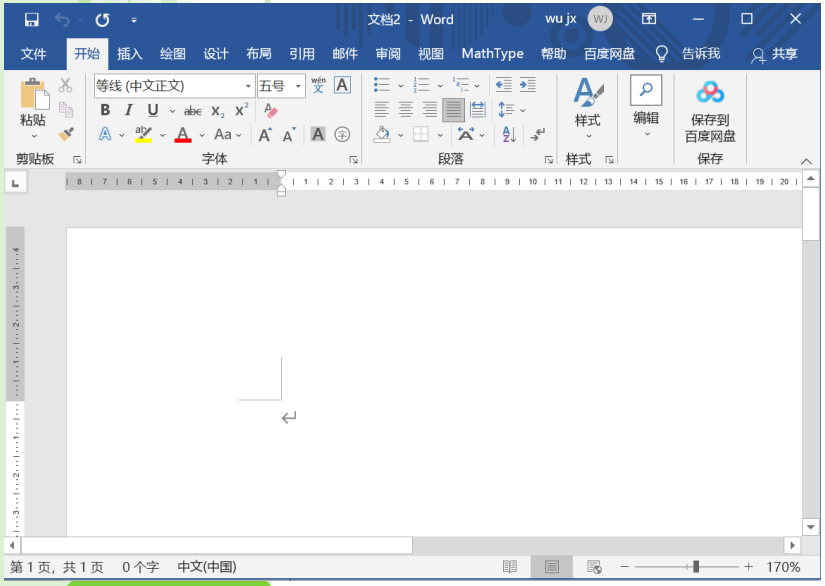
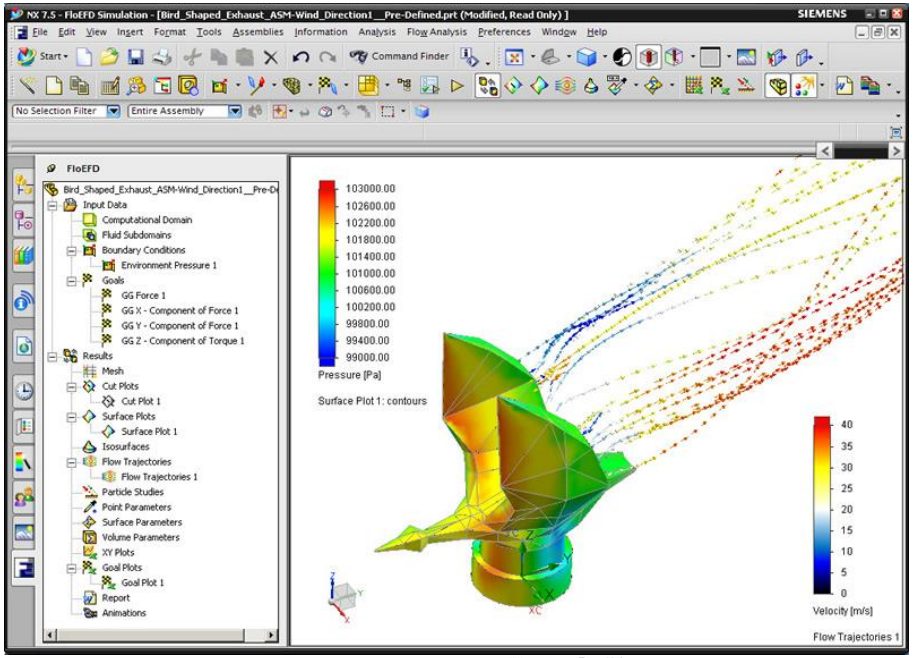
创新训练课B

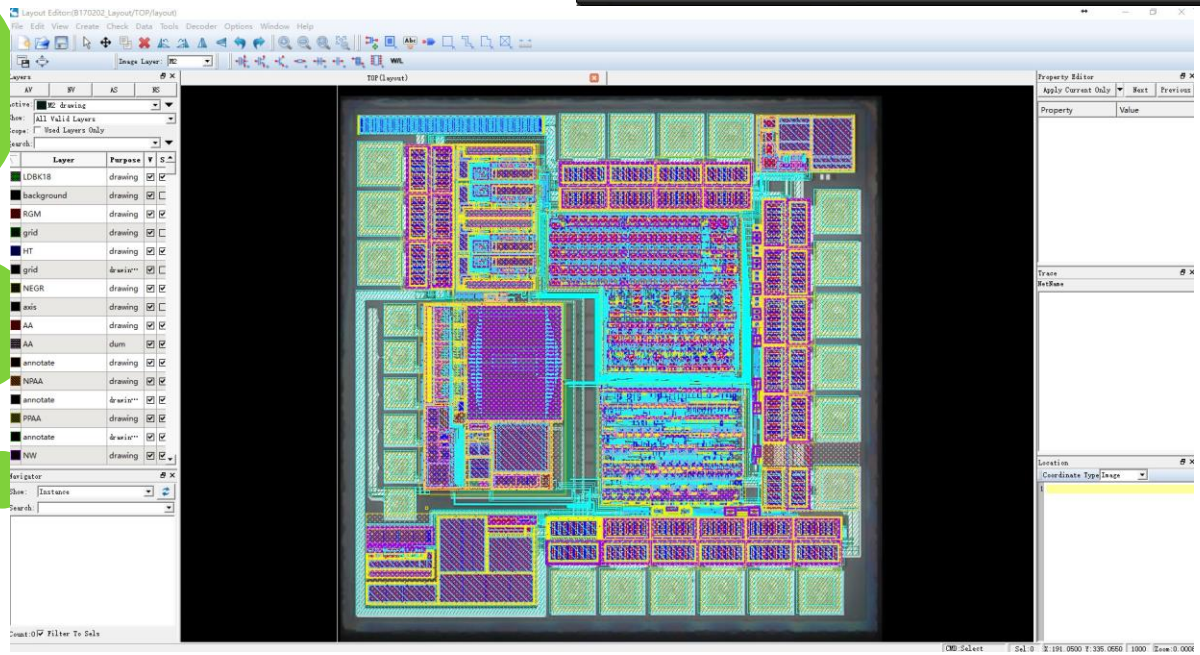
主讲人：吴晓军

邮　箱：wuxj@hit.edu.cn

大型软件


文档2 - Word
wu jx
文件 开始 插入 绘图 设计 布局 引用 邮件 审阅 视图 MathType 帮助 百度网盘 告诉我 共享
等线 (中文正文)
五号
粘贴
剪贴板
字体
段落
样式
编辑
保存到百度网盘
保存
第 1 页，共 1 页 0 个字 中文(中国)
170%


NX 7.5 - FloEFD Simulation - [Bird_Shaped_Exhaust_ASM-Wind_Direction1__Pre-Defined.prt (Modified, Read Only)]
SIEMENS
FloEFD
Input Data
Computational Domain
Fluid Subdomains
Boundary Conditions
Environment Pressure 1
Goals
GG Force 1
GG X - Component of Force 1
GG Y - Component of Force 1
GG Z - Component of Torque 1
Results
Mesh
Cut Plots
Cut Plot 1
Surface Plots
Surface Plot 1
Isosurfaces
Flow Trajectories
Flow Trajectories 1
Particle Studies
Point Parameters
Surface Parameters
Volume Parameters
XY Plots
Goal Plots
Goal Plot 1
Report
Animations
103000.00
102600.00
102200.00
101800.00
101400.00
101000.00
100600.00
100200.00
99800.00
99400.00
99000.00
Pressure [Pa]
Surface Plot 1: contours
40
35
30
25
20
15
10
5
0
Velocity [m/s]
Flow Trajectories 1

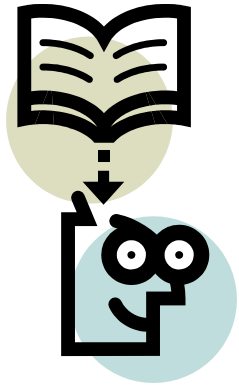


创新训练课B

软件工程导论

软件工程简介

目标



- 软件工程目标
  - 有助于正确理解和认识“软件”的概念及其特点
  - 理解软件开发面临的问题和挑战
  - 掌握软件工程的原则、方法和思想来系统地开发软件，尤其是复杂、庞大的软件的开发
  - 了解和接触软件开发所需的技术手段

软件工程的目标是提高软件的质量和生产率，最终实现软件的工业化生产。软件工程是如何规范化的开发软件。

运用

- 运用工程化思想进行软件开发
  - 需求分析
  - 软件设计
  - 程序设计
  - 软件维护

先导知识要求

- 程序设计语言----C语言程序设计

参考文献

- 软件工程导论（第六版），张海藩，清华大学出版社

1 软件工程概述

- 软件工程产生的背景(软件危机)
- 软件工程定义
- 软件工程方法学
- 软件过程模型
- 小结

1.1 软件工程产生的背景

1、什么是软件？

a. 软件的定义

程序是不是软件？

1.1 软件工程产生的背景

1、什么是软件？

a. 软件的定义

软件（ Software）是计算机系统中与硬件相互依存的另一部分，它是包括程序（Program），数据（Data）及其相关文档（ Document）的完整集合。

Software = Program + Data + Document

- 程序是按事先设计的功能和性能要求执行的指令序列
- 数据是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 软件工程产生的背景

b、软件的特征

非常复杂

- 逻辑复杂
  - 远远高于硬件的逻辑复杂度
- 开发复杂
  - 成本难以估算
  - 进度难以控制
  - 人员素质要求
  - 质量得不到保证
    - 96年Ariane火箭发射失败，浮点数转换时发生错误

