SOFTWARE TESTING

Chapter 1

Overview of Software Engineering

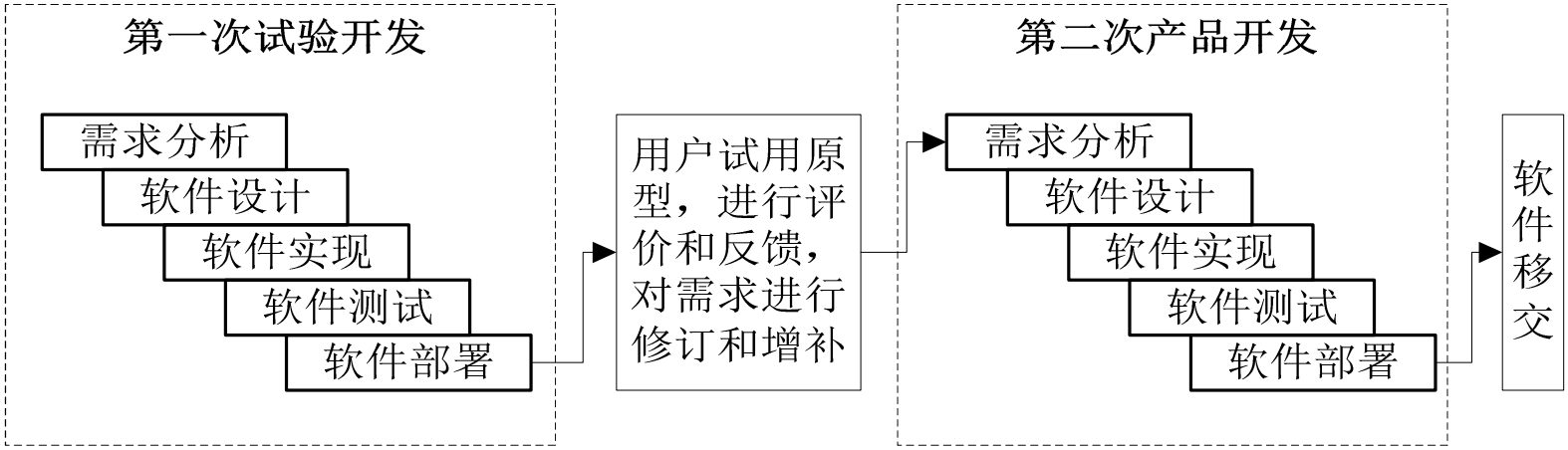
School of Data & Computer Science
Sun Yat-sen University

Approaches & Technologies

OUTLINE

演化模型

- Evolutionary Model
- 演化模型提倡两次开发
  - 第一次试验开发：得到试验性的原型产品，目标在于探索可行性，确定软件需求。
  - 第二次产品开发：在原型产品的基础上获得最终软件产品。

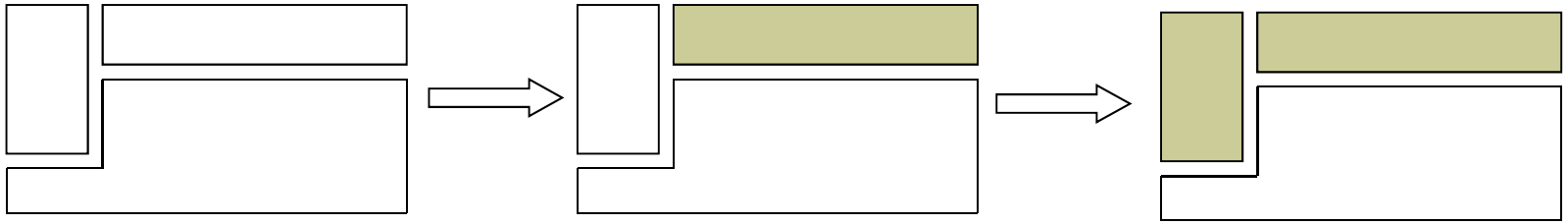


- 演化模型
  - 演化模型分类：
    - 探索式演化模型
    - 抛弃式演化模型
  - 演化模型的特点：
    - 优点：明确用户需求，提高系统质量，降低开发风险。
    - 缺点：结构性不佳，管理上有难度。
  - 演化模型适用范围：
    - 初始需求不清晰
    - 开发周期短
    - 目标系统小型或中小型规模

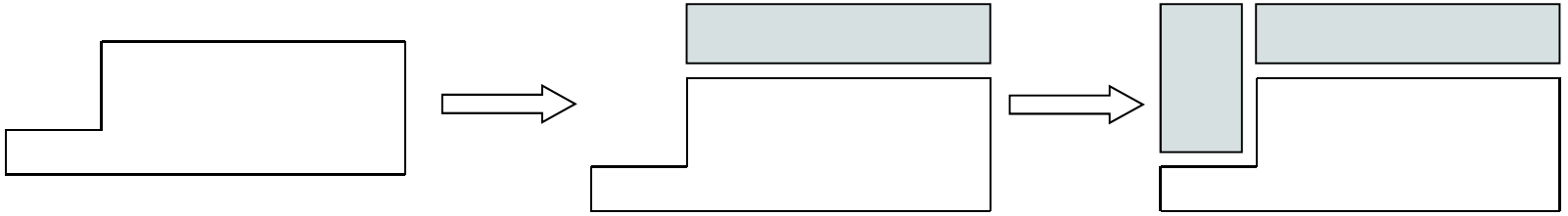
迭代和增量模型

- Iterative and Incremental Model, *Mills* 1980.

