

# 《软件开发过程与项目管理》

## Software Development Process and Project Management

任课教师： 范 国 祥

电 话： 0451-86418876-811(O)

13199561265(Mobile)

邮 箱： fgx@hit.edu.cn

哈工大计算学部

国家示范性软件学院

软件工程教研室

2023. 09

## 基本概念

- **项目(Project)**：为创建某种特定的产品或服务而组织或设计的临时的、一次性的行动，通过执行一组活动，使用受约束的资源(资金、人、原料、能源、空间等)来满足预定义的目标
- **项目管理(Project Management, PM)**：有效的组织与管理各类资源(资金、人、原料、能源、空间等)，以使项目能够在预定的范围、质量、时间和成本等约束条件下顺利交付(deliver)
  - 挑战1：在各类约束条件下交付项目
  - 挑战2：通过优化资源的分配与集成来满足预先定义的目标

## 基本概念

- **软件项目管理(Software Project Management):** 为了使软件项目能够按照预定的成本、进度、质量顺利完成,而对人员(People)、产品(Product)、过程(Process)和项目(Project)进行分析和管理的活动。
  - 软件项目管理的根本目的是为了**让软件项目尤其是大型项目的整个软件生命周期(从分析、设计、编码到测试、维护全过程)都能在管理者的控制之下,以预定成本按期,按质的完成软件交付用户使用。**
  - **挑战1: 在各类约束条件下交付项目**
  - **挑战2: 通过优化资源的分配与集成来满足预先定义的目标**
- **软件项目的特征 vs 项目管理:**
  - **软件产品的不可见性** → 软件项目复杂和抽象
  - **项目的高度不确定性** → 预定计划与实际情况存在较大偏差
  - **软件过程的多变化性** → 软件开发过程的不确定、不稳定
  - **软件人员的高技能及其高流动性** → 项目管理的风险

## 主要内容

**1 软件项目管理的案例**

**2 人员(People)**

**3 产品(Product)**

**4 过程(Process)**

**5 项目(Project)**

**6 可行性分析与估算**

**7 项目进度计划与监控**

**8\* 项目风险管理**

**9\* 项目质量管理**

## X项目的初始状态

- 一个年轻的项目经理A
- 10个经验缺乏的技术人员
- 项目交付期：紧张
- 技术风险：较高



## X项目的当前状态

- 进度已经落后于计划
- 项目经理A已经向客户汇报了一次项目开发进度，并已经演示了系统功能，已开发的功能已被用户接受并认可
- **此时：**项目组补充加入了一位水平高、经验丰富的技术人员B
- B检查了团队目前完成的代码，发现：原来写的代码效率不高，构架繁冗，不方便后期维护，也可能导致软件性能方面存在重大缺陷
- **B的建议：**重构代码和数据库设计

## X项目的当前状态

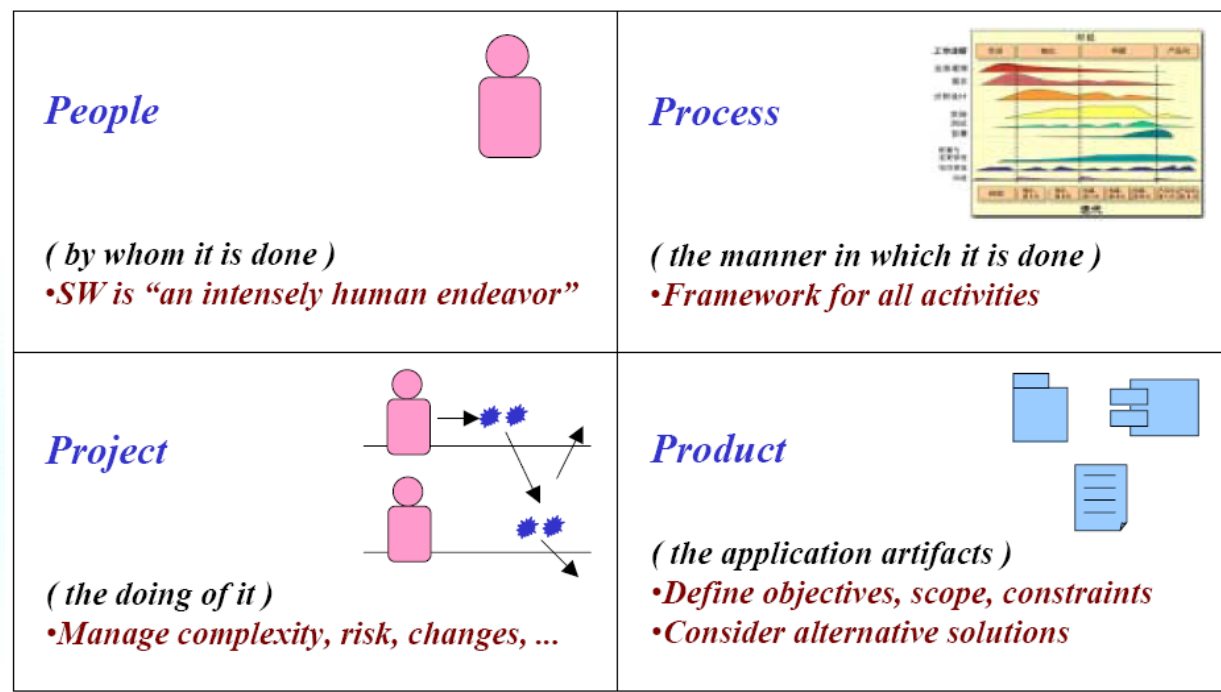
- **团队的想法：**辛苦写的代码被一票否决，心里很不舒服，但是也承认自己的代码质量不高
- **项目经理A的想法：**预计客户将来会提到这个问题，而届时再重构，会更加麻烦
- **公司CEO的意见：**已经向客户申请过一次计划调整并基本得到用户的理解，但目前进度已经落后于计划，急需完成剩余部分功能的开发，不能再次申请延期

## X项目的客观情况

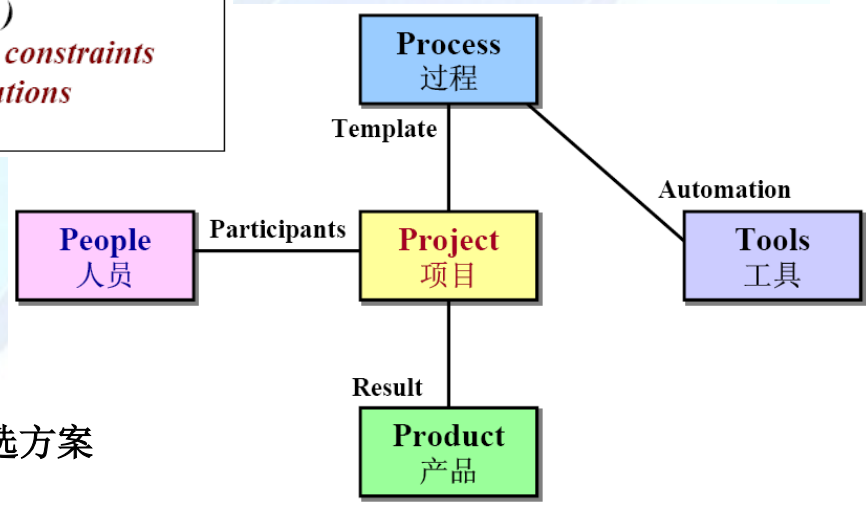
- 如果系统重构，很有可能需要再次调整计划，而用户已经明确表示计划是不可能再调整的了
- B在技术上没有问题，但是缺乏时间观念，且身兼多个项目，重构速度令人失望
- 老成员一开始挺配合系统重构，但是随着重构的深入，发现难度很大，基本等于重新开发；于是对B很有意见，不怎么配合B的指挥
- 最后，代码重构失败，导致项目严重拖期



