算机
- 目前：社会信息化、软件产业化的阶段过渡

  从技术性的软件工程阶段过渡到企业计算机，社会信息化的计算机系统工程阶段

2.1软件的发展

影响软件工程发展的关键要素



2.2 软件危机

软件危机的表现：

- 软件开发成本和进度失控，维护代价高
- 用户不满意
- 软件质量不可靠
- 软件不可维护
- 无文档资料
- 计算机系统中软件成本比重加大
- 软件开发生产率提高不能满足要求

2.2 软件危机

软件危机典型案例：

IBM 360 / OS

- 5000 人年，最多1000人，1000 kLOC
- 大约每修正1000 bug，形成一个升级版

Brooks的感慨：一只陷入泥潭的怪兽，越是挣扎，陷得越深……

2.2 软件危机

软件危机的原因：

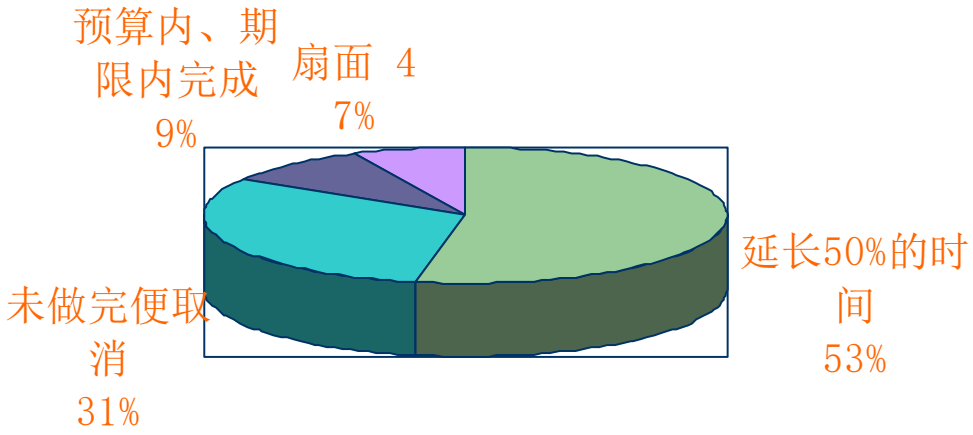
- 软件的规模和复杂性
- 人类智力的局限性
- 协同工作的困难性
- 缺乏方法学和工具
- 用户描述不精确、二义、遗漏，双方理解有偏差

2.2 软件危机

缓解软件危机的途径：

- 组织管理、协同配合的工程
- 软件工程的理论模型、技术方法
- 软件工具

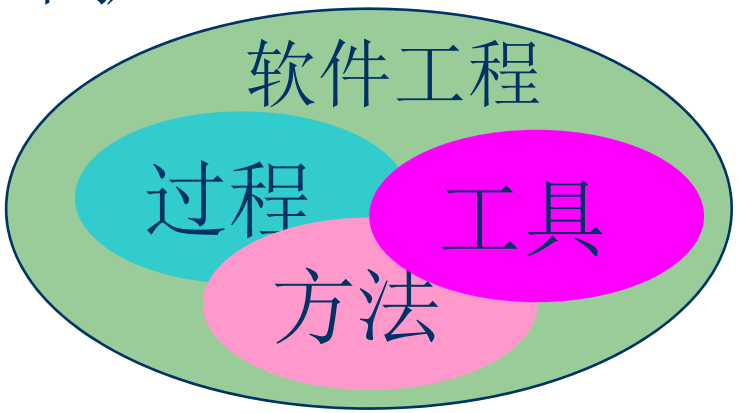
- 危机仍在延续
  - 美国软件实施现状调查 （20世纪90年代）



- 深层思考：
  1. 开发一个大型软件系统与编写一个简单程序完全不一样
  2. 按工程化的原则和方法组织软件开发

2.3 软件工程的概念

- 用工程的、科学的概念、原理、技术和方法，进行软件的开发、管理和维护



- 软件工程的三要素：
  - 过程：管理部分
  - 方法：技术手段
  - 工具：自动或半自动地支持软件的开发和管理

- 要素之间相互关联和支持

解读

- 是一门工程学科，涉及软件生产的各个方面
  - 工程都要求在时间和预算范围内获得所要求的成果
  - 工程人员既拥有一定理论、方法和工具，又能在各种情况下，选择最恰当的理论和方法来解决问题。

2.4 软件开发方法

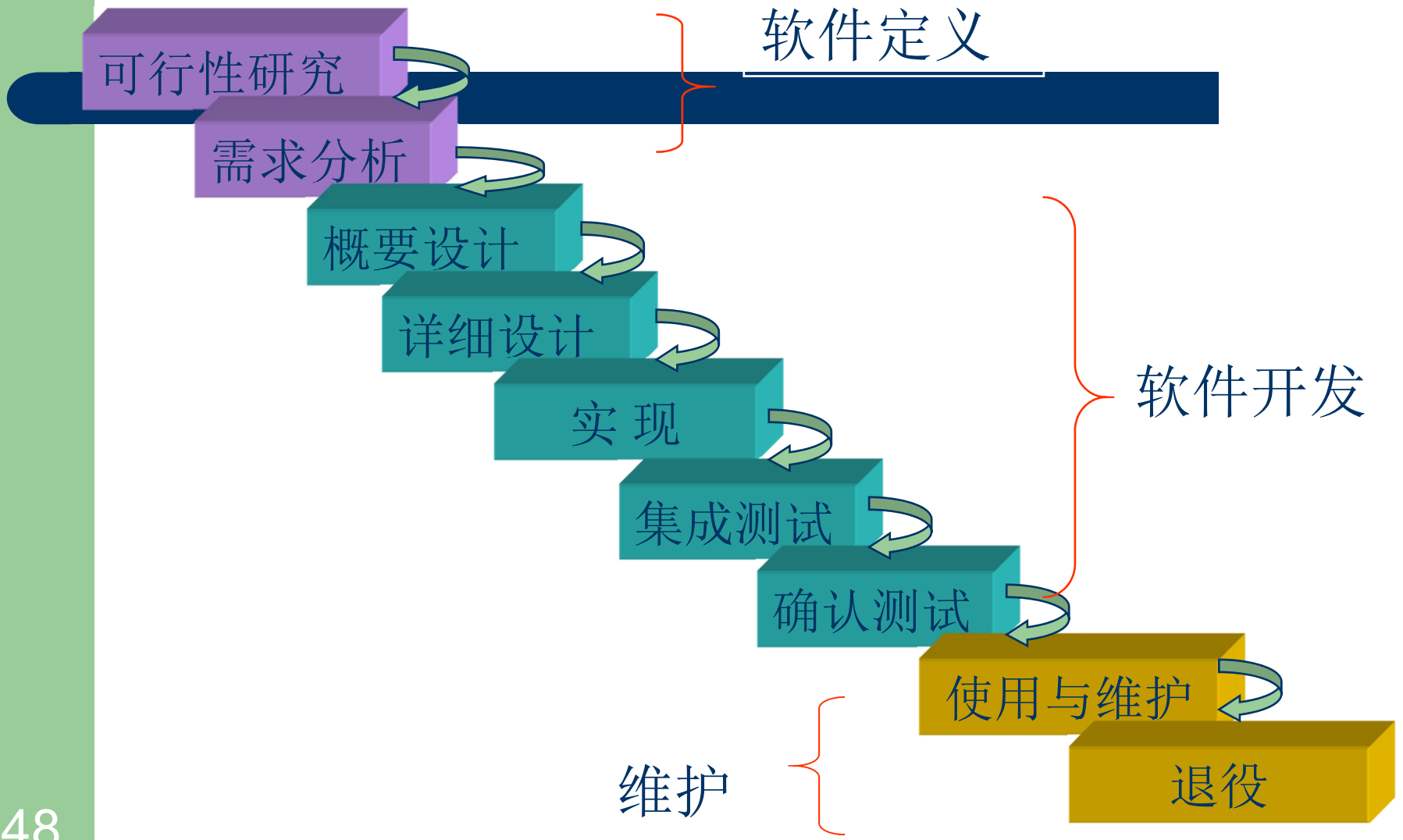
- 主流
  - 结构化方法
  - 面向对象方法
- 非主流
  - 形式化开发方法
  - 基于构件的软件开发方法
  - 面向方面AOP的软件开发方法
  - 基于Agent的软件开发方法
  - 。。。

