第一章 软件过程

2018年9月4日 8:30

一.软件开发的本质

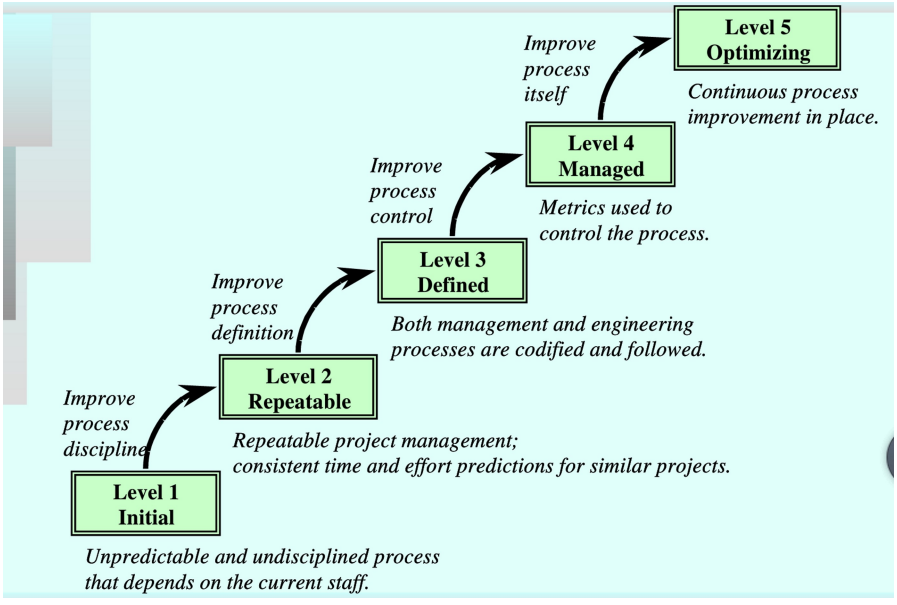
1. 70%的项目是失败的（2005年的数据）：质量差，超期，超预算，需求变更
2. 软件本身是创造性开发行为的产品
3. 软件开发不可变的事实是（无能为力解决，只能承认接受）：
   a. complexity复杂性
   b. conformity一致性：软件与平台，系统一致
   c. Changeability（需求）可变性
   d. invisibility（代码）不可见性
4. 可变的意外困难，不会增加软件开发的复杂性
   a. 社会问题
   b. 提高软件的可适应性=可理解+可维护+可扩展
5. 软件开发的意外事件与三个因素相关
   a. Stakeholder利益相关者:客户，开发者两组，
      利益相关者的定义指对系统产生影响或者被系统所影响的人，是在软件项目中存在利害关系的人
   b. Software process过程：在软件开发和维护过程中定义的活动和组织程序

过程管理标准process

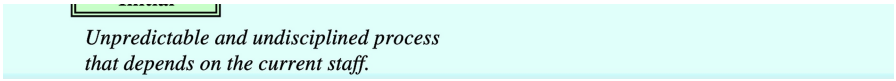
i. 迭代（迭代完发布新版本）增量（一次迭代的功能增加或减少）模型
   - 需要提前做好计划和控制，和预定义的设计框架保持一致
   1) 螺旋模型：包含了RUP，MDA，敏捷开发模型
   2) RUP（retional统一过程）：提供文本，概念，开发思路
   3) MDA（模型驱动的体系结构）
   4) 敏捷开发过程
   5) 面向方面的软件开发

ii. 能力与成熟度模型（Capability Maturity Model ,CMM）用于过程评估和改进的流行方法。
   第一级：走到哪算到哪，没有特定的规则，大部分都在第一级
   第二级：可重复级，靠经验每一次用相同的方法，东软
   第三级：定义级，已经定义好了规则，不到整个软件企业总量的10%
   第四级：管理级，对整个每一步的过程增加了监控，亚马逊
   第五级：优化级，对整个过程或标准优化的策略，基本是军方企业

iii.



