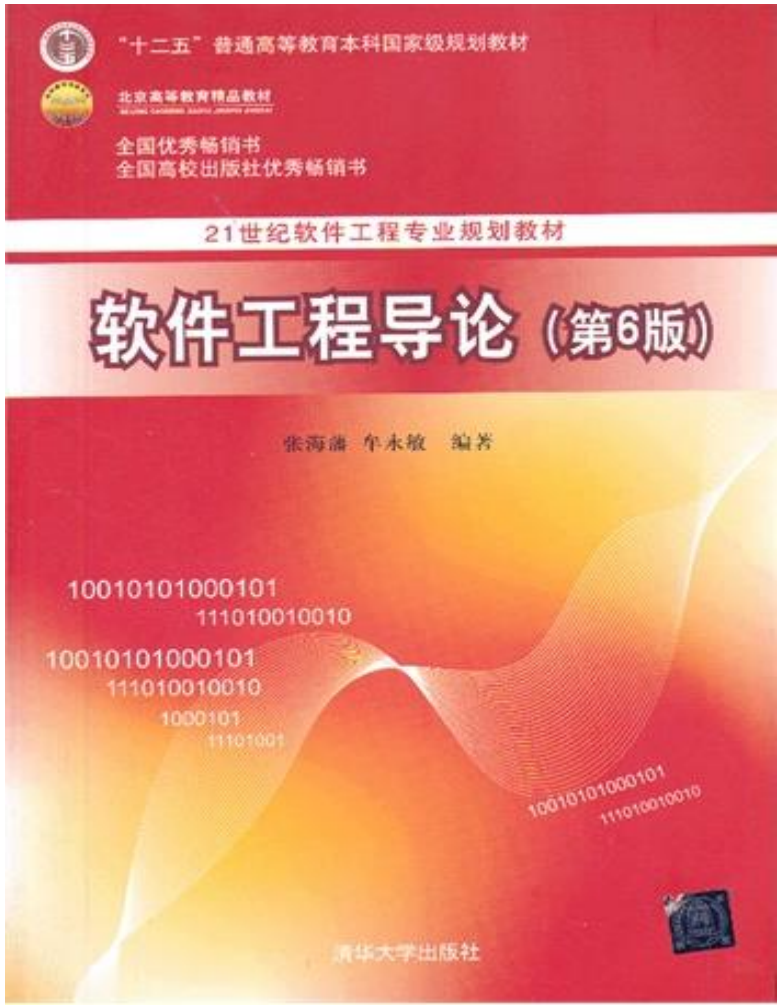
# 软件工程

# 教材

张海藩
软件工程导论
清华大学出版社



# 参考书目(References)

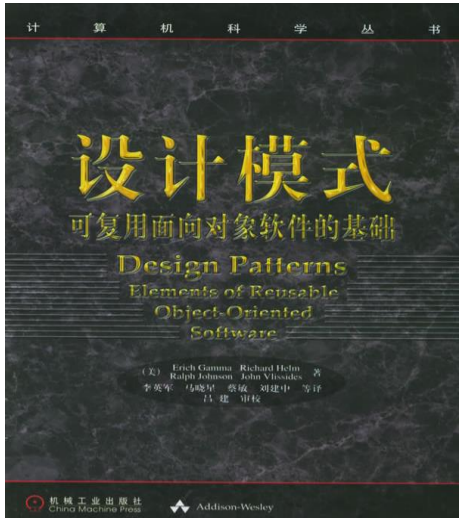
软件工程--实践者的研究方法

[美]Roger S.Pressman 译：郑人杰

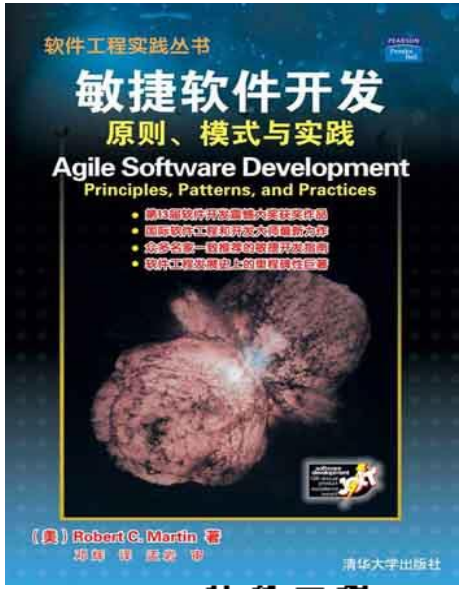
机械工业出版社

# 参考书目(References)

伽玛等著, 李英军等译
设计模式: 可复用面向对象软件的基础
机械工业出版社




Robert C. Martin, 邓辉 译
敏捷软件开发——原则、模式与实践
清华大学出版社



# 参考书目 (References)

## Software Engineering

A PRACTITIONER'S APPROACH

FIFTH EDITION

## 软件工程

实践者之路

*Roger S. Pressman, Ph.D.*

机械工业出版社 · McGraw-Hill Companies, Inc.

