


Shanghai Jiao Tong University

上海交通大学


软件工程

Module 2: 软件过程

上海交通大学计算机系



软件过程



- ◆ 软件开发方法
- ◆ 软件过程概述
- ◆ 软件生命周期模型
- ◆ 推荐的软件过程
 - 统一软件过程 RUP
 - 敏捷过程
- ◆ 选择和实施软件过程

Software Engineering

2

沈备军

软件开发方法

- ◆ 面向对象方法 (Object Oriented Method)
 - 统一建模语言(UML)
- ◆ 结构化方法 (Structured Method)
- ◆ 形式化方法 (Formal Method)

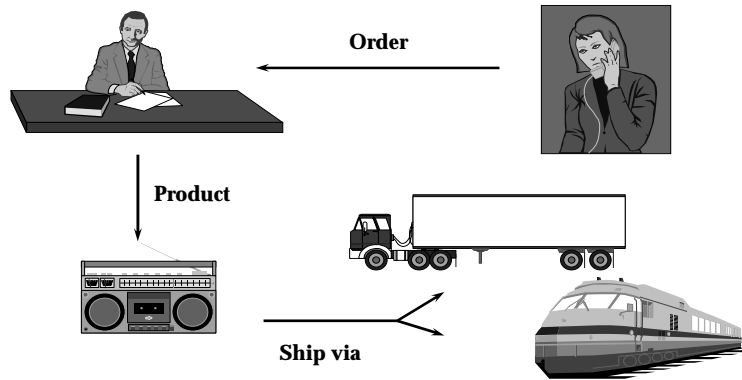
面向对象方法

- ◆ 九十年代以来的主流开发方法
 - 符合人们对客观世界的认识规律
 - 开发的系统结构易于理解、易于维护
 - 继承机制有力支持软件复用
- ◆ 常见的面向对象方法
 - Booch method 1994
 - Coad and Yourdon method 1991
 - Rumbaugh method -- OMT 1991
 - Jacobson method – OOSE 1992
 - Wirfs-Brock method 1990
 -