1.1 软件工程产生的背景

b、软件的特征

成本高(1/2)

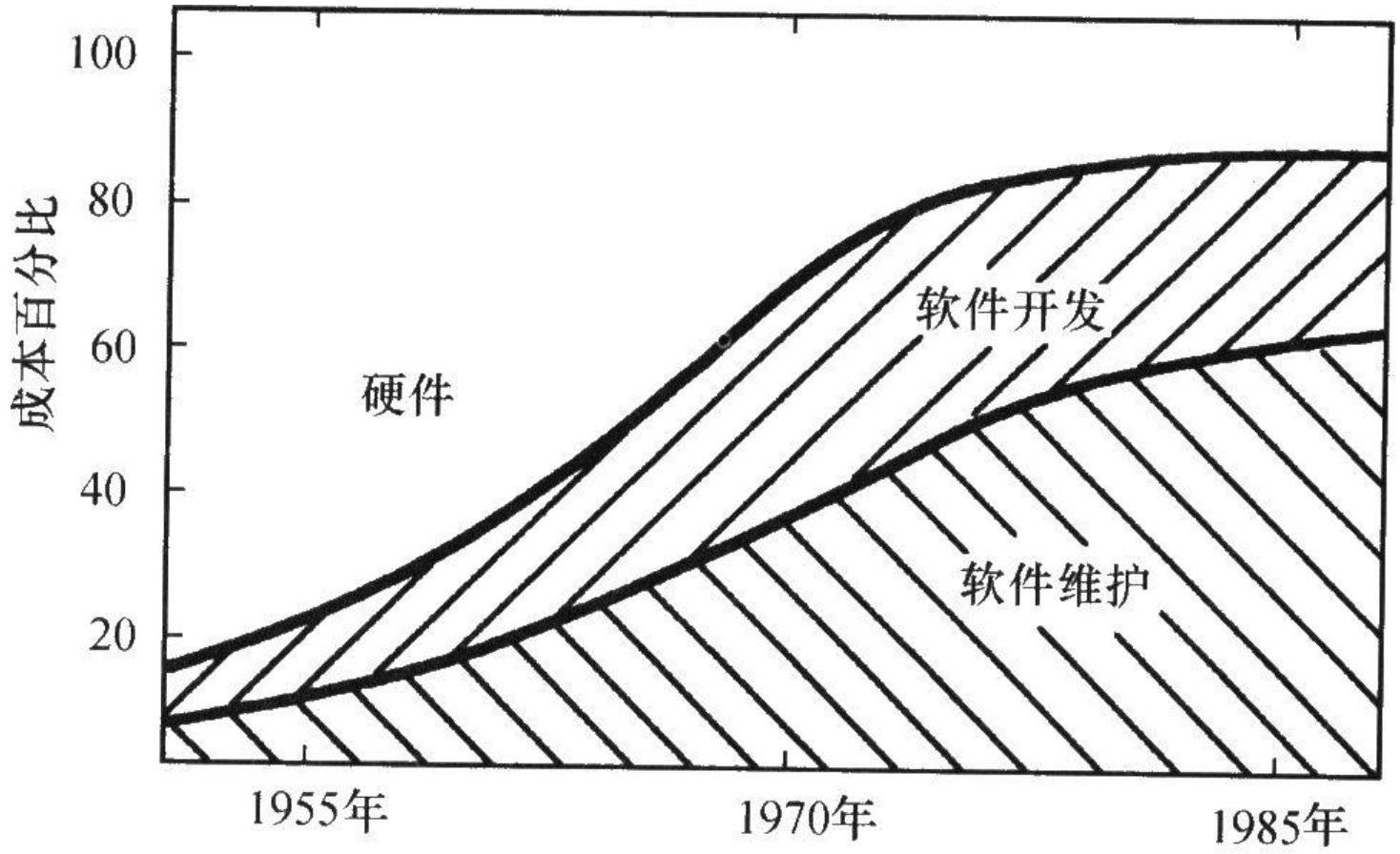


图 1.1　硬件/软件成本变化趋势

成本高(2/2)

产品	代码行	工 作 量（人年）	成本(百万)
Lotus1-2-3 Version 3.0	400k	263	22
Space Shuttle	25.6M		12
1989 Lincoln Continental	83.5M	35	1.8
City Bank Teller machine	780k	150	13.2
IBM Chechout Scanner	90k	58	3

1.1 软件工程产生的背景

b、软件的特征

风险大

- 1995年美国Standish咨询集团的统计分析(至90年代初的软件项目执行情况)
  - 成功：16.2%
  - 失败：31%
  - 受到挑战：53.8%
- 近几年来的统计数据
  - 成功：26%
  - 失败：28%
  - 受到挑战：46%

1.1 软件工程产生的背景

b、软件的特征

维护困难

- 维护形式多样化
  - 改正性：修改故障
  - 完善性：增加功能
  - 适应性：移植
- 维护成本越来越高
  - 55%到70%
- 维护带来的问题

1.1 软件工程产生的背景

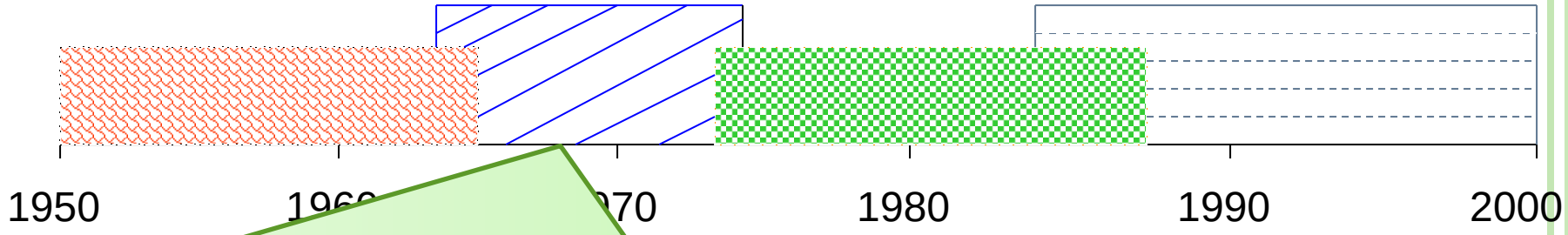
c. 软件的发展

早期
- 面向批处理
- 有限的分布
- 自定义软件

第二阶段
- 多用户
- 实时
- 数据库
- 软件产品

第三阶段
- 分布式系统
- 嵌入“智能”
- 低成本硬件
- 消费者的影响

第四阶段
- 强大的桌面系统
- 面向对象技术
- 专家系统
- 人工神经网络
- 并行计算
- 网路计算机



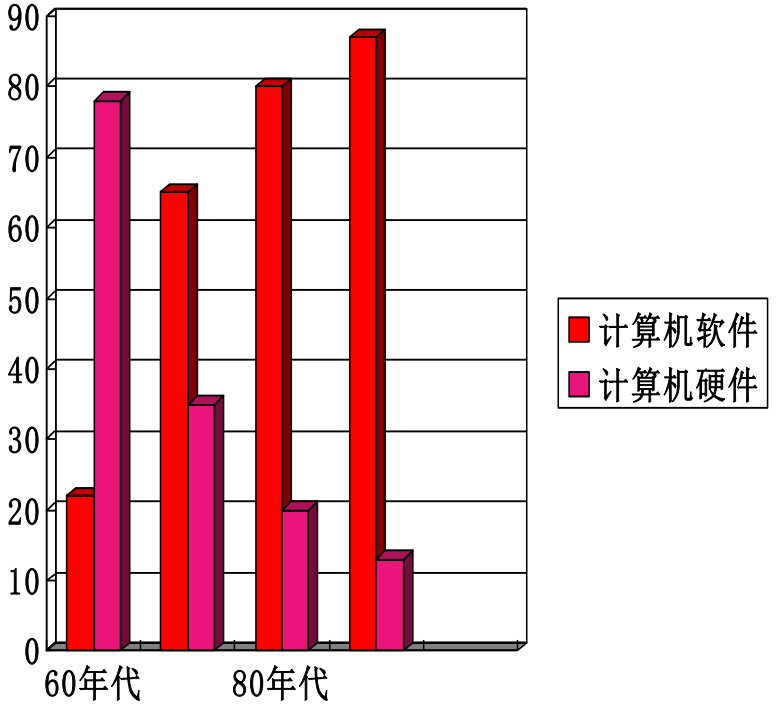
1968年10月，北大西洋公约组织（NATO）的科学家在德国召开的学术会议上正式提出了软件危机问题。

1.1 软件工程产生的背景

2、软件危机

a、软件危机的表现（1/3）

- 成本高
  - IBM 360 OS, 5000多人年，耗时4年(1963－1966)，花费2亿多美元
  - 美国空军：1955年软件占总费用(计算机系统)的18%，70年60%，85年达到85%
  - 美国全球军事指挥控制系统，硬件1亿美元，软件高达7.2亿美元
- 计算机软件和硬件费用比



