北京交通大學
BEIJING JIAOTONG UNIVERSITY

软件系统分析与设计
System Analysis & Design

M210007B [03]

Zhenyan Ji, Jingxin Su & Haiming Liu

Sunday, May 29, 2022

上节课知识回顾

上节课的主要内容

软件是人们对客观世界中问题空间与解空间的具体描述，是客观事物的一种反应，是知识的题练和“固化”

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序，数据及其相关文档的完整集合

软件是程序以及开发使用维护程序所需要的文档

软件=程序+文档

软件的特性

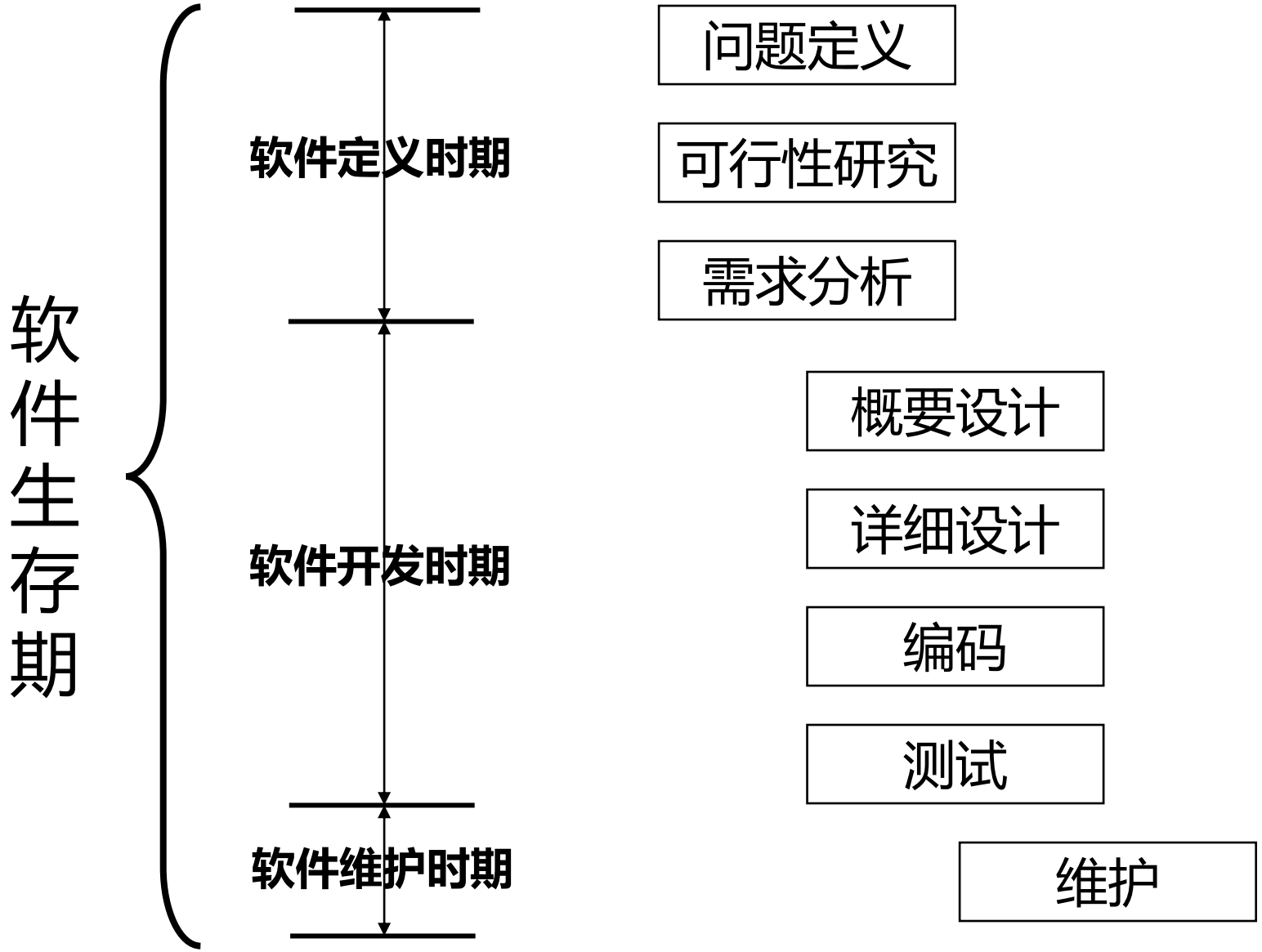
1. 形态特性
2. 智能特性
3. 开发特性
4. 质量特性
5. 生产特性
6. 管理特性
7. 环境特性
8. 维护特性
9. 废弃特性
10. 应用特性

软件危机
Software Crisis

软件工程

概括地说，软件工程是指导计算机软件开发和维护的工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它，这就是**软件工程**

软件生存期


软件生存期
软件定义时期
软件开发时期
软件维护时期
问题定义
可行性研究
需求分析
概要设计
详细设计
编码
测试
维护

软件工程三要素

三要素：过程、方法和工具

软件工程过程是为了获得高质量的软件所需要完成的一系列任务框架，它规定了完成各项任务的工作步骤

软件工程方法为软件开发提供了“如何做”的技术

软件工程工具为软件工程方法提供了自动的或半自动的软件支撑环境

课程计划

软件工程的基础

1. 软件工程基础/软件生命周期
2. **软件过程模型**
3. 软件项目管理

**系统分析与需求描述
结构化设计方法**

4. 系统分析方法与问题定义
5. 需求分析与需求获取
6. 用例建模与用例描述
7. 结构化系统分析与设计

面向对象的设计方法

8. 面向对象方法概述
9. 面向对象建模与UML
10. 软件系统动态建模
11. 软件系统静态建模

12. 数据库设计
13. 综合案例分析
14. 考前复习

/-1 软件开发过程模型
Software Development Process Model

过程

软件工程的基础是过程（process）层，软件过程将各个技术层次结合在一起，使得合理、及时地开发计算机软件成为可能。过程定义了一个框架，构建该框架是有效实施软件工程技术必不可少的。软件工程过程构成了软件项目管理控制的基础，建立了工作环境以便应用技术方法、提交工作产品（模型、文档、数据、报告、表格等）、建立里程碑、保证质量及正确的管理变更

软件过程

软件过程是工作产品构建时所执行的一系列活动、动作和任务的集合，有助于提高产品的交付质量

活动（activity）主要实现宽泛的目标（如与利益相关者进行沟通），与应用领域、项目大小、结果复杂性或者实施软件工程的重要程度没有关系

动作（action，如体系结构设计）包含主要工作产品生产过程中的一系列任务

任务（task）关注小而明确的目标，能够生产实际产品（如构建一个单元测试）

过程不是对如何构建软件的严格规定，而是已中可适应性的调整方法，以便于工作人员（软件团队）可以挑选合适的工作动作和任务集合。其目标通常是及时、高质量地交付软件，以满足软件项目资助方和最终用户的需求

软件过程与软件工程同义么?

软件开发过程模型

软件开发过程模型能清晰直观的表达开发全过程，明确规定的要完成的主要活动和任务，可以作为软件项目开发的基础

对于一个具体软件系统来说，应采用合适的开发方法，使用事宜的程序设计语言，组织具有相应技能的开发人员参与，并采用适合的管理方法和手段，对整个开发活动的全过程进行有效的控制

经典的软件开发过程模型

瀑布模型

快速原型模型

增量模型

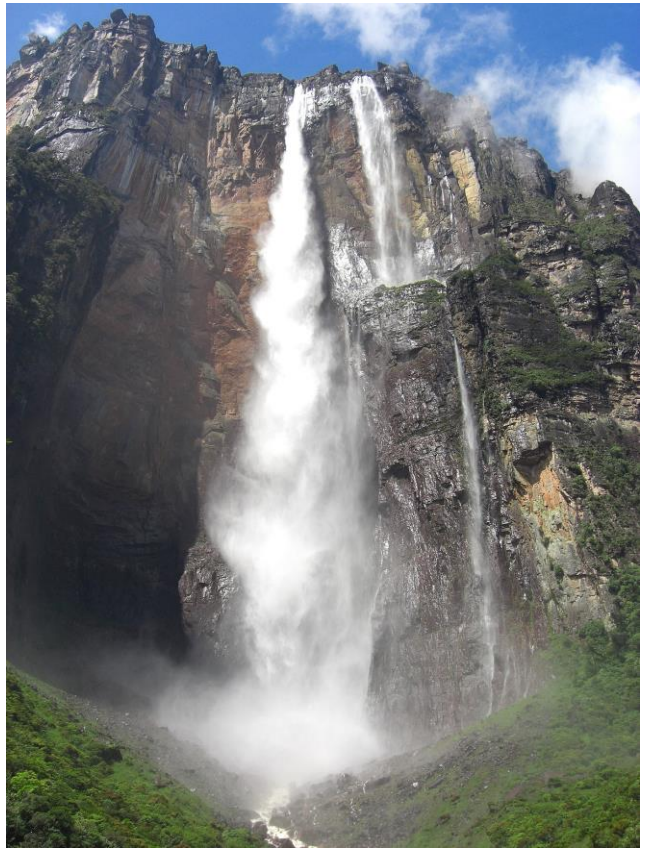
螺旋模型

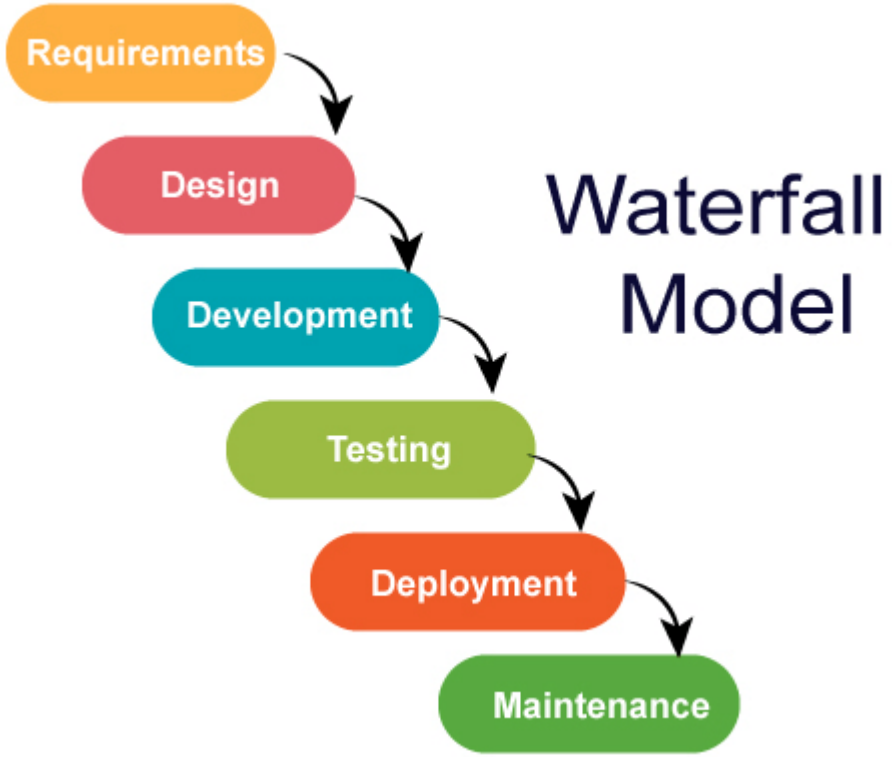
统一过程

敏捷过程

瀑布模型（waterfall model）

在20世纪80年代之前，瀑布模型一直是唯一被广泛采用的生命周期模型，也叫线性顺序模型（linear sequential model），是一个系统的、顺序的软件开发方法





