软件工程：第一章 软件工程概述

导学目标

1. 了解软件危机的起源、产生原因、主要表现特征
2. 了解软件工程的概念、基本原理、分类等
3. 掌握生命周期的各个阶段以及生命周期模型
4. 掌握统一过程的各个阶段和各个工作流之间的关系
5. 掌握软件过程的改进策略
6. 掌握敏捷开发

第一节 软件危机

1.1.1 软件危机的缘起

计算机发展的第一阶段（20世纪60年代以前）

- 软件生产个体化
  - 软件规模小
  - 编写和使用者往往是同一个人
- 软件设计是人们头脑中一个隐含的过程
  - 只有程序清单，无其他文档材料

计算机发展的第二个阶段（20世纪60年代中期-70年代中期）

- 出现了软件作坊
  - 有专门的软件开发组织，但是仍然沿用了早期个体化的软件开发方法
- 暴露问题
  - 用户有新的需求，就必须要修改程序
  - 运行环境变化，就需要修改程序
  - 程序个性化特点，使软件难以维护，成本比较大
- 软件工程诞生
  - 1968年，北大西洋公约组织计算机科学家讨论软件危机，正式提出“软件工程”

1.1.2 软件危机揭秘

定义

- 在计算机**软件开发**和**维护过程**中遇到的一系列严重问题
- 回答两个问题
  - 怎么开发软件（用户需求变了怎么办？）
  - 开发完了怎么维护

主要表现

- 开发成本和进度估计不准确
  - 报价过高

    - 延迟交付
- 用户对已交互软件不满意
    - 没有充分沟通就开始匆忙写程序
- 软件产品质量靠不住
- 软件可维护性差
    - 程序中的错误很难改正，不能适应用户新增的需要
- 软件没有适当的文档资料
    - 开发和维护起来都比较困难

产生原因

- 软件自身的特点
    - 软件缺乏可见性
    - 软件规模庞大，结构很复杂
- 软件开发与维护方法不正确

解决办法

- 技术措施
    - 改变认识（软件≠程序，软件=程序+数据+文档）
    - 开发和使用更好的工具
- 管理措施
    - 协调好人员配置，管理严密，共同完成项目

第二节 软件工程

1.2.1 软件工程定义

1993年IEEE给出了更全面更具体的定义

- 把系统化、规范化、可度量的途径应用于软件开发、运行和维护**过程中**；研究其实现途径

1.2.2 软件工程基本原理

7条基本原理

- 用分阶段的周期计划严格管理
    - 把软件的生命周期划分为若干个阶段，相应的制定出切实可行的计划，并严格按照计划对软件的开发和维护工作进行管理
- 坚持进行阶段评审
    - 在每个阶段都进行严格的评审，尽早发现错误
- 实现严格的产品控制
- 采用现代程序设计技术
- 结果应能清楚的审查
- 开发人员应该小而精
- 承认不断改进软件工程实践的重要性

1.2.3 软件工程方法学

分类

- 传统方法学
- 面向对象方法学

第三节 软件生命周期

定义

- 从软件的产生、发展、成熟到衰亡的全过程

组成

- 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
    - 编码和单元测试
    - 综合测试
- 软件维护
    - 软件维护

1.3.1 软件生命周期的8个阶段

国标《计算机软件开发规范》分为8个阶段

- 可行性研究和计划（=问题定义+可行性研究）
    - 关键问题
        1. 问题是什么?
        2. 有没有可行的解决办法?
        3. 大致的计划
    - 产物
        1. 问题定义报告（问题的性质、目标、规模）
        2. 可行性研究报告（经济、技术、社会可行性）
        3. 项目的开发计划（粗略）
- 需求分析
    - 关键问题
        1. 明确目标系统必须具备哪些功能
        2. 可行性研究的需求分析是粗略的，需求分析是完整、准确、具体、清晰
    - 产物
        1. 需求规格说明书（用正式的文档准确的记录对目标系统的需求）

- 总体设计
  - 关键任务
    1. 怎样实现目标系统
    2. 设计多种方案，推荐最佳方案，设计体系结构
  - 产物
    1. 总体设计说明书（记录总体设计的结果）
- 详细设计（针对每一个模块）
  - 关键任务
    1. 该怎样具体实现目标系统
  - 产物
    1. 详细设计说明书（设计每个模块的算法和数据结构）
- 实现
  - 关键任务
    1. 选择合适的语言和工具翻译详细设计结果，测试模块
  - 产物
    1. 实现阶段文档（程序清单、单元测试报告）
- 集成测试
  - 关键任务
    1. 将通过单元测试的模块组装起来进行测试
    2. 通过各种类型的测试使软件达到预定的要求
  - 产物
    1. 测试报告（测试计划、详细测试方案、实际测试结果）
- 确认测试
  - 关键任务
    1. 由用户根据需求规则说明书进行测试
  - 产物
    1. 测试报告（测试计划、测试方案、测试结果）
- 使用与维护
  - 关键任务
    1. 通过各种必要的活动使系统持久的满足用户的需要
  - 四类维护类型
    1. 改正性维护
    2. 适应性维护
    3. 完善性维护
    4. 预防性维护