3.1 软件过程

软件工程过程定义了:

- 方法使用的顺序
- 要求交付的文档资料
- 为保证质量和适应变化所需要的管理
- 软件开发各个阶段完成的里程碑

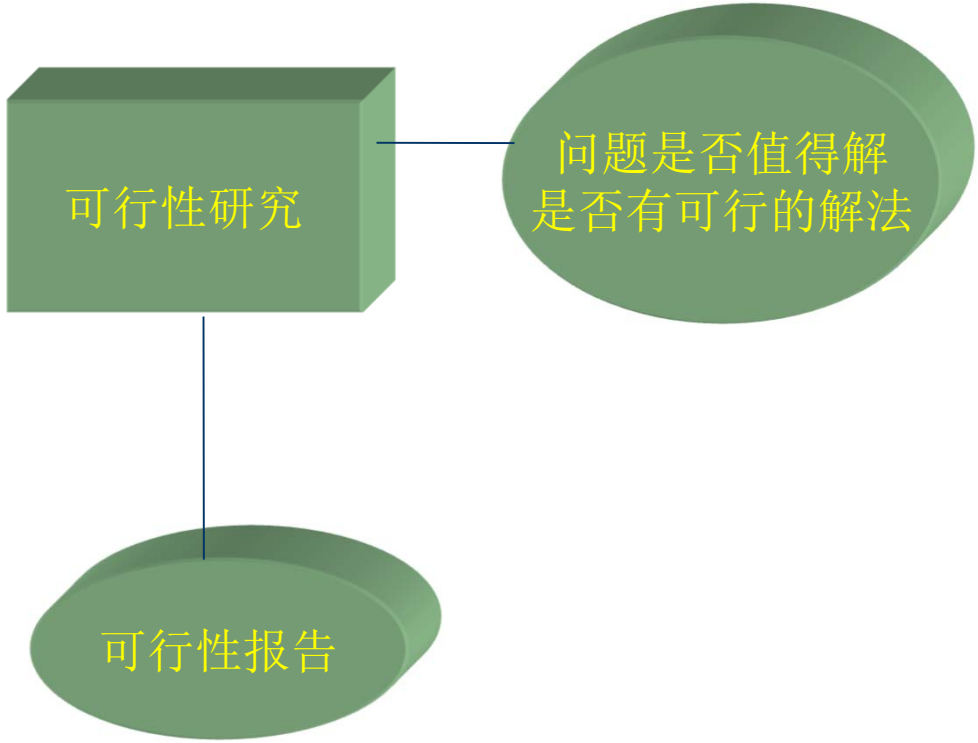
软件生命周期(Life Cycle)



软件生命周期--可行性研究 (1/2)

- 任务

  了解用户要求和现实环境，从技术、经济、市场等方面研究并论证开发该软件系统的可行性



软件生命周期--可行性研究 (2/2)

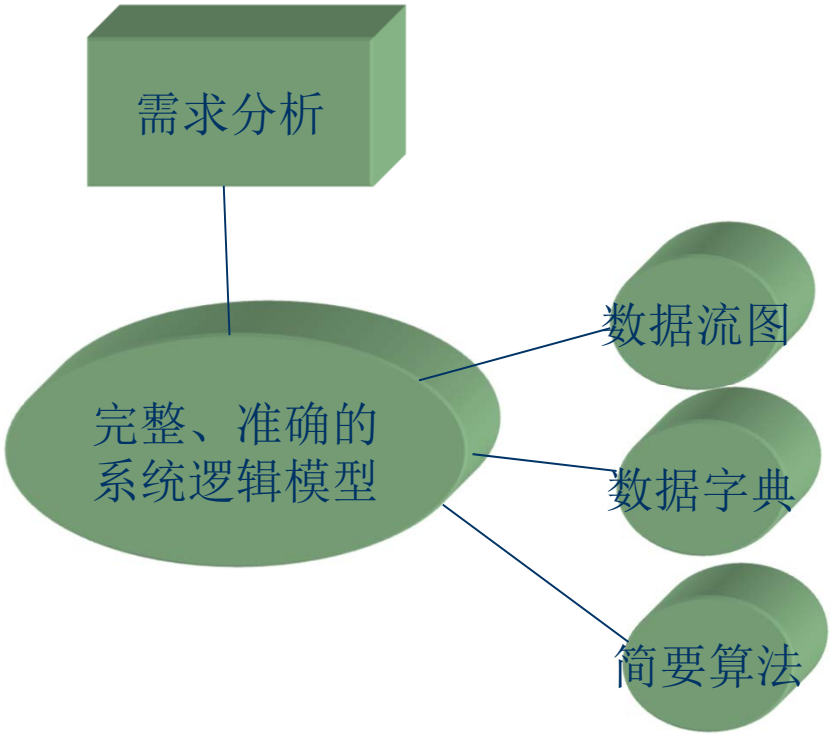
- 技术途径
  - ✓ 调查和了解用户要求 和 现实环境
  - ✓ 制作调查报告
  - ✓ 可行性论证和分析（技术、经济等）
  - ✓ 如果可行，制定初步的项目开发计划(人员、进度等)
- 阶段性产品
  - ✓ 可行性论证报告
  - ✓ 初步的项目开发计划

软件生命周期--需求分析(1/3)

- 任务

  确定用户对 待开发软件系统的需求，包括：

  - ✓ 功能
  - ✓ 性能
  - ✓ 运行环境约束



- 重要性

  - ✓ 软件开发依据
  - ✓ 软件验收的标准

软件生命周期--需求分析(2/3)

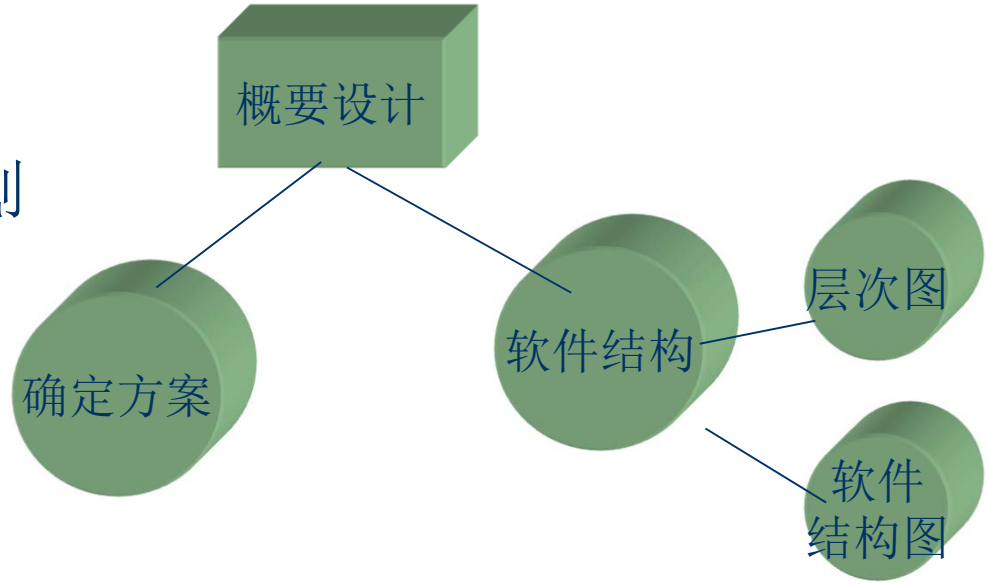
- 主要困难
  - ✓ 不清晰
  - ✓ 不完整
  - ✓ 歧义
  - ✓ 动态变化
- 技术途径和工具
  - ✓ 与用户不断、反复地交流和商讨，使需求逐步准确化、一致化、完全化
  - ✓ 抽象、问题分解、需求建模、快速原型、多视点等技术
  - ✓ 工具：Rational Rose等

软件生命周期--需求分析(3/3)

- 阶段性产品
  - ✓ 软件需求规格说明书SRS(功能，性能和设计约束等)
  - ✓ 初步的用户手册
  - ✓ 确认测试计划

软件生命周期--概要设计(1/2)

