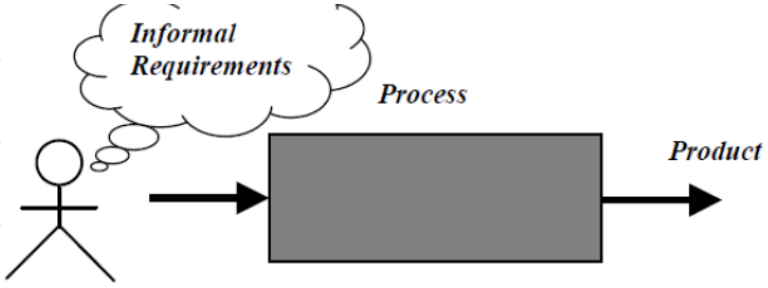
2-1 软件过程模型

2019年4月12日　　23:29

1 软件过程

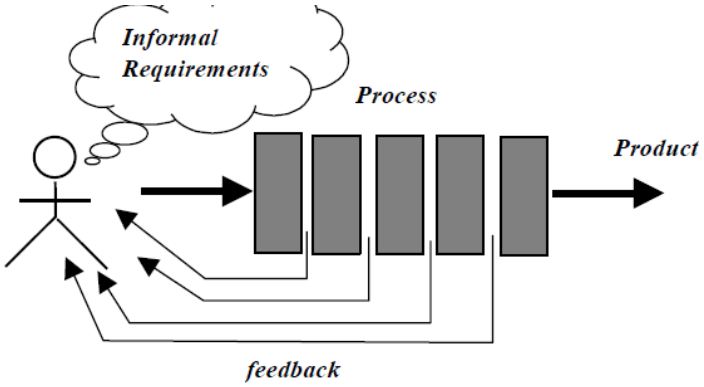
- 软件过程：一个为建造高质量软件所需要完成的活动、动作和任务的框架。
- 软件过程定义以下内容
  - 人员与分工
  - 所执行的活动
  - 活动的细节和步骤
- 软件过程通过以下方式组织和管理软件生命周期
  - 定义软件生产过程中的活动
  - 定义这些活动的顺序及其关系
- 软件过程的目的
  - 标准化(可模仿)、可预见性(降低风险)、提高开发效率、得到高质量产品
  - 提升制定时间和预算计划的能力

黑盒过程

- 

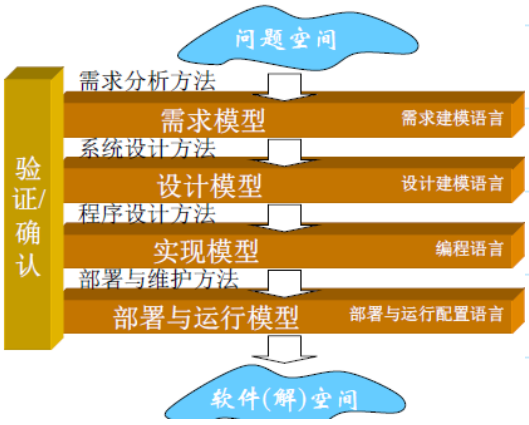
- 存在的问题：
  - 要求开发之前需求被充分理解
  - 与客户的交互只在开始(需求)和最后(发布)——类似于产品制造过程
  - 而实际情况完全不是这样

白盒过程

- 

- 优点：
  - 通过改进可见性来减少风险
  - 在开发过程中，通过不断地获得顾客的回馈允许变更——类似于服务过程

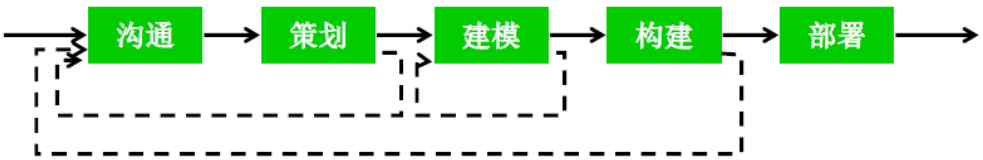
软件过程框架

- 软件通用过程框架
  - 沟通：项目启动、需求获取
  - 策划：项目估算、资源需求、进度计划
  - 建模：创建模型，进行分析和设计
  - 构建：编码和测试
  - 部署：软件交付，支持和反馈
- 软件过程的管理
  - 软件项目跟踪和控制
  - 风险管理
  - 软件质量保证
  - 技术评审
  - 测量
  - 软件配置管理
- 

