



# 软件工程

## 第三章 软件项目管理

乔立民

[qlm@hit.edu.cn](mailto:qlm@hit.edu.cn)

2011年4月6日

# 主要内容

## **3.1 软件项目管理概述**

### **3.1.1 项目管理概念及特征**

### **3.1.2 软件项目的“4P”**

### **3.1.3 软件项目策划过程**

## **3.2 软件项目估算**

## **3.3 项目进度安排**

## **3.4 项目风险管理**

## 若干基本概念

- **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；
- **项目(Project)**：精心定义的一组活动，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(例如人)，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；

# 软件项目的特征

- **软件产品的不可见性：**开发过程和产品都是看不见摸不着的，导致软件项目特别复杂和抽象；
- **项目的高度不确定性：**项目的估算与计划非常困难，有很多难以预见的问题，造成预定计划于实际情况存在较大偏差；
- **软件过程的多变化性：**迭代、增量开发、动态变化、不确定、不稳定；
- **软件人员的高技能及其高流动性：**智力密集型活动、对人的要求高、核心技术人才流动性高。

# 主要内容

## 3.1 软件项目管理概述

### 3.1.1 项目管理概念及特征

### 3.1.2 软件项目的“4P”

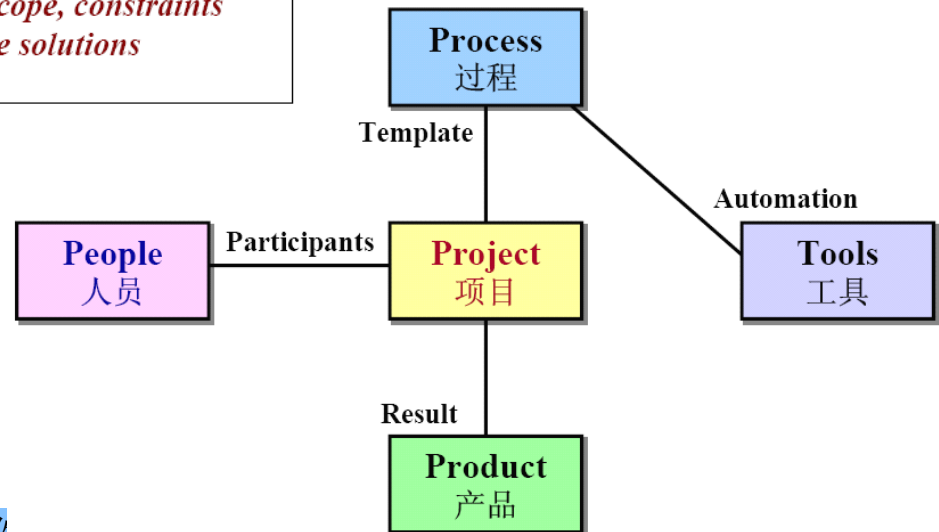
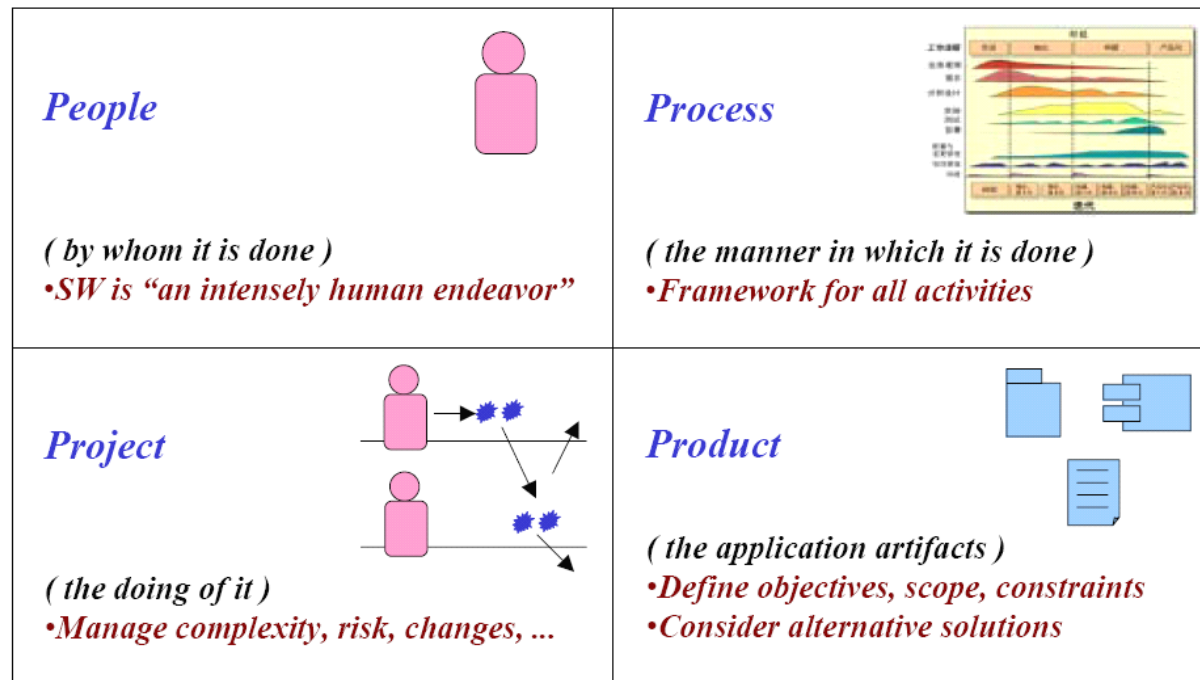
### 3.1.3 软件项目策划过程

## 3.2 软件项目估算

## 3.3 项目进度安排

## 3.4 项目风险管理

# 软件项目管理的“4P”



# People——软件项目的参与人员



# 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。
- } 项目经理



# 项目经理(Project Manager)

- 最重要的：领导力（**MOI**模型）

- Motivation (激励): 通过“推”或“拉”鼓励项目成员发挥其最大才能与潜力；
- Organization (组织): 形成能够将最初需求转换为最终产品的能力；
- Idea or Innovation (思想或创新): 即使在诸多约束条件下工作，也能鼓励项目成员去创造新的想法。

- 项目经理的关键品质

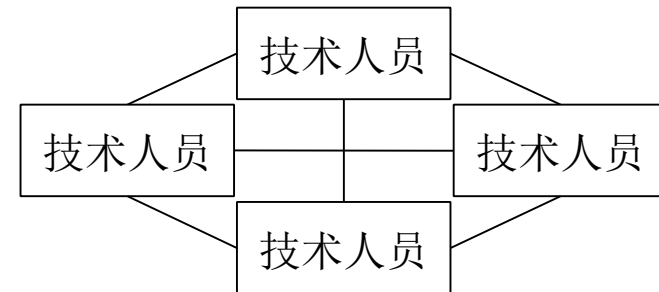
- 解决问题：准确诊断出关键技术问题、组织问题，制定解决方案
- 管理者特性：掌控整个项目，具有领导力
- 成就：通过激励优化团队效率
- 影响和队伍建设：理解“人”，在高压环境下保持控制力

# 软件开发团队

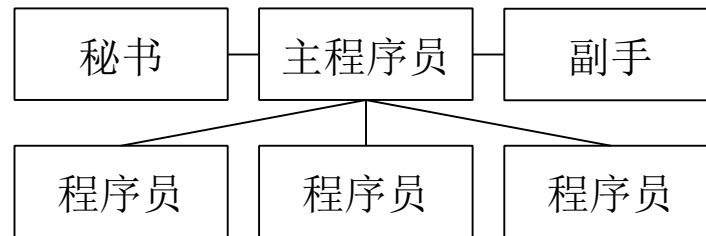
- “最好的”团队取决于项目经理的管理风格、团队里的人员数目与技能水平、项目的总体难易程度；
- 组建团队时应考虑以下要素：
  - 从项目需求来看：
    - 待解决问题的难度；
    - 待开发软件系统的规模；
    - 待开发软件系统的技能要求；
    - 交付日期的严格程度；
    - 共同工作的时间；
    - 彼此之间的人际关系与友好交际程度；
    - .....
  - 从个人能力来看：
    - 应用领域经验
    - 开发平台经验
    - 编程经验
    - 教育背景
    - 沟通能力
    - 适应能力
    - 工作态度
    - 团队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