- 任务
  - ✓ 根据SRS建立目标软件系统的总体结构
  - ✓ 设计数据库和全局数据结构
  - ✓ 确定设计约束
  - ✓ 制定集成测试计划

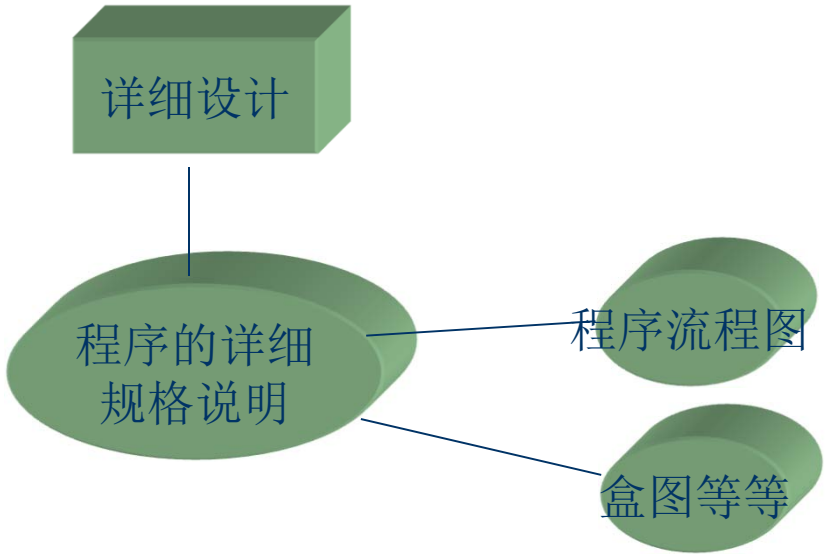


软件生命周期--概要设计(2/2)

- 技术途径和工具
  - ✓ 根据SRS，自顶向下逐步求精
  - ✓ 按照模块化原则设计软件结构
  - ✓ 工具： Rational Rose
- 阶段性产品
  - ✓ 概要设计规格说明书
  - ✓ 数据库/数据结构设计说明书
  - ✓ 集成测试计划

软件生命周期--详细设计(1/2)

- 任务
  - ✓ 细化概要设计所生成的各个模块, 并详细描述程序模块的内部细节(算法，数据结构等)
  - ✓ 制订单元测试计划



软件生命周期--详细设计(2/2)

- 技术途径
  - ✓ 根据概要设计结果进行算法和数据结构设计
  - ✓ 模块化、局部化、信息隐藏、单入口单出口原则
- 阶段性产品
  - ✓ 详细设计规格说明书
  - ✓ 单元测试计划

软件生命周期--实现(1/2)

- 任务
  - 根据详细设计规格说明书编写源程序
  - 对程序进行调试
  - 单元测试，验证程序与详细设计文档的一致性
- 技术途径和工具
  - 以详细设计规格说明书为依据，基于程序设计语言进行编码
  - 工具：Visual C++, JBuilder,Eclips

软件生命周期--实现(2/2)

- 阶段性产品
  - ✓ 源程序代码
  - ✓ 单元测试报告

软件生命周期--集成测试(1/2)

- 任务
  - ✓ 根据概要设计规格说明书，将经过单元测试的模块逐步进行集成和测试
- 技术途径
  - ✓ 以概要设计规格说明书和集成测试计划为依据，进行模块集成并测试

软件生命周期--集成测试(2/2)

- 阶段性产品
  - ✓ 生成满足概要设计要求、可运行的源程序
  - ✓ 集成测试报告

软件生命周期--确认测试(1/2)

- 任务
  - ✓ 根据软件需求规格说明书，测试软件系统是否满足用户的需求
- 技术途径
  - ✓ 由用户参与，以软件需求规格说明书和确认测试计划为依据进行确认测试

软件生命周期--确认测试(2/2)

- 阶段性产品
  - ✓ 可供用户使用的软件产品(文档，源程序)
  - ✓ 确认测试报告

软件生命周期--软件维护(1/2)

- 任务

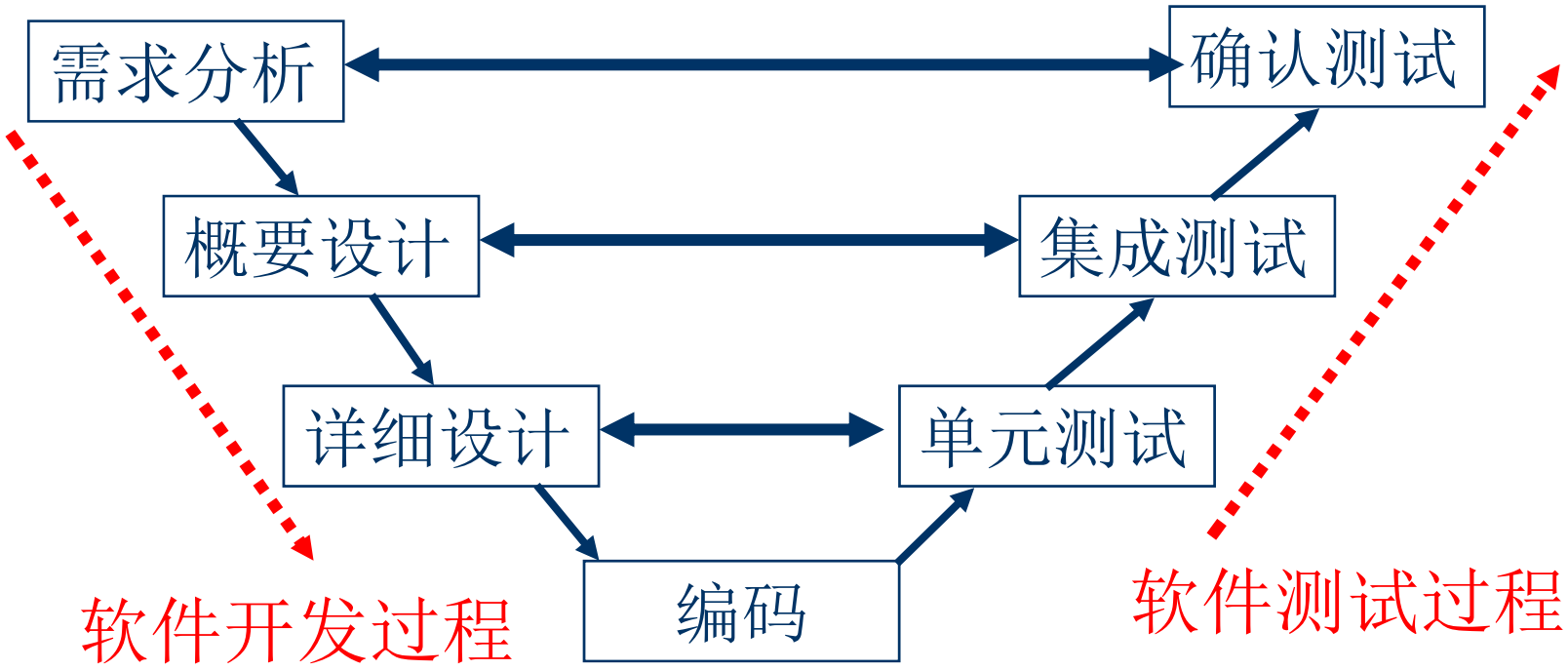
  对使用后的软件进行维护，例如:

  - ✓ 纠错性维护：修正使用过程中发现的错误
  - ✓ 完善性维护：增加新的功能
  - ✓ 适应性维护：使软件适应环境的变化；等

- 技术途径

  - ✓ 以文档和源程序为基础，按用户要求进行
  - ✓ 必须相应地修改文档
  - ✓ 进行回归测试

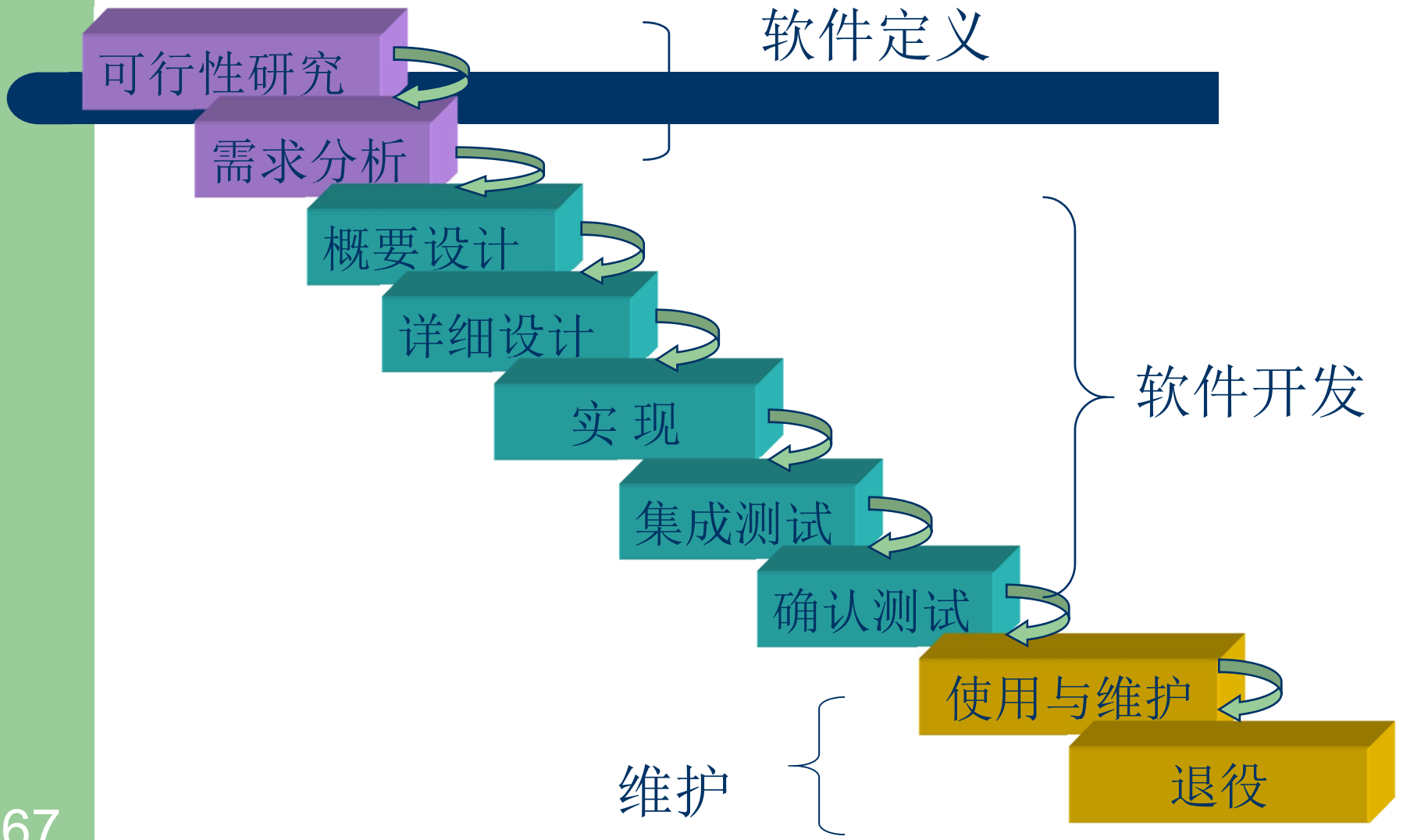
软件开发和测试活动间的关系



软件生命周期--软件维护(2/2)

- 阶段性产品
  - ✓ 版本更新的软件产品
  - ✓ 维护记录

软件生命周期小结



阶段	关键问题	结束标准（任务）
问题定义	问题是什么？	关于规模和目标的报告书
可行性研究	是否可行？	系统的高层逻辑模型； 数据流图；成本/效益分析
需求分析	系统必须做什么？	系统的逻辑模型；数据流图； 数据字典，用例视图
概要设计	总体上要解决的问题	系统流程图；成本/效益分析 层次图和结构图；对象类图等
详细设计	怎样具体的实现	HIPO或PDL
编码和单元测试	正确的程序模块	源程序清单；单元测试方案和结果
组合测试	符合要求的软件	综合测试方案和结果； 完整一致的软件配置
运行、维护	持久地满足用户需要	完整准确的维护记录

3.2 软件过程模型

定义：

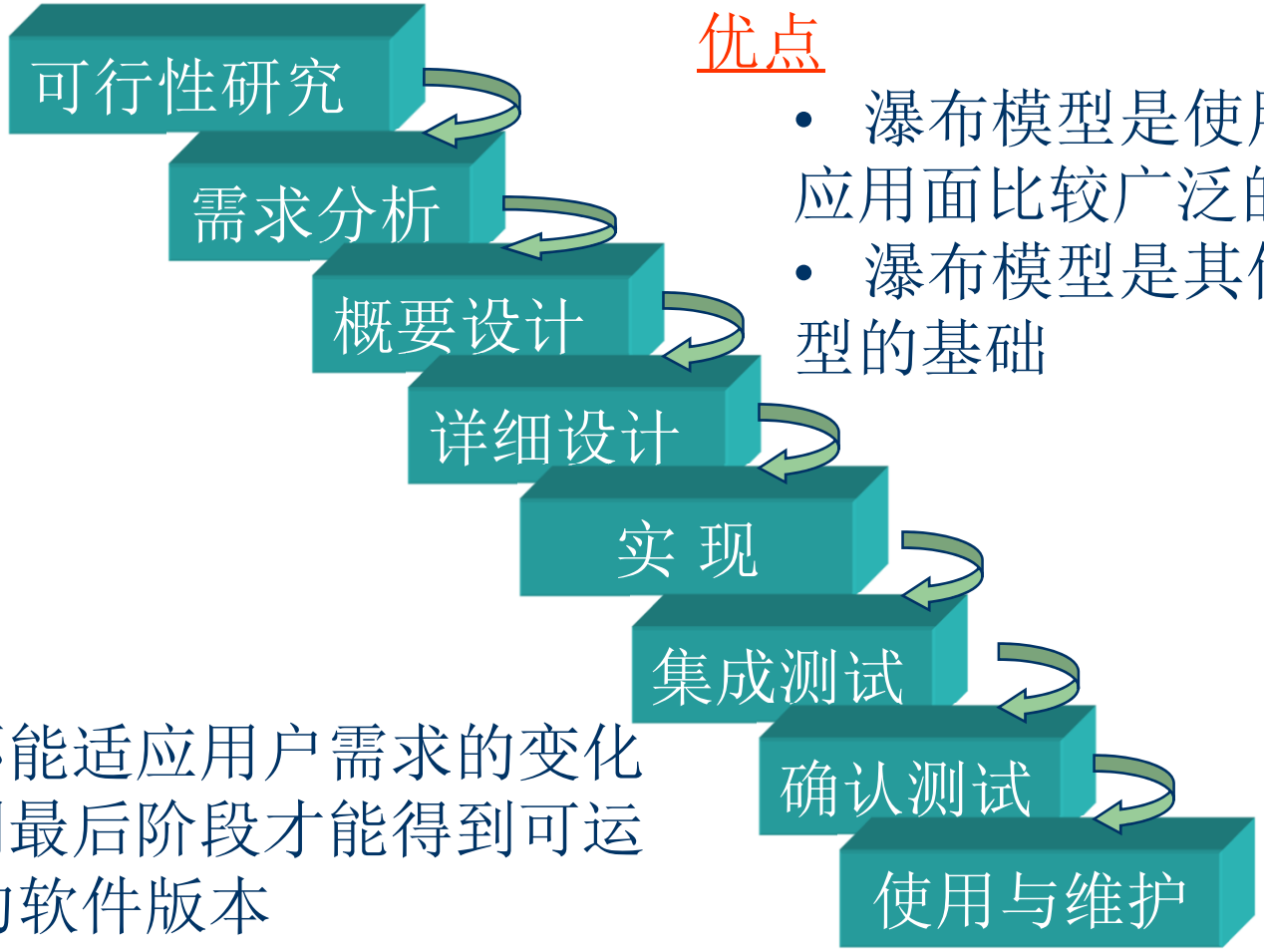
从软件项目需求定义到软件使用后废止，针对系统开发、运作和维持所实施的全部过程、活动和任务结构框架

- 瀑布模型
- 演化模型
- 螺旋模型
- 喷泉模型
- RUP

瀑布模型(1/2)

- 假设软件需求在初期几乎可完全确定
- 主要思想
  - ✓ 软件开发过程与软件生命周期是一致的
  - ✓ 将基本的过程活动看成是界限分明的独立的过程阶段
  - ✓ 相邻二阶段之间存在因果关系
  - ✓ 每一阶段的结束点作为里程碑
  - ✓ 需对阶段性产品进行评审，如果评审不合格，需返工

