



## 第二部分 软件项目开发过程与管理

- 2.1 软件项目开发过程
- 2.2 软件项目管理



## 2.2 软件项目开发管理

- 软件项目管理概念及特征
- 软件项目估算
- 项目进度安排
- 项目风险管理



## 2.2 软件项目开发管理

- 软件项目管理概念及特征
- 软件项目估算
- 项目进度安排
- 项目风险管理



# 软件项目管理基本概念

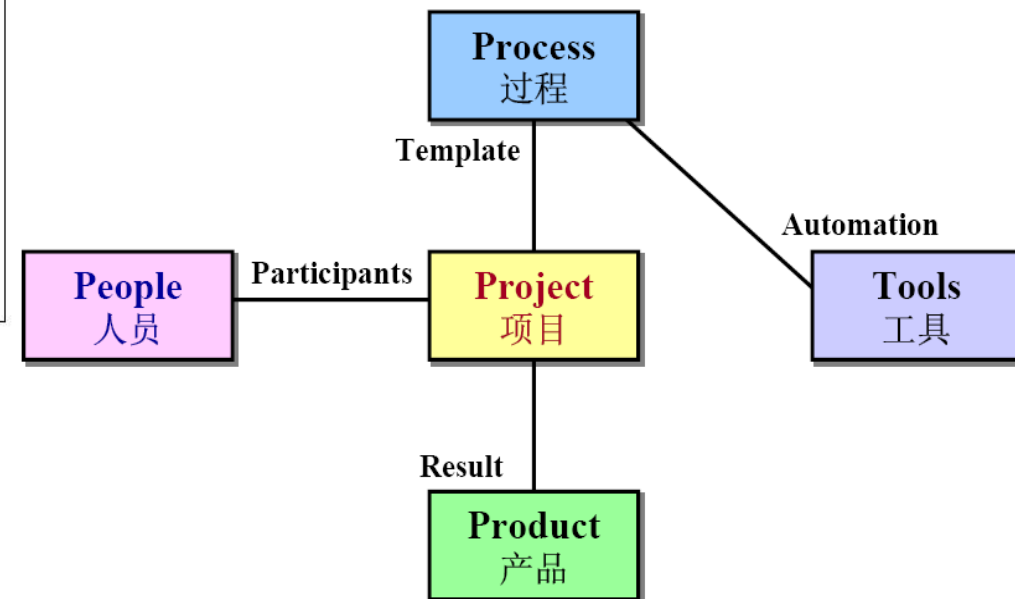
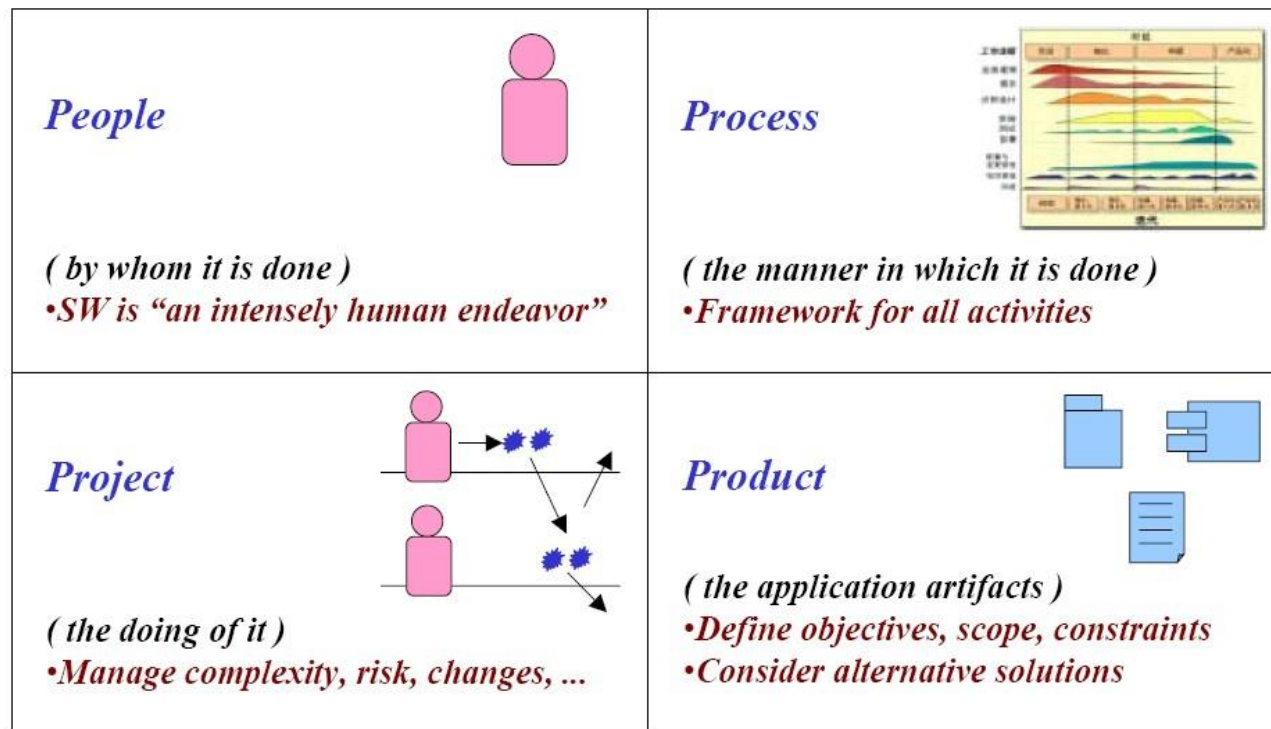
- **项目(Project):** 精心定义的一组活动, 使用受约束的资源(资金、人、原料、能源、空间等)来满足预定义的目标。
- **项目管理(Project Management, PM):** 有效的组织与管理各类资源(例如人), 以使项目能够在预定的范围、质量、时间和成本等约束条件下顺利交付(**deliver**)。
  - 挑战1: 在各类约束条件下交付项目;
  - 挑战2: 通过优化资源的分配与集成来满足预先定义的目标;



# 软件项目的特征

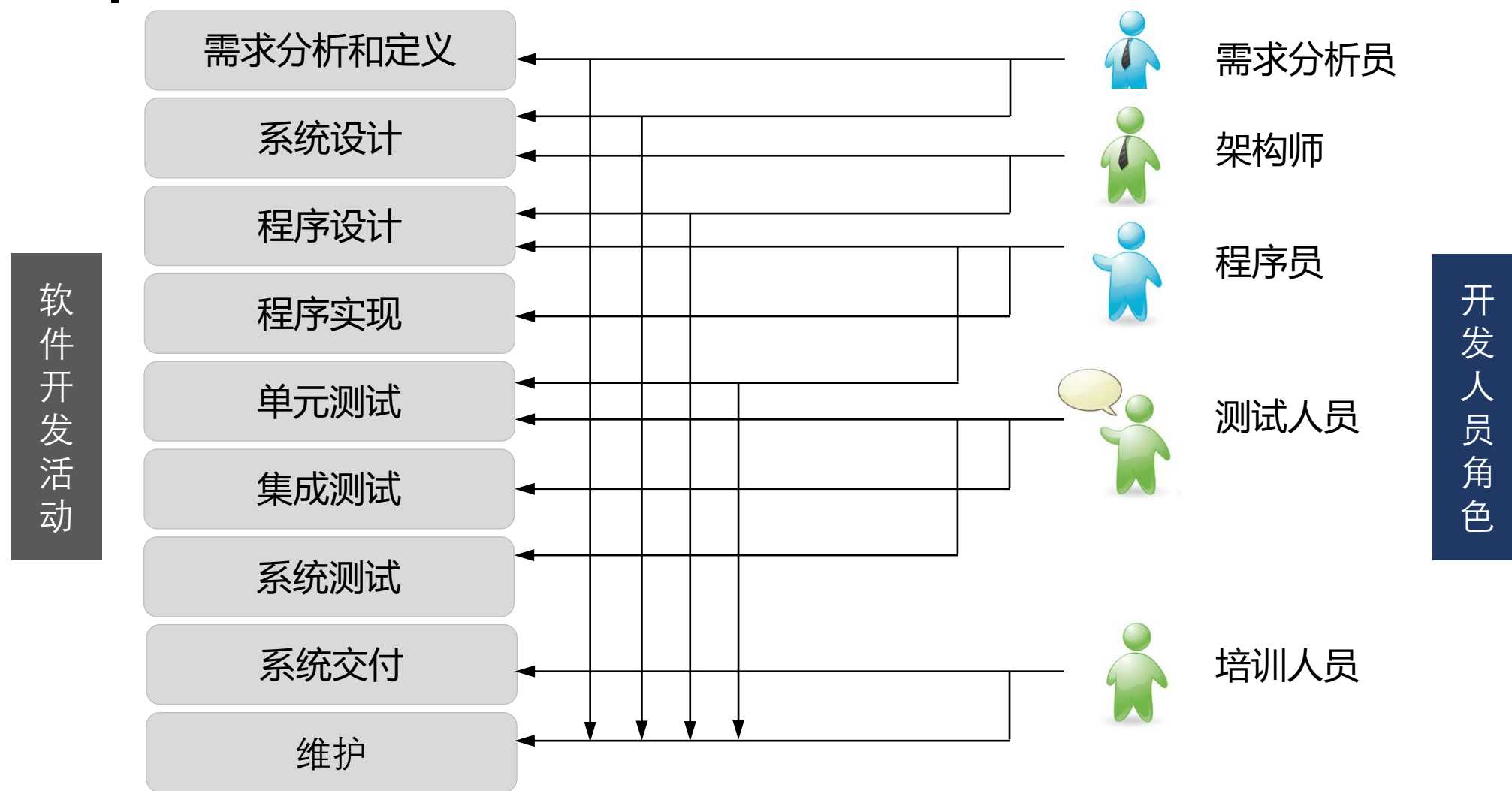
- **软件产品的不可见性：**开发过程和产品都是看不见摸不着的，导致软件项目特别复杂和抽象；
- **项目的高度不确定性：**项目的估算与计划非常困难，有很多难以预见的问题，造成预定计划与实际情况存在较大偏差；
- **软件过程的多变化性：**迭代、增量开发、动态变化、不确定、不稳定；
- **软件人员的高技能及其高流动性：**智力密集型活动、对人的要求高、核心技术人才流动性高。

