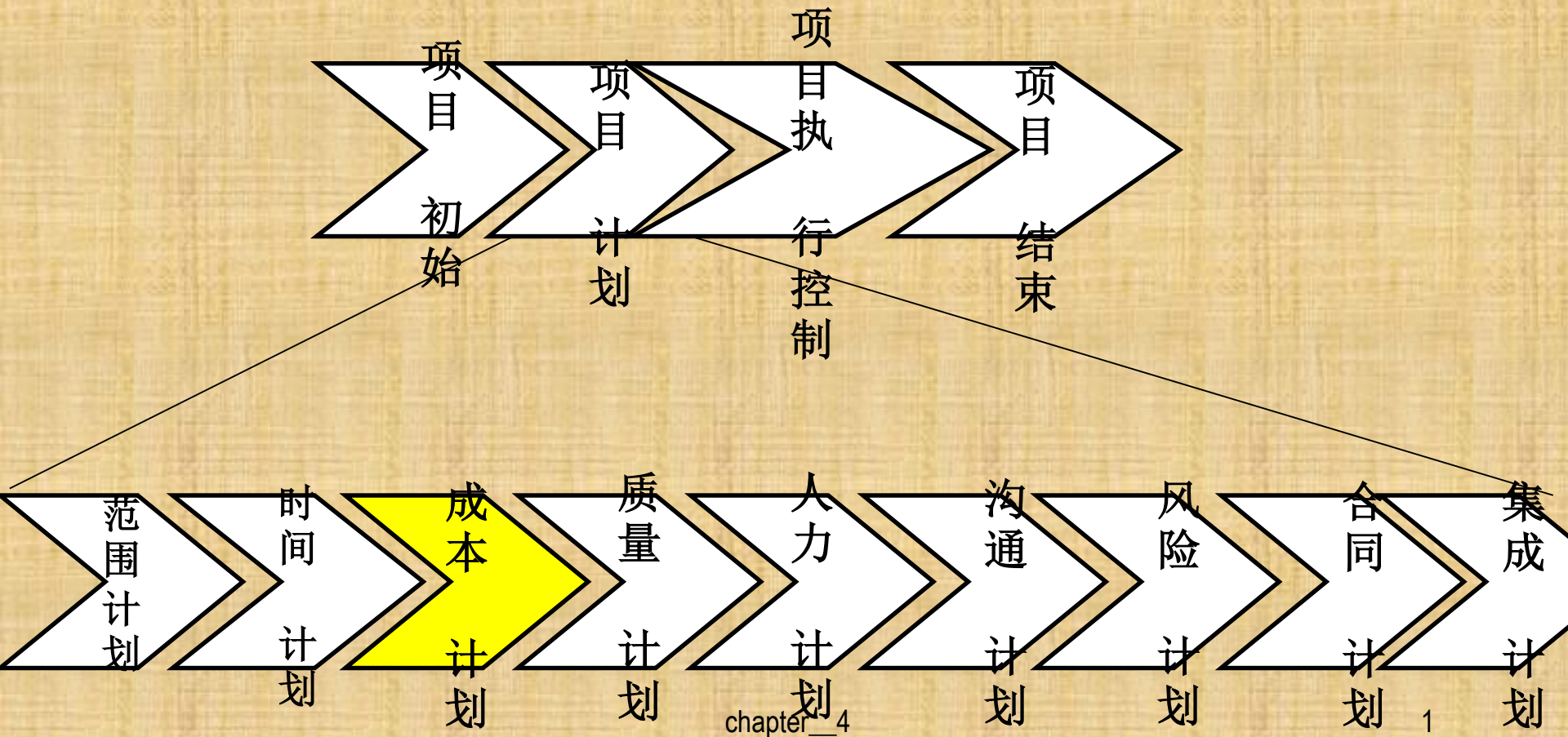


软件项目管理



RoadMap



软件项目管理

软件项目成本计划

本章要点

- ❑ 一、软件项目规模成本的概念
- ❑ 二、成本估算过程
- ❑ 三、成本估算方法
- ❑ 四、成本预算
- ❑ 五、案例分析

成本管理过程

- 资源计划编制：
 - 确定项目需要的资源种类和数量
- 成本估算：中心环节
 - 编制一个为完成项目各活动所需要的资源成本的近似估算
- 成本预算：项目进度
 - 将总成本估算分配到各单项工作活动上
- 成本控制：项目跟踪
 - 控制项目预算的变更

关于估算

- ❑ 估算不是很准确的，有误差的
- ❑ 经验（历史）数据非常重要
- ❑ 不要太迷信数学模型

软件项目规模

- ❑ 软件项目规模即工作量，是从软件项目范围中抽出的软件功能，然后确定每个软件功能所必须执行的一系列软件工程任务
- ❑ 包括：软件规划，软件管理，需求，设计，编码，测试，以及后期的维护等任务。

规模的单位

- ❑ LOC (Loc of Code)
 - ❑ 源代码程序长度的测量
- ❑ FP (Function Point)
 - ❑ 用系统的功能数量来测量
- ❑ 人月
- ❑ 人天
- ❑ 人年

