第2章 软件过程

讲授： 任胜兵
中南大学 计算机学院

2.2 软件工程经典生存周期模型

中南大學
CENTRAL SOUTH UNIVERSITY

内容提要

- 生存周期模型定义
- 瀑布模型
- 快速原型模型
- RAD模型
- 螺旋模型
- 统一过程模型

不同软件的生成周期不完全相同

01 软件生存周期模型定义

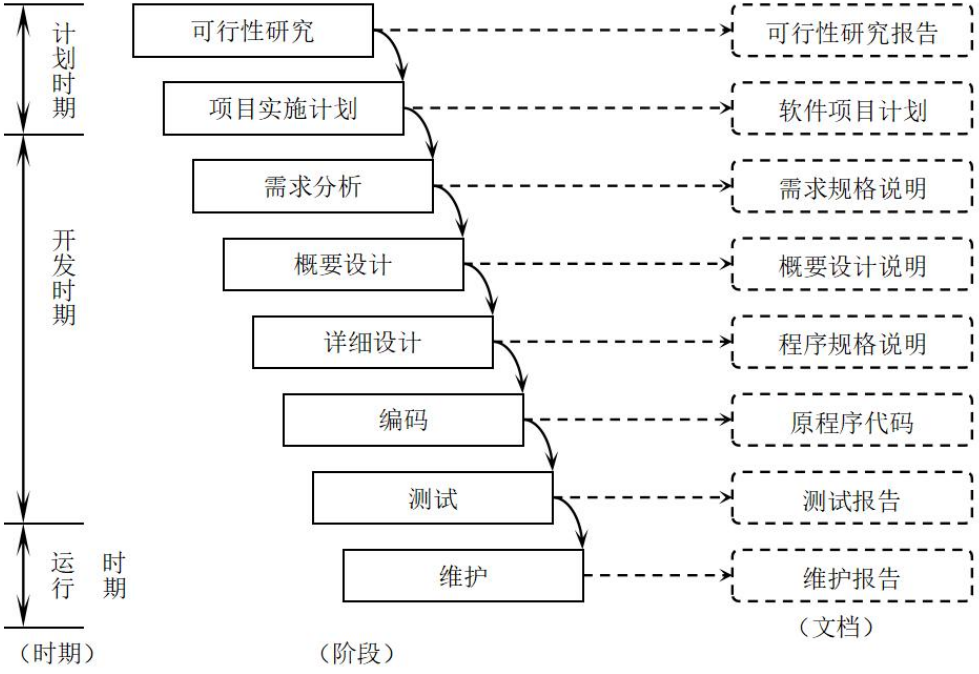
软件生存周期模型，亦称软件开发模型，描述了从软件项目需求定义开始，到开发成功后投入使用，在使用中不断增补修订，直到停止使用，这一期间各种活动如何执行的模型。

软件开发机构应该综合项目和应用的性质、将要使用的方法和工具等，选择其中合适的模型或构建，并将软件生存期过程映射到选定的模型中进行软件开发和维护。

软件生存周期模型反映了软件开发和维护活动。

瀑布模型具有划时代意义

瀑布模型（Waterfall model），也称为传统的生命周期模型。瀑布模型最初是温斯顿·罗伊斯（Winston Royce）在1970年提出的。



瀑布模型构建了软件产品流水线

瀑布模型的特点。

- 各阶段顺序相互依赖
  - 形成了软件生产流水线（软件产业化、职业化）
- 每阶段生成文档并进行评审
  - 文档驱动（便于管理）
  - 软件过程可视化（便于管理）
  - 质量保障（便于管理）
- 强调需求分析和设计
  - 推迟物理实施（有助于提高质量）