# 授课内容：

软件工程学概述
可行性研究
需求分析
总体设计
详细设计
实现
维护

面向对象方法学引论
面向对象分析
面向对象设计
面向对象实现
软件项目管理

# 第1章 软件工程学概述

1.1 软件危机

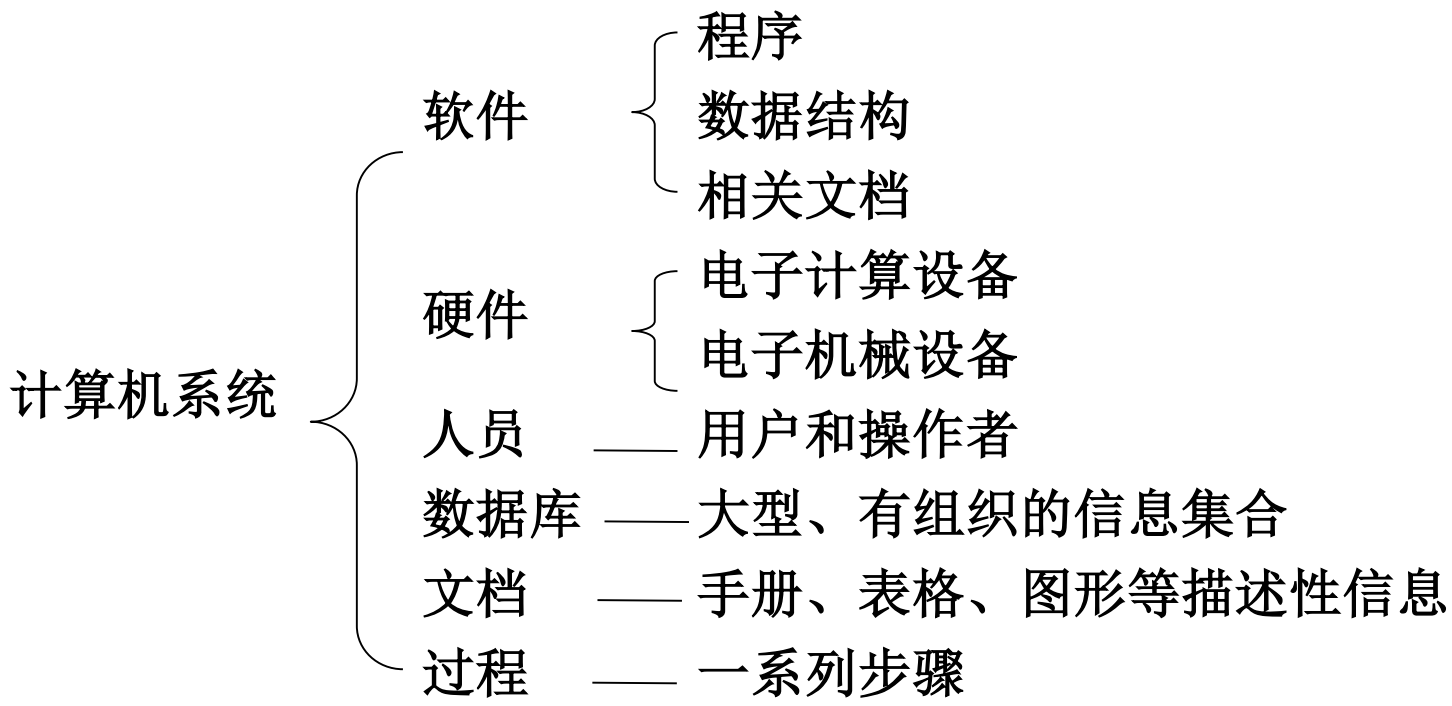
1.2 软件工程

1.3 软件生命周期

1.4 软件过程

# 1.1 软件危机

**计算机系统**：是指适当的组织在一起的一系列系统元素的集合，这些系统元素互相配合、相互协作，通过对信息的处理而完成预先定义的目标。



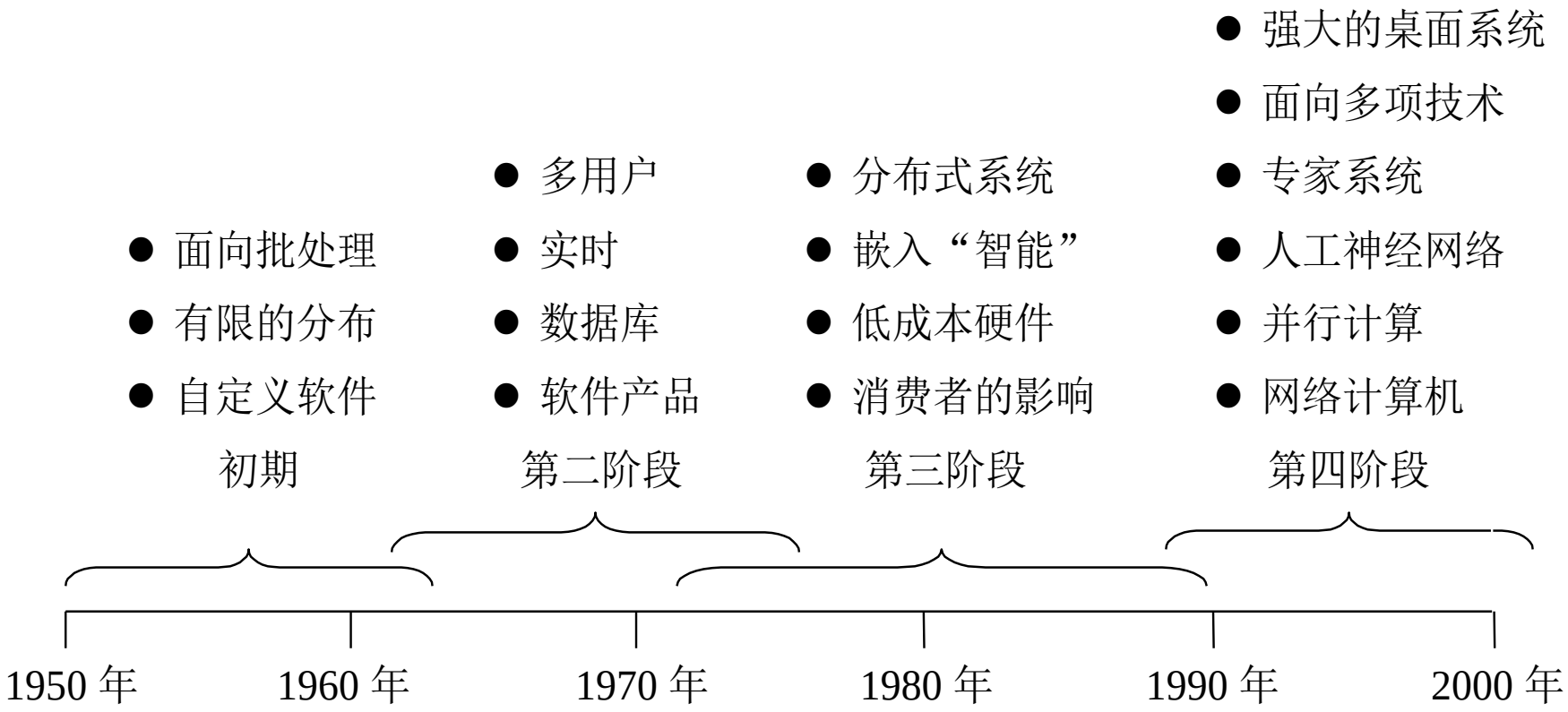
# 什么是软件

- **软件**是与计算机系统操作有关的程序、规程、规则及任何与之有关的文档及数据。包括两部分：
  - 可执行程序及相关数据；
  - 不可执行，但与软件开发、运行、维护使用和培训有关的文档；

- **程序**是用**程序设计语言**描述的、适合于计算机处理的语句序列。或者说是事先设计的能够满足计算机系统功能和性能要求的指令序列；
  - 机器语言；
  - 汇编语言：Z80, 8086/8088等;
  - 高级语言：FORTRAN,COBOL, PASCAL, C++, BASIC等；
  - 第四代语言：只需给出问题和输入以及输出的形式，即可得到结果的语言，如数据库查询语言、报表语言等；

# 什么是软件

- **数据**是使程序能正常操纵信息的数据结构；
- **文档**是一种数据媒体及其上所记录的数据。是与程序开发，维护和使用有关的图文材料。
  - 可行性分析报告；
  - 需求分析文档；
  - 系统设计文档；
  - 用户操作手册；
  - …… ……




**计算机软件发展的4个阶段**

**Early 1960s:**

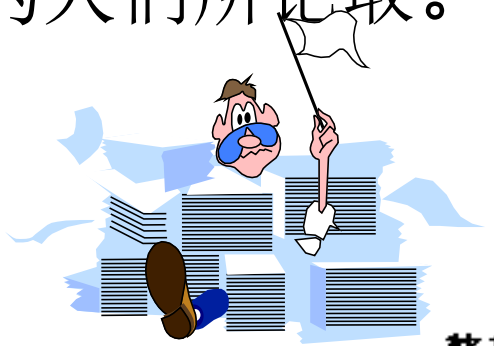
**Very few large software projects were done by some experts.**

Middle to late 1960s:

Truly large software systems were attempted.

例: 美国IBM公司在1963年至1966年开发的IBM360机的操作系统。这一项目花了5000人一年的工作量，最多时有1000人投入开发工作，写出了近100万行源程序。.....据统计，这个操作系统每次发行的新版本都是从前一版本中找出1000个程序错误而修正的结果。.....

这个项目的负责人F. D. 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，……谁也没有料到问题竟会陷入这样的困境……”。IBM360操作系统的历史教训成为软件开发项目的典型事例为人们所记取。



**Software Crisis !**

# 1.1.1 软件危机的介绍

软件危机(软件萧条、软件困扰)：是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。实际上，几乎所有软件都不同程度地存在这些问题。

软件危机包含下述两方面的问题：

- 如何开发软件，满足对软件日益增长的需求；
- 如何维护数量不断膨胀的已有软件。

## 软件危机的典型表现：

（1）对软件开发成本和进度的估计常常很不准确；

（2）用户对“已完成的”软件系统不满意的现象经常发生；

（3）软件产品的质量往往靠不住；

（4）软件常常是不可维护的；

（5）软件通常没有适当的文档资料；

（6）软件成本在计算机系统总成本中所占的比例逐年上升；

（7）软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势 。

# 1.1.2 产生软件危机的原因

## （1）与软件本身的特点有关

- 软件是逻辑部件。
- 软件不会被“用坏”，如果发现了错误，很可能是开发时期引入。
- 软件规模庞大，而且程序复杂性将随着程序规模的增加而呈指数上升。

## （2）与软件开发与维护的方法不正确有关

- 忽视软件需求分析的重要性。
- 认为软件开发就是写程序并设法使之运行。
- 轻视软件维护。

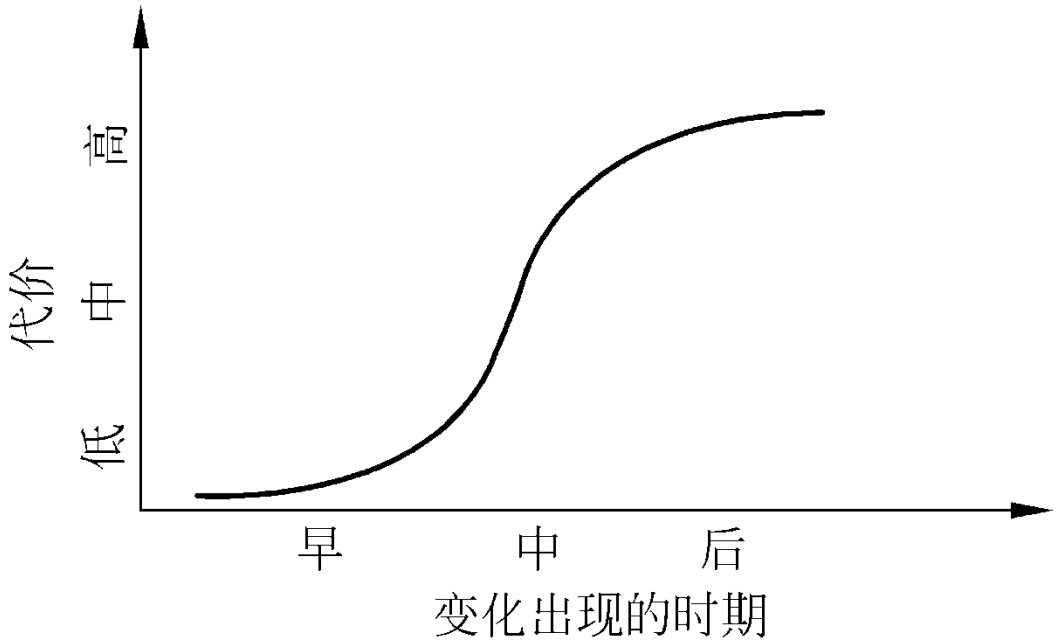



图1.1 引入同一变动付出的代价随时间变化的趋势

# 1.1.3 消除软件危机的途径

- 对计算机软件有正确的认识。
- 认识到软件开发是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。
- 应该推广使用在实践中总结出来的开发软件的成功技术和方法，并继续研究探索。
- 应该开发和使用更好的软件工具。
- 总之，为了解决软件危机，既要有技术措施(方法和工具)，又要有必要的组织管理措施。

# 1.2 软件工程

## 1.2.1 软件工程的介绍

软件工程：是指导计算机软件开发和维护的一门工程学科。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。

# 软件工程的代表性定义：

- **NATO会议：**软件工程是为了经济地获得可靠的且能在实际机器上高效运行的软件而建立和使用的完善的工程原理。
- **IEEE：**软件工程是（1）将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中；（2）研究（1）中提到的途径。

## 软件工程的本质特性：

- 软件工程关注于大型程序的构造
- 软件工程的中心课题是控制复杂性
- 软件经常变化
- 开发软件的效率非常重要
- 和谐地合作是开发软件的关键
- 软件必须有效地支持它的用户
- 在软件工程领域中是由具有一种文化背景的人替具有另一种文化背景的人创造产品