软件项目成本

- ❑ 完成软件规模相应付出的代价。
- ❑ 待开发的软件项目需要的资金。
- ❑ 人的劳动的消耗所需要的代价是软件产品的主要成本

成本的单位

- ❑ 货币单位
 - ❑ 人民币元
 - ❑ 美元
 - ❑

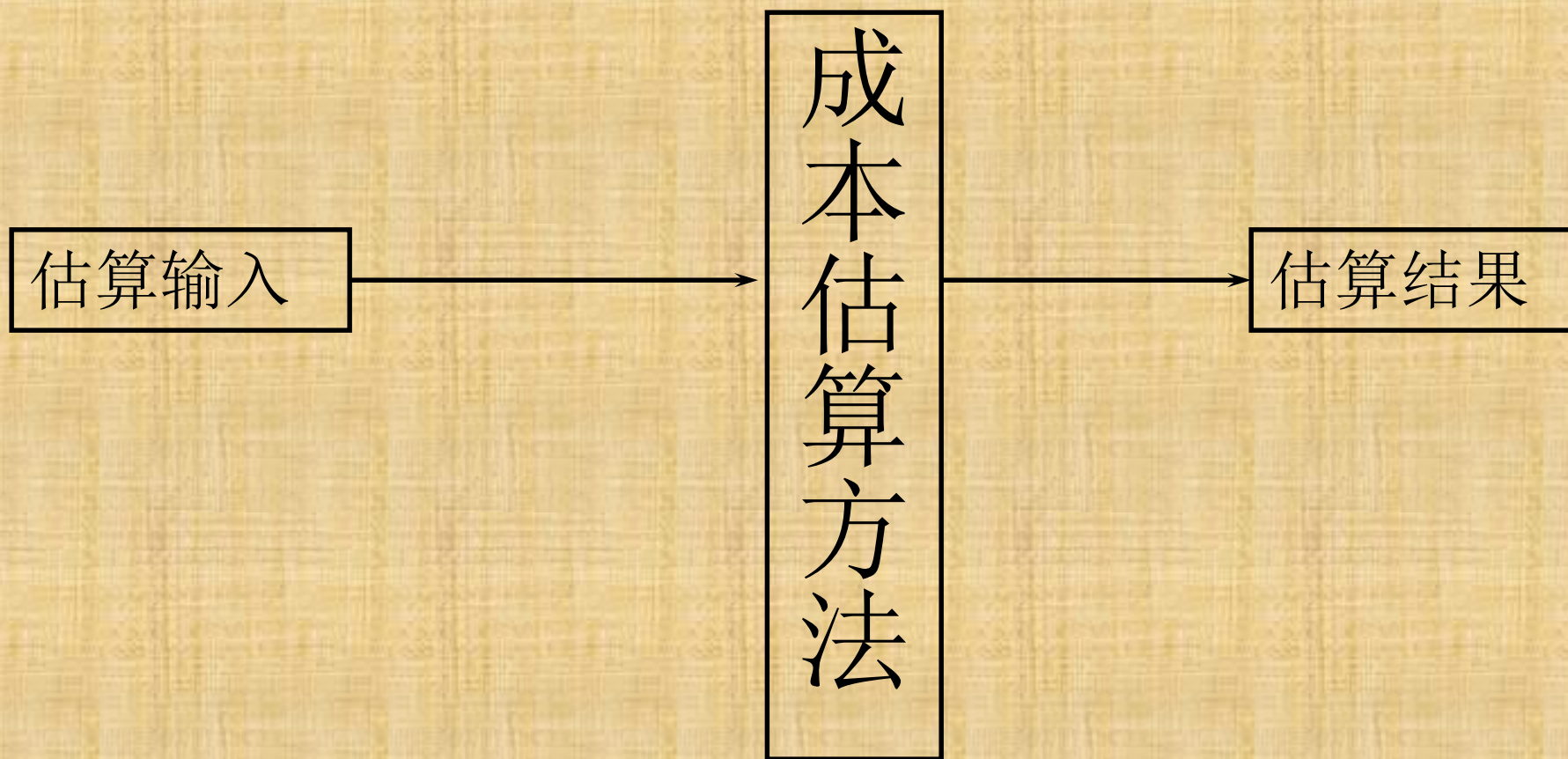
软件的规模 and 成本的关系

- ❑ 规模是成本的主要因素，是成本估算的基础
- ❑ 有了规模就确定了成本，

本章要点

- ❑ 一、软件项目规模成本的概念
- ❑ 二、成本估算过程
- ❑ 三、成本估算方法
- ❑ 四、成本预算
- ❑ 五、案例分析

成本估算过程



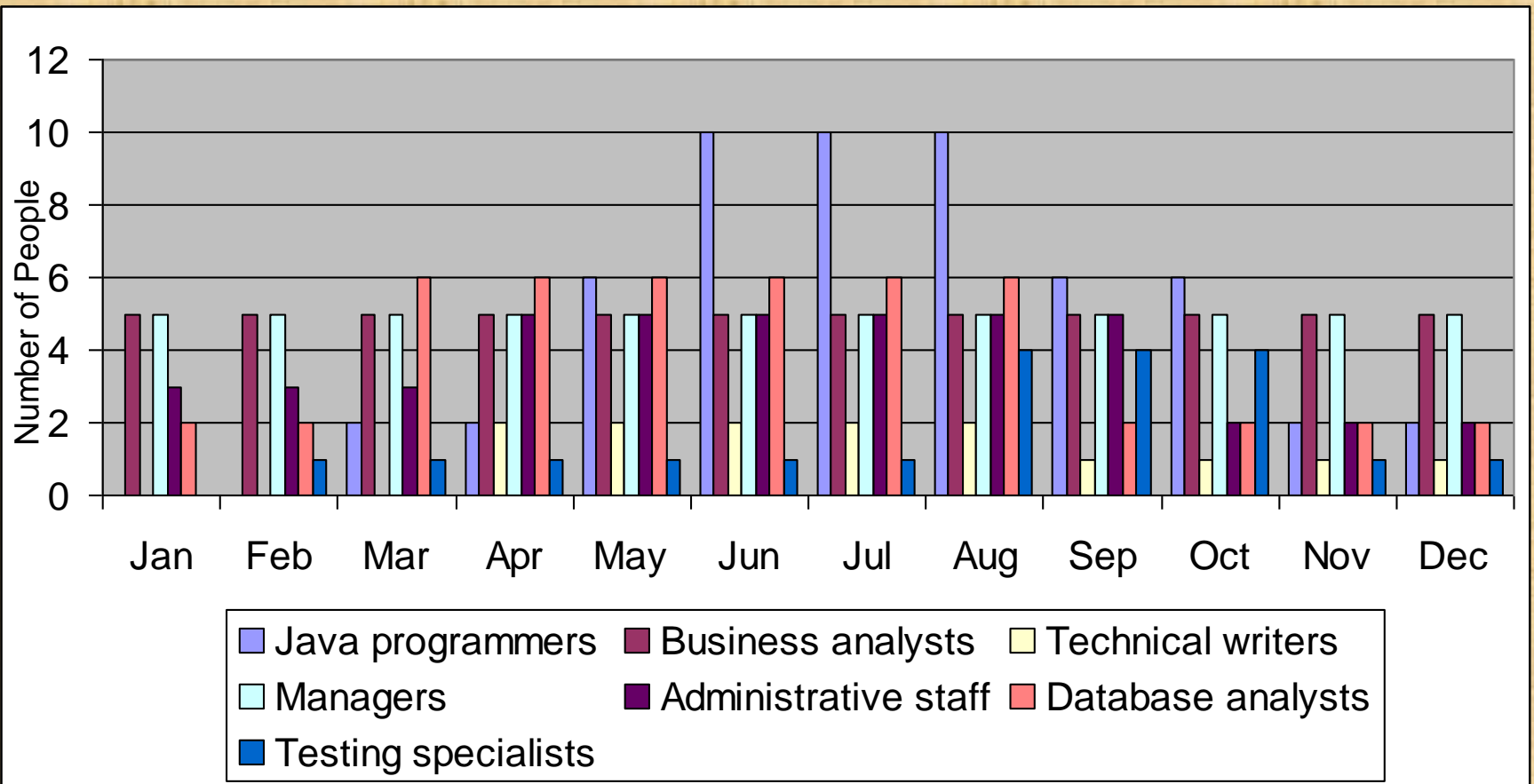
成本估算输入

- ❑ 项目需求、 WBS
- ❑ 历史项目度量
- ❑ 资源要求（资源编制计划）
- ❑ 资源消耗率：如人员成本：100元/小时
- ❑ 进度规划：项目总进度（一般是合同要求）
- ❑ 学习曲线

资源规划

- 需要的资源种类、数量等

Sample Resource Histogram for a Large IT Project



成本估算

- ❑ 直接成本
- ❑ 间接成本

直接成本

- ▣ 与具体项目相关的成本

间接成本

- ❑ 不能具体到某个项目中的成本，
- ❑ 可以分摊到各个具体项目中的成本，例如：
 - ❑ 培训
 - ❑ 房租水电
 - ❑ 员工福利
 - ❑ 市场费用
 - ❑ 管理费
 - ❑ 其他等等

项目估算输出

❑ 估算文件

- ❑ 资源，资源的数量，质量标准，估算成本等信息
- ❑ 单位：一般是货币单位
- ❑ BAC (Budget At completion)

❑ 估算说明

- ❑ 工作范围
- ❑ 估算的基础和依据
- ❑ 估算的假设
- ❑ 估算的误差变动等

估算说明

- ❑ 预测所需要的总工作量的过程。
- ❑ 是一种量化的结果
- ❑ 可以有一些误差
- ❑ 成本估算不同于项目定价
- ❑ 贯穿于软件的生存周期。

本章要点

- ❑ 一、软件项目规模成本的概念
- ❑ 二、成本估算过程
- ❑ 三、成本估算方法
- ❑ 四、成本预算
- ❑ 五、案例分析

估算的基本方法

1. 代码行、功能点、对象点、用例点
2. 类比（自顶向下）估算法
3. 自下而上估算法
4. 参数法估算法
5. 专家估算法

代码行（LOC）

从软件程序量的角度定义项目规模。

- ❑ 要求功能分解足够详细的
- ❑ 有一定的经验数据（类比和经验方法）
- ❑ 与具体的编程语言有关

代码行（LOC）缺点

1. 对代码行没有公认的可接受的标准定义
2. 代码行数量依赖于所用的编程语言和个人的编程风格.
3. 在项目早期,需求不稳定、设计不成熟、实现不确定的情况下很难准确地估算代码量.
4. 代码行强调编码的工作量,只是项目实施阶段的一部分