瀑布模型(2/2)



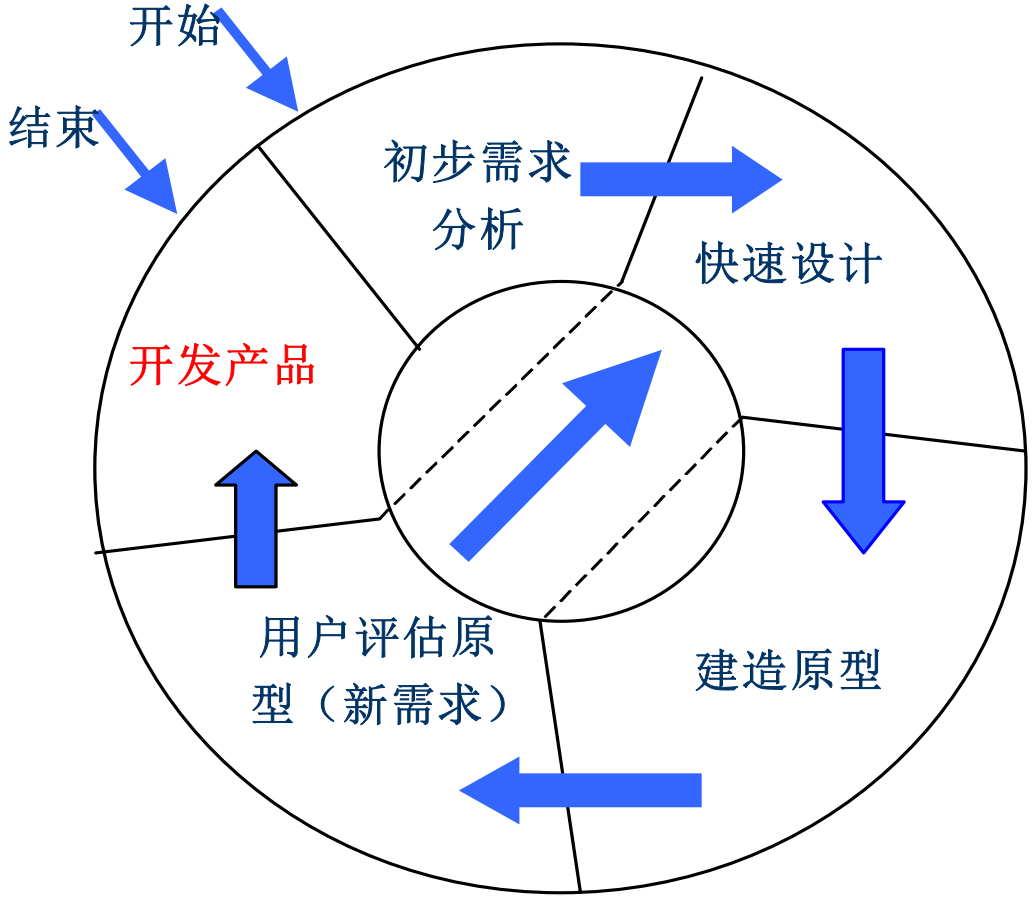
优点

- 瀑布模型是使用时间最长、应用面比较广泛的开发模型
- 瀑布模型是其他一些开发模型的基础

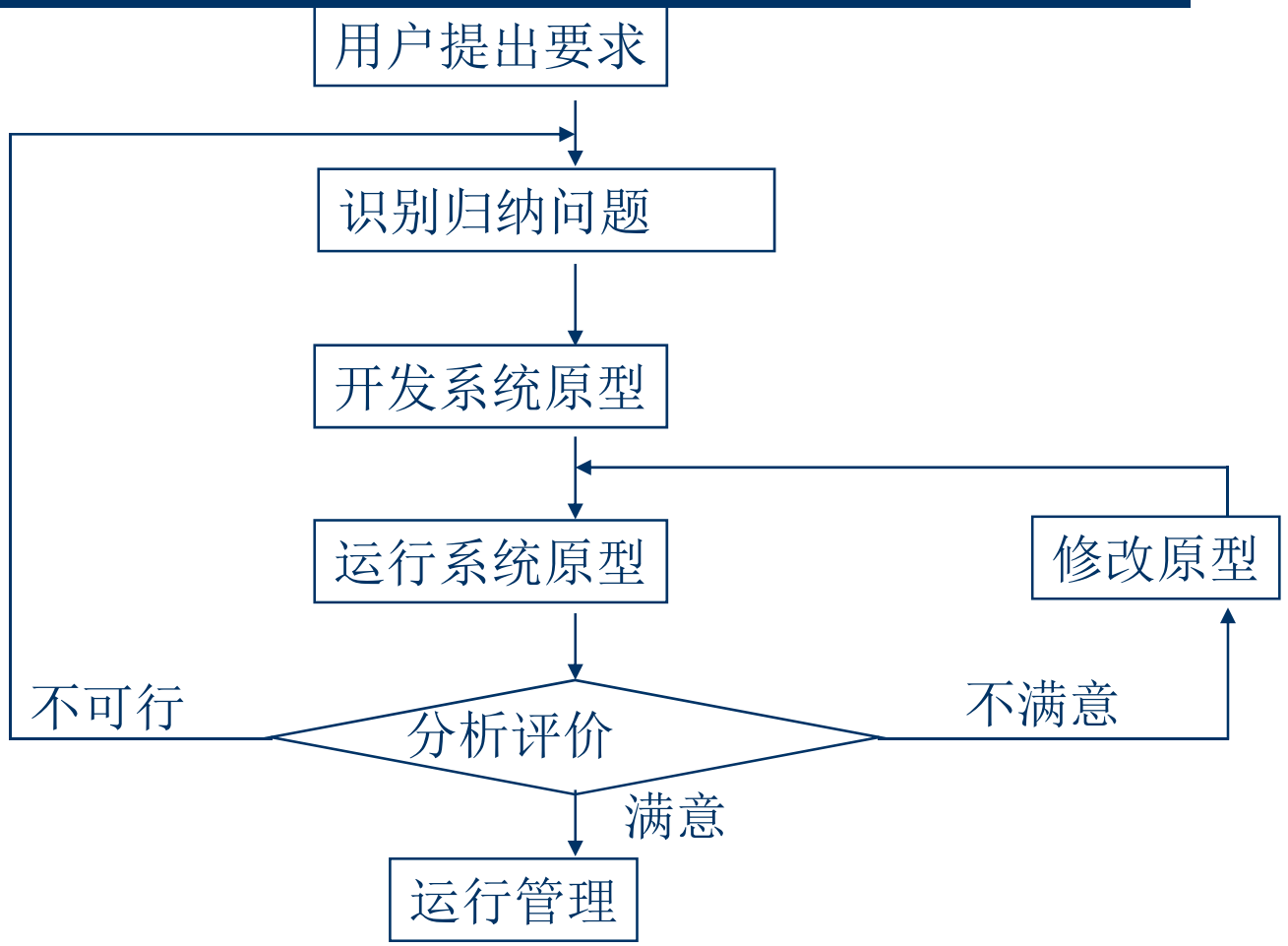
缺陷

- 不能适应用户需求的变化
- 到最后阶段才能得到可运行的软件版本

演化模型（原型模型）(1/3)



演化模型（原型模型） (2/3)



演化模型（原型模型）(3/3)

- 优点
  - ✓ 支持需求的动态变化
  - ✓ 有助于获取用户需求，便于用户对需求的理解
  - ✓ 尽早发现软件中的错误
- 缺陷
  - ✓ 需要为系统的每个新版本交付文档，不划算
  - ✓ 新需求的不断增加，使系统结构退化，变更成本上升
  - ✓ 不支持风险分析

螺旋模型(1/2)

