哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

软件工程
第二章 软件开发过程管理
3 软件过程模型

王忠杰
rainy@hit.edu.cn

2011年3月21日

主要内容

- **3.1** 软件过程
- **3.2** 典型软件过程模型

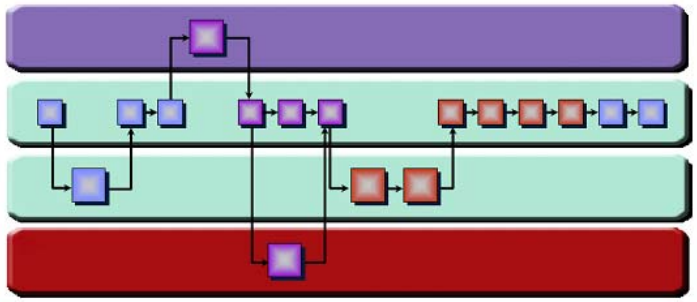
  - 瀑布模型
  - 增量过程模型
    - 增量模型
    - 快速应用程序开发(RAD)
  - 演化过程模型
    - 螺旋模型
    - 原型模型

  - * 开放源码过程
  - * 统一过程模型(RUP)
  - * 其他过程模型
    - 形式化过程
    - 软件复用过程
    - 敏捷过程模型

- **3.3** 各类过程模型的对比

3.1 软件过程

软件过程



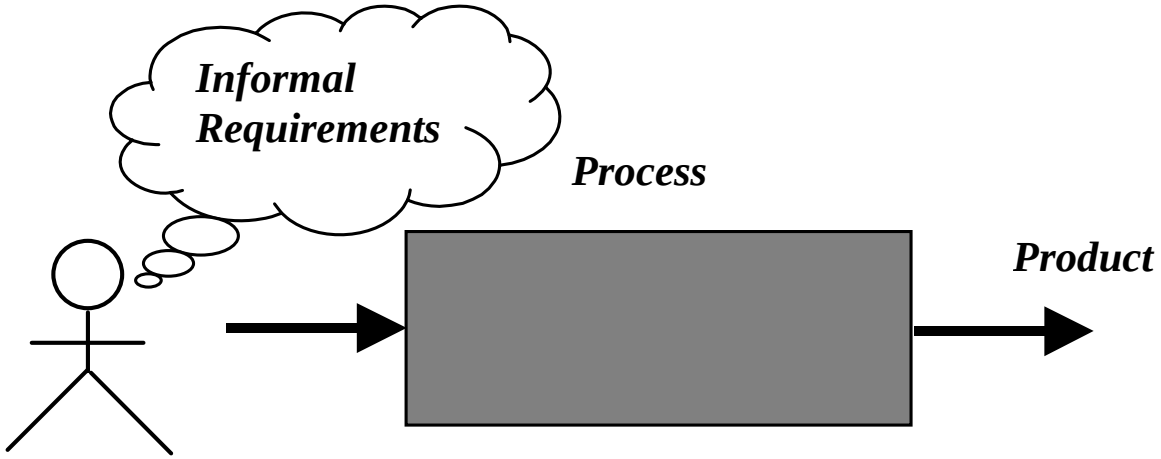
- 软件过程定义以下内容：
  - 人员与分工
  - 所执行的活动
  - 活动的细节和步骤
- 软件过程通过以下方式组织和管理软件生命周期：
  - 定义软件生产过程中的活动
  - 定义这些活动的顺序及其关系
- 软件过程的目的：
  - 标准化(可模仿)、可预见性(降低风险)、提高开发效率、得到高质量产品
  - 提升制定时间和预算计划的能力

...构造一所房子...