# 软件项目管理的“4P”



- People:** 完成软件的主体是人，需要人做出非常大的努力
- Process:** 完成软件的方式，所有活动的框架
- Project:** 完成软件要做的事情，管理复杂性、风险和变化
- Product:** 应用软件制品，定义目标、范围和约束，考题备选方案



## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

7 项目进度计划与监控

8\* 项目风险管理

9\* 项目质量管理

## 软件项目的参与人员

- **高级管理者**：负责定义业务问题
  - **项目(技术)管理者**：计划、激励、组织和控制软件开发人员
  - **开发人员**：拥有开发软件所需技能的人员
    - 系统分析员、系统架构师、设计师、程序员、测试人员、质量保证人员、...
  - **客户**：进行投资、详细描述待开发软件需求、关心项目成败的组织/人员
  - **最终用户**：一旦软件发布成为产品，最终用户就是直接使用软件的人
- } 产品经理  
项目经理

## 软件开发团队

- “最好的”团队取决于项目经理的管理风格、团队里的人员数目与技能水平、项目的总体难易程度
- 组建团队时应考虑以下要素：
  - 从项目需求来看：
    - 待解决问题的难度
    - 待开发软件系统的规模
    - 待开发软件系统的技能要求
    - 交付日期的严格程度
    - 共同工作的时间
    - 彼此之间的人际关系与友好交际程度
    - .....
  - 从个人能力来看：
    - 应用领域经验
    - 开发平台经验
    - 编程经验
    - 教育背景
    - 沟通能力
    - 适应能力
    - 工作态度
    - 团队协作能力
    - .....

## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算



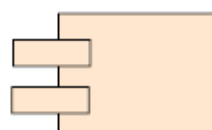
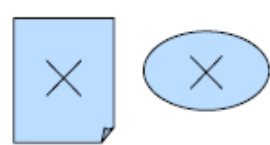

7 项目进度计划与监控

8\* 项目风险管理

9\* 项目质量管理



# 软件产品（提交物）

需求分析	软件设计	软件实现	软件测试	软件运行
				
<ul style="list-style-type: none"><li>• 用例模型</li><li>• 软件需求规格说明</li></ul>	<ul style="list-style-type: none"><li>• 软件体系结构描述</li><li>• 设计模型</li></ul>	<ul style="list-style-type: none"><li>• 源程序</li><li>• 目标代码</li><li>• 可执行构件</li></ul>	<ul style="list-style-type: none"><li>• 测试规程</li><li>• 测试用例</li></ul>	<ul style="list-style-type: none"><li>• 相关的运行时文件</li><li>• 用户手册</li></ul>
<div>开发管理文档</div> <div><div>计划文档<ul style="list-style-type: none"><li>- 工作分解结构</li><li>- 业务案例</li><li>- 发布规格说明</li><li>- 软件开发计划</li></ul></div><div>操作文档<ul style="list-style-type: none"><li>- 发布版本说明书</li><li>- 状态评估</li><li>- 软件变更申请</li><li>- 实施文档、环境</li></ul></div></div>				

## 软件产品、产品分解结构(PBS)

- 首先应确定软件范围：
  - 功能和非功能(性能、可用性、安全、法律等)
  - 软件范围应是确定的：在管理层和技术层都必须是无歧义的和可理解的
- 一旦确定了范围，需要对其进行分解——分而治之
- 项目管理通常使用“**产品结构分解(Product Breakdown Structure, PBS)**”作为产品分解的工具：
  - **PBS**：通过分层的树型结构来定义和组织项目范围内的所有产出物(产品)，自顶向下，逐级细分
  - **产出物**：项目结束时需要提交的最终产品，在项目之初就可以准确预计

## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

7 项目进度计划与监控

8\* 项目风险管理

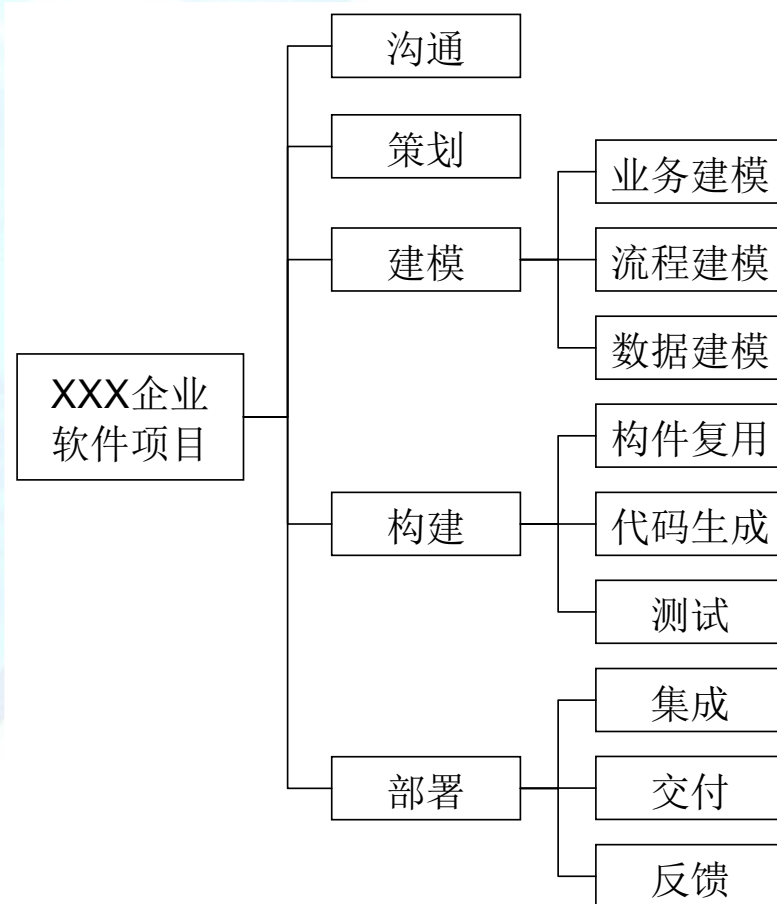
9\* 项目质量管理

## 软件过程

- Step 1: 选择合适的软件过程模型
  - 存在多种过程模型
  - 各过程模型适用不同类型的软件项目
- Step 2: 根据所选的过程模型，对其进行适应性修改
- Step 3: 确定过程中应包含的工作任务列表
  - [例]沟通活动：
    - 列出需澄清的问题清单
    - 与客户见面并说明问题
    - 共同给出范围陈述
    - 与所有相关人员一起评审
    - 根据需要修改范围陈述