瀑布模型的特点

阶段间具有顺序性和依赖性。其中包含两重含义：

1. 必须等前一阶段的工作完成之后，才能开始后一阶段的工作

2. 前一阶段的输出文档就是后一阶段的输入文档

瀑布模型的特点

推迟实现的观点：

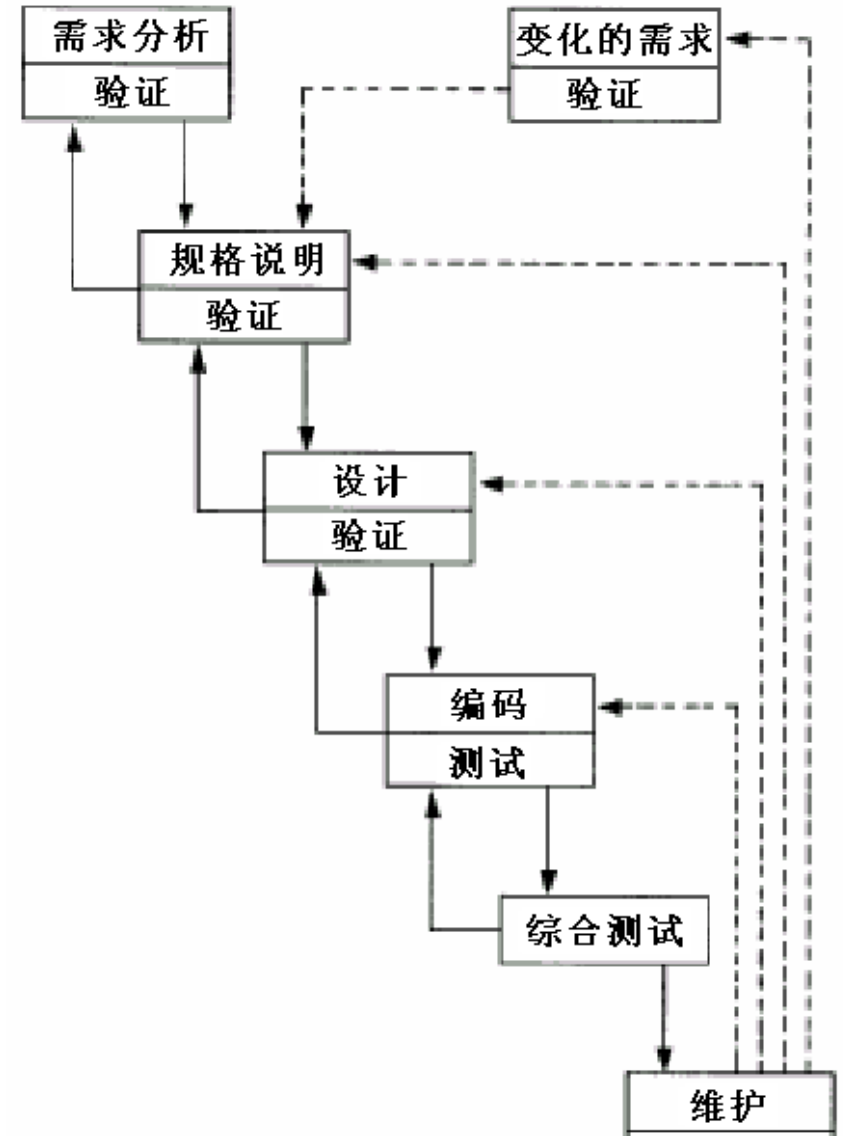
1. **瀑布模型在编码之前设置了系统分析和系统设计的各个阶段，分析与设计阶段的基本任务规定，在这两个阶段主要考虑目标系统的逻辑模型，不涉及软件的物理实现**
2. **清楚地区分逻辑设计与物理设计，尽可能推迟程序的物理实现，是按照瀑布模型开发软件的一条重要的指导思想**

瀑布模型的特点

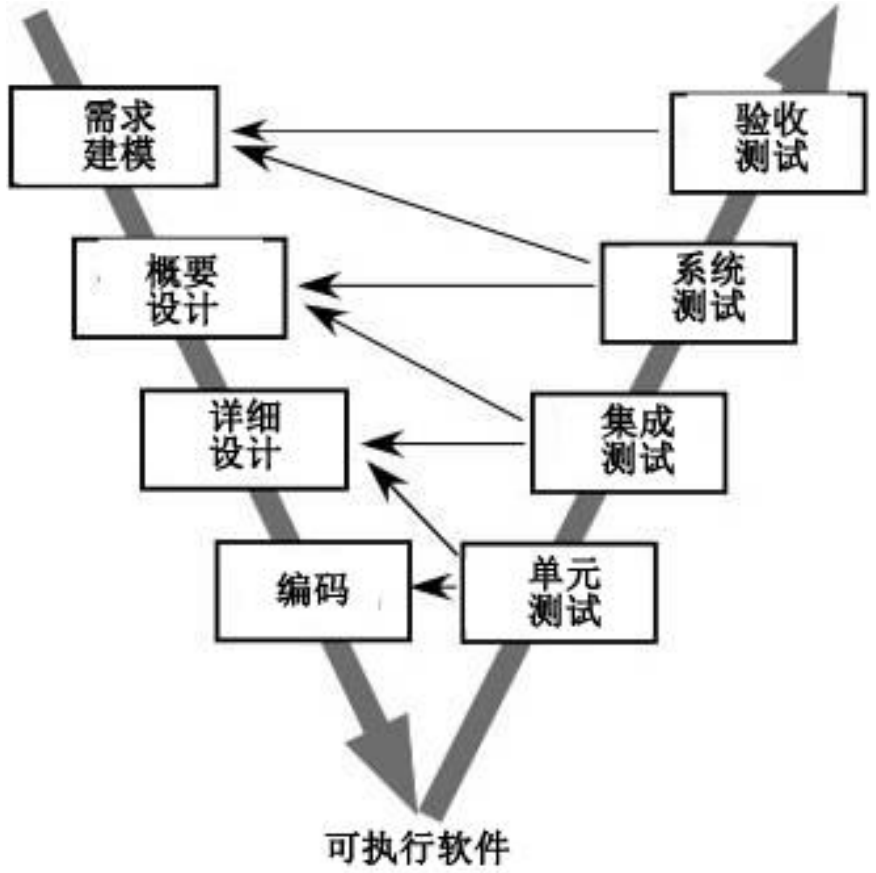
质量保证的观点

1. **每个阶段都必须完成规定的文档，没有交出合格的文档就是没有完成该阶段的任务**
2. **每个阶段结束前都要对所完成的文档进行评审，以便尽早发现问题，改正错误**

实际的瀑布模型



瀑布模型变体——V模型



瀑布模型优点

1. 可强迫开发人员采用规范化的方法。
2. 严格地规定了每个阶段必须提交的文档。
3. 要求每个阶段交出的所有产品都必须是经过验证的

瀑布模型缺点

1. 由于瀑布模型几乎完全依赖于书面的规格说明，很可能导致最终开发出的软件产品不能真正满足用户的需要。如果需求规格说明与用户需求之间有差异，就会发生这种情况
2. 瀑布模型只适用于项目开始时需求已确定的情况

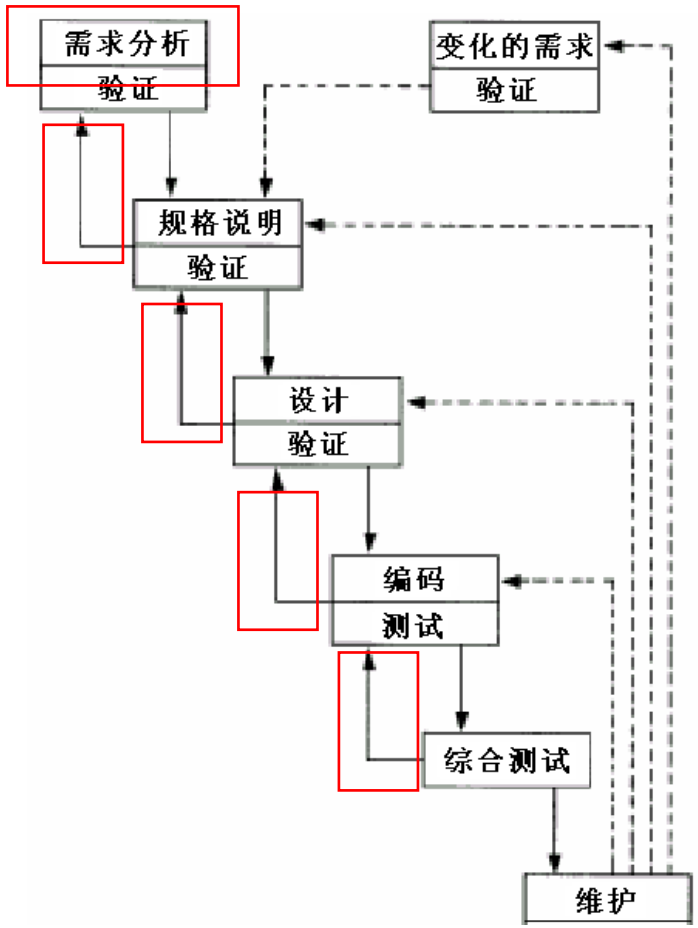
慎重考虑是否使用瀑布模型的情况

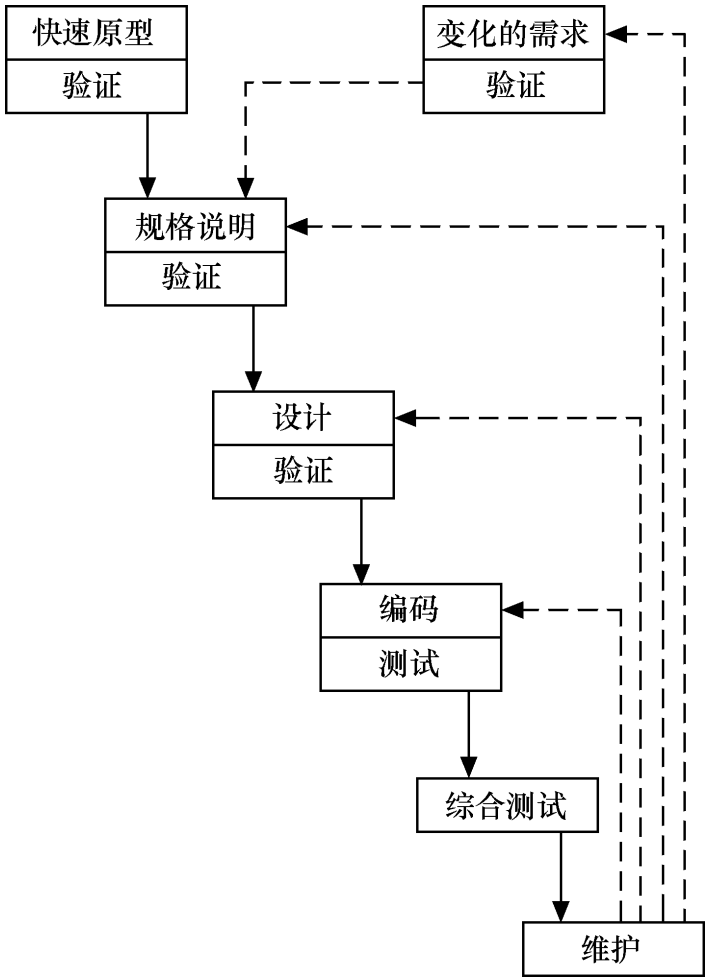
1. 不能充分理解客户需求或客户需求可能迅速发生变化
2. 系统太大、太复杂，不能一次性做完所有的事情
3. 事先拟采用的技术迅速发生变化
4. 提供的资源有限
5. 无法利用各个开发阶段的某一中间产品

可以使用瀑布模型的情况

1. 系统的所有功能、性能要求客户可以一次性准确交付
2. 是首次开发的新的系统并且淘汰全部老系统

快速原型模型


需求分析
验证
变化的需求
验证
规格说明
验证
设计
验证
编码
测试
综合测试
维护


快速原型
验证
变化的需求
验证
规格说明
验证
设计
验证
编码
测试
综合测试
维护

快速原型模型优点

1. 有助于满足用户的真实需求
2. 原型系统已经通过与用户的交互而得到验证，据此产生的规格说明文档能够正确地描述用户需求
3. 软件产品的开发基本上是按线性顺序进行
4. 因为规格说明文档正确地描述了用户需求，因此，在开发过程的后续阶段不会因为发现规格说明文档的错误而进行较大的返工

快速原型模型优点

5. 开发人员通过建立原型系统已经学到了许多东西，因此，在设计和编码阶段发生错误的可能性也比较小，这自然减少了在后续阶段需要改正前面阶段所犯错误的可能性
6. 快速原型的突出特点是“快速”。开发人员应该尽可能快地建造出原型系统，以加速软件开发过程，节约软件开发成本