iv. ISO 9000质量标准系列 工业生产的通行标准



iv. ISO 9000质量标准系列——工业生产的通行标准
   1) 如果过程是正确的，那么结果一定是正确的，即关注产品质量问题
   2) 强调必须完成什么，没强调必须怎么执行

v. IT基础架构库（ITIL）
   1) 高效的利用people,processes,products,partners
   2) 致力于方案交付和管理的操作方便
   3) 持续的服务改进方案(continuous service improvement programme,CSIP),用来实现解决方案管理的ITIL方法。该方案以实现高水平业务目标的决心为起点，接着检查是否达到里程碑，并通过巩固已达到的改进和持续任务循环而保持发展的势头。

vi. COBIT框架（控制目标信息和相关技术）,了解

产品标准
Product standard

   1) 偏向于产品标准
   2) 侧重组织需要做什么，而非如何去做
   3) 将相关的IT工作组织到4个领域
      a) 规划与组织
      b) 获取与实现
      c) 交付与支持
      d) 监控

c. modeling建模
   i. 需要开发人员在不同阶段进行沟通language和文档化tools
   ii. UML独立于任何软件开发过程，也独立于实现技术，要求：只要面向对象开发方法就行
   缺陷：必须采用面向对象的方法来生产软件
      1) 状态模型state models，类图
         - 描述静态数据结构
      2) 行为模型behavior models，用例图，时序图，活动图
         - 描述对象协作
      3) 状态转换模型state change models，状态图
         - 描述随着时间推移，系统所允许的状态
   iii. 集成方法
      1) 面向信息或面向门户的集成
         a) 面向信息关注信息的实时交换，这是在数据库应用程序接口API层次的集成
         on the level of databases or application programming interfaces (APIs) that externalize information for consumption by other application.
         b) 面向门户是特殊的面向信息的集成，将来自多个软件系统的信息具体化到一个共同的用户界面。需要来自后台系统的具体信息进行人为干预
         in which information is externalized from multiple software systems into a common user interface.
      2) 面向接口的集成，将应用接口（通过接口抽象定义的服务）连接在一起。接口显示了一个应用系统向其他应用系统所提供的有益服务。
      that links application interfaces (that is, services defined through an interface abstraction)
      3) 面向过程的集成，将应用系统连接在一起，方法是在现有应用系统的已有过程集和数据集的顶部定义一个新的过程层。

二.系统计划System planning
- 可以通过各种策略规划、业务建模、业务过程重组、策略调整、信息资源管理或诸如此类的过程决定业务策略
- 以上方法目的是为业务确定长远洞察力，然后优先考虑能够通过使用信息技术而得以解决的业务问题

1. 系统计划的不同方法

a. SWOT，strengths,weakness,opportunities,threats
   - 通过调整组织优势、劣势、机会和威胁的方式来进行IS开发项目的识别、分类、排序和选择。这是一个从确定组织使命开始的、自顶向下的方法。
   - 使命陈述捕获一个组织的独特性质，并详细说明它未来的愿景，良好的使命陈述重点在于客户的需要
   - 对内部的企业优势和劣势的识别是成功业务规划的必要非充分条件，对可利用的外部机会和可避免的外部威胁的认识是决定组织目的和目标的基础
   - 

   i. 实质是看清自己，内因：优势劣势， 外因：机遇挑战
   ii. 得到SWOT矩阵，基于每一个问题细化目标，得到策略

b. VCM，value chain model，价值链模型
   - 服务 是VCM方法的基本活动
   - 通过分析组织中完整的活动链（从原材料到销售及运送给客户的最终产品）来评估竞争优势
   - 目的是理解哪种价值链配置将产生最大竞争优势

   i. 为了创造价值，有哪些活动是必备的
   ii. 主要（基本）活动：对最终产品追加价值 they create or add value to a final product，如教学培养计划
      1) 五个阶段：
         a) 内部物流——接受对产品或服务的投入
         b) 操作——使用投入来创造产品或服务
         c) 外部物流——将产品和服务卖给买家
         d) 销售和市场（∈基本活动）——引导买家购买产品或服务
         e) 服务——维护或提高产品或服务的价值
   iii. 支撑（支持）活动：不对最终产品直接产生价值，至少不直接增加价值，它们仍然基础却不丰富产品,they are essential but they do not enrich the product，如学生食堂
   iv. 基于不同业务场景来区分主要次要活动