瀑布模型比较适合于功能和性能需求明确的软件项目的开发和维护，如编译器、操作系统等。

瀑布模型构建了软件产品流水线

瀑布模型的不足。

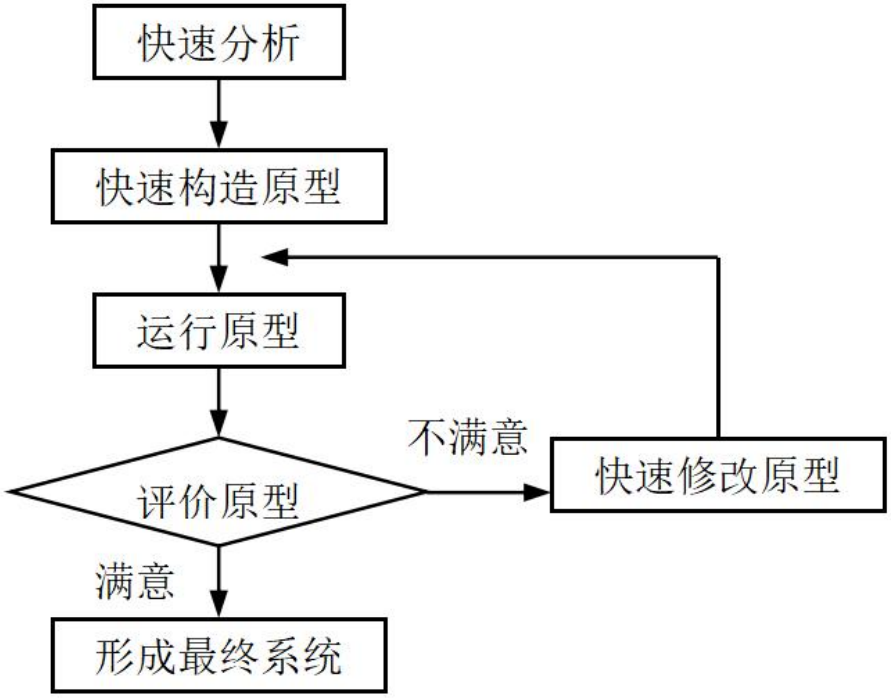
- 开发前期用户需求模糊不全面
  - 难以适应需求模糊的情况
- 在开发过程中，用户看不见系统
  - 得不到用户的反馈
  - 容易导致开发出的系统并不是用户真正需要的系统
- 需求不确定或快速变化
  - 返工代价大，难以适用非线性顺序的开发活动

瀑布模型比较适合于功能和性能需求明确的软件项目的开发和维护，如编译器、操作系统等。

瀑布模型构建了软件产品流水线

原型可用于获取用户反馈

原型是一个可以实际运行的模型，它在功能上可以看作是最终产品的一个子集（展示了目标系统的关键功能）。



快速原型不可能像最终产品一样面面俱到

快速原型模型的特点。

- 通过向用户提供原型获取用户的反馈
  - 用户使用原型能够真正反映用户的需求（百闻不如一见）
- 采用逐步求精方法完善原型
  - 避免冗长的开发过程
- 快速开发原型
  - 便于快速响应用户意见
- 不断迭代改进，更符合人们开发软件的习惯

