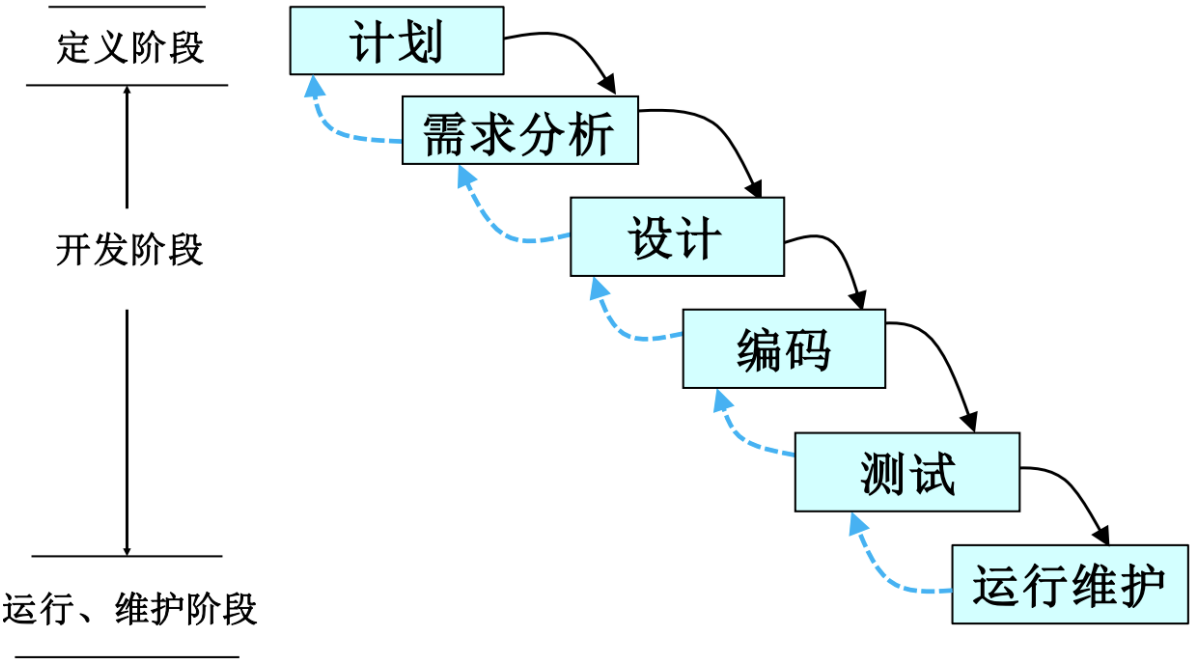
预测型过程模型

> 1. 项目开发活动通常按照固定顺序执行，前一步活动结果是后一步活 动的“蓝图”，前一步对后一步结果的预期约束十分精准
> 2. 需要提前进行大量的计划工作，然后一次性执行，执行过程是一个 连续的过程

常见的预测型过程模型：瀑布模型 (Waterfall)、V模型(V Model)、W模型（W Model）形式化过程 (Formal model)

瀑布模型

> 1. 上一个阶段结束，下一个阶段才能开始
> 2. 每个阶段均有里程碑和提交物
> 3. 上一阶段输出是下一阶段输入
> 4. 每个阶段均需要进行V&V
> 5. 侧重于文档与产出物