迭代开发

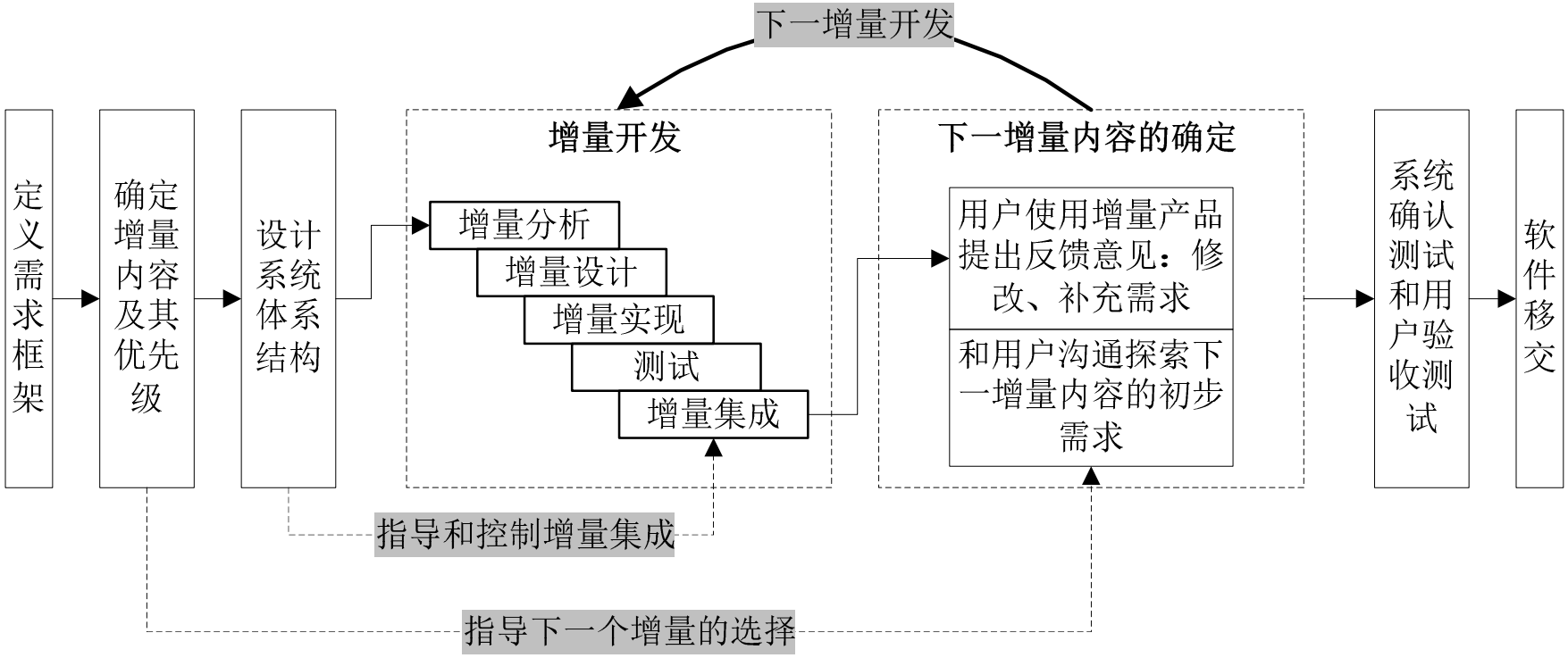


增量开发



迭代和增量模型

- 增量模型首先对系统最核心或最清晰的需求进行分析、设计、实现、测试并集成到系统中，再按优先级逐步实现后续需求。



- 迭代和增量模型
  - 增量模型的例
    - 使用增量模型开发字处理软件，可以按照以下优先级进行增量开发：
      - 增量1：实现基本的文件管理、编辑和文档生成功能；
      - 增量2：实现更加完善的编辑和文档生成功能；
      - 增量3：实现拼写和文法检查功能；
      - 增量4：完成高级的页面布局功能；
      - 。。。。。。

- 迭代和增量模型
  - 增量模型的优点
    - 增强客户对系统的信心；
    - 降低系统失败风险；
    - 提高系统可靠性；
    - 提高系统的稳定性和可维护性。
  - 增量模型的缺点
    - 建立初始模型时，作为增量基础的基本业务服务的确定有一定难度；
    - 增量粒度难以选择。

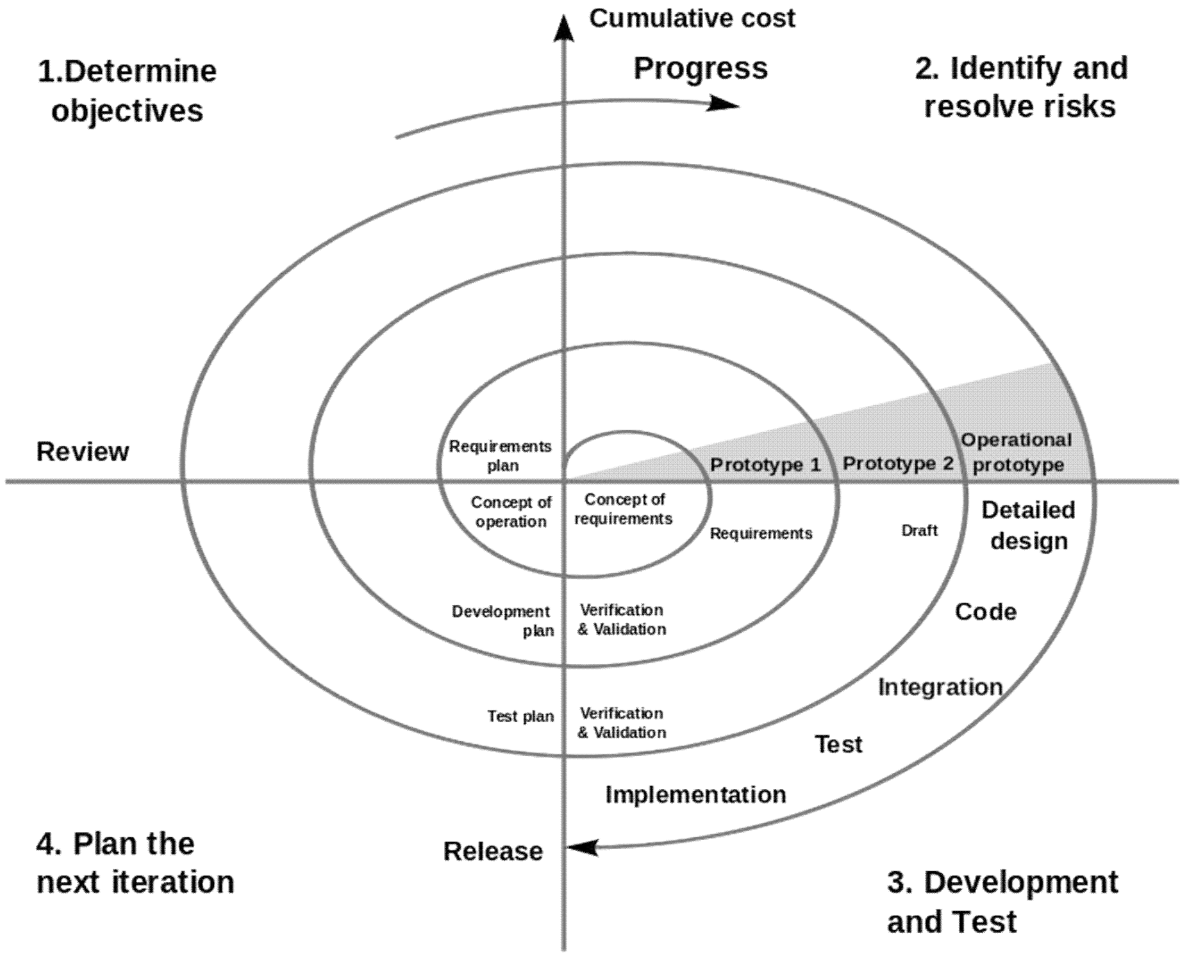
螺旋模型

- Spiral Model, *Barry W. Boehm* 1988.
  - Spiral Model combines some key aspect of the waterfall model and rapid prototyping methodologies, in an effort to combine advantages of top-down and bottom-up concepts. It provided emphasis in deliberate iterative risk analysis, particularly suited to large-scale complex systems.