增量模型

增量模型也称为渐增模型（incremental model），是Mills等于1980年提出来的

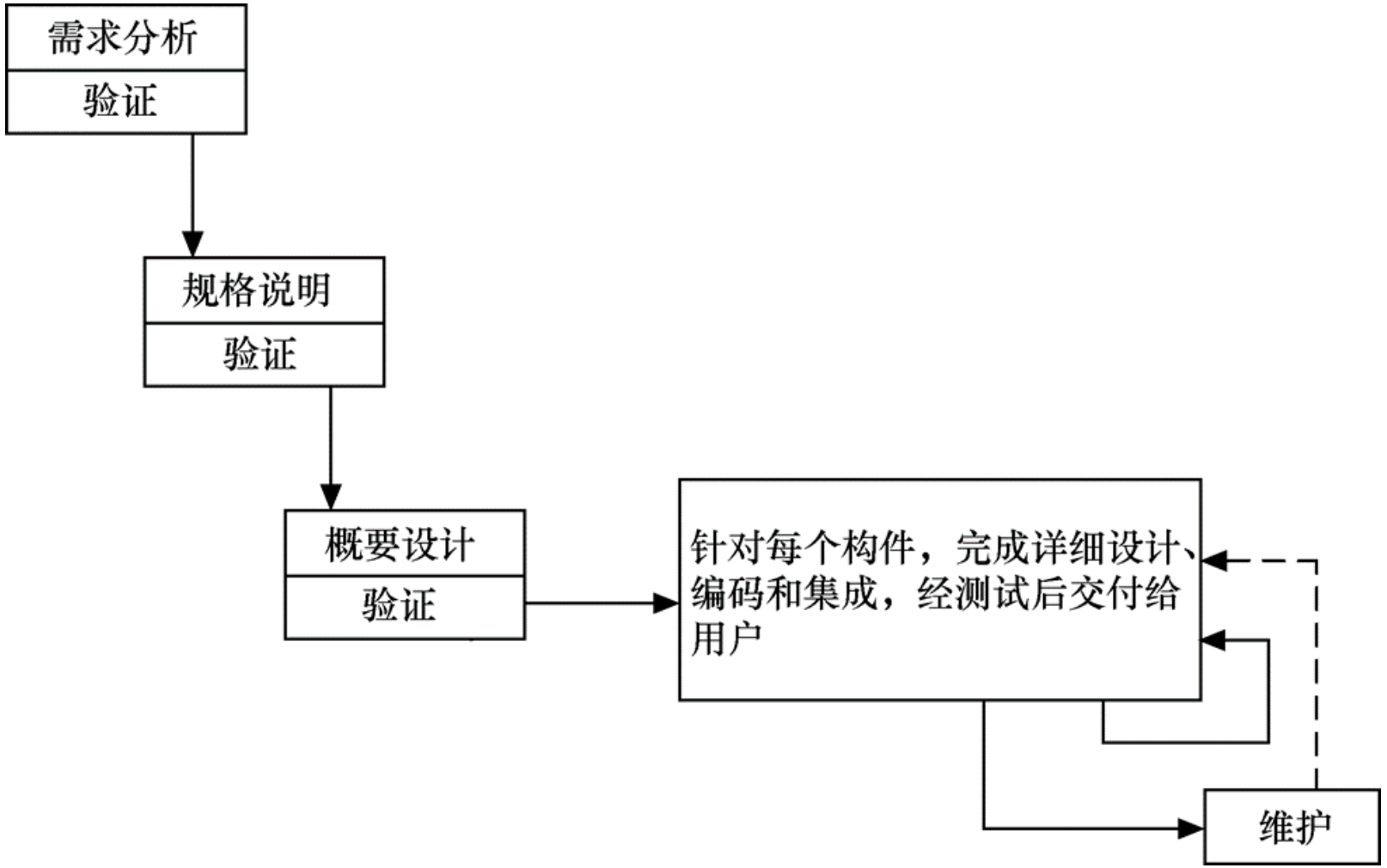
把软件产品作为一系列的增量构件来设计、编码、集成和测试

每个构件由多个相互作用的模块构成，并且能够完成特定的功能

使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能

把软件产品分解成增量构件时，唯一必须遵守的约束条件是，当把新构件集成到现有构件中时，所形成的产品必须是可测试的

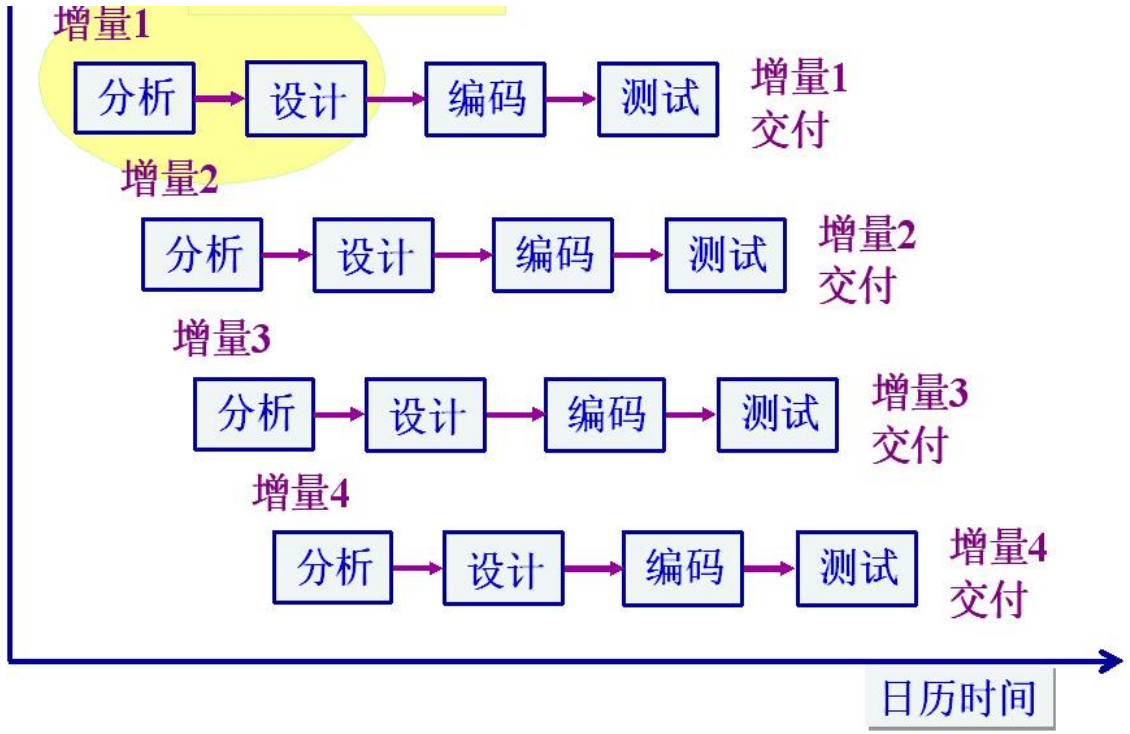
增量模型



增量模型的优点

1. 能在较短时间内向用户提交可完成一些有用的工作产品，即从第1个构件交付之日起，用户就能做一些有用的工作
2. 逐步增加产品的功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给用户组织带来的冲击
3. 项目失败的风险较低，虽然在某些增量构件中可能遇到一些问题，但其他增量构件将能够成功地交付给客户
4. 优先级最高的服务首先交付，然后再将其他增量构件逐次集成进来。因此，最重要的系统服务将接受最多的测试

增量构件开发


增量1
分析
设计
编码
测试
增量1
交付
增量2
分析
设计
编码
测试
增量2
交付
增量3
分析
设计
编码
测试
增量3
交付
增量4
分析
设计
编码
测试
增量4
交付
日历时间

以下情况可以使用增量模型

1. 需要尽短的时间内得到系统基本功能的演示或使用
2. 个版本都有中间阶段产品可提供使用
3. 系统可以被自然地分割成渐增模式
4. 开发人员与资金可以逐步增加

采用增量模型需注意的问题

1. 在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品
2. 软件体系结构必须是开放的，即向现有产品中加入新构件的过程必须简单、方便

以下情况慎重使用增量模型

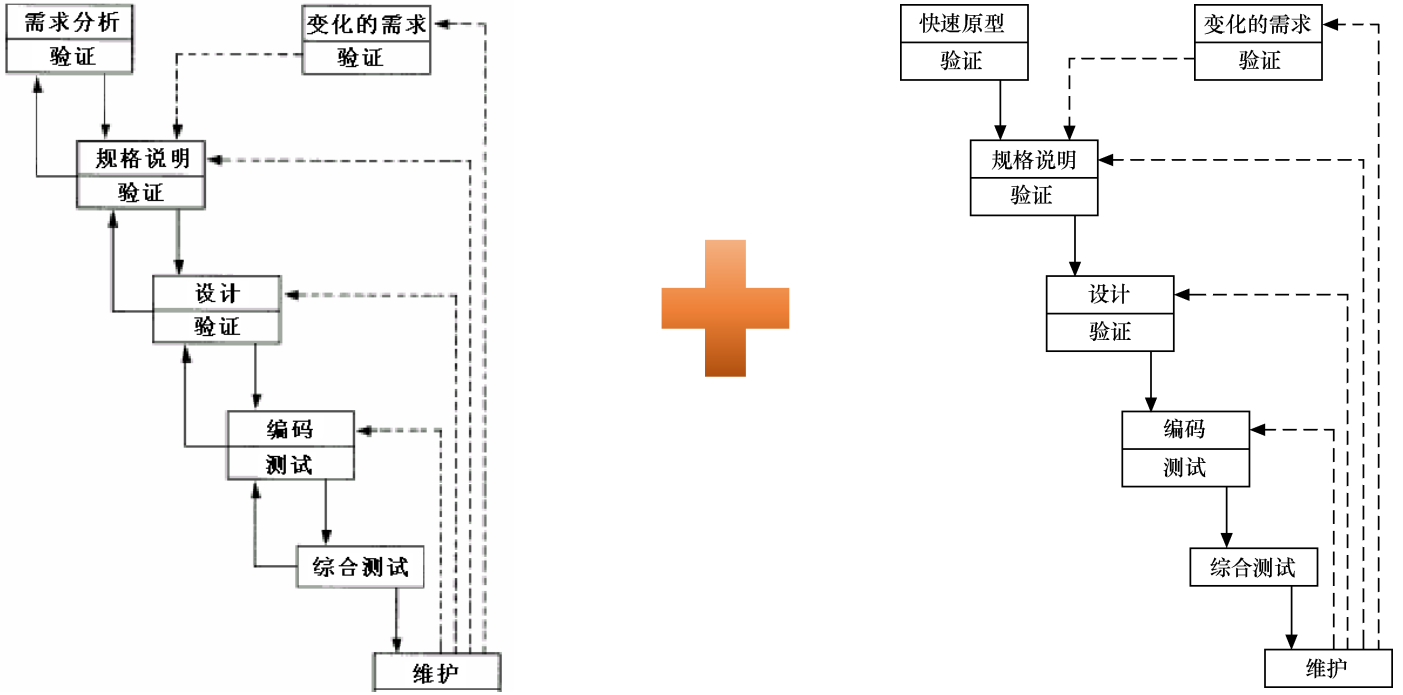
1. 不能充分理解客户需求或客户需求有可能迅速发生变化
2. 事先拟采用的技术迅速发生变化
3. 客户突然提出一些新的功能需求
4. 长时期内仅有有限的资源保证（开发人员和资金）