原型模型比较适合于需求模糊或不确定的软件项目的开发和维护。

快速原型不可能像最终产品一样面面俱到

快速原型模型的不足。

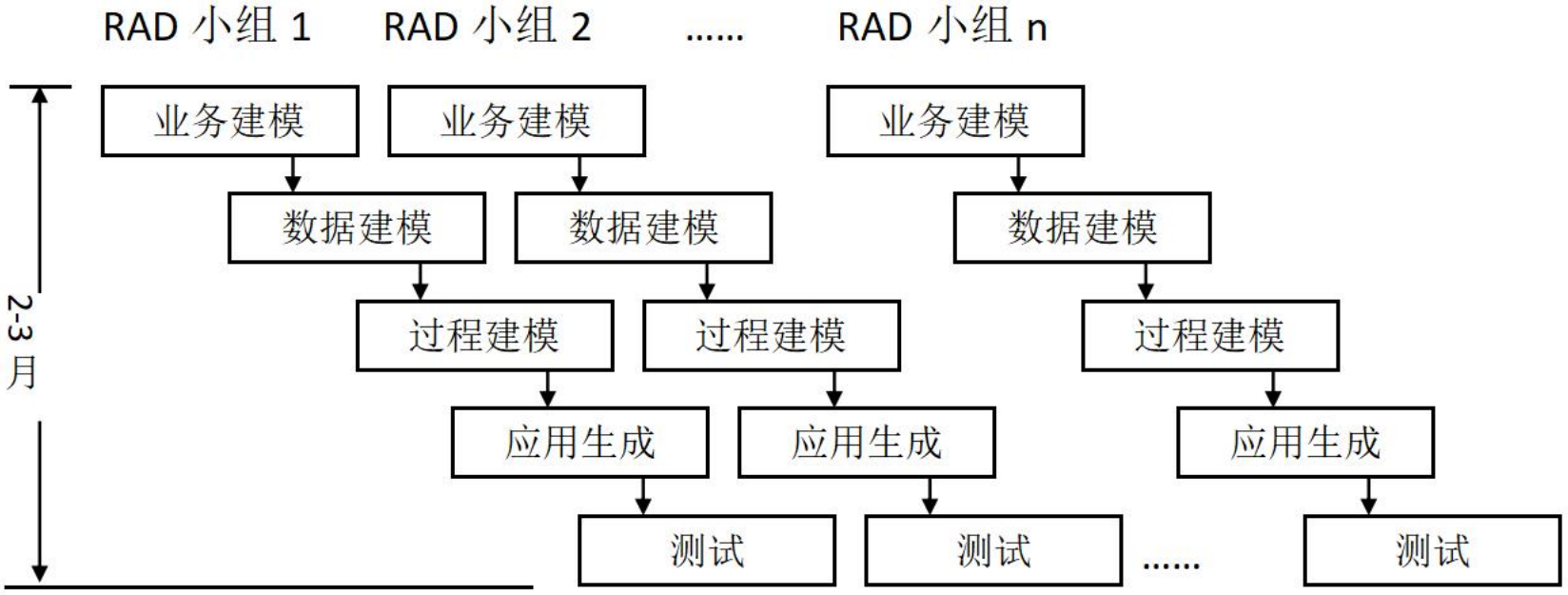
- 不宜利用原型系统作为最终产品（原型成本问题）
  - 原型可能对软件系统异常等问题虑较少，
  - 原型主要用于获取需求
  - 有些原型也可以融入最终系统
- 原型模型的“快速”特点对最终系统不适用
  - 原型开发往往尽量使用能缩短开发周期的语言和工具
  - 最终系统修改可能不像原型修改一样快

采用原型模型开发系统，用户和开发者必须达成一致：原型被建造仅仅是用来定义需求，之后便部分或全部抛弃，最终的软件要在充分考虑了性能和可维护性等质量方面之后才被开发。

将原型模型和瀑布模型结合

利用原型模型和瀑布模型的优点

RAD模型也是一种线性的软件开发模型，但是，这种模型特别强调采用极短的开发周期（比如2~3个月）。



快速应用开发模型使用基于组件的程序构造方法实现

快速应用开发模型的特点。

- 顺序开发（如同瀑布模型）
  - 业务建模：弄清业务活动中的信息流；
  - 数据建模：精化业务建模结果；
  - 处理建模：依据数据建模结果，创建处理描述；
  - 应用生成：组件复用与开发；
  - 测　　试：新的组件及所有接口。
- 强调极短的开发周期（2-3月）
  - 避免冗长的开发过程