相同的目标

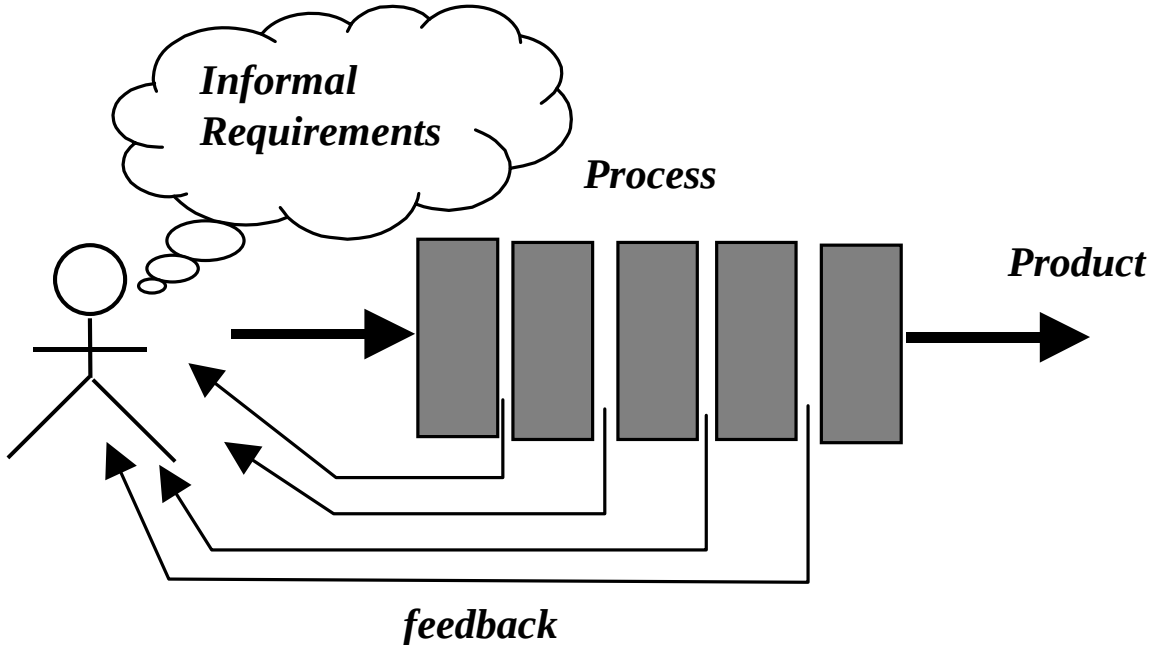
不同的活动

黑盒过程与白盒过程



- 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

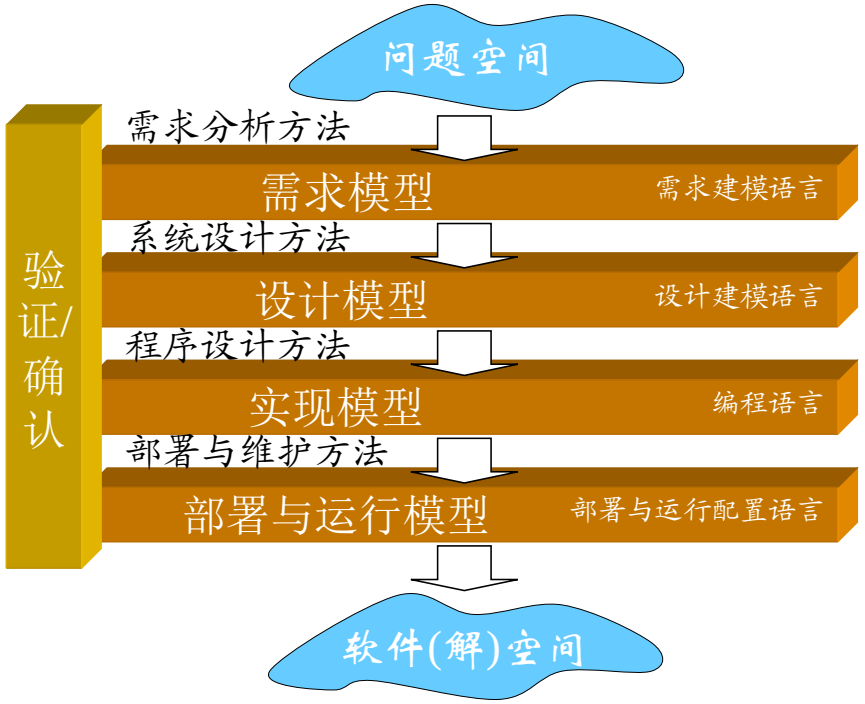
黑盒过程与白盒过程



- 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

软件过程的典型阶段

- Dream(提出设想)
- Investigation(深入调研)
- Software Specification(需求规格说明)
- Software Design(软件设计)
- Software Implementation(软件实现)
- Software Validation(软件验证)
- Software Evolution(软件演化)



3.2 典型的软件过程模型

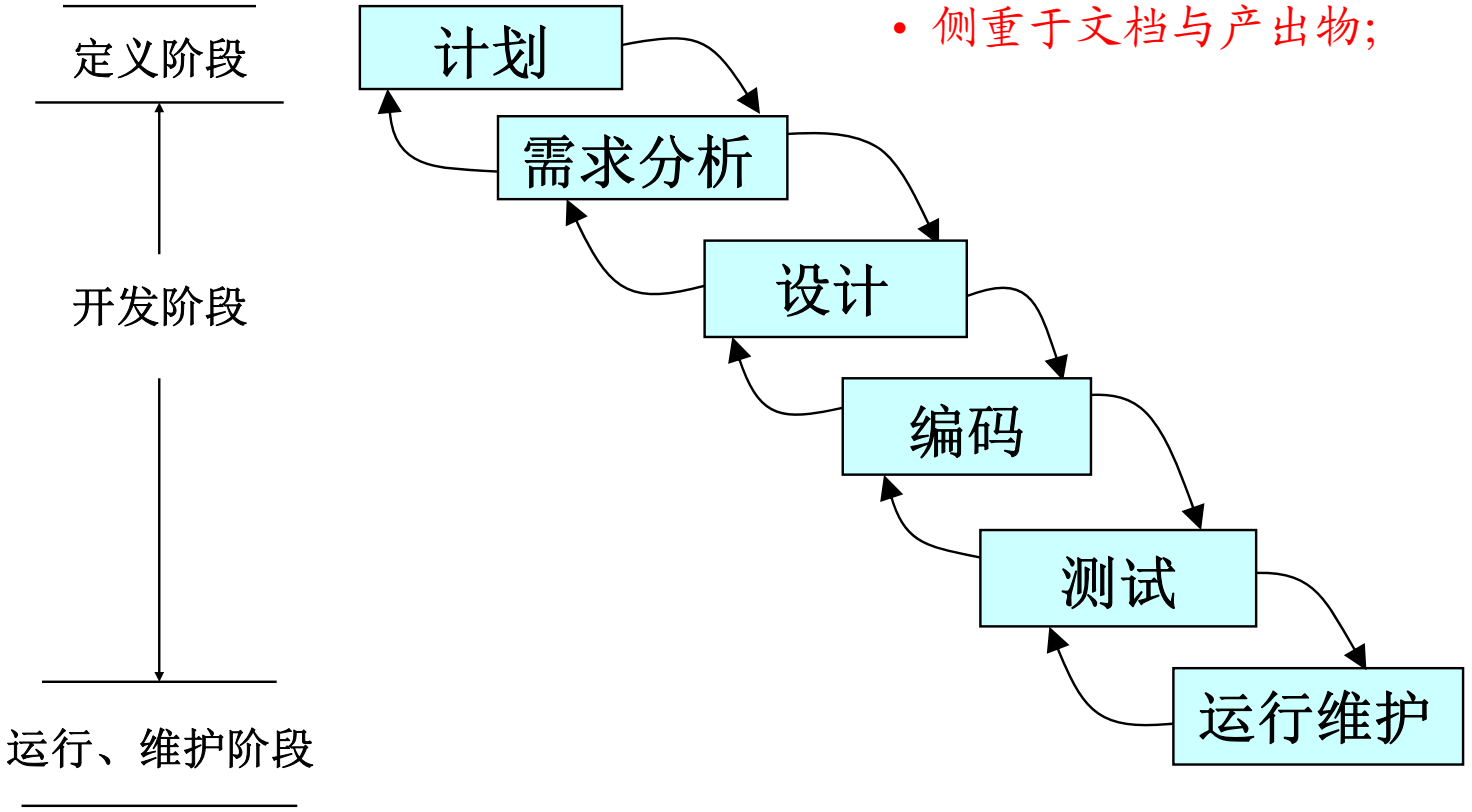
典型的软件过程模型

- **瀑布模型 (Waterfall model)**
- **增量过程模型 (Incremental process model)**
  - 增量模型 (Incremental model)
  - 快速应用程序开发(Rapid Application Development, RAD)
- **演化过程模型 (Evolutionary model)**
  - 螺旋模型 (Spiral model)
  - 原型模型 (Iterative model)
- **开放源码过程(Open source)**
- **统一过程模型(Rational Unified Process, RUP)**
- **敏捷过程模型(XP)**
- **其他过程模型 (Other models)**
  - 形式化过程 (Formal method model)
  - 软件复用过程 (Component-based reuse)

3.2.1 瀑布模型

1970年，Winston Royce

- 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；
- 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；
- 侧重于文档与产出物；