# 1.2.2 软件工程的基本原理

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

# 1.2.3 软件工程方法学

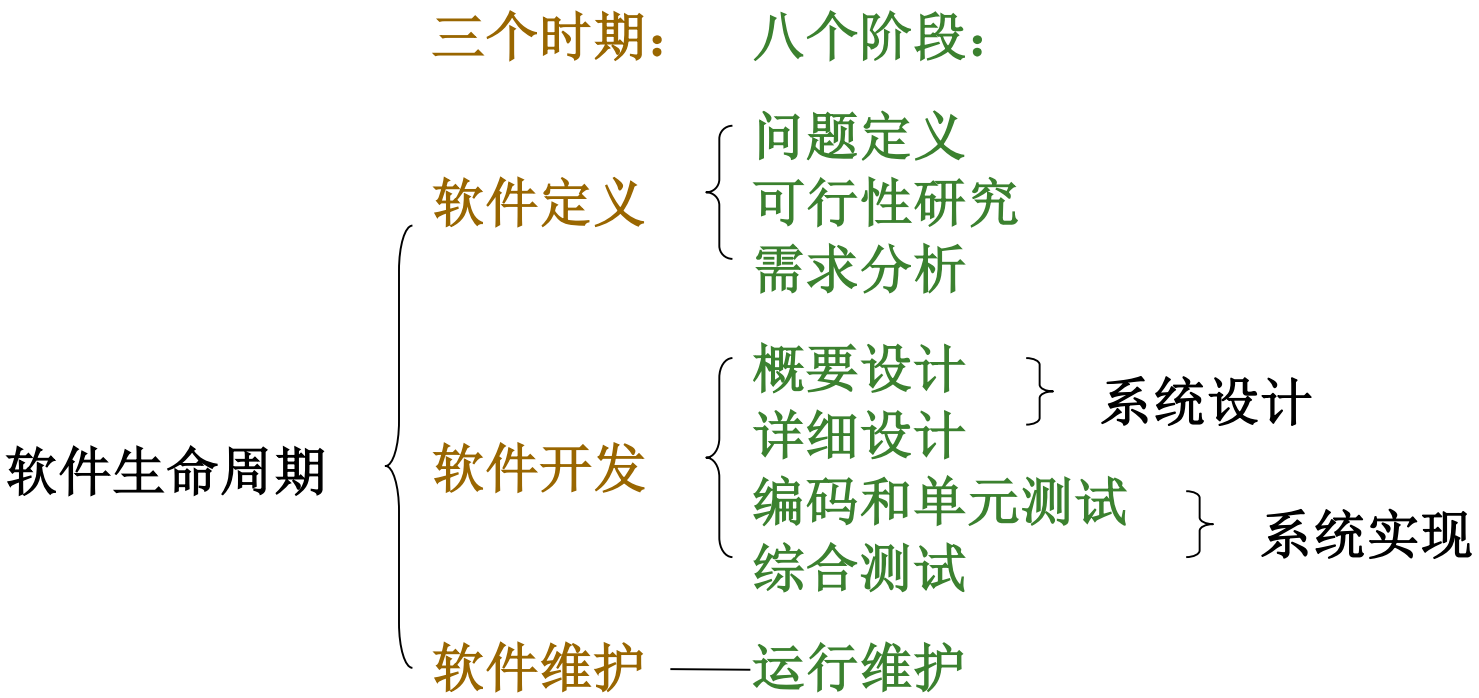
- 软件工程包括技术和管理两方面的内容。
- 管理：通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。
- 技术(软件工程方法学)：通常把在软件生命周期全过程中使用的一整套技术方法的集合称为方法学(methodology)，也称为范型(paradigm)。

# 软件工程方法学3要素：

- 方法：是完成软件开发的各项任务的技术方法，回答“怎样做”的问题；
- 工具：是为运用方法而提供的自动的或半自动的软件工程支撑环境；
- 过程：需要完成的一系列任务的框架完成各项任务的工作步骤。