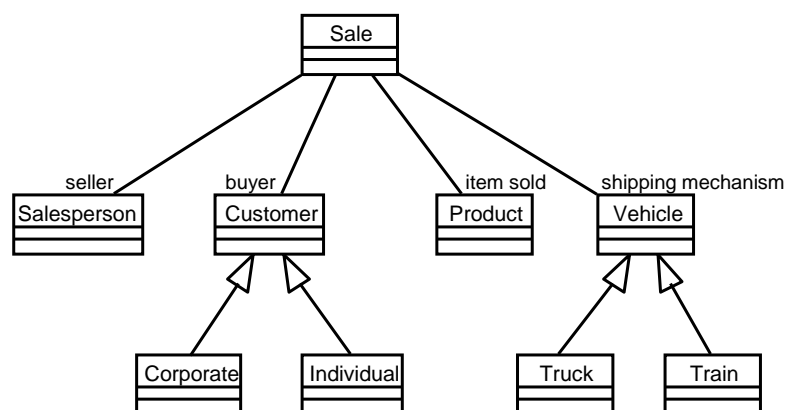


国际标准统一建模语言 UML 1997

举例：销售订单

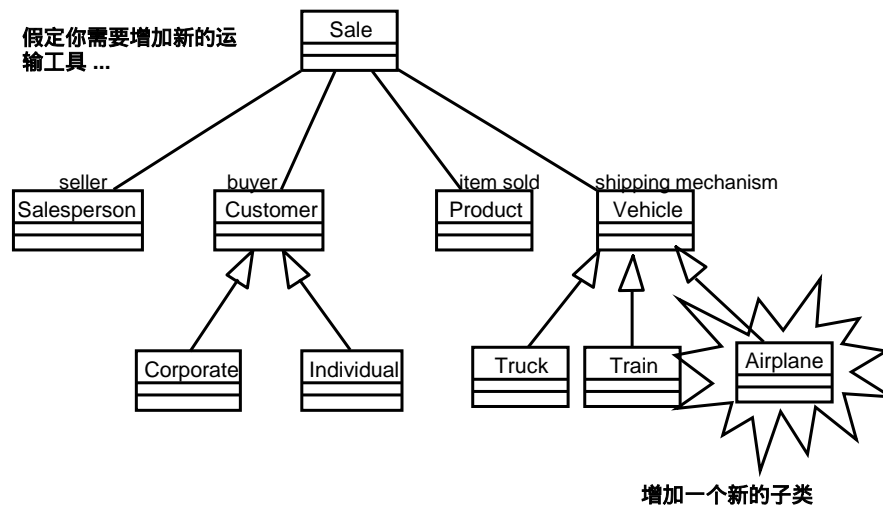


销售订单的类图



需求变更的影响

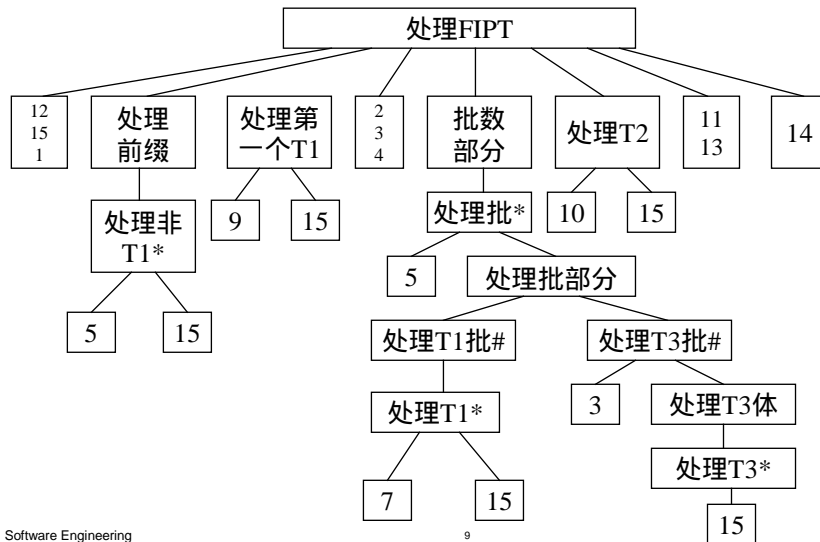
假定你需要增加新的运输工具 ...



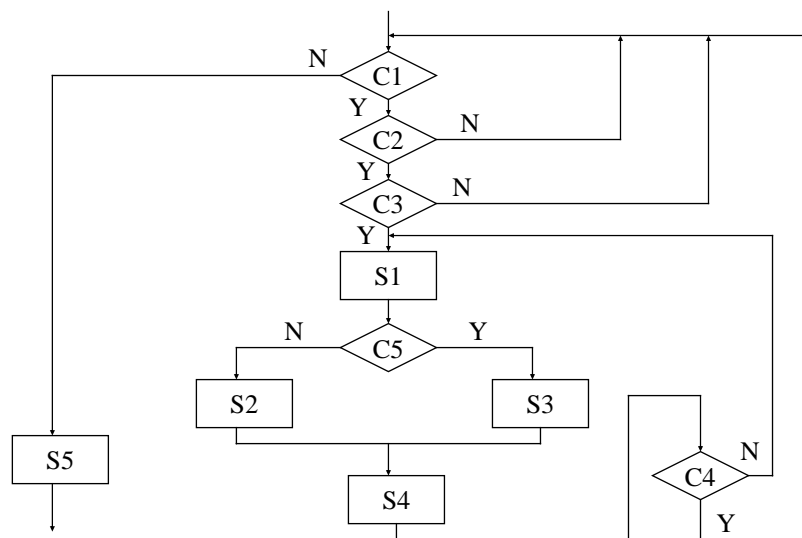
结构化方法

- ◆ 核心: 自顶向下, 逐步求精
- ◆ 手段: 分解 (模块化)、抽象
- ◆ 常用建模工具:
 - 需求建模:
 - DFD(数据流图)、DD(数据字典)、ERD(实体关系图)、STD(状态图)
 - 设计建模:
 - 结构图 (SC)
 - 流程图、N-S图、PAD图、伪代码

结构图举例



流程图举例



形式化方法

- ◆ 形式化方法是基于数学的技术开发软件，如集合论、模糊逻辑、函数。
- ◆ 形式化方法的好处：
 - 无二义性
 - 一致性
 - 正确性
 - 完整性

举例

```
-----AddBlock-----  
  BlockHandler  
Ablocks? : P BLOCKS  
-----  
Ablocks?  $\subseteq$  used  
BlockQueue'=BlockQueue   Ablocks?  
used'=used  
free'=free
```


加一个块集合到队列的尾部, 采用Z语言

形式化方法的不足

- ◆ 形式化规约主要关注于功能和数据，而问题的时序、控制和行为等方面却更难于表示。此外，有些问题元素(如，人机界面)最好用图形技术来刻画。
- ◆ 使用形式化方法来建立规约比其他方法更难于学习，并且对某些软件实践者来说它代表了一种重要的“文化冲击”。
- ◆ 难以支持大的复杂系统。

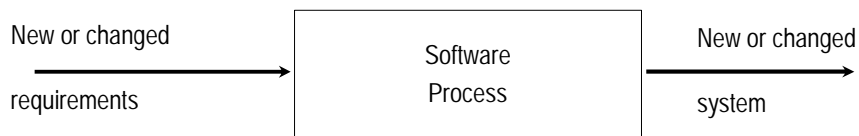
尚未成为主流的开发方法，实践和应用较少

软件过程

- ◆ 软件开发方法
-  ◆ 软件过程概述
- ◆ 软件生命周期模型
- ◆ 推荐的软件过程
 - 统一软件过程 RUP
 - 敏捷过程
- ◆ 选择和实施软件过程

什么是软件过程

- ◆ Defines Who is doing What, When to do it, and How to reach a certain goal.