软件生命周期模型

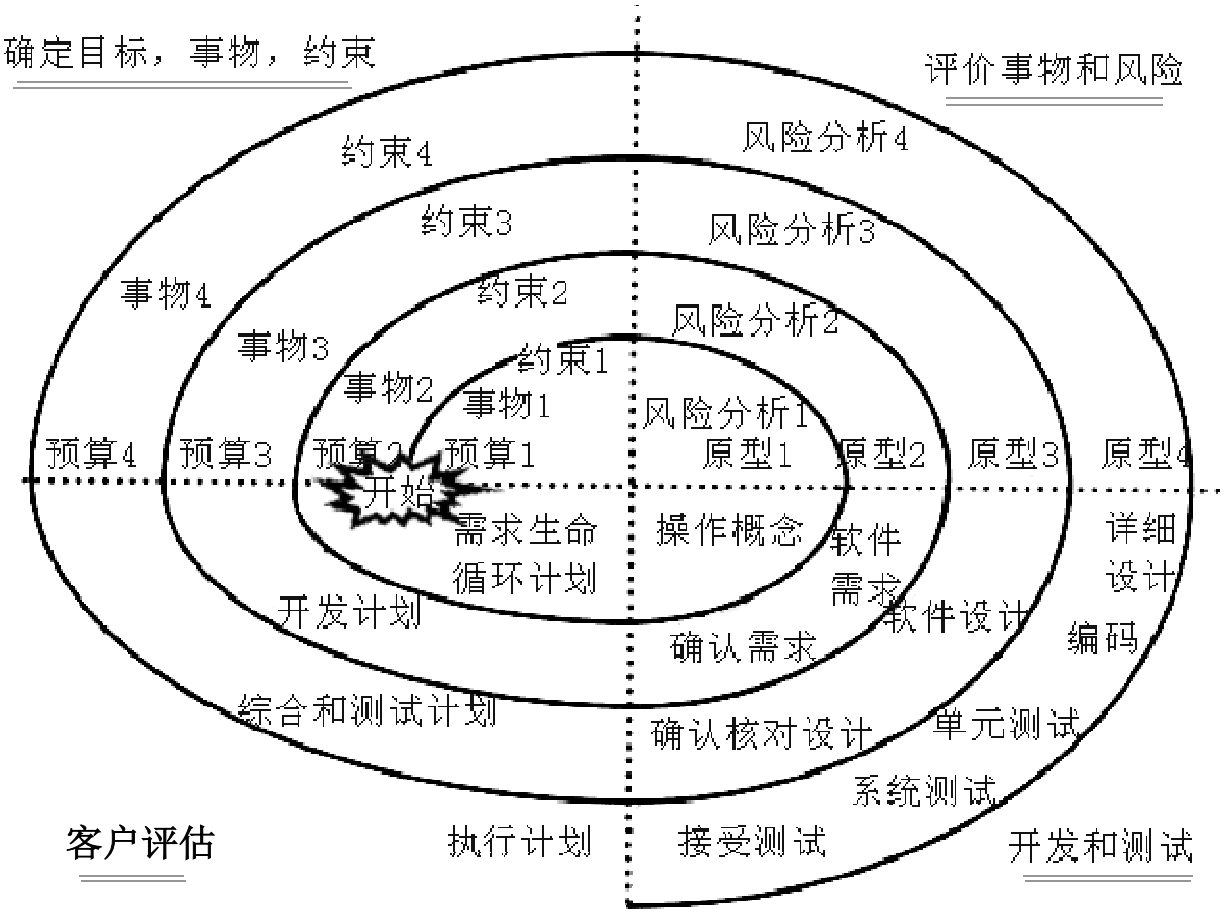
螺旋模型

- Spiral Model, *Barry W. Boehm* 1988.



螺旋模型

- Spiral Model, *Barry W. Boehm* 1988.



螺旋模型

- Basic Concepts of Spiral Model
  - Focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle.
  - ″Each cycle involves a progression through the same sequence of steps, for each part of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program.″

螺旋模型

- Basic Concepts of Spiral Model
  - Each trip around the spiral traverses four basic quadrants:
    (1) Determine objectives, alternatives, and constraints of the iteration;
    (2) Evaluate alternatives; Identify and resolve risks;
    (3) Develop and verify deliverables from the iteration; and
    (4) Plan the next iteration.
  - Begin each cycle with an identification of stakeholders and their ″win conditions″ (objectives and constraints), and end each cycle with review and commitment.

螺旋模型

- *B. W. Boehm* lists the six assumptions (the invariant) as follows:
  (1) The requirements are known in advance of implementation.
  (2) The requirements have no unresolved, high-risk implications, such as risks due to cost, schedule, performance, safety, security, user interfaces, organizational impacts, etc.
  (3) The nature of the requirements will not change very much during development or evolution.
  (4) The requirements are compatible with all the key system stakeholders' expectations, including users, customer, developers, maintainers, and investors.
  (5) The right architecture for implementing the requirements is well understood.
  (6) There is enough calendar time to proceed sequentially.
  - In the situations that these assumptions of the waterfall model do not apply, it is a project risk not to specify the requirements and proceed sequentially, or the incremental waterfall model would become a risk-driven special case of the spiral model.

螺旋模型

- Four activities that must occur in each cycle of the spiral model:
  (1) Consider the win conditions of all success-critical stakeholders.
  (2) Identify and evaluate alternative approaches for satisfying the win conditions.
  (3) Identify and resolve risks that stem from the selected approach(es).
  (4) Obtain approval from all success-critical stakeholders, plus commitment to pursue the next cycle.
- Project cycles that omit or shortchange any of these activities risk wasting effort by pursuing options that are unacceptable to key stakeholders, or are too risky.
- Some "hazardous spiral look-alike" processes violate the invariant (claimed as the six assumptions) by excluding key stakeholders from certain sequential phases or cycles. For example, system maintainers and administrators might not be invited to participate in definition and development of the system. As a result, the system is at risk of failing to satisfy their win conditions.

螺旋模型

- Three anchor point milestones that serve as progress indicators and points of commitment. They can be characterized by key questions.
  (1) Life Cycle Objectives: Is there a sufficient definition of a technical and management approach to satisfying everyone's win conditions?
  (2) Life Cycle Architecture: Is there a sufficient definition of the preferred approach to satisfying everyone‘s win conditions, and are all significant risks eliminated or mitigated (缓解)?
  (3) Initial Operational Capability: Is there sufficient preparation of the software, site, users, operators, and maintainers to satisfy everyone's win conditions by launching the system?
- If the stakeholders agree that the answer is "Yes", then the project has cleared the LCO, LCA or IOC milestone. Otherwise, the project can be abandoned, or the stakeholders can commit to another cycle to try to get to "Yes."

螺旋模型

- 螺旋模型主要针对大型软件项目的开发
  - 大型软件项目的特点：
    - 需求功能复杂：可能无法从项目开始就得到明确的规格；
    - 开发周期长：用户需求在中途经常发生变化。
  - 大型软件项目的风险管理：大型项目存在诸多风险因素，在不同程度上对软件开发过程和软件产品质量造成影响。风险不能全部消除，而只能采用避免、减轻、和接受三种应对策略。
    - 需求变更风险；
    - 进度风险、预算风险、管理能力风险、信息安全风险；
    - 应用技术风险、质量控制风险、软件设计与开发工具风险、员工技能风险；
    - 人力资源风险、政策风险、市场风险、营销风险。
  - 螺旋模型的最大特点就是引入了明确的风险管理机制。

螺旋模型

- 螺旋模型的四个象限：
  - 确定目标：确定软件项目目标；明确对软件开发过程和软件产品的约束；制定详细的项目管理计划；根据当前的需求和风险因素，制定实施方案，并进行可行性分析，选定一个实施方案，并对其进行规划。
  - 识别和解决风险：明确每一个项目风险，估计风险发生的可能性、频率、损害程度，并制定风险管理措施规避这些风险。
  - 工程实现：针对每一个阶段的任务要求执行开发和测试活动。
  - 准备下一轮迭代：客户使用原型，反馈修改意见；根据客户的反馈，对产品及其开发过程进行评审，决定是否进入螺旋线的下一个回路。