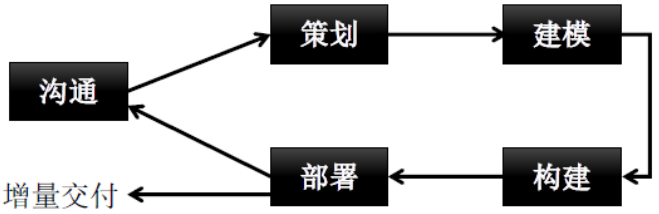
软件开发的过程流

- 线性过程流

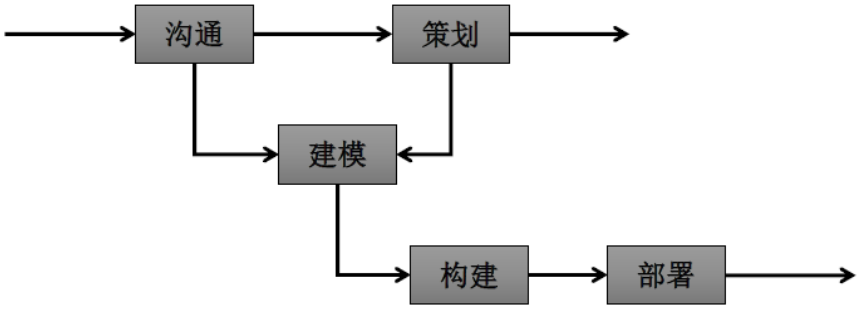
- 迭代过程流



- 演化过程流

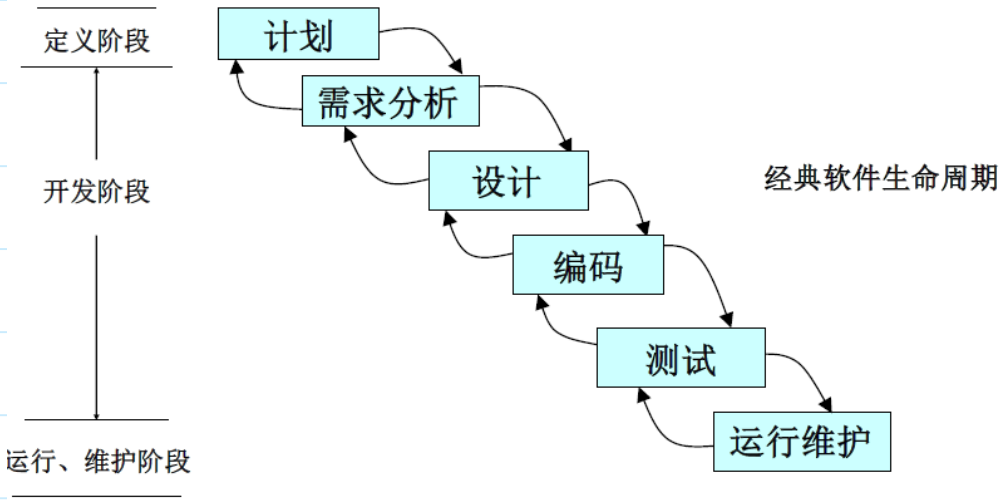


- 并行过程流



2 典型软件过程模型

2.1 瀑布模型(WaterfallModel)