2、软件危机

a、软件危机的表现（2/3）

- 软件质量得不到保证
  - 软件应用面的扩大：科学计算、军事、航空航天、工业控制、企业管理、办公、家庭
  - 软件越来越多的应用于安全攸关(safety critical)的系统，对软件质量提出更高的要求
  - 80年代欧洲亚丽安娜火箭的发射失败，原因是软件错误
  - 美国阿托拉斯火箭的发射失败，原因是软件故障
  - 英国1986年开发的办公室信息系统Folios经4年，因性能达不到要求，1989年取消
  - 日本第5代机因为软件问题在投入50亿美元后于1993年下马
- 由于软件质量问题导致失败的软件项目非常多

1.1 软件工程产生的背景

2、软件危机

a、软件危机的表现（3/3）

- 进度难以控制
  - 项目延期比比皆是
  - 由于进度问题而取消的软件项目较常见
  - 只有一小部分的项目能够按期完成

- 维护非常困难
  - 软件维护的多样性
  - 软件维护的复杂性
  - 软件维护的副作用

1.1 软件工程产生的背景

2、软件危机

b、产生软件危机的原因

- 与软件本身的特点有关 (难于维护, 逻辑复杂)
- 与软件开发与维护的方法不正确有关:
  - 软件≠程序
  - 急于求成=拔苗助长
  - 各自为阵，无方法/学

1.1 软件工程产生的背景

2、软件危机

C、软件工程（学）因危机而产生

- 开发一个具有一定规模和复杂性的软件系统与编写一个简单的程序不一样
  - 正如建设茅草屋和高楼大厦

- 大型、复杂软件系统的开发是一项工程，必须按照工程化的方法组织软件的生产和管理，必须经过分析、设计、实现、测试、维护等一系列软件过程和活动

1.1 软件工程产生的背景

2、软件危机

d、软件工程(学):克服软件危机的努力

（1）从管理的角度

软件开发过程的研究、文档的标准化以及人们的交流方式等

（2）软件开发方法的研究

结构化软件开发方法，面向对象的开发

必须充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。

1.1 软件工程产生的背景

1、解决危机的技术途径

- 提出有效的方法和工具支持软件开发
  - 1968年提出软件工程概念和思想
  - 20世纪70年代的结构化软件开发方法
  - 20世纪80年代的面向对象的软件开发方法
  - 新的技术: 软件重用、快速原型、需求工程
  - 典型技术: COM, Java, C++, J2EE, .Net, ….
  - 支撑工具和环境：Jbuilder, Visual Studio, WebLogic, …

1.1 软件工程产生的背景

2、解决危机的管理途径

- 20世纪80年代末，美国DoD(Department of Defense)和工业界开始认识到管理的重要性
  - 美国DoD的一项研究表明，70%的项目由于管理不善导致难以控制进步、成本和质量；
  - 进一步的研究发现：管理是影响软件项目成功开发的全局性因素，而技术只影响局部
  - 如果软件开发组织不能对软件项目进行有效管理，就不能充分发挥软件开发方法和工具的潜力，也就不能高效率地开发出高质量的软件产品

1.2 软件工程定义

总之：

软件工程是应用计算机科学、数学及管理科学等原理开发软件的工程。它借鉴传统工程的原则、方法，以提高质量，降低成本为目的。

1.3 软件工程方法学

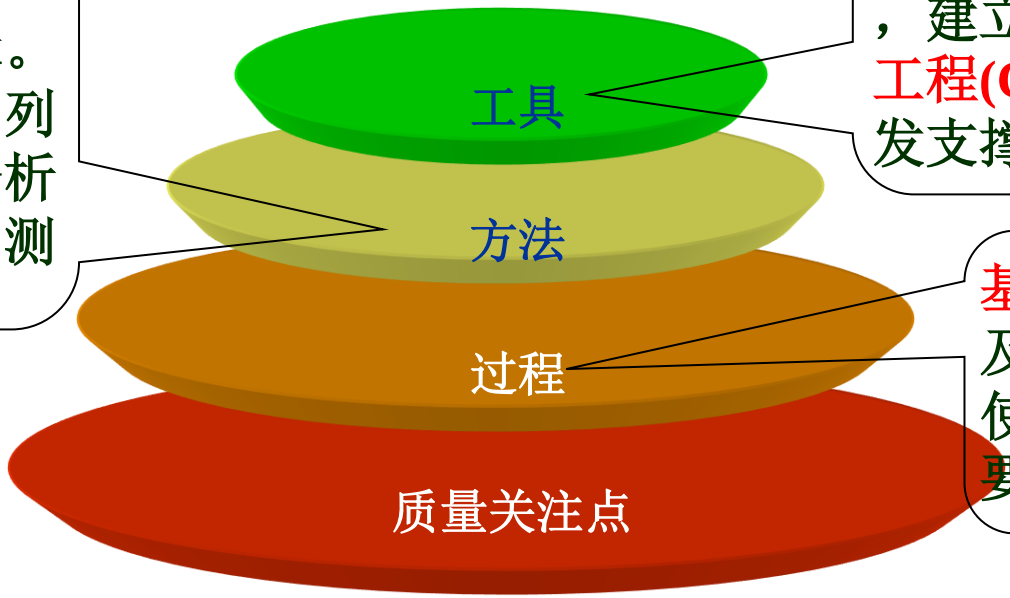
- 通常把在软件生命周期全过程中使用的一整套技术方法的集合称为软件工程方法学(methodology)，也称为范型(paradigm)。
- 在软件工程领域中，这两个术语的含义基本相同。
- 软件工程方法学三要素：方法、工具和过程