螺旋模型

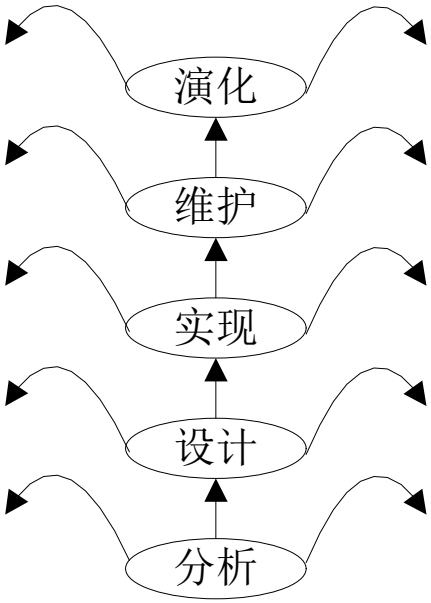
- 螺旋模型的特点：
  - 螺旋模型是风险驱动的迭代过程，强调可选方案和约束条件从而支持软件的重用，有助于将软件质量作为特殊目标融入产品开发之中。螺旋模型结合了瀑布模型和快速原型方法，将瀑布模型的多个阶段转化到多个迭代过程中，以降低项目的风险。
  - 螺旋模型的每一次迭代都包含了以下六个步骤：
    - 决定目标、替代方案和约束条件；
    - 识别和解决项目的风险；
    - 评估技术方案和替代方案；
    - 开发本次迭代的交付物，并验证迭代产出的正确性；
    - 计划下一次迭代；
    - 提交下一次迭代的步骤和方案。

螺旋模型

- 螺旋模型的特点：(续)
  - 螺旋模型区别于 RUP 的迭代模型。
    - 螺旋模型的每一次迭代只包含了瀑布模型的某一个或两个阶段。
    - RUP 的每一次迭代都会包含需求、设计、开发和测试等各个阶段的活动，RUP 迭代的目的在于逐步求精。
  - 螺旋模型的应用有一定的限制条件。
    - 螺旋模型强调风险分析，但说服外部客户接受和相信分析结果并做出相关反应并不容易，因此螺旋模型往往比较适合内部的大规模软件开发。
    - 风险分析需要耗费相当的成本，因此螺旋模型比较适合投资规模较大的软件项目。
    - 失误的风险分析可能带来更大的风险。

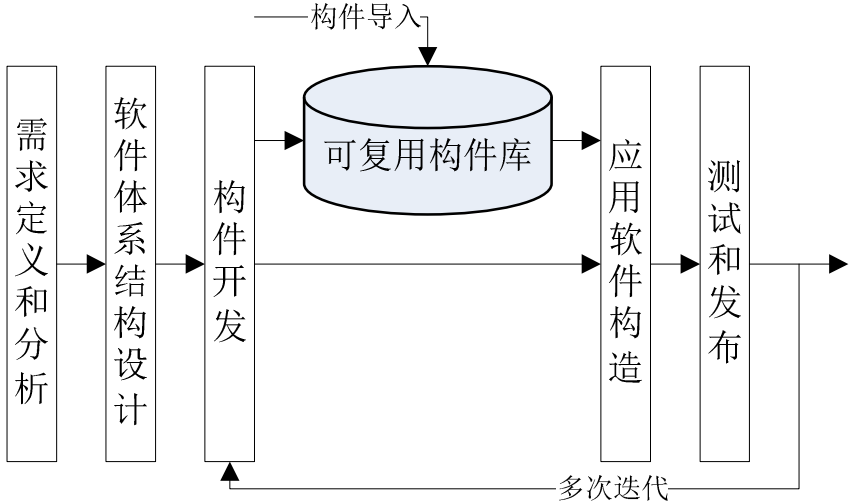
喷泉模型

- Fountain Model
- 喷泉模型的特点
  - 喷泉模型是一种迭代模型 (Iterative Model)。软件开发过程的各个阶段是相互重叠和多次反复的，就象喷泉一样，水喷上去又可以落下来，既可以落在中间，又可以落到底部。
  - 喷泉模型的各个开发阶段没有特定的次序要求，可以并行进行，可以在某个开发阶段中随时补充其他任何开发阶段中遗漏的需求。
  - 优点：
    - 提高开发效率
    - 缩短开发周期
  - 缺点：
    - 管理有一定难度



构件组装模型

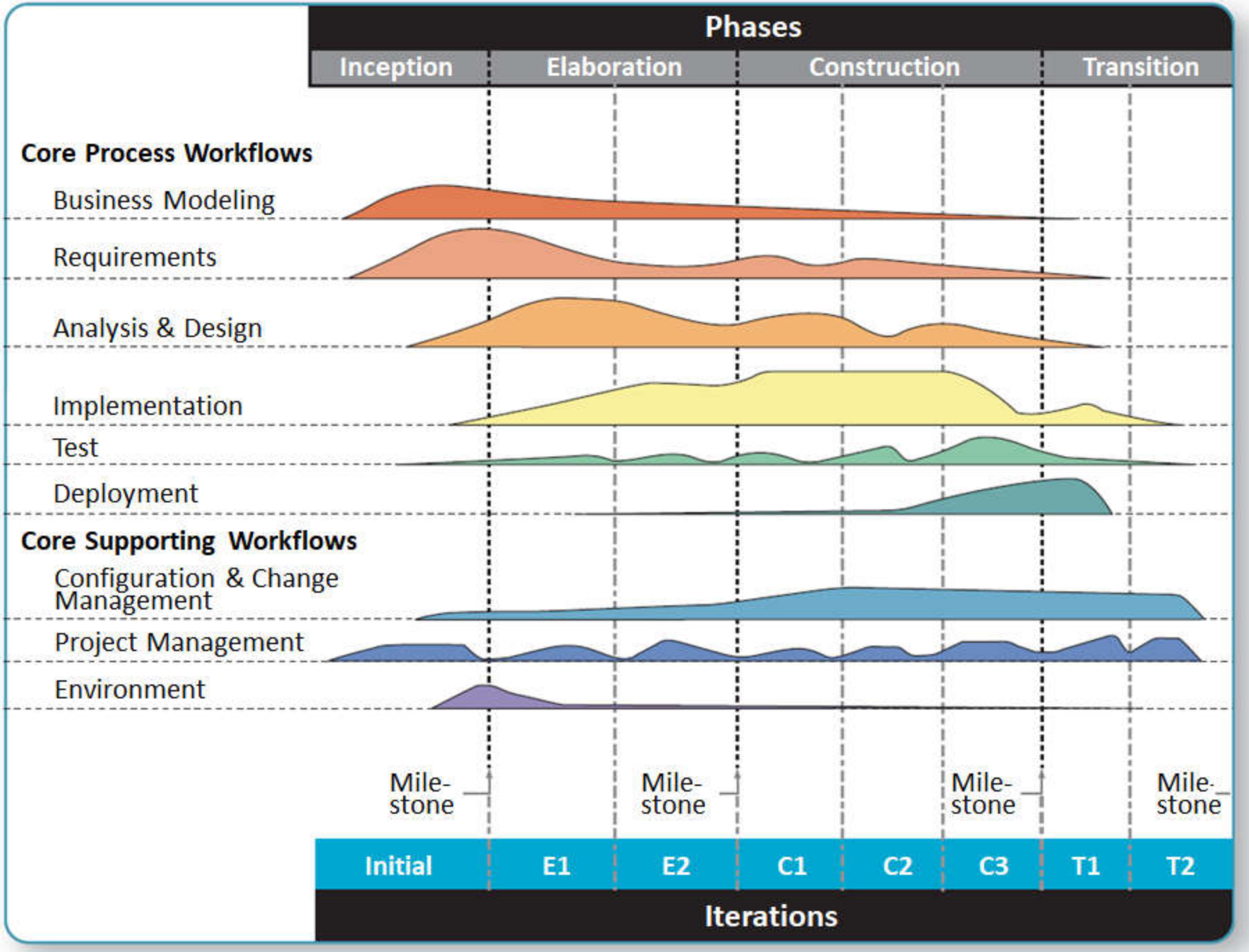
- CBSD, Component-Based Software Development Model
- 构件组装模型的概念
  - 构件组装模型利用模块化思想将整个系统模块化，并在一定构件模型的支持下复用构件库中的一个或多个软件构件，通过组装过程高效率、高质量地构造软件系统。构件组装模型本质上是演化的，开发过程是迭代的。
  - 构件组装模型的开发过程就是构件组装的过程，维护的过程就是构件升级、替换和扩充的过程。