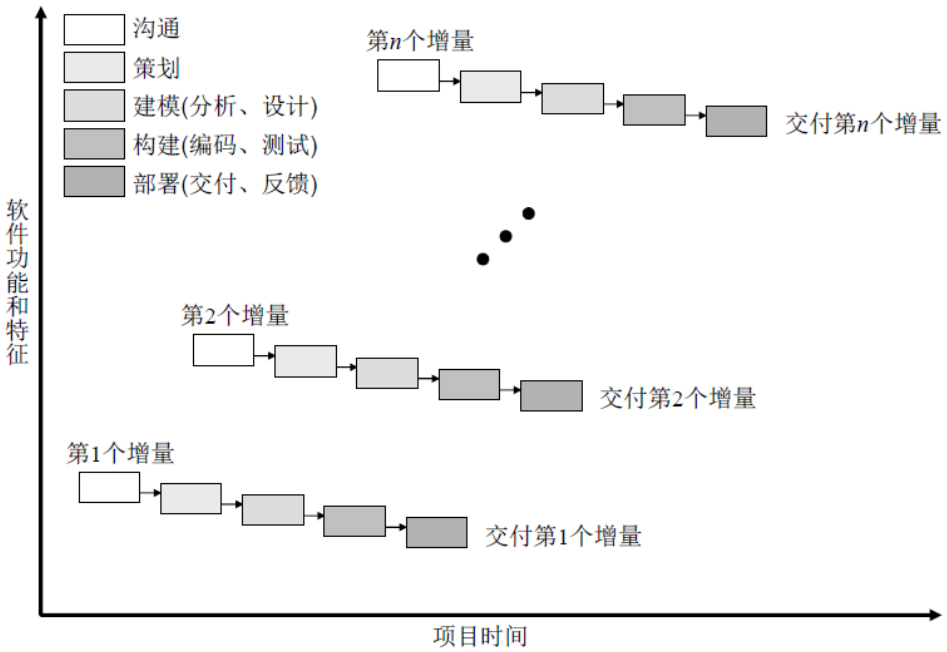


- 基本思想
  - 将软件开发过程划分为计划、分析、设计、编码、测试等阶段
  - 软件开发要遵循过程规律，按次序进行，上一个阶段结束，下一个阶段才能开始
  - 每个阶段均有里程碑和提交物
  - 工作以线性方式进行，上一阶段的输出是下一阶段的输入；
  - 每个阶段均需要进行V&V(Validation&Verification)
- 特点
  - 需求最为重要，假设需求是稳定的
  - 以文档为中心，文档是连接各阶段的关键
- 也叫做鲑鱼模型(Salmonmodel)：向前一阶段回溯，很难。
- 优点——追求效率
  - 简单、易懂、易用、快速；
  - 为项目提供了按阶段划分的检查点，项目管理比较容易；
  - 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——过于理想化
  - 在开发早期，用户难以清楚地确定所有需求，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；
  - 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。
  - 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。
- 瀑布模型带来的问题--文档的问题
  - 文档过于繁杂，占用大量的时间
  - 对于文档的评估需要各领域的专家，文档是否有效?
  - 据统计，一个中型的瀑布过程软件项目有68种文档
  - 当某一文档调整后的影响，不同文档间如何保持一致性?
  - 产品重要还是过程重要？程序重要还是文档重要?
- 瀑布模型太理想化，太单纯，已不再适合现代的软件开发模式，在大型系统开发中已经很少使用；
- 适用场合：
  - 软件项目较小，各模块间接口定义非常清晰；
  - 需求在项目开始之前已经被全面的了解，产品的定义非常稳定；
  - 需求在开发中不太可能发生重大改变；
  - 使用的技术非常成熟，团队成员都很熟悉这些技术
  - 负责各个步骤的子团队分属不同的机构或不同的地理位置，不可能做到频繁的交流
  - 外部环境的不可控因素很少。

2.2 增量过程模型

- 在很多情况下，由于初始需求的不明确，开发过程不宜采用瀑布模型；
- 因此，无须等到所有需求都出来才进行开发，只要某个需求的核心部分出来，即可进行开发；
- 另外，可能迫切需要为用户迅速提供一套功能有限的软件产品，然后在后续版本中再细化和扩展功能。
- 在这种情况下，需要选用增量方式的软件过程模型。
  - 2.2.1 增量模型
  - 2.2.2 RAD模型