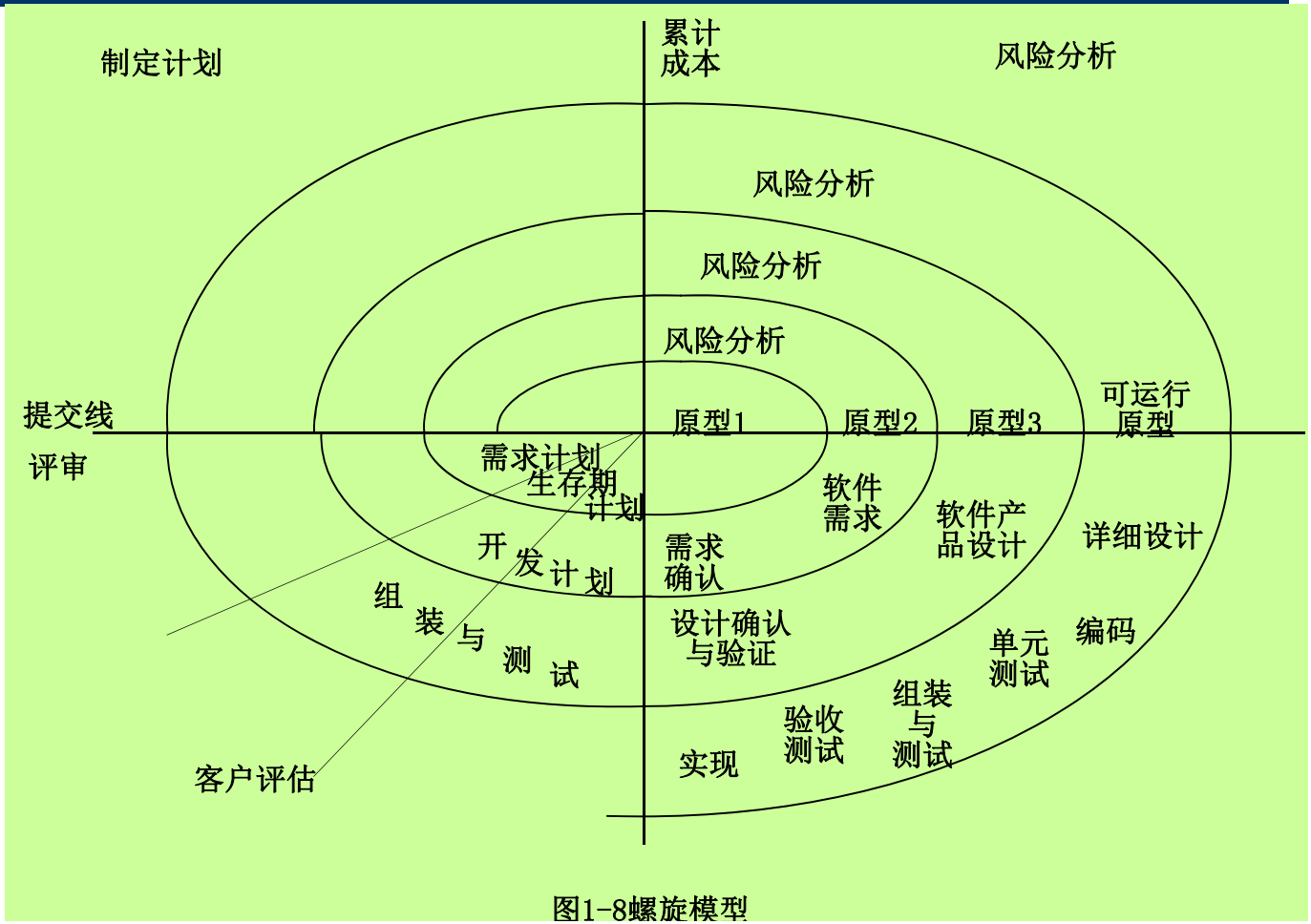
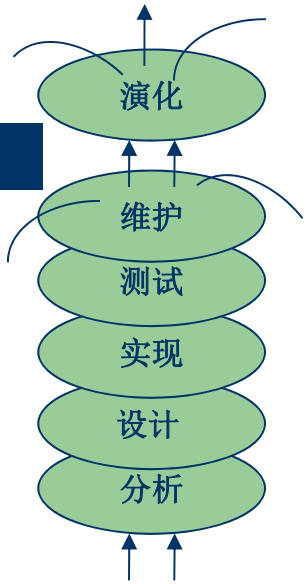


图1-8螺旋模型

螺旋模型(2/2)

- 基本思想
  - ✓ 将瀑布模型与原型模型进行有机结合
  - ✓ 增加风险分析步骤
- 优点
  - ✓ 支持需求的动态变化
  - ✓ 有助于获取用户需求，便于用户对需求的理解
  - ✓ 尽早发现软件中的错误
  - ✓ 支持风险分析，可降低或者尽早消除软件开发风险
  - ✓ 适合于需求动态变化、开发风险较大的系统

喷泉模型



- 基本思想

软件复用与生命周期中多项开发活动集成

主要支持面向对象的开发方法

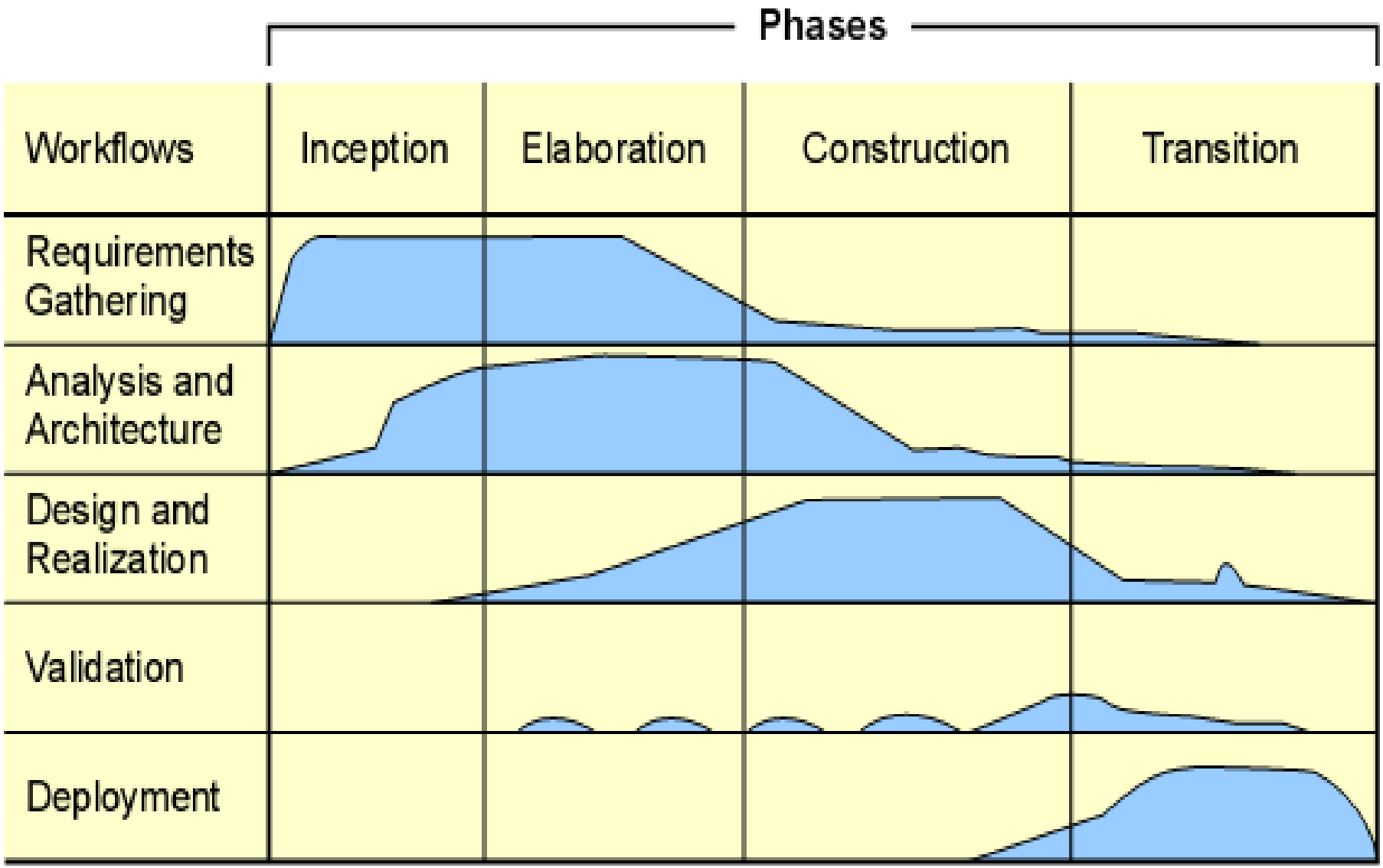
- 优点

1. 软件系统可维护性较好；
2. 各阶段相互重叠，表明了面向对象开发方法各阶段间的交叉和无缝过渡；
3. 整个模型是一个迭代的过程，包括一个阶段内部的迭代和跨阶段的迭代；
4. 模型具有增量开发特性，即能做到“分析一点、设计一点、实现一点，测试一点”，使相关功能随之加入到演化的系统中。
5. 模型由对象驱动，对象是各阶段活动的主体，也是项目管理的基本内容。
6. 该模型很自然地支持软部件的重用。