3.2.1 瀑布模型

- 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。

3.2.1 瀑布模型

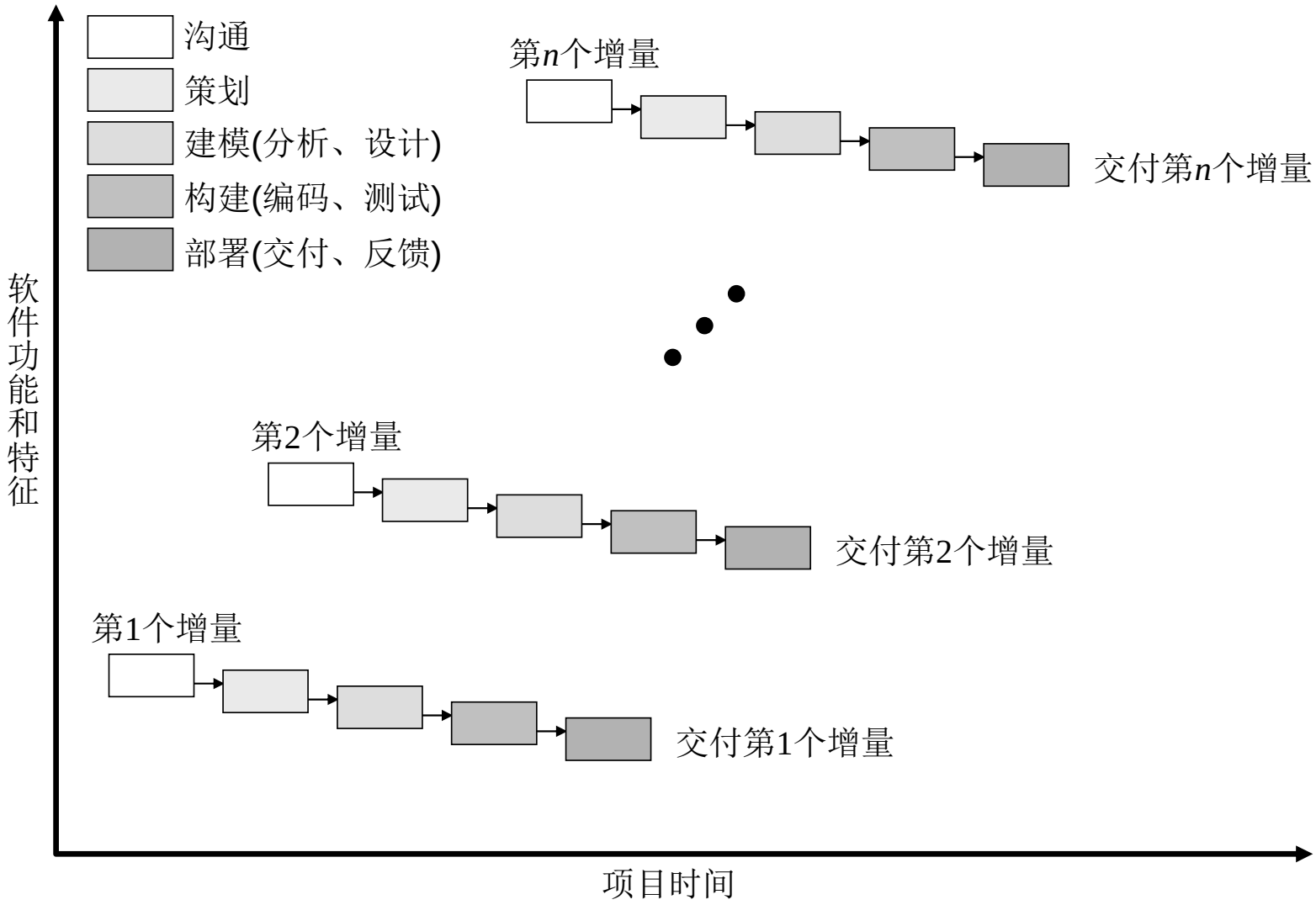
- 瀑布模型太理想化，太单纯，已不再适合现代的软件开发模式，在大型系统开发中已经很少使用；
- 但是，“线性”是最容易掌握并能熟练应用的思想方法。

- 适用场合：
  - 软件项目较小；
  - 需求在项目开始之前已经被全面的了解；
  - 需求在开发中不太可能发生重大改变；
  - 外部环境的不可控因素很少。

3.2.2 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。
  - 增量模型
  - RAD模型

(1) 增量模型



(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。

(1) 增量模型

- **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。

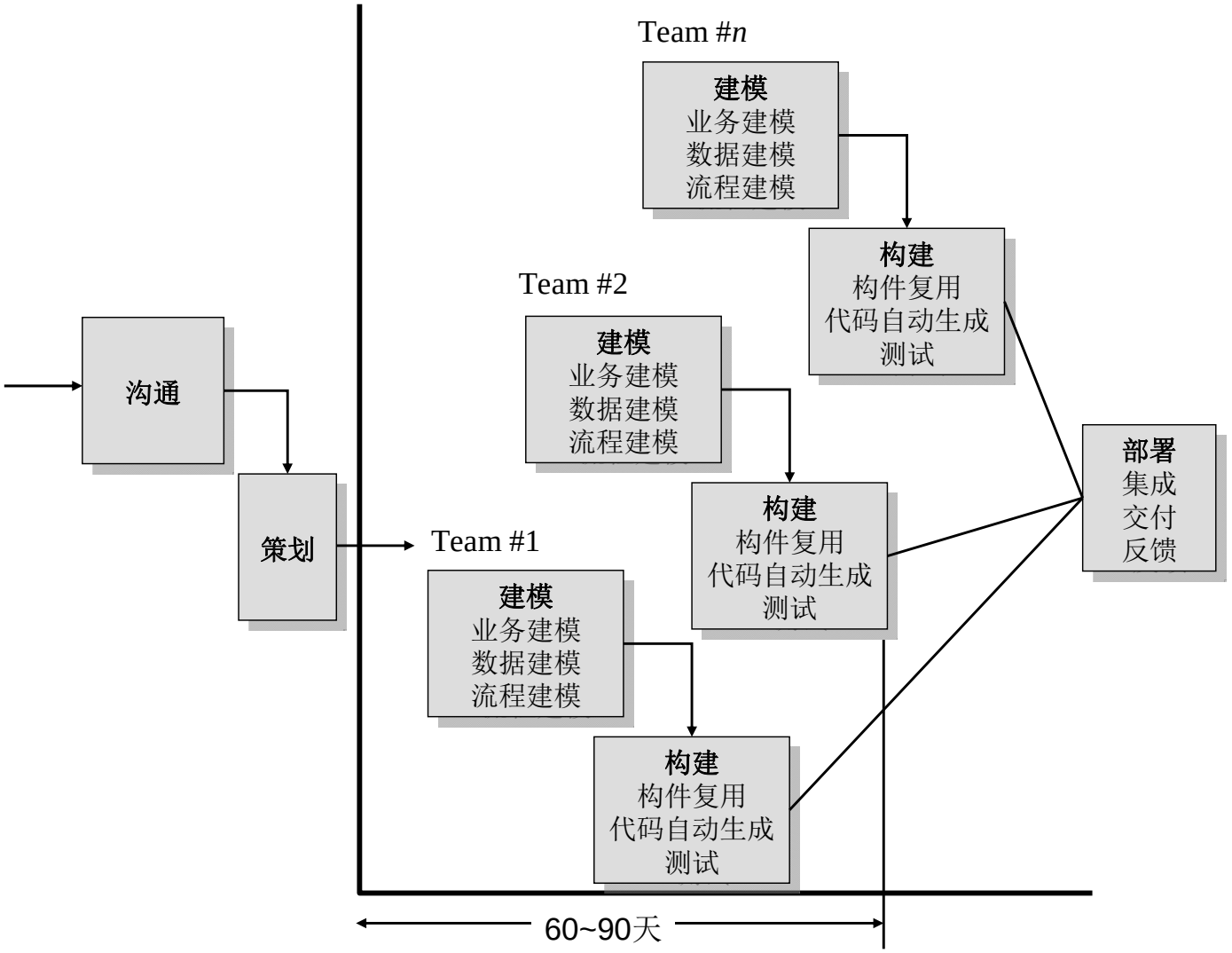
(1) 增量模型

- 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；
  - 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；
  - 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。