功能点（FP: Function point）

- ❑ 用系统的功能数量来测量其规模
- ❑ 与实现产品所使用的语言和技术没有关系的
- ❑ 两个评估
 - ❑ 内部基本功能
 - ❑ 外部基本功能
- ❑ 加权和量化

功能点的公式

- ❑ $FP = UFC * TCF$
 - ❑ UFC: 未调整功能点计数
 - ❑ TCF: 技术复杂度因子

UFC-未调整功能点计数

功能计数项：

1. 外部输入 EI
2. 外部输出 EO
3. 外部查询 EQ
4. 内部逻辑文件 ILF
5. 外部接口文件 EIF

UFC-未调整功能点计数

功能计数项的复杂度等级

	复杂度权重因素		
项	简单	一般	复杂
外部输入	3	4	6
外部输出	4	5	7
外部查询	3	4	6
外部文件	5	7	10
内部文件	7	10	15

功能点计算实例-UFC

	功能点		
项	简单	一般	复杂
外部输入	6 * 3	2 * 4	3 * 6
外部输出	7 * 4	7 * 5	0 * 7
外部查询	0 * 3	2 * 4	4 * 6
外部文件	5 * 5	2 * 7	3 * 10
内部文件	9 * 7	0 * 10	2 * 15
总计			
UFC	301	chapter__4	29

TCF-技术复杂度因子

$TCF = 0.65 + 0.01 (\sum(F_i))$: $F_i: 0-5$, $TCF: 0.65-1.35$

技术复杂度因子			
F1	可靠的备份和恢复	F2	数据通信
F3	分布式函数	F4	性能
F5	大量使用的配置	F6	联机数据输入
F7	操作简单性	F8	在线升级
F9	复杂界面	F10	复杂数据处理
F11	重复使用性	F12	安装简易性
F13	多重站点	F14	易于修改

技术复杂度因子的取值范围

调整系数	描述
0	不存在或者没有影响
1	不显著的影响
2	相当的影响
3	平均的影响
4	显著的影响
5	强大的影响

功能点计算实例

- $FP = UFC * TCF$

- $UFC = 301$

- $TCF = 0.65 + 0.01 (14 * 3) = 1.07$

- $FP = 301 * 1.07 = 322$

功能点与代码行的转换

语言	代码行/FP
Assembly	320
C	150
COBOL	105
FORTRAN	105
PASCAL	91
ADA	71
PL/1	65
PROLOG/LISP	64
SMALLTALK	21
SPREADSHEET	46

chapter 46

33

对象点（OP）

- ❑ 对象点是基于对象的软件产品规模估算。
- ❑ 著名的Probe方法——Watts Humphrey

对象规模表 (C++)

方法种类	很小	小	中	大	很大
计算	2.34	5.13	11.25	24.66	54.04
数据	2.6	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15	21.62	28.93
逻辑	7.55	10.98	15.98	23.25	33.83
设置	3.88	5.04	6.56	8.53	11.09
文本	3.75	8.00	17.07	36.41	77.66

Probe方法的步骤

1. 基于产品需求构建体系结构和概要设计
2. 对设计中的每个类（面向对象方法中的Class）的输入和交互，标识所设计的对象属于表中哪类方法并估算其复杂性
3. 将上述标识的结果构造成一个如上表形式的矩阵，然后将这个矩阵中的值与上表中对应的值相乘
4. 将上述所有相乘结果相加求和，产生估算结果

对象点的估计-举例

方法种类	很小	小	中	大	很大
计算	2.34	5.13 * 5	11.25	24.66	54.04
数据	2.6	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15 * 8	21.62	28.93
逻辑	7.55	10.98	15.98	23.25	33.83
设置	3.88	5.04	6.56	8.53 * 6	11.09
文本	3.75	8.00	17.07	36.41	77.66
规模估算	$5.13 * 5 + 16.15 * 8 + 8.53 * 6 = 206.03$				

估算的基本方法

- ❑ 代码行、功能点、对象点
- ❑ 类比（自顶向下）估算法
- ❑ 自下而上估算法
- ❑ 参数法估算法
- ❑ 专家估算法

类比-定义

- ❑ 估算人员根据以往的完成类似项目所消耗的总成本（或工作量），来推算将要开发的软件的总成本（或工作量），然后按比例将它分配到各个开发任务单元中
- ❑ 是一种自上而下的估算形式

类比—使用情况

- ❑ 有类似的历史项目数据
- ❑ 信息不足（要求不是非常精确）的时候
- ❑ 在合同期和市场招标时

类比—特点

- ❑ 简单易行, 花费少
- ❑ 具有一定的局限性
- ❑ 准确性差, 可能导致项目出现困难

类比—举例

例 3:一个待估算工作量的项目 P_0 和已完成的项目 P_1, P_2 .

Project	Project type	Programming language	Team size	Project size	Effort
P_0	MIS	C	9	180	
P_1	MIS	C	11	200	1 000
P_2	Real time	C	10	175	900

项目间的相似度计算如下:

P_0 vs. P_1	P_0 vs. P_2
$\delta(P_{01}, P_{11}) = \delta(\text{MIS}, \text{MIS}) = 0$	$\delta(P_{01}, P_{21}) = \delta(\text{MIS}, \text{Real Time}) = 1$
$\delta(P_{02}, P_{12}) = \delta(\text{C}, \text{C}) = 0$	$\delta(P_{02}, P_{22}) = \delta(\text{C}, \text{C}) = 0$
$\delta(P_{03}, P_{13}) = \delta(9, 11)$ $= [(9-11)/(9-11)]^2 = 1$	$\delta(P_{03}, P_{23}) = \delta(9, 10)$ $= [(9-10)/(9-11)]^2 = 0.25$
$\delta(P_{04}, P_{14}) = \delta(180, 200)$ $= [(180-200)/(200-175)]^2 = 0.64$	$\delta(P_{04}, P_{24}) = \delta(180, 175)$ $= [(180-175)/(200-175)]^2 = 0.04$
$\text{distance}(P_0, P_1) = (1.64/4)^{0.5} = 0.64$	$\text{distance}(P_0, P_2) = 0.57$

类比—举例

- 证券交易网站
 - 需求类似
 - 历史数据：10万

估算的基本方法

- ❑ 代码行、功能点、对象点
- ❑ 类比（自顶向下）估算法
- ❑ 自下而上估算法
- ❑ 参数法估算法
- ❑ 专家估算法

自下而上一定义

- 利用任务分解结构图, 对各个具体工作包进行详细的成本估算, 然后将结果累加起来得出项目总成本。

自下而上一使用情况

- ❑ 项目开始以后，WBS的开发阶段
- ❑ 需要进行准确估算的时候

自下而上一特点

- ❑ 这种方法相对比较准确，它的准确度来源于每个任务的估算情况
- ❑ 非常费时，估算本身也需要成本支持
- ❑ 可能发生虚报现象

自下而上一举例

银行综合业务系统估算结果

估算的基本方法

- ❑ 代码行、功能点、对象点
- ❑ 类比（自顶向下）估算法
- ❑ 自下而上估算法
- ❑ 参数法估算法
- ❑ 专家估算法

参数估算法一定义

- ❑ 模型方法
- ❑ 一种使用项目特性参数建立数据模型来估算成本的方法，是一种统计技术，如回归分析和学习曲线。

软件成本估算模型

- ❑ 理论导出：不成熟阶段
- ❑ 经验导出：软件估算常常采用

参数估算法—使用情况

- ❑ 存在成熟的项目估算模型
- ❑ 应该具有良好的数据库数据为基础

参数估算法一特点

- ❑ 比较简单, 而且也比较准确
- ❑ 如果模型选择不当或者数据不准, 也会导致偏差

参数成本估算实例（适合单价合同）

项目经理（30美元/小时）	15,000美元	
分析家（20美元/小时）	20,000美元	
技术员（13美元/小时）	2,6000美元	
总劳动开支		61,000美元
一般管理费（35%）	21,350美元	
总劳动费加管理费		82,350美元
额外费用（25%）	16,470美元	
小计		98,820美元
交通费（四次旅行，1000/次）	4000美元	
微机计算机费（2台,3500/台）	7000美元	
打印与复制费	2000美元	
总辅助费		13,000美元
总项目费用开支	chapter__4	111,820美元 54

