

1.3 软件生命周期

- 软件生命周期包含三个阶段：软件定义、软件开发及软件运行维护。

✓ **定义阶段** 又称为系统分析，由系统分析员负责完成。确定总体目标，确定可行性，功能需求，估计资源与成本，进度安排等。可以进一步划分为问题定义、可行性研究和需求分析。

✓ **开发阶段** 具体设计与实现前一时期定义的软件。通常包括总体设计，详细设计，编码与单元测试，综合测试。

✓ **运行维护阶段** 执行不同类型的维护活动，使软件持久地满足用户需求。

软件生命周期每个阶段的基本任务。

- 问题定义

回答“**要解决的问题是什么？**”

通过对客户的访问调查，写出关于问题性质、工程目标和工程规模的书面报告。

- 可行性研究

回答“**确定的问题有行得通的解决办法吗？**”

在较抽象的高层次上进行分析和设计过程，探索这个问题是否值得去解，是否有可行的解决办法。

- 需求分析

准确地回答“**为了解决这个问题，目标系统必须做什么？**”

确定目标系统必须具备哪些功能和性能需求，建立**需求分析模型**和**需求规格说明书**。

- 总体设计（又称概要设计）

回答“**概括地说，应该怎样实现目标系统？**”

描述多种方案，并在充分权衡各种方案的利弊的基础上，推荐一个最佳方案，**制定出实现最佳方案的详细计划**；**确定软件系统的体系结构**（确定程序由哪些模块组成以及模块间的关系）。

- 详细设计（又称模块设计）

回答“**应该怎样具体地实现这个系统呢？**”

把解决问题的办法具体化，设计出程序的**详细规格说明**。即详细地设计每个模块，确定实现模块功能所需要的算法和数据结构。

- 编码和单元测试

写出正确的容易理解、容易维护的**程序模块**。选取适当的程序设计语言，并对模块进行**测试**。

- 综合测试

回答“**实现的系统正确吗？是用户需要的吗？**”

通过各种类型的测试使软件达到预定的要求。通常进行**集成测试**和**验收测试**。保存测试计划、详细测试方案以及实际测试结果，作为软件配置的一个组成部分。

- 软件维护

通过各种必要的维护活动使系统持久地满足用户的需要，包括**改正性维护**，**适应性维护**，**完善性维护**，**预防性维护**。

1.4 软件过程

- 为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

✓ ISO 9000定义：软件过程是把**输入**转化为**输出**的一组彼此相关的**资源**和**活动**。

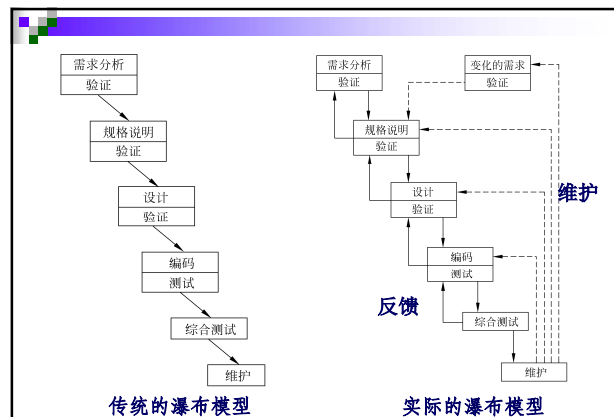
✓ 从软件开发的观点看，它就是使用适当的资源（包括人员、硬软件工具、时间等），为开发软件进行的一组开发活动，在过程结束时将输入（用户要求）转化为输出（软件产品）。

- 软件过程定义了：**方法**使用的顺序、要求交付的**文档资料**、为保证质量和适应变化所需要的**管理**、软件开发各个阶段完成的**里程碑**。
- 软件过程包含四种基本的过程活动 (包含在软件生命周期中):
 - ❖ **plan** : 软件规格说明
 - ❖ **do** : 软件开发
 - ❖ **check** : 软件确认
 - ❖ **action** : 软件演进