(1) 增量模型

- 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) RAD模型



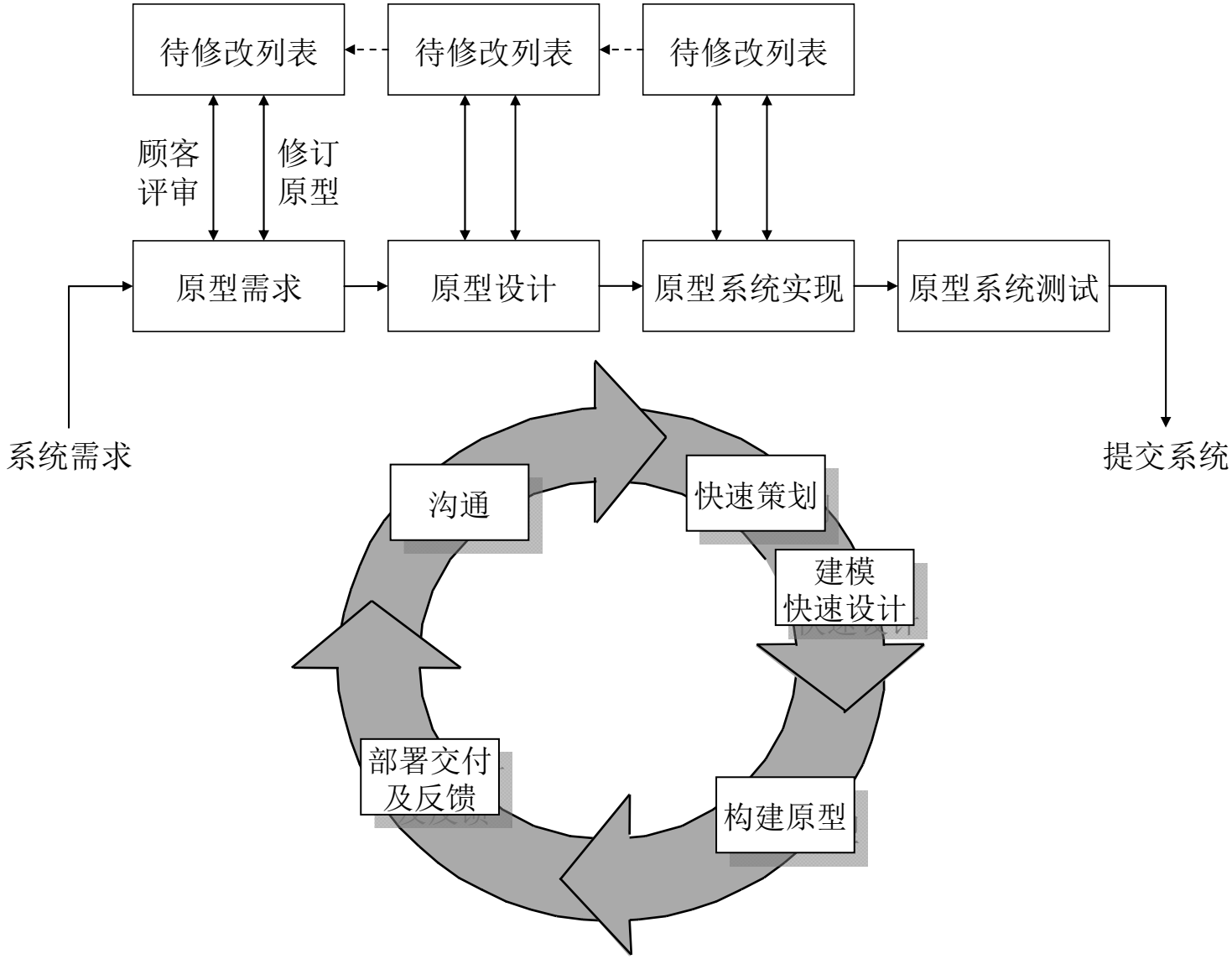
(2) RAD模型

- **快速应用开发RAD (Rapid Application Development)**
  - 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；
- **缺点：**
  - 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；
  - 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。