经验导出成本模型

- ❑ 提供工作量(规模)的直接估计
- ❑ 通过过去项目数据, 进行回归分析, 得出的回归模型

经验导出成本模型

- ❑ 整体公式: $E=A+B*S^C$
 - ❑ E: 以人月表示的工作量
 - ❑ A, B, C: 经验导出的系数
 - ❑ S: 主要的输入参数 (通常是LOC, FP等)

经验导出成本模型(续)

面向LOC驱动的

- ❑ Walston-Felix (IBM)

- ❑ $E = 5.2 * (KLOC)^{0.91}$

- ❑ Bailey-Basili

- ❑ $E = 5.5 + 0.73 * (KLOC)^{1.16}$

- ❑ . COCOMO

- ❑ $E = 3.2 * (KLOC)^{1.05}$

- ❑ Doty

- ❑ $E = 5.288 * (KLOC)^{1.047}$

经验导出成本模型(续)

面向FP驱动的

- ❑ Albrecht and Gaffney

- ❑ $E = -12.39 + 0.0545FP$

- ❑ Kemerer

- ❑ $E = 60.62 * 7.728 * 10^{-8} FP^3$

- ❑ Matson, Barnett

- ❑ $E = 585.7 + 15.12FP$

建议掌握模型

- ❑ I B M模型— (Walston-Felix)
- ❑ COCOMO模型—(Boehm)

I B M模型

1977年，IBM的Walston和Felix提出了如下的估算公式

- $E = 5.2 \times L^{0.91}$ ，L是源代码行数(以KLOC计)，E是工作量(以PM计)
- $D = 4.1 \times L^{0.36}$ ，D是项目持续时间(以月计)
- $S = 0.54 \times E^{0.6}$ ，S是人员需要量(以人计)
- $DOC = 49 \times L^{1.01}$ 。DOC是文档数量(以页计)

举例

采用java 完成项目，366功能点，则

- $L = 366 \times 46 = 16386 \text{行} = 16.386 \text{KLOC}$
- $E = 5.2 \times L^{0.91} = 5.2 \times 16.386^{0.91} = 66 \text{人月}$
- $\text{DOC} = 49 \times L^{1.01} = 49 \times 16.386^{1.01} = 826 \text{页}$

COCOMO (Constructive Cost model)

由Barry Boehm开发的

详见：www.usc.edu (南加州大学网站)

COCOMO模型发展

- COCOMO 81
- COCOMO II
- 模型系列

COCOMO 81

模型类别:

- ❑ 基本COCOMO
- ❑ 中等COCOMO
- ❑ 高级COCOMO

项目类型:

- 有机: Organic
- 嵌入式: Embedded
- 半有机: Semidetached

模型类别

- ❑ 基本COCOMO
 - ❑ 静态单变量模型
- ❑ 中等COCOMO
 - ❑ 基本模型基础上考虑影响因素，调整模型
- ❑ 高级COCOMO
 - ❑ 中等COCOMO模型基础上考虑各个步骤的影响

项目类型

- 有机: Organic,
 - 各类应用程序，例如数据处理、科学计算 等
 - 受硬件的约束比较小，程序的规模不是很大
- 嵌入式: Embedded
 - 系统程序，例如实时处理、控制程序等
 - 紧密联系的硬件、软件 and 操作的限制条件下运行，软件规模任意
- 半有机: Semidetached
 - 各类实用程序，介于上述两种软件之间，例如编译器（程序）
 - 规模和复杂度都属于中等或者更高

基本COCOMO

- ❑ $E = a (KLOC)^b \exp(b)$
- ❑ 其中：
 - ❑ E是所需的人力（人月），
 - ❑ KLOC是交付的代码行
 - ❑ a， b是依赖于项目自然属性的参数：

基本COCOMO系数表

方式	a	b
有机	2.4	1.05
半有机	3.0	1.12
嵌入式	3.6	1.2

举例

一个33.3 KLOC的软件开发项目，属于中等规模、半有机型的项目，采用基本COCOMO：

- o $a=3.0$, $b=1.12$ 。
- o $E = 3.0 * L^{1.12} = 3.0 * 33.3^{1.12} = 152$
PM

中等COCOMO

□ $E = a (KLOC)^b \times \text{乘法因子}$

□ a b 是系数

□ 乘法因子是根据成本驱动属性打分的结果，对公式的校正系数

中等COCOMO系数表

方式	a	b
有机	2.8	1.05
半有机	3.0	1.12
嵌入式	3.2	1.2

成本驱动因子

Cost drivers	Rating levels						Remark
	Very low	Low	Nominal	High	Very high	Extra high	
Product attributes							
RELY	0.75	0.88	1.00	1.15	1.40		Required software reliability
DATA		0.94	1.00	1.08	1.16		Database size
CPLX	0.70	0.85	1.00	1.15	1.30	1.65	Product complexity
Computer attributes							
TIME			1.00	1.11	1.30	1.66	Execution time constraints
STOR			1.00	1.06	1.21	1.56	Main storage constraints
VIRT		0.87	1.00	1.15	1.30		Virtual machine volatility
TURN		0.87	1.00	1.07	1.15		Computer turnaround time
Personnel attributes							
ACAP	1.46	1.19	1.00	0.86	0.71		Analyst capability
AEXP	1.29	1.13	1.00	0.91	0.82		Applications experience
PCAP	1.42	1.17	1.00	0.86	0.70		Programming capability
VEXP	1.21	1.10	1.00	0.90			Virtual machine experience
LEXP	1.14	1.07	1.00	0.95			Language experience
Process attributes							
MODP	1.24	1.10	1.00	0.91	0.82		Use of modern programming practices
TOOL	1.24	1.10	1.00	0.91	0.83		Use of software tools
SCED	1.23	1.08	1.00	1.04	1.10		Required development schedule

乘法因子计算

- 每个属性 F_i 的取值范围为：
很低、低、正常、高、很高、极高，共六级。
正常情况下 $F_i=1$ 。
- Boehm推荐的 F_i 取值范围
0.70, 0.85, 1.00, 1.15, 1.30, 1.65
- 当每个 F_i 的值选定后，乘法因子的计算如下
$$\text{乘法因子} = F_1 * F_2 * \dots F_i \dots * F_n$$

举例（续）

一个33.3 KLOC的软件开发项目，属于中等规模、半有机型的项目，采用中等COCOMO模型

➤ $a=3.0$, $b=1.12$ 。

➤ 乘法因子 $=0.70*0.85*1.....*1.15=1.09$

➤ $E = 3.0 * L^{1.12} = 3.0 * 33.3^{1.12} \times 1.09 = 166 \text{ PM}$

高级（详细）COCOMO

- ❑ 将项目分解为一系列的子系统或者子模型
- ❑ 在一组子模型的基础上更加精确地调整一个模型的属性，

高级（详细）COCOMO

Table 2 An example of different effort multipliers by phase in detailed COCOMO 81^[8]

表2 详细 COCOMO 81 工作量乘数的阶段差异性示例^[8]

Cost drivers	Development phase	Rating levels					
		Very low	Low	Nominal	High	Very high	Extra high
AEXP	RPD (requirement and product design)	1.40	1.20	1.00	0.87	0.75	--
	DD (detailed design)	1.30	1.15	1.00	0.90	0.80	--
	CUT (code and unit test)	1.25	1.10	1.00	0.92	0.85	--
	IT (integration and test)	1.25	1.10	1.00	0.92	0.85	--