# 1.3 软件生命周期

三个时期八个阶段：软件生命周期由软件定义、软件开发和运行维护(也称为软件维护)三个时期组成，每个时期又进一步划分成若干个阶段。



# 软件定义

1、软件定义：确定软件工程的工程需求。

阶段一：可行性研究

1）**技术可行性**：开发方法和工具是否可行？

2）**操作可行性**：用户能否使用？

3）**经济可行性**：成本能否被接受？

阶段二：需求分析

1）**任务**：功能需求、性能需求、运行环境约束等；

2）**重要性与困难**：需求是软件开发的关键和难点；

3）**需求分析过程**：是个反复迭代的过程；

4）**软件需求规格说明**：应该指明系统的功能需求、性能需求、接口需求、设计需求、基本结构、开发标准、验收原则等。

# 软件开发

2、软件开发：按照需求规格说明的要求，由抽象到具体，逐步形成软件的过程。包括：

1）**概要设计**：总体结构、模块间关系、功能模块接口、数据结构、设计约束、测试计划等；

2）**详细设计**：模块细化，形成可编程的程序模块。用过程设计语言设计模块细节等；

3）**实现**：将详细设计转化为程序，包括编程和调试；

4）**组装测试**：将经过单元测试的模块进行组装和测试；

5）**确认测试**：必须由客户参加，需生成测试报告、总结报告等文档；

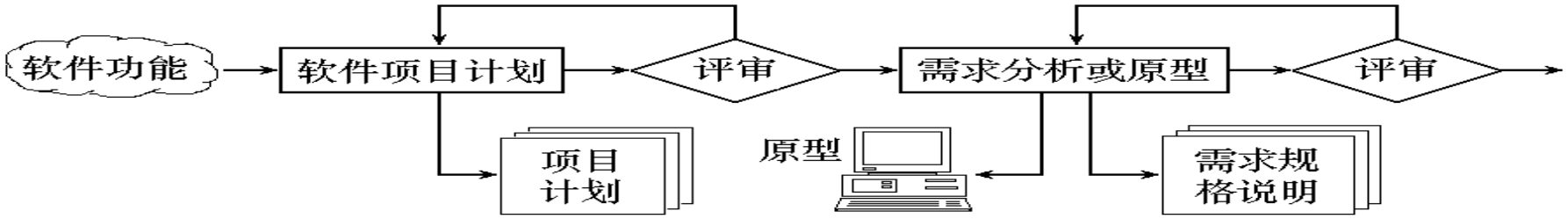
# 软件使用、维护和退役

3、软件使用、维护和退役

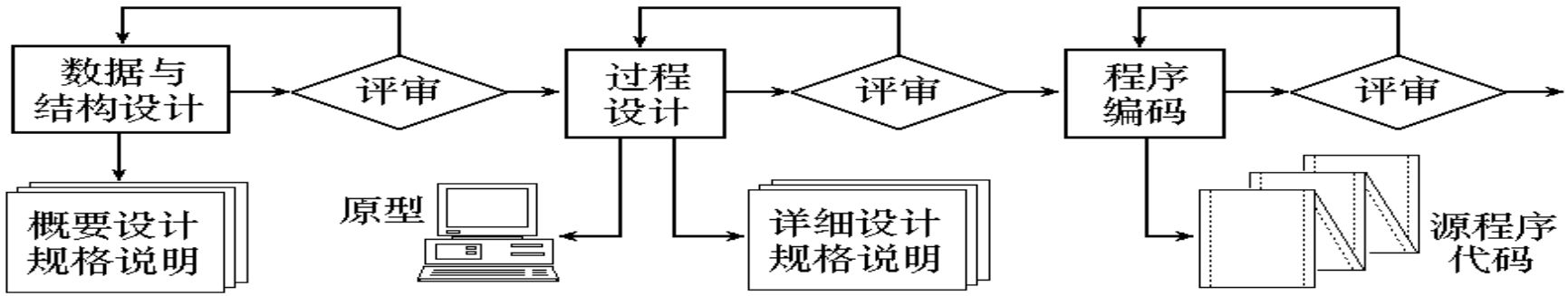
1）软件的使用：软件问题报告、软件修改报告；

2）软件的维护：对软件进行修改或对软件需求变化作出响应的过程，不仅是代码，还包括各种文档；

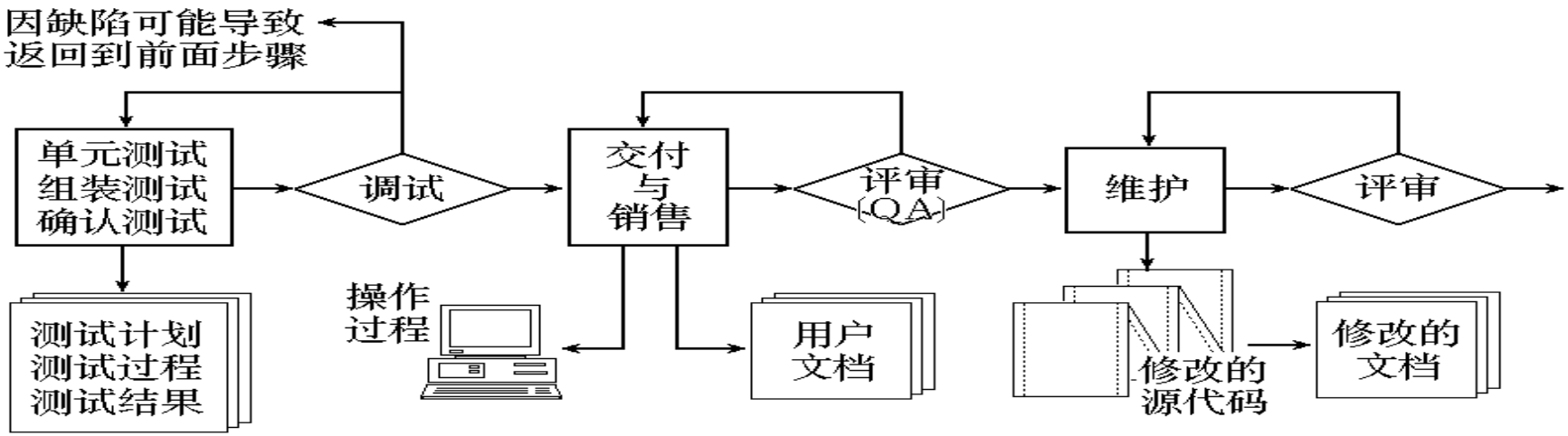
3）退役：软件停止使用；




（a）定义阶段




（b） 开发阶段




（c） 检验、交付与维护阶段

# 1.4 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。

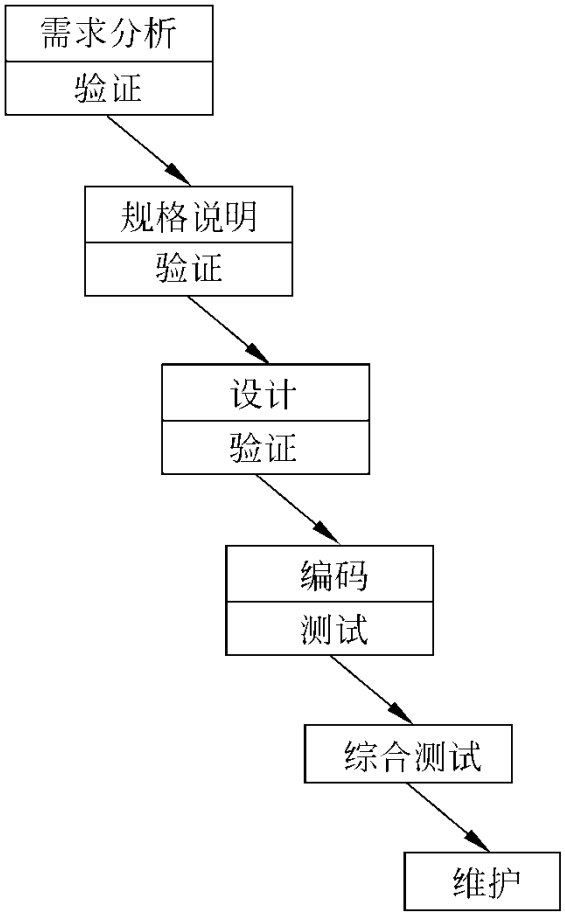
什么人（who）在什么时间（when）做什么事（what）以及怎样（how）做这些事以实现某一个特定的具体目标。

- 过程定义了运用方法的顺序、应该交付的文档资料、为保证软件质量和协调变化所需要采取的管理措施，以及标志软件开发各个阶段任务完成的里程碑。为获得高质量的软件产品，软件过程必须科学、有效。

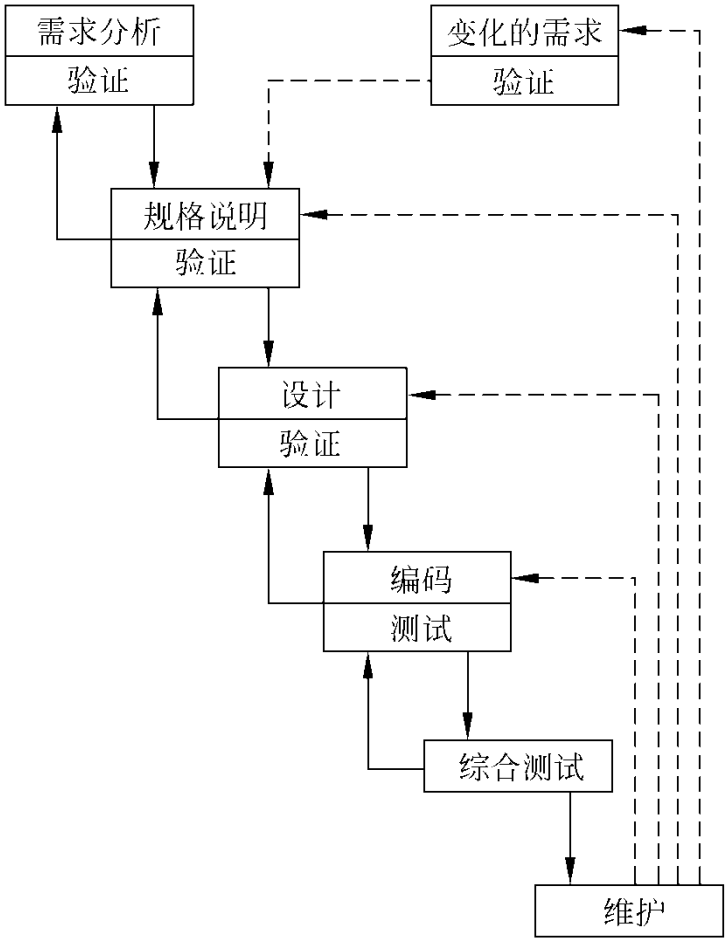
# 过程模型

- 典型的软件开发模型有：
  - 瀑布模型（waterfall model）
  - 原型模型(prototype model)
  - 增量模型（incremental model）
  - 螺旋模型（spiral model）
  - 喷泉模型（water fountain model）
  - 基于构件的开发模型（component-based development model）
  - 统一过程（Rational Unified Process）
  - 敏捷过程与极限编程（Agile eXtreme Programmming）
  - 微软过程（Microsoft）

# 1.4.1 瀑布模型




传统的瀑布模型




实际的瀑布模型

# 瀑布模型的特点：

1. 阶段间具有顺序性和依赖性

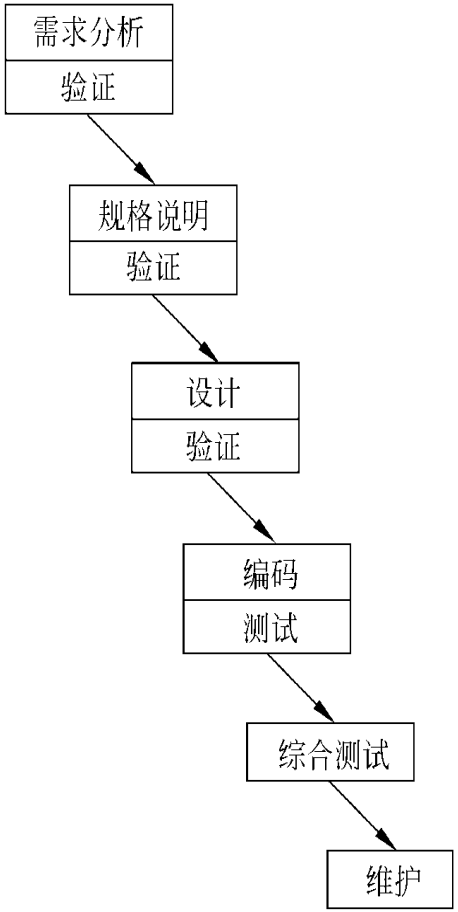
- 前一阶段的工作完成之后，才能开始后一阶段的工作；
- 前一阶段的输出文档就是后一阶段的输入文档。

2. 推迟实现的观点

- 对于规模较大的软件项目来说，往往编码开始得越早最终完成开发工作所需要的时间反而越长。

3. 质量保证的观点

- 每个阶段都必须完成规定的文档，是“文档驱动”的模型；
- 每个阶段结束前都要对所完成的文档进行评审，尽早发现问题，改正错误。

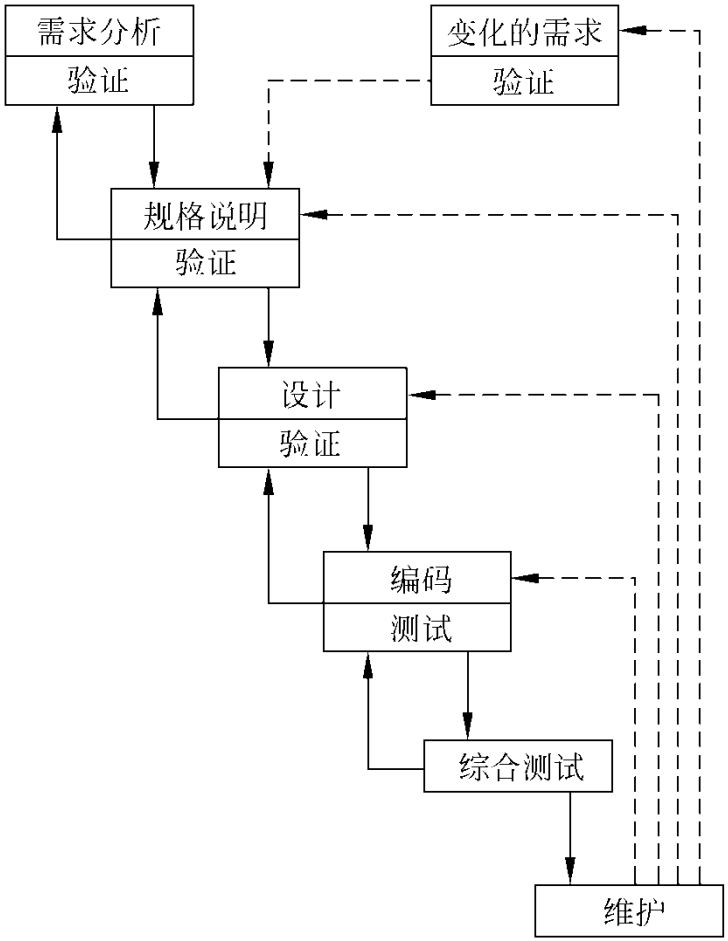


## 瀑布模型的优点：

- 可强迫开发人员采用规范的方法；
- 严格地规定了每个阶段必须提交的文档；
- 要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证。