第四节 软件过程

1.4.1 软件过程的定义

软件过程是一个为创造高品质软件所需要完成的活动、动作和任务的框架

- 白话解释：软件过程描述为了开发出用户所需要的的软件，什么人在什么时间做了什么事情以及怎么去做的过程。

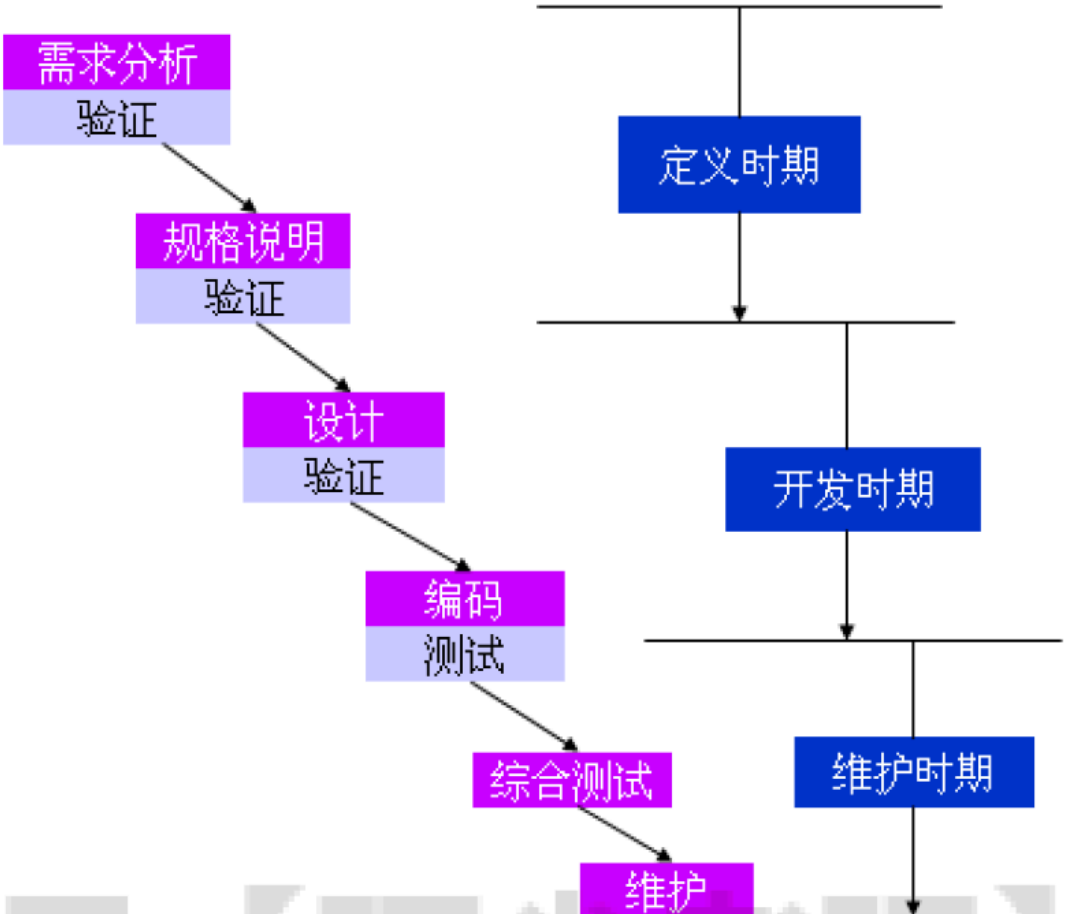
为了能够描述软件过程，通常使用生命周期模型，也叫做过程模型

实际从事软件开发工作时，软件规模、类型、开发环境以及技术方法等会影响到各阶段的划分和执行顺序，形成不同的软件生命周期模型，即过程模型。

1.4.2 软件过程模型——瀑布模型