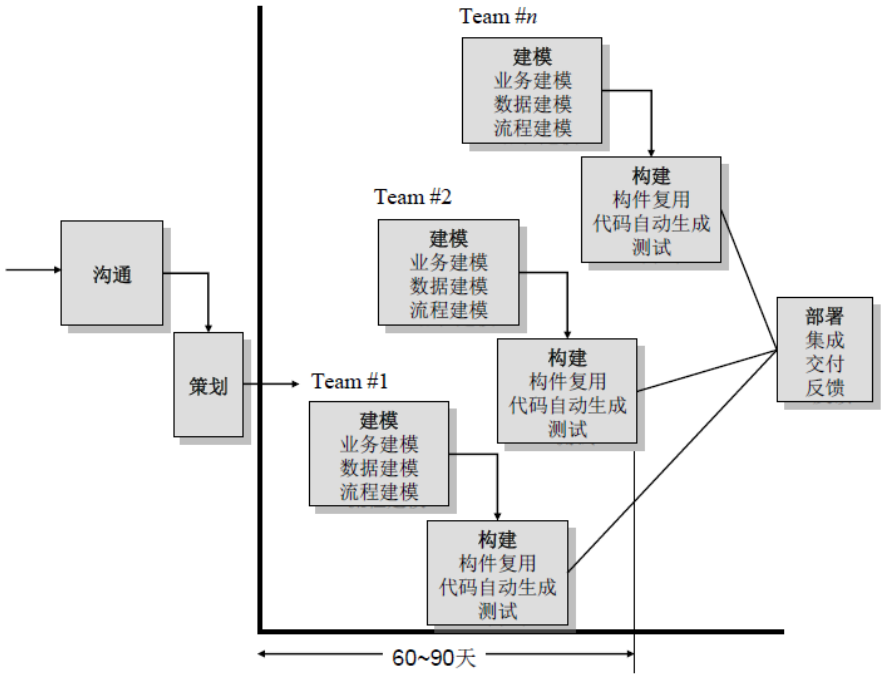
2.2.1 增量模型

- 

- 软件被作为一系列的增量来设计、实现、集成和测试，每一个增量是由多种相互作用的模块所形成的提供功能的代码片段构成。
- 本质：以迭代的方式运用瀑布模型
  - 第一个增量往往是核心产品：满足了基本的需求，但是缺少附加的特性；
  - 客户使用上一个增量的提交物并进行自己评价，制定下一个增量计划，说明需要增加的特性和功能；
  - 重复上述过程，直到最终产品产生为止。
- 举例1：开发一个类似于Word的字处理软件
  - 增量1：提供基本的文件管理、编辑和文档生成功能；
  - 增量2：提供高级的文档编辑功能；
  - 增量3：实现拼写和语法检查功能；
  - 增量4：完成高级的页面排版功能；
- 举例2：开发一个教务管理系统
  - 增量1：提供基本的学籍管理和成绩管理功能；

    - 增量2：提供选课功能；
    - 增量3：提供查询教室使用情况的功能；
    - 增量4：提供课表生成、上课名单生成、成绩录入等功能。
- 优点：
    - 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；
    - 提高对用户需求的响应：用户看到可操作的早期版本后会提出一些建议和需求，可以在后续增量中调整。
    - 人员分配灵活：如果找不到足够的开发人员，可采用增量模型，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；
    - 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；
    - 可规避风险：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。
- 困难：
    - 每个附加的增量并入现有的软件时，必须不破坏原来已构造好的东西
    - 同时，加入新增量时应简单、方便——该类软件的体系结构应当是开放的
    - 仍然无法处理需求发生变更的情况。
    - 管理人员须有足够的技术能力来协调好各增量之间的关系。

2.2.2 RAD模型

- 

- 快速应用开发RAD(RapidApplicationDevelopment)
    - 侧重于短开发周期(一般为60~90天)的增量过程模型，是瀑布模型的高速变体，通过基于构件的构建方法实现快速开发；

    - 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；
- 特点
    - 需求被很好地理解，并且项目的边界也是固定的
    - 沟通用来理解业务问题和软件产品必须具有的特征
    - 策划确保多个软件团队能够并行工作于不同的系统功能
    - 建模包括业务建模、数据建模和过程建模
    - 构建侧重于利用已有的软件构件
    - 快速部署，为迭代建立基础
- 应用举例：依据已有构件开发企业ERP系统（1-3个月）
    - 项目集中调研和规划，确定业务规范
    - 划分项目组，并行建模、构建、测试
        - Team1：供应链管理体系
        - Team2：生产管理体系
        - Team3：质量、设备管理体系
        - Team4：销售管理体系
    - 集成测试及交付
    - 部署与反馈
- 优点：
    - 充分利用企业已有资产进行项目开发
    - 提高软件交付速度
- 问题：
    - 需要大量的人力资源来创建多个相对独立的RAD团队；
    - 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；
    - 如果系统不能被合理的模块化，RAD将会带来很多问题；
    - 如果系统需求是高性能，并且需要通过调整构件接口的方式来提高性能，不能采用RAD模型
    - 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