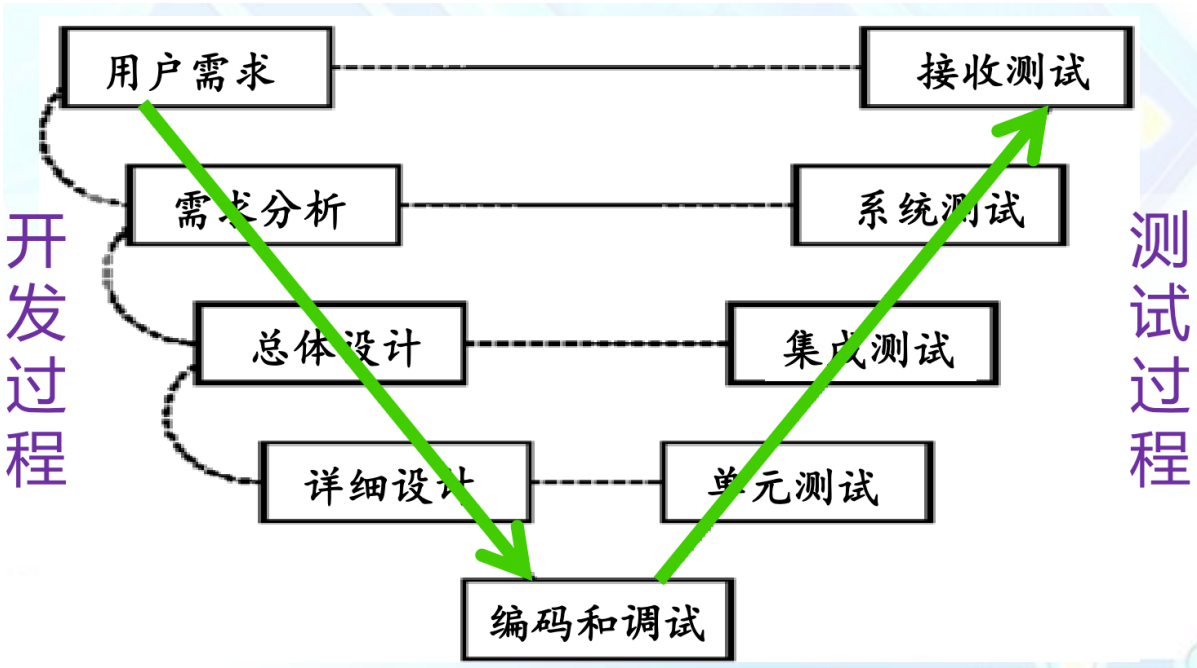
优点:

1. 效率高，简单易用
2. 为项目提供了按阶段划分的检查点，项目管理比较容易
3. 每个阶段必须提供文档，而且要求每个阶段的所有产品必须进行正式、严格的技术审查

缺点:

1. 让用户直接给开发人员一份最终版本的需求十分困难，需求的错误很难在开发后期纠正，因此难以快速响应用户需求变更
2. 开发人员与用户之间缺乏有效的沟通，开发人员的工作几乎完全依赖规格说明文档，容易导致不能满足客户需求
3. 客户必须在项目接近尾声的时候才能得到可执行的程序，对系统中存在的重大缺陷，如果在评审之前没有被发现，将可能会造成重大损失

V模型



相对于瀑布模型的优点：

1. 开发过程重视测试，项目开发成功的几率会更高
2. 避免缺陷向下游流动

迭代过程模型

> 1. 允许对未完成的工作进行反馈，可以进行修改和改进
> 2. 允许对部分已经完成的工作进行反馈，从而进行修改和改进
> 3. 逐步反馈、改进完善，直至开发完成

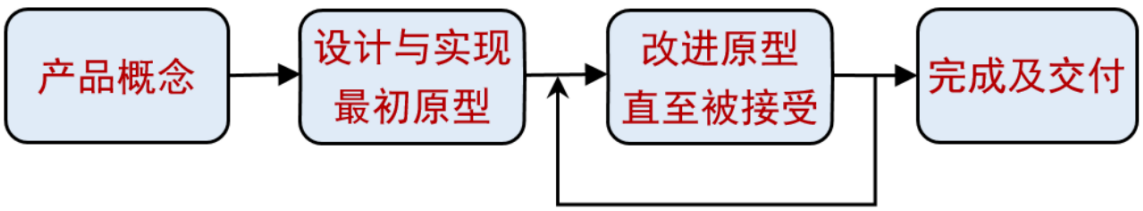
常见的迭代过程模型：螺旋模型 (Spiral)、原型模型 (Prototype)