瀑布模型是软件工程中使用最广泛的模型之一

传统的瀑布模型如图所示：

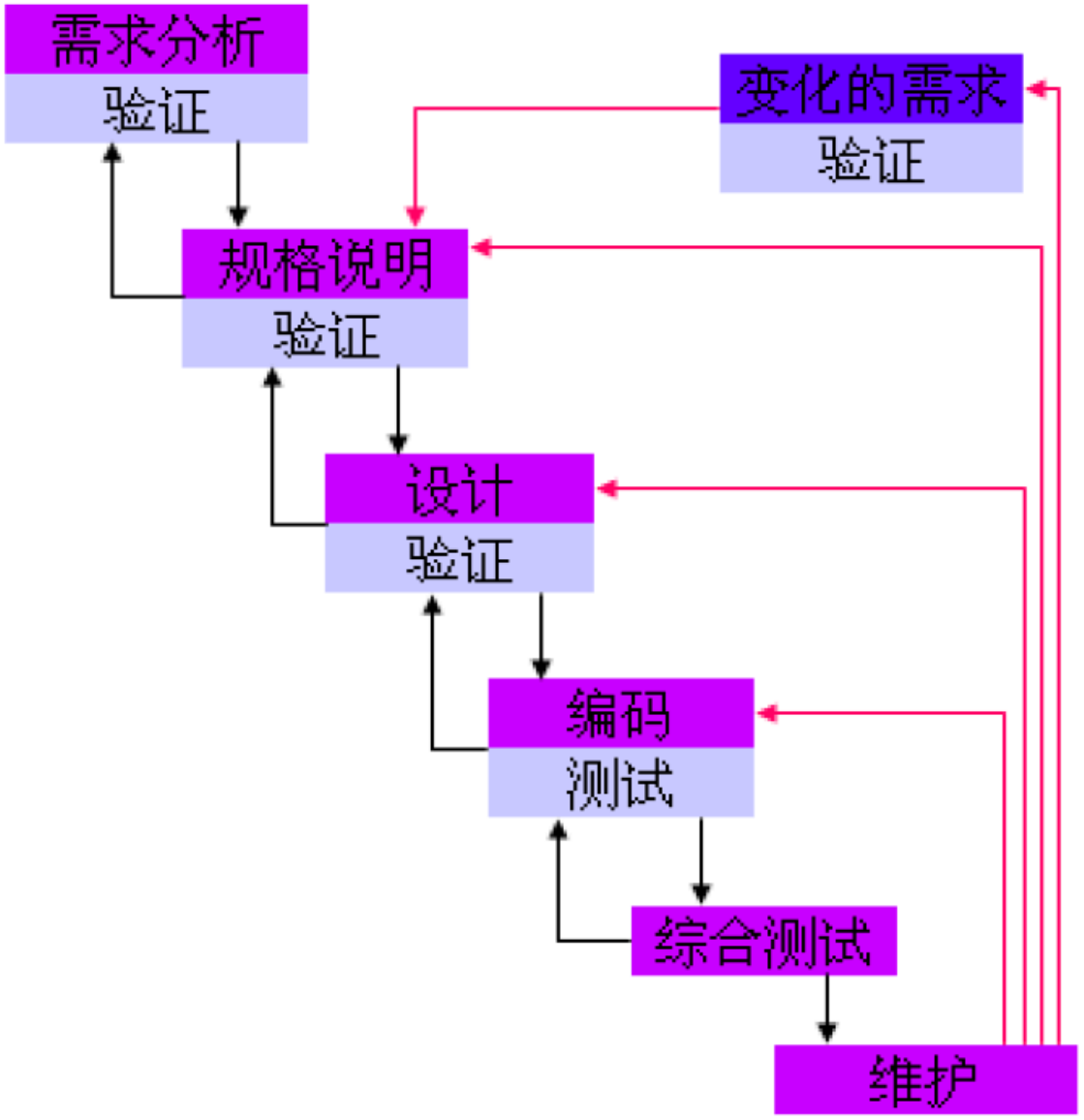


特点

- 阶段间具有**顺序性**和**依赖性**
  - 前一阶段结束之后，后一阶段才能开始
  - 前一阶段的输出文档就是后一阶段的输入文档
- **推迟实现**的观点
  - 瀑布模型在编码前设置了系统分析和系统设计各个阶段，尽可能推迟程序的物理实现
  - 实践表明，编码开始的越早，最终完成开发所需时间更长。这是因为前面工作做得不扎实，导致大量返工，甚至发生无法弥补的问题，带来灾难性后果