1.3 软件工程方法学

软件工程： 一种层次化技术

为软件开发提供“如何做”的技术。方法涵盖了一系列的任务：需求分析、设计、编程、测试和维护。

为软件开发提供自动或半自动的软件支撑环境，建立计算机辅助软件工程(CASE)的软件开发支撑系统

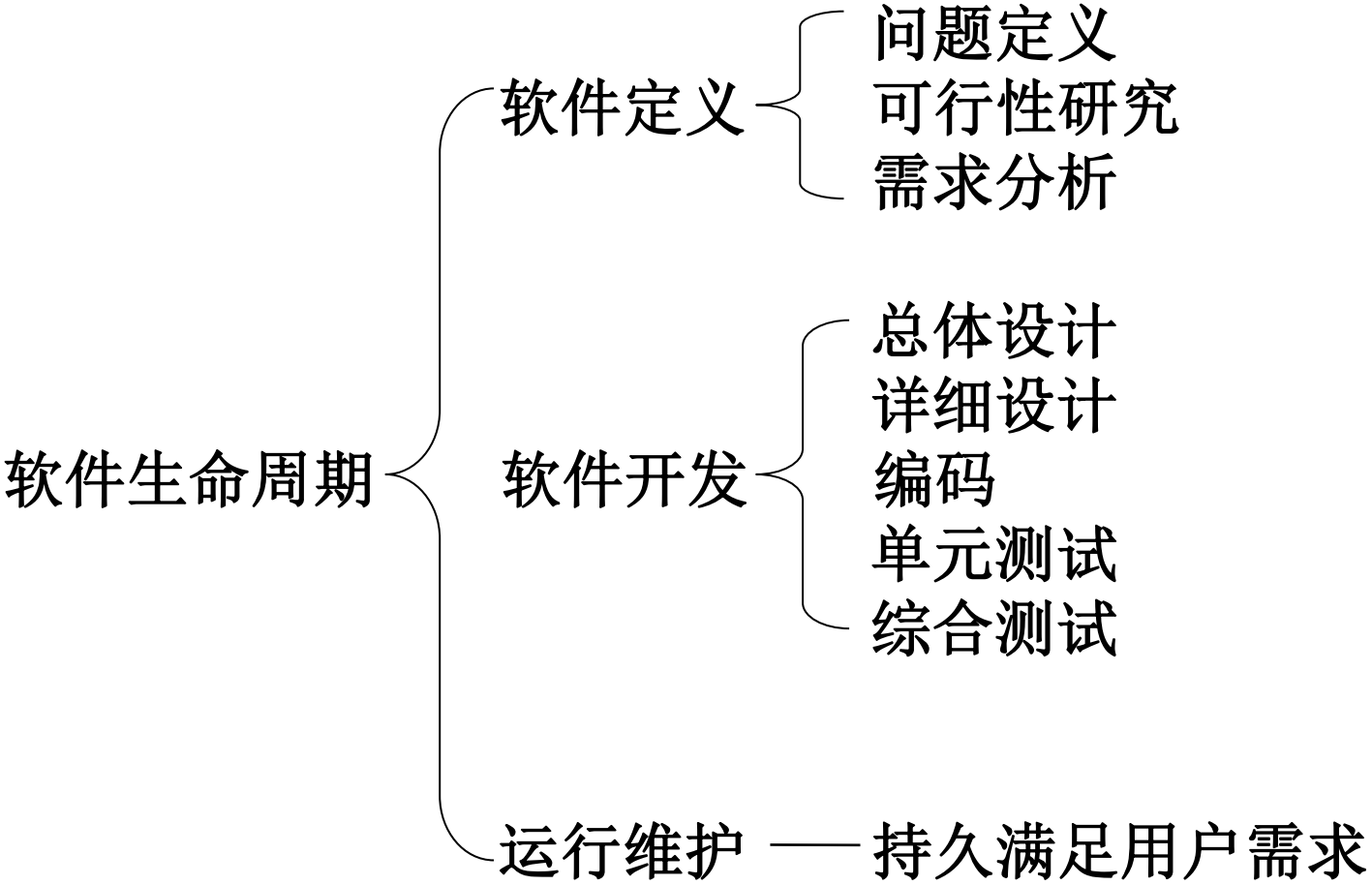


基础层，综合方法及工具，定义方法使用的顺序，所需要的管理

软件工程层次图

软件工程三个要素：工具、方法、过程

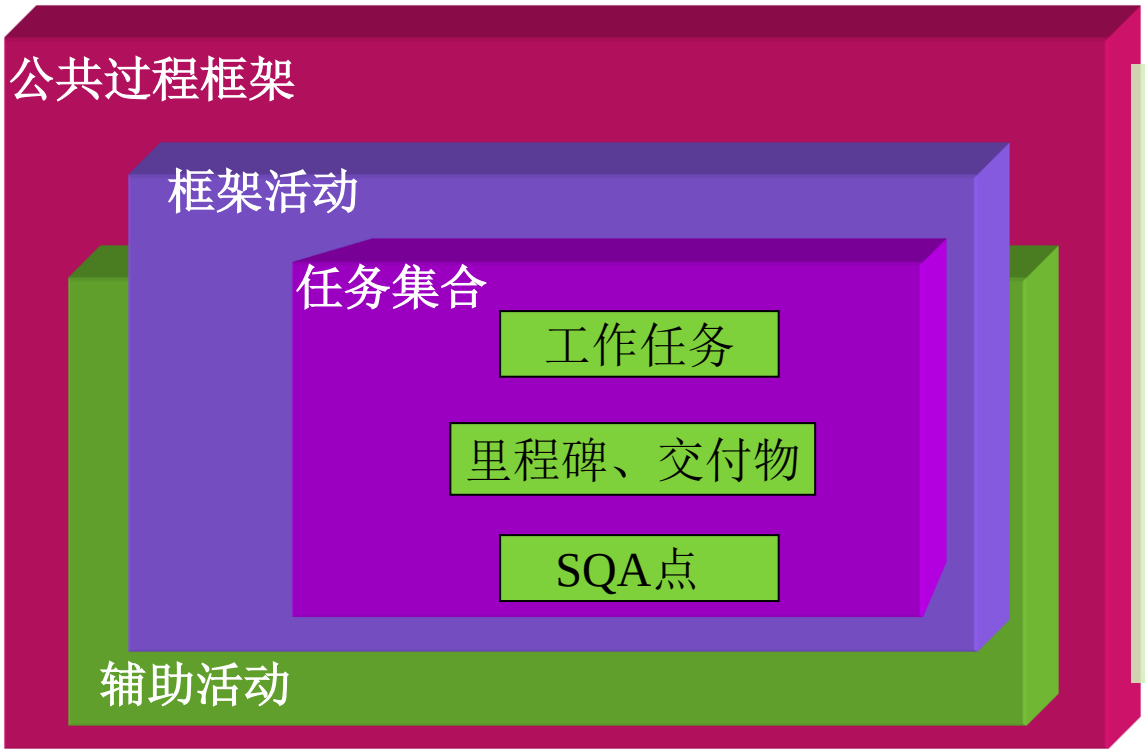
1.4 软件生命周期模型


软件生命周期
软件定义
问题定义
可行性研究
需求分析
软件开发
总体设计
详细设计
编码
单元测试
综合测试
运行维护
持久满足用户需求

1.4 软件过程模型

软件过程

软件过程是为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。



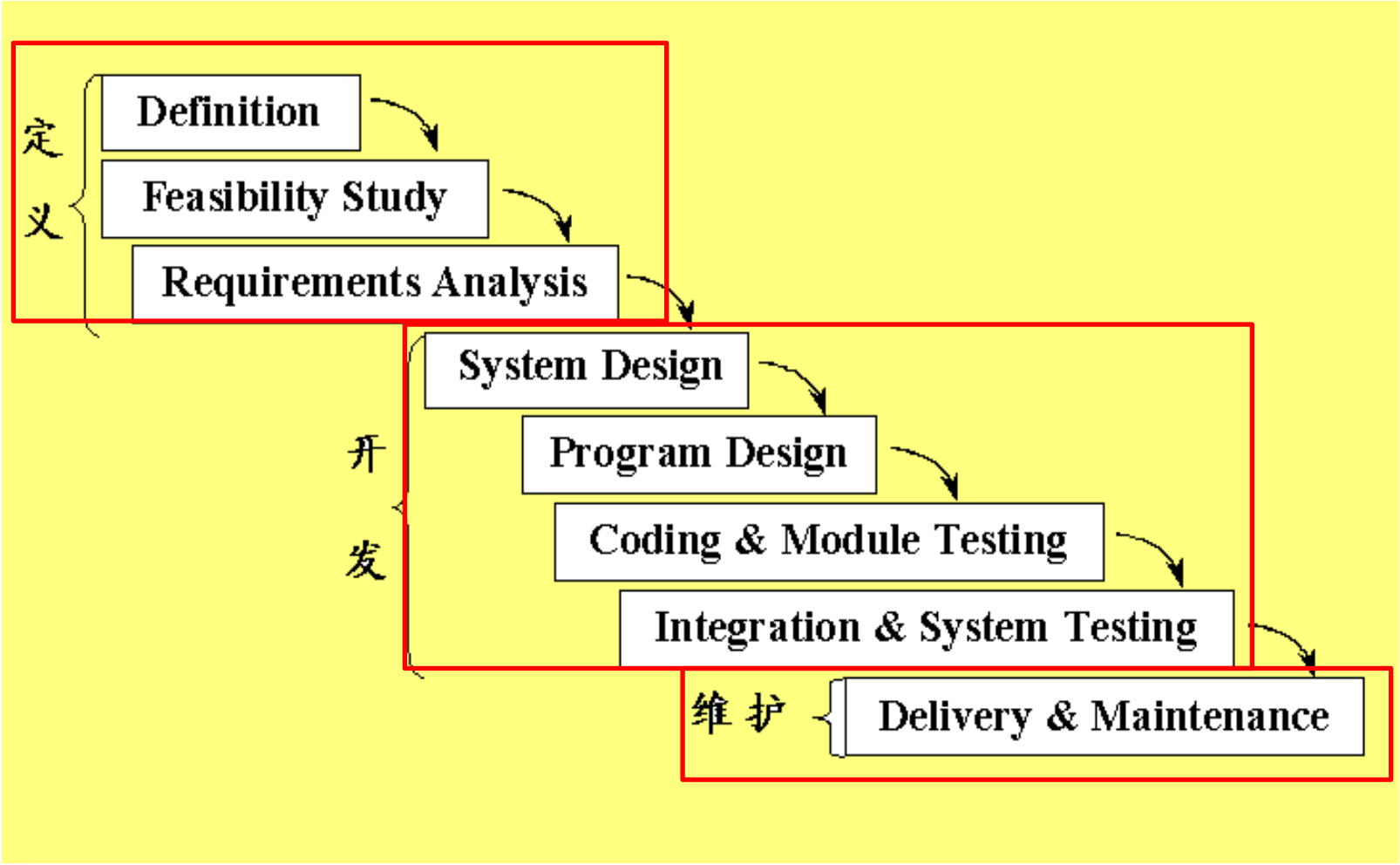
A process defines **Who** is doing **What**, **When**, and **How**, in order to reach a certain goal.