本质：循环、反复、不断调整当前系统以适应需求变化

快速原型法

步骤：

1. 双方通过沟通，明确已知的需求，并大致勾画出以后再进一步定义的东西
2. 迅速策划一个原型开发迭代并进行建模，主要集中于那些最终用户所能够看到的内容，如人机接口布局或者输出显示格式等
3. 快速设计产生原型，对原型进行部署，由客户和用户进行评价
4. 根据反馈，进一步细化需求并调整原型
5. 原型系统不断调整以逼近用户需求



原型类型：

1. 抛弃式原型，其目的只是为了收集与验证需求，不可执行，并且不会作为最终系统的一部分
2. 演化式原型，可执行，包含了系统的核心功能，最终将成为系统的一部分

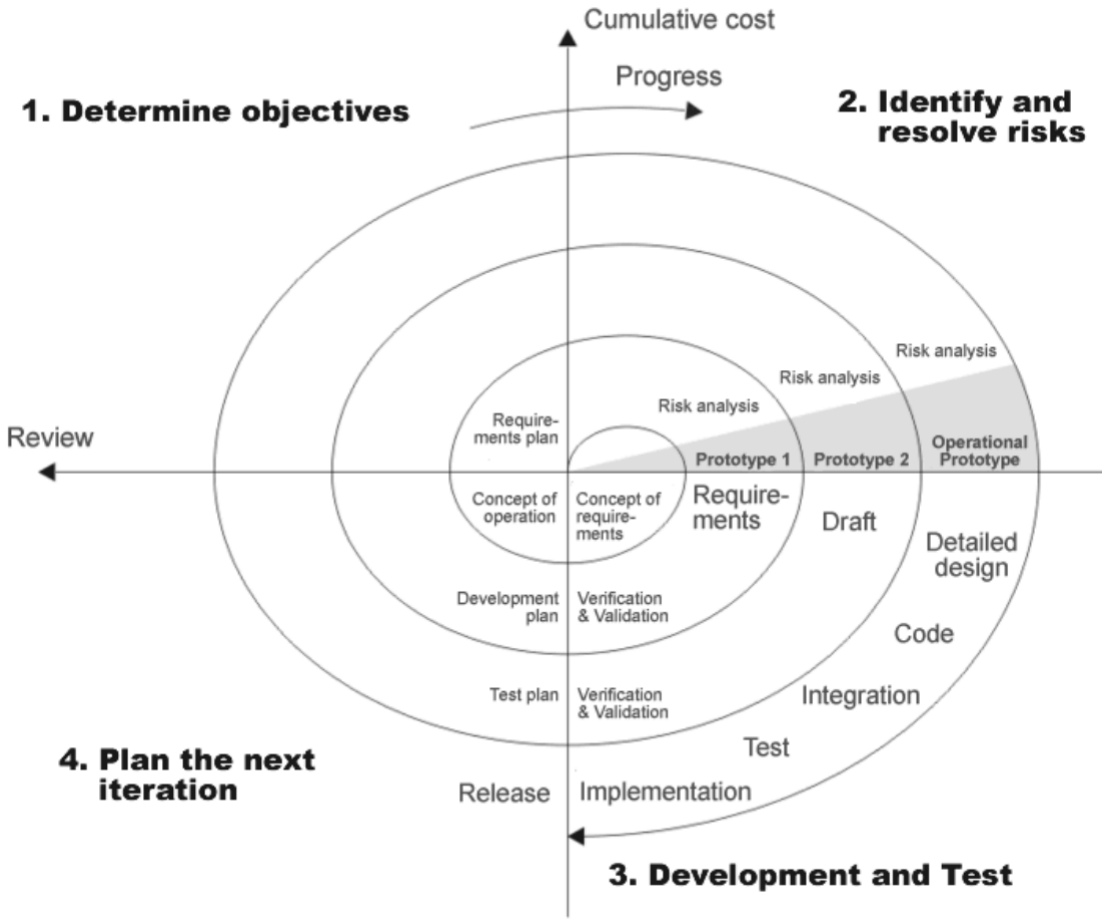
优点：

1. 提高和改善用户的参与程度，最大程度的响应用户需求的变化
2. 克服预测型模型的缺点，减少由于需求不够明确带来的开发风险

缺点：

1. 为了尽快完成原型，开发者没有考虑整体软件的质量和长期的可维护性，系统结构通常较差
2. 原型系统在完全满足用户需求之后 可能会被直接交付给客户使用

螺旋式过程模型



与增量、RAD等的最大 区别在于重视风险评估

螺旋模型沿着螺线旋转，在四个象限内表达四个方面的活动：

1. 制定计划：确定软件目标，选定实施方案，弄清项目开发的限制
2. 风险分析：分析所选方案，考虑如何识别和消除风险
3. 实施工程：实施软件开发
4. 客户评估：评价开发工作，提出修正建议