RAD模型主要用于信息系统应用软件的开发

RAD模型有原型模型和瀑布模型的特点

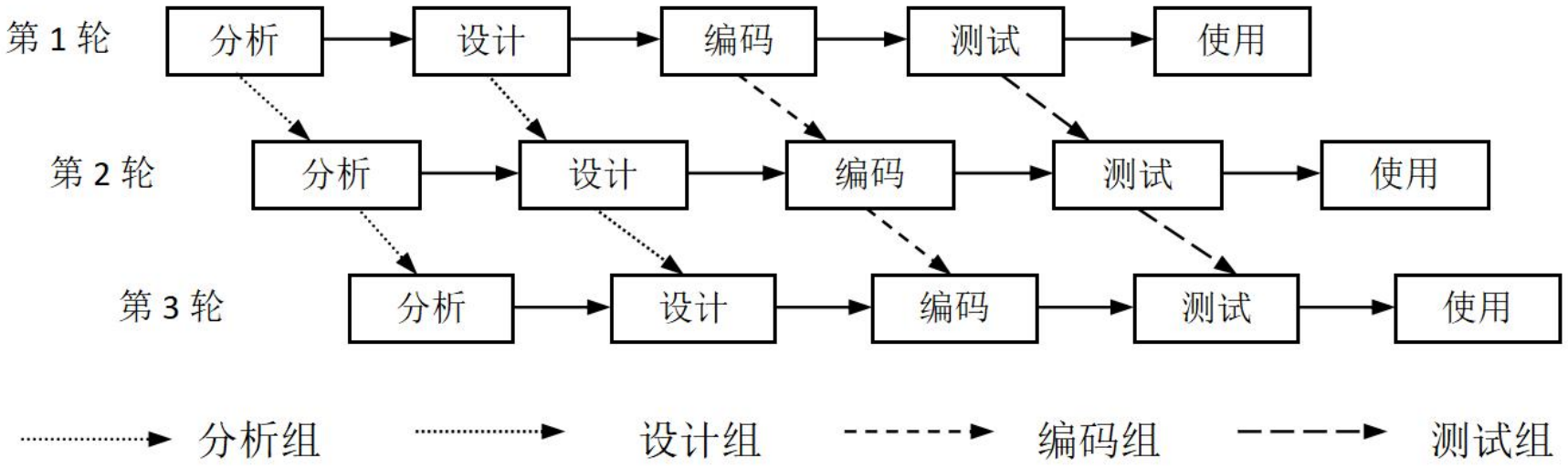
快速应用开发模型的不足。

- 需要足够的人力以创建足够的RAD小组
  - RAD需要若干RAD小组。
- 技术风险很高的情况不适合采用
  - 新软件要求与已存在的程序有高可互操性时
  - 系统难以被适当地划分为若干独立组件
- 开发者和用户需要做到在很短的时间内完成系统开发

RAD模型有原型模型和瀑布模型的特点

一次性地将整个系统交付给用户易产生不适应

演化模型都是迭代增量式的，使得软件工程师能够开发出越来越完善的软件版本。渐增模型由若干轮开发过程组成，每一轮都是在前一轮基础上进行。



以逐步增加软件产品的方式构造软件

渐增模型的特点。

- 渐增模型结合了瀑布模型的直线式特点和快速原型化模型的迭代思想
  - 渐增模型中的每一轮开发过程是直线式的
  - 渐增模型每一轮都是在前一轮基础上迭代
- 渐增模型的每一轮都是产品
  - 原型模型每一轮得到的是原型
  - 不需要大的资金支出、投资回报随功能渐增而渐增
  - 能够减少全新软件产品对用户带来的影响
  - 用户能及早使用及早发现问题
  - 能够有计划地管理技术风险
- 可以根据需要补充人员