螺旋模型

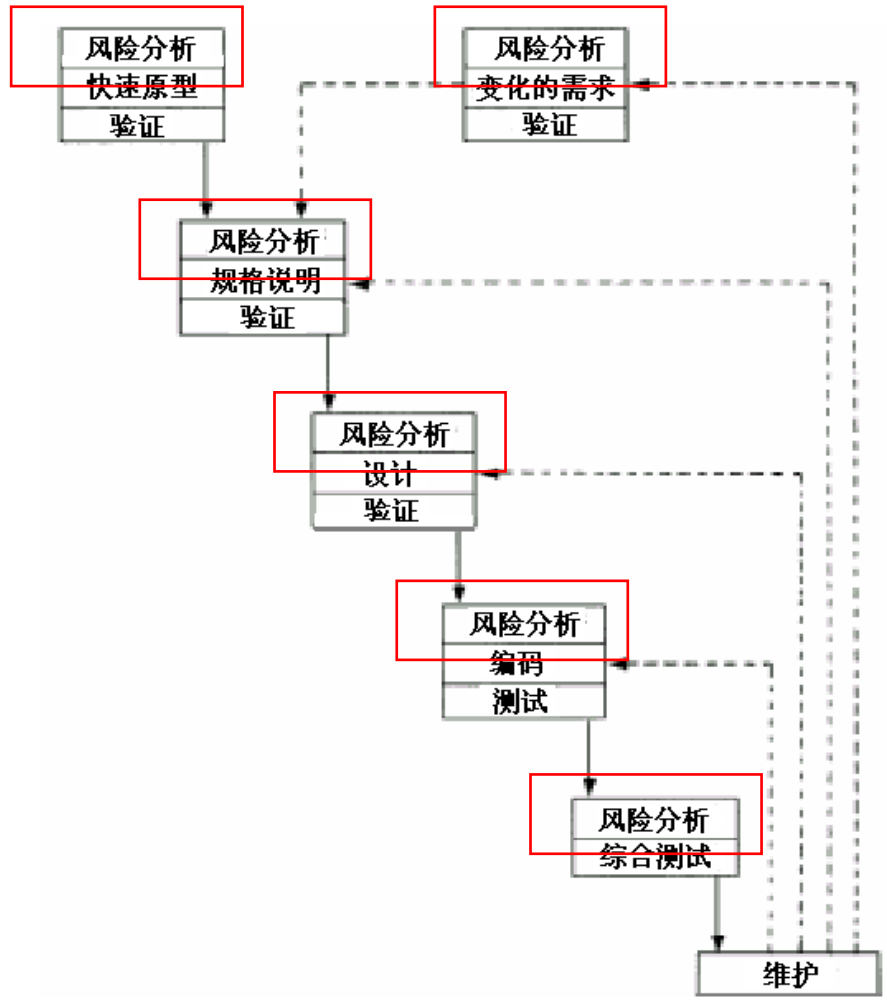
螺旋模型最初是Boehm于1988年提出来的

该模型将瀑布模型与快速原型模型结合起来，并且加入两种模型均忽略了的风险分析

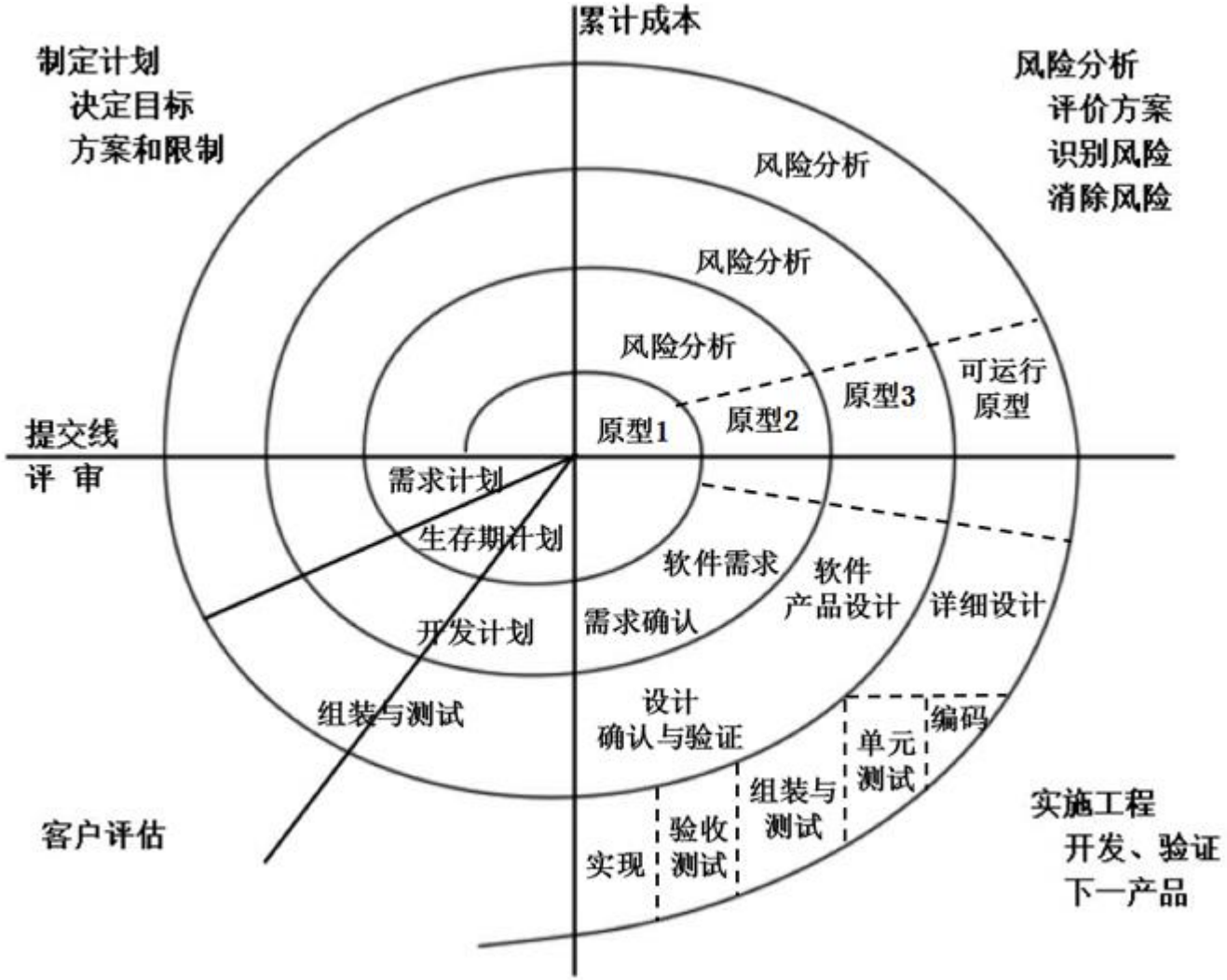
螺旋模型的基本思想是，使用原型及其他方法来尽量降低风险



螺旋模型

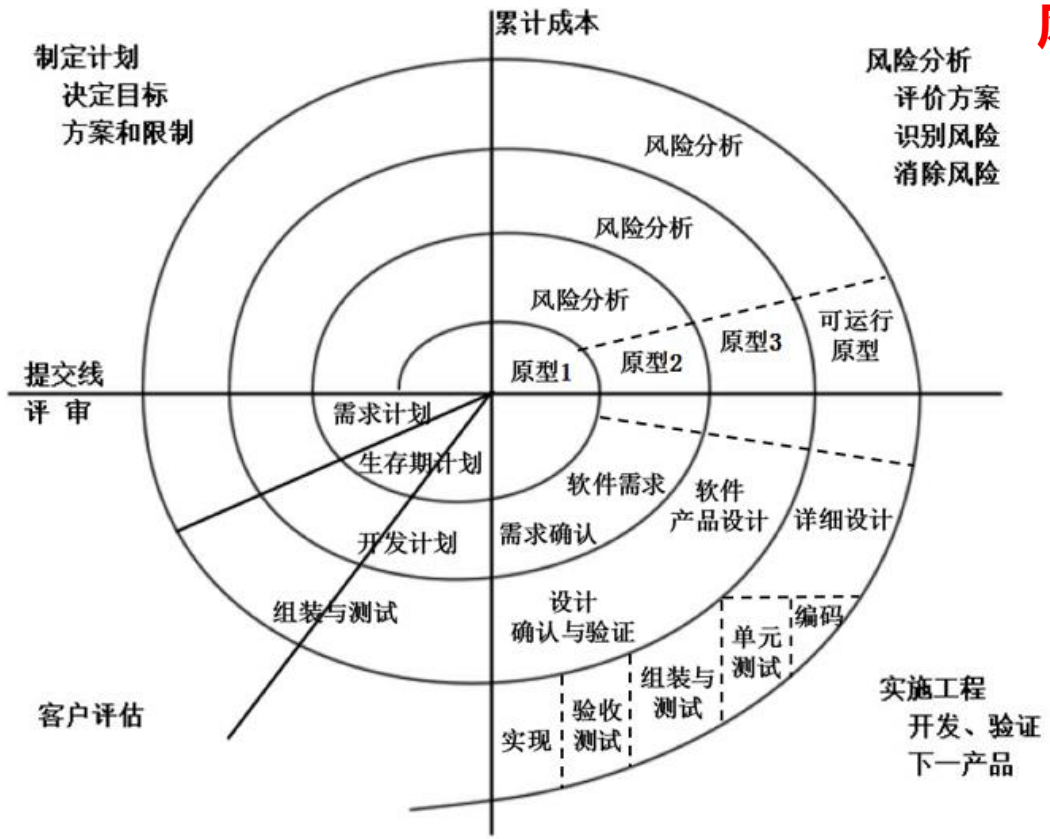

风险分析
快速原型
验证
风险分析
变化的需求
验证
风险分析
规格说明
验证
风险分析
设计
验证
风险分析
编码
测试
风险分析
综合测试
维护

完整的螺旋模型


累计成本
制定计划
决定目标
方案和限制
风险分析
评价方案
识别风险
消除风险
风险分析
风险分析
风险分析
可运行
原型
原型3
原型2
原型1
提交线
评 审
需求计划
生存期计划
开发计划
组装与测试
客户评估
软件需求
需求确认
软件
产品设计
详细设计
设计
确认与验证
编码
单元
测试
组装与
测试
验收
测试
实现
实施工程
开发、验证
下一产品

螺旋模型的4项活动

螺线上的每一个循环可划分为4个象限，分别表达了4个方面的活动



风险估计与弱化

目标设定

开发与验证

计划

螺旋模型的优点

1. 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标
2. 减少了过多测试或测试不足所带来的风险
3. 在螺旋模型中维护只是模型的另一个周期，因而在维护和开发之间并没有本质区别

螺旋模型的缺点

螺旋模型是风险驱动的，因此要求软件开发人员必须具有丰富的风险评估经验和这方面的专门知识，否则将出现真正的风险：**当项目实际上正在走向灾难时，开发人员可能还以为一切正常**