例如：

1. 第1圈：开发出产品的规格说明

2. 第2圈：开发产品的原型系统
3. 第3~n圈：不断的迭代，开发不同的软件版本 – 根据每圈交付后用户的反馈来调整预算、进度、需要迭代的次数

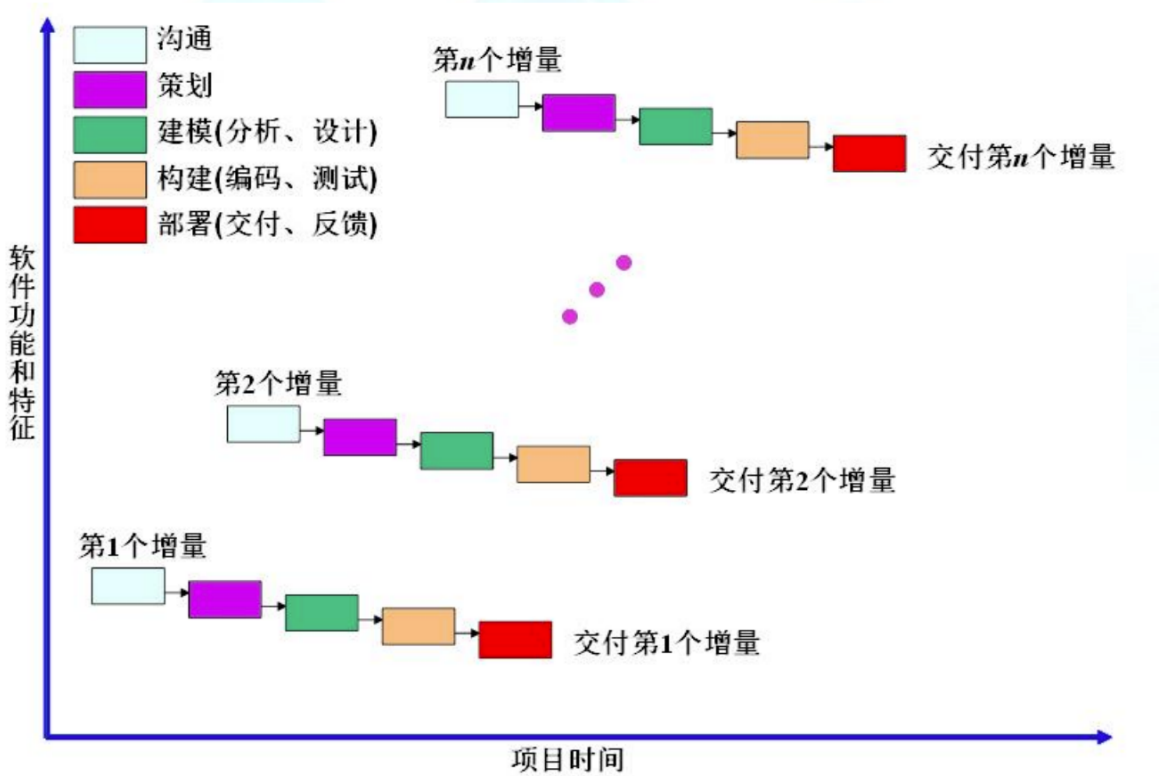
增量过程模型

> 1. 将未来系统分阶段完成，每个阶段都 会产生一个可交付成果
> 2. 每个阶段成果都是一个增量
> 3. 每个增量都是一个独立的开发过程， 包括分析、设计、实现、测试、交付