# 软件开发团队的组织方式

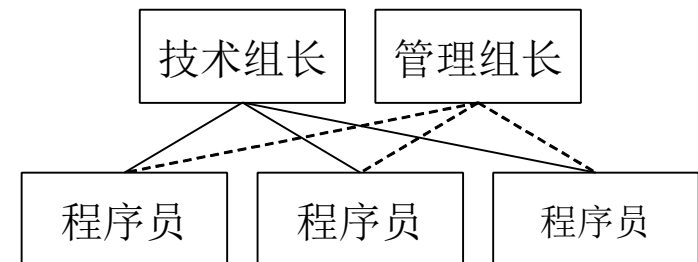
- 民主式：小组成员完全平等；



- 主程序员式（外科手术）：一个人全面负责、其他人给予支持；

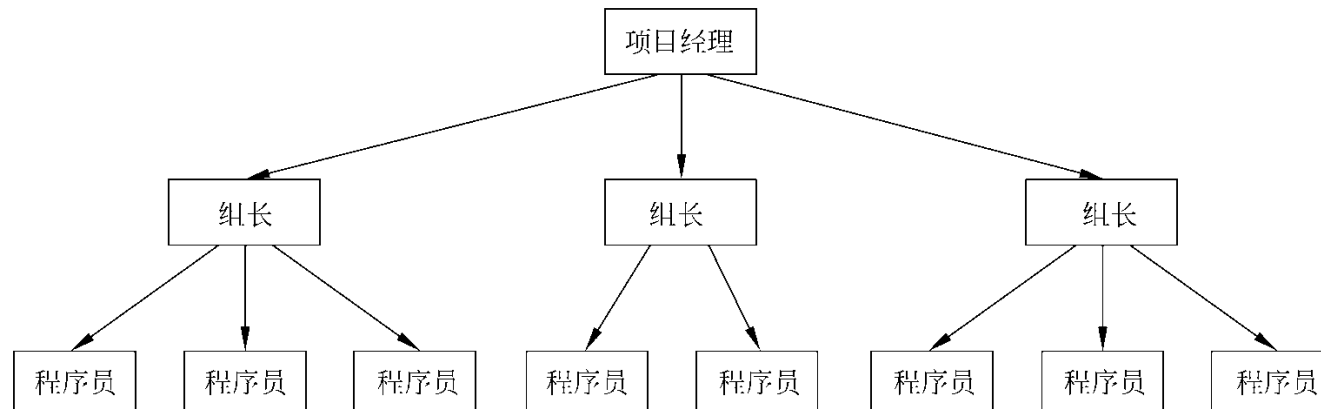


- 技术管理式：综合上述二者的特征；



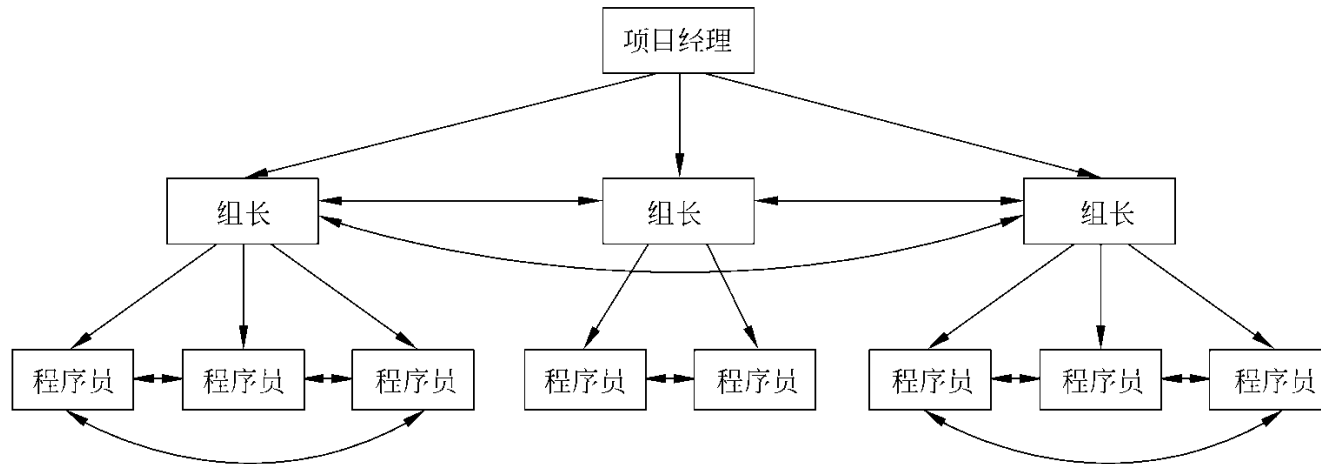
- 上述三种项目的组织方式各有什么利弊？

# 大型项目的技术管理组织结构



图例：

——> 技术管理

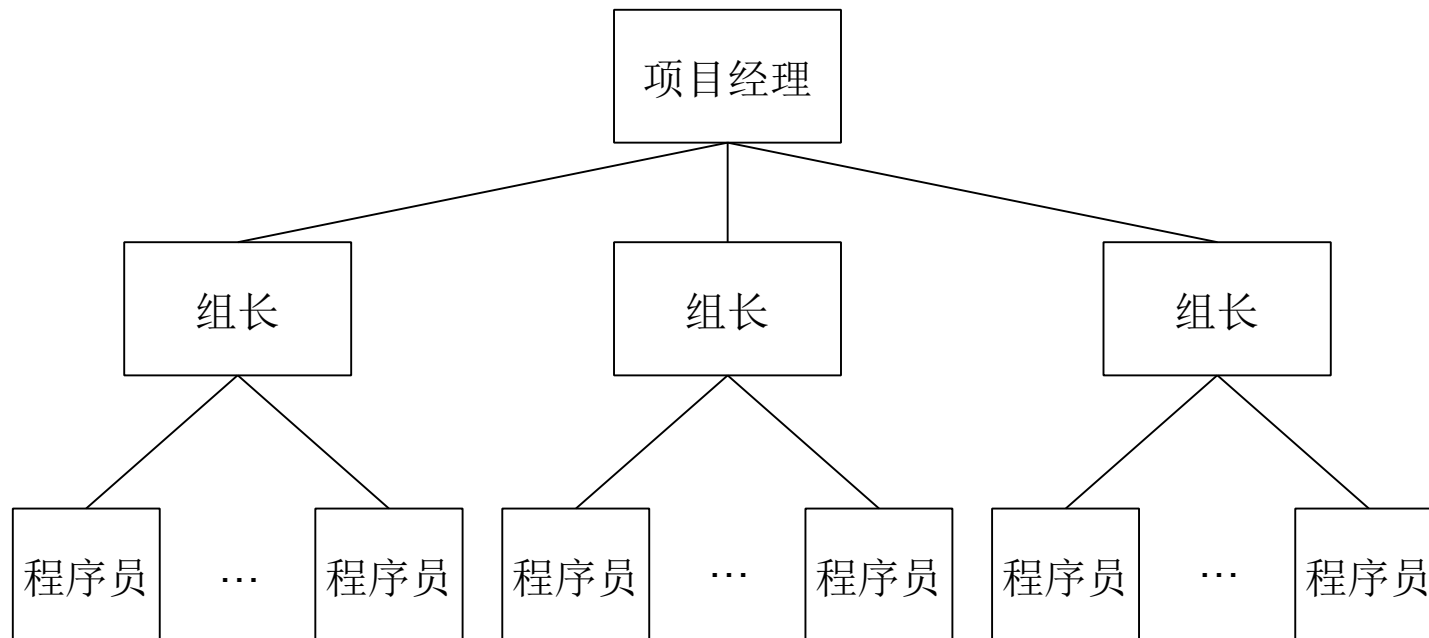


图例：

——> 技术管理

# 组织分解结构(OBS)

- 项目管理里通常使用“**组织结构分解(Organization Breakdown Structure, OBS)**”作为描述组织/人员之间关系的工具:




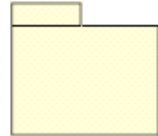
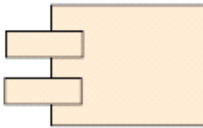
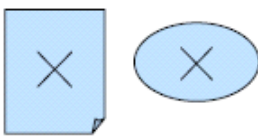
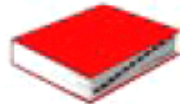
# 人员协调与沟通

- 问题1：为什么需要沟通？
- 问题2：沟通的方式有哪些？
  - 面对面交谈、电话交谈、email、面对面会议、电话会议、网络会议、项目网站、书面报告；
- 问题3：项目沟通活动有哪些？
  - 规划项目沟通；
  - 实施阶段性评审；
  - 每周小组会议；
  - .....

# 敏捷团队

- 小型充满活力的团队
- 强调团队成员的个人能力与团队协作精神相结合
- 自组织团队
  - 项目管理自主权
  - 技术决定权
  - 计划制定工作压缩到最低
  - 团队选择自己适用的手段（过程、方法、工具）

# Product——软件产品

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>				



# product——软件产品

## ■ 确定软件范围

- 项目环境
- 信息目标
- 功能和非功能(性能)
- 在管理层和技术层都必须是无歧义的和可理解的，软件范围应是确定的；

## ■ 问题分解（软件需求分析的核心活动）：分治的思想

- 必须交付的功能
- 所使用的过程

[例] 文档编辑产品

文本输入  
编辑及格式设计  
自动复制编辑  
页面布局能力  
自动生成索引和目录  
文件管理  
文档生成

拼写检查  
语句文法检查  
大型文档的引用检查  
大型文档中章节引用确认

# 产品分解结构(PBS)

- 项目管理里通常使用“**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的预计。

**PBS第3层**

1. 软件项目

1.1 可行性分析报告

1.1.1 时间可行性	2
1.1.2 成本可行性	2
1.1.3 人员可行性	1
1.1.4 技术可行性	3

1.2 需求分析报告 15

1.3 设计报告

1.3.1 概要设计书	13
1.3.2 详细设计书	17

1.4 源代码 5

1.5 测试用例 15

1.6 测试记录

1.6.1 测试环境	2
1.6.2 测试记录	4
1.6.3 测试结果	4
1.6.4 修改记录	5

1.7 项目管理文档 12

产品结构分解  
(PBS)

**PBS第2层**

1. 软件项目

1.1 可行性分析报告	8
1.2 需求分析报告	15
1.3 设计报告	30
1.4 源代码	5
1.5 测试用例	15
1.6 测试记录	15
1.7 项目管理文档	12