SQA：软件质量保证

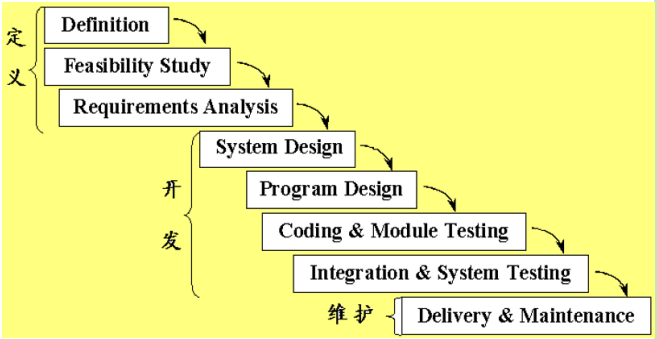
1.4 软件过程模型

（1）瀑布模型（Waterfall Model ）

传统瀑布模型



1.4 软件过程模型



传统瀑布模型的特点

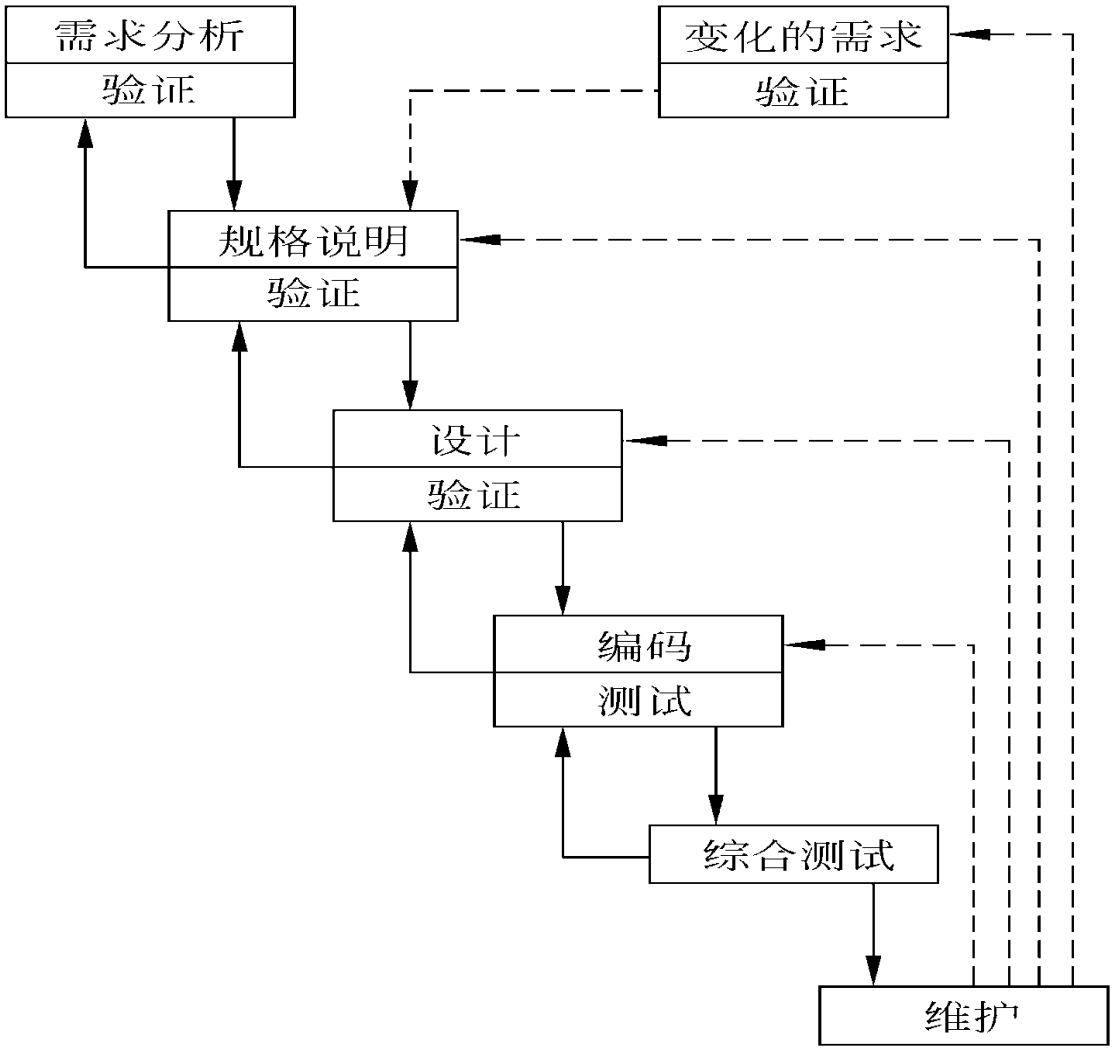
- 提供了软件过程模型的基本框架（模板）。
- 强调了每一阶段活动的严格顺序。
- 质量保证观点：以经过评审确认了的阶段工作产品（文档）驱动下一阶段的工作，便于管理。
- 是一种整体开发模型，程序的物理实现集中在开发阶段的后期，用户在最后才能看到自己的产品。

传统瀑布模型存在什么问题？

传统的瀑布模型过于理想化。事实上，人在工作过程中不可能不犯错误。在设计阶段可能发生规格说明文档中的错误。而设计上的缺陷或错误可能在实现过程中显现出来。在综合测试阶段将发现需求分析、设计或编码阶段的许多错误。

1.4 软件过程模型

实际的瀑布模型



1.4 软件过程模型

瀑布模型的特点

- 瀑布模型适合于用户需求明确、完整、无重大变化的软件项目开发。
- 瀑布模型的成功在很大程度上是由于它基本上是一种文档驱动的模型。
- “瀑布模型是由文档驱动的”这个事实也是它的一个主要缺点。

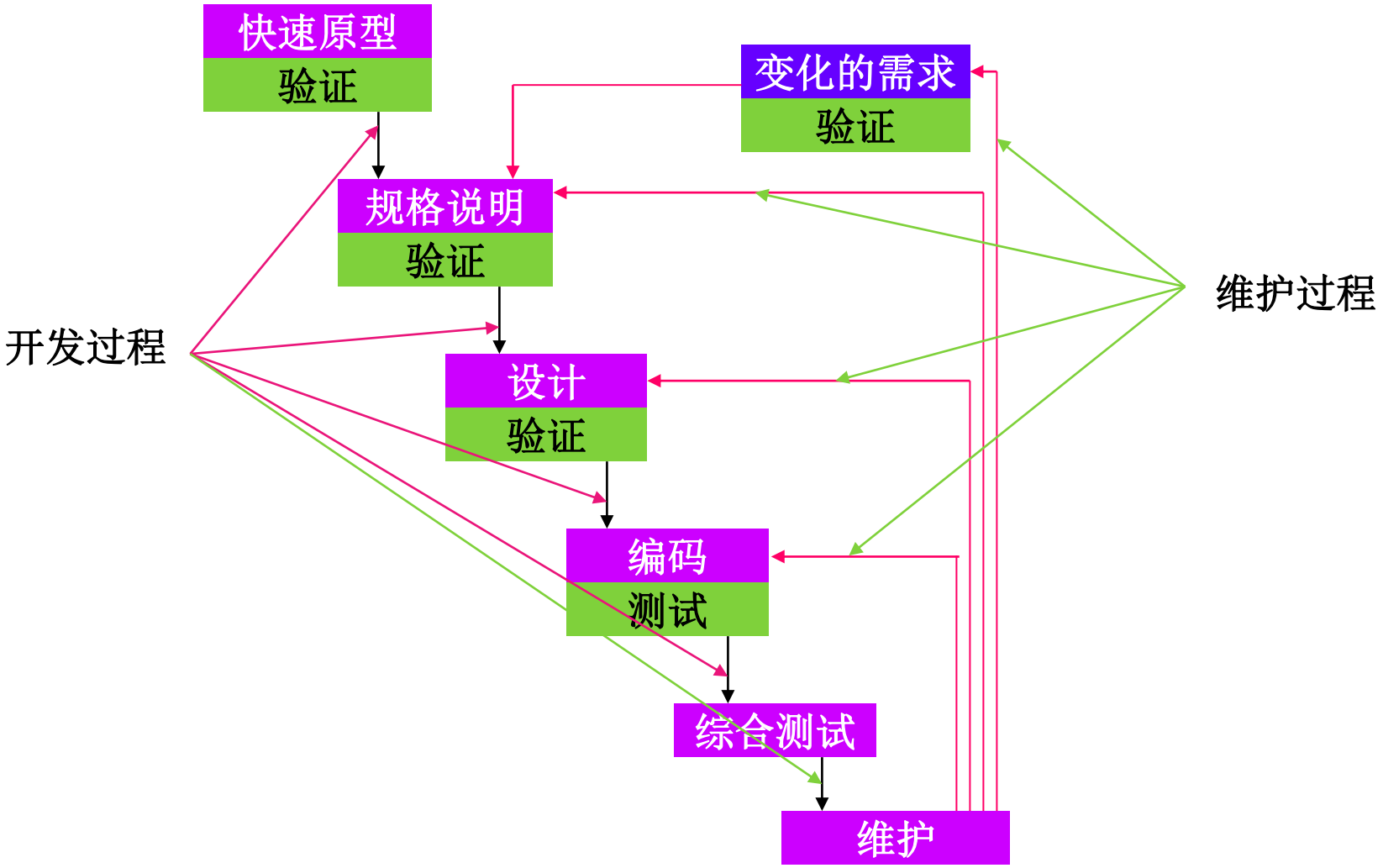
实际项目很少按照该模型给出的顺序进行；
用户常常难以清楚地给出所有需求；
用户必须有耐心，等到系统开发完成。

1.4 软件过程模型

（2）原型模型—快速原型模型（Rapid Prototype Model）

- 在用户不能给出完整、准确的需求说明，或者开发者不能确定算法的有效性、操作系统的适应性或人机交互的形式等许多情况下，可以根据用户的一组基本需求，快速建造一个原型（可运行的软件）；
- 然后进行评估，进一步精化、调整原型，使其满足用户的要求，也使开发者对将要做的事情有更好的理解。