# 软件项目管理的“4P”



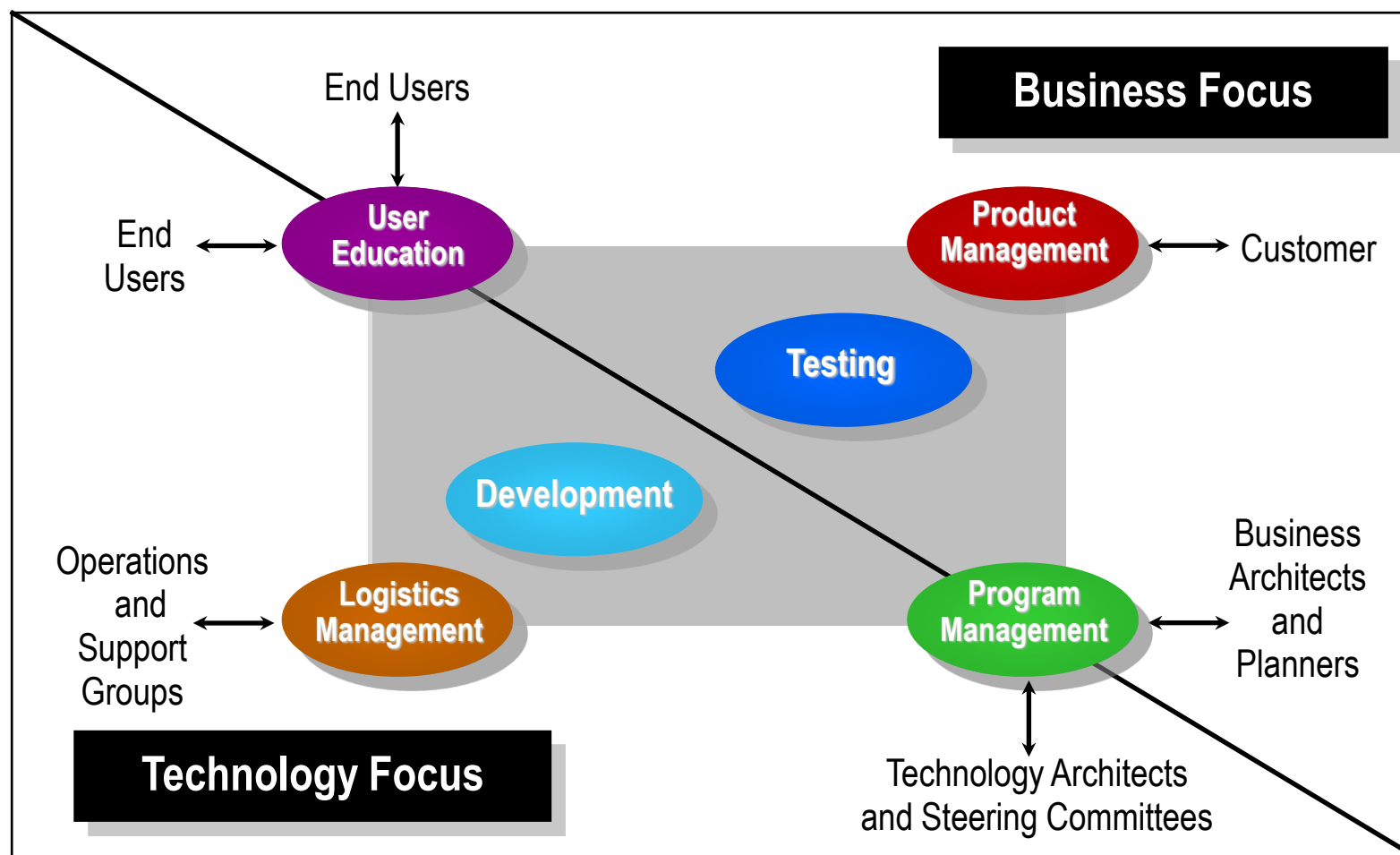


# People——软件人员





# 微软开发团队的角色分工







# 人员的选择

## 案例描述

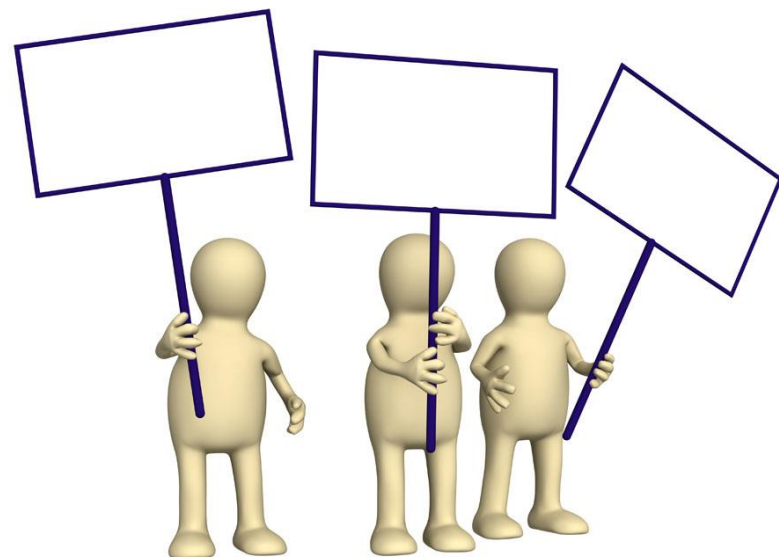
假设你是一个报警系统开发公司的项目经理，该公司希望进入帮助独立生活的老年人和残疾人的技术服务市场。你需要组建一个6人的开发团队，他们可以基于公司现有的报警处理技术开发新产品。显然，你的首要任务是从公司内部或者外部选择合适的团队成员。



**你如何选择团队成员？这样选择的理由是什么？**



# 人员的选择



应用领域经验

平台经验

编程语言经验

教育背景

解决问题能力

沟通能力

适应性

工作态度

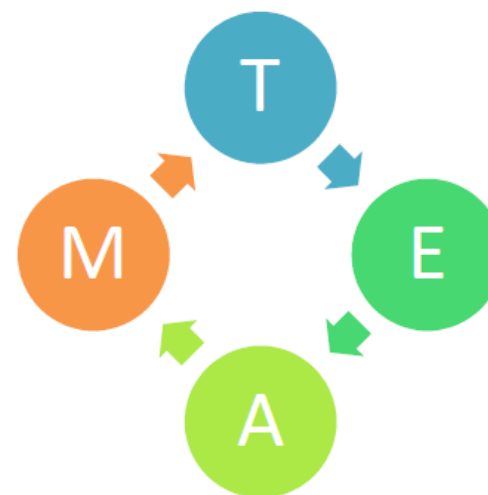
个性



# 团队的概念

**团队：**由少量的人组织，具有互补的技能，对一个共同目的、绩效目标及方法做出承诺并彼此负责。

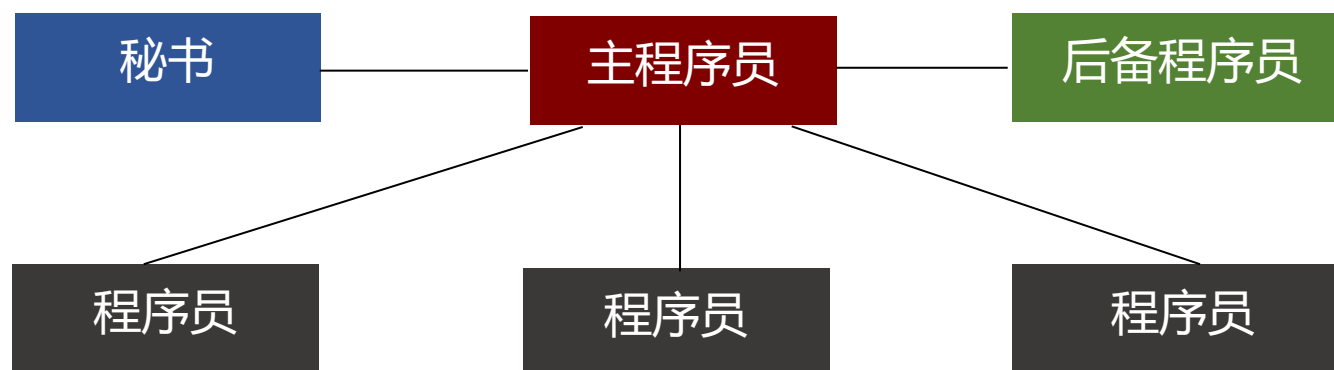
<b>T</b>	together	结合	众
<b>E</b>	everyone	大家	志
<b>A</b>	achieve	完成	成
<b>M</b>	Mission	任务	城





# 软件开发团队的组织方式

**主程序员式组织结构：**以主程序员为核心，主程序员既是项目管理者也是技术负责人，团队其他人员的职能进行专业化分工。

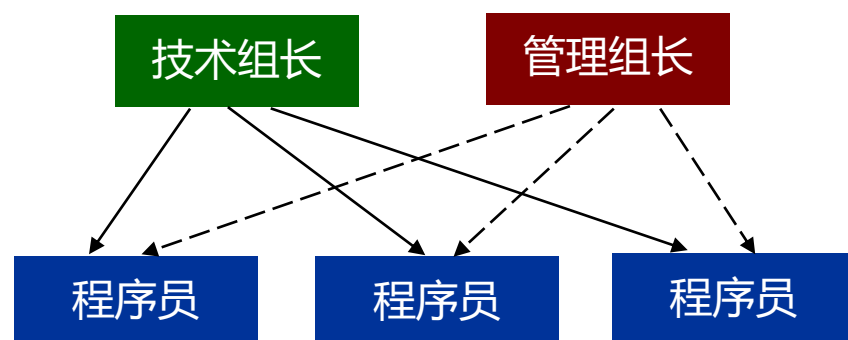


- **优点：**成员之间采取简单的交流沟通模式
- **缺点：**很难找到技术和管理才能兼备的主程序员

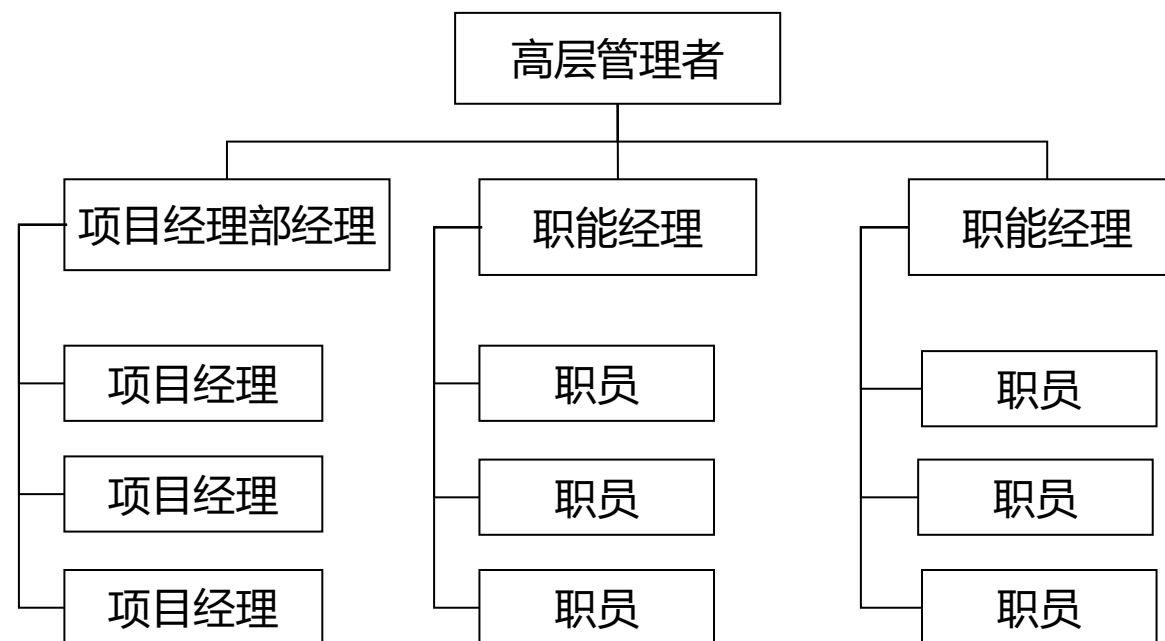


# 软件开发团队组织

**矩阵式组织结构：**将技术与管理工作进行分离，技术负责人负责技术决策，管理负责人负责非技术性事务的管理决策和绩效评价。



在这种组织中，明确划分技术负责人和管理负责人的权限是十分重要的。





# 微软开发团队组织

## 微软开发团队的特点：

- 小型的、多元化的项目组织
- 相互依赖的角色与共同分享的职责
- 具备专深的技术水平和业务技能
- 具有强烈的产品意识，关注最终发布的软件产品
- 清晰的目标和远景
- 人人参与设计
- 项目组成员在同一地点办公
- 对于规模较大的项目，采取类似小型项目组的运作模式