c. BPR，business process reengineering ,业务过程重组
   - 当今组织必须彻底改造自己，并丢弃现在使用的功能分解、分层结构和操作原则
   - 主要目的：从根本上重新设计业务过程。必须对业务过程进行识别、流程化和改进。在工作流图中对过程文档化，并经历工作流分析。

   i. 逆向分析，给你源码分析

d. ISA，信息系统体系结构information systems architecture,矩阵分解，分为5行6列,5种人6个阶段
   - 不同于以上方法，ISA是一种自底向上的方法，它为能够适应各种业务策略的IS解决方案提供一种中立的系统结构框架。
   - 五种人：规划者，所有者owner、设计者、建造者、承包者subcontractor
   - 每个参与者所从事的6种不同的描述或体系结构模型。

   i. 规定软件过程中每一个阶段的成果由谁来完成的

2. 所有系统规划的方法有一个共同点——先关注效果再关注效率，即先有效effectiveness，再高效efficiency

三.系统的三级管理systems for three management levels

1. 战略级系统Strategic，由用户中的高层管理员使用，制定系统产品长期计划，如市场预测——知识，对信息的有效使用
    a. 知识处理系统，更强调预测性，从已知得到未知
    - 数据挖掘（datamining）属于知识管理领域，其主要目的：
        - 关联（路径分析）——在数据中发现一个事件导致另一个相关事件发生的模式
        - 分类classification——发现某事实是否落入预定、感兴趣的类别中
        - 聚类clustering——与分类相似，但种类未知，它们由聚类方法发现
    - 用于数据挖掘的数据主要来源于数据仓库，而不是操作型数据库
2. 战术级系统Tactical，做短期目标，进行信息资源分配，如预算分析，工资预测，调度，客服——信息，有用的数据
    a. 分析处理系统，在数据处理基础上进行分析归纳,OLAP即联机分析处理
    - 分析处理系统与数据仓库技术联系在一起。数据仓库（data warehouse）的创建方式通常是在一个或多个事务数据库中提取的数据增量拷贝。数据仓库总是增加新数据，不移除历史数据。
    - 数据仓库的独特特征——对数据的汇总和封装，目的是给源系统提供的数据增加价值
    - 数据集市（data mart）——主要保存被汇总的历史数据
    - Data webhouse也涉及到
3. 操作级系统Operational，由底层操作员，做一些日常工作，如发工资，收银员——数据
    a. 运用工具，事务处理系统，OnLine Transaction Processing ,OLTP即联机事务处理

四.系统开发周期Software development lifecycle——核心内容

1. 建模方法或软件开发方法
    a. 现代软件基本特点是交互，面向对象
        i. 事件驱动，软件执行顺序之前不知道
        ii. 用户控制
        iii. 软件服务响应随机，是不可预测的
    b. 传统软件开发方法是结构化方法，以过程为中心
        - 系统在功能分解活动中被分解为可管理的单元，同时将系统有层次的划分为由数据流连接的业务过程。
        i. 04年前基本上是结构化
        ii. 基本工具
            □ UML不是系统开发的结构化方法的建模技术。
            1) 数据流图DFD（data flow diagrams），用数据的流动描述系统数据模型，过程建模
            2) 实体关系模型ERD（entity relationship diagrams），数据建模
            □ 结构化方法使用DFD作为开发驱动力
        iii. 不适于现代软件开发原因
            1) 结构化方法是顺序，转换的方法，而不是迭代增量的方式，现代软件要求迭代增量
            2) 写好的软件不可变无弹性，即很难对业务功能集进行扩展和沿伸，重用性差
            3) 基本上所有的软件从零开始写，不支持已存在构件的复用
    c. 面向对象方法
        i. 1970s存在，1990s开始流行
        ii. 以数据为中心，在分析阶段不需要定义类的操作，只需要定义类的属性，遵循迭代增量的过程
        iii. 问题：
            1) 理解较困难，结构较复杂，分析阶段在一个相当高的抽象层面进行
            2) 数据库大多数是关系型数据库，怎么把对象存在数据库里
            3) 项目管理更加困难
2. 生命周期——测试贯穿整个周期
    a. 业务（需求）分析——整个生命周期最重要的阶段，没有模块的概念
        i. 主要目的：获取需求
        ii. 两个阶段：