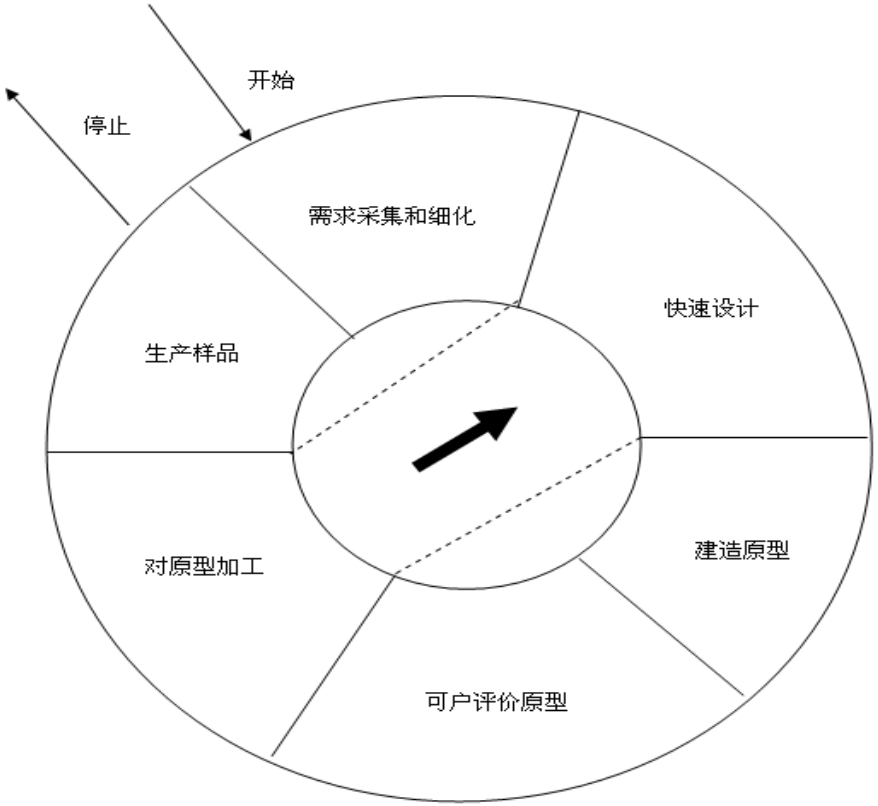
## 瀑布模型的缺点：

- 只能通过文档了解产品，不经过实践的需求是不切实际的。



# 1.4.2 快速原型模型

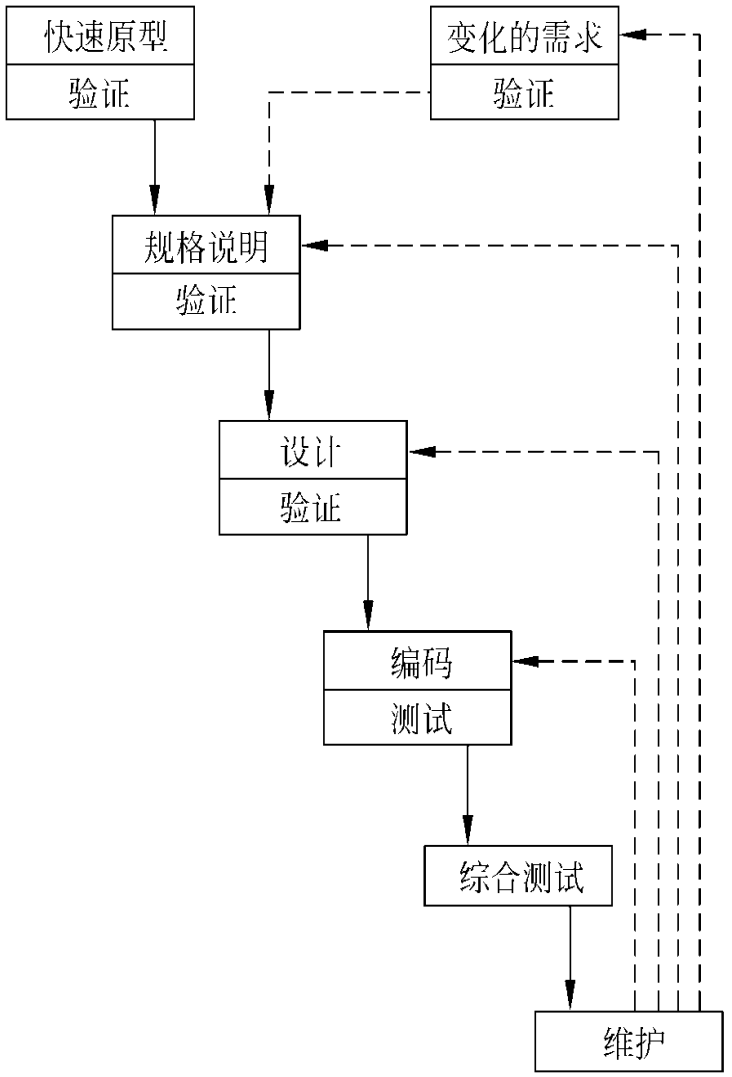
**快速原型**：是快速建立起来的可以在计算机上运行的程序，它所能完成的功能往往是最终产品能完成的功能的一个子集。




快速原型模型

# 快速原型模型的特点：

- 快速原型模型不带反馈环，软件产品的开发基本上是线性顺序进行的。
- 快速原型的本质是“快速”。应该尽可能快地建造出原型系统，以加速软件开发过程，节约成本。



根据原型的不同作用，有三类原型模型：

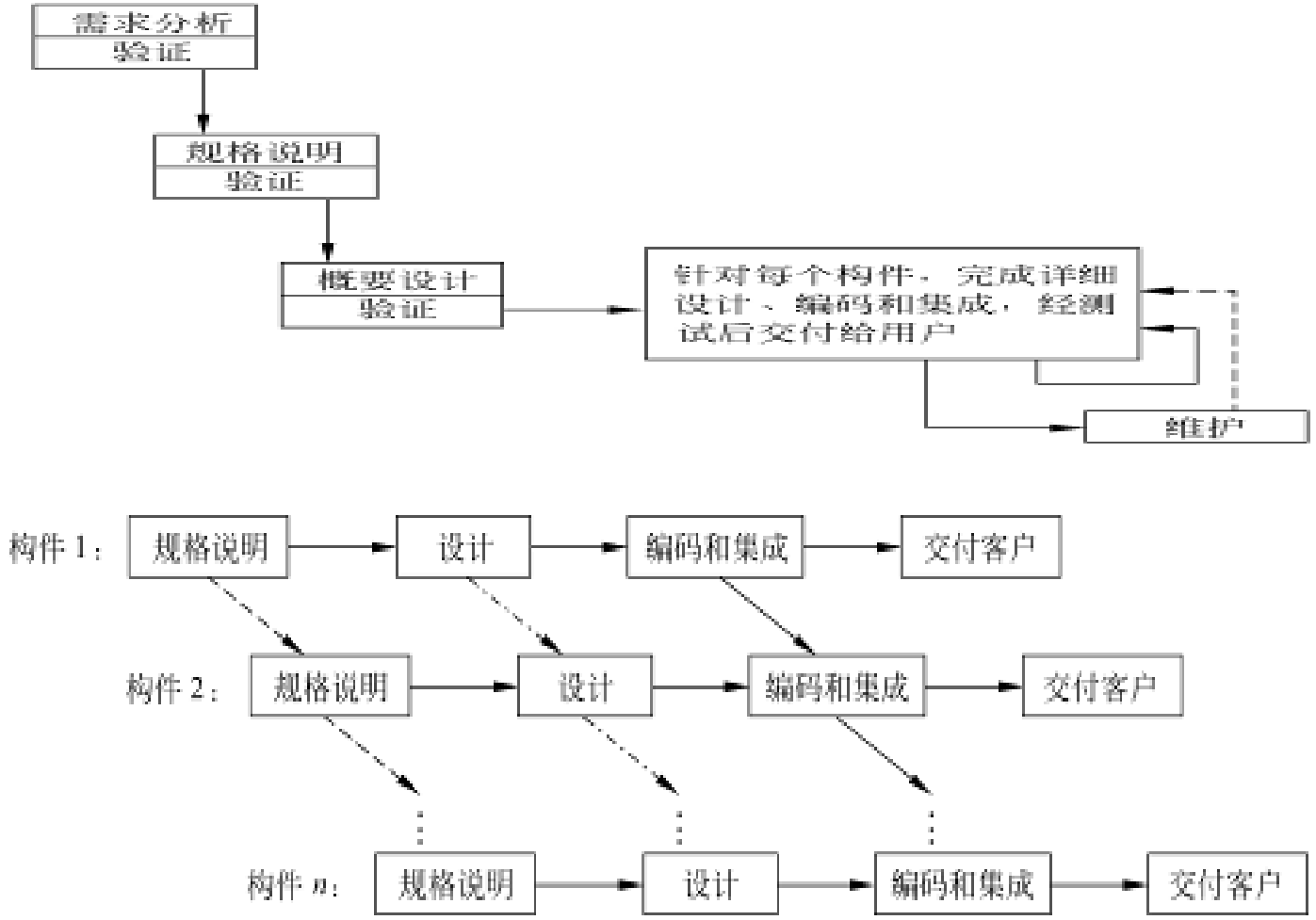
- 探索型原型——用于开发的需求分析阶段
- 实验型原型——主要用于设计阶段
- 演化型原型——用于及早向用户提交一个原型系统