估算的基本方法

- ❑ 代码行、功能点、对象点
- ❑ 类比（自顶向下）估算法
- ❑ 自下而上估算法
- ❑ 参数法估算法
- ❑ 专家估算法

专家估算法

- 由多位专家进行成本估算，一个专家可能会有偏见，最好由多位专家进行估算，取得多个估算值，最后得出综合的估算值。

专家估算法-De i phi

- ❑ 组织者发给每位专家一份软件系统的规格说明和一张记录估算值的表格，请他们估算
- ❑ 专家详细研究软件规格说明后，对该软件提出3个规模的估算值
 - ❑ 最小 a_i
 - ❑ 最可能的 m_i
 - ❑ 最大 b_i
- ❑ 组织者为专家的表格中的答复进行整理
- ❑ 计算每位专家的 $E_i = (a_i + 4m_i + b_i) / 6$,

专家估算法-Deiphi (续)

- ❑ 综合结果后： $E = E_1 + E_2 + \dots E_n / n$ (N: 表示N 个专家)
- ❑ 再组织专家无记名填表格，比较估算差，并查找原因
- ❑ 如果各个专家的估算差异超出规定的范围（例如：15%），则需重复上述过程，最终可以获得一个多数专家共识的软件规模

专家估算法-举例

■ 某多媒体信息查询系统—专家估算

- 专家1: 1, 8, 9= $\rangle (1+9+4 * 8) / 6 = 7$ (万元)
- 专家2: 4, 6, 8= $\rangle (4+8+4*6) / 6 = 6$ (万元)
- 估算结果= $(6+7) / 2 = 6.5$ (万元)

估算方法总结

- ❑ 初期
 - ❑ 类比
 - ❑ 专家估算
- ❑ 计划阶段
 - ❑ 自下而上
 - ❑ 参数模型
- ❑ 实施阶段（包括变更发生）
 - ❑ 自下而上
 - ❑ 参数模型

成本估算方法综述

- ❑ 主要考虑三种模型：类比法，自下而上法，参数法。
- ❑ 自下而上法费时费力，参数法比较简单
- ❑ 自下向上法与参数法的估计精度相似
- ❑ 类比法通常用来验证参数法和自下而上法的结果

各种方法不是孤立的，应该注意相互的结合使用

实用软件估算模型

是一种自下而上和参数法的结合模型, 步骤如下:

1. 对任务进行分解: 1, 2, ..., i ...
2. 估算每个任务的成本 E_i
3. 直接成本 $=E_1+E_2+.....+ E_i+.....+ E_n$
4. 项目总估算成本= 直接成本+间接成本
5. 项目总报价=项目总估算成本+风险利润
 1. 风险利润=利润+风险基金+税

估算每个任务的成本

- ❑ 直接估算成本 E_i
- ❑ 先估算规模 Q_i ，然后估算成本 $E_i = Q_i * \text{人力成本参数}$
 - ❑ 唯一估计值： $Q_i = \text{Avg}$
 - ❑ PERT算法： $Q_i = (\text{Max} + 4\text{Avg} + \text{Min}) / 6$



直接成本估算

- ❑ 直接成本=规模*人力成本参数
 - ❑ 直接成本组成
 - ❑ 开发成本
 - ❑ 管理成本
 - ❑ 质量成本
-
- ❑ 例如：人力成本参数=2万/人月，30人月规模的项目的直接成本是 60万

直接成本估算—简易估算：

开发（工作量）规模：

Scale (Dev)（单位：人月）

管理、质量（工作量）规模：

Scale (Mgn) = $a * \text{Scale (Dev)}$

[a为比例系数：例如：20%—25%]

直接成本 = $\text{Scale (Dev)} + a * \text{Scale (Dev)}$



项目总估算成本

估算成本=直接成本+间接成本

间接成本估算：

1. 按照企业模型直接估算：
2. 简易算法：
 - 间接成本=直接成本*间接成本系数
 - 间接成本= 规模*人力成本参数*间接成本系数
 - 例如：间接成本系数=0.3

项目总估算成本

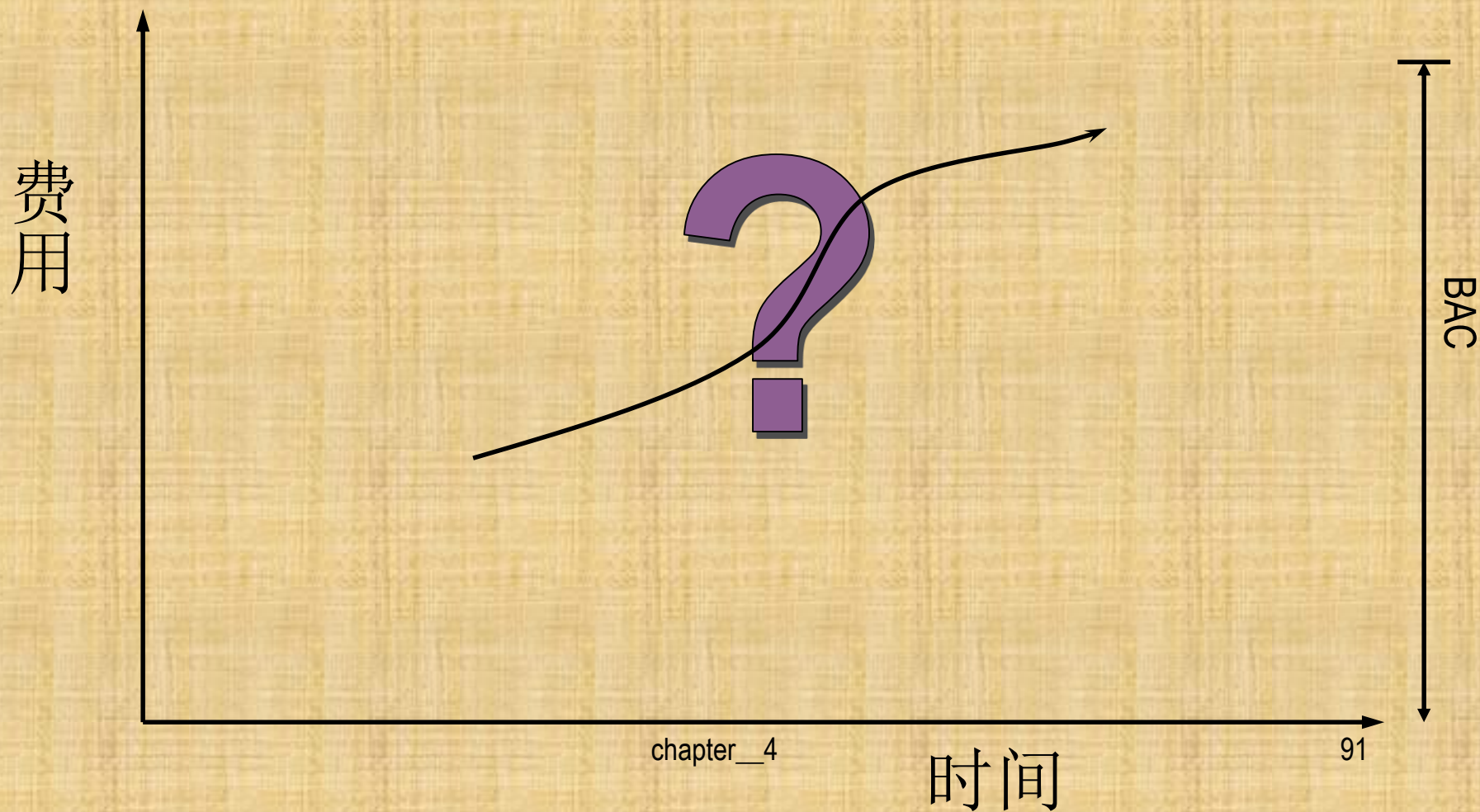
- ❑ 估算成本=直接成本+间接成本
 - ❑ 估算成本=直接成本+直接成本*间接成本系数
 - ❑ 估算成本=直接成本(1+间接成本系数)
 - ❑ 估算成本=规模*人力成本参数 (1+间接成本系数)
 - ❑ 成本系数=人力成本参数* (1+间接成本系数)
-
- ❑ 简易算法:
 - ❑ 估算成本=规模*成本系数
 - ❑ 例如：成本系数= 3万/人月