3.2.3 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：
  - 快速原型法
  - 螺旋模型

(1) 快速原型法



快速原型法的步骤

Step 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。

Step 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方面，如人机接口布局或者输出显示格式等；

Step 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；

Step 4：根据反馈，进一步细化需求并调整原型；

Step 5：原型系统不断调整以逼近用户需求。

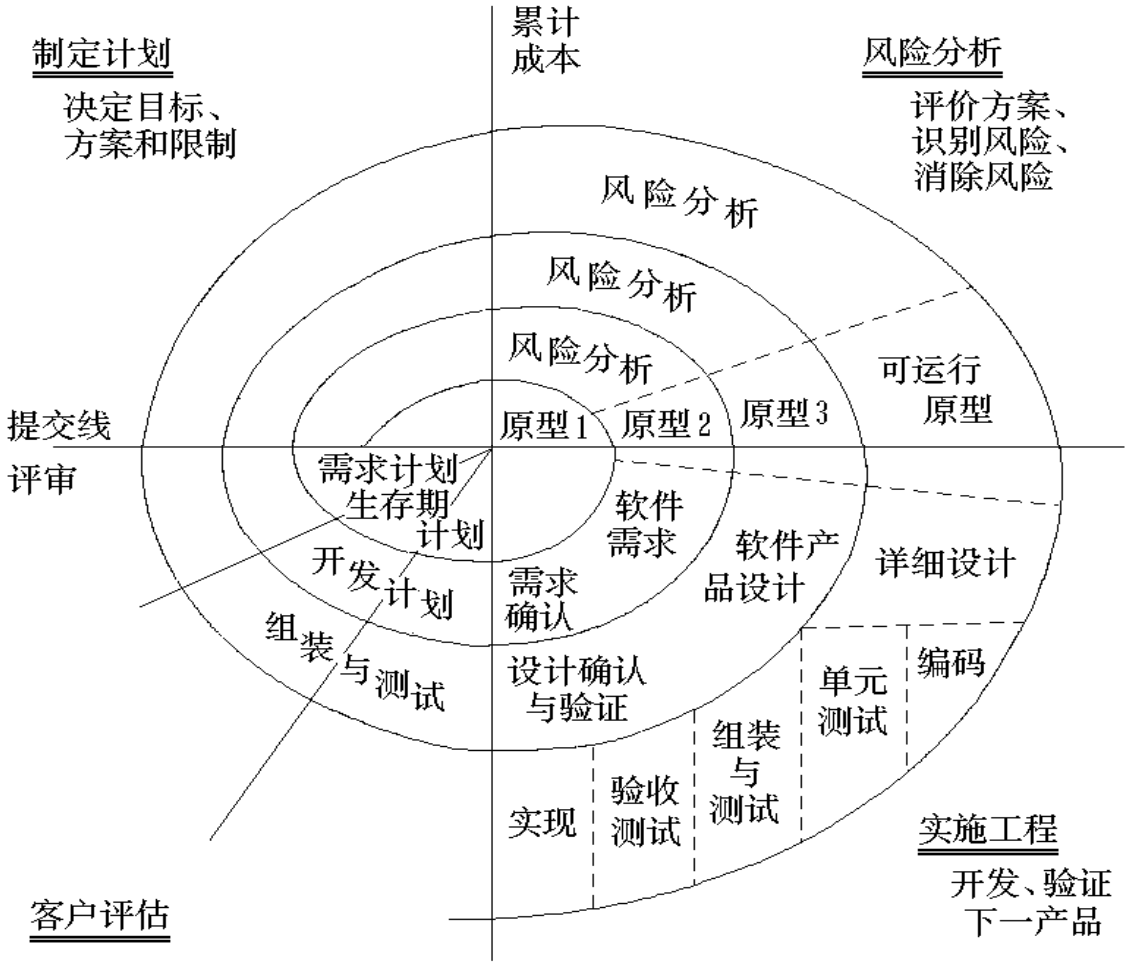
“原型”的类型

- **Throwaway prototyping(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)；
- **Evolutionary prototyping(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；

快速原型法的优缺点

- 优点：
  - 提高和改善客户/用户的参与程度，最大程度的响应用户需求的变化；
- 缺点：
  - 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；
  - 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) 螺旋式过程模型



(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；

(2) 螺旋式过程模型

- 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。
- 缺陷：
  - 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；

3.2.4* 开放源码过程

- 什么是**open source**？它有哪些优点？有哪些缺点？
- 你知道哪些**open source**的软件？向大家简要分享你对它们的了解。
- 你所了解的基于**open source**的软件开发过程是什么样子？