## 工作分解结构(WBS)

- 项目管理里通常使用“**工作结构分解(Work Breakdown Structure, WBS)**”作为过程分解的工具
- **WBS**: 通过分层的树型结构来定义和组织工作任务之间的分解关系, 自顶向下, 逐级细分
  - [例]RAD过程模型的WBS分解结构





## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

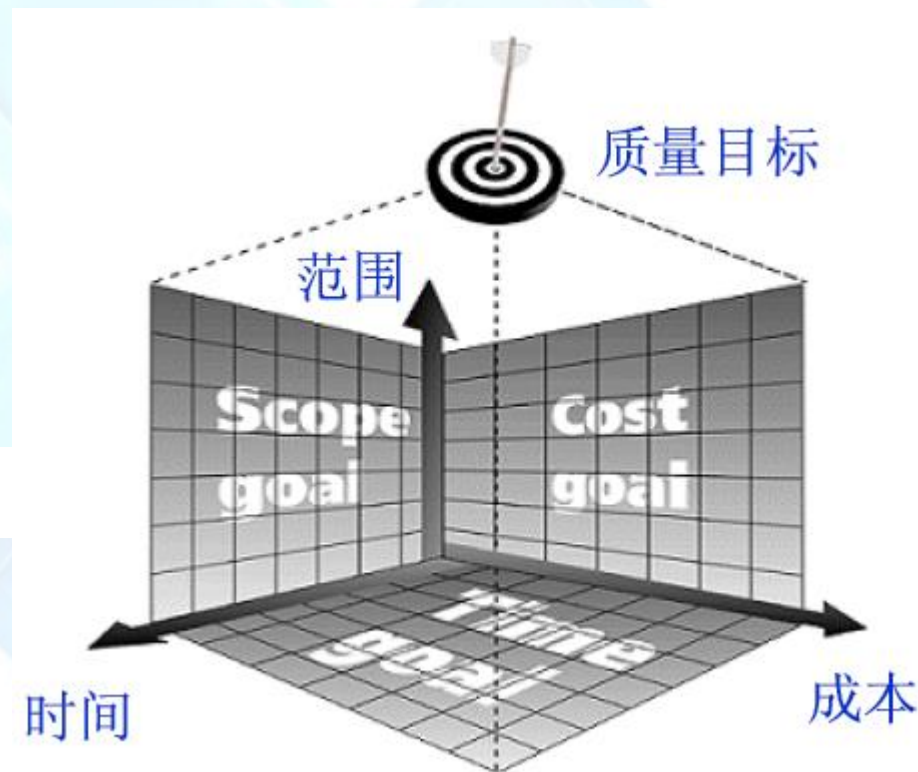
7 项目进度计划与监控

8\* 项目风险管理

9\* 项目质量管理

## 项目关注的四个方面

- 项目关注的四个方面
  - 范围(Scope)
  - 时间(Time)
  - 成本(Cost)
  - 质量(Quality)
- 项目管理的主要任务
  - 项目可行性分析与估算
  - 项目进度安排
  - 项目风险管理
  - 项目质量管理
  - 项目跟踪与控制



## W<sup>5</sup>HH原则

- **Why** 为什么要开发这个系统？
- **What** 将要做什么？
- **When** 什么时候做？
- **Who** 某功能由谁来做？
- **Where** 他们的机构组织位于何处？
- **How** 如何完成技术与管理工作？
- **How much** 各种资源分别需要多少？

## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

7 项目进度计划与监控

8\* 项目风险管理

9\* 项目质量管理

## 可行性分析与估算

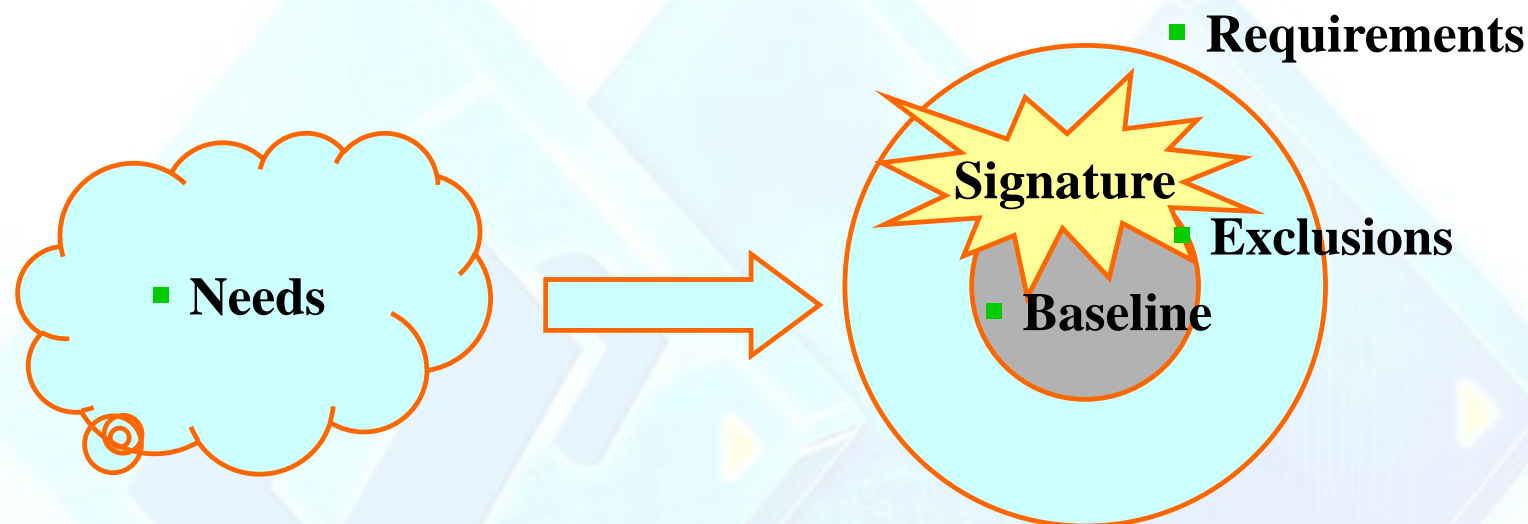
- 在项目开始之前，必须预先估计三件事情：
  - 需要多少工作量
  - 需要多少时间
  - 需要多少人员
- 此外，还必须预测所需要的资源(硬件和软件)以及蕴含的风险
- 从而得出“该项目是否可行”的结论



## 确定范围

- **范围(Scope):** 描述将要交付给最终用户的功能和特性、输入输出数据、用户界面、系统的性能、约束条件、接口和可靠性等，以及期望的时间、成本目标
- 两种方法：
  - 与所有项目成员交流之后，写出对软件范围的叙述性描述
  - 由最终用户给出一组用例
- 注意：
  - 并不是客户所有的需求都“来者不拒”，需要分别对待
  - 用户签字确认

## 确定范围



- **Needs:** 客户/最终用户的请求、想法和业务需求
- **Requirements:** 对未来系统所应具备的功能的陈述
- **Exclusions:** 不包含在未来系统中的功能的陈述
- **Baseline:** 对未来系统中应包含的功能的陈述