快速原型模型的运用方式：

- 抛弃策略——探索型和实验型采用此策略
- 附加策略——演化型快速原型采用此策略

# 1.4.3 增量模型

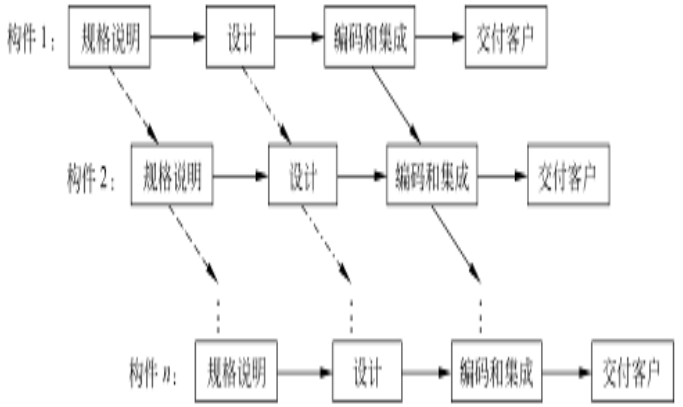
- 增量模型把软件产品作为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成，并且能够完成特定的功能。


需求分析
验证
规格说明
验证
概要设计
验证
针对每个构件，完成详细设计、编码和集成，经测试后交付给用户
维护
构件 1：
规格说明
设计
编码和集成
交付客户
构件 2：
规格说明
设计
编码和集成
交付客户
构件 n：
规格说明
设计
编码和集成
交付客户

# 增量模型

## 增量模型的优点：

- 人员分配灵活，刚开始不用投入大量人力资源。
- 当配备的人员不能在设定的期限内完成产品时，它提供了一种先推出核心产品的途径。
- 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。




## 增量模型的难点：

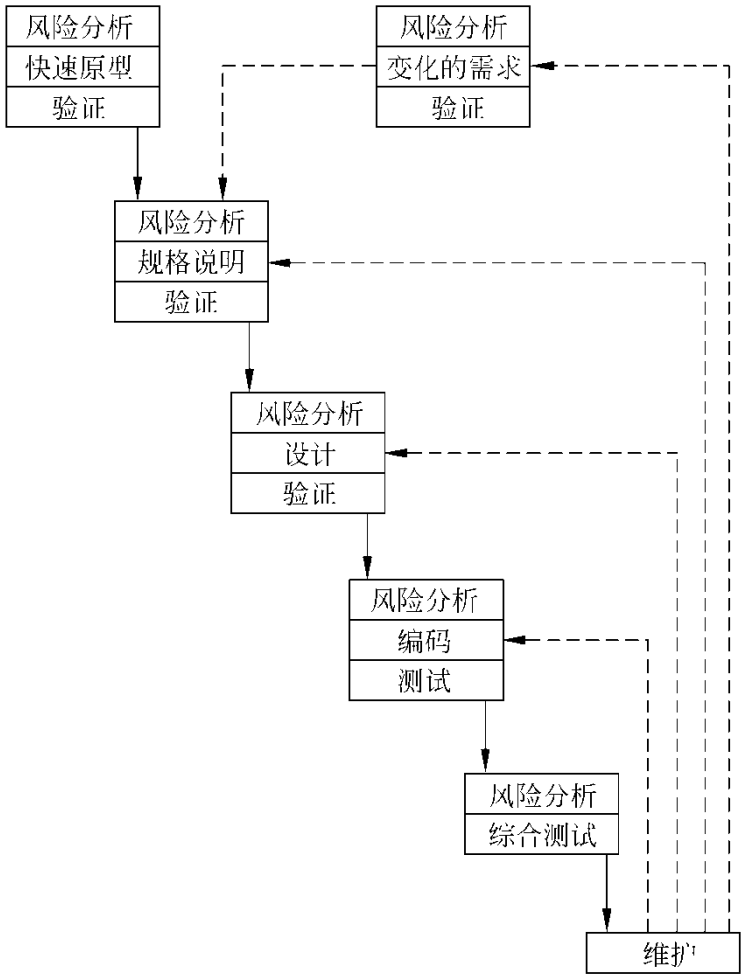
- 软件体系结构必须是开放的。
- 不同的构件并行地构建有可能

加快工程进度，但是冒无法集成到一起的风险。

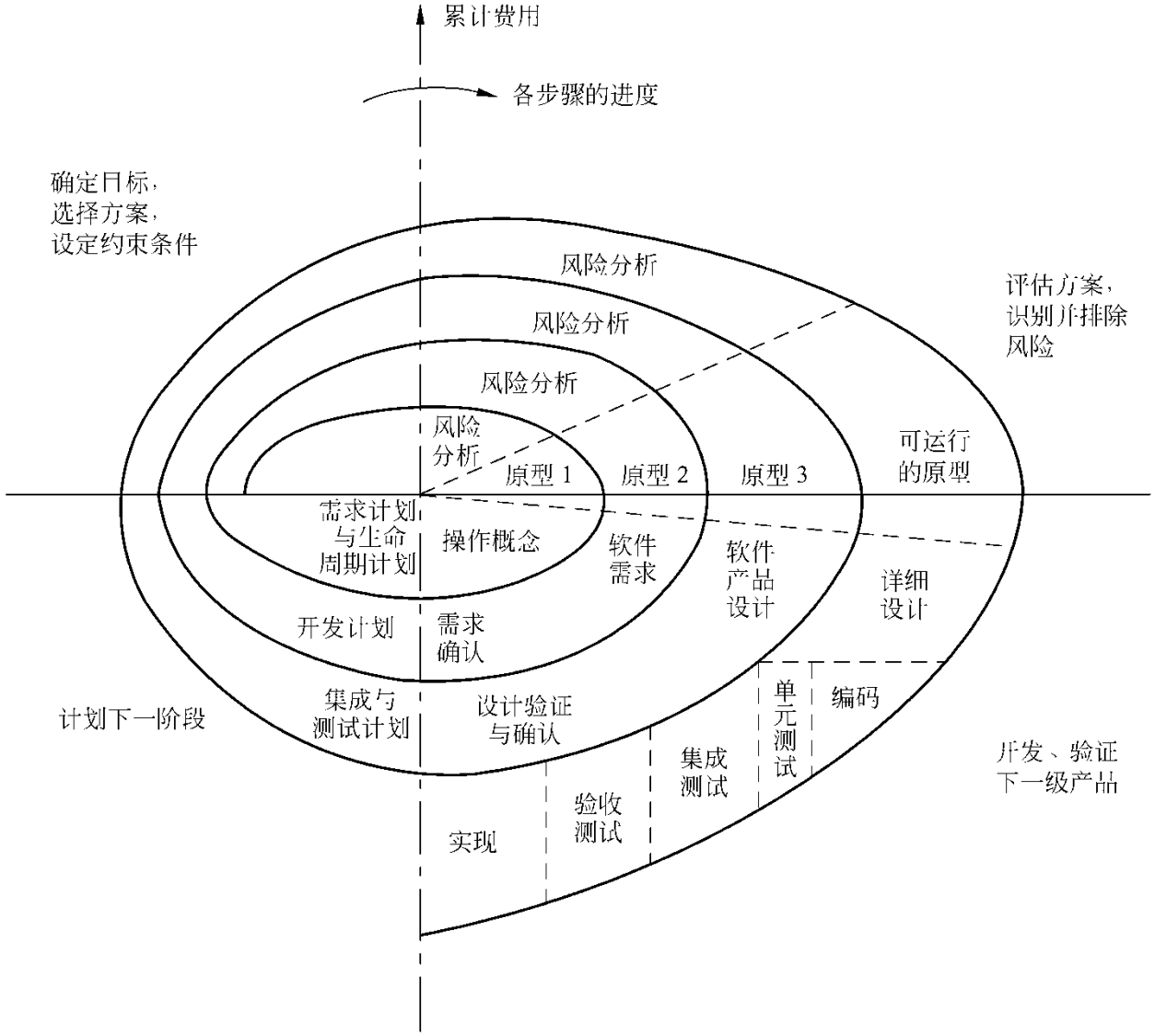
# 1.4.4 螺旋模型

## 螺旋模型的基本思想：

使用原型及其他方法来尽量降低风险。把它看作在每个阶段之前都增加了风险分析过程的快速原型模型。




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
开发计划
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
验收
测试
实现
开发、验证
下一级产品

# 完整的螺旋模型

## 螺旋模型的缺点：

- 采用螺旋模型需要具有相当丰富的风险评估经验和专门知识，在风险较大的项目开发中，如果未能够及时标识风险，势必造成重大损失。
- 过多的迭代次数会增加开发成本，延迟提交时间。