100

**PBS第1层**

1. 软件项目

00

100

# Process——软件过程

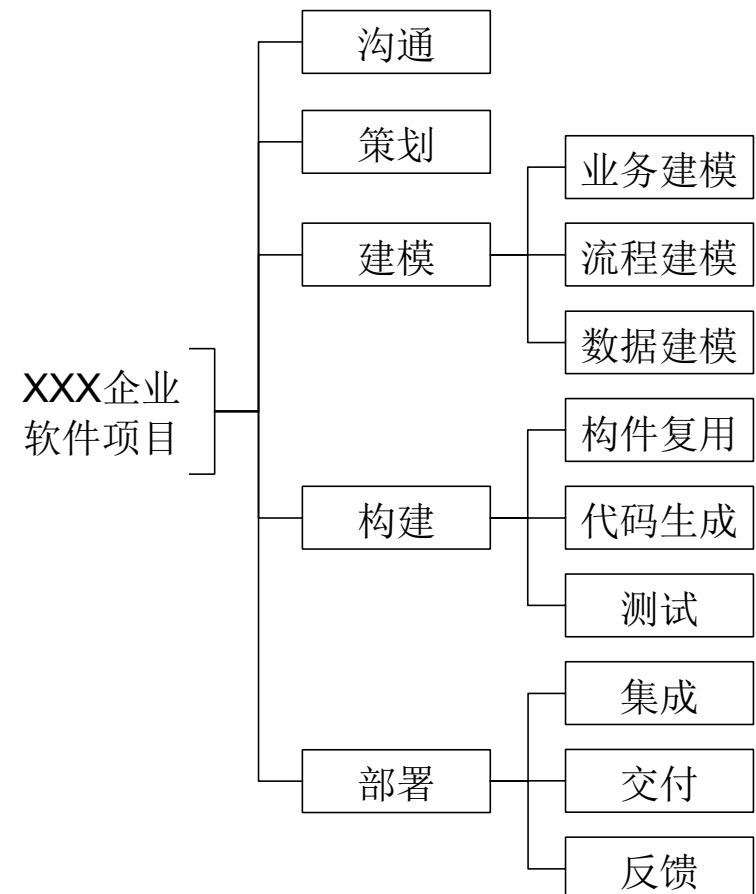
- **Step 1:** 选择软件过程模型，适合于：
  - 需要该产品的客户和从事开发工作的人员
  - 产品本身的特性
  - 软件项目团队工作的项目环境
- **Step 2:** 基于过程框架制定初步的项目计划（改进过程以适应项目）
- **Step 3:** 过程分解，制定完整计划，确定工作任务列表；
  - [例]沟通活动：
    - 列出需澄清的问题清单；
    - 与客户见面并说明问题；
    - 共同给出范围陈述
    - 与所有相关人员一起评审；
    - 根据需要修改范围陈述。

# 过程分解

- 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



## “产品”与“过程”的合并

- 还记得软件项目的“产品与过程二象性”吗？
- 将“产品分解结构**PBS**”与“工作分解结构**WBS**”之间建立联关系。

# 合并产品和过程

资源需求  
开始/结束日期  
工作产品

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。

通用过程框架活动	沟通			策划			建模			构建			部署	
软件工程任务														
产品功能														
文本输入														
编辑及格式设计														
自动复制编辑														
页面布局能力														
自动生成索引和目录														
文件管理														
文档生成														

## 例：复杂的沟通活动

1. 评审客户要求
2. 计划并安排与客户召开正式的、有人主持的会议
3. 研究如何详细说明推荐的解决方案和已有的方法
4. 为正式会议准备一份“工作文档”和议程
5. 召开会议
6. 共同制定能够反映软件的数据、功能和行为特征的小规格说明
7. 评审每一小份小规格说明或用例，确认其正确性、一致性和无歧义性
8. 将这些小规格说明组装起来形成一份范围文档
9. 和所有相关人员一起评审范围文档或用例集
10. 根据需要修改范围文档或用例



# Project——项目

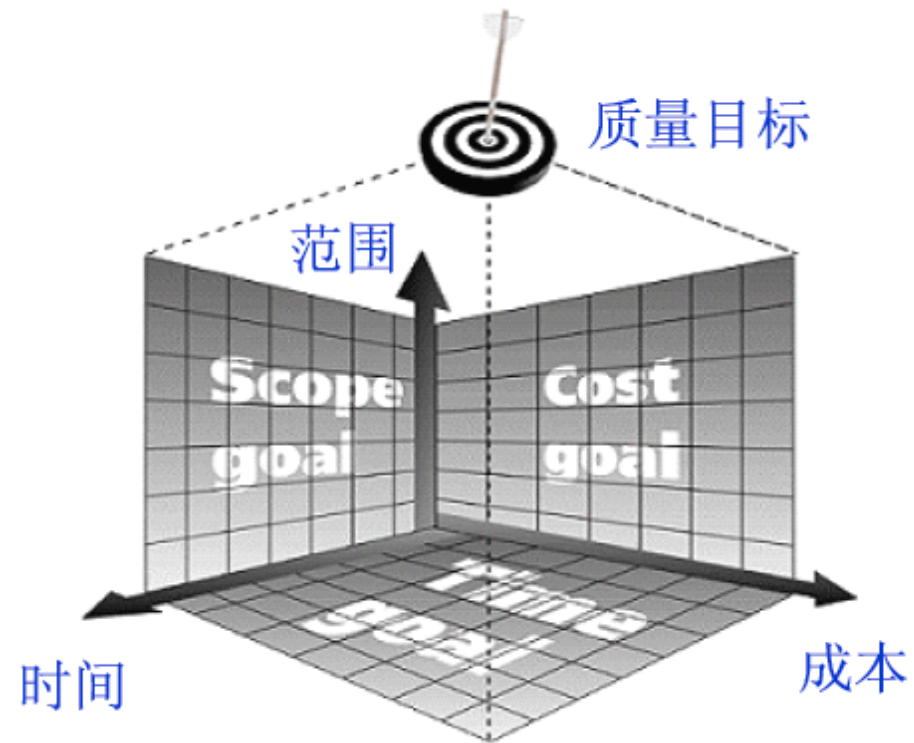
## ■ 项目管理中的危险信号

- 软件人员不了解其客户的要求
- 产品范围定义的很糟糕
- 没有很好地管理变更
- 选择的技术发生了变化
- 业务需求发生变化（或未被很好地定义）
- 最后期限是不切实际的
- 客户抵制
- 失去赞助（或从来没有真正得到过赞助）
- 项目团队缺乏具有合适技能的人员
- 管理者（和实践者）没有很好地利用已学到的最佳实践和教训

## 90—90规则

# 项目关注的四个方面

- 质量(Quality)
- 范围(Scope)
- 时间(Time)
- 成本(Cost)



# Project——项目

## ■ **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

## 3.1 软件项目管理概述

### 3.1.1 项目管理概念及特征

### 3.1.2 软件项目的“4P”

### 3.1.3 软件项目策划过程

## 3.2 软件项目估算

## 3.3 项目进度安排

## 3.4 项目风险管理

# 项目策划过程

## ■ 软件通用过程框架

- 沟通：项目启动、需求获取
- 策划：项目估算、资源需求、进度计划
- 建模：创建模型，进行分析和设计
- 构建：编码和测试
- 部署：软件交付，支持和反馈

## ■ 策划的目标：提供一个能使管理人员对资源、成本及进度做出合理估算的框架

- 定义“最好的情况”、“最坏的情况”
- 软件项目的不确定性与动态调整

# 项目策划任务集

## 1. 规定项目范围

## 2. 确定可行性

## 3. 分析风险

## 4. 确定需要的资源

- 确定需要的人力资源
- 确定可复用的软件资源
- 标识环境资源

## 5. 估算成本和工作量

- 分解问题
- 使用规模、功能点、过程任务或用例等方法进行两种以上的估算
- 调和不同的估算

## 6. 制定项目进度计划

- 建立一组有意义的任务集合
- 定义任务网络
- 使用进度计划工具制定任务表
- 定义进度跟踪机制

# 主要内容

## 3.1 软件项目管理概述

## 3.2 软件项目估算

### 3.2.1 可行性分析

### 3.2.2 软件项目估算

## 3.3 项目进度安排

## 3.4 项目风险管理

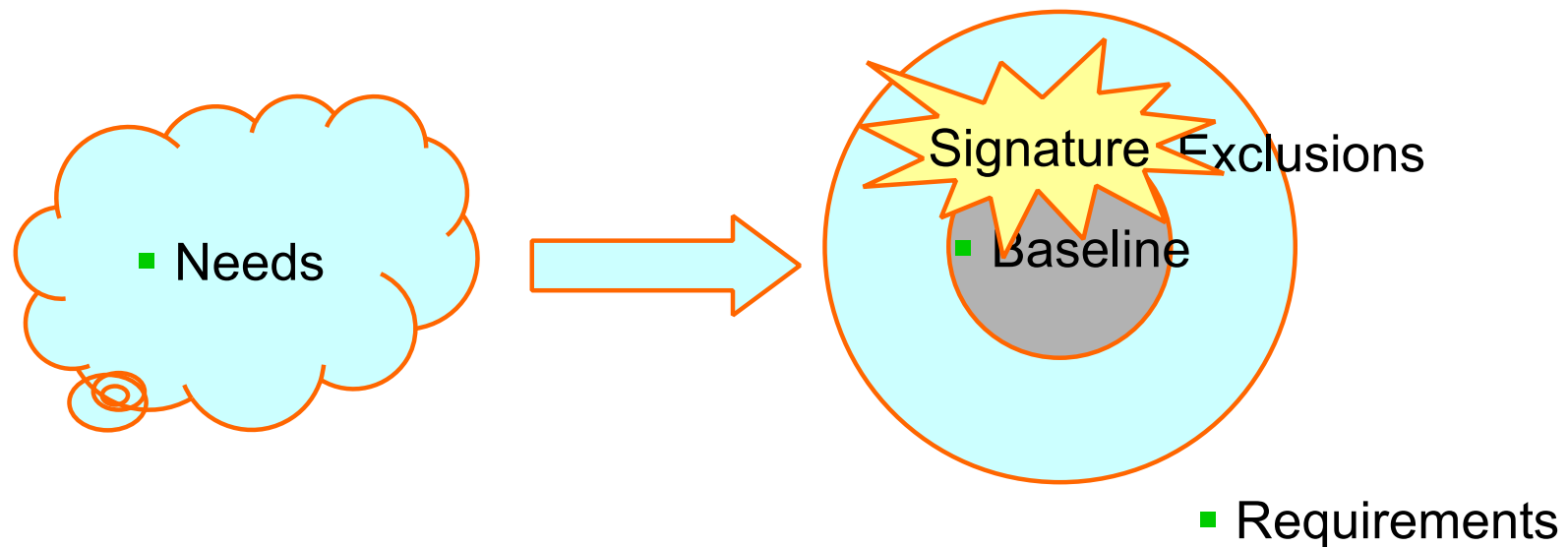
# 估算与可行性分析

- 软件估算的内容（定量）
  - 需要多少工作量（成本）
  - 需要多少时间（进度）
  - 需要多少人员（资源）
- 软件估算的依据
  - 历史信息
  - 经验
  - 定量预言
- 软件估算风险
  - 定量的不确定性
  - 资源不稳定性