1.4 软件过程模型



1.4 软件过程模型

1、原型模型存在的问题

⑴ 为了使原型尽快的工作，没有考虑软件的总体质量和长期的可维护性。

⑵ 为了演示，可能采用不合适的操作系统、编程语言、效率低的算法，这些不理想的选择成了系统的组成部分。

⑶ 开发过程不便于管理。

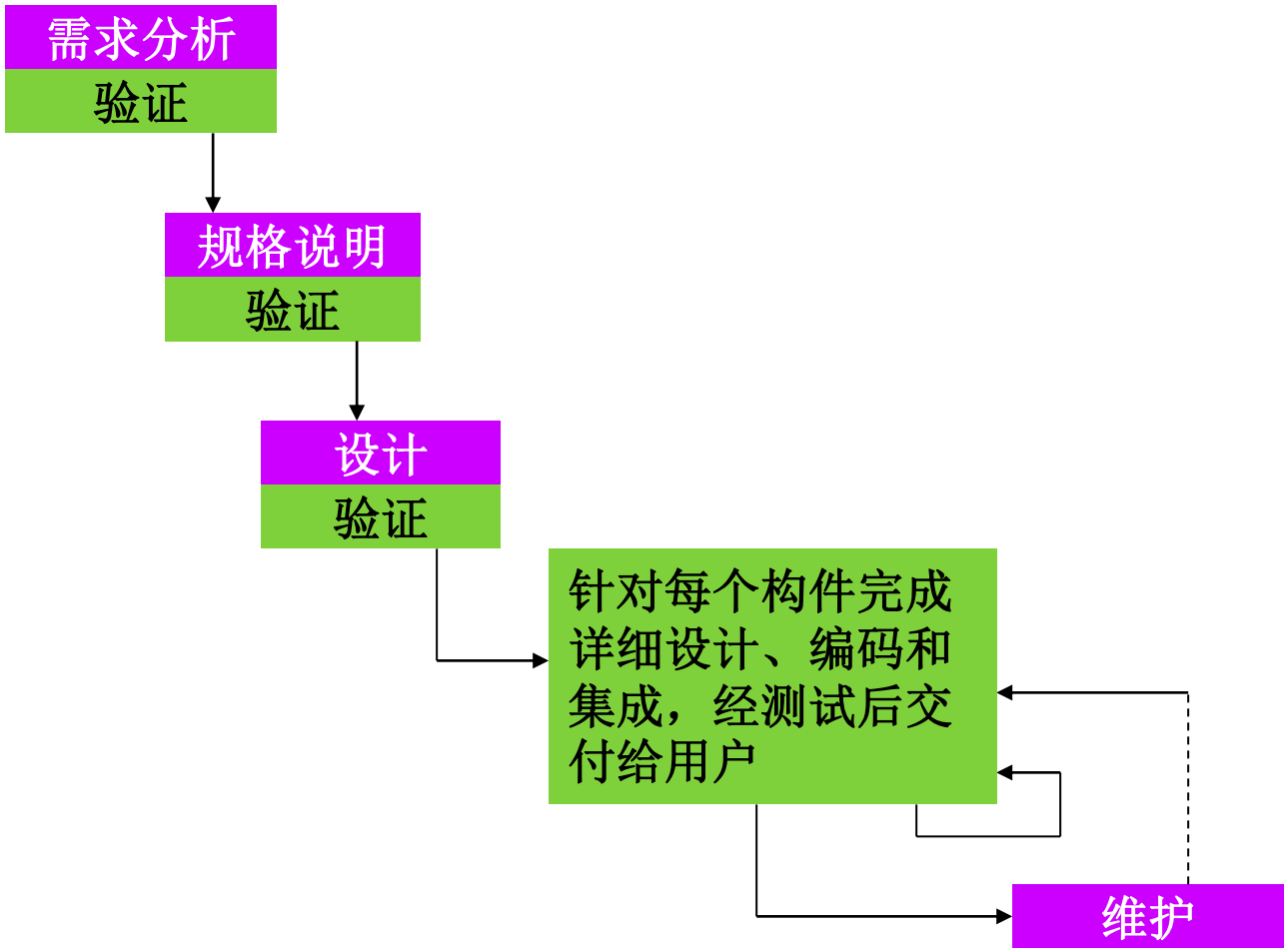
2、有效的使用原型模式

建造原型仅是为了定义需求，之后就被抛弃（或被部分抛弃），实际的软件在充分考虑了质量和可维护性之后才被开发。

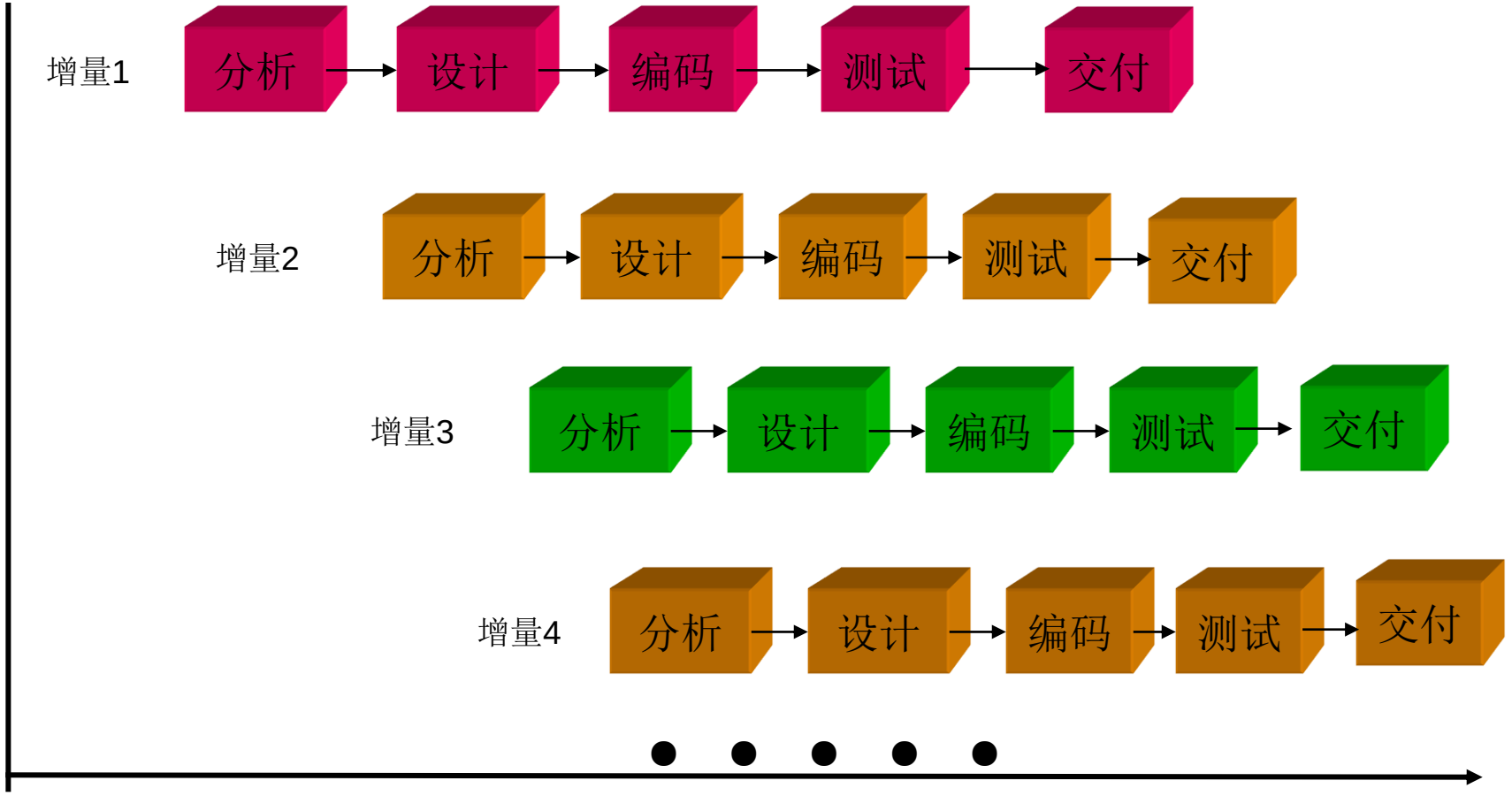
1.4 软件过程模型

（3）增量模型 （Incremental Model）

是一种渐进地开发逐步完善的软件版本的模型。



1.4 软件过程模型



增量模型

特点：

- 反复的应用瀑布模型的基本成分和原型模型的迭代特征，每一个过程产生一个“增量”的发布或提交，该增量均是一个可运行的产品。
- 早期的版本实现用户的基本需求，并提供给用户评估的平台。

1.4 软件过程模型

增量模型的优点

1）在较短时间内向用户提交可完成部分工作的产品，并分批、逐步地向用户提交产品。从第一个构件交付之日起，用户就能做一些有用的工作。

2）整个软件产品被分解成许多个增量构件，开发人员可以一个构件一个构件地逐步开发。

3）逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

4）采用增量模型比采用瀑布模型和快速原型模型需要更精心的设计，但在设计阶段多付出的劳动将在维护阶段获得回报。

1.4 软件过程模型

增量模型的困难

1）在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。

2）必须把软件的体系结构设计得便于按这种方式进行扩充，向现有产品中加入新构件的过程必须简单、方便，软件体系结构必须是开放的。

3）开发人员既要把软件系统看作整体，又要看成可独立的构件，相互矛盾。

4）多个构件并行开发，具有无法集成的风险。

1.4 软件过程模型

（4）螺旋模型 (Spiral Model)