- 质量保证观点
  - 每个阶段都必须完成规定的文档
  - 每个阶段结束前都要对所完成的文档进行评审

传统瀑布模型过于理想化了，事实上人在工作的过程中不可能不犯错误，因此实际的瀑布模型都是带反馈环的。带反馈环的瀑布模型如图：



瀑布模型的优缺点

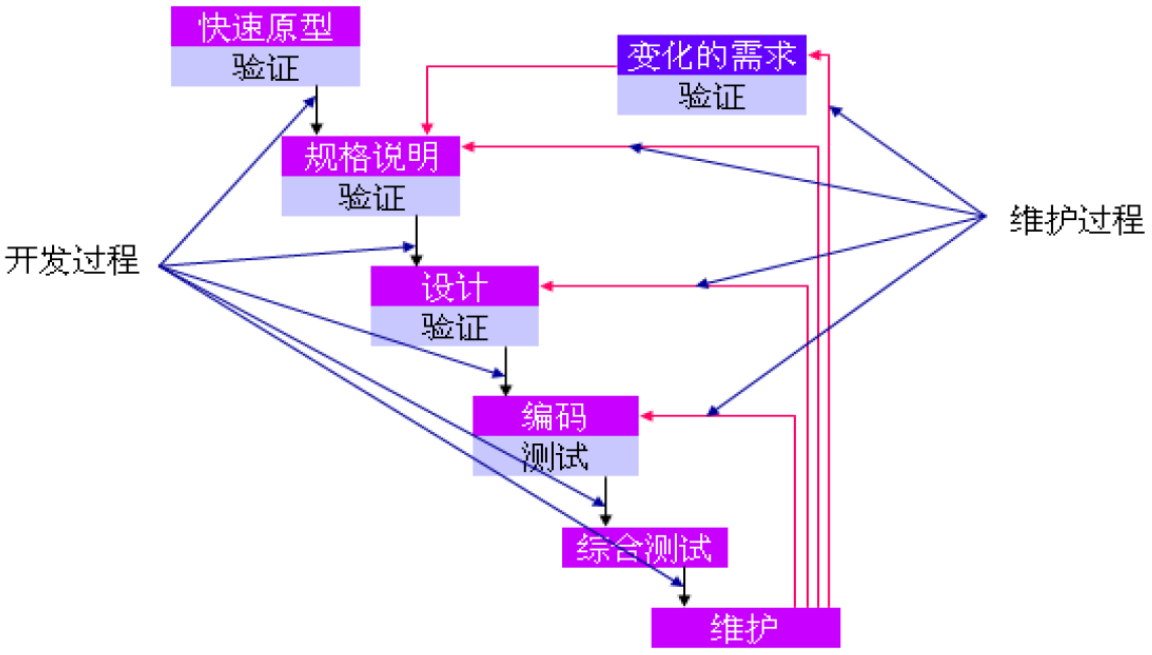
- 优点
  - 提高软件质量
  - 降低维护成本
- 缺点
  - 模型缺乏灵活性
  - 要求用户在不经过实践就提出完整准确的需求不切实际

1.4.3 软件过程模型——快速原型模型

定义

- 快速建立一个能反应用户主要需求的原型系统，根据用户的使用意见反复修改原型系统，开发出真正符合用户需求端的产品

快速原型模型如下



快速原型模型的优缺点

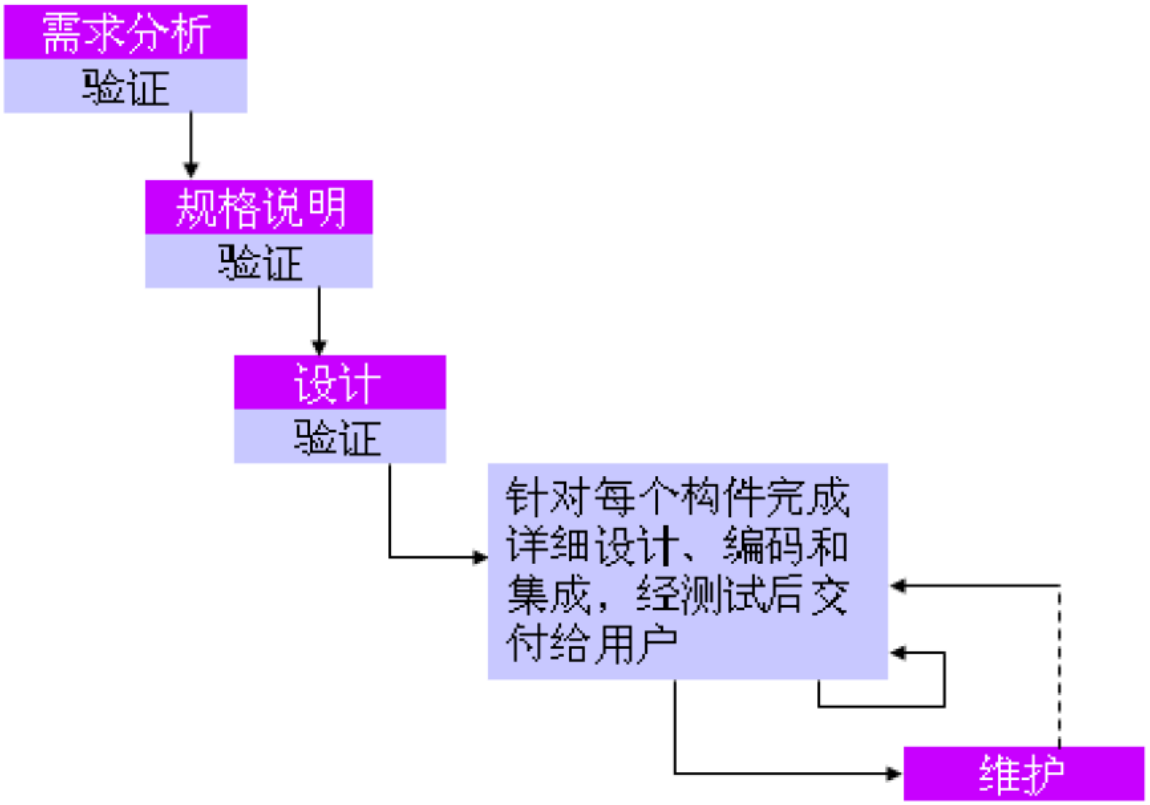
- 优点
  - 在需求确定上优于瀑布模型
  - 开发人员通过建立原型系统学会了很多东西
  - 有些原型系统可以称为最终产品的一部分
- 缺点
  - 快速建立的原型系统并且经过连续的修改，可能导致产品质量低下，原型系统的内部结构不好