- 软件**生命周期模型**来描述**软件过程**。
- 为了简洁地描述软件过程,把总体设计和详细设计合并在一起称为“设计”;把**问题定义和可行性研究**归并到**需求分析**中;**规格说明**作为一个单独阶段。
- 常见的软件过程模型有:
 - 瀑布模型,快速原型模型,增量模型,螺旋模型
 - 敏捷过程,微软过程
 - 喷泉模型,Rational 统一过程

1. 瀑布模型

- 各项活动按自上而下,相互衔接的固定次序,如同瀑布逐级下落,每项活动均处于一个质量环(输入-处理-输出-评审)中。
- 阶段间具有**顺序性**和**依赖性**
- 推迟实现的观点
- 质量保证的观点
 - 每个阶段必须完成规定的文档
 - 每个阶段结束前完成文档审查



传统的瀑布模型

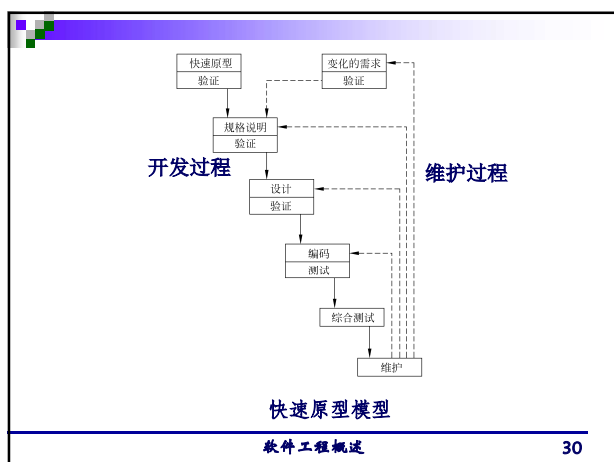
实际的瀑布模型

瀑布模型是一种文档驱动的模式。

- 优点:
 - ✓ 可强迫开发人员采用规范的方法;
 - ✓ 严格地规定了每个阶段必须提交的文档(可维护性);
 - ✓ 要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证;
- 缺点:
 - ✓ 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;
 - ✓ 缺乏灵活性。

2. 快速原型模型

- 所谓**快速原型模型**,就是快速建立起来的可以在计算机上运行的程序,它所能完成的功能往往是最终产品能完成的功能的一个子集。
- 快速原型模型的第一步是快速建立一个能反映用户主要需求的原型系统,让用户在计算机上试用它,通过实践来了解目标系统的概貌。**原型的用途是获知用户的真正需求。**
- 软件产品的开发基本上是**线性顺序进行**的,不带反馈环,这也是这种过程模型的主要优点。



快速原型模型是对瀑布模型的改进。主要优点是：有助于保证用户的真实需求得到满足。此外，

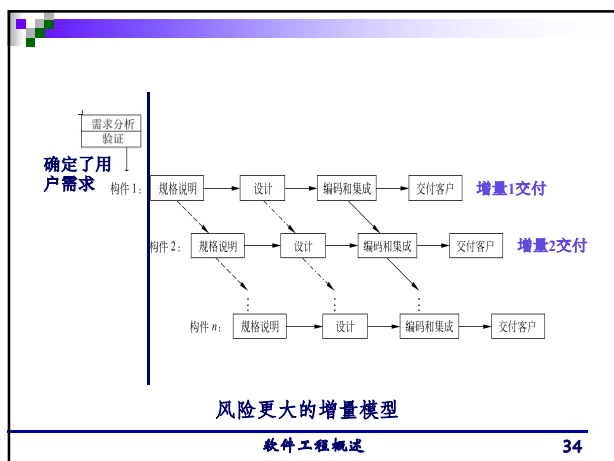
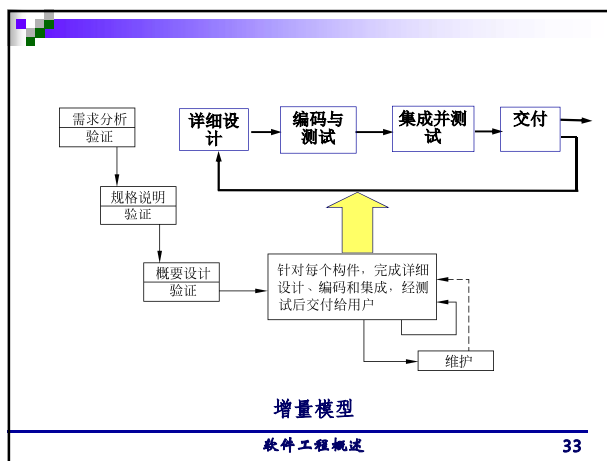
- ✓ 根据原型系统产生的规格说明文档正确地描述了用户需求，因此不存在较大的返工(线性的过程)。
- ✓ 开发人员通过建立原型系统已经学到了许多东西。
- ✓ 开发人员应该尽可能快地建造出原型系统，以加速软件开发过程，节约软件开发成本(体现快速的本质)。
- ✓ 快速原型的某个部分可以用到最终的软件产品中(复用)。