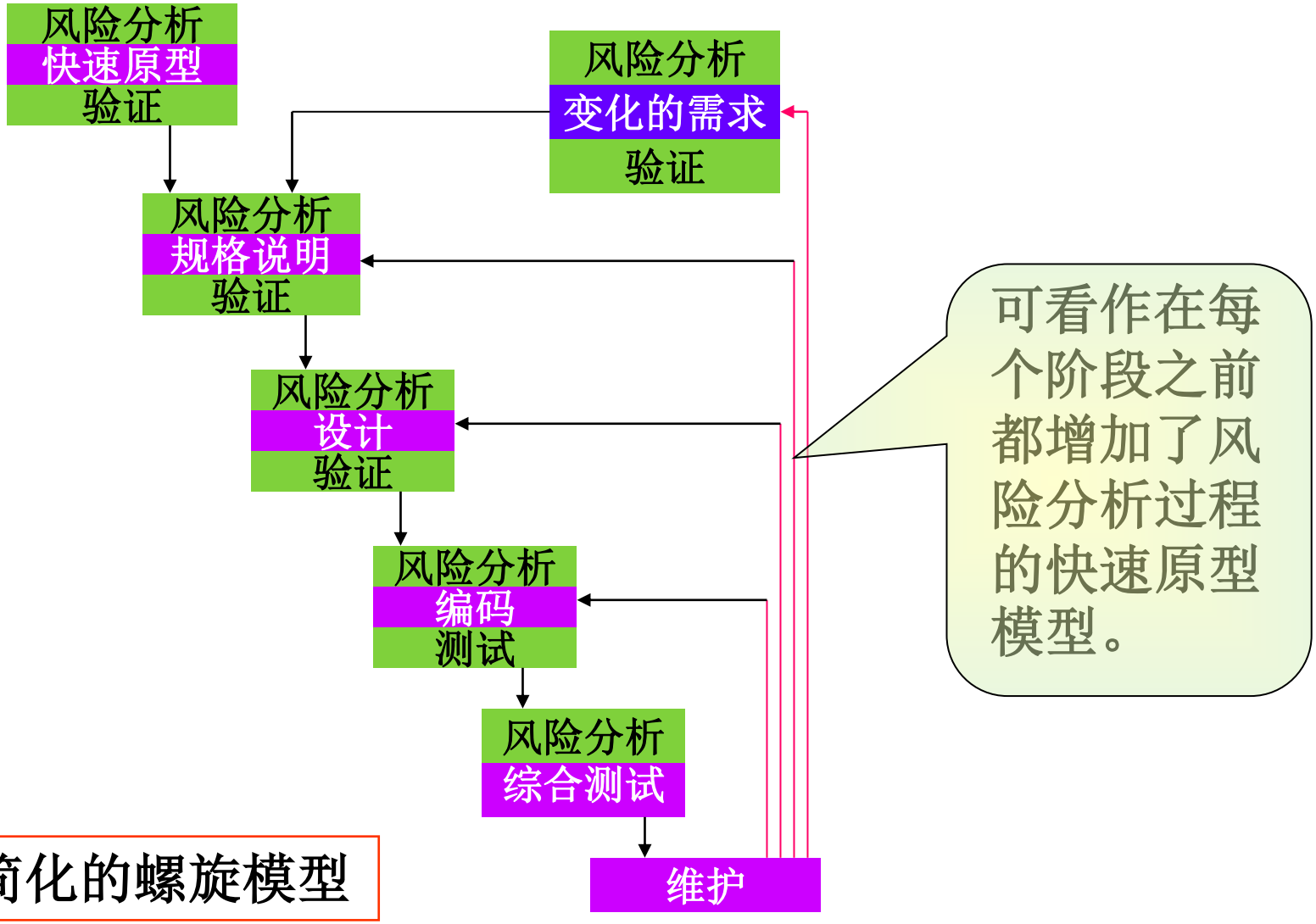
1） 软件风险是任何软件开发项目中都普遍存在的实际问题，项目越大，软件越复杂，该项目所冒的风险也越大。

2）对于复杂的大型软件，开发一个原型往往达不到要求。螺旋模型将瀑布模型和增量模型结合起来，加入了风险分析。

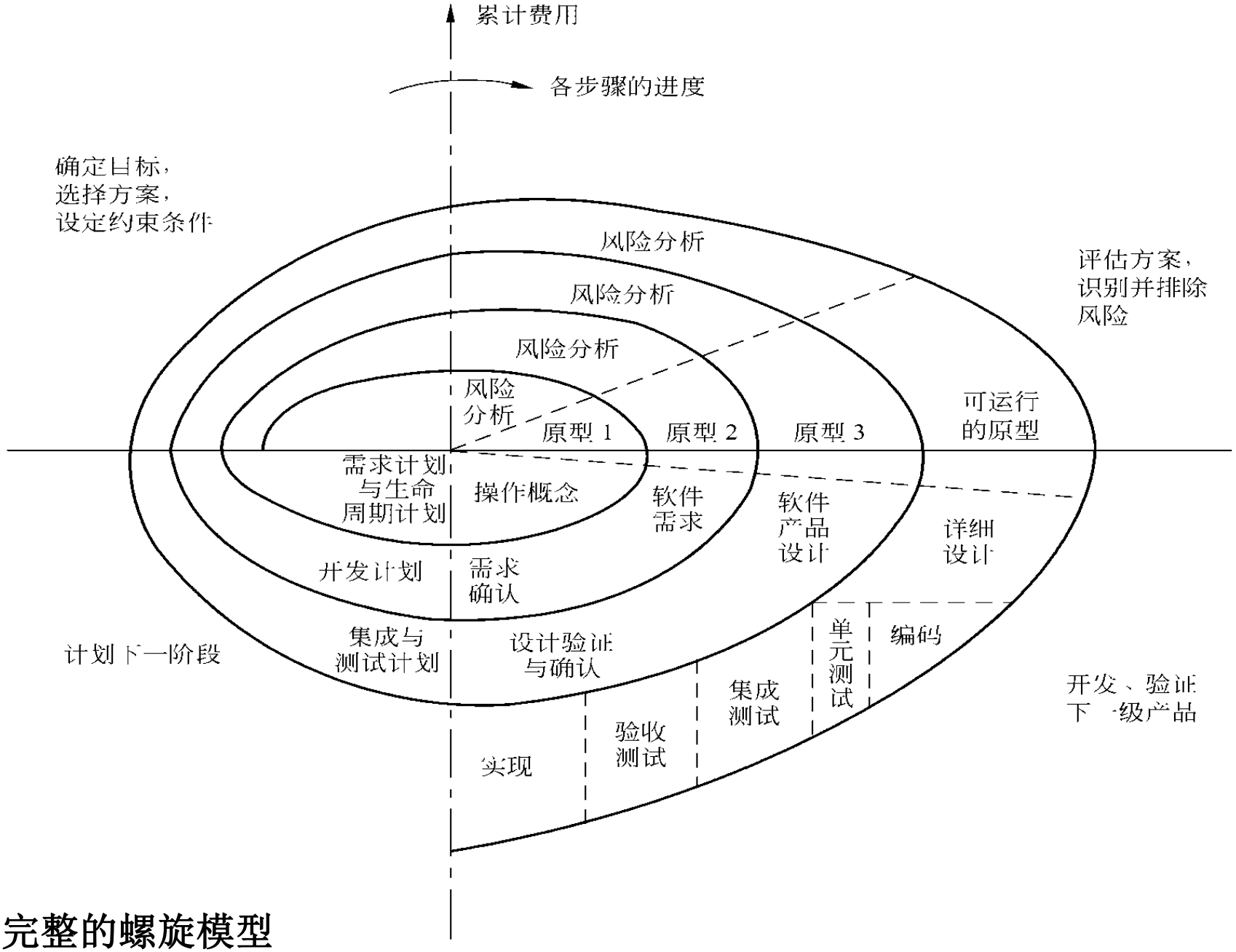
3）在该模型中，软件开发是一系列的增量发布，早期的迭代中，发布的增量可能是一个纸上的模型或原型，在以后的迭代中，逐步产生系统更加完善的版本。

4）螺旋模型的基本思想是降低风险。

1.4 软件过程模型



简化的螺旋模型


累计费用
各步骤的进度
确定目标，
选择方案，
设定约束条件
风险分析
风险分析
风险分析
风险
分析
评估方案，
识别并排除
风险
原型 1
原型 2
原型 3
可运行
的原型
需求计划
与生命
周期计划
操作概念
软件
需求
软件
产品
设计
详细
设计
并发计划
需求
确认
计划下一阶段
集成与
测试计划
设计验证
与确认
单元测试
编码
集成
测试
开发、验证
下一级产品
验收
测试
实现