## 软件项目估算的动态调整



# 1 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: 将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。

# 1 确定范围

- **范围(Scope):** 描述了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；
  - 由最终用户开发一组用例。
- **注意**
  - 并不是客户所有的要求都“来者不拒”，需要分别对待！
  - 软件范围一定要用户同意

# 范围的可行性分析

## ■ 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？

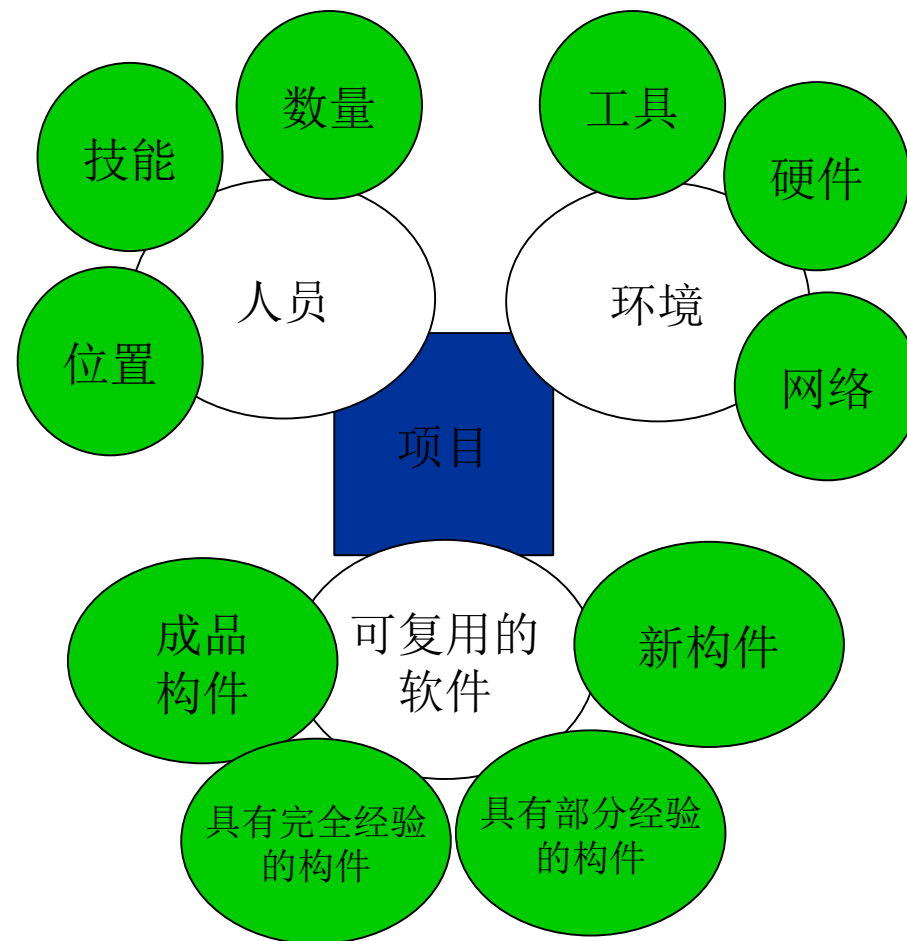
## 2 确定资源

### ■ 资源

- 人力资源
- 可复用的软件资源
- 环境资源

### ■ 说明资源的特征

- 资源的描述
- 可用性说明
- 何时需要资源
- 使用资源的持续时间



# 主要内容

## 3.1 软件项目管理概述

## 3.2 项目可行性分析与估算

### 3.2.1 可行性分析

### 3.2.2 软件项目估算

- 分解技术
- 经验估算模型

## 3.3 项目进度安排

## 3.4 项目风险管理

# 软件项目估算（成本和工作量）

## ■ 软件项目估算方式

- 把估算推迟到项目的后期进行X
- 根据已经完成的类似项目进行估算
- 使用比较简单的分解技术，生成项目的成本和工作量的估算
- 使用一个或多个经验模型来进行软件成本和工作量的估算

## ■ 到目前为止，因为软件项目变化的要素太多，所以对软件的成本估算从来没有达到精确

## ■ 估算效果的好坏取决于估算使用的历史数据

# 1 分解估算技术

## ■ 软件规模估算

- 直接测量（代码行LOC）
- 间接测量（功能点FP）

## ■ 问题分解与过程分解（产品与过程）

### — 基于问题的估算

- 将软件分解成可独立估算的功能问题（功能、类、变更、过程）
- 估算每个功能的LOC或FP
- 合并得到项目的总体估算

### — 基于过程的估算

## ■ 调和不同的估算方法

# 代码行技术(LOC)

## ■ LOC: Lines of Code

- 从过去开发类似产品的经验和历史数据出发，估计出所开发软件的代码行数。

$$L = \frac{a + 4m + b}{6}$$

$L$  估计的代码行数

$a$  乐观值

$b$  悲观值

$m$  可能值

$$C = \mu \times L$$

$L$  估计的代码行数

$\mu$  每行代码的单位成本

$C$  总成本

$$PM = \frac{L}{v}$$

$L$  估计的代码行数

$v$  平均生产率( $LOC/pm$ )

$PM$  总的工作量( $pm$ )



# 代码行技术(LOC)

功能	LOC估算			
	乐观值	可能值	悲观值	估算值
用户接口及控制设备	<b>2000</b>	<b>2200</b>	<b>3000</b>	<b>2300</b>
二维几何分析	<b>3000</b>	<b>5700</b>	<b>6000</b>	<b>5300</b>
三维几何分析	<b>4600</b>	<b>6900</b>	<b>8600</b>	<b>6800</b>
数据库管理	<b>3000</b>	<b>3250</b>	<b>4100</b>	<b>3350</b>
图形显示	<b>4000</b>	<b>4900</b>	<b>6100</b>	<b>4950</b>
外部设备控制	<b>1800</b>	<b>2100</b>	<b>2400</b>	<b>2100</b>
设计分析模块	<b>7000</b>	<b>8600</b>	<b>9000</b>	<b>8400</b>
总代码行数(L)				<b>33200</b>
平均生产率(v)	<b>620LOC/人月</b>			
每行代码单位成本(u)	月平均工资 <b>8000</b> , $u=8000/v=13$ 元			
总成本(C)	<b><math>c=u*L</math></b>			<b>431600</b>
总工作量	<b><math>pm=L/v</math></b>			<b>54人月</b>

# 功能点技术FP

- 功能点(**Function Point, FP**), 以功能点为单位来估计软件规模, 关注五个方面的功能:
  - 外部输入(EI): 用户进行添加或修改数据的UI
  - 外部输出(EO): 软件为用户产生的输出UI
  - 外部查询(EQ): 软件可产生的独立查询
  - 内部逻辑文件(ILF): 软件修改或保存的逻辑记录集合(数据表或文件)
  - 外部接口(EIF): 与其它系统进行信息交换或共享的文件
- 问题分解关注的不是软件功能, 而是信息域的值
- 估算得到一个系统的总功能点数, 然后据此估算工作量和成本。

## Step 1: 计算未调整功能点

信息域值	乐观值	可能值	悲观值	估算值 (P)	加权因子 (r)			FP值
					简单	中等	复杂	
外部输入	20	24	30	24	3	4	6	97
外部输出	12	15	22	16	4	5	7	78
外部查询	16	22	28	22	3	4	6	88
内部逻辑文件	4	4	5	4	7	10	15	42
外部接口文件	2	2	3	2	5	7	10	15
未调整功能点总数								320

$$P = \frac{a + 4m + b}{6}$$

*P* 估算值  
*a* 乐观值  
*b* 悲观值  
*m* 可能值

$$FP = r \times P$$

*r* 加权因子  
 FP 总成本

## Step 2: 估计调整因子

- 由于软件规模会受到诸如性能、数据处理、可用性、可维护性等多种技术因素的影响，需要对计算得到的**FP**进行适当调整。

技术因素	影响值	技术因素	影响值
备份与恢复	4	主文件联机更新	3
数据通信	2	信息域值复杂度	5
分布式处理	0	内部处理复杂度	5
关键性能	4	设计可复用的代码	4
现有操作环境	3	设计中的转换与安装	3
联机数据输入	4	多次安装	5
多屏幕输入切换	5	易于变更的应用设计	5

$$K = 0.65 + 0.01 \times \sum_{i=1}^{14} (F_i) = 1.17$$

## Step 3: 计算调整功能点和总成本

$$FP_{estimated} = 320 \times 1.17 = 375$$

- 平均生产率(**v**): **6.5FP/pm**
- 月平均工资: **8000元**
- 每个**FP**的成本(**u**): **1230元**
- 总成本(**C**):  **$C = FP * u = 375 * 1230 = 461250$ 元**
- 总工作量(**PM**):  **$PM = FP / v = 375 / 6.5 = 58$ 人月**