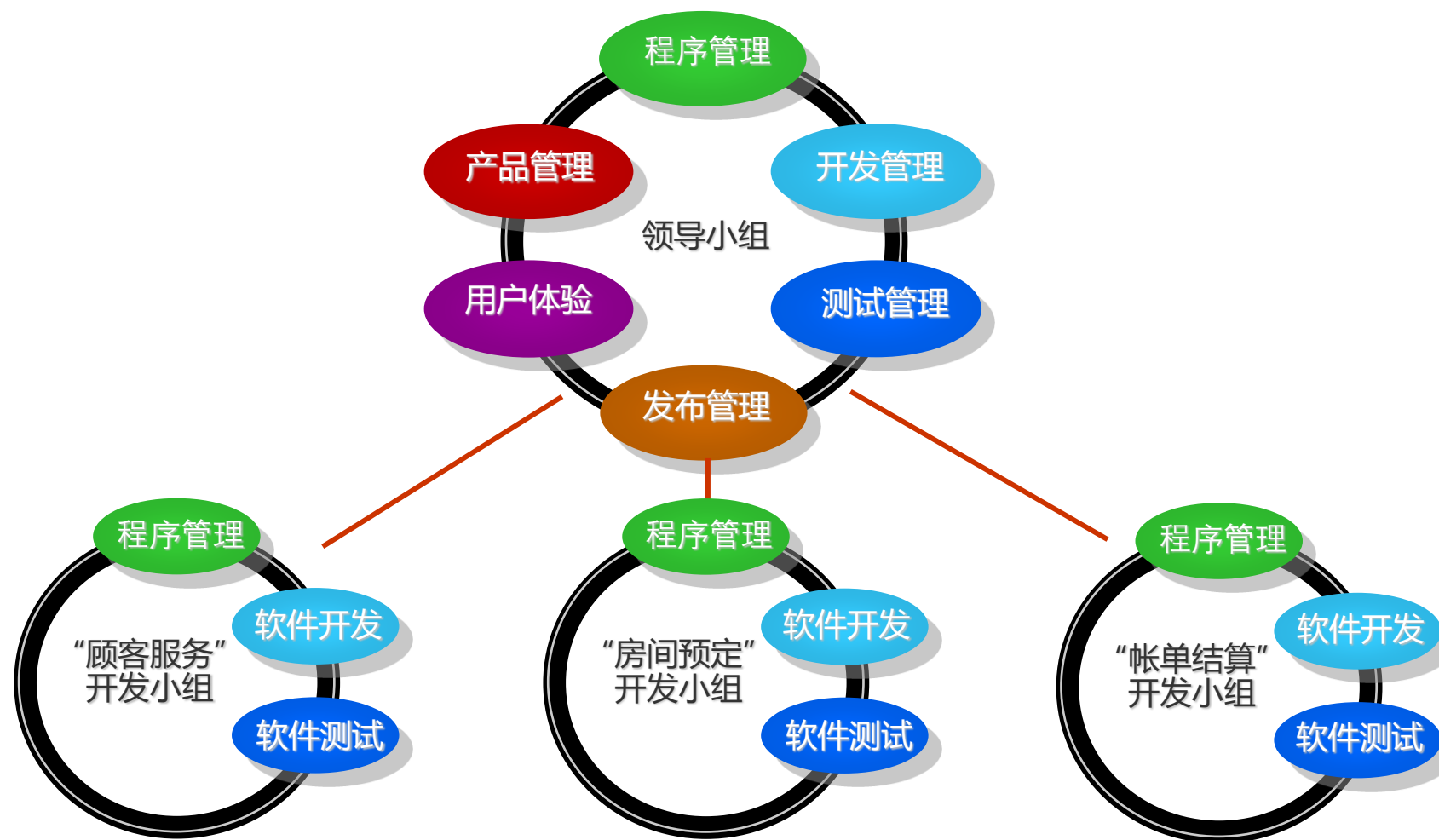


**Microsoft®**



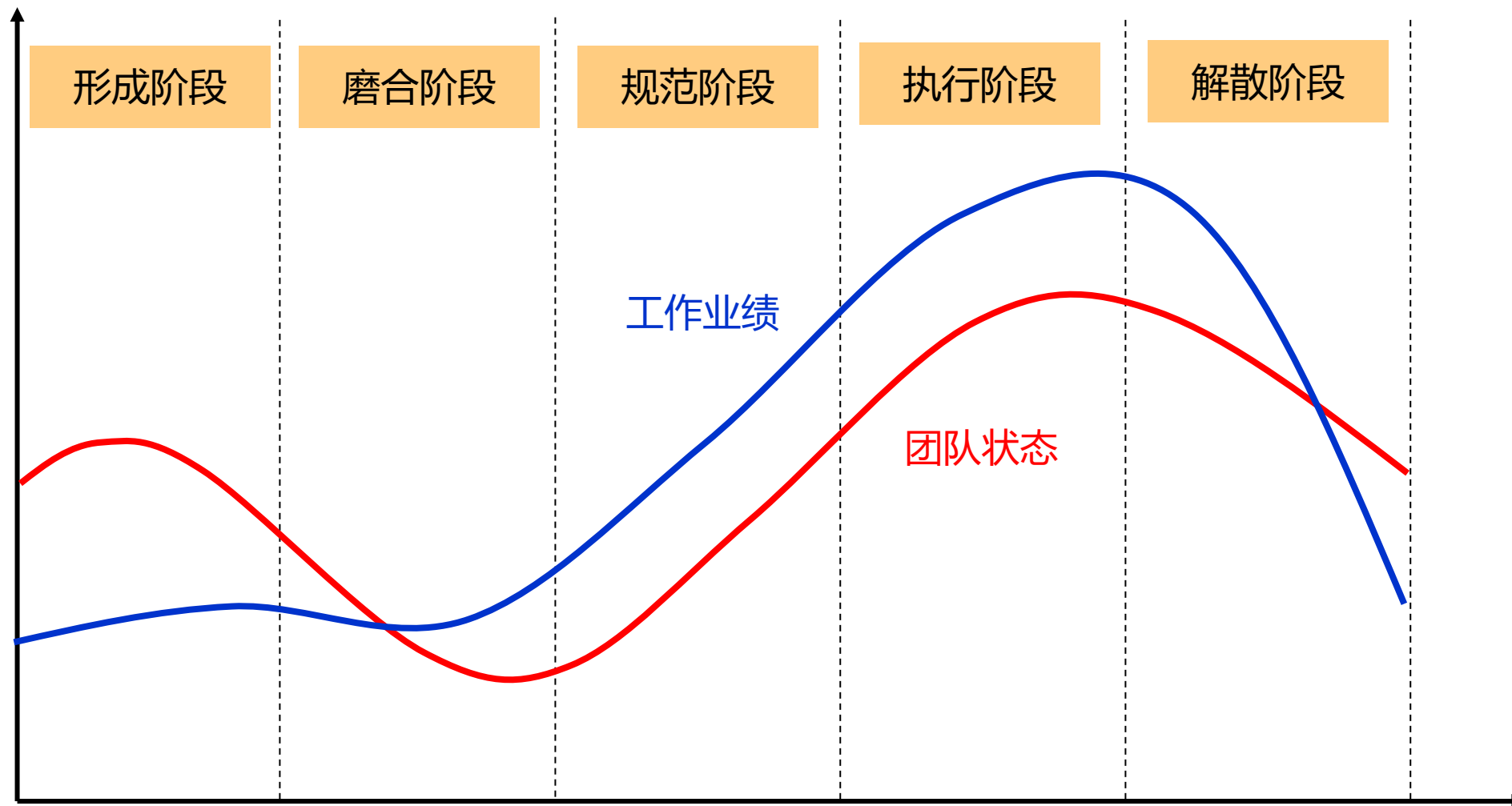


# 微软开发团队组织







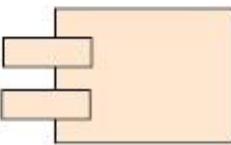
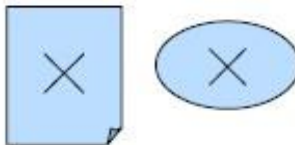

# 团队发展生命周期







# 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>				



# 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，任务对应产出物；**
  - [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。





# 过程分解的情况

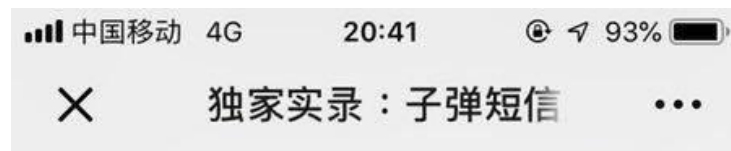
举例一：1个农夫可在**10**天内采摘完一块草莓地，那么同样的草莓地是否可以用**10**个农夫可在**1**天内采摘完？

举例二：1头大象需要孕育**22**个月才能生下1头小象，那么增加大象的数量是否可以加快这个过程呢？



# 过程分解的情况

举例三：假设开发某个模块需要**2**人月的工作量，那么是否可以认为**2**个程序员在**1**个月内就可以完成这个模块的开发？



曾在锤子待过的张敬，对罗永浩的习惯再熟悉不过，在他的印象里，罗永浩几乎每天都只睡三四个小时，凌晨三四点钟拉产品经理通宵开会，对产品、对需求都是常态。

“他太渴望成功了”，在张敬看来，这么多年，锤子科技一直没成功，他一直急于想要证明自己。

罗永浩的“急”，需要马上看到成果。“你一个人要干三天，我给你三个人，你给我干一天。三个人不够，我给你加十个人、加二十个人。”财经采访的几位快如科技前员工都表示，公司内部没有人敢反驳罗永浩的强势，他提出的问题，永远排在优先处理项。

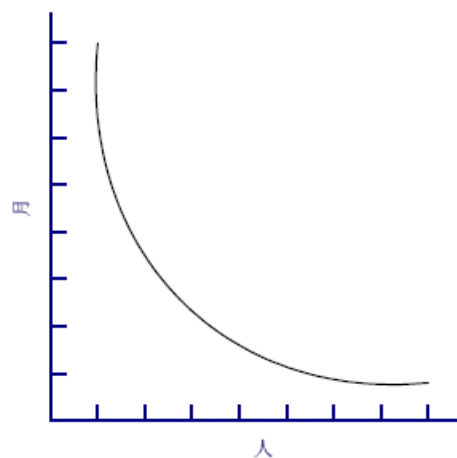
尽管团队一直加班加点修改问题，子弹短信还是无法维持巅峰时期的日活。



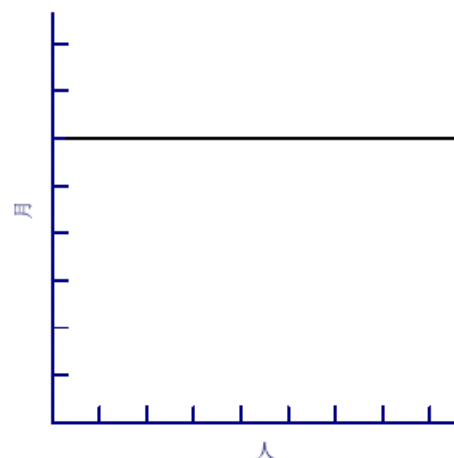
# 过程分解与沟通

人员数量与项目时间的关系：

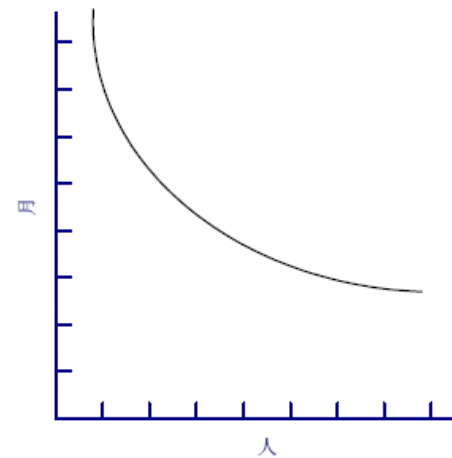
- 有些任务可以共同分担，有些任务则不行
- 沟通花费大量的时间



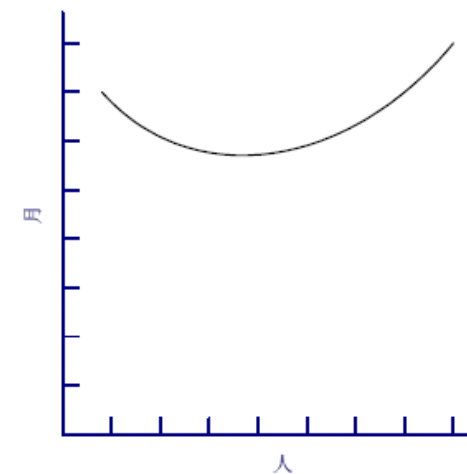
完全可分解的任务



无法分解的任务



需沟通的可分解任务



关系错综复杂的任务



# Project——项目

## ■ W<sup>5</sup>HH原则

### ■ 通过提出一系列问题，来导出对关键项目特性和项目计划的定义：

- 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？



# 项目的基本要素

一个或一组满足客户需求的工作产品，通常称作可交付物。

结果

工作

为获得结果而必须做的工作，被分为任务或活动的更小单元。

项目

安排工作单元的执行顺序，确定完成期限、开始时间和结束时间。

进度表

资源

完成工作需要的人员、资金、设备等，人员通常按照角色分工。





# 软件项目管理计划

软件项目管理计划是一个用来协调所有其他计划、以指导项目实施和控制的文件，它应该随着项目的进展和信息的补充进行定期完善。

引言	项目的目标、影响项目管理的各种约束条件
项目组织	开发团队的组织方式、人员构成与分工
风险分析	可能的风险以及发生的可能性、降低风险的策略
资源需求	项目所需的硬件与软件资源
工作分解	将项目分解成一系列的活动，指定项目里程碑和可交付的文档
项目进度	项目中各活动之间的依赖关系、完成每个里程碑预期需要的时间、在活动中的人员分配
监控和报告机制	需要提交的管理报告、提交时间以及项目监控机制