项目总报价

1. **项目总报价=项目总估算成本+风险利润**
 1. 项目利润=估算成本*a%
 2. 风险基金=估算成本*b%
 3. 税=估算成本*c% (例如:c为5.5左右)
2. **项目总报价=(a+b+c) %*项目总估算成本+项目总估算成本**

总估算成本（BAC）



估算准确度

类型	准确度	说明
量级估算:合同前 Order of magnitude	-25~~+75%	概念和启动阶段 决策
预算估算:合同期 Budget	-10~~+25%	编制初步计划
确定性估算:WBS后 Definitive	-5~~+10%	工作分解后的详 细计划

估算不准的原因

- ❑ 基础数据不足
- ❑ 缺乏经验的估算人员
- ❑ 签约前后不连贯
- ❑ 低劣的推测技术
- ❑ 估算对需求的敏感性

避免低劣估算

1. 避免无准备的估算
2. 留出估算的时间，并做好计划
3. 使用以前的项目数据
4. 使用开发人员提供的数据为基础估算
5. 分类法估算
6. 详细的较低层次上的估算
7. 使用软件估算工具
8. 使用几种不同估算技术，并比较它们的结果

估算的表达方式技巧

1. 加减限定
 1. 6个人月, +3人月, -1人月
2. 范围
 1. 5-9人月

估算的表达方式技巧

3. 风险量化：见下例

估算：6个人月，+3，-2

+1人月：延迟交付转换子系统

-1人月：新成员的工作效率高

+1人月：采用的新工具没有预计的好

-1人月：采用的新工具比预计的好

+0.5人月：员工病事假

+0.5人月：低估规模

估算的表达方式技巧

n 4. 情况：

见下：案例估算的例子

情况	估算
最佳情况	100人月
计划情况	110人月
最差情况	150人月

《软件工程中的真相与假象》

- 真相8:造成软件项目失控最普遍的两个原因之一就是软件估算不足(另外的原因是需求不稳定).
- 真相9:往往在错误的时间,甚至在还没有定义需求也即没有理解问题之前,就进行软件估算.
- 真相10:往往由错误的人员,即不是由软件开发人员或者项目经理,而是由高层管理或者市场营销,进行软件估算.
- 真相11:由错误的人员、在错误的时间做出的软件估算不但往往是错误的,也很少随着项目的进行做必要的调整.

《软件工程中的真相与假象》

- 真相12:虽然软件估算会发生如此错误,但是人们不去关心产生错误的原因,却还在努力按照错误估算的时间进度执行.
- 真相13:管理人员和技术人员对软件估算的认识是完全隔绝的,因此也难以达到一致的目标.
- 真相14:很少有真正的可行性研究.
- 假象6:要估算软件项目开发时间和开发成本,必须首先估算软件项目

