软件生命模型

软件工程过程

基本目标

- 合理的进度
- 有限的经费
- 一定的质量

主要活动

- 软件规格说明：规定软件的功能及其使用限制
- 软件开发：产生满足规格说明的软件
- 软件确认：通过有效性验证以保证软件能够满足客户的要求
- 软件演进：为了满足客户的要求变更，软件必须在使用过程中不断改进

软件生命周期

软件过程模型分类

- 工作流模型：强调活动的组织控制策略
- 数据流模型：强调工件的变换关系
- 角色/动作模型：强调角色的划分、角色的协作关系

软件生命周期的概念

指软件从考虑其概念开始，到该软件不再使用位置的整个时期

基本步骤

- 制定计划
    - 确定总目标
    - 给出功能、性能、可靠性以及接口等方面的要求
    - 可行性研究
    - 估计可利用资源、成本、效益、开发进度
    - 指定实施计划，连通可行性研究报告一并提交管理部门审查
- 需求分析
    - 分析提出的要求并进行详细定义
    - 编写需求规格说明书以及初步的系统用户手册
    - 提交管理机构审查
- 软件设计
    - 概要设计：把需求转换成体系结构，体系结构中的模块和某些需求对应
    - 详细设计：对每个模块进行具体描述
    - 编写设计说明书
- 程序编码：程序应是结构良好、清晰易读，且与设计相一致
- 测试

    - 单元测试
    - 组装测试
    - 确认测试：按规定的各项需求，注意进行有效性测试
- 运行维护：修改错误，变更功能或增强功能

传统软件生命周期模型

瀑布模型

基本工程活动

- 系统需求
- 软件需求
- 分析
- 程序设计
- 编码
- 测试
- 运行

特征

- 本活动的工作对象来自上一项活动的输出，一般为代表本阶段结束的里程碑式文档
- 根据本阶段的活动规程执行相应的任务
- 产生本阶段活动相关产出，即软件工件，作为下一活动的输入
- 对本阶段进行评审

优点

- 降低软件开发的复杂程度，提高软件开发的透明程度，便于将软件工程过程和软件管理过程有机融合，从而提高软件开发的可管理程度
- 推迟软件实现，强调在软件实现前必须进行分析和设计工作，可以有效避免代码频繁修改、代码结构不清晰、代码质量低下等问题
- 以项目评审和文档控制为手段有效的地对整个开发过程进行指导，保证阶段之间正确的连接，能够及时发现并纠正开发过程中存在的缺陷

缺点

- 模型缺乏灵活性，特别是无法解决软件需求不明确或不准确的问题
- 模型的风险控制能力较弱。
- 规定的文档过多，极大增大系统的工作量；管理人员只以文档来评估项目完成情况，往往导致错误的结论

演化模型

提倡两次开发：

- 第一次是试验开发，得到试验性的原型产品，其目标只是在于探索可行性，弄清软件需求；
- 第二次在此基础上获得较为满意的软件产品。

特点

- 优点
  - 明确用户需求、提高系统质量、降低开发风险
- 缺点
  - 难于管理
  - 抛弃了瀑布模型使用文档控制的优点
  - 可能导致最终软件的系统结构较差

使用范围

- 需求不清楚
- 小型或中型项目
- 开发周期短

增量模型

首先针对核心需求确定系统体系结构，然后按优先级逐步实现增量需求。增量开发可以依据需求是否明确，采用瀑布模型或演化模型

优点

- 客户在第一次增量后就可以使用到系统核心功能，增加客户信心
- 失败风险较低，即使某个增量开发失败，核心功能依旧可以提供客户使用
- 最高优先级的功能得到多次测试，保障了系统重要功能的可靠性
- 增量都是在同一个体系结构下集成的，提高了系统的稳定性和可维护性

缺点

- 增量粒度难以选择
- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

* 增量粒度难以选择

- 确定所有基本业务比较困难

喷泉模型

认为开发过程的各个过程时相互重叠和多次反复的，各个开发过程没有特定的次序要求，完全可以并行，可以在某个开发阶段随时补充其他开发阶段遗漏的需求

保持开发流程的无间歇性

优点

- 提高开发效率
- 缩短开发周期

缺点

- 难于管理

螺旋模型

将过程用螺旋线表示，每个回路表示软件工程的一个阶段，每个回路被分在四个象限

四个象限

- 制定计划
- 风险分析
- 实时工程
- 客户评价

使用范围

- 大型项目
- 开发周期长
- 风险高

V 模型和 W 模型

- V 模型细化了测试的级别，分别和开发阶段对应
    - 单元测试
    - 集成测试：检查单元之间的接口，同时检查单元完成的功能是否和接口一致
    - 系统测试：检查系统功能是否达到系统要求，同时检查系统和外围系统之间的接口是否存在缺陷
    - 验收测试：检查系统是否满足客户的需求
- W 模型：为开发阶段增加了确认和验收的活动，强调测试的对象不仅仅是程序
- 没有改变开发流程的串行关系，只适合需求比较稳定的项目

构建组装模型

将系统模块化，构建组装模型的开发过程就是构建组装的过程，维护过程就是构建升级、替换和扩充的过程

优点

- 充分利用软件复用，提高开发效率
- 允许多个项目同时开发，降低了费用，提高可维护性

缺点

- 缺乏通用的构建组装结构标准，风险较大

- 构建可重用性和系统高效性之间不易协调
- 过分依赖构建，构建质量影响软件总体质量

快速应用开发模型（RAD）

强调极短的开发周期，使用构建组装方法进行快速开发

缺点

- 并非所有应用都适合 RAD
- 开发人员和客户完成需求分析的时间较短，沟通不当可能导致失败
- 适合管理信息系统的开发，不适用如技术风险较高、与外围系统的互操作性较高的系统

原型方法

原因

- 开发早期用户很难准确表达全面需求
- 随着工作的进行，用户可能产生新需求
- 开发者可能遇到没有预料到的实际困难，需要改变需求来解脱困境

分类

- 废弃策略：废弃原来的原型系统
    - 探索性：弄清用户对于系统的需求。针对需求模糊、用户和开发者都缺乏经验的情况
    - 实验型：用于开发前，考核技术实现方案是否合适，分析和设计是否可靠
- 追加策略：将新的需求添加到原型系统中
    - 进化型：通过不断改进原型适应需求变化，将原型方法扩散到开发全过程

优点

- 有助于理解用户需求的真实想法
- 容易确定系统性能，服务的可应用性，设计的可行性
- 原型有的可以原封不动作为最终成品，有的可以稍加修改称为最终系统一部分，有利于构建最终系统

缺点

- 大型系统不经过系统分析而直接使用原型模拟来获得系统的整体划分非常困难
- 对于需要大量运算、逻辑性较强的程序模块，很难构造出模块原型供人评价
- 对于原有应用的业务流程、信息流混乱的情况，原型的构造与使用较困难
- 文档容易被忽略
- 建立原型的工作会被浪费
- 项目难以规划和管理

新型软件生命周期模型