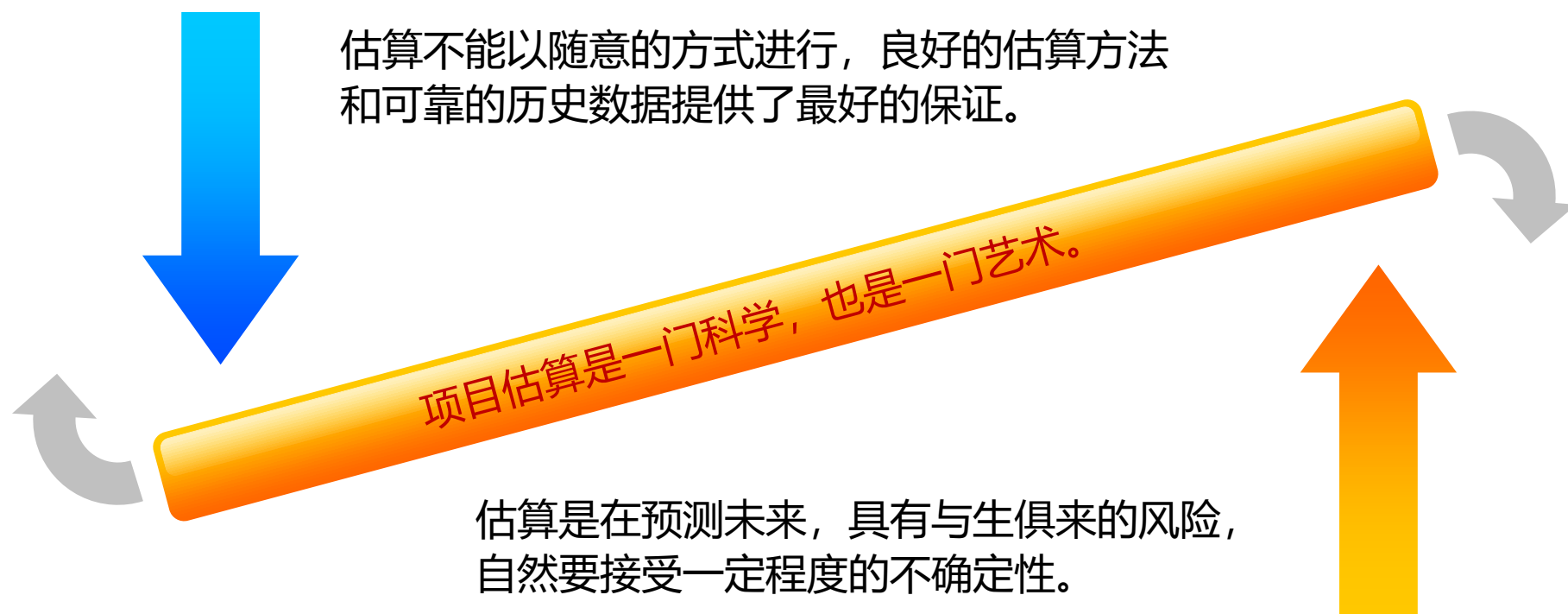
## 2.2 软件项目开发管理

- 软件项目管理概念及特征
- 软件项目估算
- 项目进度安排
- 项目风险管理



# 软件项目估算

项目估算是对项目的规模、工作量、时间和成本等进行预算和估计的过程。





# 项目估算的挑战

为什么软件开发的周期总是预估的2~3倍?





# 项目估算的挑战

没有很好地制定计划是软件项目常见的一个严重错误，很多技术人员宁愿从事技术工作，也不愿意花费时间制定计划。



软件项目估算的挑战是项目的复杂性和不确定性

- 软件规模越大，复杂性越高，不确定性就越大
- 需求的不确定性会对项目估算产生很大影响
- 没有可靠的历史数据使项目估算缺少参照物

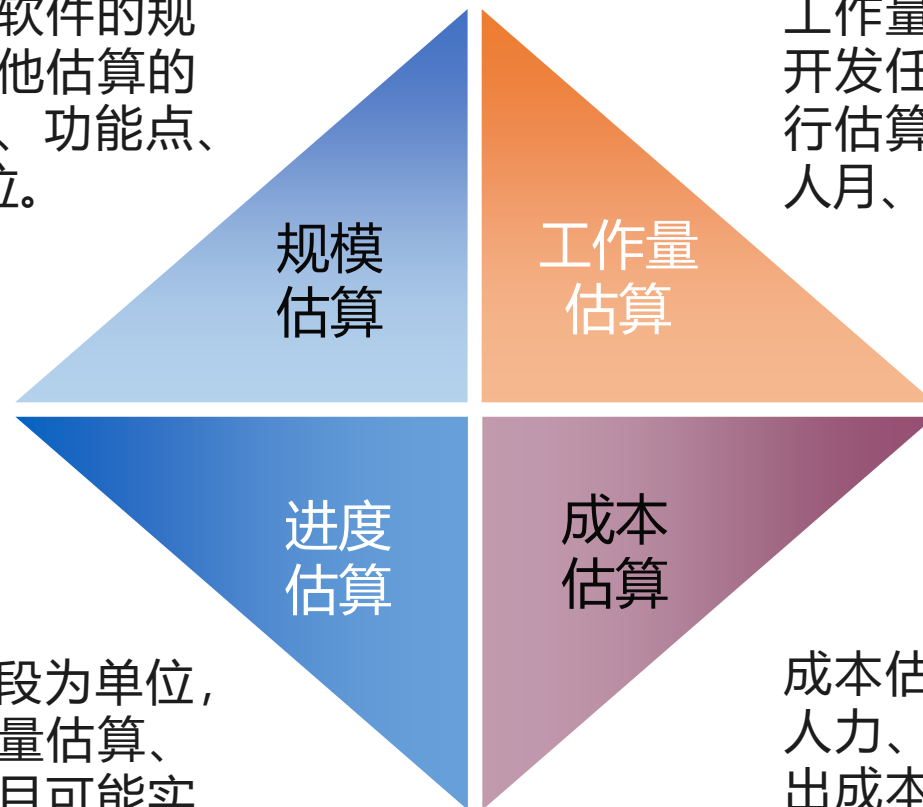
项目管理者不应该被估算所困扰，应该克服困难，做出一个相对的有价值的估算。



# 项目估算内容

规模估算是对所开发软件的规模进行估计，它是其他估算的基础，通常以代码行、功能点、故事点等作为衡量单位。

工作量估算是结合投入人力和开发任务所需要的工作时间进行估算，通常以人时、人天、人月、人年等作为衡量单位。



进度估算是以项目阶段为单位，通过任务分解、工作量估算、有效资源分配等项目可能实施的进度给出正确的评估。

成本估算是包括项目投入的人力、物质、有形和无形的支出成本进行估算，一般软件项目以人力成本为主要部分。



# 估算的复杂不确定性

- 项目复杂性对计划固有的不确定性有很大的影响。
- 项目规模是另一个影响估算精确度和功效的重要因素。
- 结构的不确定程度也影响着估算风险。
- 历史信息的有效性对估算的风险有很大的影响。

总结：估算的风险取决于资源、成本及进度的定量估算中存在的不确定性。

# 软件项目估算

## 分段估算:

- 从宏观估算开始, 在项目执行中对各阶段的估算进行细化
- 适用于最终产品不可知或不确定性很大的项目

Phase	Need 1	Specifications 2	Design 3	Produce 4	Deliver 5
1		Macro estimate			
2		Detailed estimate	Macro estimate		
3			Detailed estimate	Macro estimate	
4				Detailed estimate	Macro estimate
5					Detailed estimate





# 基本估算方法

## 专家判断:

- 通过借鉴历史信息，专家提供项目估算所需的信息，或根据以往类似项目的经验，给出相关参数的估算上限。







# 基本估算方法

**参数估算：**通过对大量的项目历史数据进行统计分析，使用项目特性参数建立经验估算模型，估算诸如成本、预算和持续时间等活动参数。





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

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

$$(2000 + 4 * 2200 + 3000) / 6 = 2300$$

$$C = \mu \times L$$

$$13 * 33200 = 431600$$

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

$$33200 / 620 = 54$$



# 功能点技术方法FP

**功能点方法**是依据软件信息域的基本特征和对软件复杂性的估计，估算出软件规模。这种方法适合于在软件开发初期进行估算，并以功能点为单位度量软件规模。

功能点估算使用**5**种信息域：

- **外部输入(External Input)**: 通过界面等的输入，插入和更新等操作都是典型外部输入
- **外部输出(External Output)**: 仅仅是输出，例如导出、报表、打印等
- **外部查询(External Queriers)**: 先输入数据，再根据输入数据计算得到输出，例如查询
- **内部逻辑文件(Internal Logical Files)**: 可以理解为业务对象，可能对应多个数据表
- **外部接口文件(External Interface Files)**: 其它应用提供的接口数据

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。



# 功能点技术方法FP

信息域加权因子:

信息域参数	加权因子			合计
	简单	中等	复杂	
外部输入	3	4	6	$\Sigma$
外部输出	4	5	7	$\Sigma$
外部查询	3	4	6	$\Sigma$
内部逻辑文件	7	10	15	$\Sigma$
外部逻辑文件	5	7	10	$\Sigma$
未调整功能点 UFC				$\Sigma$

系统复杂度调整值  $F_i$ : 取值 0.5

$F_1$	可靠的备份和恢复	$F_8$	在线升级
$F_2$	数据通信	$F_9$	复杂的界面
$F_3$	分布式处理	$F_{10}$	复杂的数据处理
$F_4$	性能	$F_{11}$	代码复用性
$F_5$	大量使用的配置	$F_{12}$	安装简易性
$F_6$	联机数据输入	$F_{13}$	多重站点
$F_7$	操作简单性	$F_{14}$	易于修改

功能点计算:  $FP = \underline{UFC} \times [0.65 + 0.01 \times \sum F_i]$



# 代码行和功能点比较

- 代码行：如果没有实际开发，无法准确的估算出来需要多少代码，依赖与具体的开发语言。
- 功能点：开始可以估计到，但是不好估算，人性化要求很高，用户的体验要求高。



# COCOMO模型

结构性成本模型 COCOMO (COConstructive COst MOdel) 是一种利用经验模型进行成本估算的方法。

$$PM_{nominal} = A * (Size)^B$$

- PMnominal: 人月工作量
- A: 工作量调整因子
- B: 规模调整因子
- Size: 规模, 单位是千行代码或功能点数

类型	A	B	说明
组织型	2.4	1.05	相对小的团队在一个高度熟悉的内部环境中开发规模较小, 接口需求较灵活的系统。
嵌入型	3.6	1.2	开发的产品在高度约束的条件下进行, 对系统改变的成本很高。
半独立型	3.0	1.12	介于上述两者中间