1) 获取和确定业务需求，需要业务分析人员（可以没有软件开发背景，需要与人良好沟通）
2) 将业务需求逻辑转换为系统逻辑，对需求进行详细说明或建模，需要系统分析员

iii. 需求确定determination
1) 服务陈述
a) Business rule,系统的业务规则，如两周发一次工资
b) Computation,系统计算过程，如怎么发这个工资，根据前两周销售业绩发工资
2) 约束陈述
a) 使用约束——使用时给的限定，如选课时每个人只能查看自己的账号信息
b) 开发约束——与系统开发过程所需要软硬件平台的约束

iv. 需求获取specification规格说明
1) 建模方法和工具：
a) case工具
b) 需求文档
2) 最重要的两个规格说明技术
a) 类图（描述系统的对象及对象间的关系，静态数据结构），需求分析阶段得不到完整类图
b) 用例图
3) 在理想状态下，需求分析阶段可以与软硬件平台无关，且非功能性需求在用例图和类图中无法描述

b. 系统设计（架构设计+详细设计）
i. 架构设计Architectural design——根据模块（构件）进行的系统描述
1) 选择一个解决方案的策略，比如用哪个版本的数据库
2) 系统模块的划分
3) 客户机服务器模型经常被扩展为三层体系结构，中间层是逻辑层，实现为动态链接库。
□ 一个良好的体系结构设计会产生可适应（支持）的系统——即可理解、可维护、可升级的系统

ii. 详细设计Detailed design——软件构件内部运行的描述
1) 考虑模块内部的工作过程，如确定需要哪些数据结构（类图描述），算法（伪代码描述）

c. 实现（编码+调试）
▪ 涉及所购买软件的安装和客户定制软件的编码
i. 成果：可运行的程序代码
ii. 客户端的实现和服务器端的实现

d. 集成部署
▪ 软件以较小的模块开发，在为客户用于生产而部署之前，这些构件需要与已经可操作的模块组装且集成在一起。这就是继承和部署阶段
▪ 模块集成会比任何早期生命周期阶段（包括实现阶段）花费更多时间和精力
▪ 面向对象的系统必须为集成和部署而设计
i. 集成（模块集成+应用程序集成）
ii. 部署

e. 运行维护
i. 操作：把系统交给用户进行实际使用
ii. 维护
1) 内务处理（日常维护）——以保持系统的可访问性和可操作性
2) 适应性维护——对系统运行的监控和审核，对系统的功能进行调整，以满足变化的环境
3) 完善性维护，让系统越变越好2.0

3. 跨越生命周期的活动
a. 项目计划project Planing
i. 最终软件产品交付什么，花费，时间，风险，里程碑，资源分配，也包括开发方法、过程、工具、标准和团队组织的选择。