Rational统一过程（课本第2章）

- 统一软件开发过程是由Rational的Booch，Jacobson，和Rumbaugh创建的方法的公开版本。它也被称为统一过程（RUP）。
- 四个阶段：
  - 概念阶段——创建软件远景模型。
  - 细化——在系统架构上添加已定义好的绝大部分用例。
  - 构建——构造软件。
  - 移交——把软件从测试版转换成产品。

每个阶段都可能有多重迭代。


Phases
Workflows
Inception
Elaboration
Construction
Transition
Requirements Gathering
Analysis and Architecture
Design and Realization
Validation
Deployment

开始阶段的成果

- 一个最初的用例模型（10%-20%）
- 一个最初的项目词汇表
- 一个最初的商业用例
- 一个最初的风险评估
- 一个项目规划，明确阶段和迭代
- 一个商业模型
- 一个或多个原型

细化阶段的成果

- 用例模型（〉80%），识别所有的用例和角色；
- 一些增加的描述，包括非功能性需求及于特定用例无关的需求；
- 软件构架描述和一个可执行的构架原型；
- 一个修正后的风险表和商业用例；
- 一份整个项目的开发规划，包括粗略项目规划及对每个迭代的评价准则；
- 一个更新的开发用例，指定要使用的过程；
- 一份初步的用户手册

构造阶段的成果

- 构造阶段的产品是一个可以立即提交给它的端客户使用的产品，至少应包括：
- 在足够的平台上集成的软件产品；
- 用户手册；
- 对当前版本的描述。

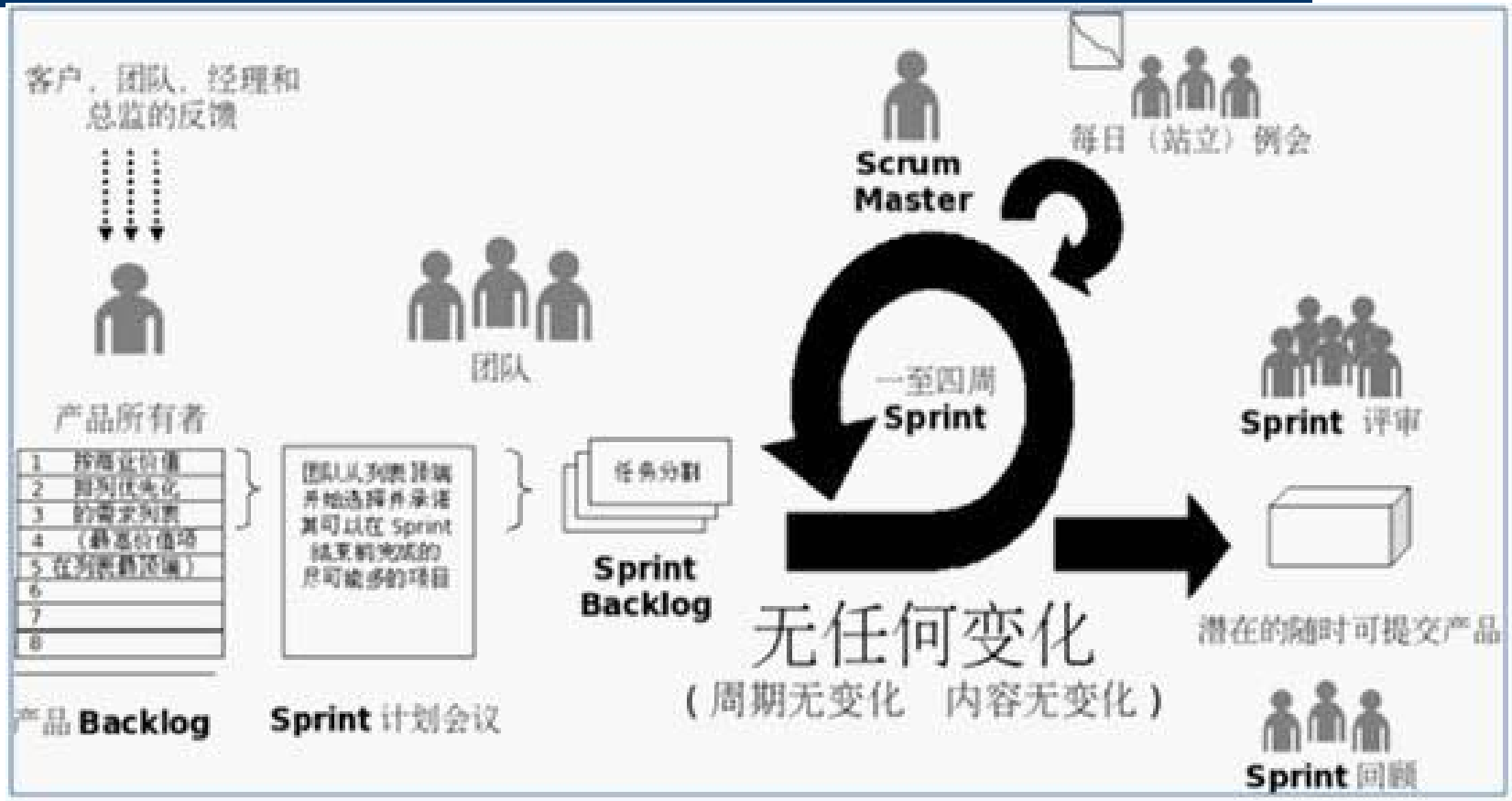
过渡阶段的工作内容

- 建立用户使用环境（典型beta用户，产品安装环境）；
- 进行基础数据准备和用户培训；
- 进行β测试和系统确认测试；
- 进行软件适应性修改和正式版本生成；
- 准备用户使用手册和相关技术文档；
- 软件开发技术总结、效果评估和产品发布

敏捷开发（Agile）

- 敏捷软件开发是一种从1990年代开始逐渐引起广泛关注的一些新型软件开发方法，是一种应对快速变化的需求的一种软件开发能力。
- 强调程序员团队与业务专家之间的紧密协作、面对面的沟通、频繁交付新的软件版本、紧凑而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重软件开发中人的作用。

敏捷流程-以Scrum为例



敏捷开发方法

- 测试驱动开发（Test-Driven Development，TDD）
- 极限编程（Extreme Programming，XP）
- Scrum
- **FDD**
- **BDD**

TDD

- 基本思想：在开发功能代码之前，先编写测试代码
  - 在明确要开发某个功能后，首先思考如何对这个功能进行测试，并完成测试代码的编写，然后编写相关的代码满足这些测试用例。然后循环进行添加其他功能，直到完全部功能的开发。

TDD

- 过程

1） 明确当前要完成的功能，记录成一个 TODO 列表。
2） 快速完成针对此功能的测试用例编写。
3） 测试代码编译不通过。
4） 编写对应的功能代码。
5） 测试通过。
6） 对代码进行重构，并保证测试通过。
7） 循环完成所有功能的开发。

TDD

- 适用场合
  - ✓ 新软件的开发
  - ✓ 历史系统的维护

XP

- XP是一个轻量级的、灵巧的软件开发方法
- 它的基础和价值观是交流、朴素、反馈和勇气，即任何一个软件项目都可以从四个方面入手进行改善：
  - ✓ 加强交流
  - ✓ 从简单做起
  - ✓ 寻求反馈
  - ✓ 勇于实事求是

XP

- XP是一种近螺旋式的开发方法
- 将任务/系统细分为可以在较短周期解决的一个个子任务/模块
- 强调测试、代码质量和及早发现问题
- 通过一个个短小的迭代周期，获得一个个阶段性的进展，形成一个版本供用户参考，以便及时对用户可能的需求变更作出响应。

XP

- 特征
  - 增量和反复式的开发
  - 反复性，通常是自动重复的单元测试，回归测试
  - 结对编程
  - 客户是开发团队一份子
  - 软件重构
  - 共享的代码所有权
  - 简单
  - 反馈
  - 。。。