1.4.4 软件过程模型——增量模型

定义

- 把软件产品作为一系列的增量构建来设计、编码、集成和测试

增量模型如图所示



与瀑布模型和快速原型模型的区别

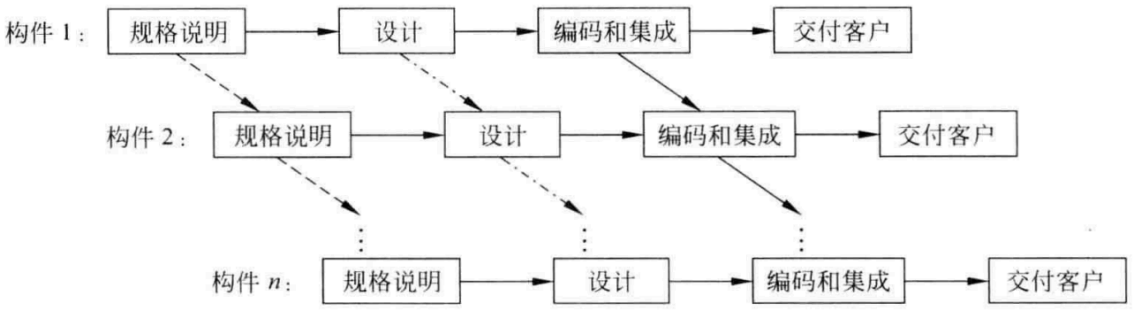
- 瀑布模型和快速原型模型就是一次把一个满足所有需求的产品提交给用户
- 增量模型分批向用户提交产品

增量模型的优缺点

- 优点
  - 较短时间内向用户提交可完成部分工作产品
  - 用户有充裕的时间学习和适应新产品
  - 软件结构是开放的，方便向现有产品添加新构件
- 缺点
  - 向软件中添加新构建，不影响原产品是比较困难的