统一过程
Unified Process，UP
Rational Unfied Process, RUP

统一过程尝试着从传统的软件过程中挖掘最好的特征和性质，但是以敏捷软件开发中许多最好的原则来实现

统一过程认识到与客户沟通以及从用户的角度描述系统并保持改描述的一致性的重要性

强调软件体系结构的重要作用，并帮助架构师专注与正确的目标，例如可理解性、对未来改变的可适应性及复用

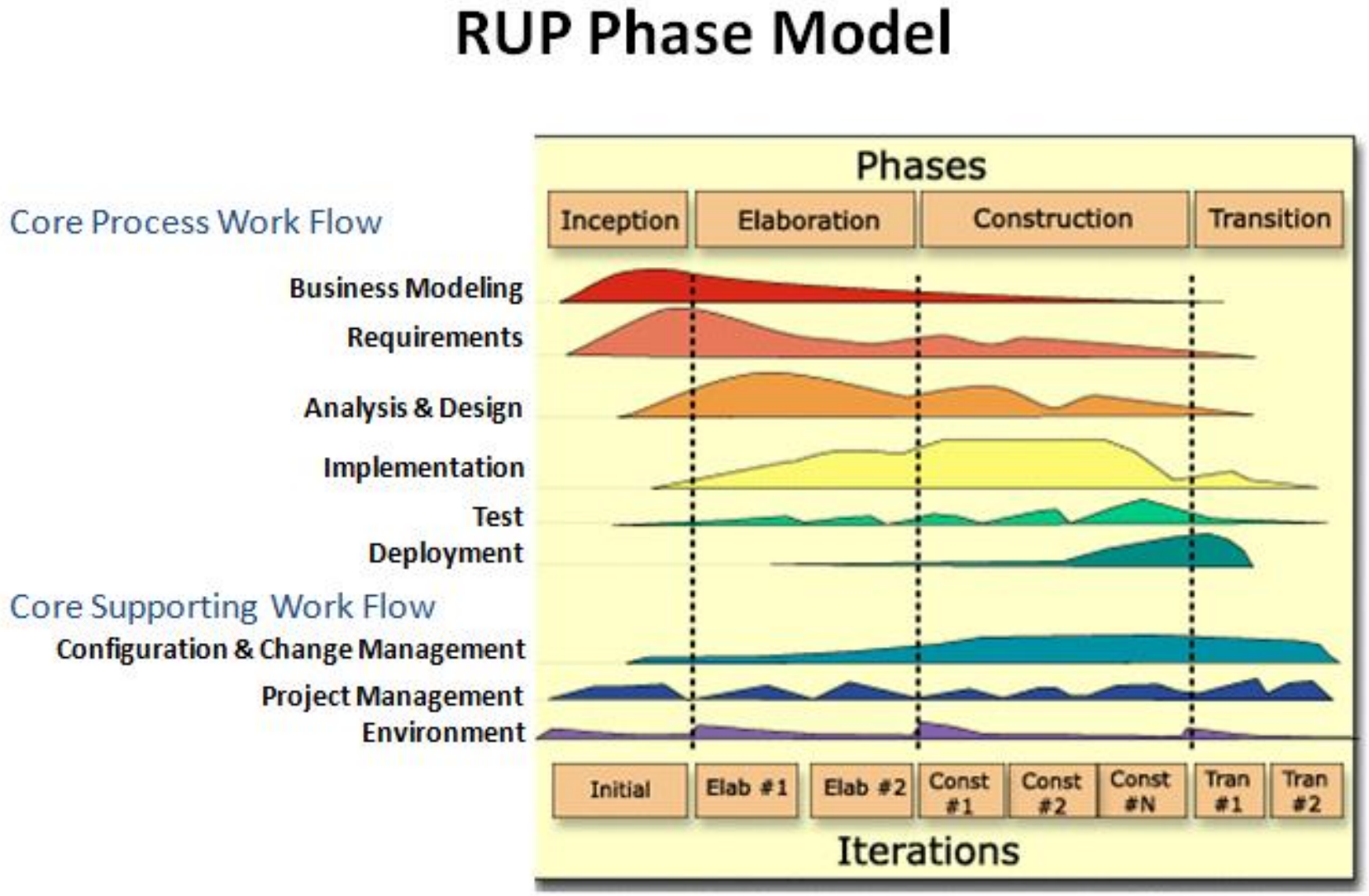
建立了迭代、增量的过程流，提供了演进的特征

统一过程是用UML进行面向对象软件工程的框架

统一过程的工作流

有6个核心工

1. 业务建模
2. 需求工作，需明确系统的功能需求
3. 分析和设
4. 实现工作行单元测试，将构件集成
5. 测试工作所有需求、查错并改正
6. 部署工作流：制作软件的外部版本、软件打包、分发、为用户提供帮助和支持



统一过程的阶段

统一过程有5个阶段，分别：

1. **起始阶段**（inception phase），包括客户沟通和策划活动。改阶段识别基本的业务需求，并用用例初步描述每一类用户所需要的主要特征和功能。策划阶段将识别各种资源，评估主要风险，并为软件增量制定初步的进度计划表
2. **细化阶段**（elaboration phase），包括沟通和模型的建模活动。该阶段识扩展了初始阶段定义的用例，并创建了体系结构基线以包括软件的5种视图——用例模型、分析模型、设计模型、实现模型和部署模型。该阶段通常要对项目计划进行修订

统一过程的阶段

3. **构建阶段**（construction phase），软件增量（例如发布的版本）所要求的必须具备的特征和功能在源代码中实现。随着构件的实现，对每一个构件设计并实施单元测试。另外，还实施了其他集成活动（构件组装和集成测试）
4. **转换阶段**（transition phase），交付和反馈活动的一部分，软件被提交给最终用户进行Beta测试，用户反馈报告缺陷及必要的变更。在转换阶段结束时，软件增量成为可用的发部版本
5. **生产阶段**（production phase），在该阶段对持续使用的软件进行监控，提供运行环境（基础设施）的支持，提交并评估缺陷报告和变更请求

统一过程的主要工作产品

起始阶段

愿景文档
初始用例模型
初始项目术语表
初始风险评估
项目计划
　阶段及迭代
业务模型
　如果需要一个或
　多个原型

细化阶段

用例模型
补充需求
　包括非功能需求
分析模型
软件体系结构描述
可执行的体系结构原型
初步的设计模型
修订的风险列表
项目计划，包括
　迭代计划
　调整的工作流
　里程碑
　技术工作产品
初始用户手册

构建阶段

设计模型
软件构件
集成的软件增量
测试计划及步骤
测试用例
支持文档
　用户手册
　安装手册
　对于并发增量的描述

转化阶段

提交软件增量
Beta测试报告
综合用户反馈

敏捷过程模型

2001年，Kent Beck等17名编程大师发表“敏捷软件开发”宣言：

我们正在通过亲身实践以及帮助他人实践的方式来揭示更好的软件开发之路，通过这项工作，我们认为：

1. **个体和交互胜过过程和工具**
2. **可工作软件胜过宽泛的文档**
3. **客户合作胜过合同谈判**
4. **响应变化胜过遵循计划**

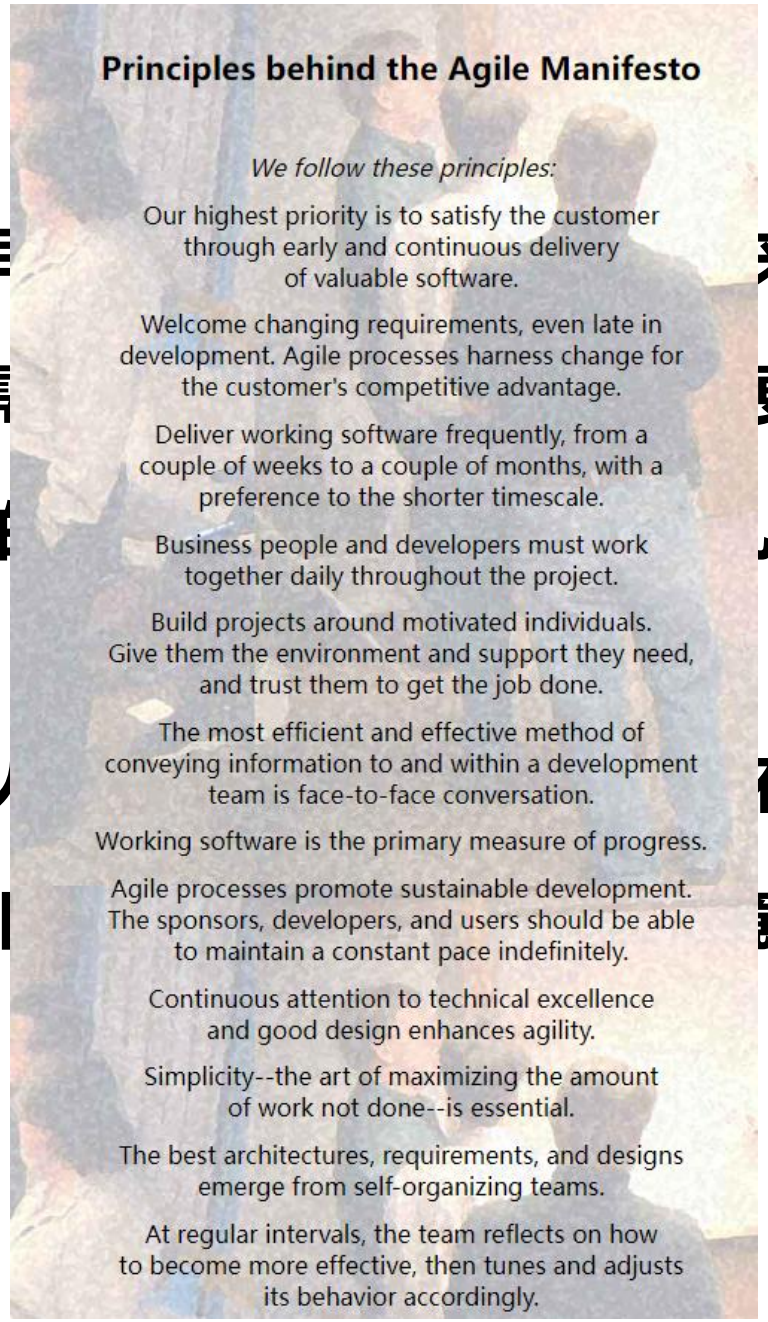
敏捷过程模型

任何一个敏捷过程都可以由所强调的3个关键假设识别出来，这3个假设可适用于大多数软件项目：

1. 提前预测哪些需求是稳定的、哪些需求会变化非常困难。同样的，预测项目进行中客户优先级的变化也很困难
2. 对很多软件，设计和构建是交错进行的。事实上，两种活动应当顺序开展以保证通过构建实施来验证设计模型，而在通过构建验证之前很难估计应该设计到什么程度
3. 从制订计划的角度来看，分析、设计、构建和测试并不像我们所设想的那么容易预测

敏捷原则

1. 我们最优先要做的是通过尽早
来使客户满意
2. 即使在开发的后期，也欢迎需
更为客户创造竞争优势
3. 经常交付可运行软件，交付的
个月，交付的时间间隔越短越好
4. 在整个项目开发期间，业务人
在一起工作
5. 围绕有积极性的个人构建项目
和支持，并且信任他们能够完成工作



敏捷原则

7. 在团队内部，最富有效果和效率的信息传递方法是面对面交谈

8. 可运行软件是进度的首要度量标准

9. 敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够保持一种长期、稳定的开发速度

10. 不断地关注优秀的技能和好的设计会增强敏捷能力

11. 简单是必要的

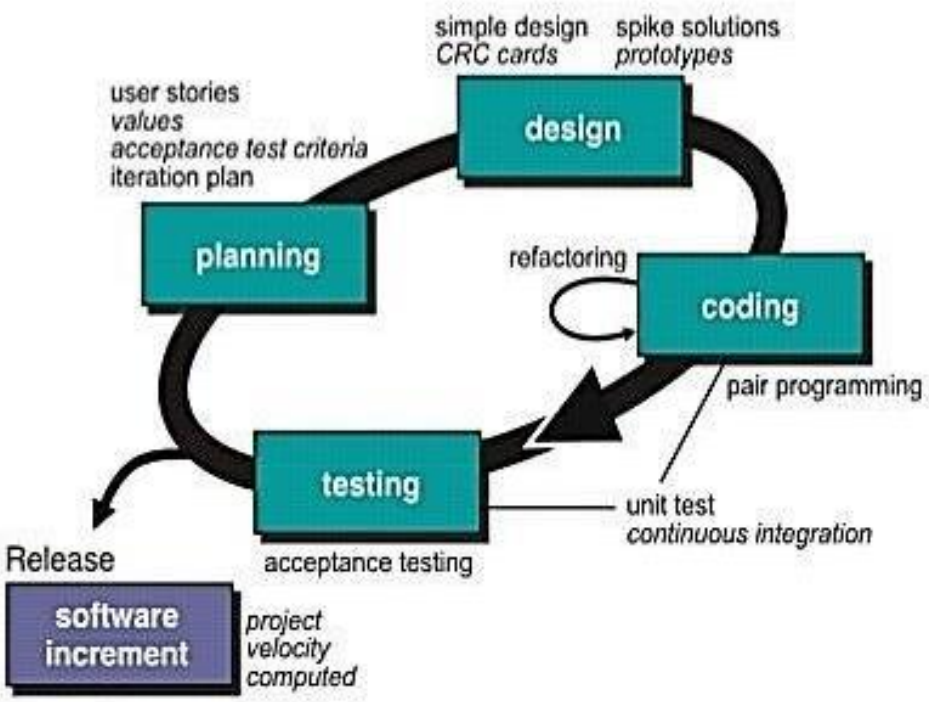
12. 好的架构、需求和设计出自于自组织团队

13. 每隔一定时间，团队会反省如何才能更有效地工作，并相应调整自己的行为

极限编程（eXtreme Programming，XP）

使用最广泛的敏捷过程，最初由Kent Beck提出

XP包含了策划、设计、编码和测试4个框架活动的规则和实践



策划

1. 开始于建立“用户故事”
2. 敏捷团队评估每一个故事并给出成本（cost）
3. 故事被分组用于可交付增量
4. 对发布日期做出承诺
5. 在第一个发行版本（软件增量）之后，“项目速度” 用于帮助建立后续发行版本（软件增量）的发布日期

设计

1. XP设计严格遵循保持简洁（keep it simple， KIS）原则，通常更愿意使用简单设计而不是更为复杂的表述
2. 鼓励使用类-责任-协作者（Class-Responsibility-Collaborator ,CRC ）卡，作为面向对象环境中识别和组织类的有效机制
3. 在设计中遇到困难，XP推荐立即建立这部分设计的可执行原型，实现并评估设计原型
4. 鼓励“重构”——种迭代的改进内部程序设计的方法