ii. 典型的约束是时间和费用——每个项目都有清晰的期限和紧张的预算

b. 度量评估Metrics——测量开发时间，工作量等

i. 常常在软件质量和复杂性的范围内讨论度量

ii. 用于测量正确性、有效性，完整性，可用性，可维护性，灵活性，可测试性等测量因素

- 度量还可以在生命周期的不同阶段测量开发模型，评估过程的效果并改善工作质量。

c. 测试Testing——跨越整个软件生命周期的一项活动，系统涉及一点变化就需要测试

i. 基于执行的测试

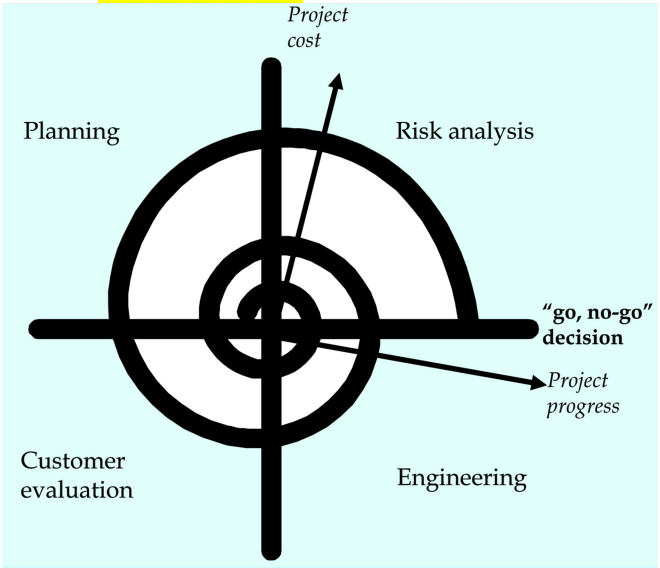
1) 规格说明测试

2) 代码测试

五.开发模型和方法（关于软件的生产方式）

- 现代开发过程是迭代和增量的，一个项目由许多迭代组成，每一次迭代交付一个软件的增量版本。增量改进了系统的功能性、可用性、性能及其他特性，但是并不改变系统的范围

1. 螺旋模型——最本质，风险驱动

a. 

2. IBM Rational 统一过程RUP

a. 软件开发过程平台

b. RUP在二维关系中组织项目

i. 横向维度——项目迭代的连续阶段

1) 初始

2) 细化

3) 构造

4) 转换

ii. 纵向维度——软件开发领域

1) 业务建模

2) 需求

3) 分析和设计

4) 实现

5) 测试

6) 配置及变更管理的部署和支持活动

7) 项目管理和环境

iii. 像螺旋模型一样，RUP强调迭代开发和早期的、持续的风险分析的重要性

iv. RUP具有普遍适用性，但是也使用了IBM retional软件开发工具为团队提供特别指导

3. 模型驱动的开发MDA，基于UML，利用模板

a. 旨在得到与平台无关的模型，包括系统状态和行为的完全规格说明

b. 下一步骤中，MDA提供了工具和技术来创建特定平台的模型

c.

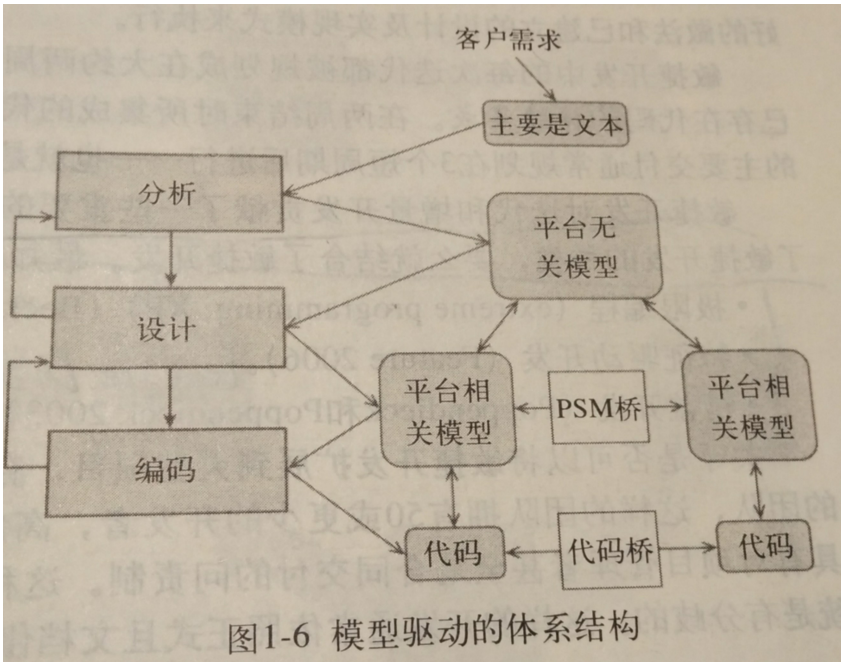


图1-6 模型驱动的体系结构

4. 敏捷开发模型，越快越好，尽快给出成果让用户评价
   a. 编写程序来通过验收测试，这个过程称为测试驱动的开发，并导致策划编程intentional programming——在开始为程序编码前，具体指定验收测试中程序想要获得的能力和机会。
5. 面向切面的开发，适用于功能上有区别但是大部分相同的软件
   a. 面向方面的开发正是应用迭代和增量过程来生产适应性软件的另一种方式。

六.实例学习Case studys