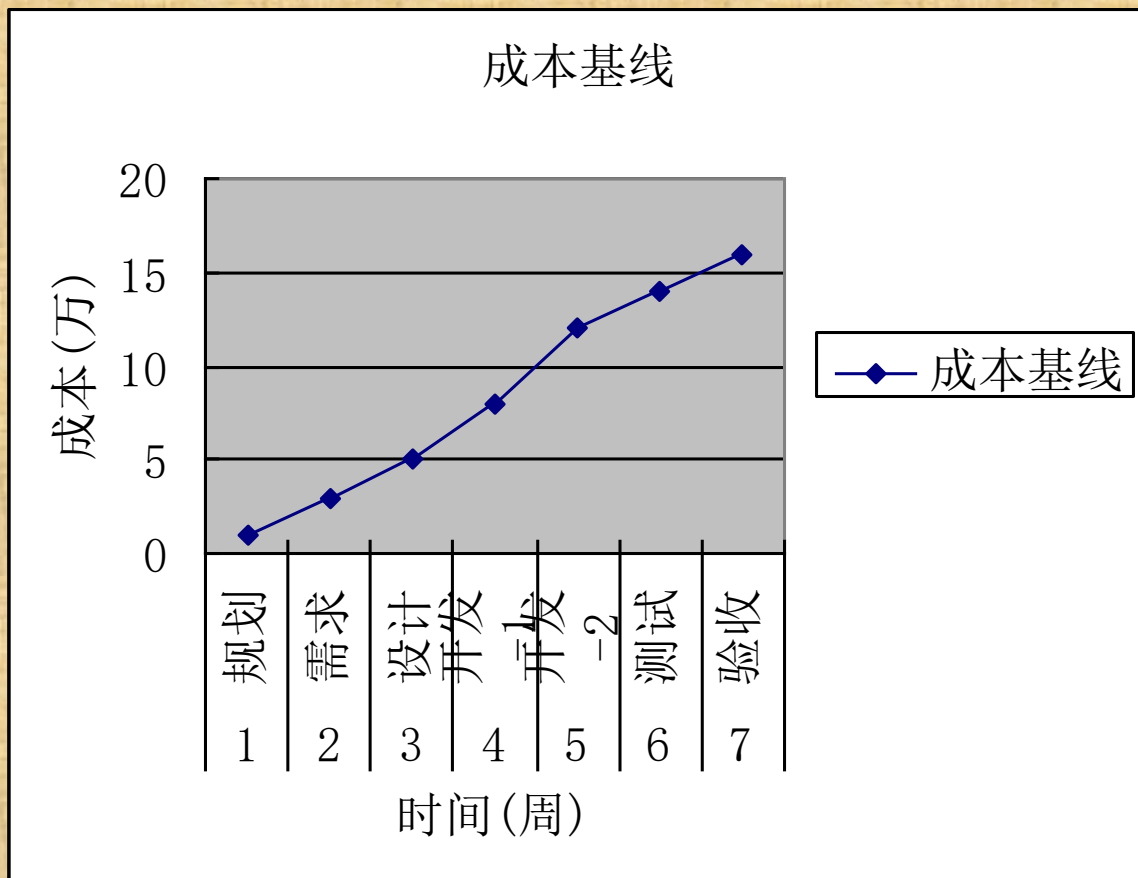
本章要点

- ❑ 一、软件项目规模成本的概念
- ❑ 二、成本估算过程
- ❑ 三、成本估算方法
- ❑ 四、成本预算
- ❑ 五、案例分析

成本预算

- 成本预算是将项目的总成本按照项目的进度分摊到各个工作单元中去。成本预算将总的成本安排到各个任务中
- 成本预算的目的是产生成本基线

成本基线



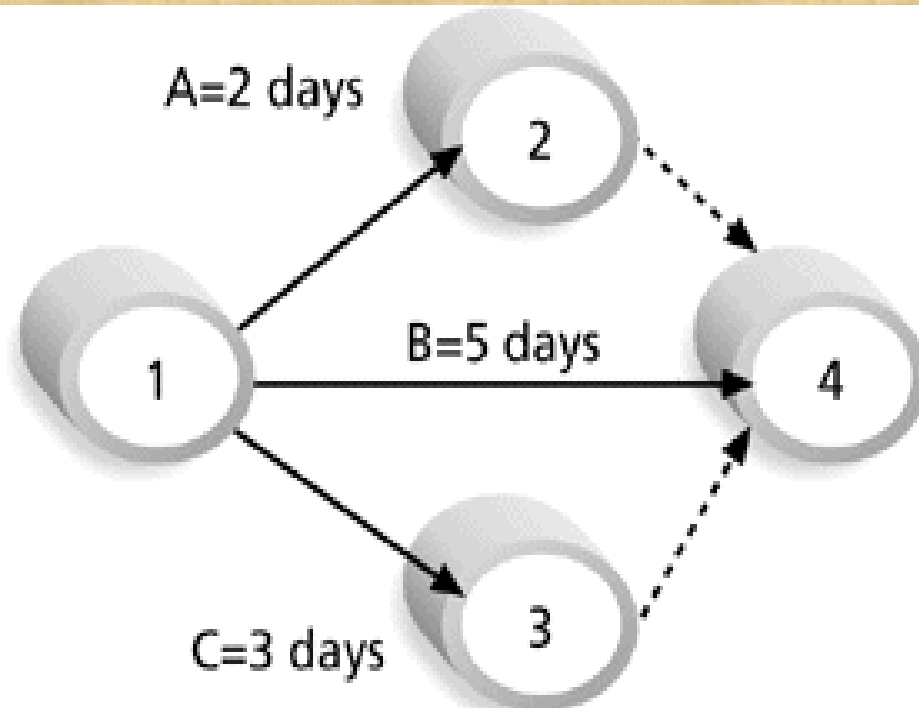
编制项目进度计划步骤

1. 进度编制
2. 资源调整
3. 成本预算
4. 计划优化调整
5. 计划基线

资源调整尝试法

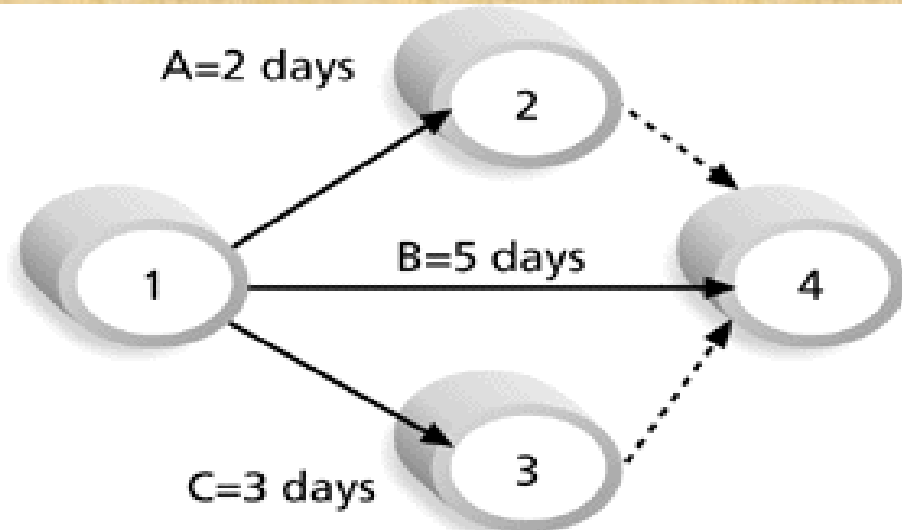
- ❑ 资源优化配置
- ❑ 通过调整进度计划，形成平稳连续的资源需求
 - ❑ 最有效的利用资源
 - ❑ 使资源闲置的时间最小化
 - ❑ 尽量避免超出资源能力
- ❑ 方法
 - ❑ 维持工期不变，资源平衡，使资源强度尽可能平衡
 - ❑ 使工期最短，在满足资源约束条件下

资源平衡法

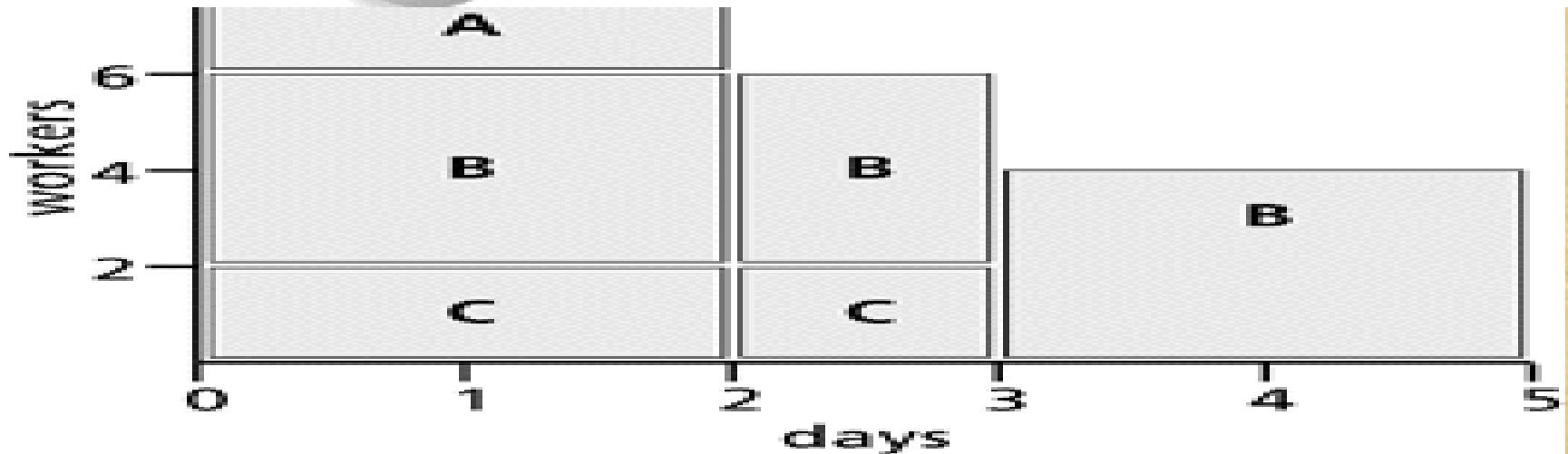


The project network with Activities A, B, and C and durations as shown. Activity A has 3 days of slack, and Activity C has 2 days of slack. Assume Activity A has 2 workers, B has 4 workers, and C has 2 workers.

资源平衡法

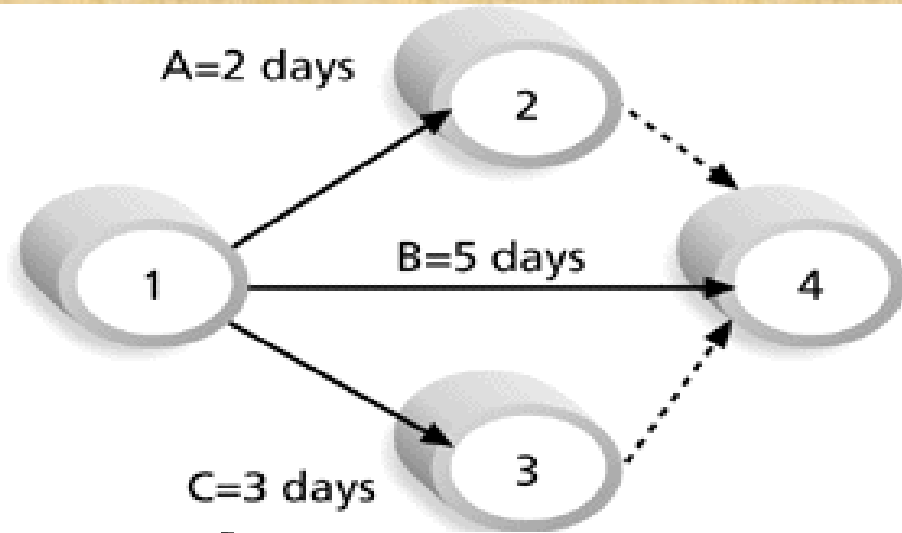


The project network with Activities A, B, and C and durations as shown. Activity A has 3 days of slack, and Activity C has 2 days of slack. Assume Activity A has 2 workers, B has 4 workers, and C has 2 workers.