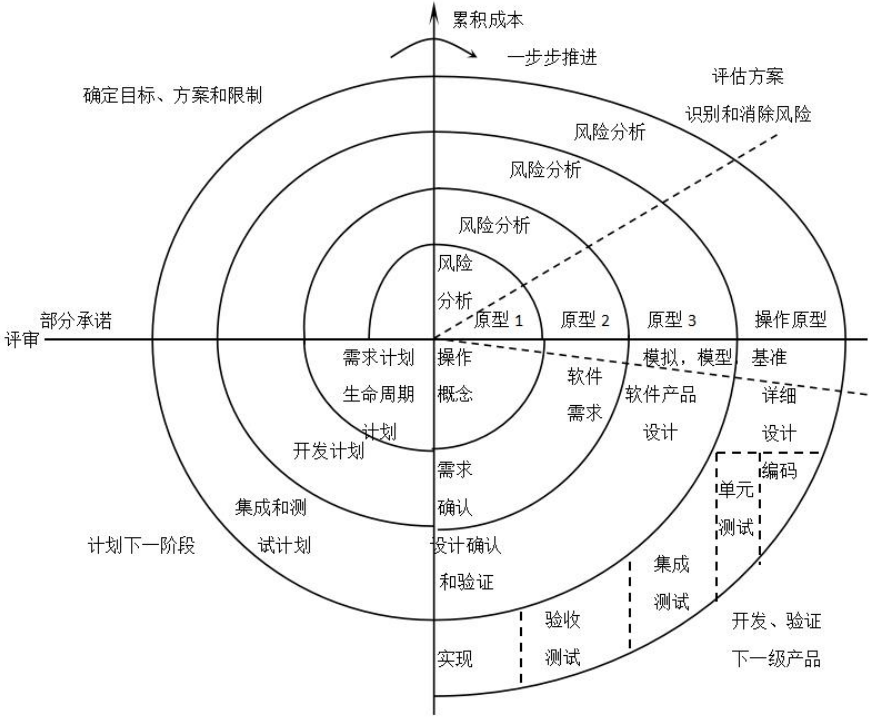
RAD模型有原型模型和瀑布模型的特点

渐增模型的不足。

- 对设计水平要求较高
  - 如果产品整体结构设计不当，则难以为其增加新的增量
- 由于采用增量开发，故难于进行彻底的测试
  - 有些增量开发在后期

软件项目的开发过程中几乎总是伴随着风险

通过使用原型或其它类似的手段以降低风险的思想，这是螺旋模型（Spiral model）中蕴涵的基本思想。最早由巴利·玻姆（Barry Boehm）于1988年提出。



螺旋模型是风险驱动的

螺旋模型的特点。

- 螺旋模型是风险驱动的
  - 风险分析使得用户和开发者能够更好地理解和对待每一个阶段的风险
  - 风险分析的目的是去风险
- 结合了诸多模型的特点
  - 保持了传统生命周期模型中系统的阶段性方法
  - 通过建造原型来排除风险
  - 自内向外，逐步延伸，体现了渐增模型的特点
- 螺旋模型适合于大型软件的开发
  - 尤其是内部创新性项目

螺旋模型结合了诸多模型的特点

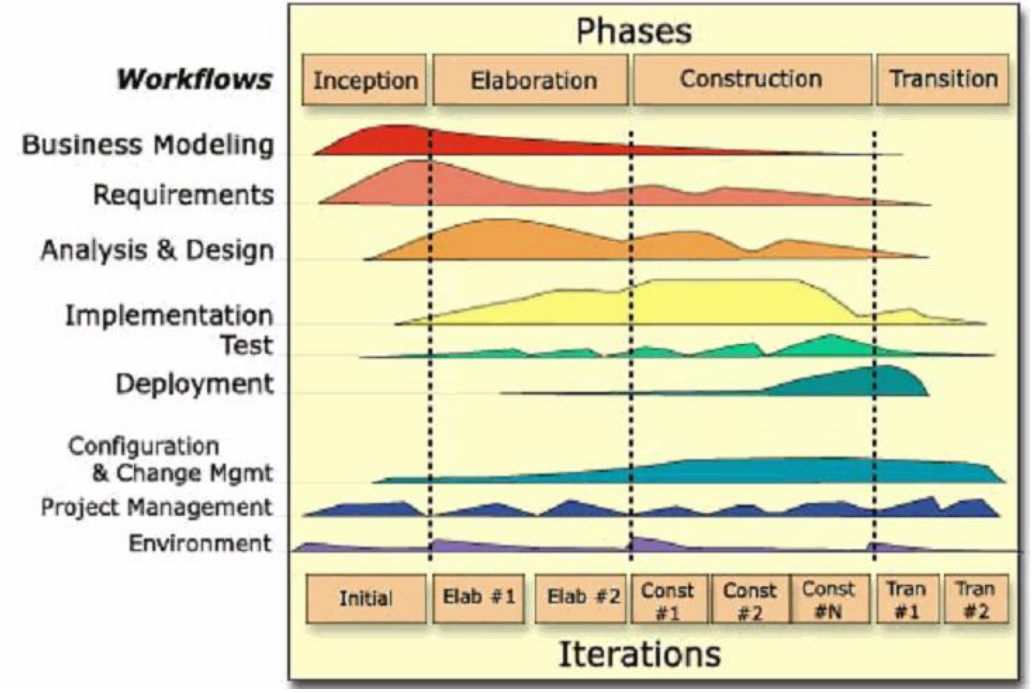
螺旋模型的不足。

- 要求软件开发人员善长风险分析
  - 重大风险没有被开发人员识别
  - 不是风险当成风险
- 风险分析会导致项目终止而终止合同，出现违约诉讼
  - 一般适合内部大系统开发
- 风险分析会增加项目成本
  - 对于小项目，风险分析的成本可能与整个项目的成本相当