软件工程概述 31

3. 增量模型

- **增量模型**是迭代和演进的过程。
- 增量模型把软件产品分解成一系列的增量构件，在增量开发迭代中逐步加入。分解时必须遵守约束条件：**当把新构件集成到现有软件中时，所形成的产品必须是可测试的。**
- 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。
- 第一个构件实现软件的基本需求，提供**核心功能**。原先完成的增量可以为后期的增量提供服务。

软件工程概述 32



分批地逐步向用户提交产品，整个软件产品被分解成许多个增量构件，开发人员一个构件接一个构件地向用户提交产品。

- **优点**
 - ✓ 能在**较短时间**内向用户提交可完成部分工作的产品。
 - ✓ 逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品。
- **缺点**
 - ✓ 对体系结构的设计要求高(必须开放的、具有可扩充性)，要求更精心地设计。

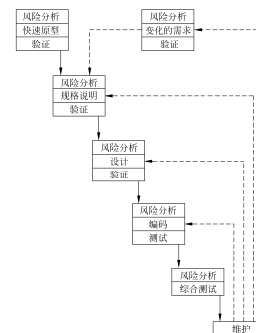
软件工程概述 35

4. 螺旋模型

- 软件风险是任何软件开发项目中都普遍存在的实际问题。
- 螺旋模型的基本思想是：将瀑布模型与迭代过程结合起来，并且加入风险分析；或者说，在每个阶段之前都增加了风险分析过程的快速原型模型。使用原型及其他方法来尽量降低风险。
- 螺旋模型沿着螺旋线旋转，自内向外每旋转一圈便开发出更完善的一个新版本。
制定计划； 风险分析； 实施工程； 客户评估

软件工程概述

36



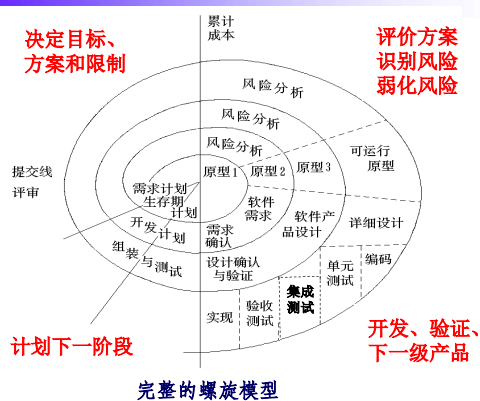
简化的螺旋模型

软件工程概述

37

决定目标、
方案和限制

评价方案
识别风险
弱化风险



完整的螺旋模型

软件工程概述

38

螺旋模型是一种风险驱动的过程模型。主要适用于内部开发的大规模软件项目（风险评估的代价与可能的中止）。

■ 主要的优点在于：

- ✓ 减少了过多测试或测试不足所带来的风险；
- ✓ 维护只是模型的另一个周期，在维护和开发之间并没有本质区别；
- ✓ 对可选方案和约束条件的强调有利于软件的重用。