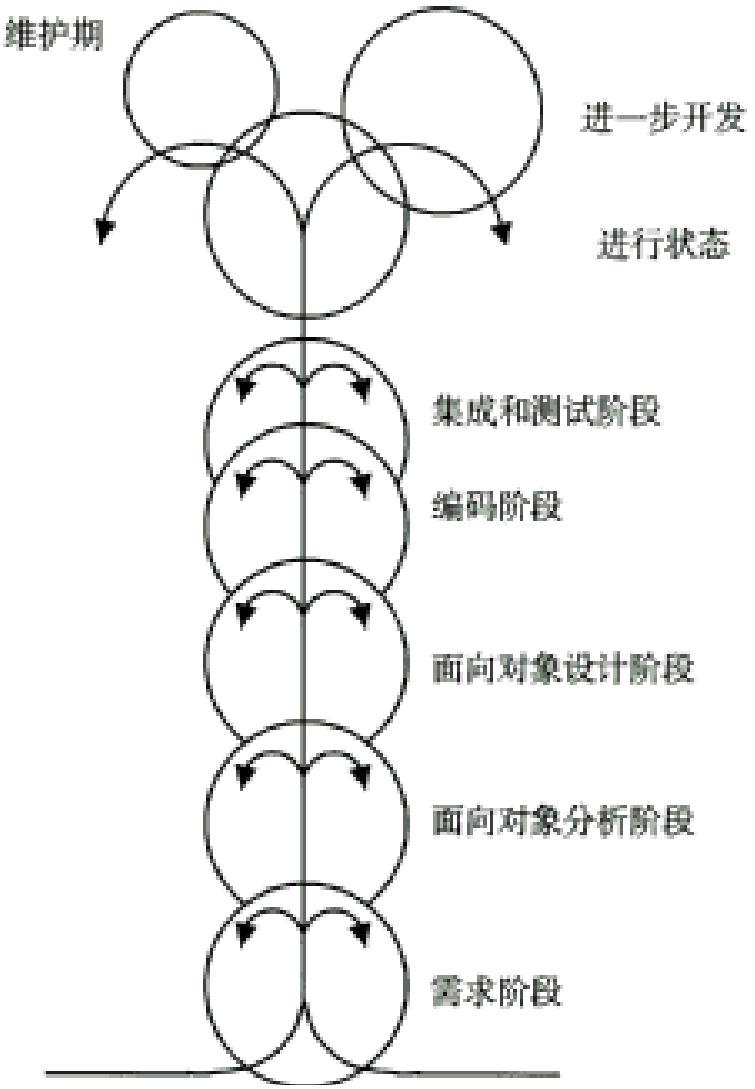
## 螺旋模型适用于：

- 特别适用于庞大、复杂并具有高风险的系统。
- 适用于内部开发的大规模软件项目。

# 1.4.5 喷泉模型

**喷泉模型：**是典型的面向对象生命周期模型。

“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。



## 喷泉模型的优点：

- 该模型的各个阶段没有明显的界限，开发人员可以同步进行开发。
- 多次反复地增加或明确目标系统，而不是本质性的改动，降低错误的可能性。

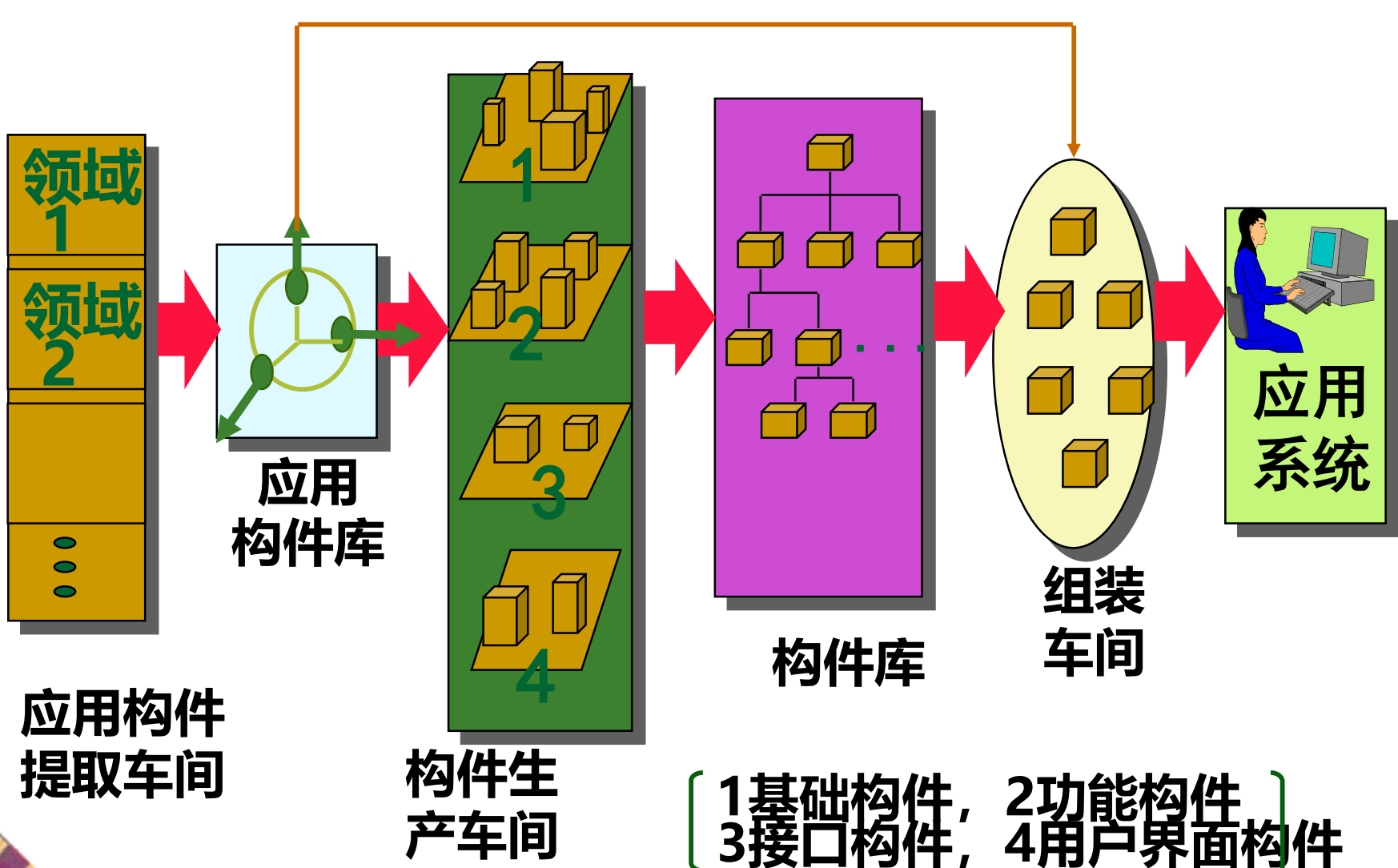
## 喷泉模型的缺点：

- 由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，不利于项目的管理。
- 要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