# 用例点估算

	基本流	扩展流	业务规则	折合标准用例
功能A	12	3	4	2.3
功能B	8	4	3	1.8
功能C	6	2	3	1.4
合计				5.5

标准用例 = (基本流 + 扩展流 + 2 × 业务规则) / 10

生产率 = 6 工作日 / 单位用例

工作量 = 6 × 5.5 = 33 工作日





# 故事点方法

## 故事点：

- 故事点是用于表达用户故事、功能或其他工作的总体规模的度量单位，它是一个相对度量单位。
- 使用时可以给每个故事分配一个点值，点值本身并不重要，重要的是点值的相对大小。

## 理想日：

- 理想日是用于表达用户故事、功能或其他工作的总体规模的另外一种度量单位，它是一个绝对度量单位。
- 理想时间是某件事在剔除所有外围活动以后所需的时间；一般为一天有效工作时间的 60-80% 比较合理，但绝不会是全部。



# 故事点方法

**故事点的基本做法：**把一些常见“标准任务”给出一个“标准点数”，形成比较基线；估算时只要是同一类型任务，直接写故事点数而非天数。



举例：自动售货机

用户故事	说明
购买饮料	用户投钱并购买指定饮料
取消购买	用户投钱之后取消购买
输入管理密码	授权人输入管理密码，以便进行补货、定价、取钱等操作
补充饮料	在输入管理密码后，授权人补充饮料
设定价格	在输入管理密码后，授权人可以重新设定饮料价格
取出钱款	在输入管理密码后，授权人可以取出钱箱中的钱
打印月报	在输入管理密码后，授权人打印月销售报表
发出报警	在异常情况发生时，系统自动打开安全警报



# 故事点方法

**故事点的基本做法：**把一些常见“标准任务”给出一个“标准点数”，形成比较基线；估算时只要是同一类型任务，直接写故事点数而非天数。



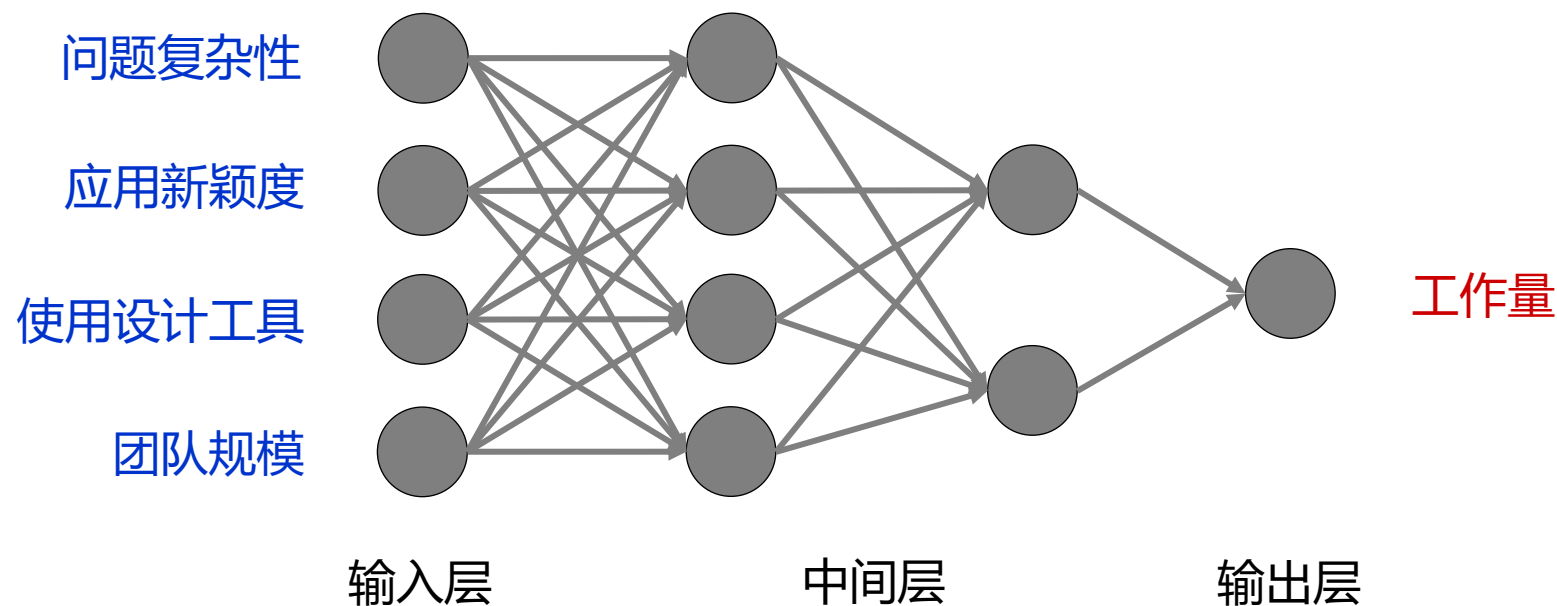
举例：自动售货机

用户故事	故事点	
购买饮料	4	19
取消购买	2	
输入管理密码	1	
补充饮料	3	
设定价格	2	
取出钱款	1	
打印月报	4	
发出报警	2	



# 机器学习方法

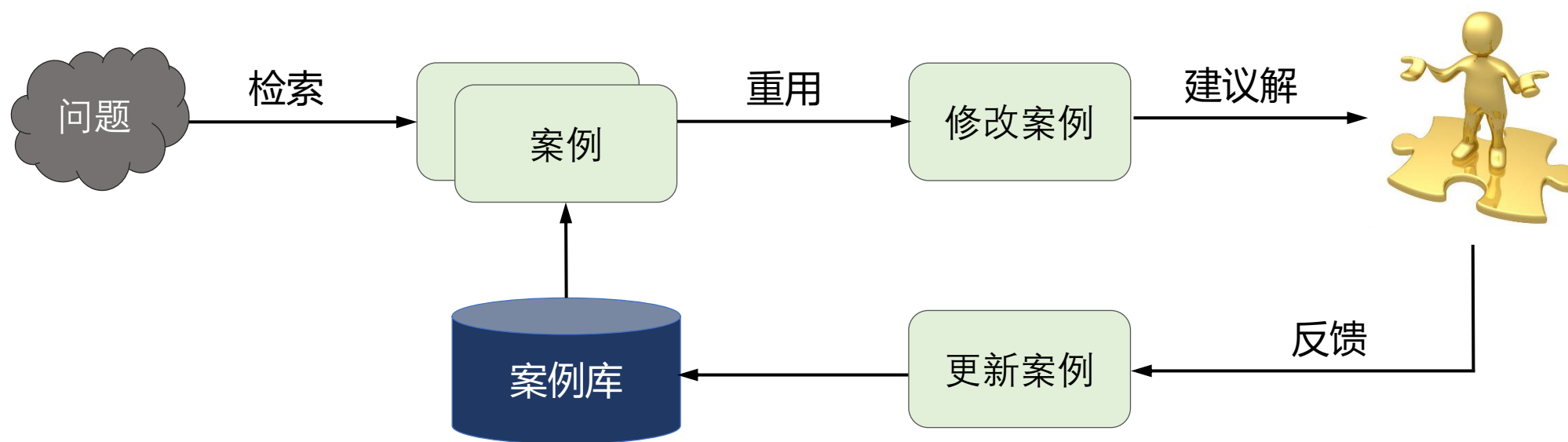
人工神经网络是采用一种学习方法导出一种预测模型，首先建立神经网络，再使用一组历史项目数据（样本数据）训练网络，训练后的网络可以用于估算新项目的工作量。





# 机器学习方法

基于案例的推理方法可以用于基于类推的估算，即识别出与新项目类似的案例，再调整这些案例，使其适合新项目的参数。





## 2.2 软件项目开发管理

- 软件项目管理概念及特征
- 软件项目估算
- 项目进度安排
- 项目风险管理



# 项目进度安排

## ■ 软件延期交付:

- 不切实际的开发日期;
- 客户需求变更;
- 对完成该工作量所需资源估计不足;
- 出现了无法应付的技术难题;
- 出现了无法预计的人力问题;
- 由于项目团队成员之间的交流不畅而导致的延期;
- 项目管理者未发现项目进度拖后, 也未能采取措施解决这一问题;



# 失败案例

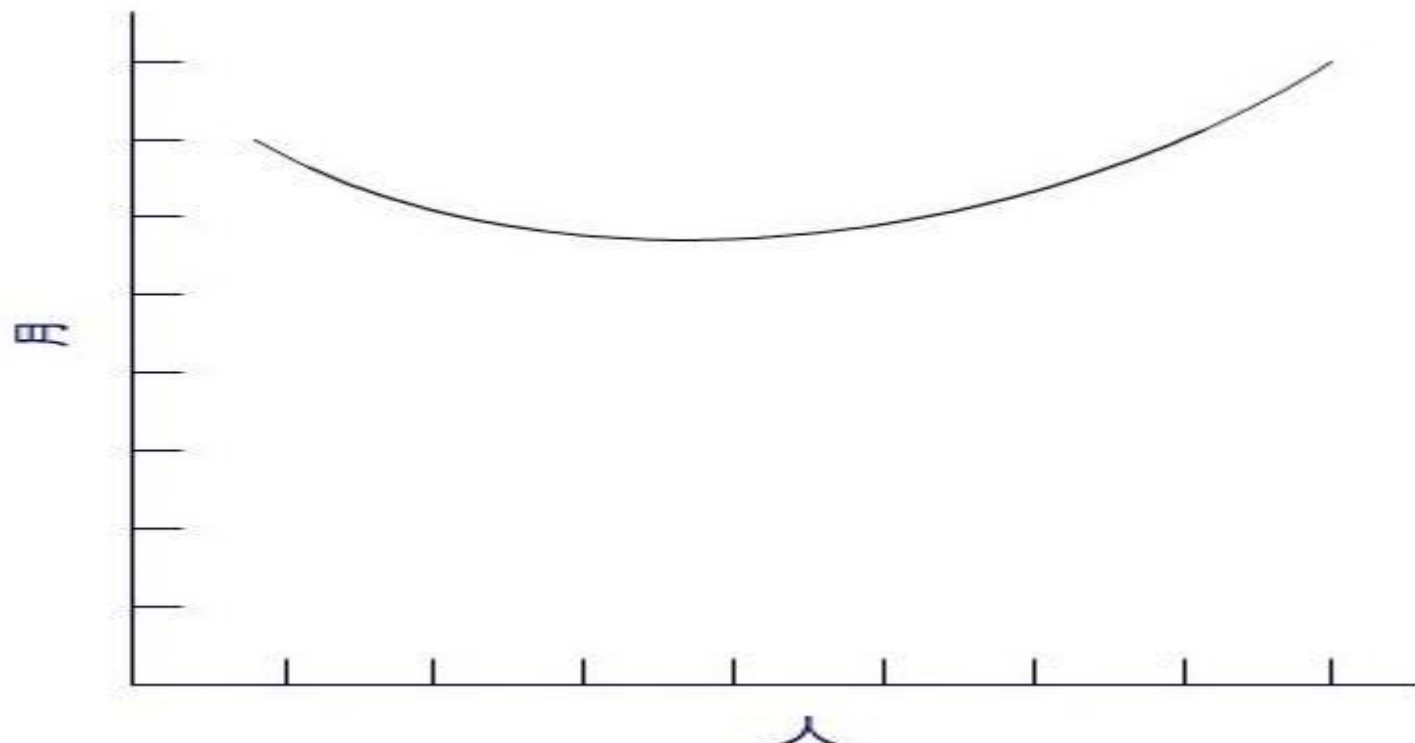
- IBM公司开发OS/360系统，共有**4000**多个模块，约**100**万条指令，投入**5000**人年，耗资数亿美元，结果还是延期交付。在交付使用后的系统中仍发现大量（**2000**个以上）的错误。
- 丹佛新国际机场是曼哈顿机场的两倍，宽为希思机场的**10**倍，可以全天候同时起降三架喷气式客机；投资**1.93**亿美元建立了一个地下行李传送系统，总长**21**英里，有**4000**台遥控车，可按不同线路在**20**家不同的航空公司柜台、登机门和行李领取处之间发送和传递行李；支持该系统的是**5000**个电子眼、**400**台无线电接受机、**56**台条形码扫描仪和**100**台计算机。按原定计划要在**1993**年万圣节前启用，但一直到**1994**年**6**月，机场的计划者还无法预测行李系统何时能达到可使机场开放的稳定程度。