## 可行性分析

- 技术可行性：
  - 项目在技术上可行吗？它在技术水平范围内吗？能够将缺陷减少到一定程度吗？
- 经济可行性：
  - 它在经济上可行吗？能以可负担的成本完成开发吗？
- 时间可行性：
  - 项目投入市场的时间可以按预期完成吗？
- 资源可行性：
  - 组织拥有取得成功所需要的资源吗？

## 软件项目估算

- 如何估算时间、成本、资源？
  - 靠经验？
  - 靠数学公式？
- 到目前为止，因为变化的要素太多，所以对软件的估算从来没有达到精确
- 但是，估计得越精确，项目成功的可能性就越高
- 方法：
  - 代码行估算法
  - 功能点估算法
  - 用例点估算法
  - 类比估算法

## 软件项目估算

- 除此之外，还有其他很多估算方法
- 不同的方法采用不同的计算公式，考虑的因素不同，复杂程度也不同
- 都是根据实际项目的经验所总结出来的
- 不能说“谁好谁坏”，应用的时候可以依据自身的经验对其进行修正
- 阅读有关COCOMO II的相关材料
  - **COCOMO (COnstructive COst MOdel): 软件构造性成本模型**
  - **Barry Boehm提出的一种软件成本估算方法；使用一种基本的回归分析公式，从项目历史和现状中的某些特征作为参数来进行计算**
  - **主要用于工作量估算与成本估算**
  - **是最广泛使用和最全面的软件估算模型**



估算时间	程序员所想象的	程序员所忘记的	实际时间
30秒	只需要做一个很小的代码改动。我准确地知道怎么改，在哪里改。花费30秒敲键盘即可。	启动计算机，开发环境和获取正确源码的时间。用于构件，测试，检查和文档修复的时间。	1小时
5分钟	小事一桩，我只要上谷歌查一下语法就可以修复它了。	很少有一次就能找到完全正确的信息。即使找到，在它工作前，也需要做一些调整。外加构件，测试等等时间。	2小时
1 小时	我知道怎么做，但是写这些代码需要花费一些时间。	面对未来可能发生的问题，1小时稍纵即逝。有些东西总是会出错。	2小时
4小时	需要写一些代码，但是我粗略地知道步骤。我知道标准框架中的Wizzabanga模块可以做到，不过我得查看文档，了解它的准确地调用方式。	这个大概是唯一现实的估算。它为意外的错误留下了足够大的余地，而这个任务也小到足以把握。	4小时
8 小时	我先要把Balunga类重构成2个，然后为Wizzabanga模块加一个调用，最后为GUI加一些字段。	总会有许多系统的不同部分依赖着Balunga类。大概有40个不同的文件需要修改。为GUI新加的字段，同样也需要加到数据库中。8小时太长，无法完全把握。总会有比程序员估算时更多的步骤出现。	12-16小时
2 天	真的有一大堆代码要写。我需要往数据库里加一些新table，显示table的GUI，还有读写table的代码逻辑。	对于大多数开发者来说，两天的工作量已经大到难以估算了。肯定会有什么东西被遗漏掉。不仅仅是一些小事情，而是整个一大块主要功能会被遗忘在估算中。	5 天
1 周	哎哟，这真是一项艰巨的任务。虽然我还没有思路，但我不能说我不知道。一周应该够了，我希望，我真心希望，但是我不能要求更多了，否则他们会认为我不够称职。	这个任务已经大到超过大多数程序员的理解了。它应该被发回给架构师，帮忙将它划分成更小的部分，然后提供一些解决问题的方向。架构师可能会发现一种更简单的方法来完成它，或者发现其实有更多超乎想象的工作。。。	2-20 天

## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

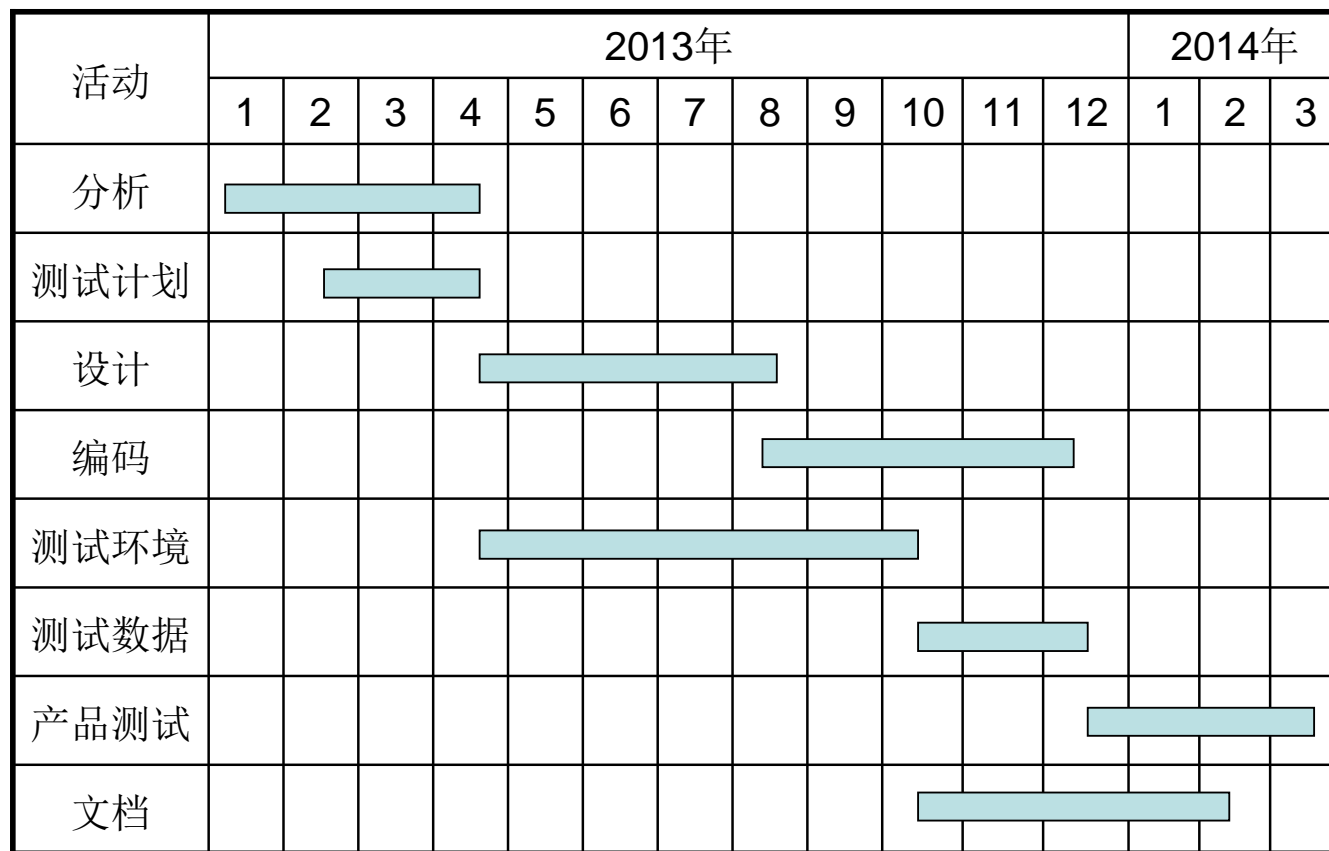
7 项目进度计划与监控

8\* 项目风险管理

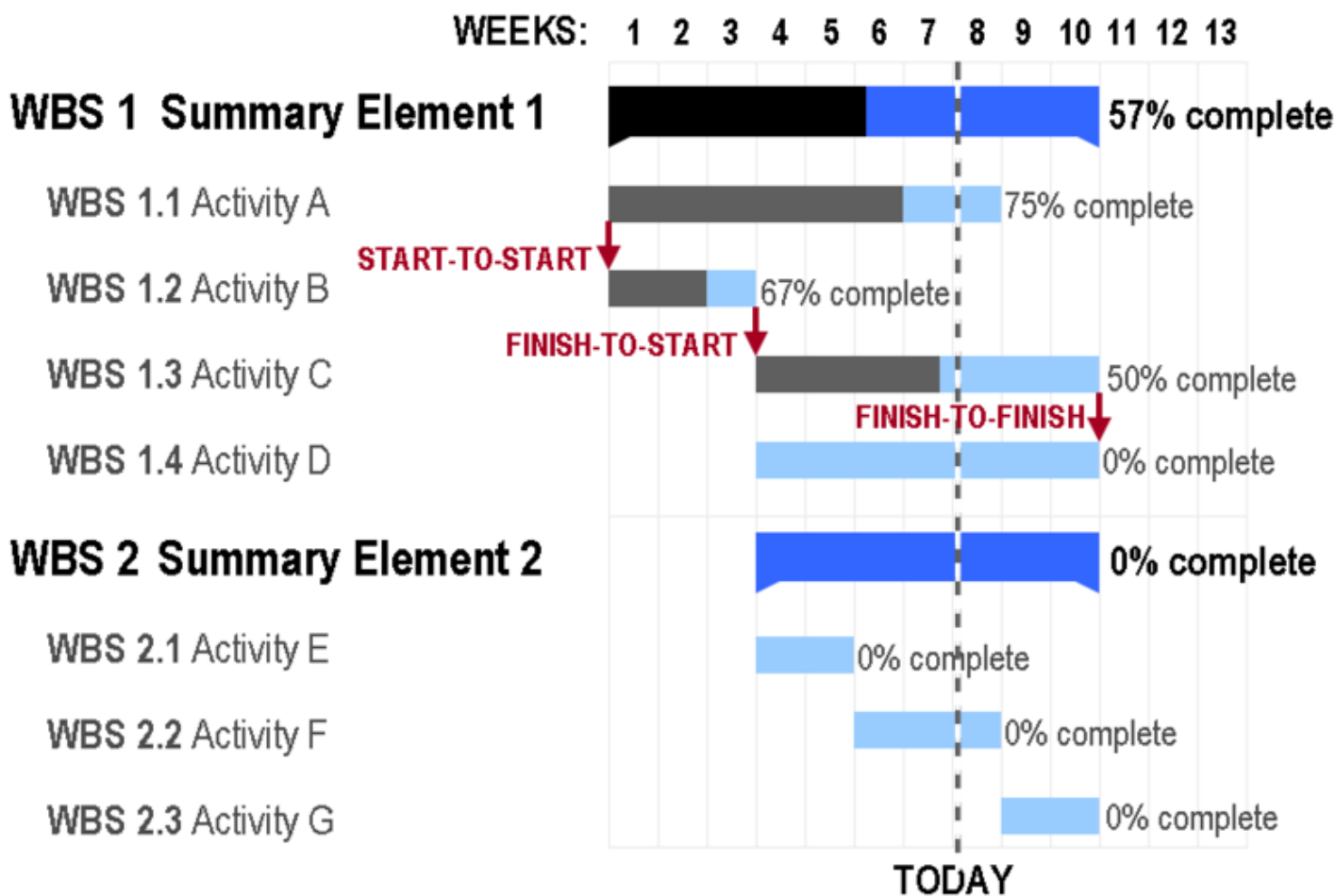
9\* 项目质量管理

## 绘制任务进度安排图

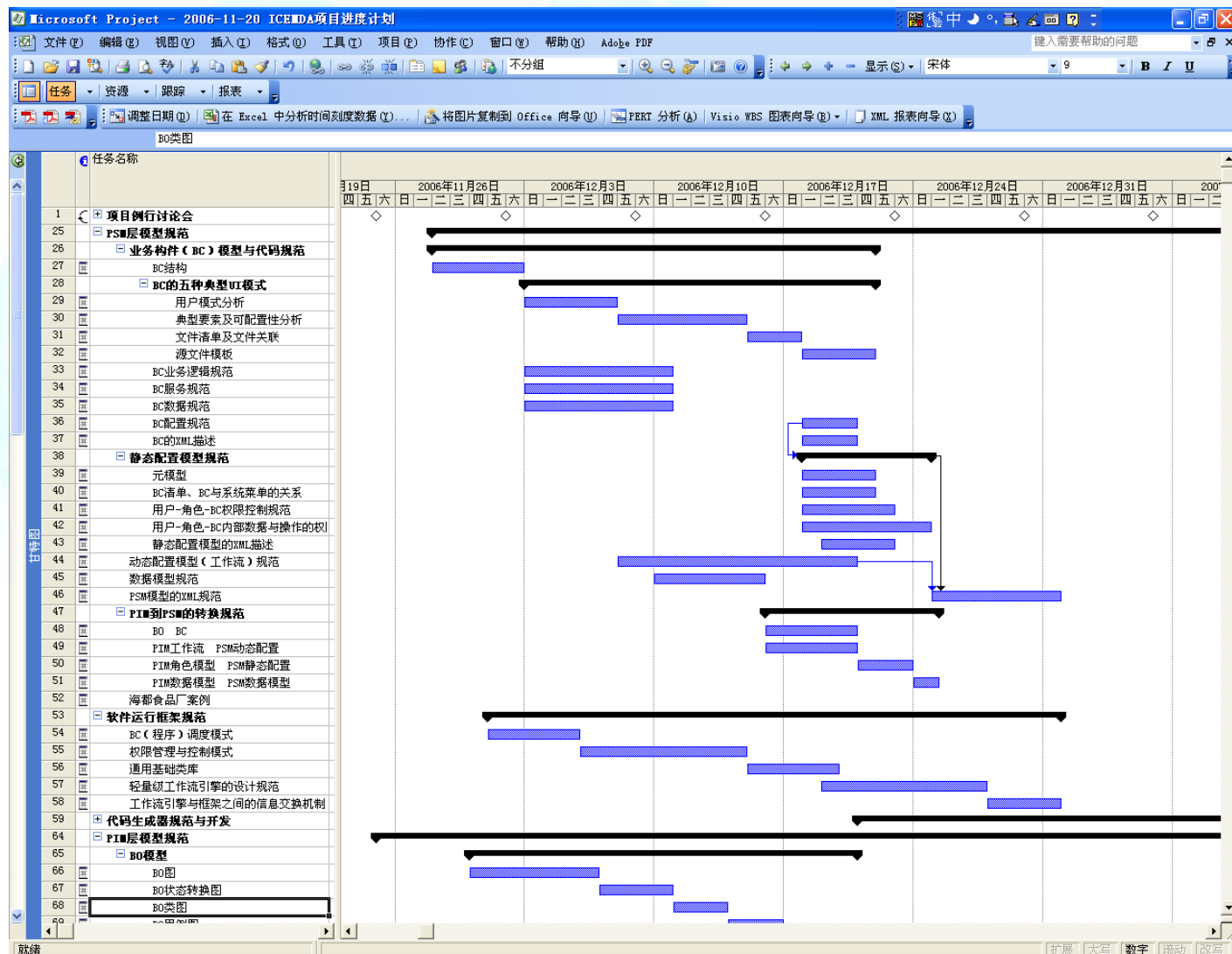
- 项目管理里通常采用甘特图(Gantt Chart)来描述任务的进度安排