2.3 演化过程模型

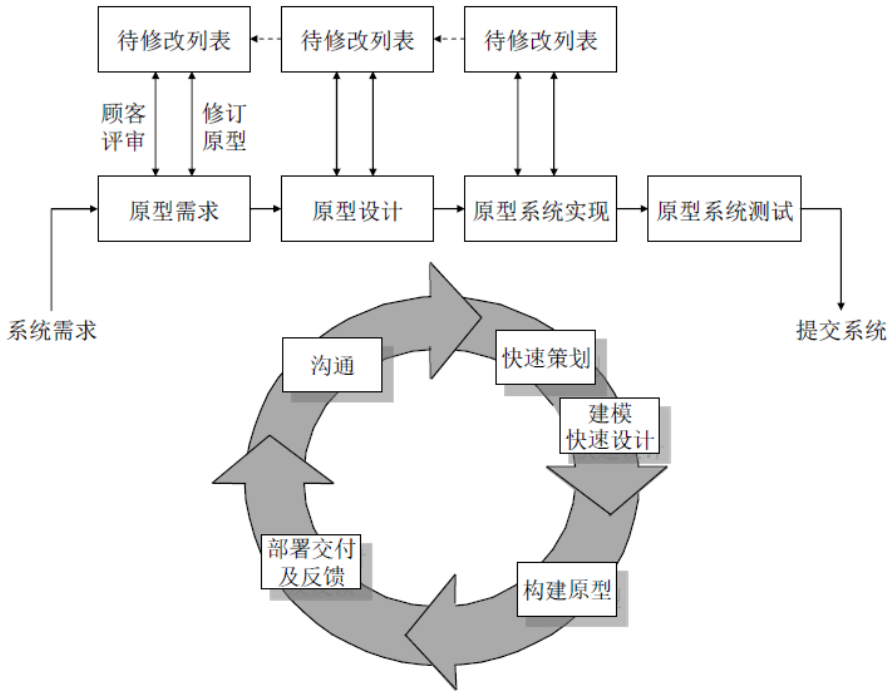
- 软件系统会随着时间的推移而发生变化，在开发过程中，需求经常发生变化，直接导致产品难以实现；
- 严格的交付时间使得开发团队不可能圆满完成软件产品，但是必须交付功能有限的版本以应对竞争或压力；
- 很好的理解和核心产品与系统需求，但对其他扩展的细节问题却没有定义。
- 在上述情况下，需要一种专门应对不断演变的软件过程模型，即“演化过程模型”。
- 本质：循环、反复、不断调整当前系统以适应需求变化；

- 包括两种形态：

  2.3.1 快速原型法

  2.3.2 螺旋模型

2.3.1 快速原型法

- 

- 快速原型法的步骤

  试验性开发，客户提出了软件的一些基本功能，但是没有详细定义其他需求；或者开发人员对于一些技术的使用不确定。

  - Step1：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。
  - Step2：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方面，如人机接口布局或者输出显示格式等；
  - Step3：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
  - Step4：根据反馈，抛弃掉不合适的部分，进一步细化需求并调整原型
  - Step5：原型系统不断调整以逼近用户需求。

- “原型”的类型
  - Throwawayprototyping(抛弃式原型)
    - 最初的原型在完成并得到用户认可之后，将不会作为交付给用户的最终系统的一部分，而是被抛弃，其目的只是为了收集与验证需求；
    - 该类原型可能是不可执行的(例如，只包含用户界面)；
  - Evolutionaryprototyping(演化式原型)
    - 最初构造的原型将具备较高的质量，包含了系统的核心功能，然后通过收集需求对其进行不断的改善和精化；
    - 该类原型是可执行的，将成为最终系统的一部分；