前面的增量模型表明，必须在开始实现各个构件之前就全部完成需求分析、规格说明和概要设计阶段，相对来说风险较小

风险更大的增量模型一旦确定用户需求后，并行的构建不同的构件，其模型如下：

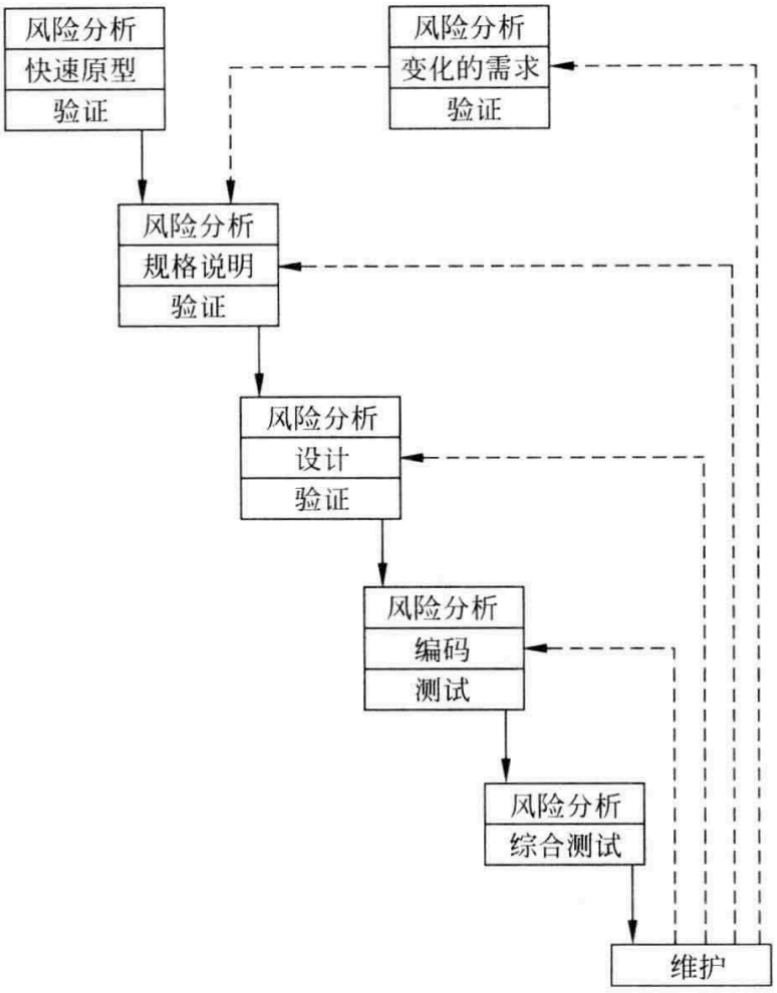


1.4.5 软件过程模型——螺旋模型

定义

- 通过使用原型或者其他方法方法最小化风险。可以简单的将这个生命周期模型看作是每个阶段之前带有分析的原型模型

简化螺旋生命周期模型如下

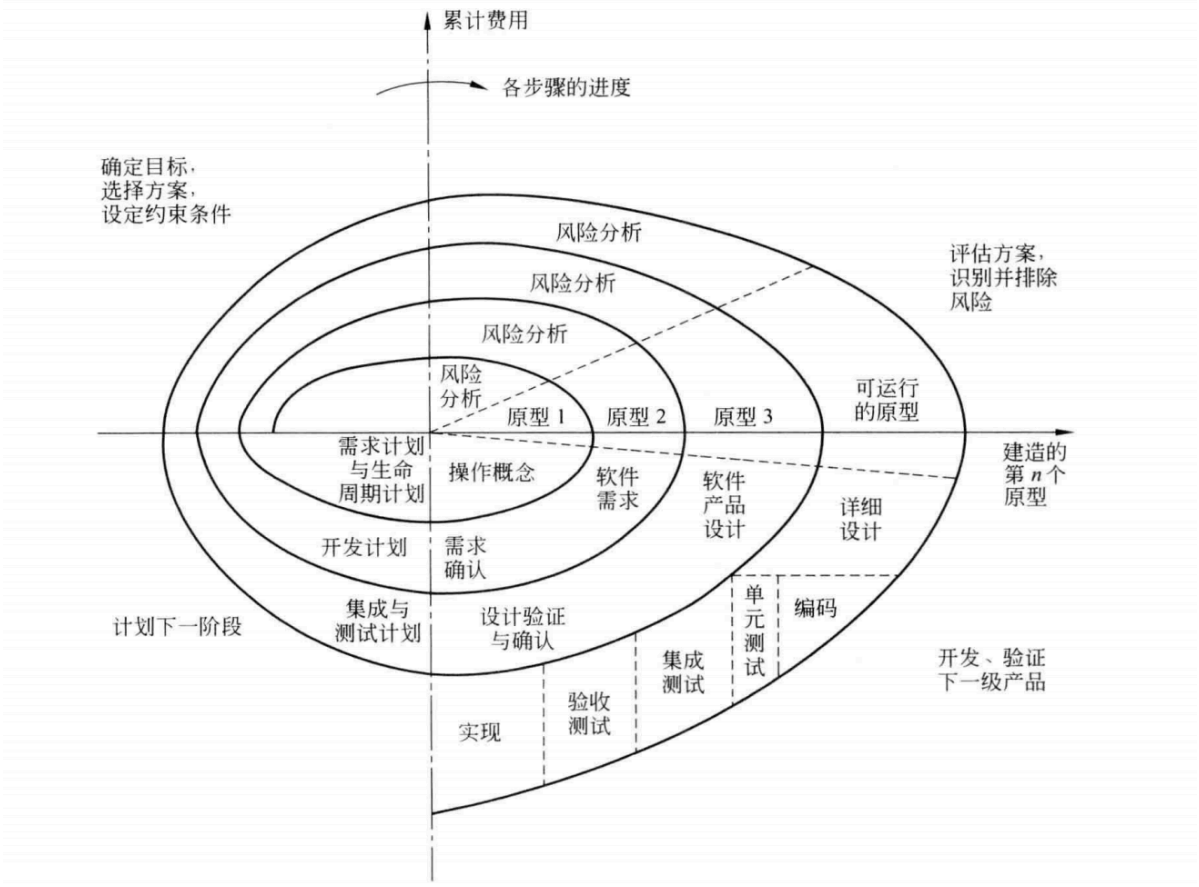


笛卡尔坐标四象限表达四方面活动

- 制定计划：确定方案、选择目标、确定约束条件
- 风险分析：评估方案、确认风险、减小风险
- 实施工程：开发、验证下一级产品
- 客户评估：评价开发工作、计划下一阶段