# 基于过程的估算

- 将过程分解为一组较小的任务，并估算完成每个任务所需的工作量
  - 沟通
  - 策划
    - 可行性分析
    - 计划
  - 建模
    - 分析
    - 设计
  - 构建
    - 编码
    - 测试
  - 部署

# 基于过程估算实例

活动	客户沟通	策划	风险分析	建模		构建		客户评估	合计
				分析	设计	编码	测试		
用户接口及控制设备				0.5	2.5	0.4	5		8.4
二维几何分析				0.75	4	0.6	25		7.35
三维几何分析				0.5	4	1	3		8.5
数据库管理				0.5	3	1	1.5		6
计算机图形显示设备				0.5	3	0.75	1.5		5.75
外部设备控制功能				0.25	2	0.5	1.5		4.25
设计分析模块				0.5	2	0.5	2		5
合计	0.25	0.25	0.25	3.5	20.5	4.5	16.5		46
%工作量	1%	1%	1%	8%	45%	10%	36%		

月平均工资：**8000元**

总成本(c): **C=8000\*46=36800元**

## 2 经验估算模型

- **COCOMO II** 模型
- 软件方程式



## 小结：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。

# 主要内容

## **3.1 软件项目管理概述**

## **3.2 项目可行性分析与估算**

## **3.3 项目进度安排**

### **3.3.1 项目进度计划**

### **3.3.2 项目进度跟踪**

## **3.4 项目风险管理**

## 小练习

- 你手头有五项任务需要去做，请为其编制执行计划

任务	预计时间	预计成本	预计收益	所需资源
<b>P1</b>	<b>1周</b>	<b>¥100</b>	<b>¥140</b>	<b>15人</b>
<b>P2</b>	<b>2周</b>	<b>¥100</b>	<b>¥150</b>	<b>15人</b>
<b>P3</b>	<b>2周</b>	<b>¥150</b>	<b>¥200</b>	<b>15人</b>
<b>P4</b>	<b>1周</b>	<b>¥200</b>	<b>¥300</b>	<b>15人</b>
<b>P5</b>	<b>2周</b>	<b>¥150</b>	<b>¥250</b>	<b>15人</b>
<b>P6</b>	<b>1周</b>	<b>¥50</b>	<b>¥80</b>	<b>15人</b>

### — 约束条件：

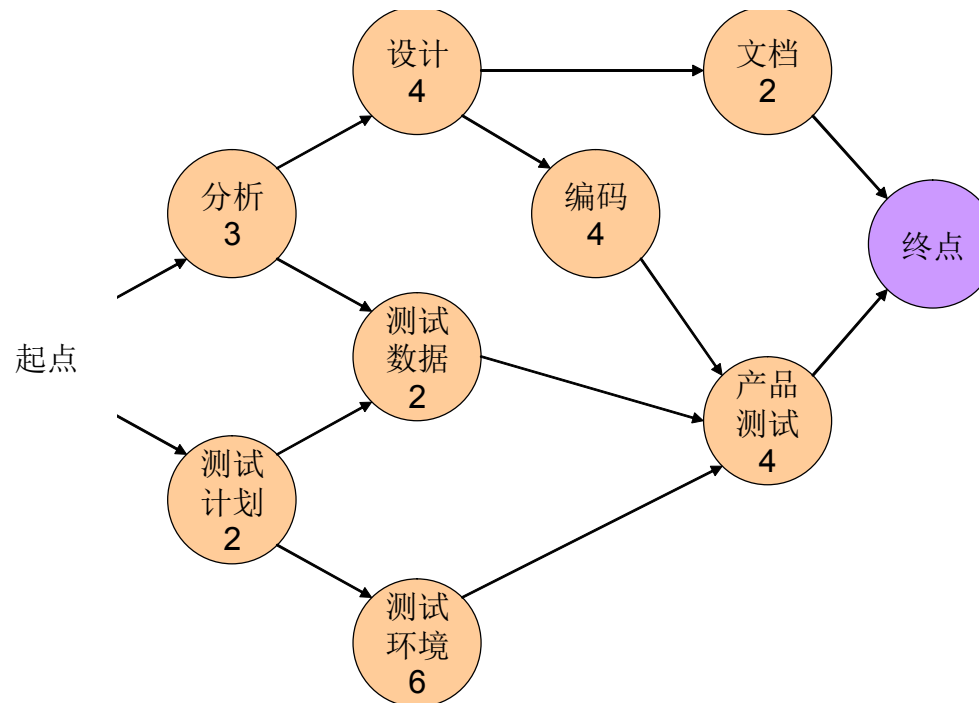
- 你共有四周时间，每周的预算是¥150；
- 你共有30人的可用资源。
- 前一周的收益可以作为下一周的成本而投入进去。

# 1 软件项目进度计划

- 目标：
  - 使软件项目在截止日期之前成功的完成并提交给客户；
- 活动：
  - Step 1: 定义任务网络
  - Step 2: 为任务(task)安排特定的日期(scheduling)
  - Step 3: 将资源(resources)分配给任务
  - Step 4: 明确产出结果(outcomes)
  - Step 5: 明确里程碑(milestones)