编码

1. XP建议在 起单元测试，开发者就 旦编码完成，就可以立
2. 鼓励 “结 人面对同一台计算机 实时质量保证的机制

测试

1. 所建立的单元测试应当使用一个可以自动实施的框架（易于执行和可重复），这种方式支持每当代码修改（经常发生）之后就实现一次回归测试策略
2. “验收测试” 由客户定义，将着眼于客户可见的、可评审的系统级的特征和功能

软件开发模型的选择

在软件开发过程中，使开发工作受到制约和影响的主要因素在于项目性质和软件开发环境

应对待软件开发的规模、类型、质量要求、交付工期、需求明确度、投资、可复用性及开发者掌握的资源等情况进行重点分析

这些内容反映了项目软件的基本特性以及开发环境对开发过程的基本影响

软件开发模型选择的因素

1. 软件规模
2. 软件类型
3. 软件质量要求
4. 交付工期
5. 客户需求明确度
6. 投资
7. 可复用性
8. 开发者掌握的资源

/-2 复习
Review

软件开发过程模型

软件开发过程模型能清晰直观的表达开发全过程，明确规定的要完成的主要活动和任务，可以作为软件项目开发的基础

对于一个具体软件系统来说，应采用合适的开发方法，使用事宜的程序设计语言，组织具有相应技能的开发人员参与，并采用适合的管理方法和手段，对整个开发活动的全过程进行有效的控制

经典的软件开发过程模型

瀑布模型

快速原型模型

增量模型

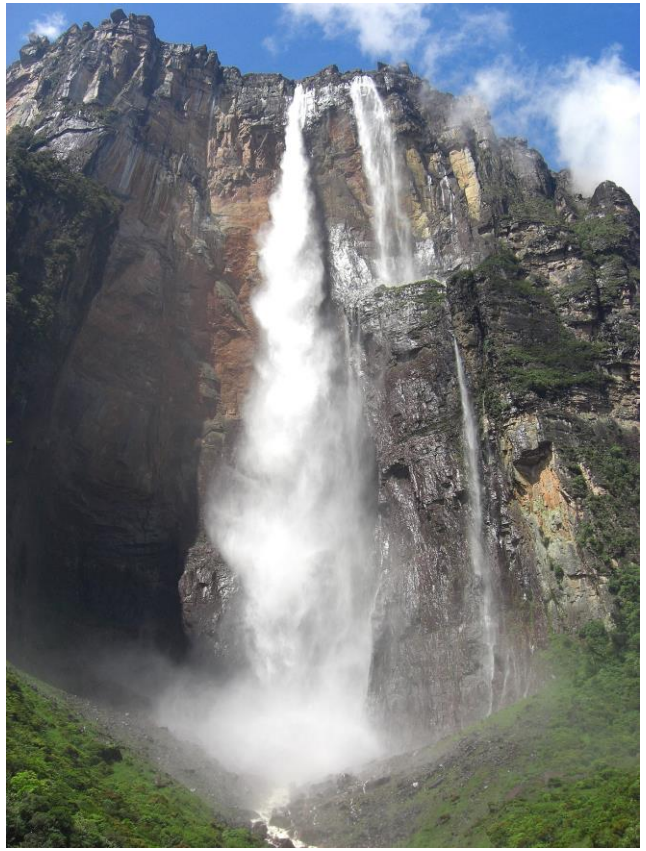
螺旋模型

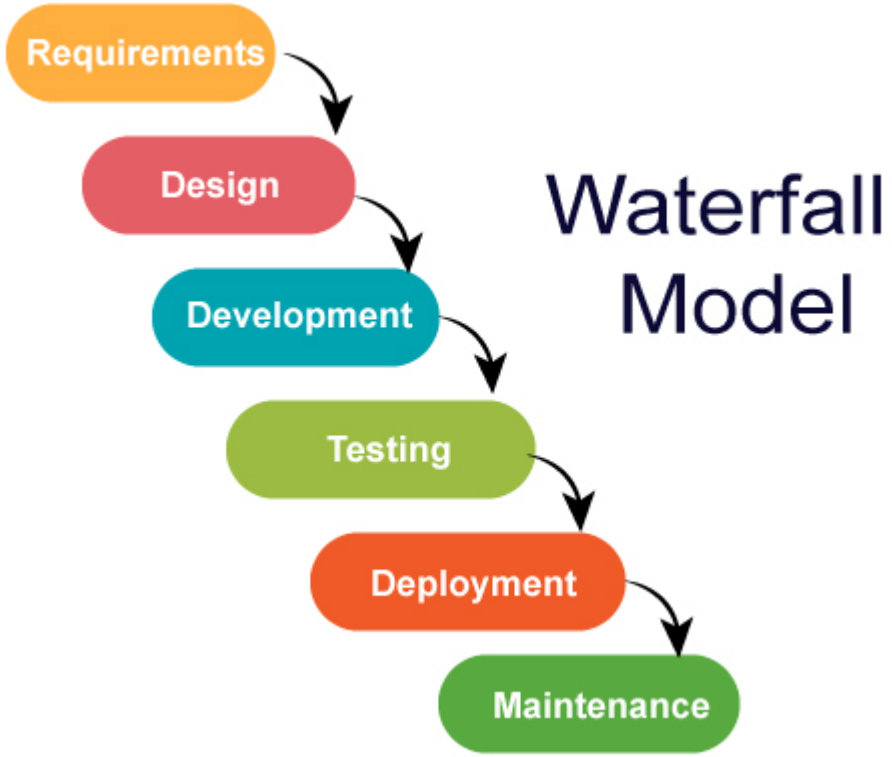
统一过程

敏捷过程

瀑布模型（waterfall model）

在20世纪80年代之前，瀑布模型一直是唯一被广泛采用的生命周期模型，也叫线性顺序模型（linear sequential model），是一个系统的、顺序的软件开发方法





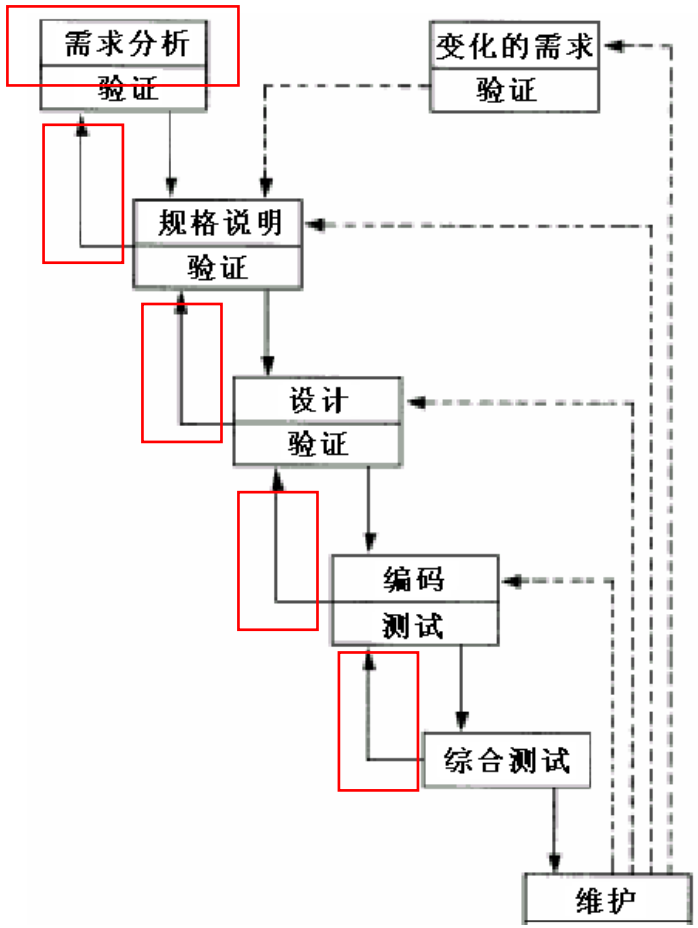
瀑布模型优点

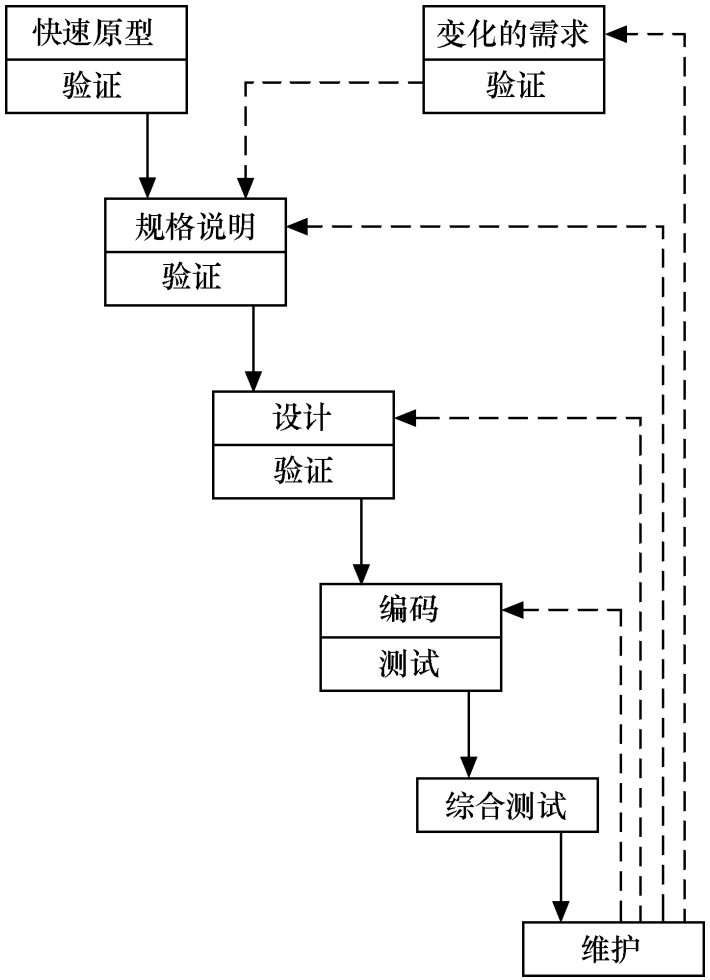
1. 可强迫开发人员采用规范化的方法。
2. 严格地规定了每个阶段必须提交的文档。
3. 要求每个阶段交出的所有产品都必须是经过验证的

瀑布模型缺点

1. 由于瀑布模型几乎完全依赖于书面的规格说明，很可能导致最终开发出的软件产品不能真正满足用户的需要。如果需求规格说明与用户需求之间有差异，就会发生这种情况
2. 瀑布模型只适用于项目开始时需求已确定的情况

快速原型模型


需求分析
验证
变化的需求
验证
规格说明
验证
设计
验证
编码
测试
综合测试
维护


快速原型
验证
变化的需求
验证
规格说明
验证
设计
验证
编码
测试
综合测试
维护

快速原型模型优点

1. 有助于满足用户的真实需求
2. 原型系统已经通过与用户的交互而得到验证，据此产生的规格说明文档能够正确地描述用户需求
3. 软件产品的开发基本上是按线性顺序进行
4. 因为规格说明文档正确地描述了用户需求，因此，在开发过程的后续阶段不会因为发现规格说明文档的错误而进行较大的返工

快速原型模型优点

5. 开发人员通过建立原型系统已经学到了许多东西，因此，在设计和编码阶段发生错误的可能性也比较小，这自然减少了在后续阶段需要改正前面阶段所犯错误的可能性
6. 快速原型的突出特点是“快速”。开发人员应该尽可能快地建造出原型系统，以加速软件开发过程，节约软件开发成本