沿螺线自内向外每旋转一圈开发出更完善新版本

完整的螺旋生命周期模型如图



螺旋模型的优缺点

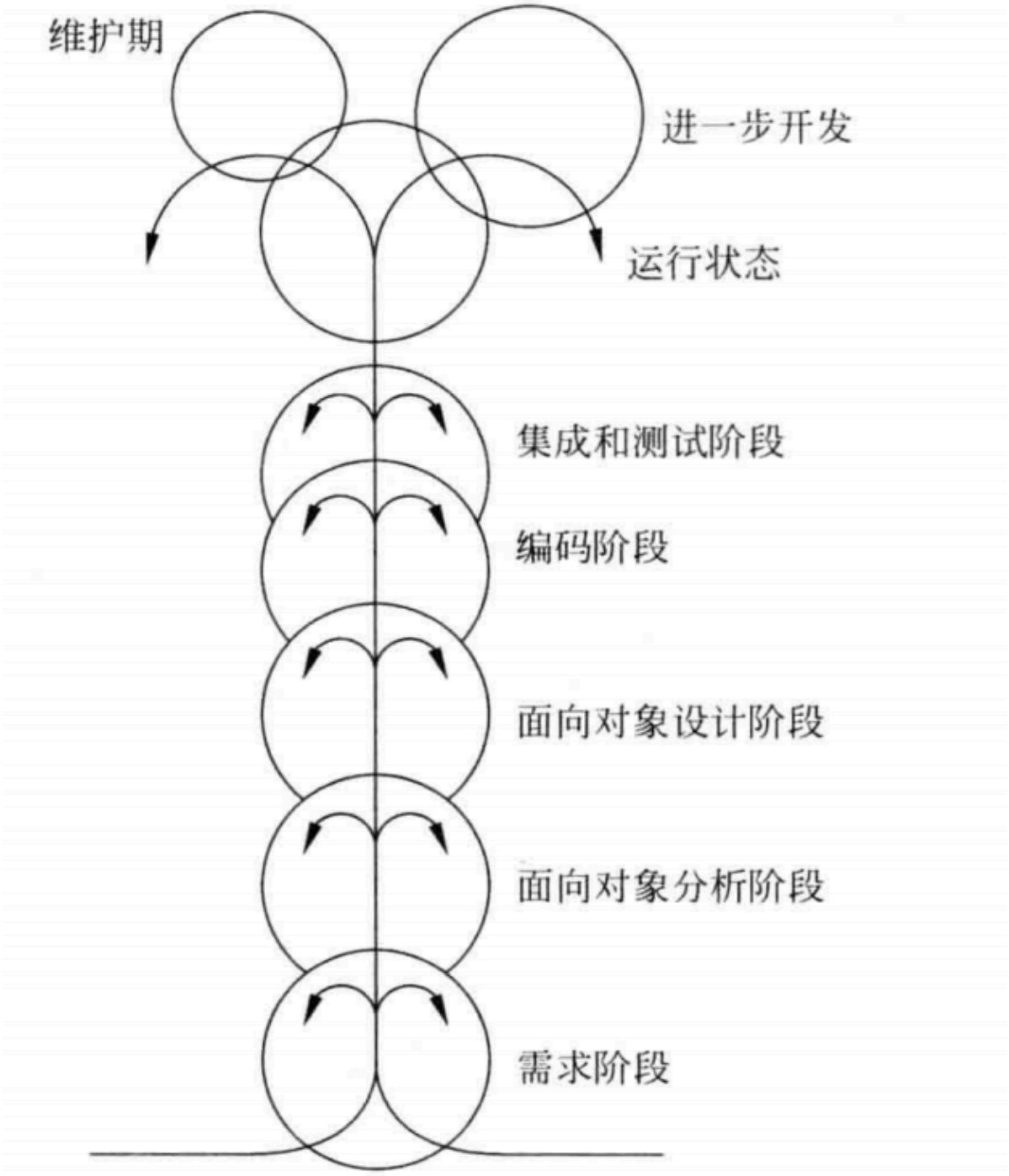
- 优点
  - 适用于内部开发的大规模项目
- 缺点
  - 需要风险评估的经验

1.4.6 软件过程模型——喷泉模型

面向对象生命周期模型，体现迭代和无缝特性

- 迭代
  - 求精，系统某部分常被重复工作多次，相关功能在每次迭代中逐渐加入演进系统
- 无缝
  - 分析、设计、编码各阶段间不存在明显边界

喷泉模型如图



1.4.7 软件过程模型——统一过程模型

> 前面几个都是一维，统一过程为二维

统一过程模型应视为一种自适应方法学。也就是说，要根据具体所开发的软件产品进行修改。

统一过程的五个核心工作流

- 需求流
  - 任务：准确确定客户的需求并从客户的角度找出存在的限制条件
- 分析流
  - 任务：分析和提取需求，清楚的指出产品要做什么
- 设计流

    - 任务：细化分析流制品，直至达到程序员可实现的形式
- 实现流
    - 任务：用选择的实现语言实现目标软件产品
- 测试流
    - 任务：对实现流产品进行测试