## Step 1: 定义任务网络

- 任务不是独立存在的，各任务在顺序上存在相互依赖关系。
- 项目管理里通常采用“**网络图(Network Diagram)**”的形式对此进行描述。

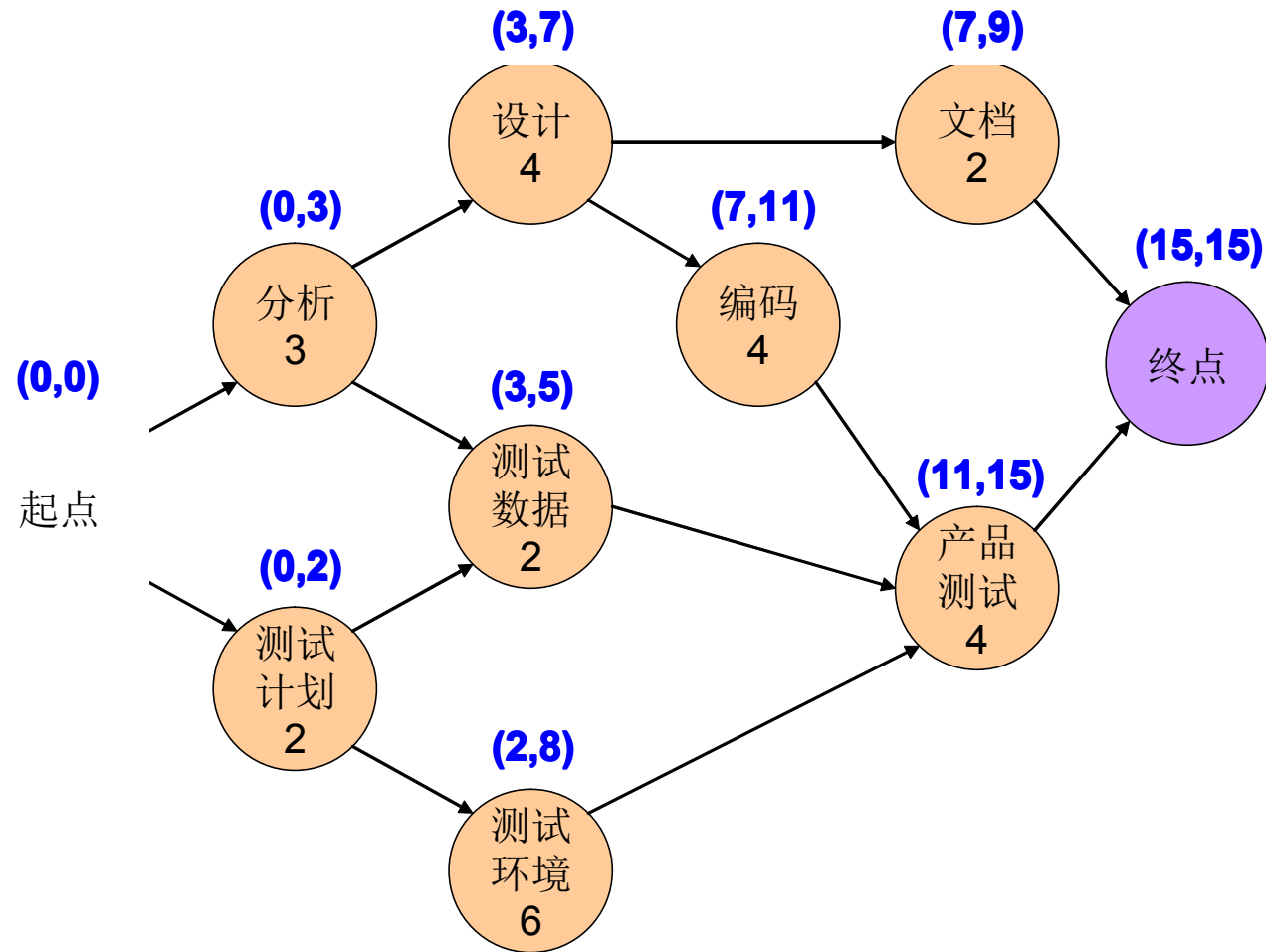


## Step 2: 任务进度安排

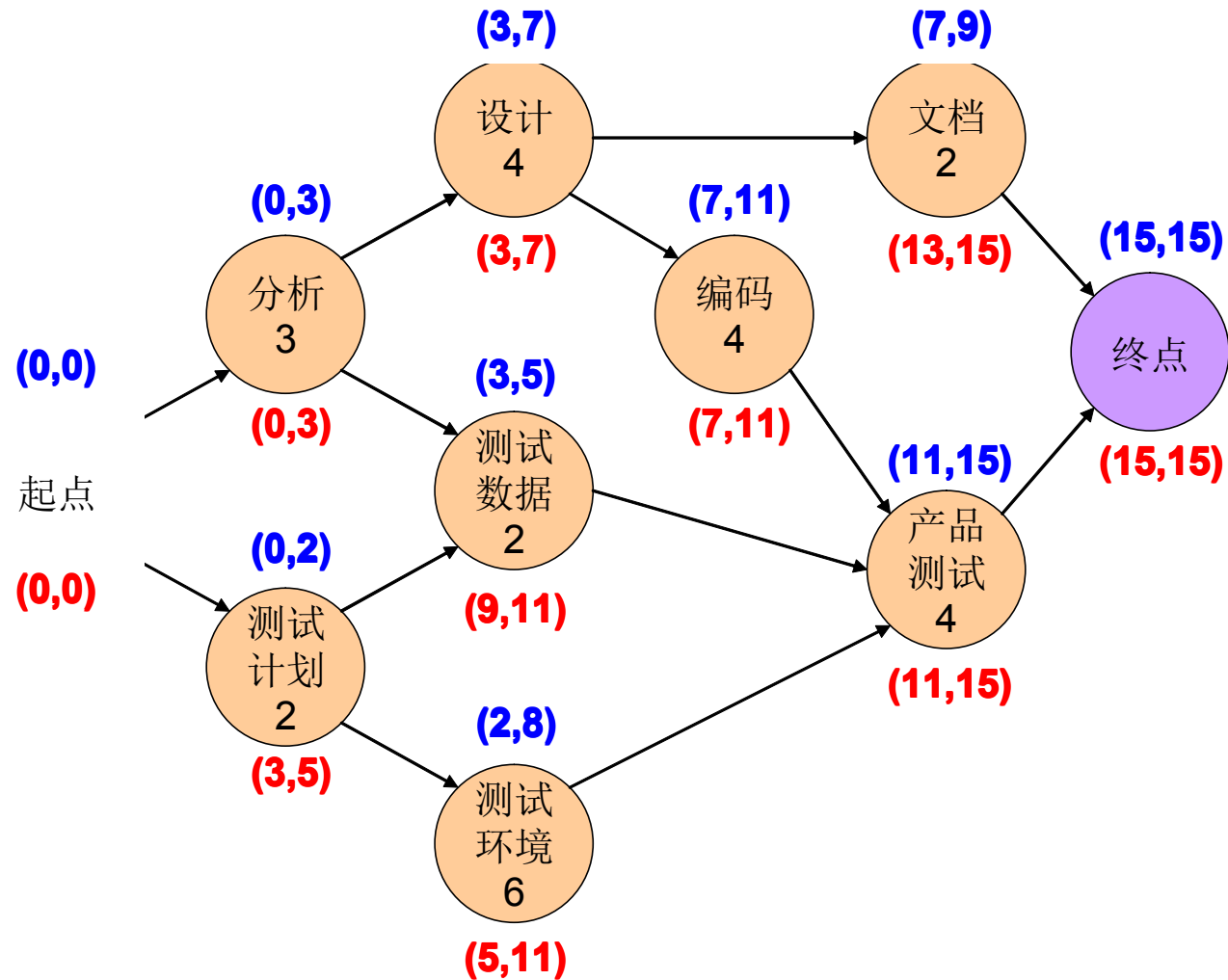
- 在绘制出任务网络图之后，下一步需要为每一个任务分配具体的执行时间。
- 两种具体的技术：
  - 程序评估及评审技术(PERT)
  - 关键路径方法(CPM)
- 步骤：
  - Step 2.1: 标出任务的最早开始(ES)与结束时间(EF);
  - Step 2.2: 标出任务的最迟开始(LS)与结束时间(LF);
  - Step 2.3: 计算关键路径(Critical Path);
  - Step 2.4: 确定任务的开始/结束时间。
  - Step 2.5: 绘制最终的任务进度安排(甘特图, Gantt Diagram)