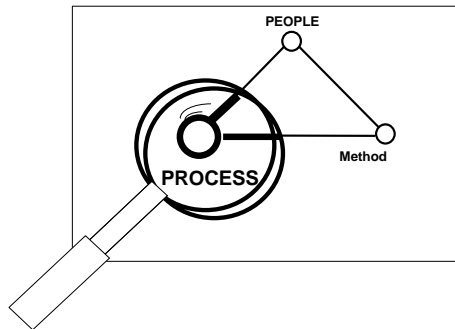


软件过程的组成

- ◆ 软件过程，也称为软件生存周期过程，是指软件生存周期中的一系列相关过程，其中过程就是活动的集合，活动是任务的集合，任务要起到把输入加工成输出的作用。
- ◆ 活动的执行可以是顺序的、迭代的（重复的）、并行的、嵌套的，或者是有条件地引发的。

软件过程的杠杆作用点

- ◆ 每个人都体会到主动积极的优质劳动力的重要性，但是...



...如果不理解过程，或者过程不是在“最佳实践”下运行，即使我们的技术精英也无法使工作达到最佳的状态。

过程是产品成本、进度和质量的主要决定因素

软件过程和软件方法是相对独立的

软件过程

- ◆ 软件开发方法
- ◆ 软件过程概述
- ◆ 软件生命周期模型
- ◆ 推荐的软件过程
 - 统一软件过程 RUP
 - 敏捷过程
- ◆ 选择和实施软件过程

什么是软件生存周期模型

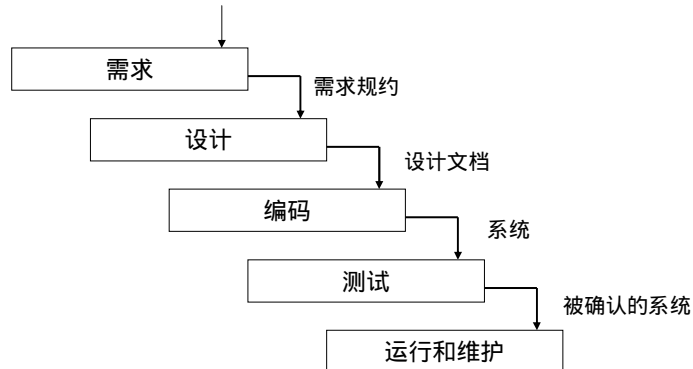
- ◆ 是软件生命周期的一个框架，规定了软件开发、运作和维护等所需的过程、活动和任务。
- ◆ 又称软件开发模型

软件生存周期模型

- ◆ 线性顺序模型 Waterfall Model
- ◆ 增量式模型 Incremental Model
- ◆ 演化模型 Evolutionary Model
 - 原型 Prototyping
 - 螺旋模型 Spiral Model
 - 并发开发模型 Concurrent Development Model
- ◆ 特殊过程模型
 - 基于构件的开发 Component-Based Development
 - 形式化方法模型 Formal Methods Model
 - 面向侧面的软件开发 Aspect-Oriented Software Development

瀑布型(Waterfall)

- ◆ 最早的软件开发模型
- ◆ 1970年W. Royce提出
- ◆ 又称为线性顺序模型



瀑布型特点

- ◆ 特点
 - 强调阶段的划分及其顺序性
 - 强调各阶段工作及其文档的完备性
 - 每个阶段结束之前，都从技术和管理两个角度进行严格的审查
 - 是一种严格线性的、按阶段顺序的、逐步细化的开发模式
- ◆ 适用时机
 - 所有功能、性能等要求能一次理解和描述时
 - 所有的系统功能一次交付时
 - 必须同时淘汰全部老系统时