## 甘特图(Gantt Chart)



## Microsoft Project 中的 Gantt 图

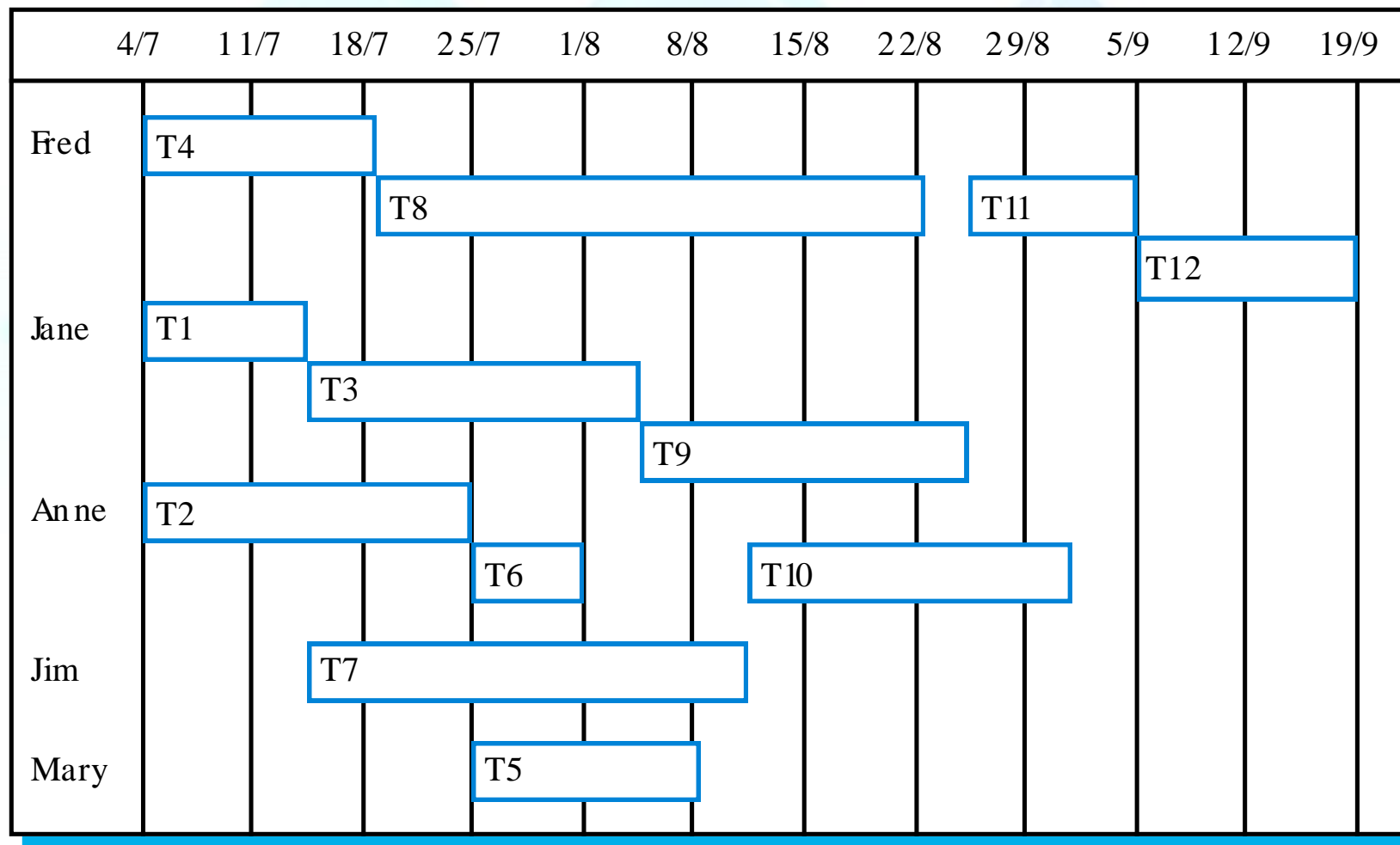




## 资源、产出与里程碑

- 将资源(resources)分配给任务
  - 资金
  - 人员
  - 设备
  - 环境
- 明确产出结果(outcomes)
  - 每一项任务的产出结果是什么？对应于PBS中的哪一部分？
- 明确里程碑(milestones)
  - 项目的关键产出物，标志着某一阶段的完成