开放源码的一些例子

- Apache: World's most popular web server

  http://www.apache.org

- Linux: World's fastest growing server operating system

  http://www.linux.org

- MySQL: World's most popular open source database

  http://www.mysql.com

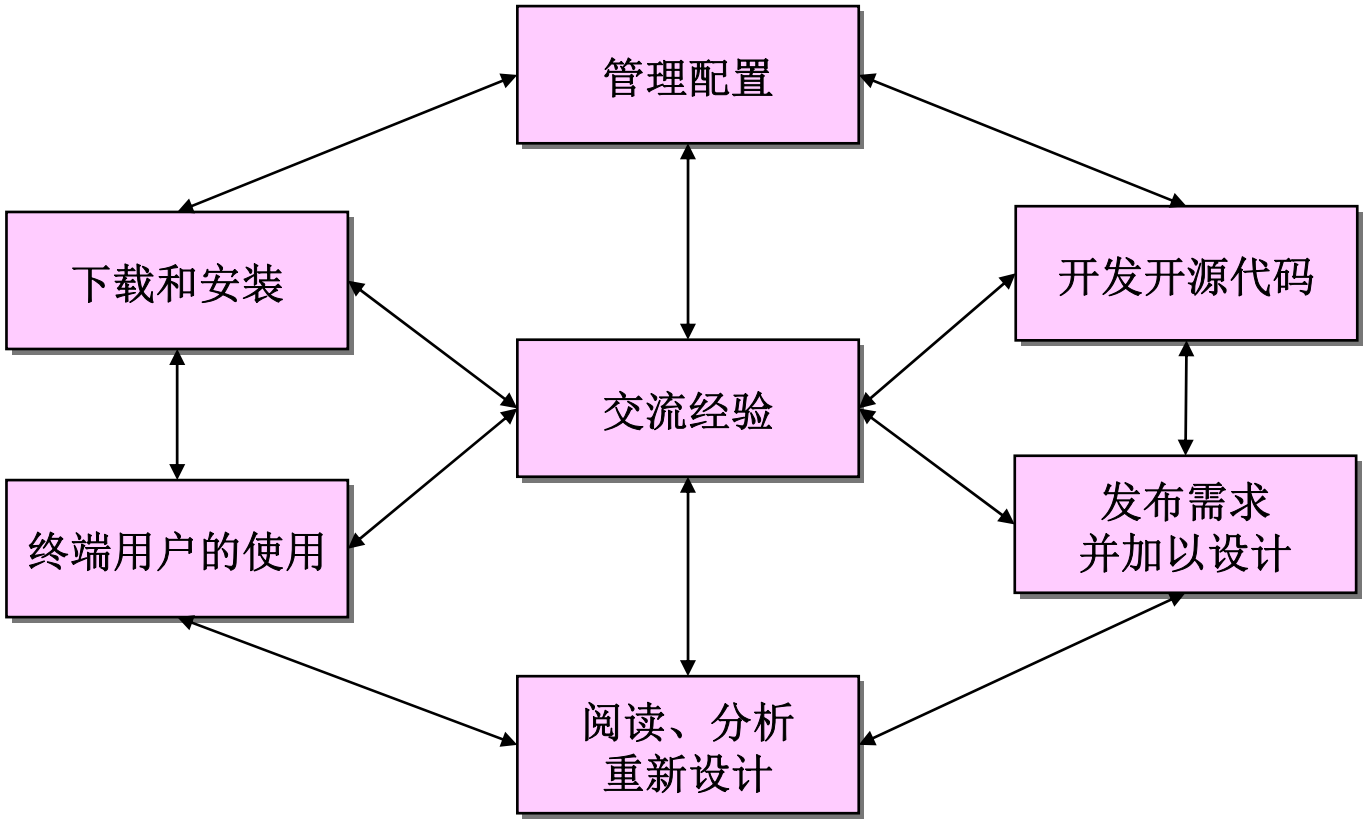
- Mozilla: An open-source web browser

  http://www.mozilla.org

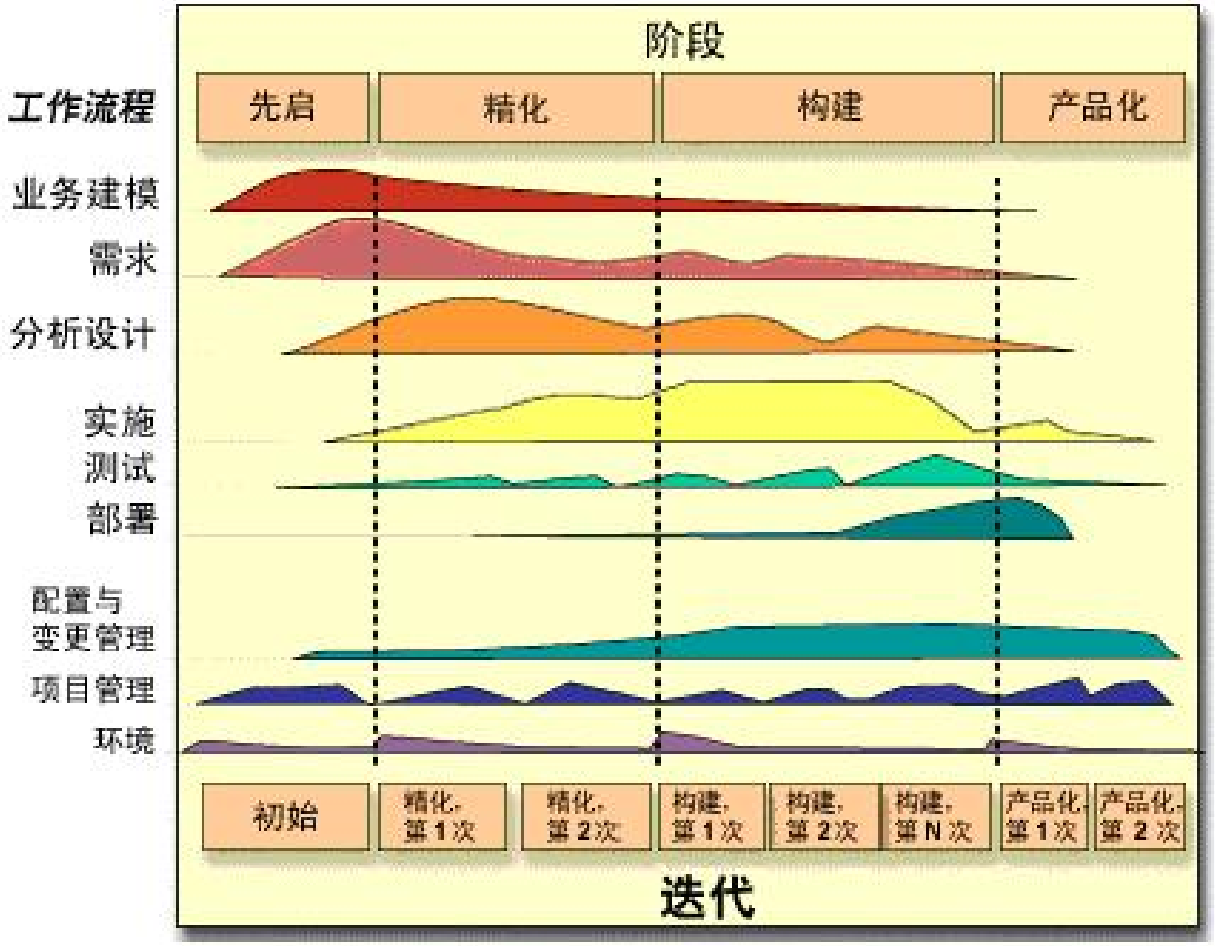
- SourceForge: Resources for open-source developers and a directory of in-development open-source software.

  http://www.sourceforge.net

3.2.4 开放源码过程

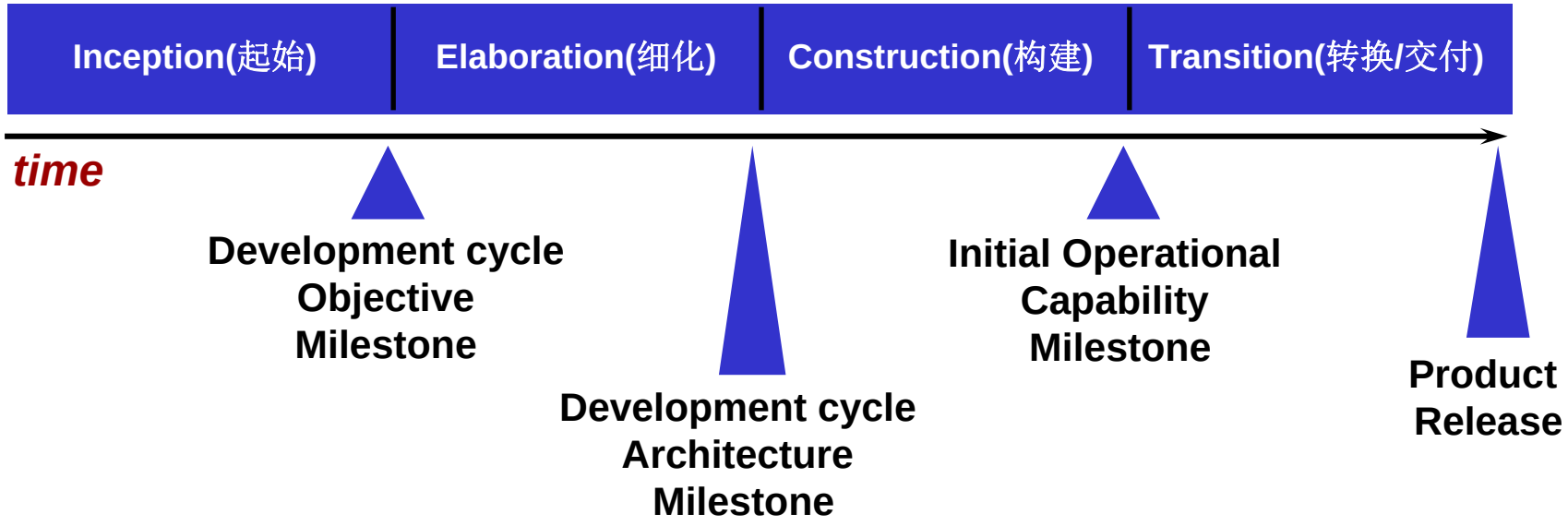


3.2.5* 统一过程模型(RUP)



3.2.5 统一过程模型(RUP)

- **RUP (Rational Unified Process)由Rational公司最先提出，是一种面向对象(OO)的软件开发方法论；**



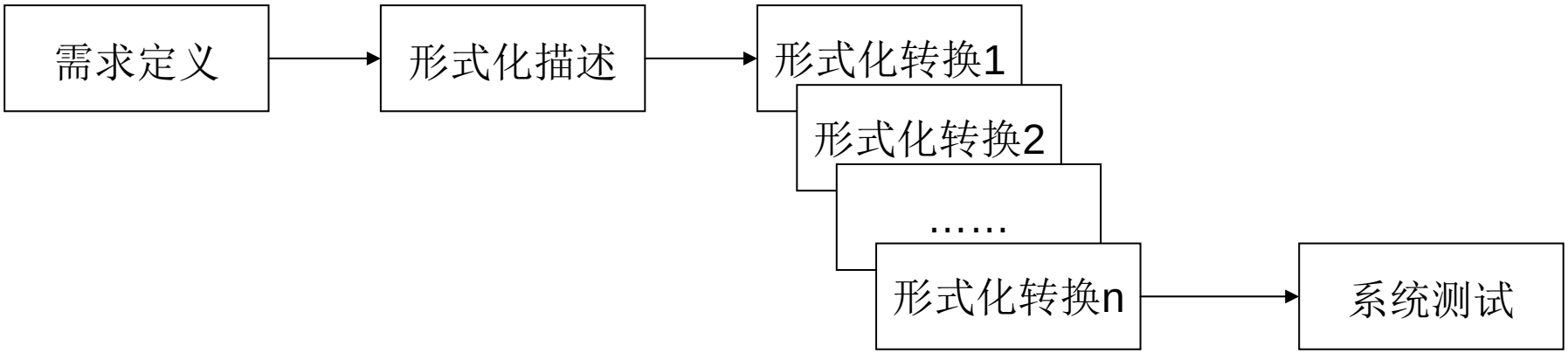
RUP的四个技术阶段

- **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Use Case (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，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)；