最早开始	持续时间	最早结束
任务名称		
最晚开始	松弛量	最晚结束

## Step 2.1 标出最早开始/结束日期



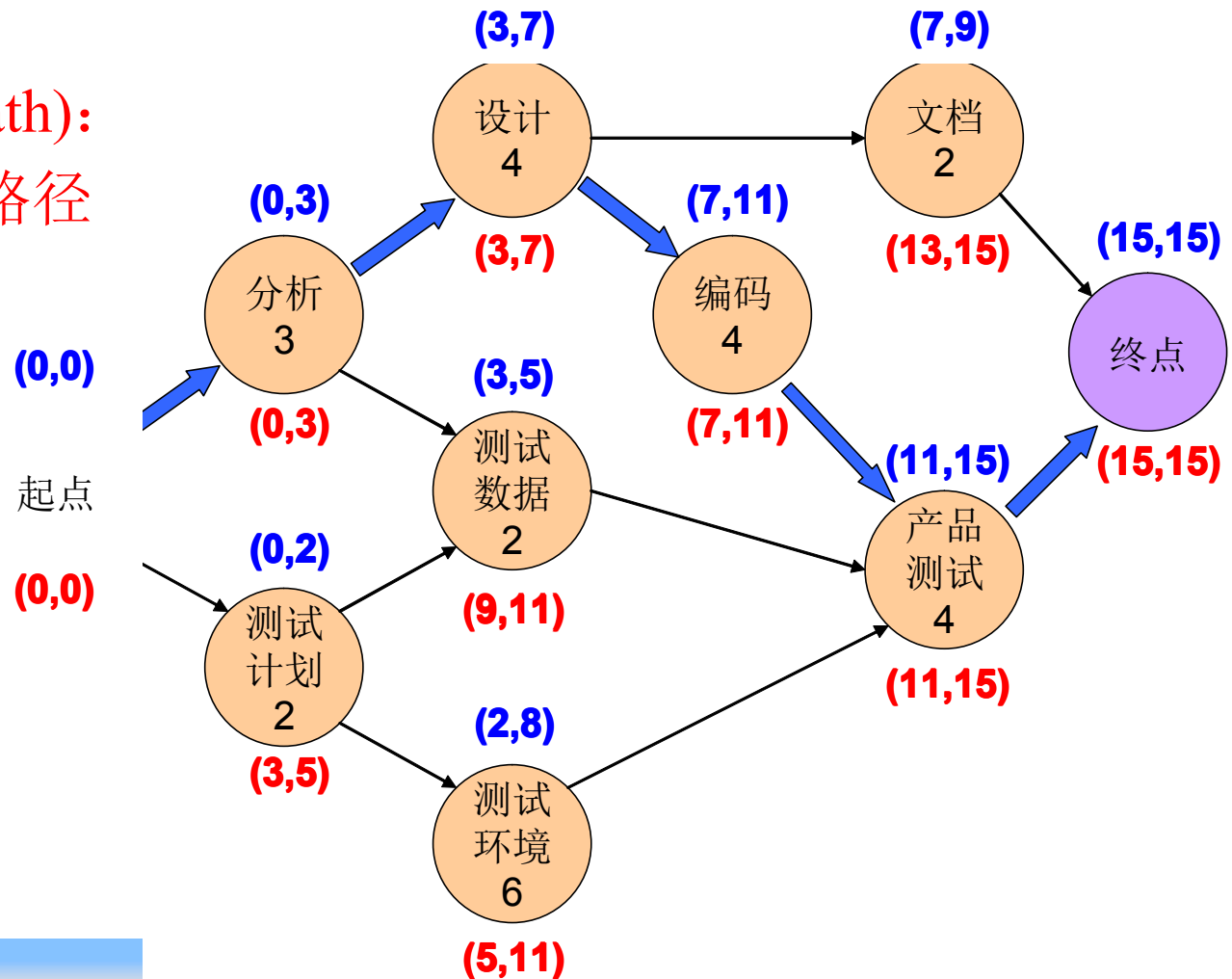
## Step 2.2 标出最晚开始/结束日期





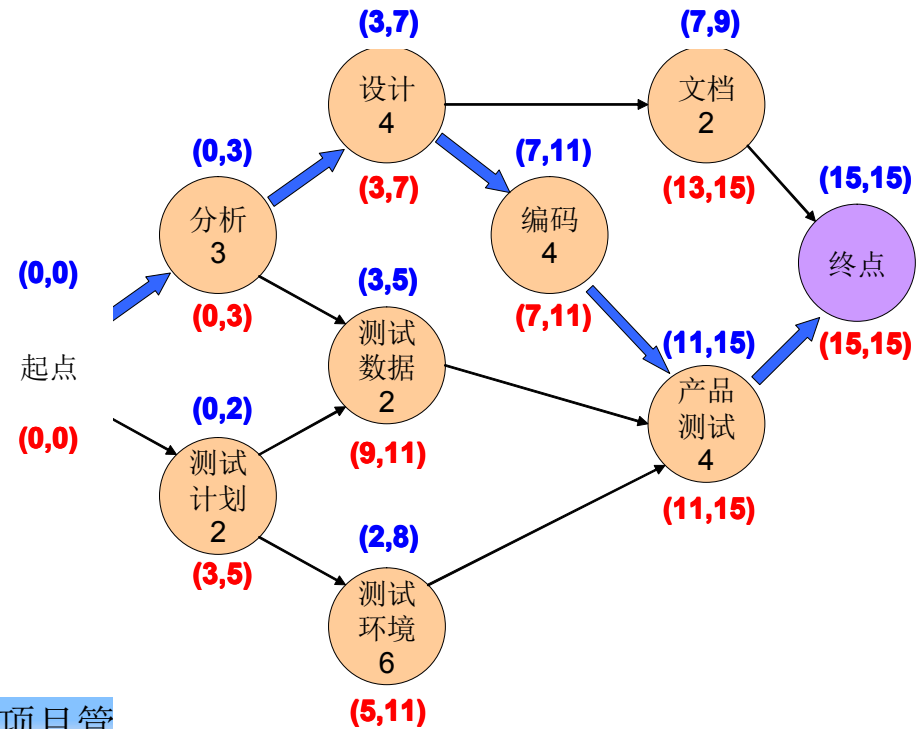
## Step 2.3: 计算关键路径

关键路径(Critical Path):  
持续时间最长的路径



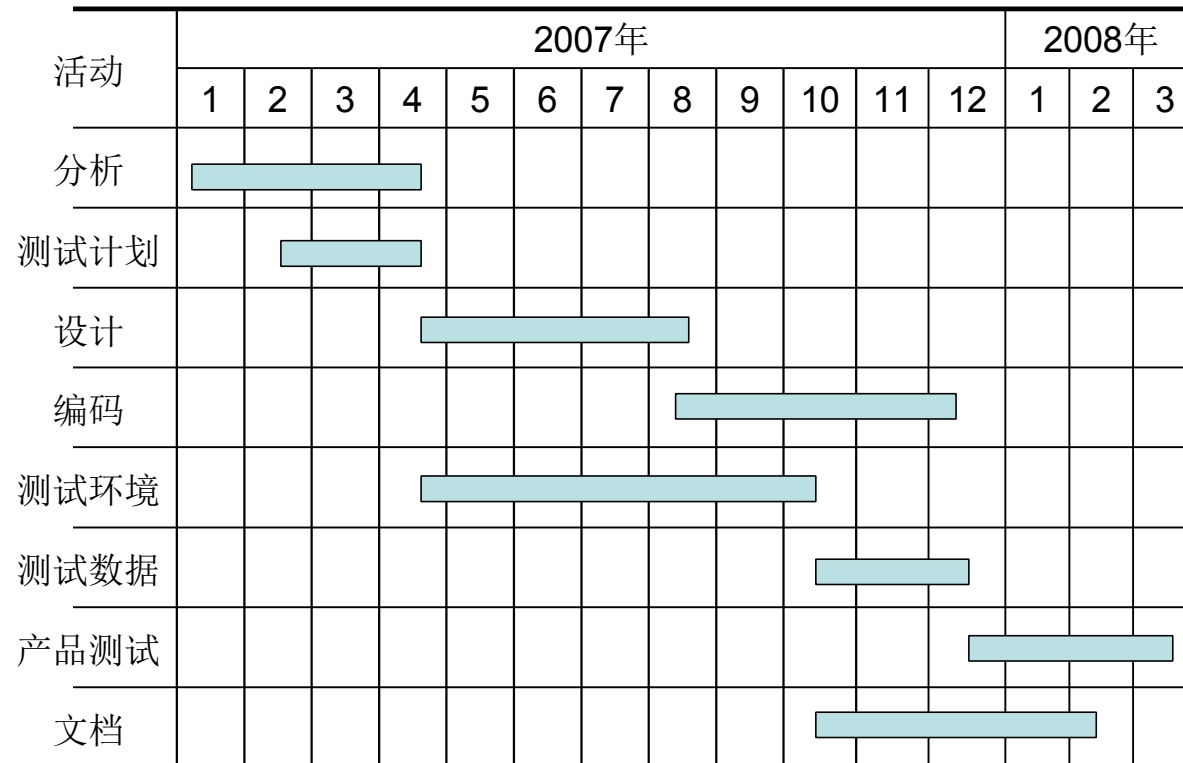
## Step 2.4: 确定最终开始/结束日期

- **[关键定义]松弛量(Slack):** 在不影响最终交付日期的前提下, 一个任务最多可被推迟的总天数。
- 关键路径上的任务的松弛量=0;
- 先确定关键路径上各任务的日期;
- 然后确定其他任务的日期;
- 通过缩短关键路径上的任务的持续时间, 可极大优化整个项目的持续时间。

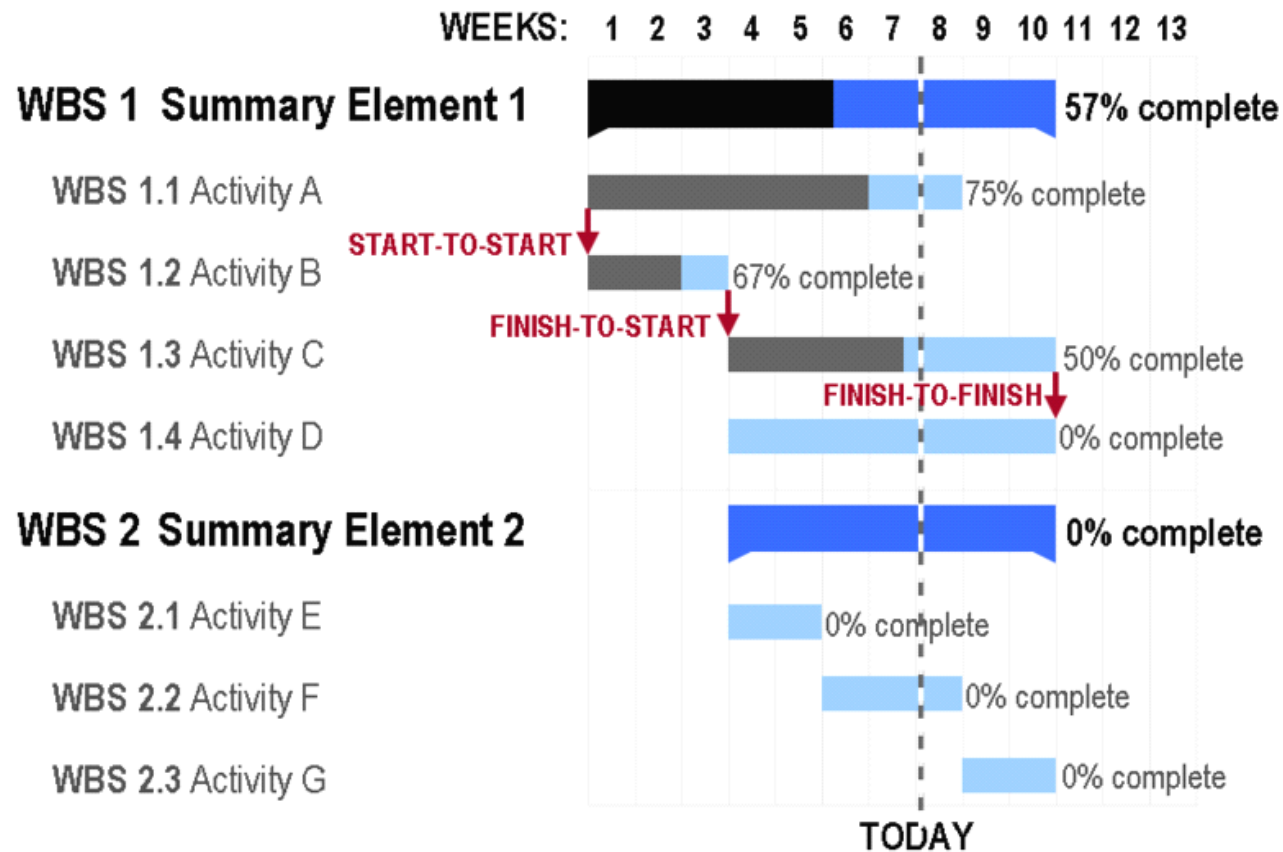


## Step 2.5: 绘制任务进度安排图

- 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)





## Step 3~5 资源、产出与里程碑

**Step 1:** 定义任务网络

**Step 2:** 为任务(task)安排特定的日期(scheduling)

**Step 3:** 将资源(resources)分配给任务

资金

人员

设备

环境

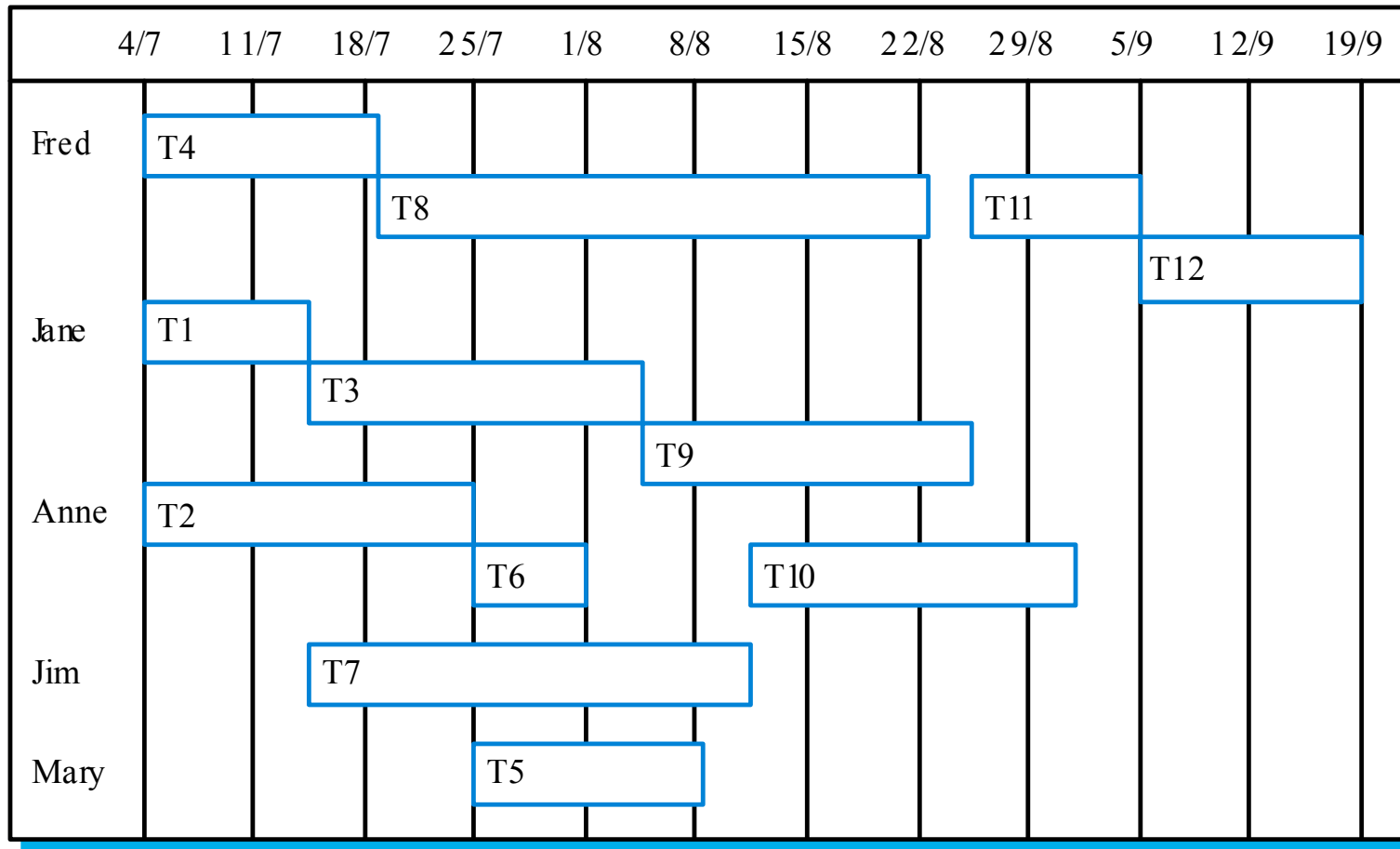
**Step 4:** 明确产出结果(outcomes)