# 人员工作量关系

人月神话中的图示，随着人员的增加开发时间的变化情况





# 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: 将项目划分为多个活动、任务。 **PBS、WBS**
- Step 2: 确定任务项目依赖性。 **定义任务网络**
- Step 3: 时间分配。 **为任务安排工作时间段**
- Step 4: 确认任务资源。 **确认每个任务的资源需求**
- Step 5: 确定责任。 **确定每个任务的负责人和参与人**
- Step 6: 明确结果。 **明确任务的输出结果（文档或部分软件）**
- Step 7: 确定里程碑(milestones)。 **确定任务与项目里程碑关联**

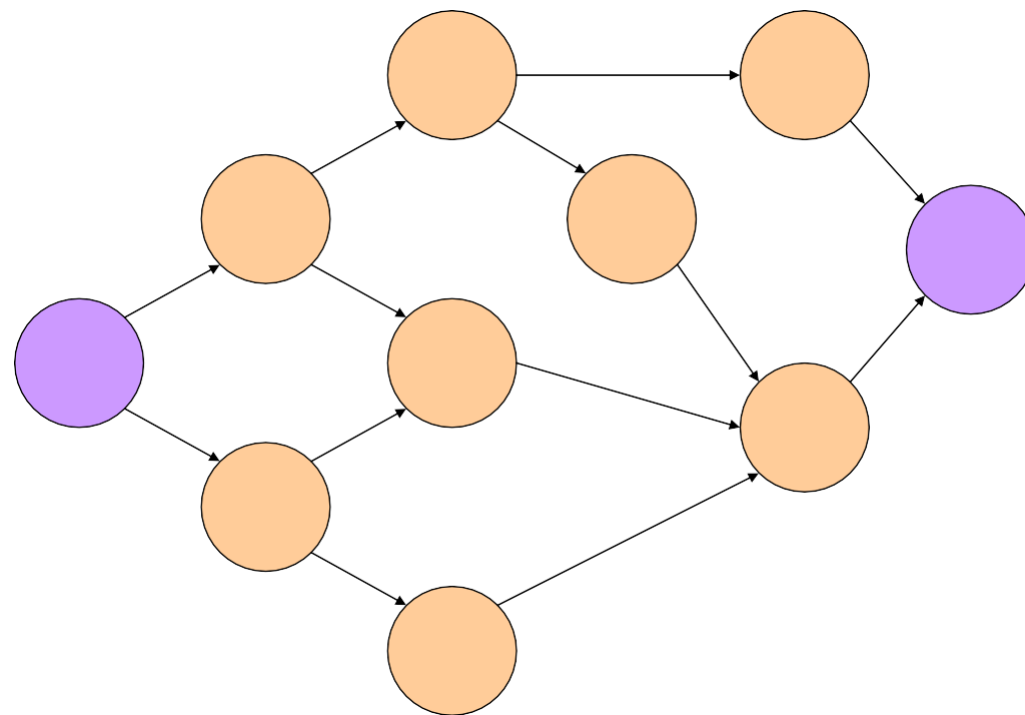


## Step1: 工作量分配

- 40-20-40 法则
- 40%的工作量分配给前期的分析和设计
- 20%的工作量分配给编码
- 40%分配给测试

## Step 2: 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。





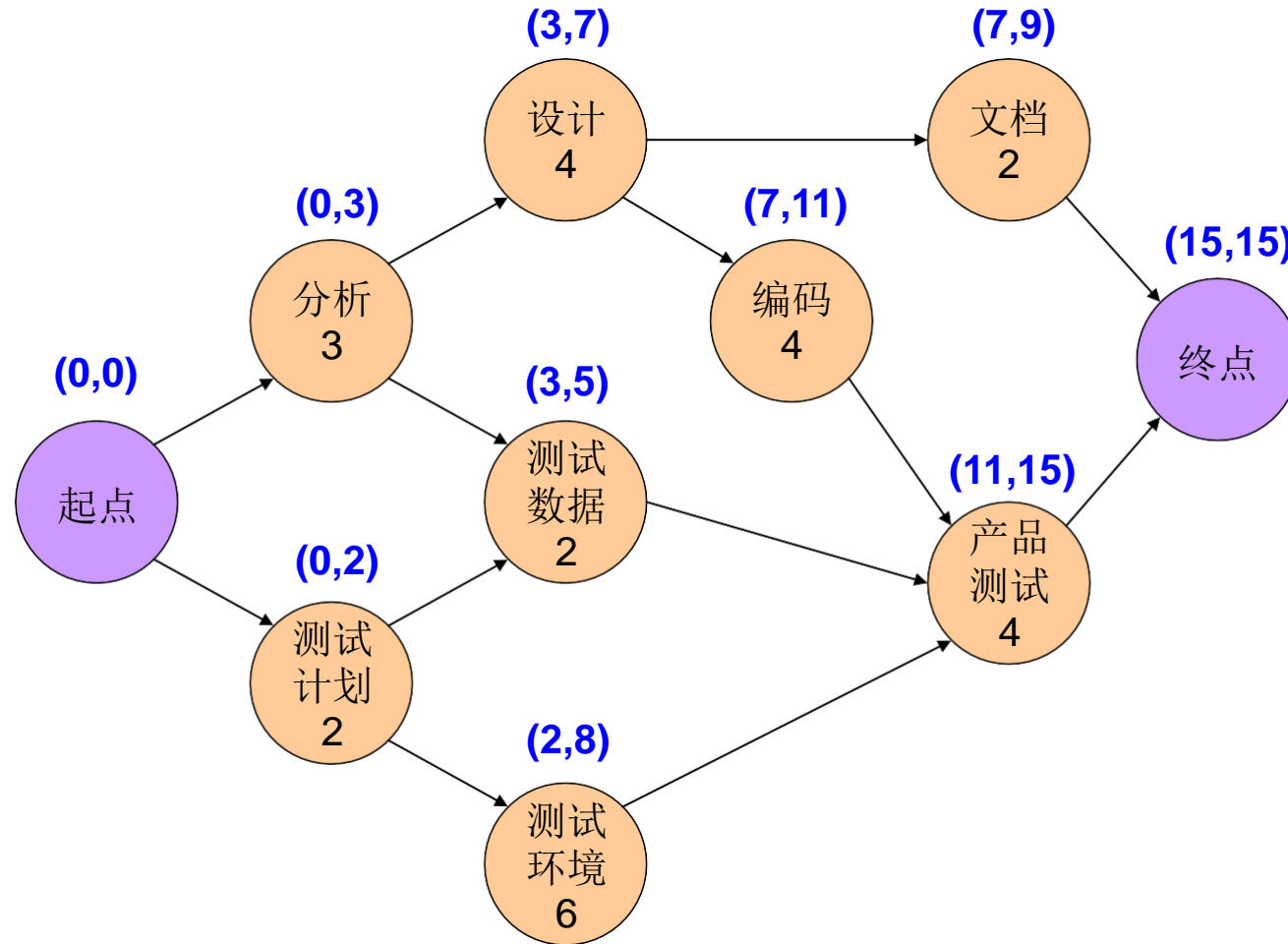
## Step 3: 时间分配

- 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 3.1: 标出任务的最早开始(ES)与结束时间(EF);
  - Step 3.2: 标出任务的最迟开始(LS)与结束时间(LF);
  - Step 3.3: 计算关键路径(Critical Path);
  - Step 3.4: 确定任务的开始/结束时间。
  - Step 3.5: 绘制最终的任务进度安排(甘特图, Gantt Diagram)

最早开始(ES)	持续时间	最早结束(EF)
任务名称		
最晚开始(LS)	松弛量	最晚结束(LF)