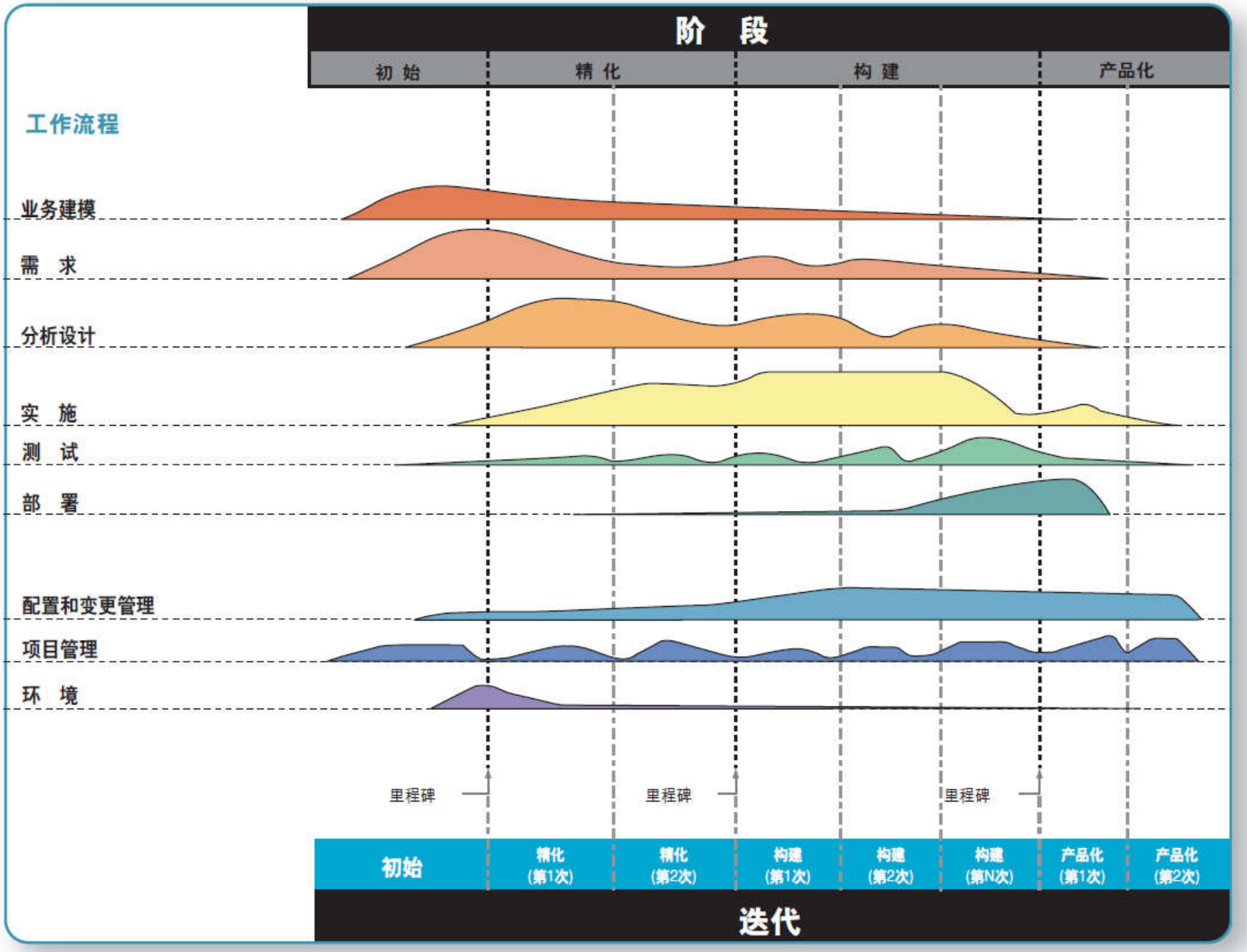


- 构件组装模型
  - 优点：
    - 充分利用软件复用，提高软件开发效率；
    - 允许多个项目同时开发，降低费用，提高可维护性，可实现分步提交软件产品。
  - 缺点：
    - 缺乏通用的构件组装结构标准，因此带来较大的开发风险；
    - 构件可重用性和系统高效性之间不易协调；
    - 由于过分依赖构件，构件的质量影响着最终产品的质量。

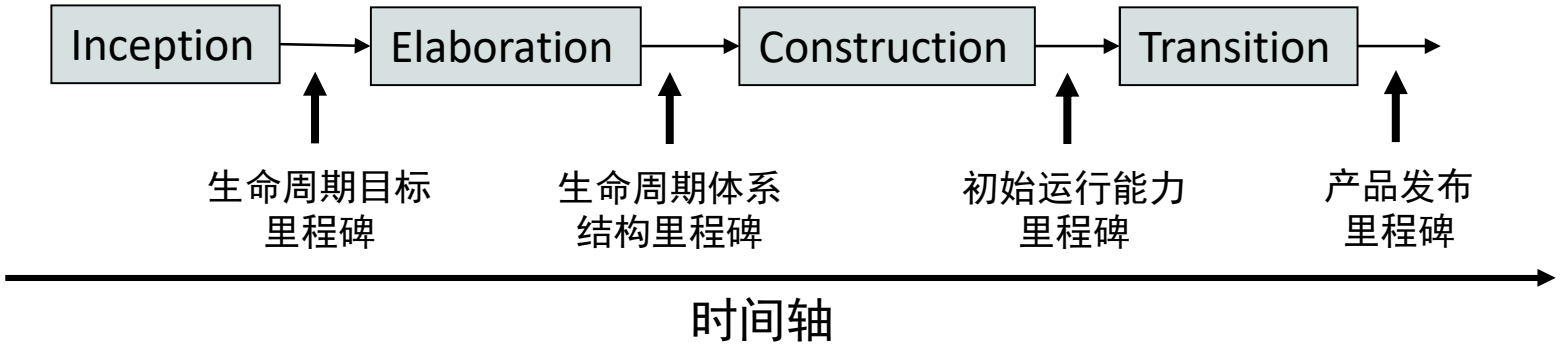
- RUP 统一过程模型
  - RUP 模型的特点
    - RUP (Rational Unified Process) 是一种基于 UML 的、以构架为中心、用例驱动与风险驱动相结合的迭代增量过程。它将软件开发过程要素和软件工件要素整合在统一的软件工程框架中，是一个面向对象的程序开发方法论。
    - *Ivar Jacobson*
      - 1987年开发出 Objectory (Object System Co.)，1991年被爱立信收购，1995年 Objectory 被 Rational 公司收购。
      - 1997年创建 UML 语言 (with *G. Booch* & *J. Rombaugh*)，Objectory 进化为 RUP。
      - 2003年 Rational 公司被 IBM 收购。
    - http://www.ibm.com/developerworks/cn/rational/

- RUP 统一过程模型
  - RUP 的基本结构
    - 传统的瀑布模型是一个单维时间顺序模型，开发工作被划分为多个连续的阶段。在一个时间段内，只能实施某一个阶段的工作比如分析、设计或者实现。
    - RUP 中，软件开发生命周期根据时间和 RUP 的核心工作流程划分为二维空间。
      - 时间维度从组织管理的角度描述整个软件开发生命周期，是 RUP 的动态组成部分，它可进一步描述为周期 (Cycle)、阶段 (Phase) 和迭代 (Iteration)。
      - 核心工作流程维度从技术角度描述 RUP 的静态组成部分，它可进一步描述为工作流 (Workflow)、角色 (Worker)、行为 (Activities) 和产品/工件 (Artifact)。