增量模型

增量模型也称为渐增模型（incremental model），是Mills等于1980年提出来的

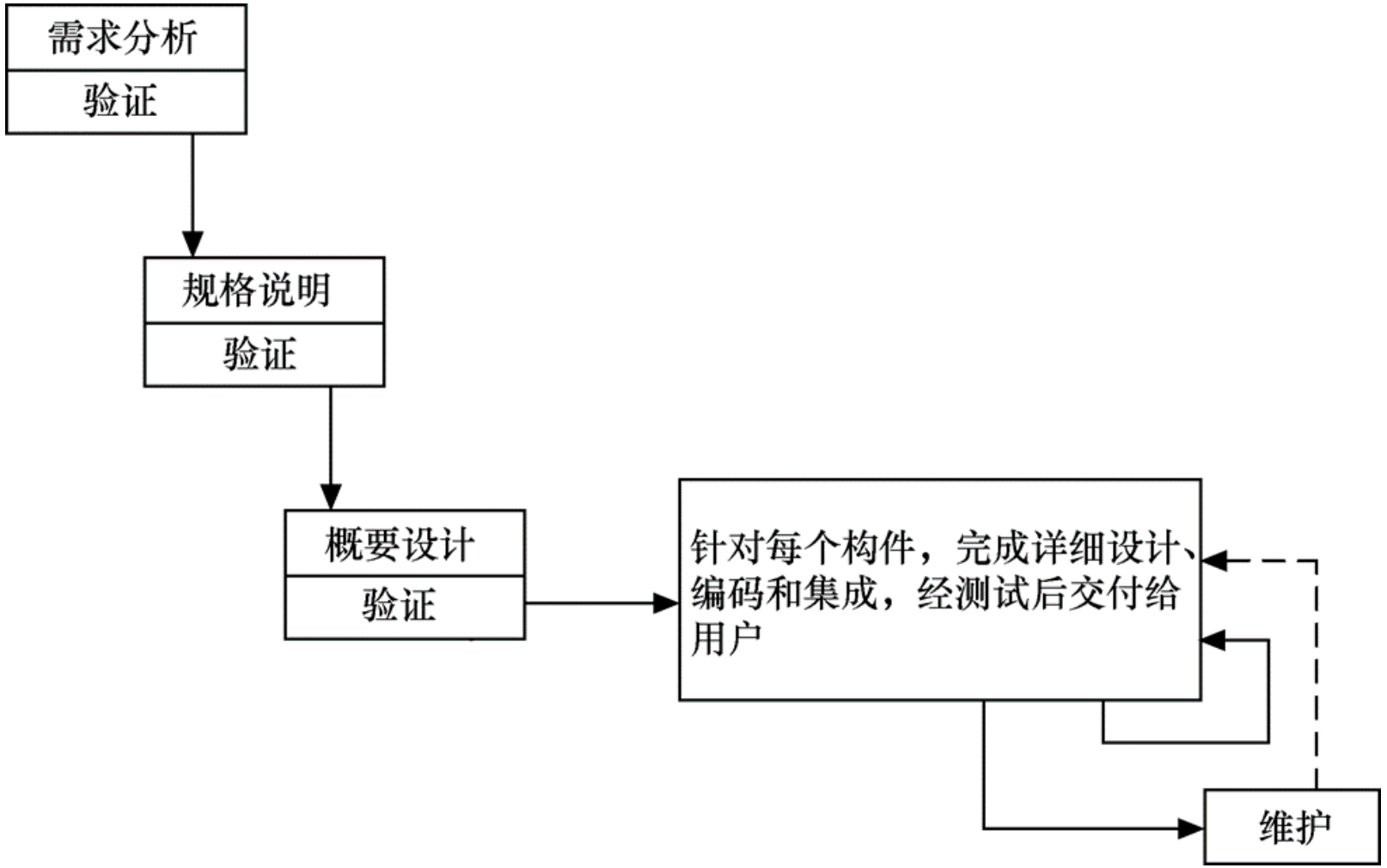
把软件产品作为一系列的增量构件来设计、编码、集成和测试

每个构件由多个相互作用的模块构成，并且能够完成特定的功能

使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能

把软件产品分解成增量构件时，唯一必须遵守的约束条件是，当把新构件集成到现有构件中时，所形成的产品必须是可测试的

增量模型



增量模型的优点

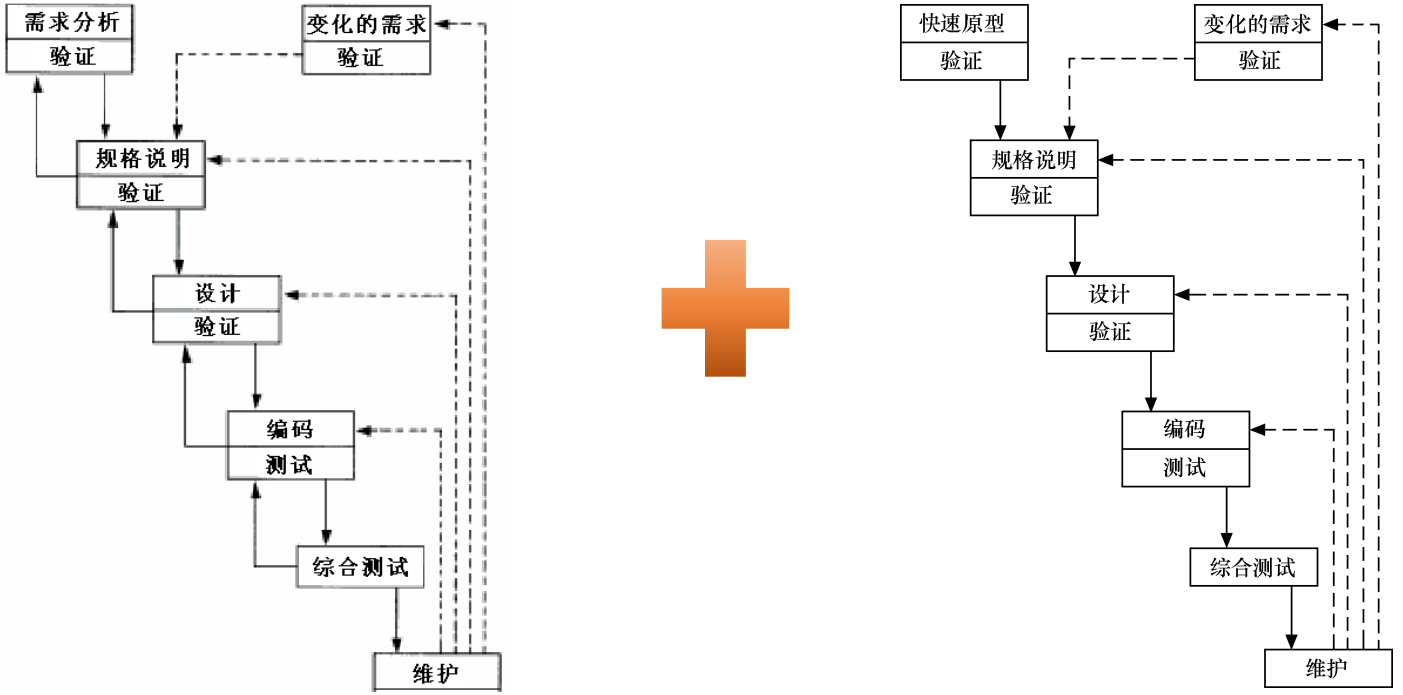
1. 能在较短时间内向用户提交可完成一些有用的工作产品，即从第1个构件交付之日起，用户就能做一些有用的工作
2. 逐步增加产品的功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给用户组织带来的冲击
3. 项目失败的风险较低，虽然在某些增量构件中可能遇到一些问题，但其他增量构件将能够成功地交付给客户
4. 优先级最高的服务首先交付，然后再将其他增量构件逐次集成进来。因此，最重要的系统服务将接受最多的测试

螺旋模型

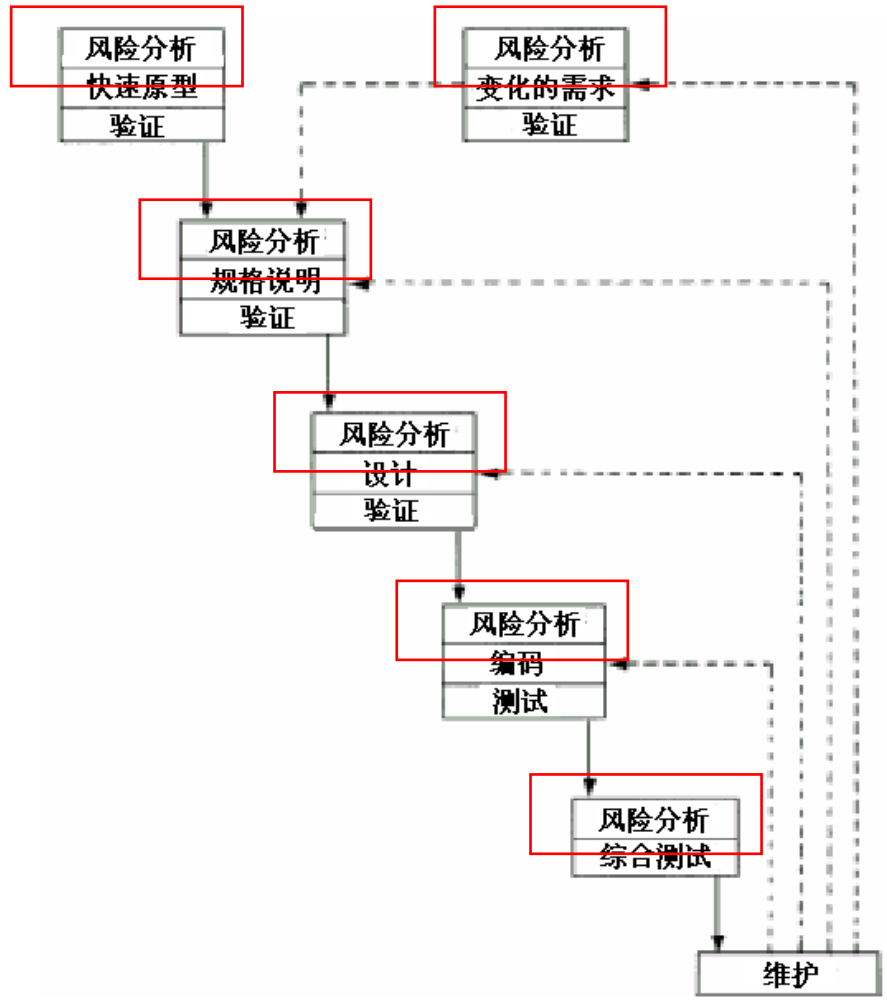
螺旋模型最初是Boehm于1988年提出来的

该模型将瀑布模型与快速原型模型结合起来，并且加入两种模型均忽略了的风险分析

螺旋模型的基本思想是，使用原型及其他方法来尽量降低风险



螺旋模型


风险分析
快速原型
验证
风险分析
变化的需求
验证
风险分析
规格说明
验证
风险分析
设计
验证
风险分析
编码
测试
风险分析
综合测试
维护

螺旋模型的优点

1. 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标
2. 减少了过多测试或测试不足所带来的风险
3. 在螺旋模型中维护只是模型的另一个周期，因而在维护和开发之间并没有本质区别

螺旋模型的缺点

螺旋模型是风险驱动的，因此要求软件开发人员必须具有丰富的风险评估经验和这方面的专门知识，否则将出现真正的风险：**当项目实际上正在走向灾难时，开发人员可能还以为一切正常**