瀑布型的价值

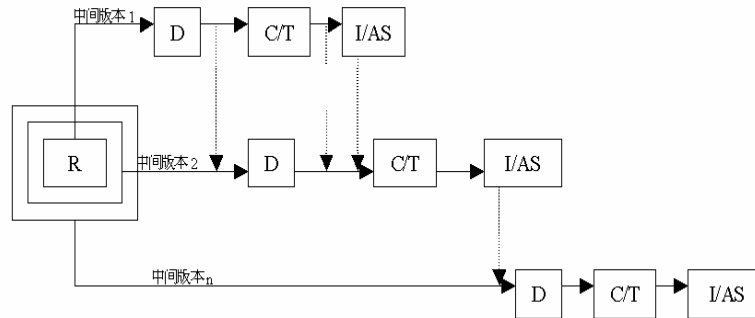
- ◆ 结构简单明了；历史较长、应用面广泛、为广大软件工作者所熟悉；已有与之配套的一组十分成熟的开发方法和丰富的支撑工具。
- ◆ 一种较为有效的管理模式：订计划、成本预算、组织开发人员,阶段评审,文档管理,从而对软件质量有一定的保证。

瀑布型的风险和缺点

- ◆ 获得完善的需求规约是非常困难的；
- ◆ 难以适应快速变化需求；
- ◆ 系统太大时,难以一次做完；
- ◆ 反馈信息慢；
- ◆ 极可能引起开发后期的大量返工，如返工到需求、设计等早期活动；
- ◆ ...

增量型 (Incremental)

- ◆ 构造一系列可执行的中间版本 (Version by Version)



可能的信息流

R: 需求 C/T: 编码和测试
D: 设计 I/AS: 安装和验收支持

增量型需考虑的风险

- ◆ 需求未被很好地理解
- ◆ 一次要求所有功能
- ◆ 需求迅速发生变化
- ◆ 事先打算采用的技术迅速发生变化
- ◆ 长时期内仅有有限的资源 (人员/资金)

增量型适用时机

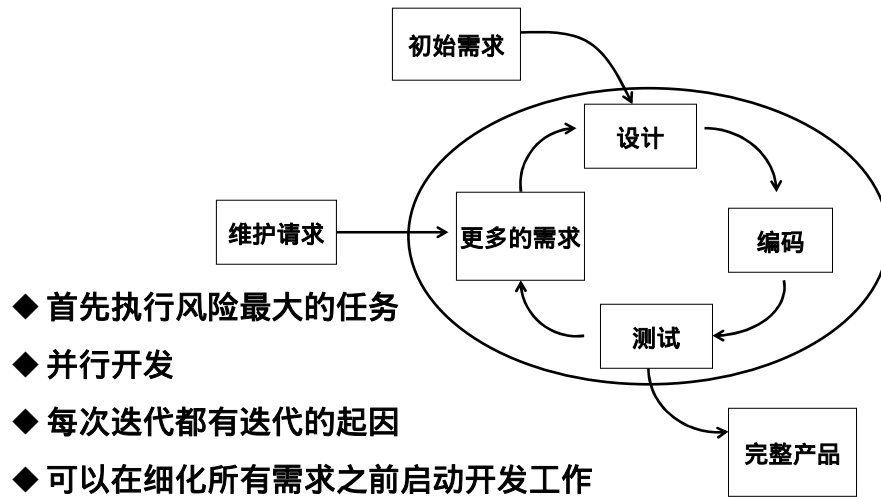
- ◆ 需要早期获得功能；
- ◆ 中间产品可以提供使用；
- ◆ 系统被自然地分割成增量；
- ◆ 工作人员/资金可以逐步增加。

演化型(Evolutionary)

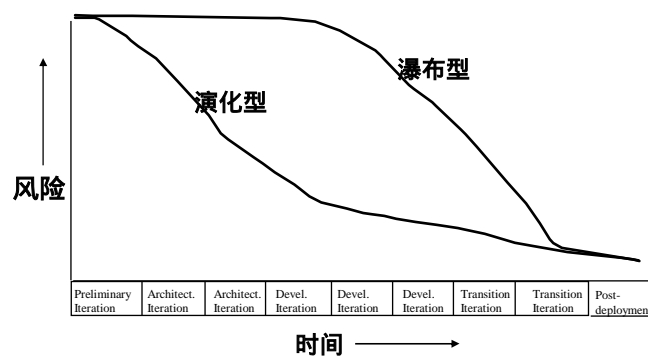
- ◆ 现状：
 - 软件需求在软件开发过程中常常发生改变，想要一次迭代就开发出最终产品是不可能的
 - 紧迫的市场期限使得难以一下子完成一个完善的软件产品
- ◆ 解决方案：演化模型
 - 只要核心需求能够被很好地理解，就可以进行渐进式开发，其余需求可以在后续的迭代中进一步定义和实现。这种过程模型称为演化模型，它能很好地适应随时间演化的产品的开发。
- ◆ 特点：
 - 迭代的开发方法，渐进地开发各个可执行版本，逐步完善软件产品。每个版本在开发时，开发过程中的活动和任务顺序地或部分重叠平行地被采用。
 - 与增量模型的区别是：需求在开发早期不能被完全了解和确定，在一部分被定义后开发就开始了，然后在每个相继的版本中逐步完善。