Phases
Inception
Elaboration
Construction
Transition
Core Process Workflows
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Core Supporting Workflows
Configuration & Change Management
Project Management
Environment
Mile-stone
Mile-stone
Mile-stone
Mile-stone
Initial
E1
E2
C1
C2
C3
T1
T2
Iterations


阶 段
初 始
精 化
构 建
产品化
工作流程
业务建模
需 求
分析设计
实 施
测 试
部 署
配置和变更管理
项目管理
环 境
里程碑
里程碑
里程碑
初始
精化 (第1次)
精化 (第2次)
构建 (第1次)
构建 (第2次)
构建 (第N次)
产品化 (第1次)
产品化 (第2次)
迭代

- RUP 统一过程模型
  - RUP 的时间维度结构
    - RUP 中的软件生命周期在时间维度上被分解为四个顺序的阶段：初始阶段 (Inception)、精化阶段 (Elaboration)、构建阶段 (Construction) 和产品交付阶段 (Transition)。每个阶段结束于一个主要的里程碑 (Major Milestone)，并在阶段结尾执行一次评估以确定这个阶段的目标是否已经满足。如果评估结果令人满意的话，可以允许项目进入下一个阶段。



- RUP 统一过程模型
  - RUP 的时间维度结构
    - 初始阶段
      - 目标：为系统建立业务案例 (Business Case) 并确定项目的边界。业务案例包括项目的验收规范、风险评估、所需资源估计、阶段计划等。确定项目边界需要识别所有与系统交互的外部实体，并在较高层次上定义外部实体与系统交互的特性，主要包括识别外部角色 (Actor)、识别所有用例并详细描述一些重要的用例。
      - 里程碑：生命周期目标 (Lifecycle Objective) 里程碑，包括一些重要的文档，如：项目构想 (Vision)、原始用例模型、原始业务风险评估、一个或者多个原型、原始业务案例等。通过对文档的评审确定用例需求理解正确、项目风险评估合理、阶段计划可行等。

- RUP 统一过程模型
  - RUP 的时间维度结构
    - 精化阶段
      - 目标：分析问题领域，建立健全的体系结构基础，编制项目计划，完成项目中高风险需求部分的开发。
      - 里程碑：生命周期体系结构 (Lifecycle Architecture) 里程碑，包括风险分析文档、软件体系结构基线、项目计划、可执行的进化原型、初始版本的用户手册等。通过评审确定软件体系结构已经稳定、高风险的业务需求和技术机制已经解决、修订的项目计划可行等。

- RUP 统一过程模型
  - RUP 的时间维度结构
    - 构建阶段
      - 目标：完成所有剩余的技术构件和稳定业务需求功能的开发，并集成为产品，详细测试所有功能。构建阶段只是一个制造过程，其重点放在管理资源及控制开发过程以优化成本、进度和质量。
      - 里程碑：初始运行能力 (Initial Operational Capability) 里程碑，包括可以运行的软件产品、用户手册等，它决定了产品是否可以在测试环境中进行部署。此刻，要确定软件、环境、用户是否可以开始系统的运行。

- RUP 统一过程模型
  - RUP 的时间维度结构
    - 产品化阶段/移交阶段
      - 目标：确保软件对最终用户是可用的。产品化阶段可以跨越几次迭代，包括为发布做准备的产品测试，基于用户反馈的少量调整。
      - 里程碑：产品发布 (Product Release) 里程碑，确定最终目标是否实现，是否应该开始产品下一个版本的另一个开发周期。在一些情况下这个里程碑可能与下一个周期的初始阶段相重合。

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - RUP 又将软件生命周期分为9个核心工作流
      - 6个核心过程工作流
        - 业务建模 (Business Modeling)
        - 需求 (Requirements)
        - 分析设计 (Analysis & Design)
        - 实施 (Implementation)
        - 测试 (Test)
        - 部署 (Deployment)
      - 3个核心支持工作流
        - 配置和变更管理 (Configuration & Change Management)
        - 项目管理 (Project Management)
        - 环境 (Environment)

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 业务建模
      - 目的
        - 了解目标组织中的问题，确定潜在的改进之处；
        - 确保客户和终端用户对目标组织有常识性的了解；
        - 提取足以支持目标组织的系统需求；
        - 理解待部署系统的组织结构和动态过程。
      - 角色
        - 业务分析师
        - 业务架构师
        - 业务设计师
        - 技术审查员

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 需求
      - 目的
        - 建立并维护客户和其它利益相关方对于系统功能的一致性；
        - 帮助系统开发人员更换地理解系统需求；
        - 确定或者划定系统的边界；
        - 提供基础以规划迭代的技术内容；
        - 提出基础以估计开发系统的代价和时间；
        - 确定系统的用户接口，主要侧重于用户需求和目标。
      - 角色
        - 系统分析师

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 分析设计
      - 目的
        - 将需求转换成系统将要实现的功能的设计；
        - 为系统设计出一个健壮的架构；
        - 调整设计，使其与实现环境相匹配，并且达到最佳性能。
      - 角色
        - 系统分析师
        - 软件架构师
        - 数据库设计员
        - 用户接口设计员
        - 其它设计人员

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 实施
      - 目的
        - 根据在各个层上组织的子系统实现，明确代码是如何组织的；
        - 根据实现要素来设计设计要素；
        - 以单元方式测试已经开发的组件；
        - 将单个实现人员生成的结果集成到可执行系统中。
      - 角色
        - 程序开发人员
        - 系统集成人员

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 测试
      - 目的
        - 找出并且记录软件质量中的缺陷；
        - 对已知的软件质量提出建议；
        - 通过具体的证据，证明设计和需求规范中所提出的假设是否正确；
        - 验证软件产品功能是否和设计要求一样；
        - 验证是否已经正确实现需求。
      - 角色
        - 测试经理
        - 测试分析师
        - 测试设计人员
        - 测试执行人员

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 部署
      - 目的
        - 管理用于保证软件产品可以有效提供给终端用户使用的相关活动。
      - 角色
        - 部署管理人员
        - 配置管理人员
        - 培训管理人员

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 配置和变更管理
      - 目的
        - 控制工件的变更，并且维护项目工件的完整性
      - 角色
        - 配置管理人员
        - 变更控制人员
        - 集成管理人员
        - 其它参与人员

- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 项目管理
      - 目的
        - 为管理侧重于软件开发的项目提供一个框架；
        - 给规划、人员分工、执行和监控项目提供实际指南；
        - 为管理风险提供一个框架。
      - 角色
        - 项目经理
        - 管理评审员
        - 评审协调人员