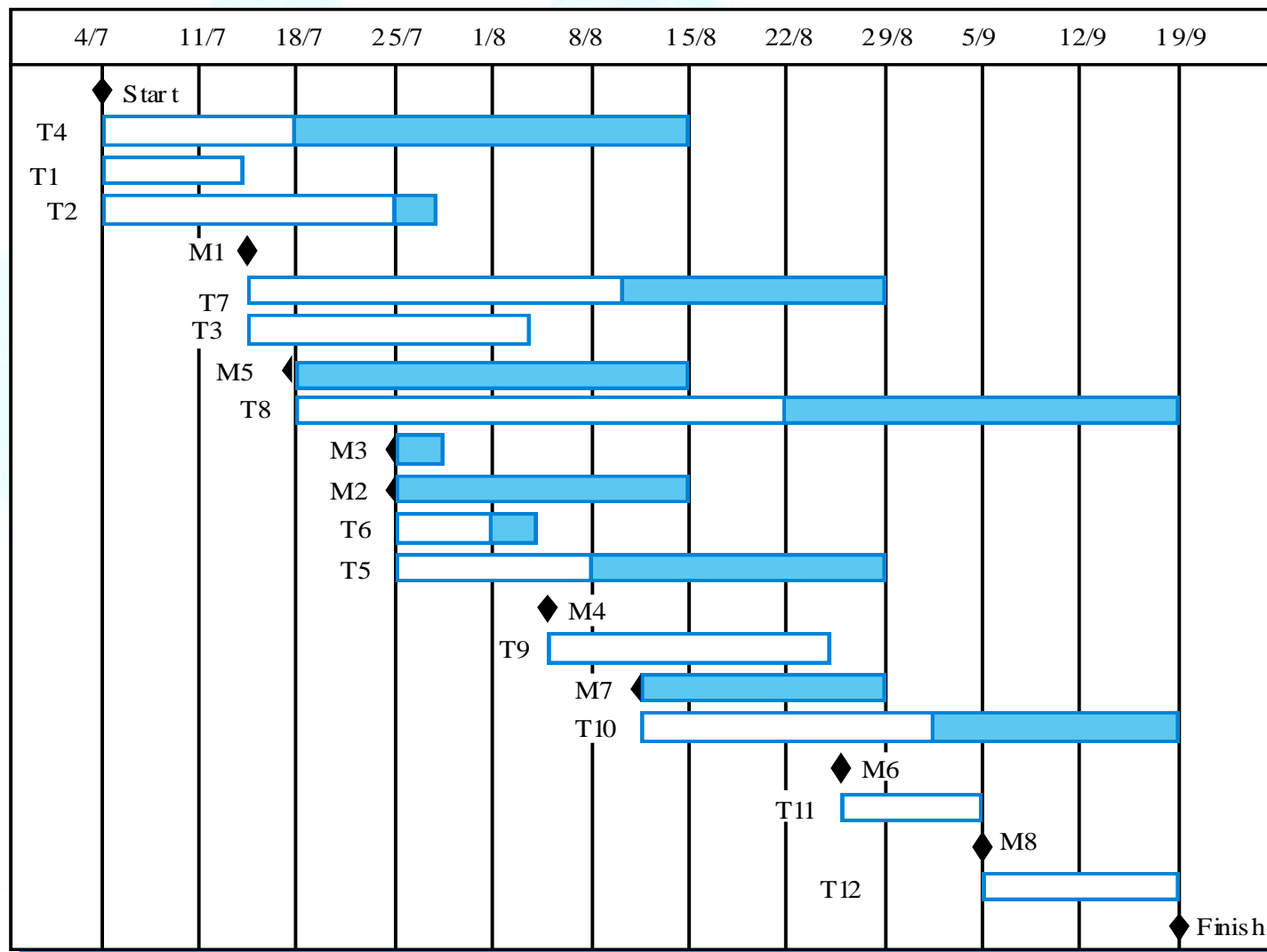
# 人员/资源分配图



## 项目进度跟踪

- 项目进度表只是提供了一张进度路线图，在实际执行过程中，需要定期对其进行跟踪和控制，以决定是否需要对进度计划进行调整
  - 定期举行项目状态会议，各成员分别报告进展和存在问题
  - 评审进展和产出物
  - 判断项目里程碑是否在预定日期内完成
  - 比较各项目的实际开始/结束日期与计划开始/结束日期
  - 找出问题，并寻找对策
  - 定量评估项目进展
  - 决策是否需要调整

# 项目进度跟踪Gantt图



## XP/Scrum敏捷开发中的进度计划与监控

- 以“迭代”为单位：每次迭代包含多少个用户故事或用例
- 每次迭代为30天左右
- 针对每个用户故事，团队成员联合估算和协商开发代价(时间)
- 使用任务墙(Task Board)/燃尽图(Burndown Chart)等作为进度监控工具，评估迭代的当前进展情况



## 敏捷开发的项目管理工具：禅道ZenTaoMPS

- 禅道是第一款国产的开源项目管理软件，它的核心管理思想基于敏捷方法Scrum，内置了产品管理和项目管理
- 完全符合Scrum全元素、全流程

禅道ZenTaoMPS界面截图展示了其项目管理功能。左侧为导航栏，包含地盘、产品、迭代、执行、测试、DevOps、看板、文档、组织等模块。右侧主区域显示了“设置”和“冲刺”两个主要功能。

**设置 (Settings) 界面：**

- 系统设置：备份、聊天、安全等系统各要素配置。
- 功能开关：打开、关闭系统部分功能。
- 人员管理：维护部门、人员、分组。
- 模型配置：二次开发支持。
- 功能配置：支持对系统功能进行配置。
- 通知设置：配置通知路径，自定义需要通知的动作。
- 插件管理：浏览、安装插件。

**冲刺 (Sprints) 界面：**

显示了当前产品的冲刺列表。表格列出了冲刺名称、所属产品、状态、负责人、计划开始、计划完成、预计、消耗、剩余、进度、燃尽图和操作。

名称	所属产品	状态	负责人	计划开始	计划完成	预计	消耗	剩余	进度	燃尽图	操作
冲刺1	通用批发零售业务	未开始	马超	2023-04-24	2023-05-05	0h	0h	0h	0		[图标]
冲刺2	通用批发零售业务	未开始	马超	2023-05-08	2023-05-19	0h	0h	0h	0		[图标]
冲刺3	通用批发零售业务	未开始	马超	2023-05-22	2023-06-02	0h	0h	0h	0		[图标]

底部显示：本页共3个冲刺，未开始3，进行中0。共3项 每页100项

**研发需求 (R&D Requirements) 界面：**

显示了当前产品的研发需求列表。表格列出了需求ID、名称、计划、状态、预计、评审者、阶段、指派给和操作。

ID	研发需求名称	计划	状态	预计	评审者	阶段	指派给	操作
010	确认通过	激活	0h	陈宏	未开始	未指派	[图标]	
009	有待明确	评审中	0h	未开始	未指派	[图标]		
008	拒绝	评审中	0h	未开始	未指派	[图标]		

底部显示：选中1个研发需求，预计0.0工时，用例覆盖率0%。共3项 每页20项

## 敏捷开发的项目管理工具：VersionOne

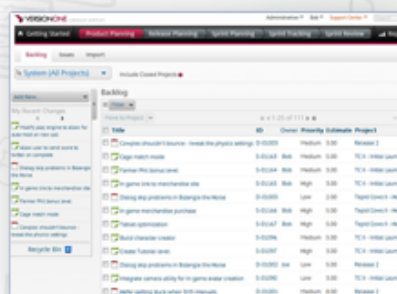
- <http://www.versionone.com>
- 敏捷领域最流行的商业化项目管理工具之一



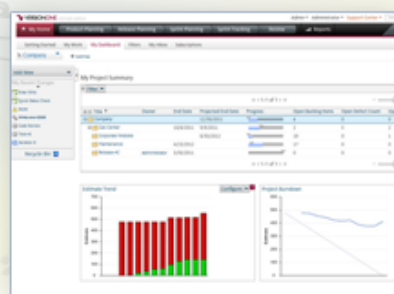
[Product](#) [Platform](#) [Training](#) [Customers](#) [Partners](#) [About Us](#)

## Agile Made Easier

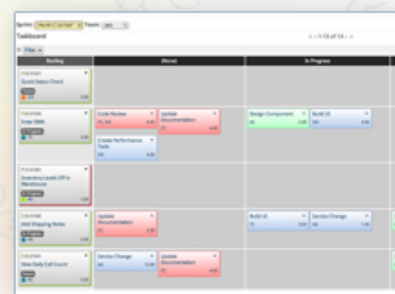
All-in-one agile management tool for projects and teams of any size.