风险分析既是螺旋模型的优点也是其缺点

迭代增量式软件开发更适合现代软件系统开发

统一过程（Unified process，简称UP）模型是Rational软件公司（现已被IBM收购）开发的一个基于统一建模语言(Unified Modeling Language，简称UML)的迭代增量式软件开发模型。Rational统一过程（Rational Unified Process，简称RUP）是统一过程模型的一个商业版本



统一过程模型是迭代增量式软件开发模型

统一过程模型各阶段产物。

表 2-1 起始阶段里程碑应满足的条件和应交付的产品

条件	产品
项目涉众就项目目标达成共识	愿景文档（构想），陈述项目的主要需求、特征和约束
项目涉众就系统范围达成共识	初始用例模型（完成 10%~20%）
项目涉众就关键需求达成共识	项目词汇表
项目涉众就成本和进度估算达成共识	初始项目计划
项目管理者提出了一个业务案例	业务案例
项目管理者完成了风险评估	风险评估文档或数据库
通过技术研究和原型化确认了可行性	一个或多个废弃型原型
勾勒了系统体系结构轮廓	初始体系结构文档

起始阶段完成里程碑：主要需求不再变化

统一过程模型各阶段产物。

表 2-2 细化阶段里程碑应满足的条件和应交付的产品

条件	产品
构建了一个有弹性且鲁棒的可执行体系结构基线	可执行体系结构基线
可执行体系结构基线已表明重要的风险已被识别和解决	UML 静态模型、动态模型和用例模型
产品构思已稳定	愿景文档
风险评估已修正	更新的风险评估文档
业务案例已修订，且涉众已就此达成共识	更新的业务案例
项目计划已细化，涉众已就此达成共识，且业务案例已按照项目计划进行了确认	更新的项目计划
涉众同意项目继续	签署文档

细化阶段完成里程碑：体系结构不再变化

统一过程模型各阶段产物。

表 2-3 构造阶段里程碑应满足的条件和应交付的产品

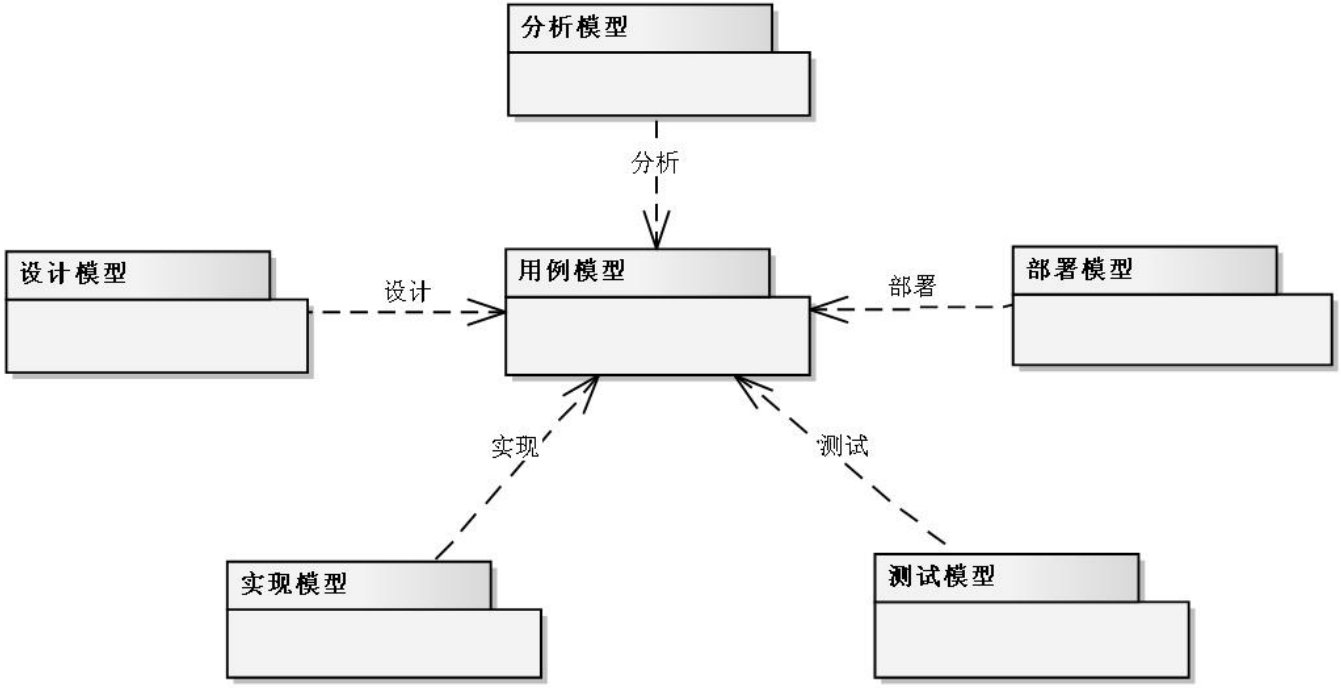
条件	产品
软件产品足够稳定且品质足可以保证部署到用户环境	软件产品、UML 模型、测试用例
涉众已就软件移交到用户环境达成一致，并准备移交	用户手册、发布描述
比照计划费用，项目实际费用是可接受的	项目计划