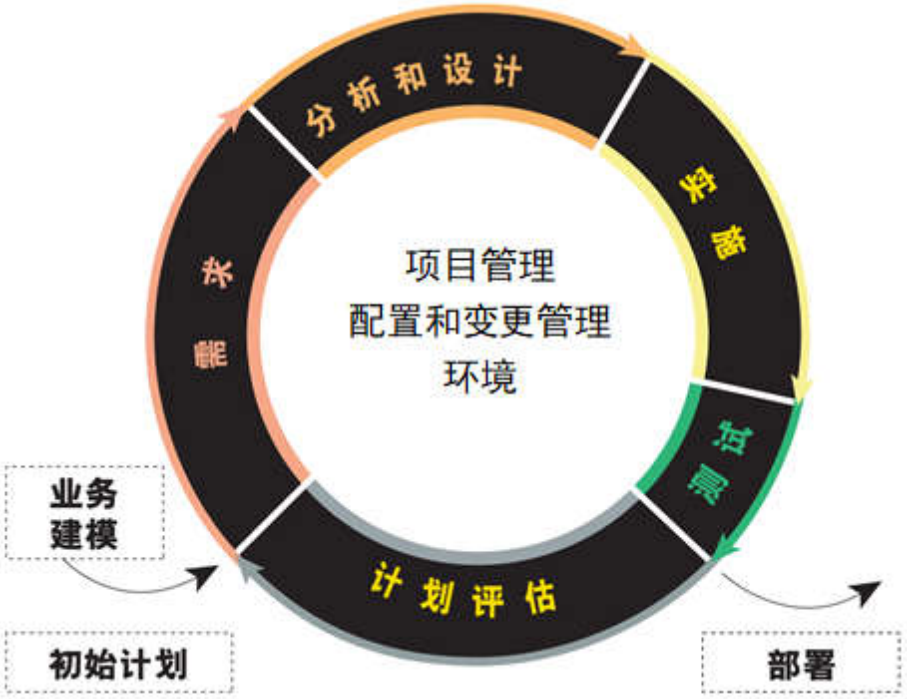
- RUP 统一过程模型
  - RUP 的核心工作流程维度结构
    - 环境
      - 目的
        - 向软件开发组织提供软件开发环境，从而为开发团队提供支持。
      - 角色
        - 过程工程师
        - 系统管理员
        - 工具专家
        - 技术资料开发工程师

- RUP 统一过程模型
  - RUP 的关键原则
    - Adapt to process 对过程加以调整改造
      - 关键点：适当裁剪过程；调整生命周期阶段的过程正规性；持续的过程改进；不确定的驱动计划和估计。
    - Balance competing stakeholder priorities 平衡竞争利益相关者的优先级
      - 关键点：定义、理解和划分业务与用户优先次序的需求；为项目和需求制定优先次序，并将需求和软件能力相结合；理解你可以利用的资产是什么？平衡资产重用与用户需求。
    - Collaborate across teams 团队间协作
      - 关键点：激发开发成员的最佳表现；创建自我管理的团队；鼓励跨职能的协作；提供一个有效的协作环境；管理发展中的工件和任务；整合业务、软件和运作团队。

- RUP 统一过程模型
  - RUP 的关键原则
    - Demonstrate value iteratively 通过迭代反复地证实目标价值
      - 关键点：启用反馈；接受改变；消除风险；调整计划。
    - Elevate the level of abstraction 提升抽象层次
      - 关键点：重新利用已有的资源；模式与重用；服务与重用；使用更高级别的工具和语言。
    - Focus continuously on quality 持续关注质量
      - 关键点：确保产品质量的团队责任制；与可论证能力集成同步的、尽早和不断的测试；逐渐建造测试自动化。

- RUP 统一过程模型
  - RUP 的迭代增量开发思想
    - RUP 是一种融合了喷泉模型和增量模型的综合生命周期模型。每一次迭代就是为了完成一定阶段性小目标而从事的一系列开发活动 。

- RUP 统一过程模型
  - RUP 的迭代增量开发思想
    - RUP 是一种融合了喷泉模型和增量模型的综合生命周期模型。每一次迭代就是为了完成一定阶段性小目标而从事的一系列开发活动 。



- RUP 统一过程模型
  - RUP 的迭代增量开发思想 (续)
    - RUP 通过迭代增量建模思想提高了风险控制能力，体现在：
      - 迭代计划安排是风险驱动的，高风险因素集中在前两个阶段解决，特别是体系结构级的风险在精化阶段就得到了解决，及早降低了系统风险。
      - 每一次迭代都包括需求、设计、实施、部署和测试活动，因此，每一个中间产品都得到了集成测试，而且这个集成测试在一个统一的软件体系结构指导下完成。
      - 每一个阶段结束时采用严格的质量评审保证里程碑文档的质量。
      - 由于中间版本的产品是逐步产生的，而且核心功能和性能需求已经包含在前续版本中，所以，可以根据市场竞争的情况适时推出中间版本产品，降低市场风险。

RUP 统一过程模型

RUP 的最佳实践

(1) 短时间分区式的迭代：2-6周，不鼓励时间的推迟。

(2) 适应性开发：小步骤、快速反馈和调整。

(3) 在早期迭代中解决高技术风险和高业务价值的问题。

(4) 不断地让用户参与迭代结果的评估，并及时获取反馈信息，以逐步阐明问题并引导项目进展。

(5) 在早期迭代中建立内聚的核心架构。

(6) 不断地验证质量，尽早、经常和实际地进行测试。

(7) 使用用例驱动软件建模：用例是获取需求、制定计划、进行设计、测试、编写终端用户文档的驱动力量。

(8) 可视化软件建模：使用 UML (Unified Modeling Language，统一建模语言) 进行软件建模。

RUP 统一过程模型

RUP 的最佳实践 (续)

(9) 细化管理需求：实现需求的提出、记录、等级划分和追踪。拙劣的需求管理是项目陷入麻烦的一个常见原因。

(10) 实现变更请求和配置管理。

RUP 统一过程模型

RUP 的剪裁

RUP 是一个通用的过程模板，包含了很多开发指南、工件、开发过程所涉及到的角色说明等，具体应用时按照需要做裁剪。

(1) 确定本项目需要的工作流程。

(2) 确定每个工作流需要的工件。

(3) 确定4个时间阶段之间的演进计划。以风险控制为原则，决定每个阶段实施的工作流、每个工作流的执行程度、生成的工件及其完成程度等。

(4) 确定每个时间阶段内的迭代计划。规划 RUP 的4个时间阶段中每次迭代开发的内容。

(5) 规划工作流的内部结构。用活动图 (Activity Diagram) 规划工作流中涉及的角色、角色负责的活动及产出的工件。

RUP 统一过程模型

RUP 的测试思想

- RUP 强调自动和快速地持续测试，把测试过程分为单元测试、集成测试、系统测试和验收测试4大阶段，测试类型涵盖软件的功能、性能和可靠性。
- RUP 的测试分类如表所示。