## 喷泉模型适用于：

- 适用于面向对象的软件开发过程。

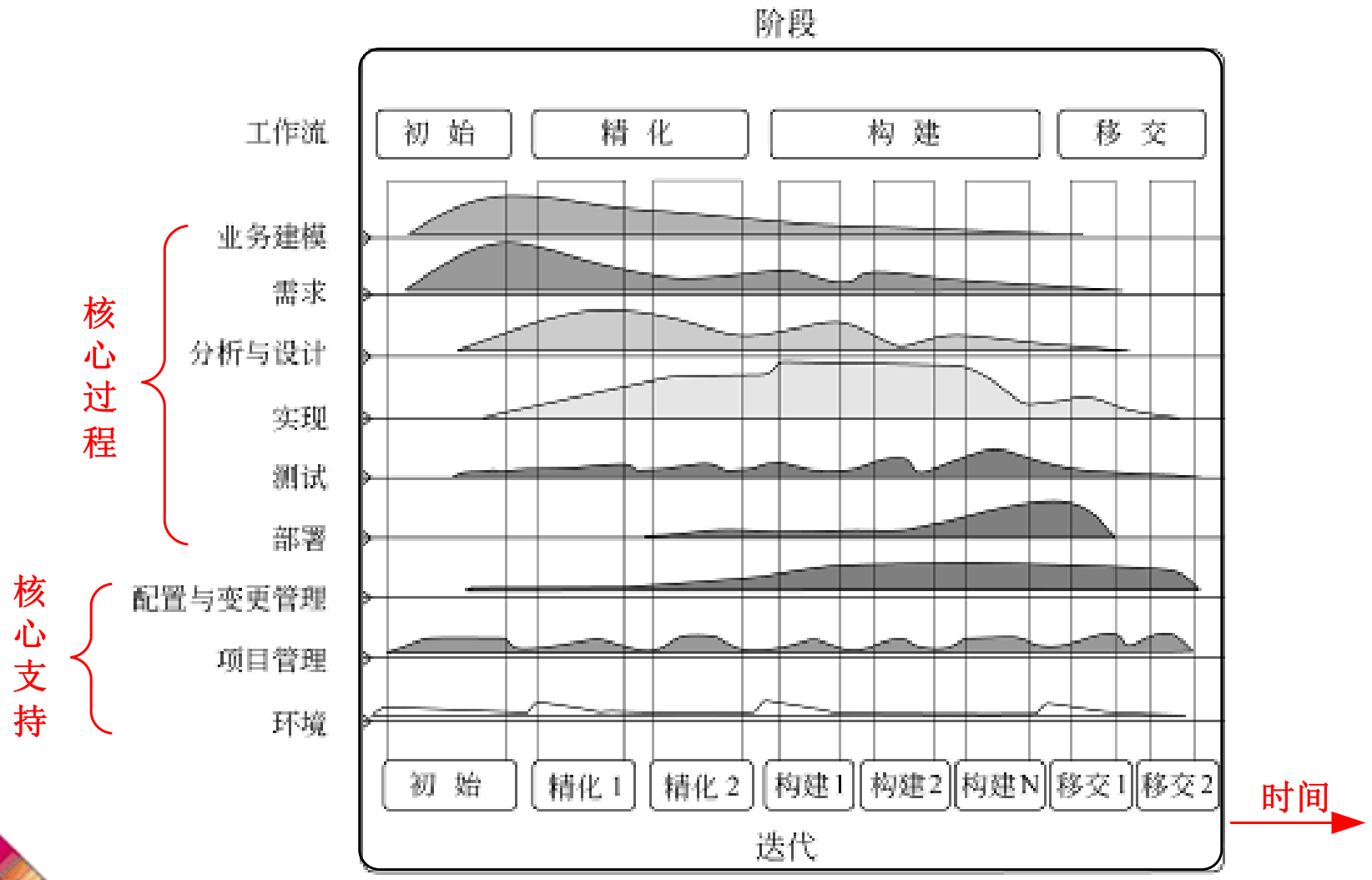
# 基于构件的开发模型：


领域1
领域2
应用构件提取车间
应用构件库
1
2
3
4
构件生产车间
构件库
组装车间
应用系统
1基础构件，2功能构件
3接口构件，4用户界面构件

# 1.4.6 Rational统一过程

- Rational统一过程(Rational Unified Process, RUP)是由Rational软件公司推出的一种完整而完美的软件过程。
- RUP是一种迭代的，以架构为中心的，用例驱动的软件开发方法。
- RUP是一种具有明确定义和结构的软件工程过程。
- RUP还是一个过程产品，提供了可定制的软件工程的过程框架。
- RUP被广泛应用在不同工业领域中的不同企业中。

# RUP软件开发生命周期(二维)：



# 1.4.7 敏捷过程与极限编程

敏捷软件开发宣言(价值观声明)：

- 个体和交互胜过过程和工具
- 可以工作的软件胜过面面俱到的文档
- 客户合作胜过合同谈判
- 响应变化胜过遵循计划

根据上述价值观声明提出的软件过程统称为敏捷过程。

# 极限编程(eXtreme Programming, XP):

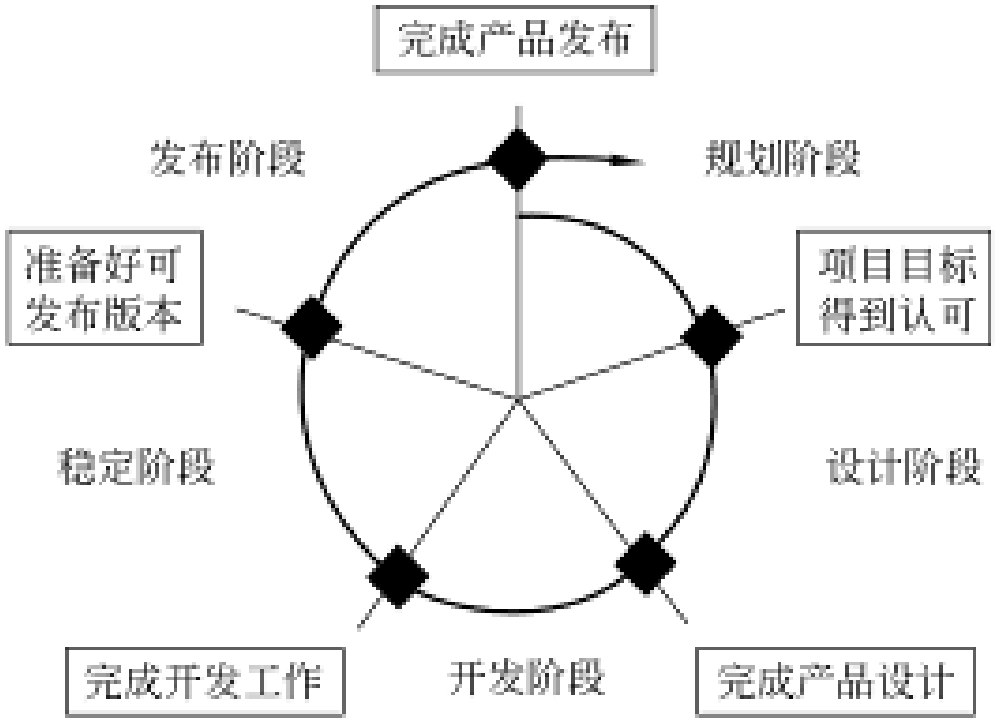
- 敏捷过程中最富盛名的一个
- 广泛适用于需求模糊且经常改变的场合
- 使得敏捷过程能够较好地适应商业竞争环境下对小型项目提出的有限资源和有限开发时间的约束

# 极限编程

- 极限编程的有效实践
  - 客户作为开发团队的成员
  - 使用用户素材
  - 短交付日期
  - 结对编程
  - 测试驱动方案
  - 集体所有
  - 持续集成
  - 及时调整计划

# 1.4.8 微软过程

- 微软软件生命周期

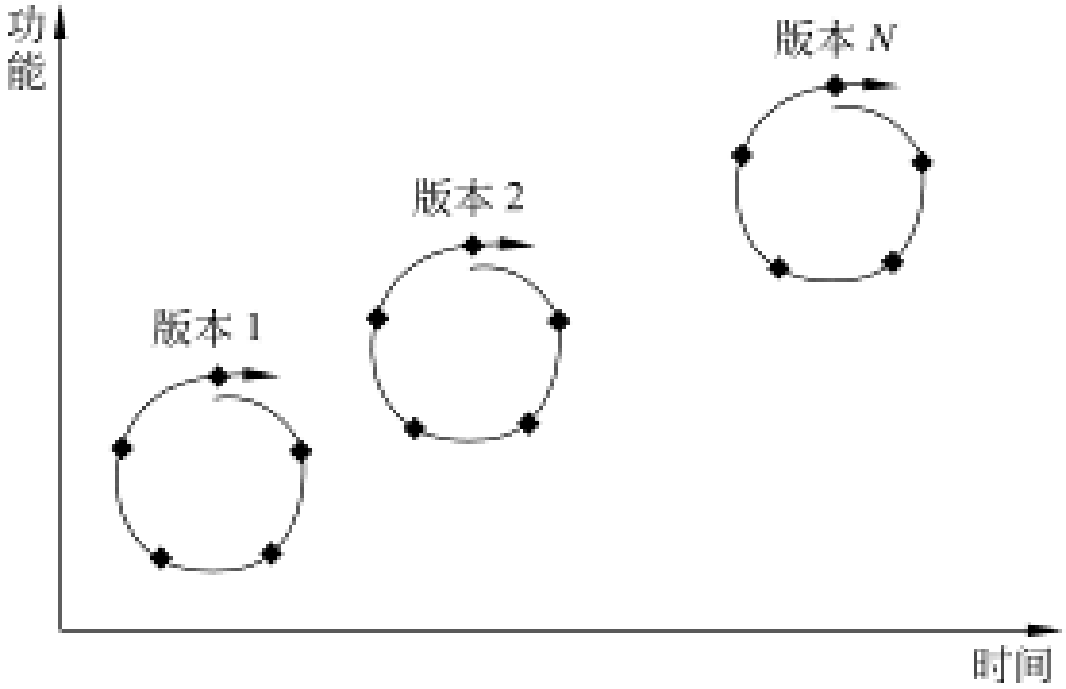


# 微软过程

- 微软过程准则
  - 项目计划应该兼顾未来的不确定因素
  - 用有效的风险管理未来不确定因素的影响
  - 快速生成测试软件的过渡版本，提高产品的稳定性和可预测性
  - 采用快速循环递进的过程
  - 用创造性的工作来平衡产品特性和产品成本
  - 项目进度表应该具有较高稳定性和权威性
  - 使用小型项目组并发地完成开发工作
  - 在项目早期把软件配置项极限化，项目后期则冻结产品
  - 使用原型验证概念，对项目进行早期论证
  - 零缺陷作为追求的目标
  - 里程碑评审会的目的是改进工作，切忌相互指责

# 微软过程模型

- 每一个生命周期发布一个递进的版本，各生命周期持续快速地迭代循环
- 优点： 综合了Rational统一过程和敏捷过程的优点
- 缺点：对方法、工具和产品等方面不够全面



# 软件神话—管理神话

负责软件的管理者像大多数其他行业的管理者一样，都有巨大的压力，要维持预算、保持进度，还要提高质量。就像溺水者抓住一根救命稻草，软件管理者常常抓住软件神话不放，这些神话能够缓解其压力的话（哪怕是暂时的）。

- 神话：如果我们已经落后于计划，可以增加更多的程序员赶上进度。

现实：给一个已经延迟的软件项目增加人手只会使其更加延迟。

# 软件神话—客户神话

在许多情况下，客户相信关于软件的神话，因为负责软件开发的管理者和开发人员很少去纠正客户的错误理解。导致客户过高的期望值，并最终引起对开发人员的不满意。

- 神话：软件需求确实是经常变更的，但这些变更能够很容易地满足，因为软件是灵活的。

现实：软件需求确实是变更的，但这些变更产生的影响会随着其被引入的时间而不同的。

# 软件神话—实践者神话

- 神话1：一个成功项目唯一应该提交的就是运行程序。

  现实：运行程序仅是软件配置的一部分，软件配置包括很多东西。

- 神话2：软件工程将使我们创建大量的、不必要的文档，并总是延缓我们的进度。

  现实：软件工程并不是为创建文档，而了创建质量。

# 小结

1 软件危机

2 软件工程

3 软件生命周期

4 软件过程