- 快速原型开发应用举例

应用举例：开发一个教务管理系统

- 第一次迭代：完成基本的学籍管理、选课和成绩管理功能；（6周）

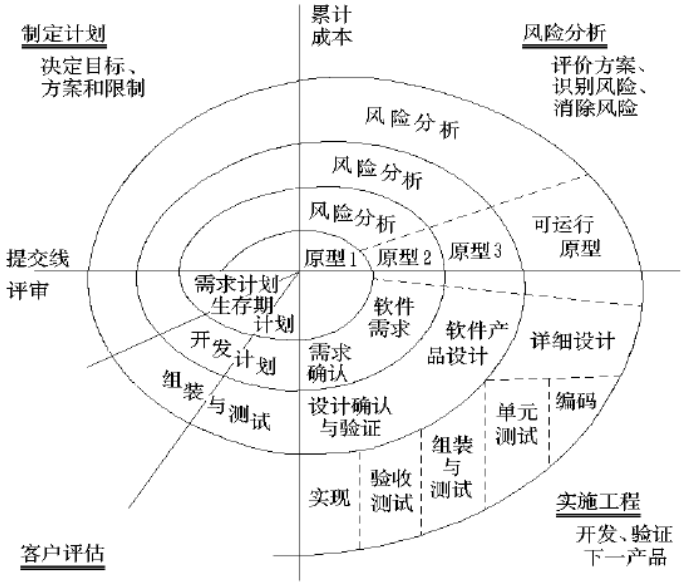
  客户反馈基本满意，但是对大数据量运行速度慢、效率低，不需要学生自己维护学籍的功能等
- 第二次迭代：修改细节，提高成绩统计和报表执行效率（2周）

  客户反馈：需要严格的权限控制，报表打印格式不符合要求
- 第三次迭代：完善打印和权限控制功能；（2周）

  客户反馈：可以进行正式应用验证

- 优点：
  - 快速开发出可以演示的系统，方便与客户沟通，提高和改善客户/用户的参与程度；
  - 采用迭代技术能够使开发者逐步弄清客户的需求，最大程度的响应用户需求的变化；
- 问题：
  - 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；
  - 用户可能混淆原型系统与最终系统，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；
  - 额外的开发费用。

2.3.2 螺旋式过程模型

- 

- 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：
  - 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；
  - 风险分析：分析所选方案，考虑如何识别和消除风险；
  - 实施工程：实施软件开发；
  - 客户评估：评价开发工作，提出修正建议。
- 举例：
  - 第1圈：开发出产品的规格说明；
  - 第2圈：开发产品的原型系统；

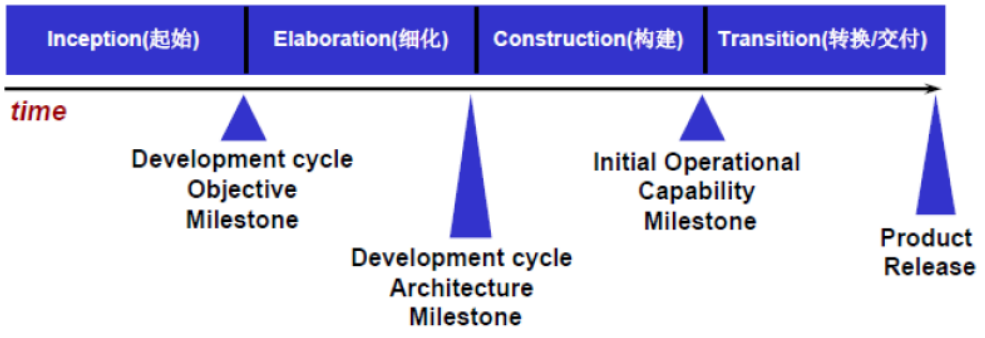
    - 第3~n圈：不断的迭代，开发不同的软件版本；
    - 根据每圈交付后用户的反馈来调整预算、进度、需要迭代的次数；
- 出发点：开发过程中及时识别和分析风险，并采取适当措施以消除或减少风险来的危害。
- 优点：结合了原型的迭代性质与瀑布模型的系统性和可控性，是一种风险驱动型的过程模型：
    - 采用循环的方式逐步加深系统定义和实现的深度，同时更好的理解、应对和降低风险；
    - 确定一系列里程碑，确保各方都得到可行的系统解决方案；
    - 始终保持可操作性，直到软件生命周期的结束；
    - 由风险驱动，支持现有软件的复用。
- 问题：
    - 适用于大规模软件项目，特别是内部项目，周期长、成本高；
    - 软件开发人员应该擅长寻找可能的风险，准确的分析风险，否则将会带来更大的风险。