3.2.6* 其他过程模型

- 形式化过程 **(formal method model)**
- 基于构件的软件复用过程 **(component-based reuse)**
- 敏捷过程模型**(agile process model)**

(1) 形式化模型



- 使用严格的数学形式来刻画每一阶段的产物(需求、设计、程序、测试);
- 应用一系列形式化方法在各阶段的产物之间进行自动/半自动的转换。

(1) 形式化模型

- 优点：
  - 应用数学分析方法，歧义性、不完整性、不一致性等问题更容易被发现和改正，目的是“提供无缺陷的软件”。
- 缺陷：
  - 形式化数学方法难以理解，可视性太差，对开发人员技能要求较高；
  - 构造形式化模型是一件非常耗时的工作，成本也很高；
  - 软件系统中的某些方面难以用形式化模型表达出来(如用户界面)；
- 应用场合：
  - 对可靠性和安全性要求较高的一些关键系统，在真正被投入使用之前，需要严格保证100%的正确。传统的方法靠人去验证，难以奏效。

——太过于理想化，因此仅停留在理论研究中，实践中很少使用

(2) 基于构件的软件复用

- 该过程模型的主要思想是复用(reuse)；
- 针对一个新的软件系统，不是完全从一无所有开始入手，而是通过使用已有的软件单元(称为“软构件”)来构造系统；
- 主要过程：
  - 需求分析；
  - 体系结构设计；
  - 构件获取(购买、重新开发)；
  - 构件修改与测试；
  - 构件组装；
  - 集成测试；

(3) 敏捷过程模型(agile process model)

- 开发过程中的“变化”是无处不在的，也是不可避免的；
- 在实际项目中，很难预测需求和系统何时以及如何发生变化；
- 对开发者来说，将变化的意识贯穿在每一项开发活动中；

- **2001年，17位编程大师共同发布《敏捷软件开发宣言》：**
  - “人”以及“人与人的互动” 胜于 “过程”和”工具”
    Individuals and interactions over processes and tools
  - 可运行的软件 胜于 面面俱到的文档
  - Working software over comprehensive documentation
  - 客户合作 胜于 合同谈判
    Customer collaboration over contract negotiation
  - 响应变化 胜于 遵循计划
    Responding to change over following a plan

敏捷宣言背后遵循的原则：小步快跑，及时反馈

- 产品设计开发过程中最重要的是：通过及早并频繁地交付有价值的软件来赢得客户的满意。
- 要欢迎改变需求，即使是在开发的后期。敏捷过程中的“变”，是为了让客户保持竞争优势。
- 交付要频繁，交付的软件要可运行。交付周期从数周到数月不等，但间隔的时间要尽量短。
- 在整个项目过程中，开发人员必须每天都与业务人员一起工作。
- 项目组所选的人要积极。然后，给予他们工作所需要的环境、支持和信任。
- 面对面交谈是开发团队内部和开发团队间传递信息最有效率和效果的方法。
- 可运行的软件是衡量进度的首要指标。
- 敏捷过程提倡可持续的开发。出资人、开发者和用户应始终保持稳定的步调（迭代周期）。
- 对技术的精益求精以及对设计的不断完善，将会提高敏捷性。
- 尽量去掉不必要做的工作——“简洁”的艺术。
- 最佳的架构、需求和设计产生于自组织的团队。
- 团队要定期反思“如何能变得更有效率”，然后对自己的行为进行相应的优化和调整。