每一项任务的产出结果是什么？对应于PBS中的哪一部分？

**Step 5:** 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

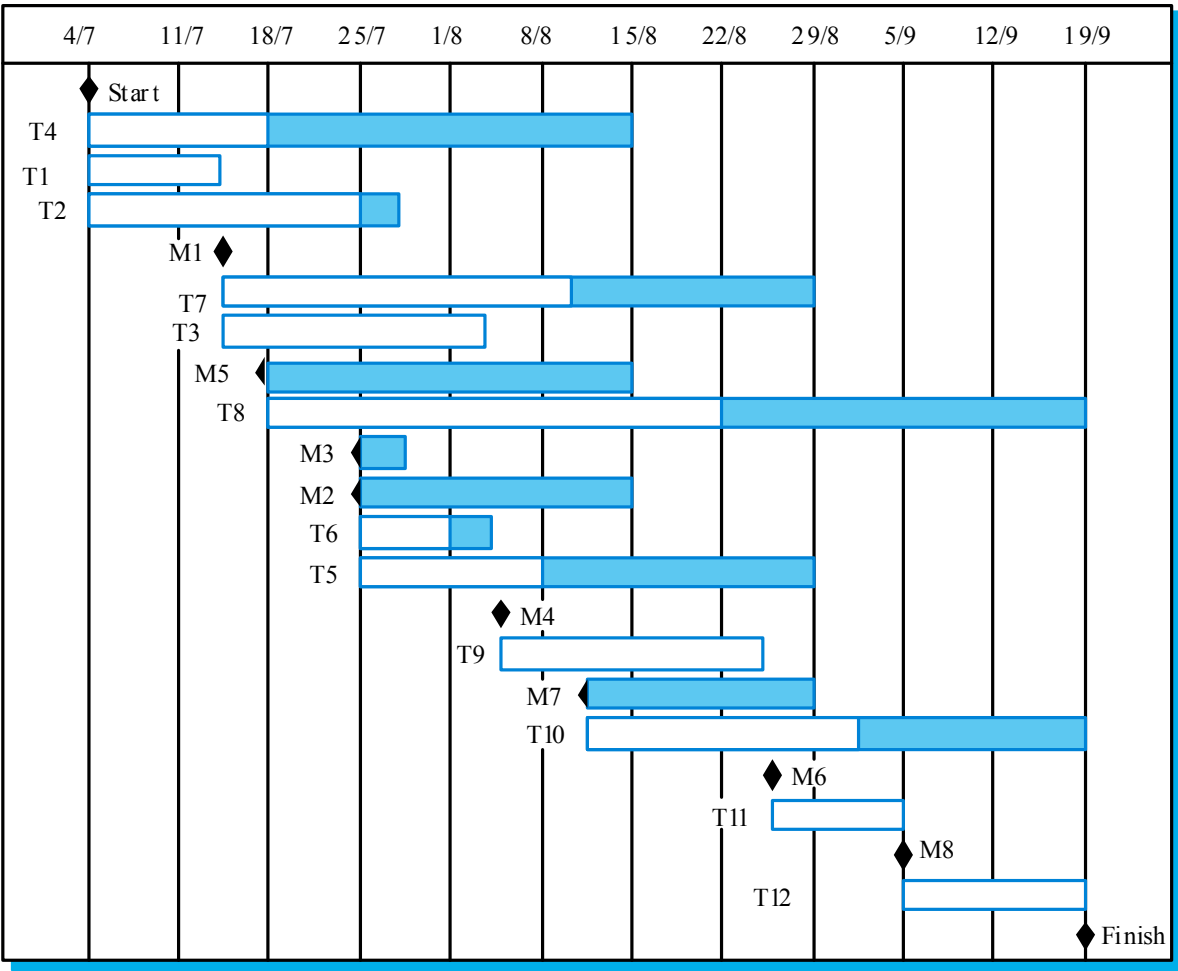


## 2 项目进度跟踪

- 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；
  - 比较个项目的实际开始/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



## 主要内容

**3.1** 软件项目管理概述

**3.2** 项目可行性分析与估算

**3.3** 项目进度安排

**3.4** 项目风险管理

# 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？

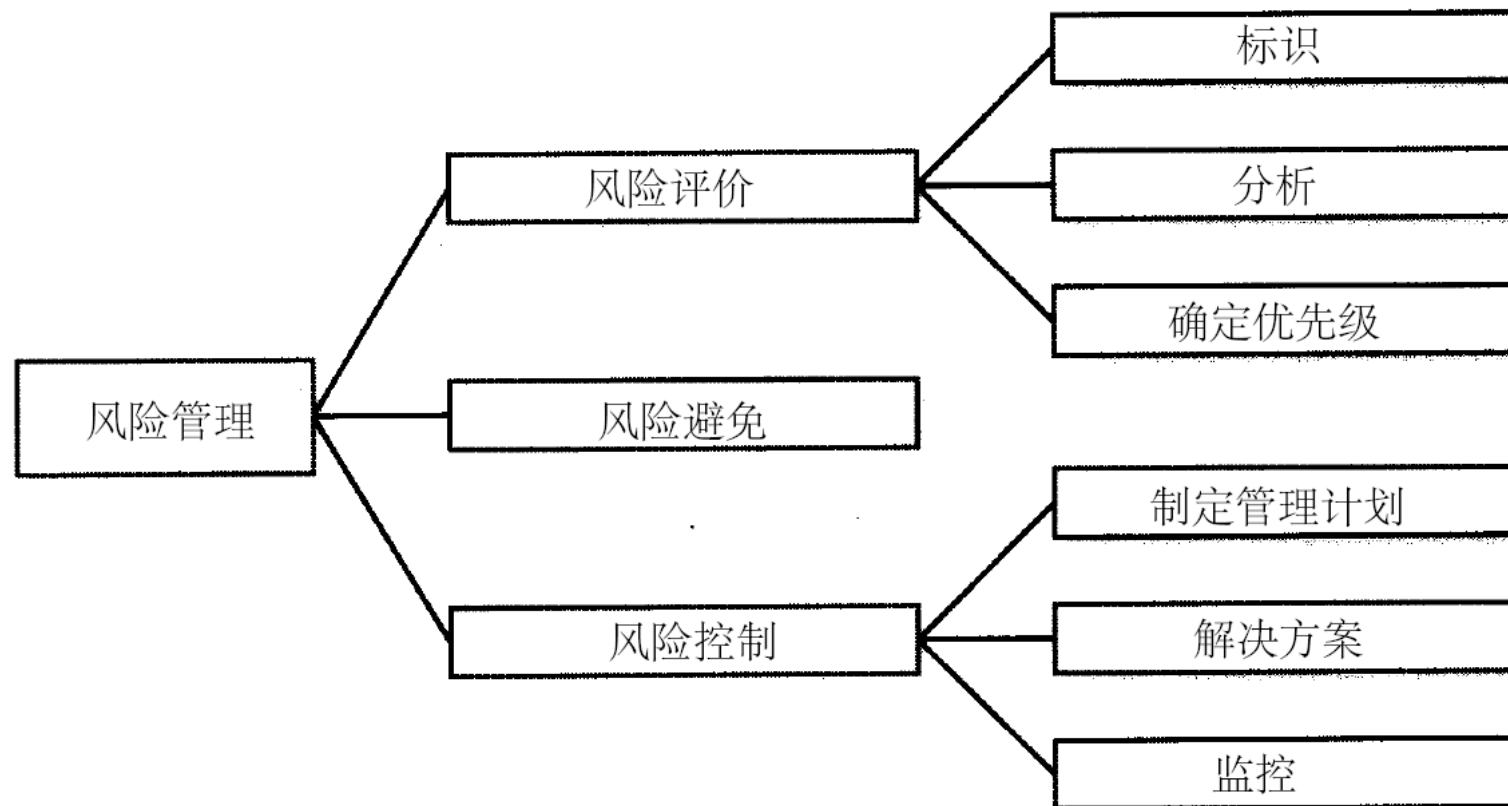
# 风险预测与分析

- **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





结束

**2011年4月6日**