All of your projects & teams in one place



Visibility across your software lifecycle



Easy collaboration with team members

Join more than 50,000 teams

[Try VersionOne](#)

[Why VersionOne](#)

[Pricing & Editions](#)

[Join a Live Demo](#)

## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

7 项目进度计划与监控

8\* 项目风险管理

9\* 项目质量管理

## 软件项目风险

- 软件规模风险：
  - 估算准确程度？
  - 用户需求可能发生变化的频度与规模？
- 商业影响风险：
  - 交付期限？
  - 政府出台新政策？
- 客户相关风险：
  - 陌生客户？客户高层的重视程度？
  - 客户的配合程度？
- 软件过程风险：
  - 开发者不了解/不熟悉选定的过程模型？
  - 没有维护足够的文档？
- 开发环境风险：
  - 无法得到可用的工具？
  - 没有或不会使用工具？
- 开发技术风险：
  - 之前无该技术的经验？
  - 该技术难以实现某些需求？
- 开发人员风险：
  - 没有足够的经验与技能？
  - 某些人员会中途离开？

## 风险预测与分析

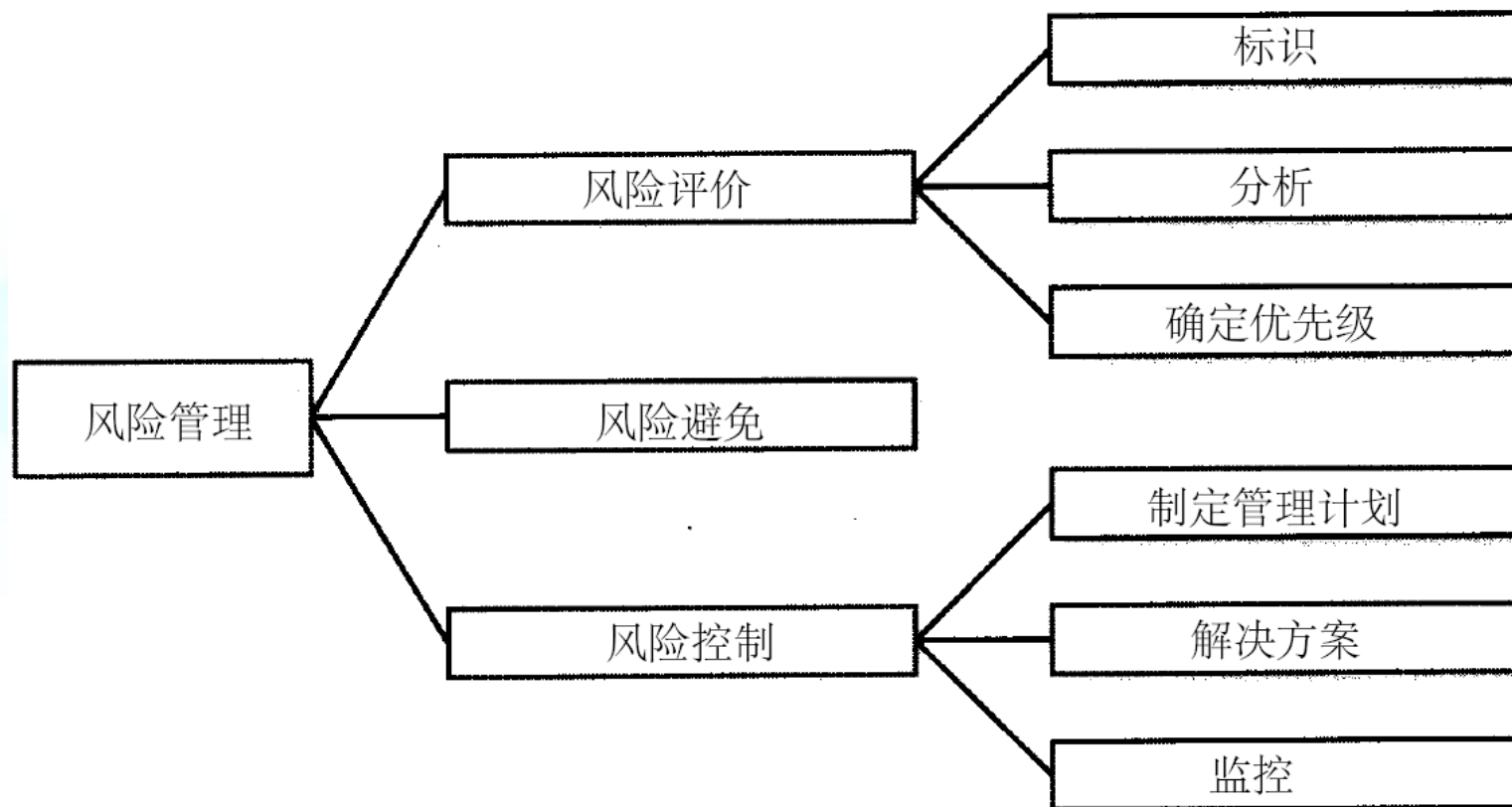
- Step 1: 列出可能的风险
- Step 2: 估计风险发生的可能性或概率
- Step 3: 建立风险表
- Step 4: 估计风险可能产生的影响或后果
- Step 5: 风险求精
- Step 6: 风险环节、监测和管理
- Step 7: 风险应急计划



## 风险预测与分析

风险	风险类型	概率	影响程度	后果	应急计划
规模估算不准确	产品规模	60%	严重的	...	...
用户数量超出想象	产品规模	30%	轻微的	...	...
最终用户抵制新系统	产品规模	70%	严重的	...	...
交付日期将推迟	商业影响	40%	灾难的	...	...
用户将改变需求	产品规模	50%	轻微的	...	...
技术到不到预期效果	开发技术	40%	灾难的	...	...
人员缺乏经验	人员	80%	严重的	...	...
缺少对工具的培训	开发环境	30%	可忽略的	...	...
人员变动频繁	人员	80%	严重的	...	...
.....					

## 风险管理的其他方面



## 主要内容

1 软件项目管理的案例

2 人员(People)

3 产品(Product)

4 过程(Process)

5 项目(Project)

6 可行性分析与估算

7 项目进度计划与监控

8\* 项目风险管理

9\* 项目质量管理

## 项目质量管理

- 观点1: “质量不是检验出来的, 而是设计/开发出来的”
  - 需要在软件全生命周期内考虑最终产品的质量
- 观点2: “评审、评审、再评审”
  - 准备SQA计划; 定期评审; 记录偏差; 改善
- 观点3: “产品/过程二象性”
  - 质量管理需要同时考虑产品与过程两个方面
- 观点4: “越往后, 后果越严重”
  - 早期的质量问题既容易发现, 也容易消除; 而后期的质量问题将带来严重后果

## 项目质量管理

- 观点5: “缺陷放大”
  - 如果早期犯下的错误没有发现, 将会在随后的过程里无休止的放大
- 观点6: “犯错误的是人, 但错误是在产品中存在的”
  - 重点关注产品, 不要去针对人
- 观点7: “不要被表面现象所迷惑”
  - 找到质量问题之后, 要深究和追随其内部的原因, 会挖出更大的问题