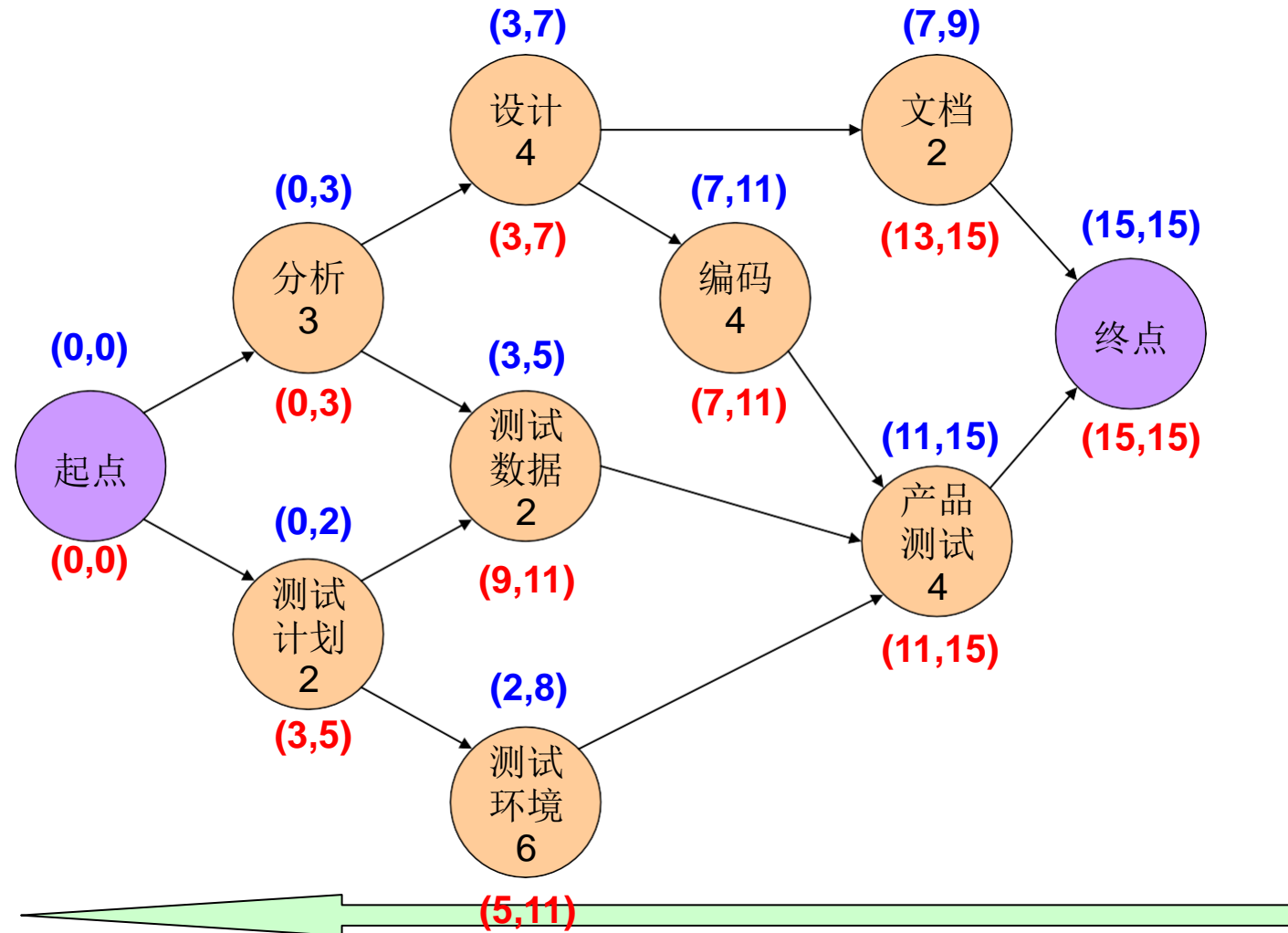


# Step 3.1 标出最早开始/结束日期





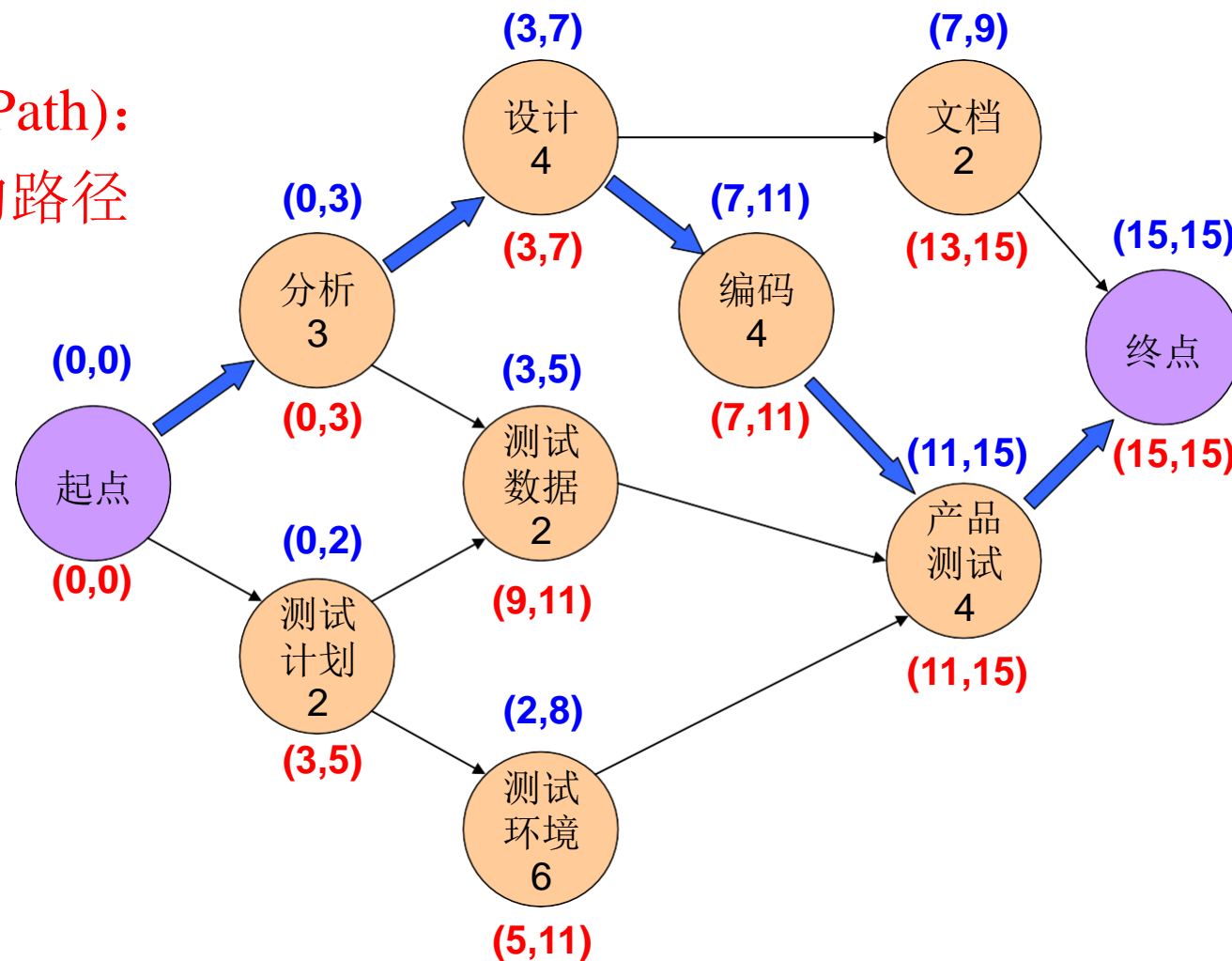
## Step 3.2 标出最晚开始/结束日期





## Step 3.3: 计算关键路径

关键路径(Critical Path):  
持续时间最长的路径

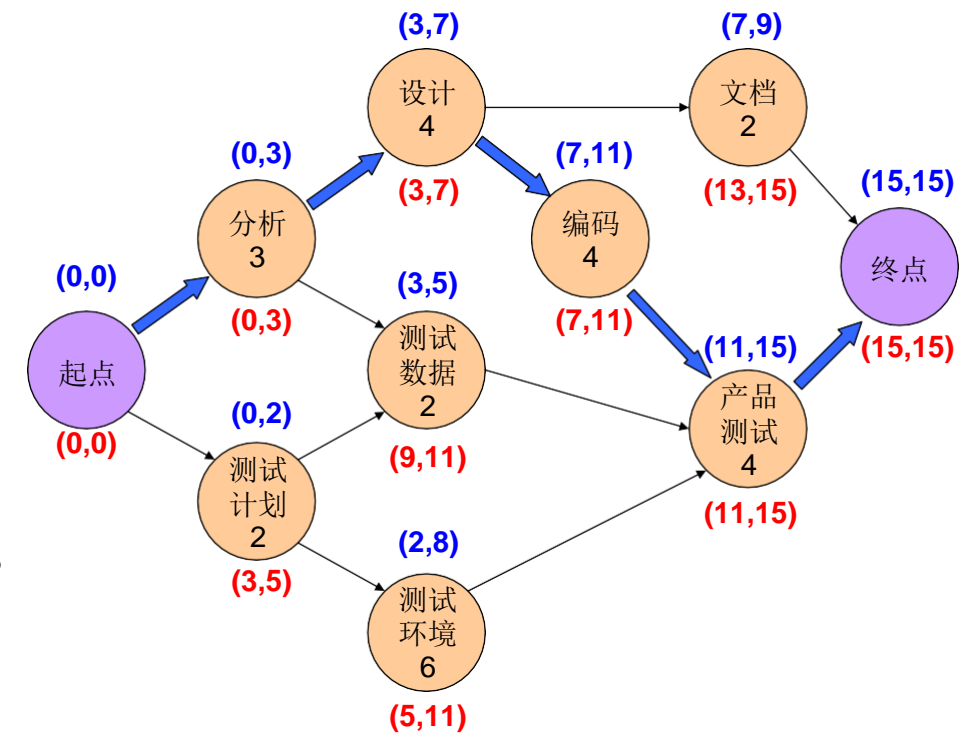






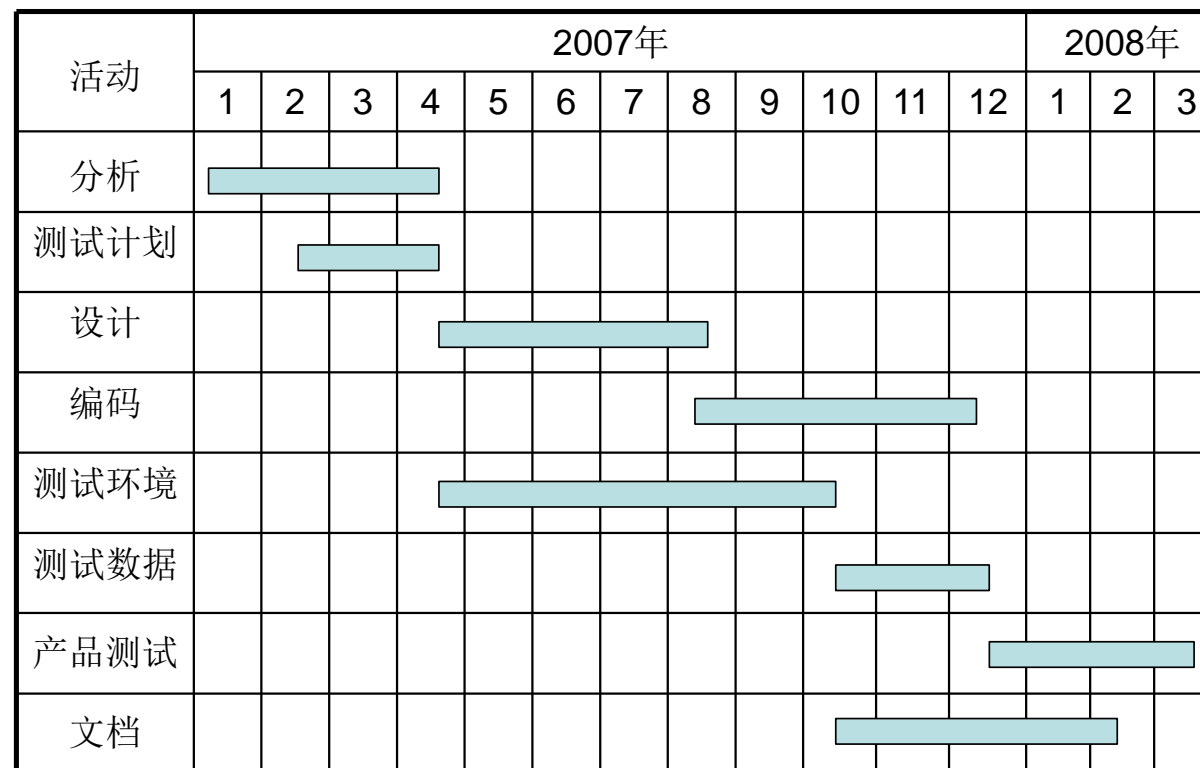
## Step 3.4: 确定最终开始/结束日期

- 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。



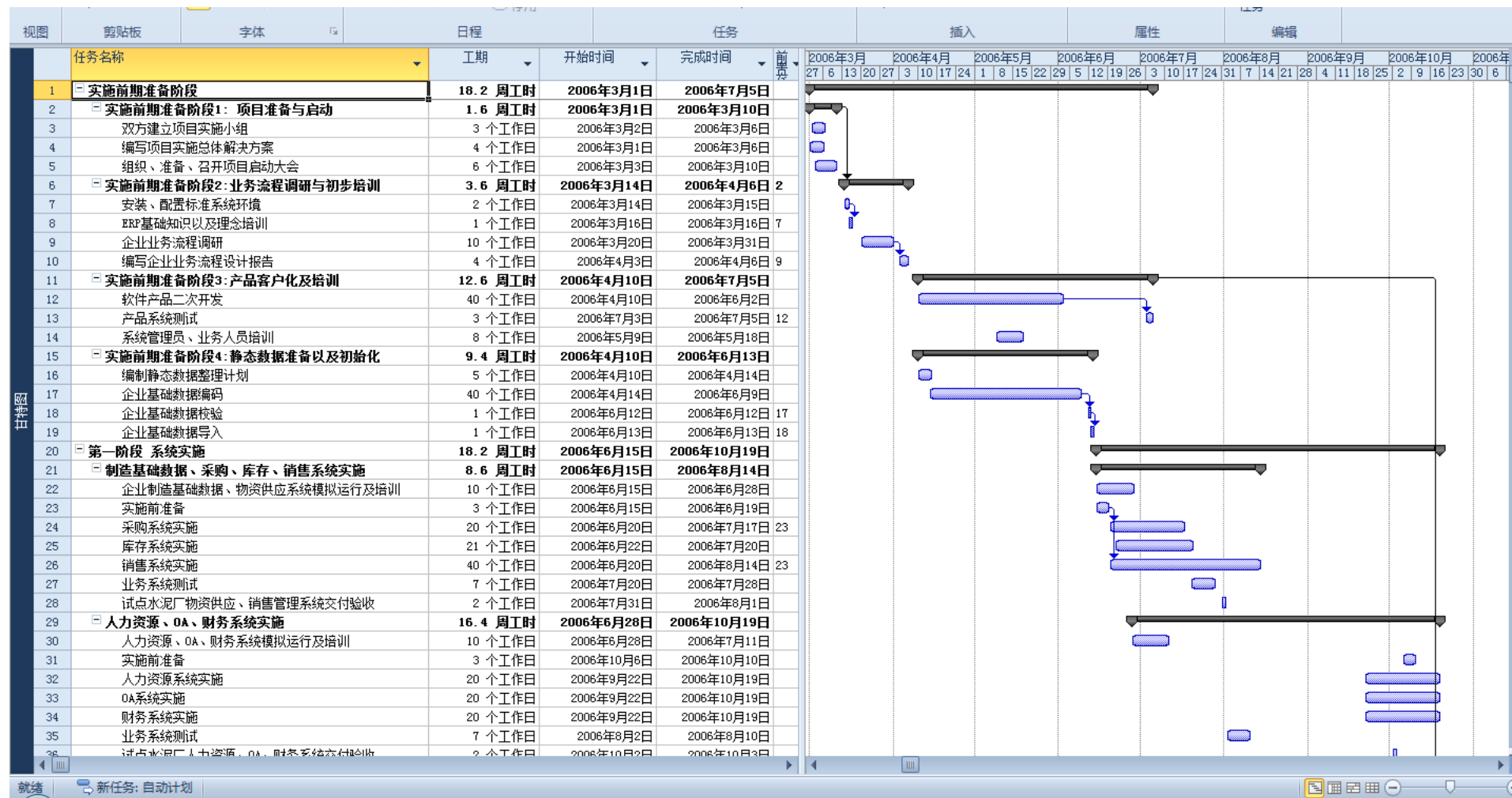
# Step 3.5: 绘制任务进度安排图

甘特图 (Gantt Chart) 是一种通用的显示进度方法，其横轴表示时间，纵轴表示活动，线条表示在整个期间上计划和实际的活动完成情况。





# Microsoft Project 中的 Gantt 图



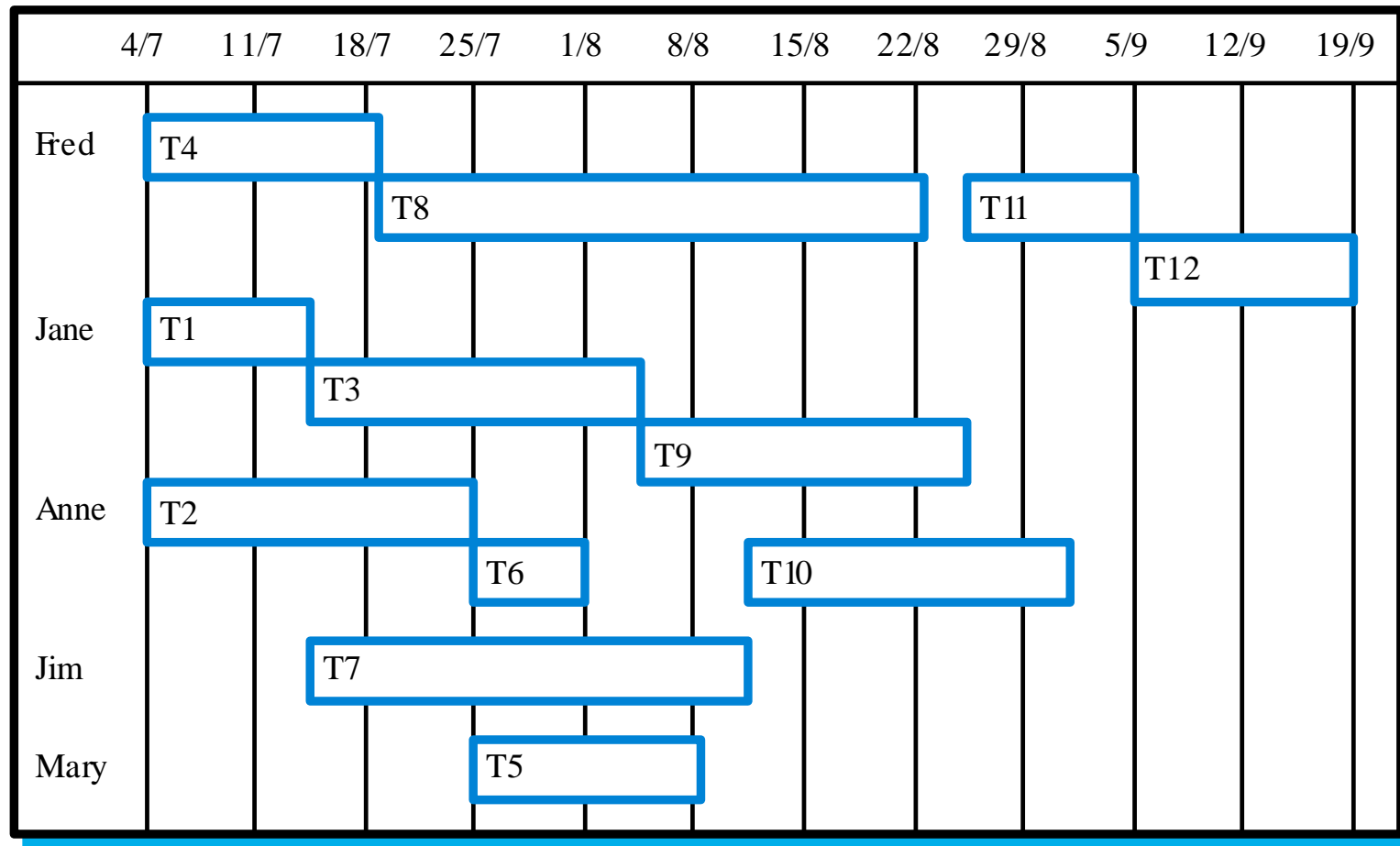


## Step 4~7 资源、产出与里程碑

- Step 1: 将项目划分为多个活动、任务。PBS、WBS
- Step 2: 确定任务项目依赖性。定义任务网络
- Step 3: 时间分配。为任务安排工作时间段
- Step 4: 确认任务资源。确认每个任务的资源需求
- Step 5: 确定责任。确定每个任务的负责人和参与人
- Step 6: 明确结果。明确任务的输出结果（文档或部分软件）
- Step 7: 确定里程碑(milestones)。确定任务与项目里程碑关联



# 人员/资源分配图



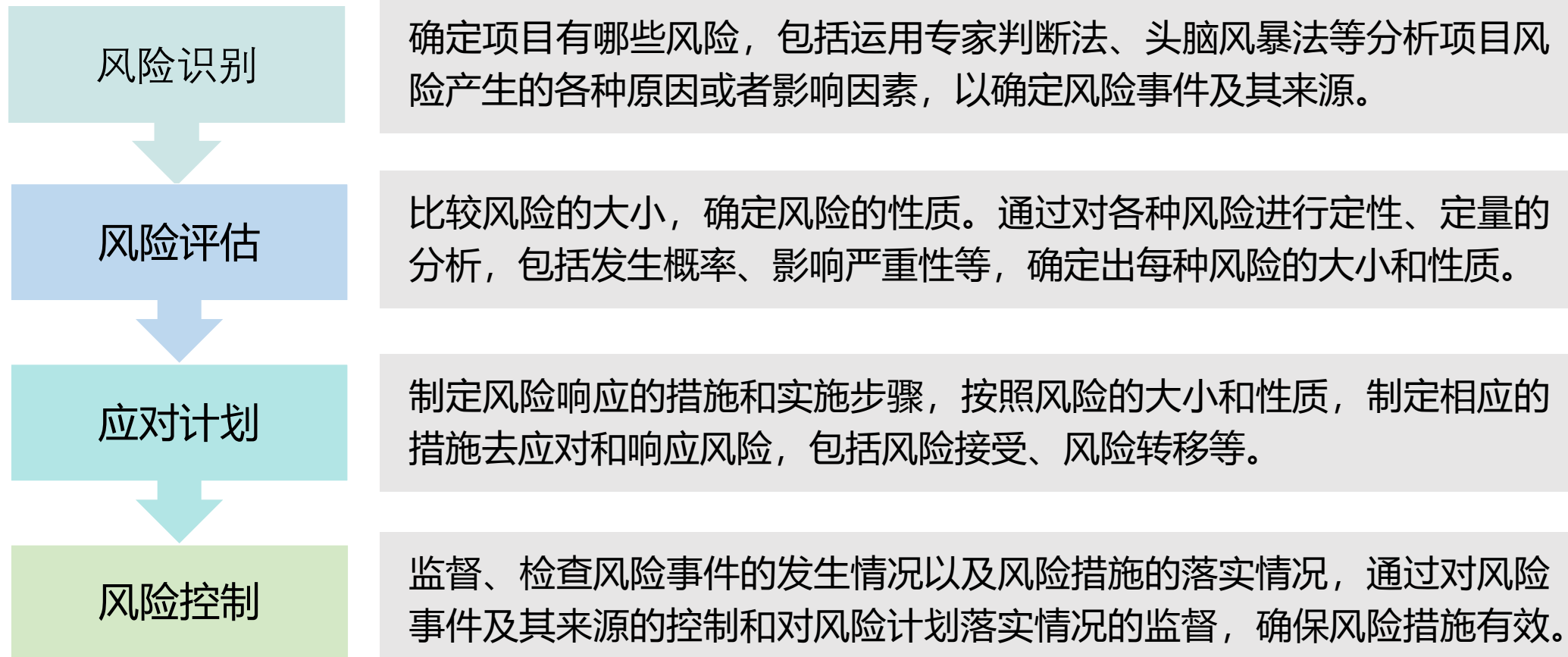


## 2.2 软件项目开发管理

- 软件项目管理概念及特征
- 软件项目估算
- 项目进度安排
- 项目风险管理