演化模型举例

基于风险的、顺序执行的演化模型



演化型价值：降低风险



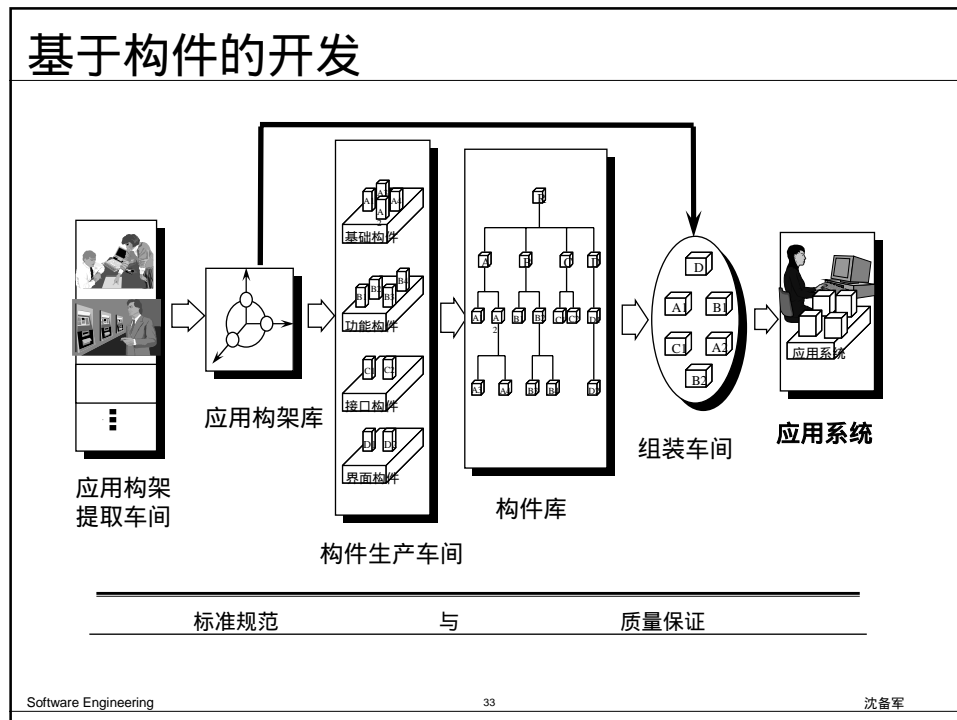
演化模型是目前采用最广泛的模型

演化模型的类型

- ◆ 常见类型
 - 原型 Prototyping
 - 螺旋模型 Spiral Model
 - 并发开发模型 Concurrent Development Model

特殊过程模型

- ◆ 基于构件的开发 Component-Based Development
- ◆ 形式化方法模型 Formal Methods Model
- ◆ 面向侧面的软件开发 Aspect-Oriented Software Development



构件和可复用构件

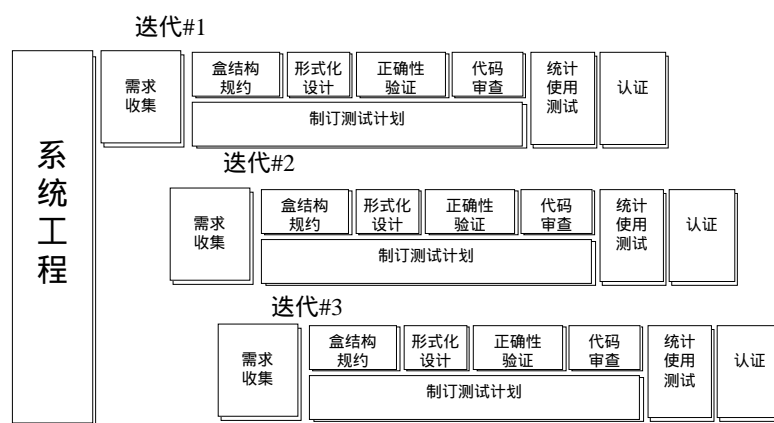
- ◆ 构件 (Component) :可以被明确标识的软件制品。
- ◆ 可复用构件：可被其它系统的开发者复用的构件。
- ◆ 可复用构件概念的外延化
 - 随着对软件复用理解的深入，构件的概念已延伸到需求、构架、设计、设计模式、测试计划、测试案例和数据以及其他对开发活动有用的信息，统称为可复用软件构件。