表 2-4 移交阶段里程碑应满足的条件和应交付的产品

条件	产品
β 测试已完成，缺陷已修复，用户承认系统已成功部署，并积极使用产品	软件产品
已与用户就产品支持策略达成一致，并已实施	用户支持计划 更新的用户手册

构造阶段完成里程碑：主要代码不再变化

统一过程模型各阶段产物。



构造阶段完成里程碑：主要代码不再变化

统一过程模型工作流。

表 2-5 统一过程模型工作流描述

工 作 流	描 述
业务建模	使用业务用例模型和领域模型对系统业务过程建模，包括业务流程、角色以及职责
需求	找出与系统交互的外部参与者，并开发用例模型（UML 用例图）完成对系统需求的建模，即捕获系统应该做什么
分析设计	将软件需求转换成软件系统的设计，创建分析模型（鲁棒分析图）、设计模型（UML 类图）和部署模型（UML 部署图），实现用例模型
实现	依据设计模型创建组件模型，即实现模型（UML 组件图），实现系统中的组件，并将组件合理组织到系统中
测试	与实现工作流紧密相连，发现缺陷并确保在部署之前缺陷已经修复，测试模型用 UML 包图描述，包括测试用例、测试过程、测试组件等
部署	创建并向用户分发软件产品，并安装到用户场所，确保用户可以正常使用产品
配置和变更管理	指导软件工程师确定配置项，限制对这些配置项的变更，审核变更，定义并管理配置项，保证系统的变更是在受控的状态下完成，并已详细记录已实现的变更
项目管理	用于管理系统开发，包括风险管理、项目计划制定以及项目进度监控
环境	为软件开发组织提供软件开发环境，支持项目开发，包括制定项目的开发过程和提供项目合适的软件开发工具

构造阶段完成里程碑：主要代码不再变化

统一过程模型的特点。

- 用例驱动
  - 用例是指系统外部参与者完成的一系列动作
  - 用例确定了开发目标并驱动每次迭代的工作
- 以体系结构为中心
  - 体系结构提供了一种结构来指导迭代过程中的工作。
  - 体系结构有助于项目涉众对新系统有一个宏观的认识
  - 有助于软件重用
- 迭代增量式地进行软件开发
  - 有助于适应需求的变化
  - 能使项目计划更加灵活

统一过程模型是迭代增量式软件开发模型

统一过程模型的不足。

- 太复杂
  - 依据项目需要进行裁剪
  - 但统一过程模型并未给出具体的裁剪实施方法
- 太重
  - 各阶段必须开发大量的制品
  - 开发的大量制品为开发过程服务
  - 不适合资源有限且面临时间压力的软件项目

统一过程模型是迭代增量式软件开发模型

感谢各位聆听！
祝大家学习愉快！