# 项目风险管理





# 风险识别

## ■ 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：建立风险表

- Step 1: 列出可能的风险及风险类型；
- Step 2: 估计风险发生的可能性或概率；
- Step 3: 估计风险可能产生的影响或后果；
- Step 4: 确定风险缓解策略

编号	风险描述	风险类型	概率	影响程度	后果	风险缓解策略
1	规模估算不准确	产品规模	60%	严重的	...	迭代方法
2	用户数量超出想象	产品规模	30%	轻微的	...	详细调研
3	最终用户抵制新系统	商业影响	70%	严重的	...	提前与用户沟通，了解使用习惯
4	交付日期将提前	商业影响	40%	灾难的	...	核心功能尽早开发
5	用户将改变需求	产品规模	50%	轻微的	...	架构设计灵活
6	技术到不到预期效果	开发技术	40%	灾难的	...	尝试多种可能技术
7	人员缺乏经验	人员	80%	严重的	...	安排客户培训业务
8	缺少对工具的培训	开发环境	30%	可忽略的	...	查资料、自学
9	人员变动频繁	人员	80%	严重的	...	加强团队凝聚力、关键岗位配备后备人员
	.....					



# 风险缓解、监测和管理

## ■ 风险缓解

- 确定引起风险因素
- 制定风险缓解方案

## ■ 风险监测

- 监测风险因素
- 监测风险缓解措施的效力

## ■ 风险管理及应急计划

- 缓解工作已经失败，风险已经发生
- 启动应急计划



# 风险管理的7个原则

- 保持全面观点：考虑软件风险及软件所要解决的任务问题；
- 采用长远观点：考虑将来要发生的风险，并制定应急计划使将来发生的风险成为可管理的；
- 鼓励广泛交流：如果有人提出一个潜在的风险，要重视它；
- 结合：考虑风险时必须与软件过程相结合；
- 强调持续的过程：整个软件过程中，要持续保持警惕。随着信息量增加，要修改已识别的风险；随着知识的增加，要加入新的风险；
- 开发共享的产品：如果所有利益相关者共享相同版本的软件产品，有利于风险识别和评估；
- 鼓励协同工作：汇聚利益相关者的智慧、技能和知识；



# 小结

- 软件管理概念及特征
- 软件项目估算
- 项目进度安排
- 项目风险管理