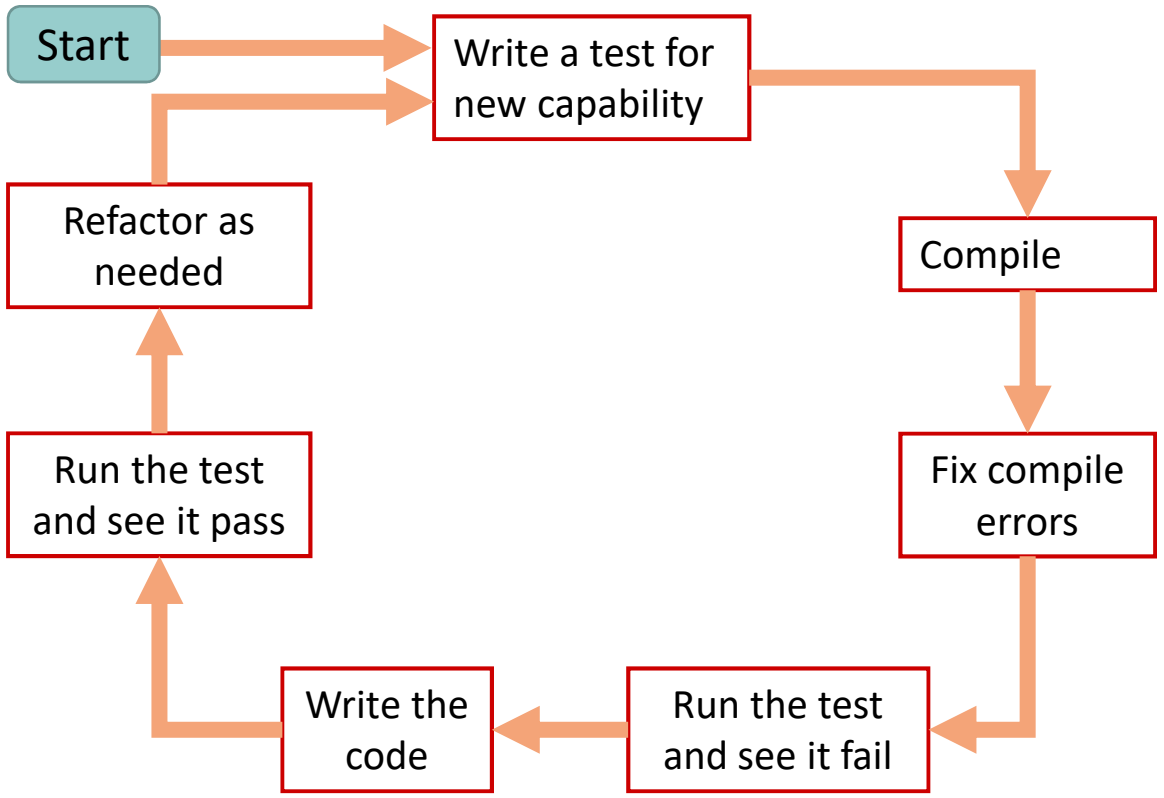
测试分类	测试类型
功能	配置测试
	功能测试
	安装测试
	安全测试
	容量测试
性能	基准测试
	竞争测试
	负载测试
	性能曲线测试
	强度测试
可靠性	完整性测试
	结构性测试

- RUP 统一过程模型
  - RUP 的软件工具
    - Rational 测试产品系列
      - Rational Quality Manager 测试过程管理工具
      - Rational Functional Tester 自动化测试工具
      - Rational Performance Tester 性能测试工具
      - Rational AppScan 安全测试工具
    - RUP 过程是一种方法，它可以方便地结合使用 Rational 工具套件，也可以采用其它厂商提供的合适的工具。

- RUP 统一过程模型
  - RUP 的适用性
    - RUP 被称为重型软件过程模型。它以用例为中心，使用 UML 的模型图作为交流语言，在项目之初就建立详细的用例说明，制定详细计划，大部分活动产生丰富完备的文档。
      - RUP 是一个“大象”过程，一般的小型团队难以支撑。
    - 轻量级开发
      - RUP 过程的剪裁
      - 敏捷运动 (2001年开始)

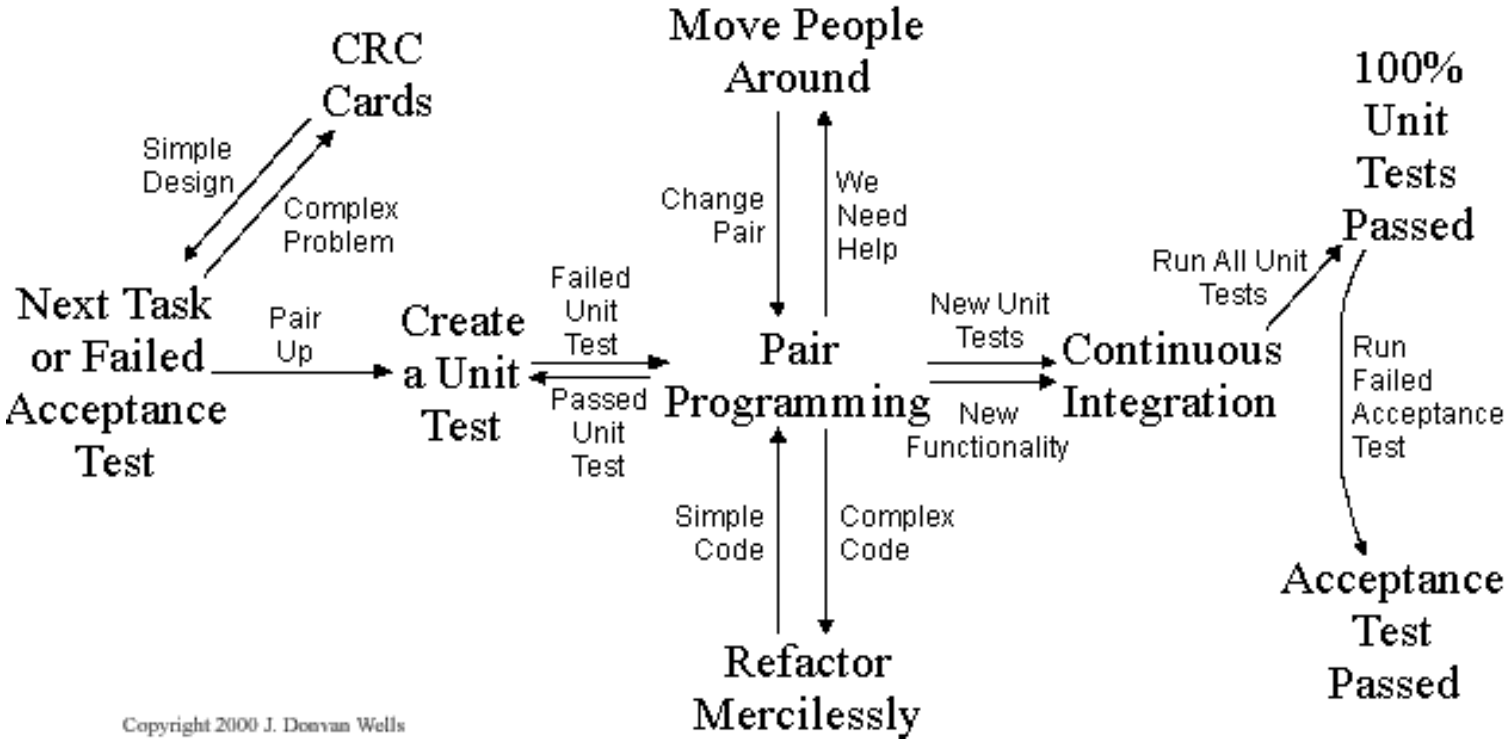
TDD

- Test-Driven Development (测试驱动开发方法)



- Refactor 重构

- TDD
  - Test-Driven Development



TDD Sub-Cycle

- UML
  - UML (Unified Modeling Language, 统一建模语言) 是面向对象技术领域内占主导地位的标准建模语言，支持从需求分析开始的软件开发的全过程。
  - UML 定义了三大类、12种模型图：
    - 结构类 Structural Diagrams：用4种模型图描述系统应用的静态结构，包括类图、对象图、组件图和配置图；
    - 行为类 Behavior Diagrams：用5种模型图描述系统动态行为的各个方面，包括用例图、序列图、行为图、协作图和状态图；
    - 模型管理类 Model Management Diagrams ：用3种模型图来组织和管理各种应用模型，包括软件包、子系统、模型等。

- 软件项目管理和复审
  - 软件项目管理
    - 软件度量、项目估算、进度控制、人员组织、配置管理、项目计划等。
      - 统计数据表明，大多数软件开发项目的失败，并不是由于软件开发技术方面的原因，而是由不适当的管理造成。
  - 软件项目复审
    - 软件项目复审在生命周期的每个阶段结束前进行。
    - 技术复审
      - 从技术角度确保软件开发质量；
      - 尽早发现问题以降低软件成本；
      - 复审过程：准备、简介、阅读文档、会议、返工、复查。
    - 管理复审
      - 成本、进度、经费等。

Thank you!