■ 缺点：

- ✓ 要求软件开发人员具有丰富的风险评估经验和这方面的专门知识。

软件工程概述

39

5. 敏捷过程

为了使软件开发团对具有高效工作和快速响应变化的能力，17位著名的软件专家于2001年联合起草了敏捷软件开发宣言。

敏捷软件开发由4个观点：

- ✓ 个体和交互胜过过程和工具
- ✓ 可以工作的软件胜过面面俱到的文档
- ✓ 客户合作胜过合同谈判
- ✓ 响应变化胜过遵循计划

软件工程概述

40

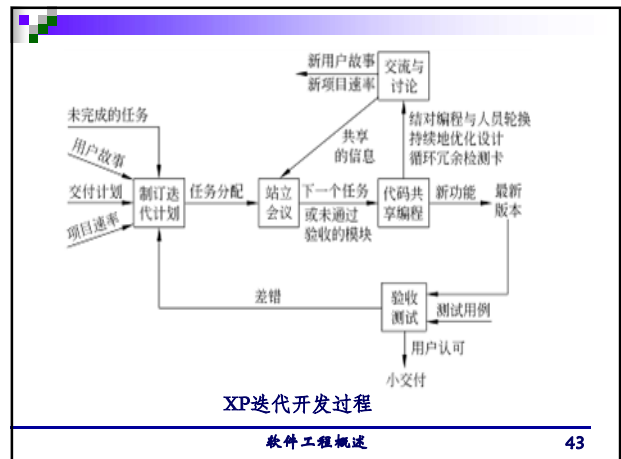
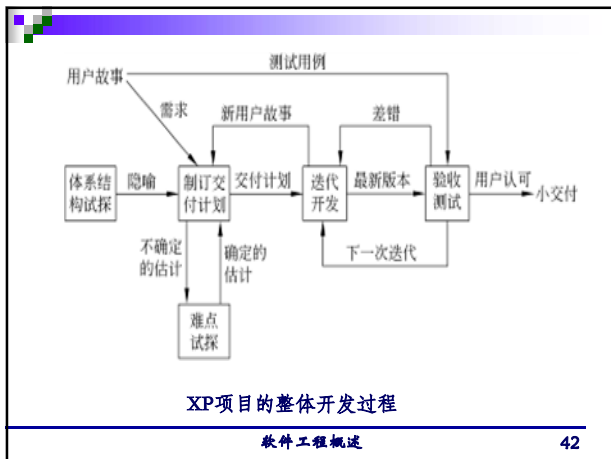
极限编程

极限编程（XP）是敏捷过程一个典型的开发方法。适用于需求模糊且经常变化的场合。有效实践包括：

- ✓ 客户作为开发的团队
- ✓ 使用用户素材
- ✓ 短交付周期
- ✓ 验收测试
- ✓ 结对编程
- ✓ 测试驱动开发
- ✓ 集体所有
- ✓ 持续集成
- ✓ 可持续发展
- ✓ 开放的工作空间
- ✓ 及时调整计划
- ✓ 简单的设计
- ✓ 重构
- ✓ 使用隐喻

软件工程概述

41

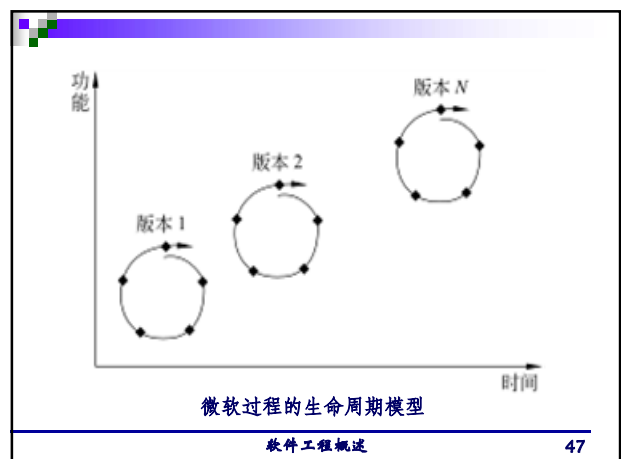
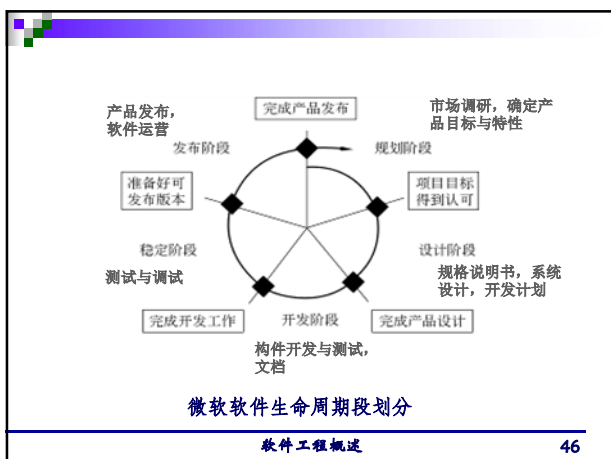