常见的增量过程模型：增量模型 (Incremental)、快速应用开发(RAD)

增量模型

> 软件被作为一系列的增量来设计、实现、集成和测试，每一个增量是由多个相互作用的模块所形成的特定功能模块



本质：以迭代的方式运用瀑布模型

特点：

1. 第一个增量往往是核心产品
2. 客户使用上一个增量的提交物并进行自我评价，制定下一个增量计划，说明需要增加的特性和功能

优点：

1. 在时间要求较高的情况下交付产品：在各个阶段并不交付一个可运行的完整产品，而是交付满足客户需求的一个子集的可运行产品， 对客户起到“镇静剂”的作用
2. 人员分配灵活：早期的增量由少量人员实现，如果客户反响较好，则在下一个增量中投入更多的人力
3. 逐步增加产品功能可以使用户有较充裕的时间来学习和适应新产品， 避免全新软件可能带来的冲击

4. 因为具有较高优先权的模块被首先交付，而后面的增量也不断被集成进来，这使得最重要的功能肯定接受了最多的测试，从而使得项目总体性失败的风险比较低

使用增量模型时的难点：

1. 每个附加的增量并入现有软件时，必须不破坏原来已构造好的部分
2. 加入新增量时应简单、方便
3. 仍然无法处理需求发生变更的情况
4. 管理人员必须有足够的技术能力来协调好各增量之间的关系

RAD模型

> 侧重于短开发周期（一般为60~90天）的增量过程模型
>
> 多个团队并行进行开发，但启动时间有先后，先启动团队的提交物将作为后启动团队的输入

缺点：

1. 需要大量的人力资源来创建多个相对独立的RAD团队
2. 如果没有在短时间内为急速完成整个系统做好准备，RAD项目将会失败
3. 如果系统不能被合理的模块化，RAD将会带来很多问题
4. 技术风险很高的情况下（采用很多新技术、软件需与其他已有软件 建立集成等等），不宜采用RAD

敏捷过程模型

> 1. 应对变化、增量开发、快速反馈、快速迭代、快速交付
> 2. 基于迭代的敏捷过程：每次迭代，都是针对最重要的系统功能，团队合作开发
> 3. 基于流程的敏捷过程：不是按照迭代计划，而是根据团队能力，领取任务，恒速开发

见：http://t.csdn.cn/yAgdL

总结

一句话归纳各类过程模型

- 瀑布模型：将全部需求以整体方式向前推进，无迭代
  —— 基本模型
- 增量模型：将需求分成多份，串行推进，无迭代
  —— 串行的瀑布
- RAD模型：将需求分成多份，可并行推进，无迭代
  —— 并行的瀑布
- 原型模型：始终结果可见，不断迭代修正原型，直到开发完成
  —— 基本模型
- 螺旋模型：按瀑布阶段划分，各阶段分别迭代(原型+风险分析)
  —— 原型+瀑布
- 敏捷模型：将需求分成尽量小的碎片，以碎片为单位进行高速迭代
  —— 增量+迭代+原型

各类过程模型

适用方式 客观因素	敏捷 (Agile)	计划驱动 (Plan-driven)	形式化的开发方法 (Formal Method)
产品可靠性要求	不高，容忍经常出错	必须有较高可靠性	有极高的可靠性和质量要求
需求变化	经常变化	不经常变化	固定的需求，需求可以建模
团队人员数量	不多	较多	不多
人员经验	有资深程序员带队	以中层技术人员为主	资深专家
公司文化	鼓励变化，行业充满变数	崇尚秩序，按时交付	精益求精
实际例子	写一个微博网站	开发下一版本办公软件；给商业用户开发软件	开发底层正则表达式解析模块；科学计算；复杂系统核心组件
用错方式的后果	用敏捷的方法开发登月火箭控制程序，前N批宇航员都挂了	用敏捷方法，商业用户未必受得了2-4周一次更新的频率	敏捷方法的大部分招数都和这类用户无关，用户关心的是：把可靠性提高到99.99%，不要让微小的错误把系统搞崩溃！