XP

- 适用场合
  - ✓ 小而精的团队
  - ✓ 可测的用户需求

敏捷开发

- 更多关于敏捷开发的内容可阅读 邹欣的《构建之法-现代软件工程》第6章 敏捷流程
- http://www.cnblogs.com/xinz/archive/2012/10/05/2712602.html
- http://www.cnblogs.com/xinz/archive/2011/04/27/2031118.html

如何选择开发方法

- 对于一个给定的工程而言，下列因素将会引导你选择不同的方法。
  - 公司文化——面向过程或者面向产品。
  - 团队的组成——缺少经验的开发者可能更加依靠团队的组织协调与安排，而且他们只能承担一个角色。
  - 项目规模——一个大的项目可能需要更多的文档（与利益相关人之间的交流）。
  - 需求的稳定性——需求变更的频率。

4. 软件工程的目标(1/3)

- 总体目标

  在给定成本、进度的前提下，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性并满足用户需要的软件产品

4. 软件工程的目标(2/3)

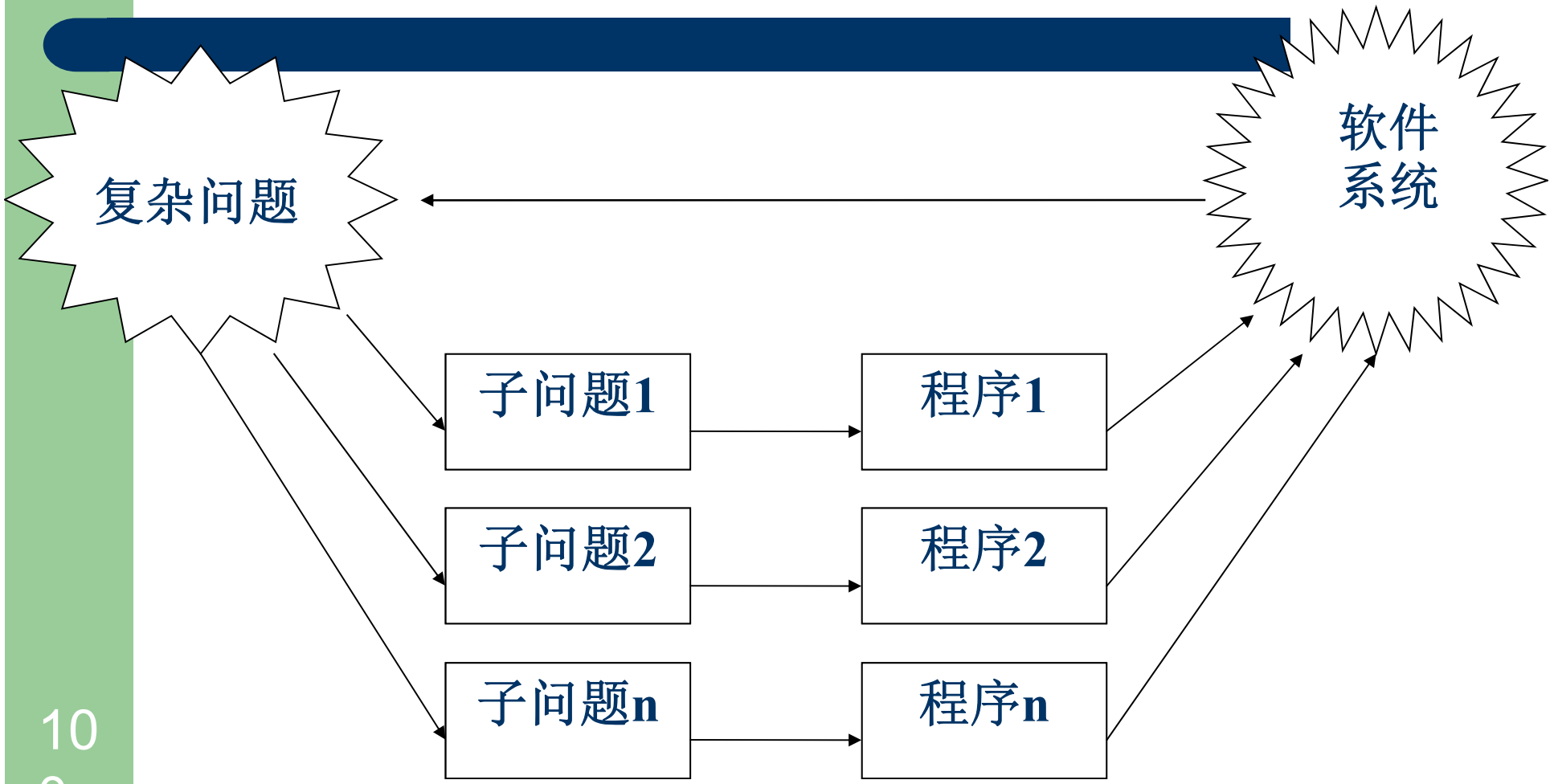
- 正确性
  - 满足用户的需求(功能、性能等)
- 可靠性
  - 具有能够防止因概念、设计和结构等方面的不完善而造成的系统失效，具有挽回因操作不当而造成软件系统失效的能力
- 有效性
  - 充分利用计算机的时间和空间资源

4. 软件工程的目标(3/3)

- 可维护性
  - 便于对软件增加新功能、改进性能、修改错误
- 可重用性
  - 软件（部分）易于被再次使用
- 可追踪性
  - 对软件进行正向和反向追踪的能力
- 可移植性
  - 从一个环境搬迁到另一个环境的难易程度
- 可互操作性
  - 多个软件元素相互通讯、协同完成任务的能力

5. 软件工程的原则(1/3)

- 抽象
  - 关注事物基本、重要的部分，忽略不相关成分
  - 聚焦于问题本质，简化问题，分阶段控制问题复杂度，有条不紊地推进复杂、庞大软件系统开发
- 模块化
  - 程序模块：逻辑上相对独立、具有良好的接口定义的编程单位，例如过程、函数、类、程序包等
  - 模块化：将复杂的系统分解为由多个相对独立的模块加以实现，有助于抽象、信息隐藏以及表示复杂的系统


复杂问题
软件
系统
子问题1
程序1
子问题2
程序2
子问题n
程序n

5 软件工程的原则(2/3)

- 信息隐藏
  - 将模块中的软件实现信息封装起来，外部只知道其功能以及对外接口，而不知道其内部细节
  - 有助于控制修改的影响传播范围
  - 有助于软件开发人员的注意力集中于更高的抽象层次
- 局部化
  - 缩小软件元素的作用范围
  - 在物理模块内集中逻辑上相互关联的计算资源
  - 确保模块内各成分关系密切而模快之间的关系松散，保证模块具有良好的独立性

5 软件工程的原则(3/3)

- 一致性
  - 整个软件系统和开发过程均使用统一的符号、概念和术语
  - 文档与程序一致
- 完备性
  - 整个软件系统不丢失任何重要的成分，软件完全实现系统所需的功能、行为和性能
- 可验证性
  - 软件系统应易于检查、测试和评审

小结

- 软件是程序以及相关文档的集合
- 软件危机的主要原因之一在于：缺乏系统的方法和工具
- 软件工程：过程、方法和工具
- 软件工程的目标和原则

计算机辅助软件工程

- 计算机辅助软件工程(Computer Aided Software Engineering, CASE)
  - ✓ 在软件工程活动中，开发人员按照软件工程原则、方法和技术，借助计算机及其软件工具的帮助来开发、管理和维护软件产品的过程
- CASE工具
  - ✓ 支持CASE的工具(语法制导编辑器，编译器，调试工具等)

软件工程中的一些观念讨论

- 我们拥有一套讲述如何开发软件的书籍，书中充满了标准与示例，可以帮助我们解决软件开发中遇到的任何问题。
- 我们拥有最好的开发工具、最好的计算机，一定能做出优秀的软件。
- 如果我们落后于计划，可以增加更多的程序员来解决。
- 既然需求分析很困难，不管三七二十一先把软件做了再说，反正软件是灵活的，随时可以修改。

软件工程中的一些观念讨论

- 如果软件运行较慢，是换一台更快的计算机，还是设计一种更快的算法？
- 有最好的软件工程方法，最好的编程语言吗？
- 编程时是否应该多使用技巧？
- 软件中的错误是否可按严重程度分等级？