软件过程模型——统一过程模型

统一过程模型四个阶段

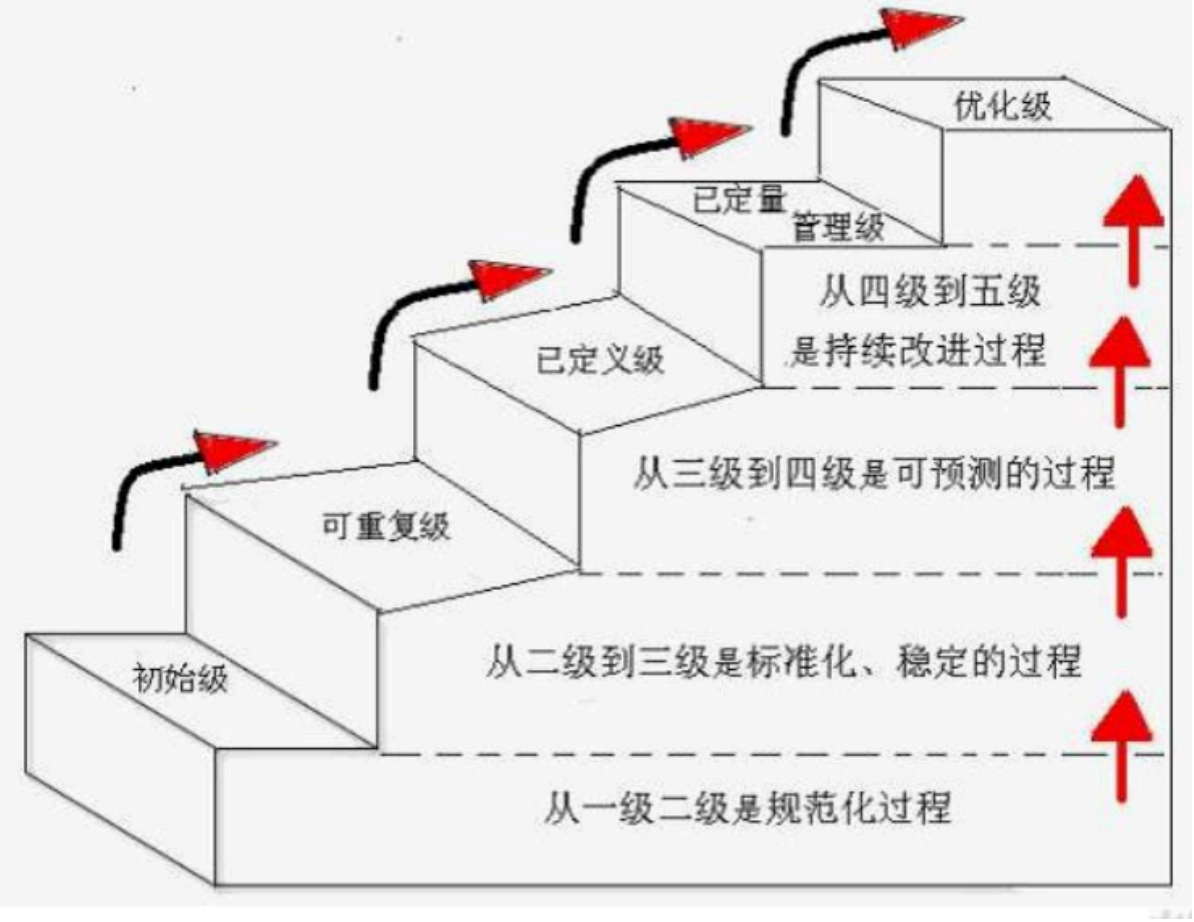
- 开始阶段：确定产品是否值得开发
- 细化阶段：针对目标，将目标细化成需求
- 构件阶段：产生软件可供测试的第一个版本
- 转换阶段：确保客户的需求得到满足

1.4.8 软件过程改进——SW-CMM模型

SW-CMM的五个成熟度级别

- 初始级
    - 最低级别，有效的软件过程管理方法本质上没有得到使用
- 可重复级
    - 使用了基本的软件项目管理措施
- 定义级
    - 有充分的软件生产文档
- 管理级
    - 为每个项目设计了质量目标和生产目标
- 最优级
    - 持续改进软件过程

SW-CMM模型如图



1.4.9 敏捷开发

什么是敏捷?

- Ivar Jacobson给出一个非常有用的论述：每个人都是敏捷的。敏捷**团队**是适应**变更**的灵活团队。变更就是软件开发本身，软件构建有变更、团队成员有变更、适应新技术会带来变更，各种变更都会对开发的软件产品以及项目本身造成影响。我们必须接受"支持变更"的思想，它应该根植于软件开发中的每一件事中，因为它是软件的心脏与灵魂。敏捷团队意识到软件是团队中所有人共同开发完成的，这些人的**个人技能**和合作能力是项目成功的关键所在。

Scrum敏捷过程模型

- Scrum得名于橄榄球比赛，是Jeff Sutherland和他的团队在20世纪90年代早期发展的一种敏捷过程模型，近年来，Schwaber和Beedle对其做了近一步的发展
- Scrum过程由"**需求、分析、设计、演化、交付**"等框架性活动组成。每一个框架活动过程中，发生于一个过程模式中的工作任务称为一个冲刺。冲刺中进行的工作适用于当前的问题，由Scrum团队规定并常常进行实时修改

Scrum过程流说明

- 待定项：一个能为用户提供商业价值的项目需求或者特性的优先级列表。待定项中可以随时加入新项，产品经理根据需要评估待定项并修改优先级
- 冲刺：由一些工作单元组成，这些工作单元是达到待定项中的需求所必须的，并且能在预定的时间段内（一般是30天）完成。冲刺过程中不允许有变更，因此冲刺给开发人员团队提供了短暂而稳定的工作环境
- Scrum例会：Scrum团队每天召开的短会，一般是15分钟，主要回答三个问题：
    - 上次例会后做了什么?
    - 遇到了什么困难?

    - 下次例会前计划做些什么？
- 演示：向客户交互软件增量，为客户演示所实现的功能并由客户对其进行评价。注意**演示不需要包含计划中的所有功能**，但是该时间段内的功能必须完成