Software Engineering 34 沈备军

形式化方法模型

- ◆ 净室软件工程 Clean Room Software Engineering
 - 通过在第一次正确地书写代码增量并在测试前验证它们的正确性来避免对成本很高的错误消除过程的依赖；在代码增量积聚到系统的过程的同时进行代码增量的统计性测试。
- ◆ 特点：可生成高质量的软件

净室软件过程图示

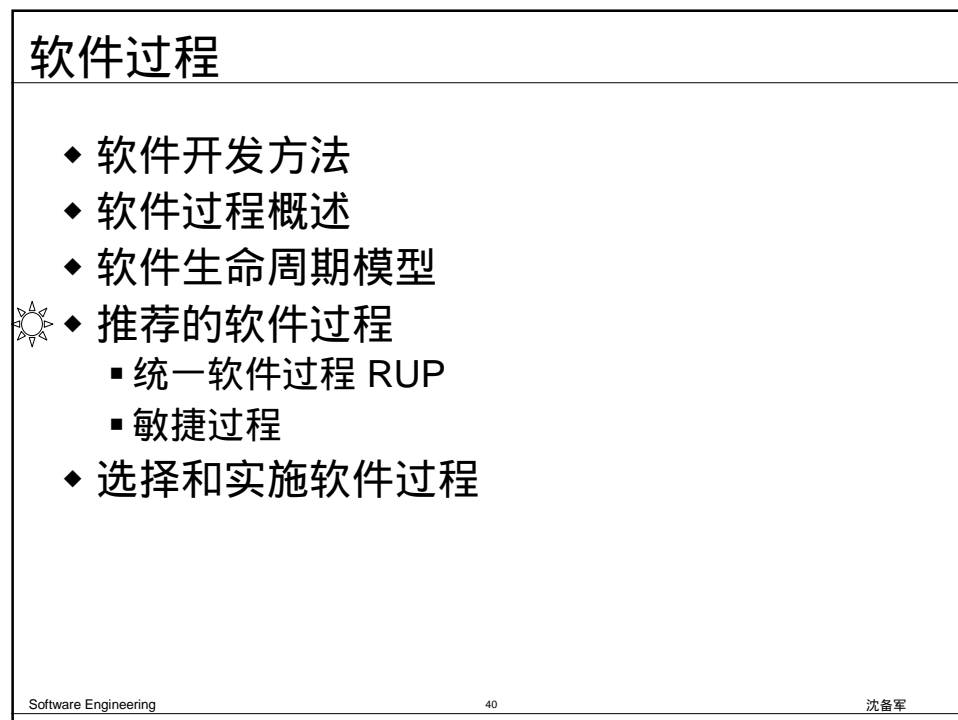
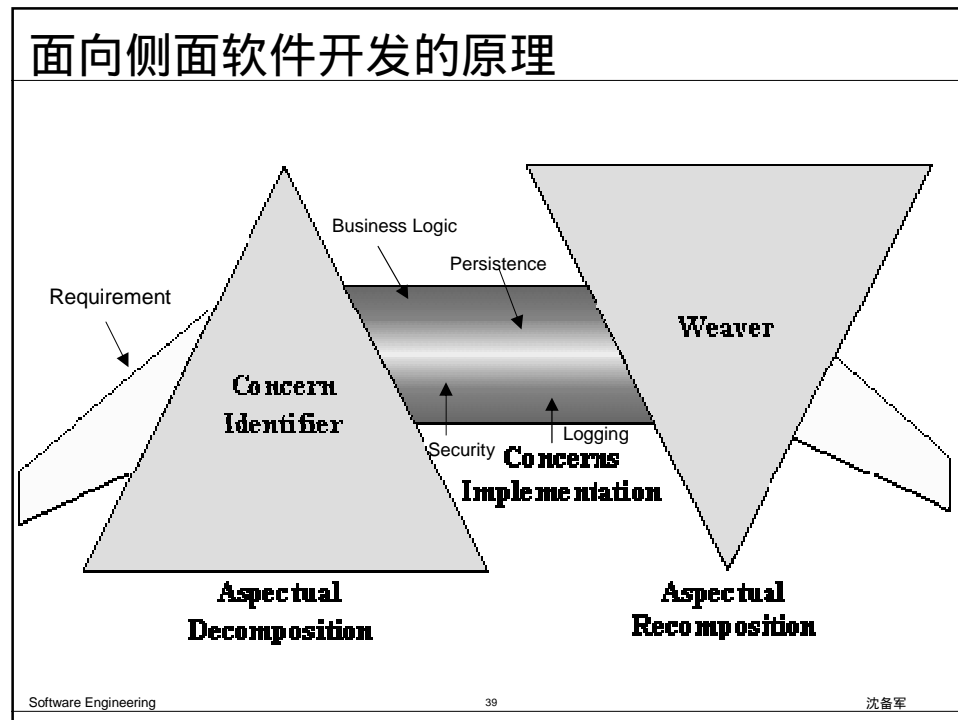