完整的螺旋模型

1.4 软件过程模型

螺旋模型的优点

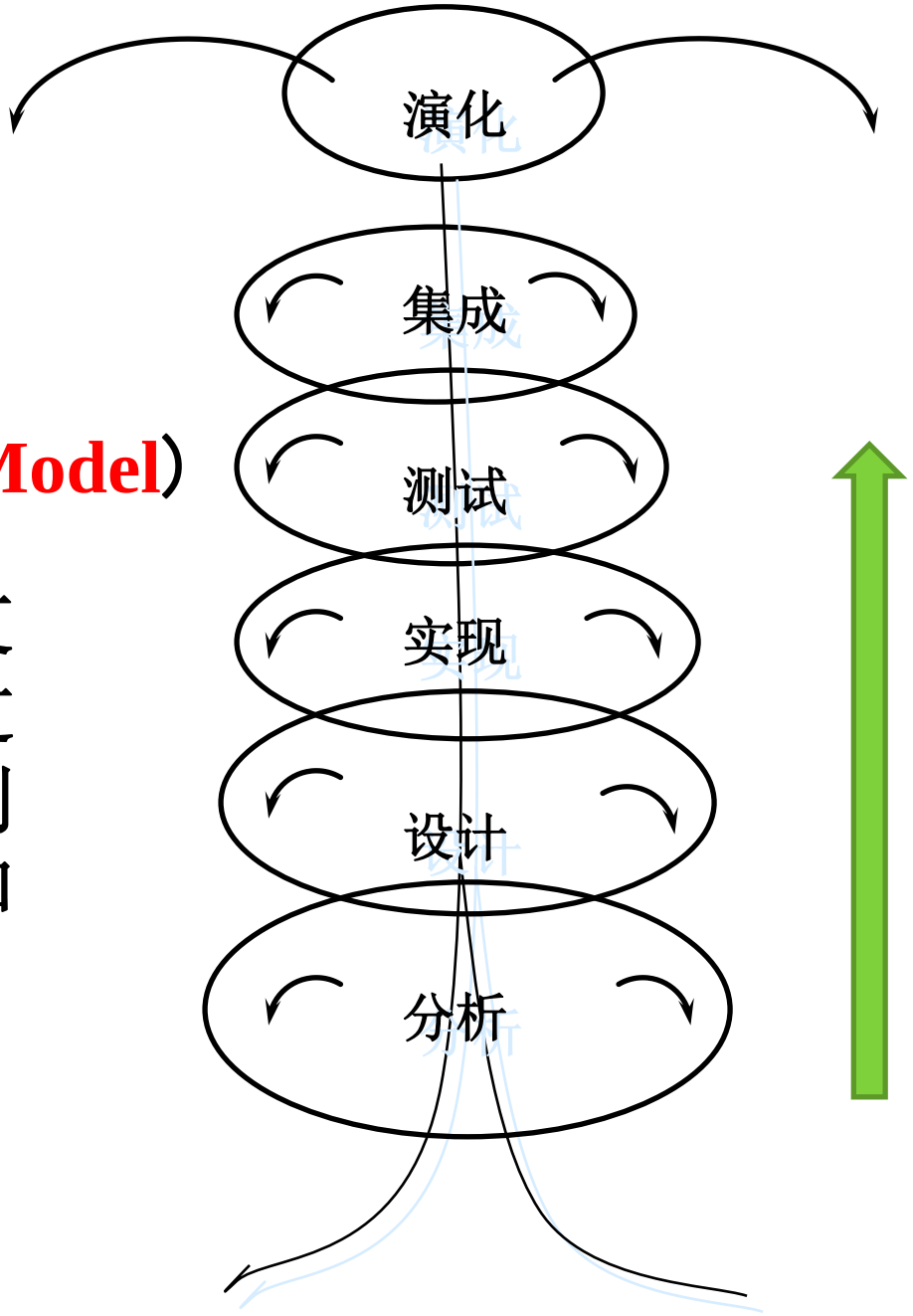
- 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标
- 减少了过多测试或测试不足
- 风险驱动：需要相当丰富的风险评估经验和专门知识，否则风险更大
- 主要适用于内部开发的大规模软件项目，随着过程的进展演化，开发者和用户能够更好的识别和对待每一个演化级别上的风险
- 随着迭代次数的增加，工作量加大，软件开发成本增加

1.4 软件过程模型

（5）面向对象模型

➢喷泉模型(Fountain Model)

特点：主要用于支持面向对象开发过程，体现了软件创建所固有的迭代和无间隙的特征



1.4 软件过程模型

➢可重用部件组装模型 (构件集成模型)
(Component Integration Model)

1） 构件（components)也称为组件，是一段实现一系列有确定接口的程序体，具有自己的功能和逻辑，能同其他构件集成起来协调工作。

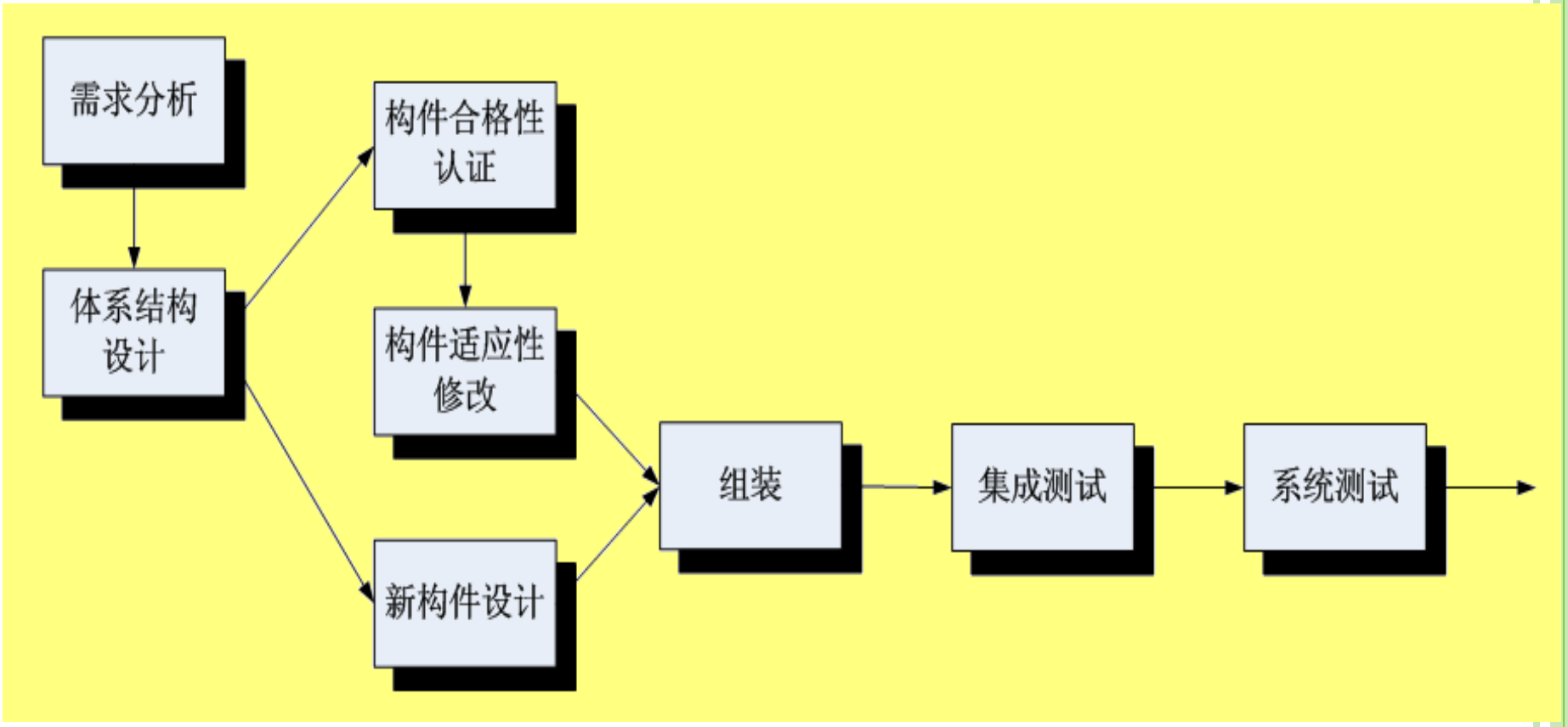
2）支持软件重用（Reusability），对缩短软件开发周期、降低项目成本有重要的现实意义。

3）建造符合某应用领域体系结构标准的构件，可以用来搭建分布式的、跨越不同操作平台(集成化软件开发环境ISEE)的软件，扩展了软件的应用前景，促进了软件标准化、商品化的发展。

3） “基于构件的软件工程”(CBSE)。

1.4 软件过程模型

构件集成模型如下图所示：

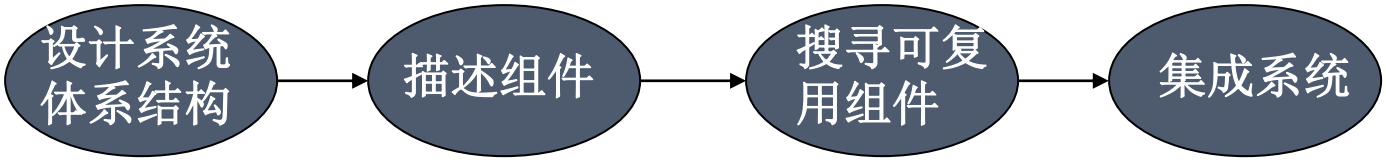


- 软件体系结构被建立后，必须用构件去充实，构件可从复用库中获得，或者根据专门需要而开发。
- 整个过程可以演化地进行，面向对象方法给予技术上的支持。

1.4 软件过程模型

Sommerville提出基于组件开发有两种思路：

- 高层设计：对设计中的组件给出描述，以便找出可复用的组件，这些组件可在体系结构层次上加入或更详细的设计层次上加入。
- 根据需求搜寻可复用组件，再将设计建立在获得的组件基础上。
- 这两种思路可结合起来。



先完成架构设计的复用

复用驱动设计

1.5 小结

- **软件 =程序+数据+文档**
- **软件危机: 原因,现象,办法 (软件工程学)**
- **软件工程(学):开发、运行和维护软件的系统方法**
- **软件工程3个要素：方法、工具和过程。**
- **软件生命周期：定义,开发,运行维护**
- **软件过程：瀑布模型+统一软件开发过程 RUP(Rational Unified Process)**