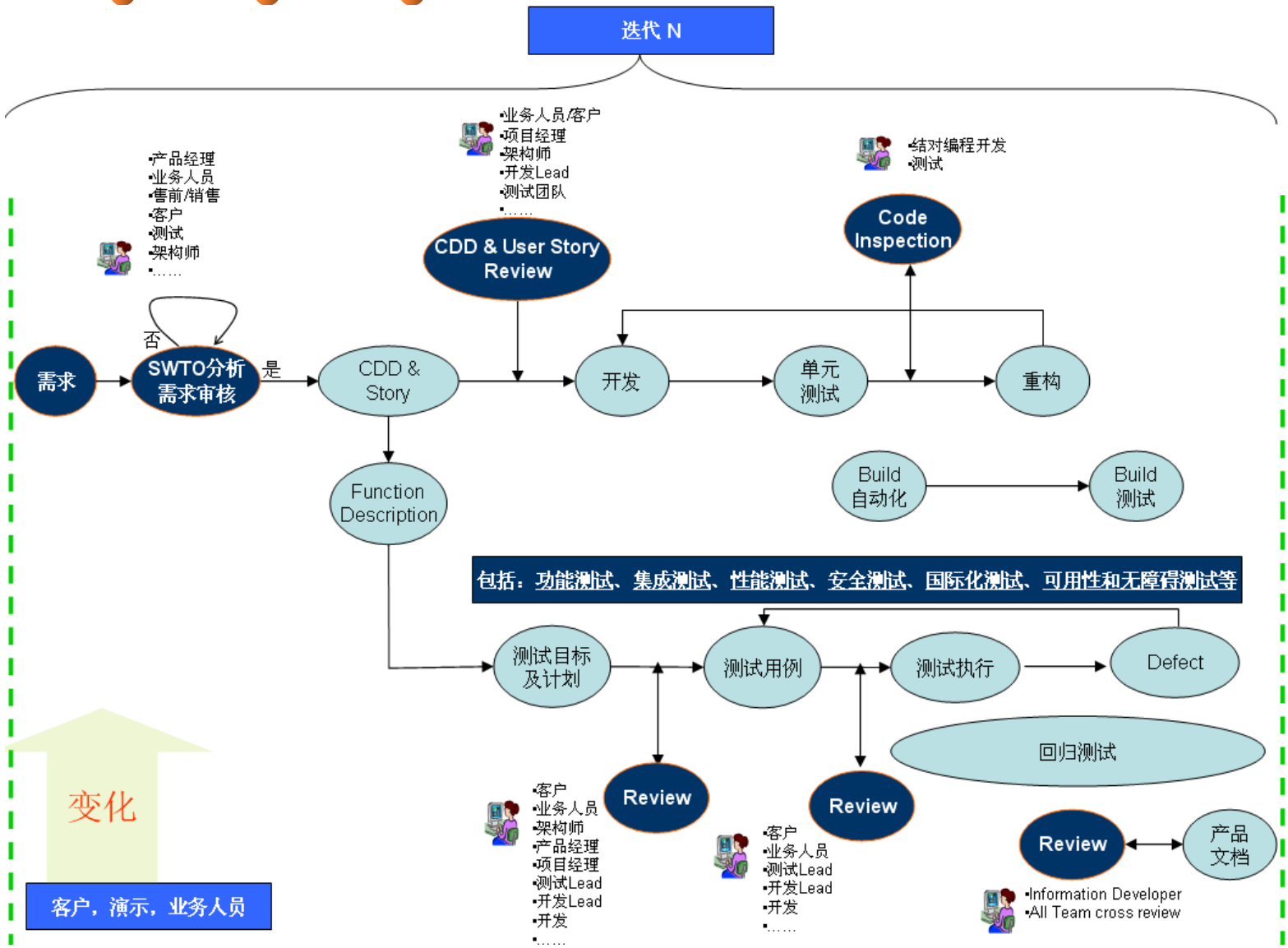
不强调文档，转向强调可运行的软件片段；

开发者与顾客之间的频繁沟通；

快速开发，快速反馈，快速修改；

连续不断的短周期迭代；

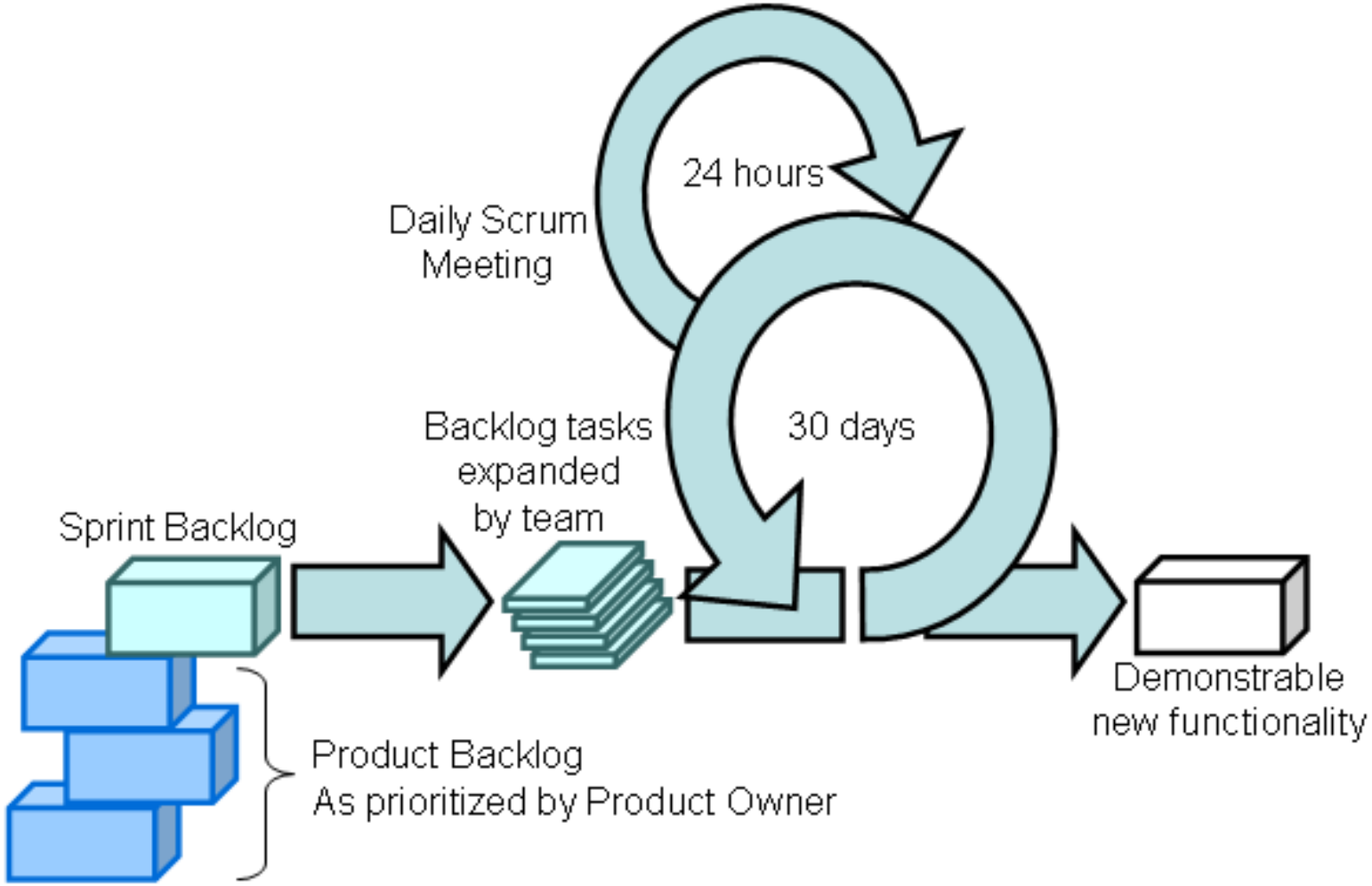
敏捷过程模型



敏捷过程模型的典型代表：Scrum

- **Scrum**
  - 一个敏捷开发框架：增量/迭代的开发过程。
  - 整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个Sprint，每个Sprint的建议长度是2到4周。
  - 使用产品Backlog来管理需求，是一个按照商业价值排序的需求列表，列表条目的体现形式通常为用户故事。
  - 总是先开发对客户具有较高价值的需求。
  - 在Sprint中，Scrum团队从产品Backlog中挑选最高优先级的需求进行开发。挑选的需求在Sprint计划会议上经过讨论、分析和估算得到相应的任务列表(backlog)。
  - 在每个迭代结束时，Scrum团队将递交潜在可交付的产品增量。

敏捷过程模型的典型代表：Scrum



敏捷过程模型的典型代表：Scrum

- 三个角色：
  - 产品负责人(Product Owner)：确定产品的功能，负责维护产品Backlog、deadline、priority、ROI；验收结果；
  - ScrumMaster：团队leader，保证开发过程按计划进行；组织每日站会、Sprint计划会议、Sprint评审会议和Sprint回顾会议；通过外在/内在协调，确保团队资源完全可被利用并且全部是高产出的；
  - Scrum团队：在每个Sprint中将产品Backlog中的条目转化成为潜在可交付的功能增量；规模在5-9人；具备交付产品增量所需要的各种技能；
- 六个时间箱：
  - Sprint (冲刺)：代表一个2-4周的迭代；
  - 发布计划会议(Release Planning Meeting) →Product Backlog
  - Sprint计划会议(Sprint Planning Meeting) → Sprint Backlog
  - 每日站会(Daily Scrum Meeting)
  - Sprint评审会(Sprint Review Meeting)
  - Sprint 回顾会议(Sprint Retrospective Meeting)

  } Sprint Burndown Chart
  Release Burndown Chart

3.3 各种过程模型的比较

各种过程模型的比较

- 针对本次课程所讲授的软件过程模型，通过对比分析，找出这些过程模型之间的异同，并分析各自的优缺点；
- 分别用若干个简短的词来描述每一种软件过程模型的核心特征；
- 考虑以下维度：
  - 时间效率、成本、人力资源、开发质量、顾客满意度、需求扩展、需求变化、风险、与顾客交互程度、适用项目规模、适用deadline紧急程度、项目管理的方便程度、等等。

结束

2011年3月21日