敏捷过程具有**对变化和不确定性的更快速、更敏捷**的反映特性，而且能够保持**可持续的开发速度**。

敏捷过程能够较好地适应商业竞争环境下对小型项目提出的**有限资源**和**有限开发时间**的约束。

软件工程概述 44

- ### 6. 微软过程
- 微软公司采用的软件开发过程，已经被实践证明是非常成功和行之有效的。包括如下准则：
- ✓ 项目计划兼顾不确定因素
 - ✓ 有效的风险管理
 - ✓ 经常生成并快速测试软件过渡版本
 - ✓ 采用快速循环、递进的开发过程
 - ✓ 平衡产品特性和产品成本
 - ✓ 项目进度表应具有稳定性和权威性
 - ✓ 使用小型项目组
 - ✓ 软件配置项基线化
 - ✓ 使用原型验证概念
 - ✓ 追求零缺陷
 - ✓ 里程碑评审会
- 软件工程概述 45



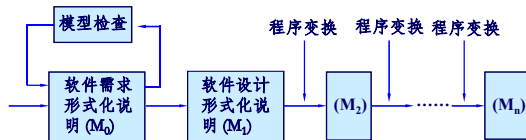
微软过程综合了RUP和敏捷过程的许多优点，是对很多成功项目的开发经验总结。

适用于商业环境下具有**有限资源**和**有限开发时间**约束的项目。

7. 变换模型

- 变换模型是一种**基于形式化规格说明语言及程序变换**的软件开发模型。
- 它采用形式化的软件开发方法，对形式化的软件规格说明进行一系列自动的或半自动的程序变换，最终映射成为计算机系统能够接受的程序系统。
- 多步程序变换过程的重要性质是：每一步程序变换的正确性仅与该步变换所依据的规范 M_i 以及对变换后的假设 M_{i+1} 有关。

- 在此意义上，变换步骤独立于其他变换步骤。这称为变换的独立性。
- 该模型**只适合于软件的形式化开发方法**；需要严格的数学理论和形式化技术支持；需要一整套开发环境（如程序变换工具、定理证明工具等）的支持。



软件过程模型比较

- 针对不同的开发环境、组织环境、项目特点，选择合适的软件过程模型。
- ✓ **瀑布模型** 历史悠久、规范的、文档驱动的方法；最终开发出的软件产品可能并不是用户真正需要的。
- ✓ **快速原型模型** 改进瀑布模型的缺点，让用户试用原型并收集用户反馈意见的办法，获取用户的真实需求。

- ✓ **增量模型** 早期阶段使投资获得明显回报和较易维护；要求软件具有开放的结构。
- ✓ **螺旋模型** 风险驱动的过程，适用于内部开发的大型软件项目，但要求开发人员具有风险分析和排除风险的经验及专门知识。
- ✓ **敏捷过程** 一种有效应对需求不清和快速变化的小型项目。开发过程不是很规范。
- ✓ **微软过程** 一种适用于商业竞争环境下软件项目开发。

- ✓ **喷泉模型*** 面向对象方法学常用的过程模型，强调不同阶段的迭代和无缝的特性。
- ✓ **RUP*** 一个非常规范、全面的过程模型，并在实践中广泛应用。