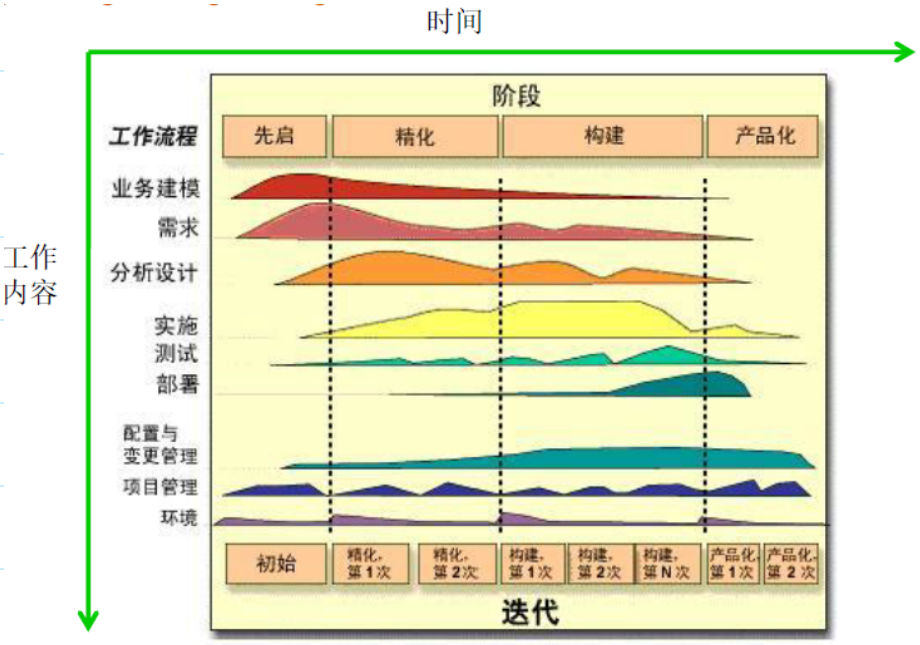
总结：演化过程模型的缺点

- 演化过程模型的目的：
    - 需求的变更频繁，要求在非常短的期限内实现，以充分满足客户/用户要求、及时投入市场；
- 存在的问题：
    - 由于构建产品所需的周期数据不确定，给项目管理带来困难；
    - 演化速度太快，项目陷入混乱；演化速度太慢，影响生产率；
    - 为追求软件的高质量而牺牲了开发速度、灵活性和可扩展性；

2.4 (*仅了解)统一过程模型(RUP)(RationalUnifiedProcess)

是一种面向对象(OO)的软件开发方法论；

- 



- RUP的四个技术阶段--每个阶段多次迭代，一次迭代是一个瀑布过程
  - 起始(Inception)
    - 通过沟通与策划，定义项目范围、识别业务需求、提出大致的架构、制定开发计划，使用UseCase(用例)描述主要特征功能；
  - 细化(Elaboration)
    - 扩展体系结构及用例，从五个角度来构建软件模型(用例模型、需求模型、设计模型、实现模型和部署模型)，并建立一个baseline(基线)；评审与修订项目计划；
  - 构建(Construction)
    - 将体系结构模型作为输入，完善上一阶段的分析和设计模型；
    - 开发或获取软构件，并对每个构件实施单元测试；
    - 构件组装和集成测试；
  - 转换与交付(Transition)
    - 软件被提交给用户进行测试，接收用户反馈报告并进行变更；
    - 创建必要的支持信息(如用户手册、安装步骤、FAQ等)；
- RUP核心工作流程
  - 6个过程工作流
    - 业务建模
    - 需求
    - 分析设计
    - 实现
    - 测试
    - 部署
  - 3个支持工作流
    - 配置和变更管理
    - 项目管理

        - 环境
- RUP特征
    - 符合最佳实践
        - 迭代开发
        - 需求管理
        - 架构和构件的使用
        - 建模和UML
        - 过程质量和产品质量
        - 配置管理和变更管理
    - 其它特征
        - 用例驱动的开发
        - 过程配置
        - 工具支持
- RUP的适用范围--大规模软件研发
    - 规模
        - 大规模的软件研发（50人以上）
        - 职责明确的小组划分或分布式团队
        - 大项目特点：高度管理复杂性，高度技术复杂性
    - 项目持续时间
        - 软件研发过程复杂，时间1年以上
    - 特点
        - 规范、严格的软件开发过程
        - 以文档为中心，因为各阶段的里程碑是文档提交物
        - 重点在过程管理与提升
    - RUP可裁剪适应小规模的软件开发过程