统一过程
Unified Process，UP
Rational Unfied Process, RUP

统一过程尝试着从传统的软件过程中挖掘最好的特征和性质，但是以敏捷软件开发中许多最好的原则来实现

统一过程认识到与客户沟通以及从用户的角度描述系统并保持改描述的一致性的重要性

强调软件体系结构的重要作用，并帮助架构师专注与正确的目标，例如可理解性、对未来改变的可适应性及复用

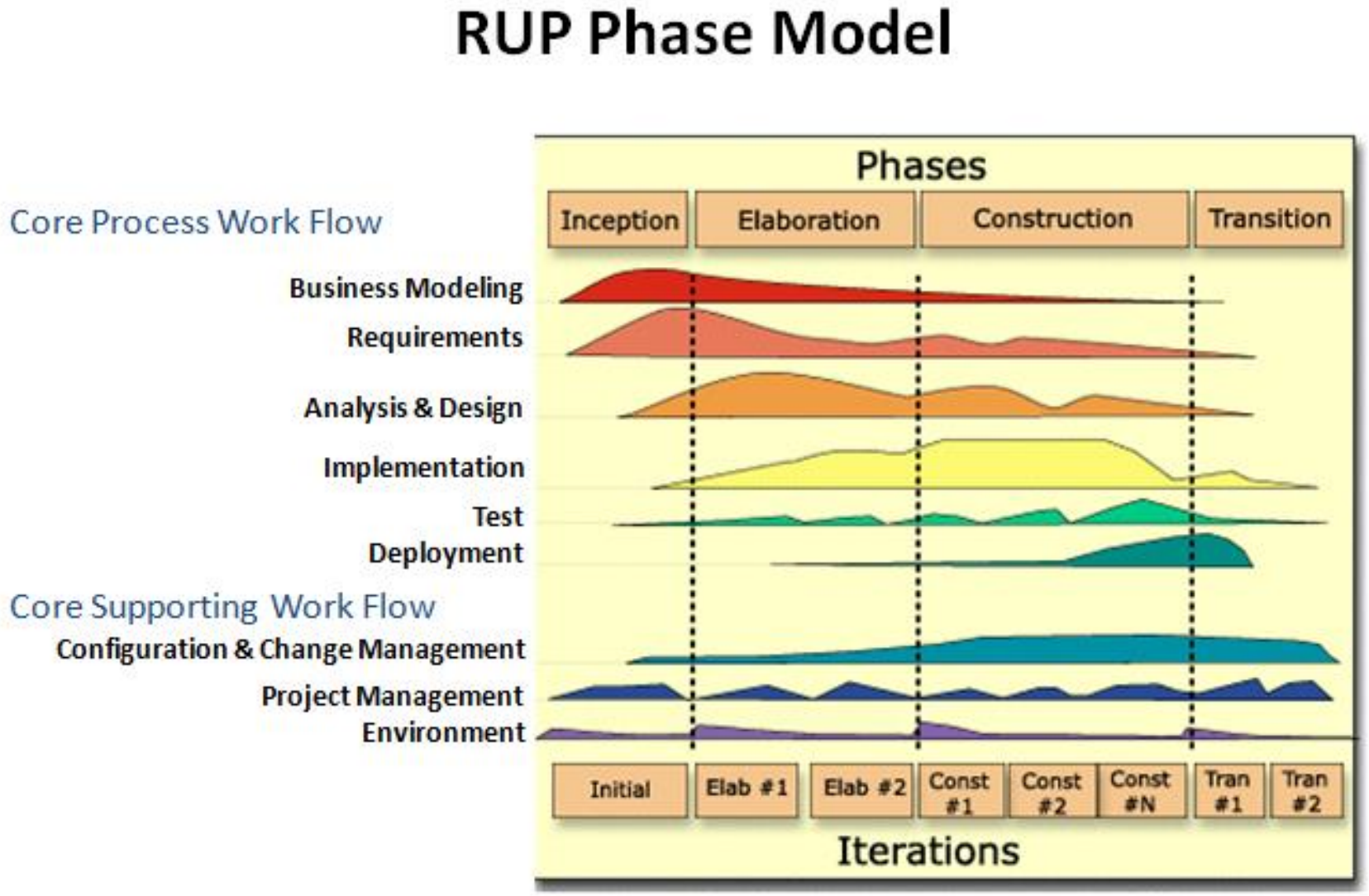
建立了迭代、增量的过程流，提供了演进的特征

统一过程是用UML进行面向对象软件工程的框架

统一过程的工作流

有6个核心工

1. 业务建模
2. 需求工作，需明确系统的功能需求
3. 分析和设
4. 实现工作行单元测试，将构件集成
5. 测试工作所有需求、查错并改正
6. 部署工作流：制作软件的外部版本、软件打包、分发、为用户提供帮助和支持



统一过程的阶段

统一过程有5个阶段，分别：

1. **起始阶段**（inception phase），包括客户沟通和策划活动。改阶段识别基本的业务需求，并用用例初步描述每一类用户所需要的主要特征和功能。策划阶段将识别各种资源，评估主要风险，并为软件增量制定初步的进度计划表
2. **细化阶段**（elaboration phase），包括沟通和模型的建模活动。该阶段识扩展了初始阶段定义的用例，并创建了体系结构基线以包括软件的5种视图——用例模型、分析模型、设计模型、实现模型和部署模型。该阶段通常要对项目计划进行修订

统一过程的阶段

3. **构建阶段**（construction phase），软件增量（例如发布的版本）所要求的必须具备的特征和功能在源代码中实现。随着构件的实现，对每一个构件设计并实施单元测试。另外，还实施了其他集成活动（构件组装和集成测试）
4. **转换阶段**（transition phase），交付和反馈活动的一部分，软件被提交给最终用户进行Beta测试，用户反馈报告缺陷及必要的变更。在转换阶段结束时，软件增量成为可用的发部版本
5. **生产阶段**（production phase），在该阶段对持续使用的软件进行监控，提供运行环境（基础设施）的支持，提交并评估缺陷报告和变更请求

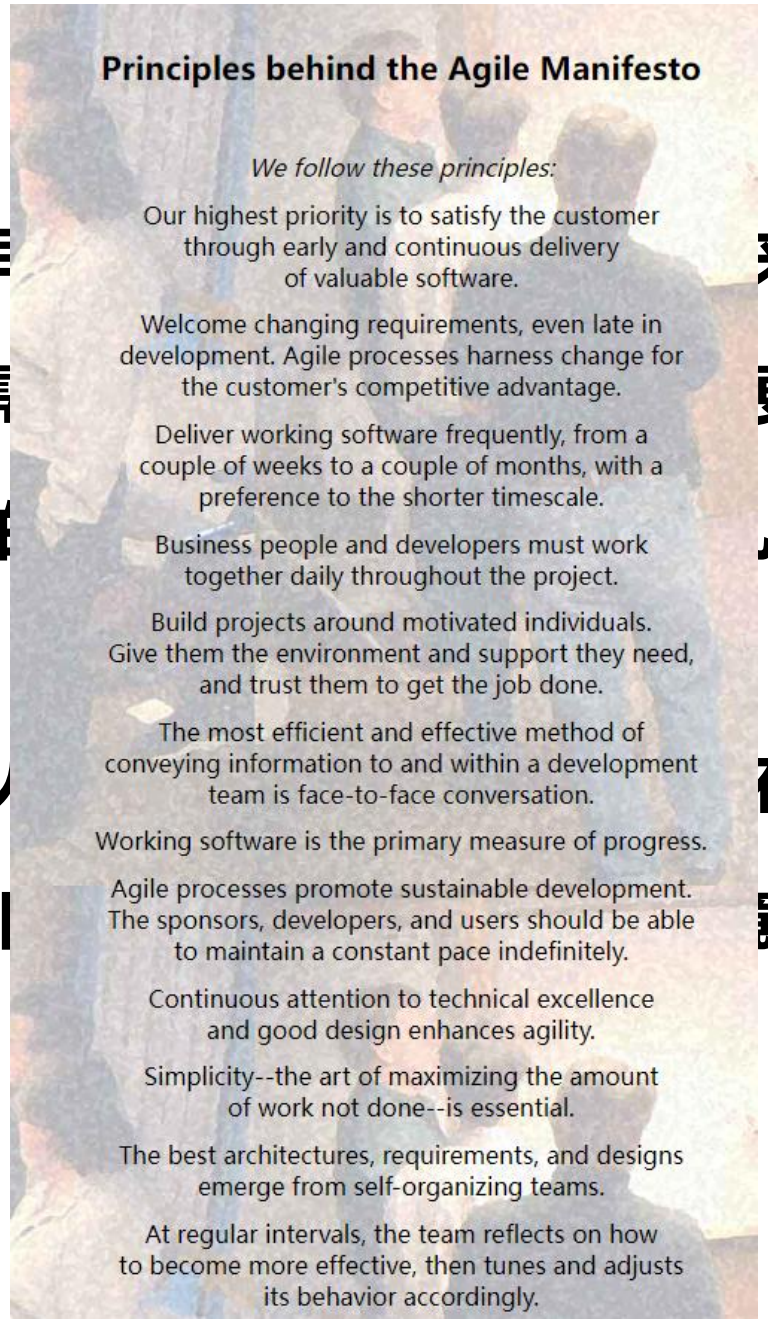
敏捷过程模型

任何一个敏捷过程都可以由所强调的3个关键假设识别出来，这3个假设可适用于大多数软件项目：

1. 提前预测哪些需求是稳定的、哪些需求会变化非常困难。同样的，预测项目进行中客户优先级的变化也很困难
2. 对很多软件，设计和构建是交错进行的。事实上，两种活动应当顺序开展以保证通过构建实施来验证设计模型，而在通过构建验证之前很难估计应该设计到什么程度
3. 从制订计划的角度来看，分析、设计、构建和测试并不像我们所设想的那么容易预测

敏捷原则

1. 我们最优先要做的是通过尽早
来使客户满意
2. 即使在开发的后期，也欢迎需
更为客户创造竞争优势
3. 经常交付可运行软件，交付的
个月，交付的时间间隔越短越好
4. 在整个项目开发期间，业务人
在一起工作
5. 围绕有积极性的个人构建项目
和支持，并且信任他们能够完成工作



敏捷原则

7. 在团队内部，最富有效果和效率的信息传递方法是面对面交谈

8. 可运行软件是进度的首要度量标准

9. 敏捷过程提倡可持续的开发速度。责任人、开发者和用户应该能够保持一种长期、稳定的开发速度

10. 不断地关注优秀的技能和好的设计会增强敏捷能力

11. 简单是必要的

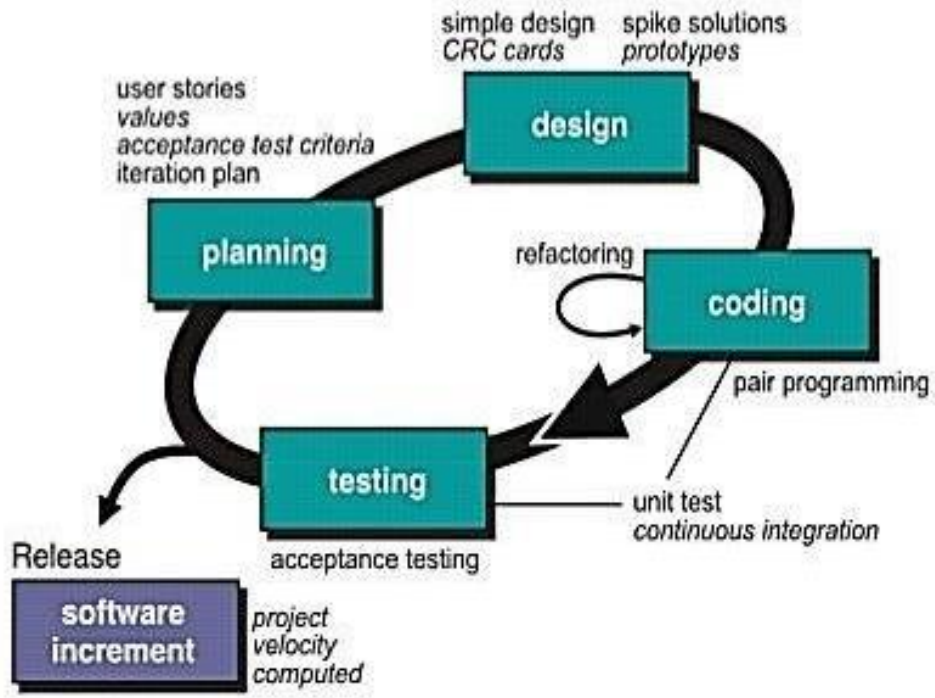
12. 好的架构、需求和设计出自于自组织团队

13. 每隔一定时间，团队会反省如何才能更有效地工作，并相应调整自己的行为

极限编程（eXtreme Programming，XP）

使用最广泛的敏捷过程，最初由Kent Beck提出

XP包含了策划、设计、编码和测试4个框架活动的规则和实践



软件开发模型的选择

在软件开发过程中，使开发工作受到制约和影响的主要因素在于项目性质和软件开发环境

应对待软件开发的规模、类型、质量要求、交付工期、需求明确度、投资、可复用性及开发者掌握的资源等情况进行重点分析

这些内容反映了项目软件的基本特性以及开发环境对开发过程的基本影响

软件开发模型选择的因素

1. 软件规模
2. 软件类型
3. 软件质量要求
4. 交付工期
5. 客户需求明确度
6. 投资
7. 可复用性
8. 开发者掌握的资源

本节内容

Readings

《 软件工程概论》，郑人杰，马素霞等著，第二章

《UML系统建模与系统分析》，刁成嘉著，第一章

《软件工程：实践者的研究方法》，罗杰S.普莱斯曼等著，第二章

Thanks