净室软件工程技术

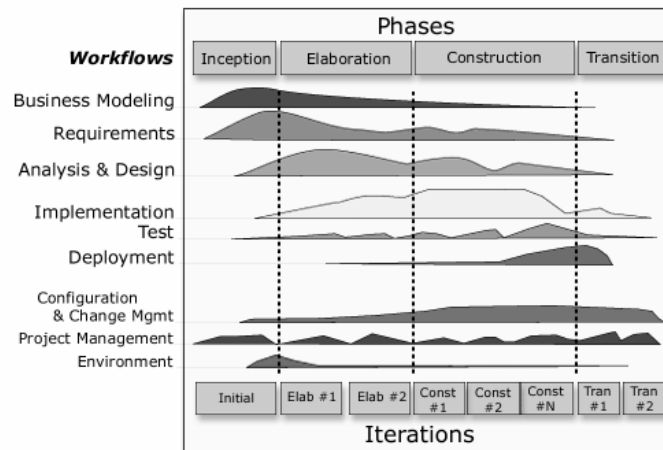
- ◆ 是一种增量软件过程模型，以严格的工程过程预防缺陷,而不是事后排除
- ◆ 基于函数的规范和设计
 - 盒结构:黑盒、状态盒、明盒
- ◆ 基于函数理论的正确性验证
- ◆ 统计测试
 - 抽样,测试用例--随机样本,概率分布

面向侧面的软件开发

- ◆ 现实问题：
 - 对多数复杂的系统，在多种需求关注点和实现的模块之间的对应中，往往出现交叉分割（横切，crosscut）的现象。
- ◆ 解决方案：面向侧面的软件开发
aspect-oriented software development
- ◆ 举例
 - 对于一个信用卡应用程序来说，存款、取款、帐单管理是它的主关注点，日志和持久化将成为横切整个对象结构的横切关注点。

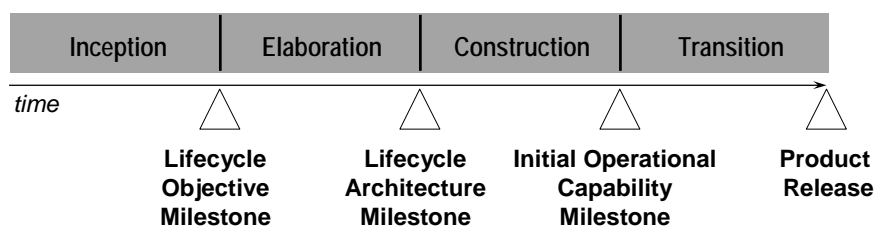


统一软件过程 RUP



RUP是一个风险驱动的、基于UML和构件式架构的迭代、递增型开发过程。

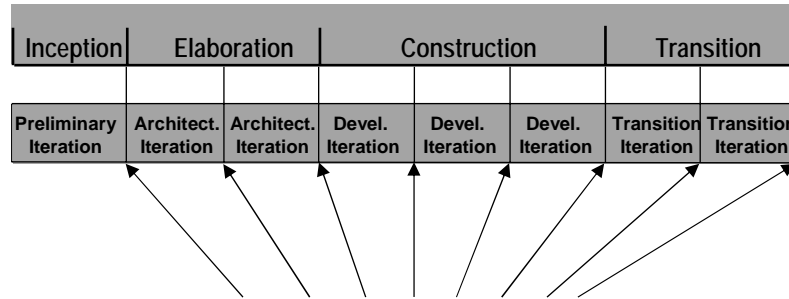
RUP的四个阶段



- Inception - Define the scope of project
- Elaboration - Plan project, specify features, baseline architecture
- Construction - Build the product
- Transition - Transition the product into end user community

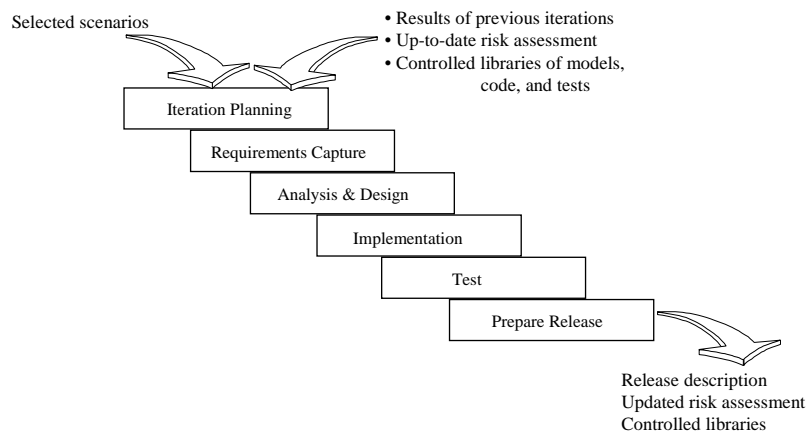
每个阶段结束是一个大的里程碑

阶段和迭代



An iteration is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release (internal or external)

一个迭代周期：一个小的瀑布模型



敏捷过程

- ◆ 敏捷过程很容易适应变化并迅速做出自我调整，在保证质量的前提下，实现企业效益的最大化。
- ◆ 敏捷过程在保证软件开发有成功产出的前提下，尽量减少开发过程中的活动和制品，Just enough
- ◆ 2001年2月，新方法的一些创始人在美国犹他州成立Agile 联盟 (www.agilealliance.org)

XP SCRUM Crystal ASD dx
FDD DSDM Lean Development

敏捷过程的核心理念

- ◆ 基于适应而非预测
 - Agile方法通过快速、短迭代式的开发，不断产出和演化可运行软件，根据用户的反馈信息作适应性调整，然后进入下一轮快速短迭代式开发
- ◆ 以人为导向而非过程导向
 - 努力营造诚信、开放的组织氛围，根据项目中信息流通的具体情况，按高内聚、松耦合的原则，将项目组划分为若干小组（每个小组以不超过10人为宜，组员均在一个工作间内工作），通过小组内各种渠道的沟通，来减少中间制品的工作负担，提高应变能力

--Martin Fowler "New Methodology"

敏捷过程的适用范围

Martin Fowler认为：新方法不是到处可适用的

适合采用敏捷过程的情况：

- 需求不确定、易挥发
- 有责任感和积极向上的开发人员
- 用户容易沟通并能参与
- 小于10个人的项目团队

极限编程（XP）

- ◆ 由Kent Beck、Ward Cunningham、Ron Jeffries等人提出反响最大、最为完善的敏捷过程方法。
- ◆ 价值观：
 沟通、反馈、简化、勇气
- ◆ 特点：
 测试成为开发的核心；
 纪律性与灵活性巧妙结合。

www.extremeprogramming.org

XP关键做法

- ◆ 现场客户 (On-site Customer)
- ◆ 计划博弈 (Planning Game)
- ◆ 系统隐喻 (System Metaphor)
- ◆ 简化设计 (Simple Design)
- ◆ 集体拥有代码 (Collective Code Ownership)
- ◆ 结对编程 (Pair Programming)
- ◆ 测试驱动 (Test-driven)
- ◆ 小型发布 (Small Releases)
- ◆ 重构 (Refactoring)
- ◆ 持续集成 (Continuous integration)
- ◆ 每周40小时工作制 (40-hour Weeks)
- ◆ 代码规范 (Coding Standards)


RUP与XP的共性

- ◆ 基础都是面向对象方法 (取代传统的结构化方法)
- ◆ 都重视代码、文档的最小化和设计的简化
- ◆ 采用动态适应变化的演进式迭代周期 (取代传统的瀑布型生命周期)
- ◆ 需求和测试驱动
- ◆ 鼓励用户积极参与

RUP与XP的区别

- ◆ XP以代码为中心，编码和设计活动融为一体，弱化了架构的概念。
- ◆ RUP过程通常以架构为中心，细化阶段的主要目的就是构造出一个可运行的架构原型，作为将来添加需求功能的稳固基础。
- ◆ XP不包含业务建模、部署、过程管理等概念。
- ◆ RUP适合各种规模的项目，XP只适用于小团队。

软件过程

- ◆ 软件开发方法
- ◆ 软件过程概述
- ◆ 软件生命周期模型
- ◆ 推荐的软件过程
 - 统一软件过程 RUP
 - 敏捷过程
-  ◆ 选择和实施软件过程

最佳过程？

“One Size does not Fit All”

不同的项目需要不同过程

- ◆ 具体环境：项目、产品（军用、民用等）、资源、团队、文化、地域（集中、分布）.....
- ◆ 层次：组织过程、项目过程、团队过程、个人过程
- ◆ 业务目标
- ◆ 开发类型：新产品、重用、COTS、维护、服务、产品线.....

过程选择原则

- ◆ 越大的团体需要越大的过程。
- ◆ 越关键的系统（未发现的缺陷将产生更严重的灾害）在构建的正确性方面需要越多的透明度（更大的密度）。
- ◆ 在过程大小或密度上相对小的增加会引起项目成本相对大的增长。
- ◆ 最有效的沟通（交换意见）方式是面对面的互动，例如在白板前的讨论。

——Alistair Cockburn, Selecting a Project's Methodology