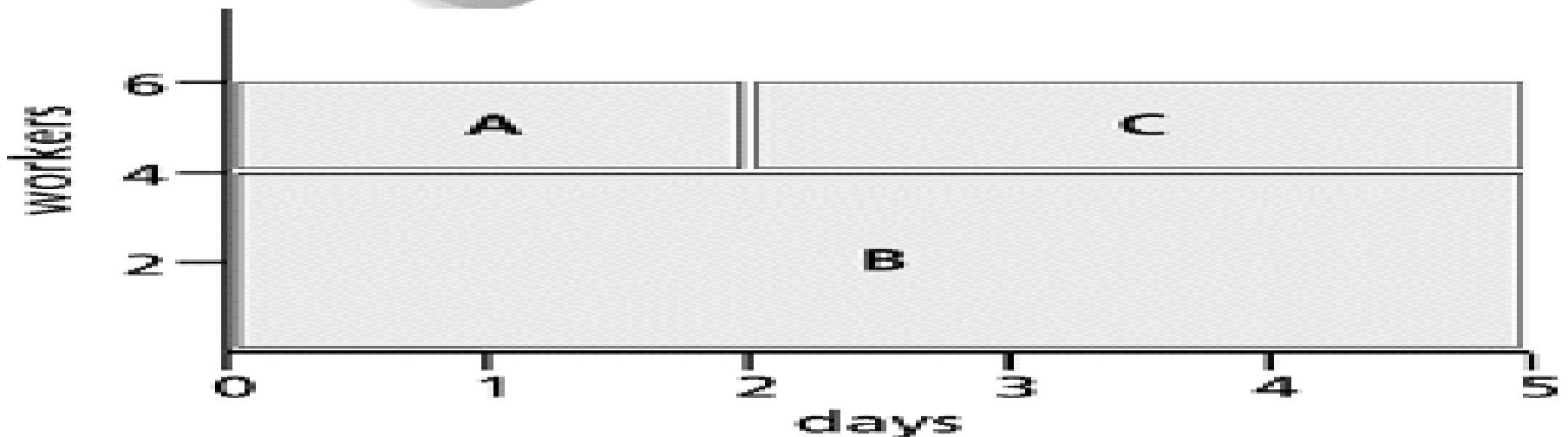


Resource usage if all Activities start on day one

资源平衡法



The project network with Activities A, B, and C and durations as shown. Activity A has 3 days of slack, and Activity C has 2 days of slack. Assume Activity A has 2 workers, B has 4 workers, and C has 2 workers.



Resource usage if Activity C is delayed 2 days, its total slack

编制项目进度计划步骤

1. 进度编制
2. 资源调整
3. 成本预算
4. 计划优化调整
5. 计划基线

项目成本预算

分配项目成本（预算）包括三种情况：

1. 分配资源成本
2. 给任务分配固定资源成本
3. 给任务分配固定成本

分配资源成本

- ❑ 资源成本与资源的基本费率紧密相连
- ❑ 设置资源费率
 - ❑ 标准费率
 - ❑ 加班费率
 - ❑ 每次使用费率
 - ❑ 。 。 。 。 。 。

分配固定资源成本

- ❑ 当一个项目的资源需要固定数量的资金时，用户可以向任务分配固定资源成本。
- ❑ 例如：需要的硬件设备

分配固定成本

- ❑ 有些任务是固定成本的类型的任务，也就是说，用户知道某项任务的成本不变，不管任务的工期有多长，或不管任务使用了那些资源。在这种情况下，用户向任务直接分配成本。
- ❑ 例如：培训任务

编制项目进度计划步骤

1. 进度编制
2. 资源调整
3. 成本预算
4. 计划优化调整
5. 计划基线

计划优化调整

1. 调整资源, 解决资源冲突
2. 调整进度, 优化项目, 缩短工期
3. 调整项目成本预算, 以便减少项目费用.

调整资源, 解决资源冲突

资源冲突(过度分配)主要有两种表现:

- 1、分配给一个资源的工时总量大于它的最大可用工时量。
- 2、同一种资源被分配给时间上重叠的几个任务或项目中。

解决资源冲突的方法

- ❑ 资源调配
- ❑ 推迟资源开始工作时间
- ❑ 替换资源
- ❑ 设置资源加班时间
- ❑ 调整资源日历
- ❑ 只使用资源的一部分工作时间

优化进度, 缩短工期

- ❑ 项目中各任务的执行时间是否合理, 有无冲突现象
- ❑ 尽可能缩短项目工期

优化进度, 缩短工期

1. 分解关键任务
2. 给任务增加资源
3. 缩减关键任务的工期
4. 重叠关键任务
5. 设置日历增加工作时间
6. 通过分配加班工时来缩短关键任务

缩减项目工期

1、分解关键任务

注意：通过“分解关键任务”可以缩短任务工期，但有时候，受资源量的限制，有些任务是不能同步进行的，所以这时任务分拆也无助于缩短项目周期。

缩减项目工期

2、给任务增加资源

注意：

- 增加的资源数量不能大于资源的最大可用量。
- 增加资源必须是主导项目工期的关键路径上。
- 关键任务的缩短可能会变成非关键任务，因此，此时增加过多的资源是无法达到继续缩短总工期的目的。

缩减项目工期

3、缩减关键任务的工期

注意：在任务已分配了资源的情况下，缩短任务工期意味着增加资源的工作量，可能导致资源的过度分配。

缩减项目工期

4、重叠或延迟链接任务

方法有两种：

- 改变任务的链接关系
- 在链接任务之间增加负延迟

缩减项目工期

5、设置日历增加工作时间

可以通过改变资源的日历来调整工期，比如将资源原来的休息时间改变成工作来实现。这样通过增加资源的工作时间来缩短任务的工期。

缩减项目工期

6、通过分配加班工时来缩短关键任务

需要在关键任务上为资源设置加班时间，以缩短任务工期。

调整项目成本预算

降低预算成本的方法：

1. 降低资源的费率
2. 减少任务的工时
3. 减少加班
4. 替换资源
5. 减少任务的固定成本
6. 删除任务

减少项目成本

1. 降低资源的费率

降低资源的费率往往会打击工作人员的积极性，但可以通过降低其他资源的费率来实现，比如降低能源消耗、设备费用等。

2. 减少任务的工时

适当的减少工时，可以降低任务的费用。但减少工时同时也影响项目的工期。

减少项目成本

3. 减少加班

加班需要支付加班费率，这通常要高于资源费率，所以减少加班可以有效的减少任务成本。

减少项目成本

4. 替换资源

用廉价的资源替换比较高价的资源，但有一个前提，那就是替换的资源同样能胜任这项任务。

5. 减少任务的固定成本

固定成本就是任务本身所需要的成本。

减少项目成本

6. 删除任务

确认删除改任务对项目没有影响或影响在可控范围内才可采用。

Most likely forgotten areas in development planning?

☐ Establish Development Environment

- ☐ Platform

- ☐ Machine allocation

- ☐ Software license

- ☐ Installation and setup

☐ Establish build schedule

- ☐ daily build

- ☐ weekly build

- ☐ development build

- ☐ test build

☐ Quality management

☐ Configuration management

Most likely forgotten areas in development planning?

☐ Install

- ☐ Installation design, coding and testing
- ☐ Multiplatform consideration

☐ Documentation Plan

- ☐ How many books?
 - ☐ getting started
 - ☐ administration guide
 - ☐ application and programming reference
 - ☐ installation guide
 - ☐ release note
 - ☐ readme
- ☐ soft-copy? hard-copy? other?

最后审查

❑ 角色

- ❑ 是否每个任务都有完整的角色，如果需要就增加角色

❑ 人员

- ❑ 这些角色都做什么？必要的分配任务

❑ 技能

- ❑ 分配任务的人是否具备应有的能力

❑ 可行性

- ❑ 什么时候真的需要这些人

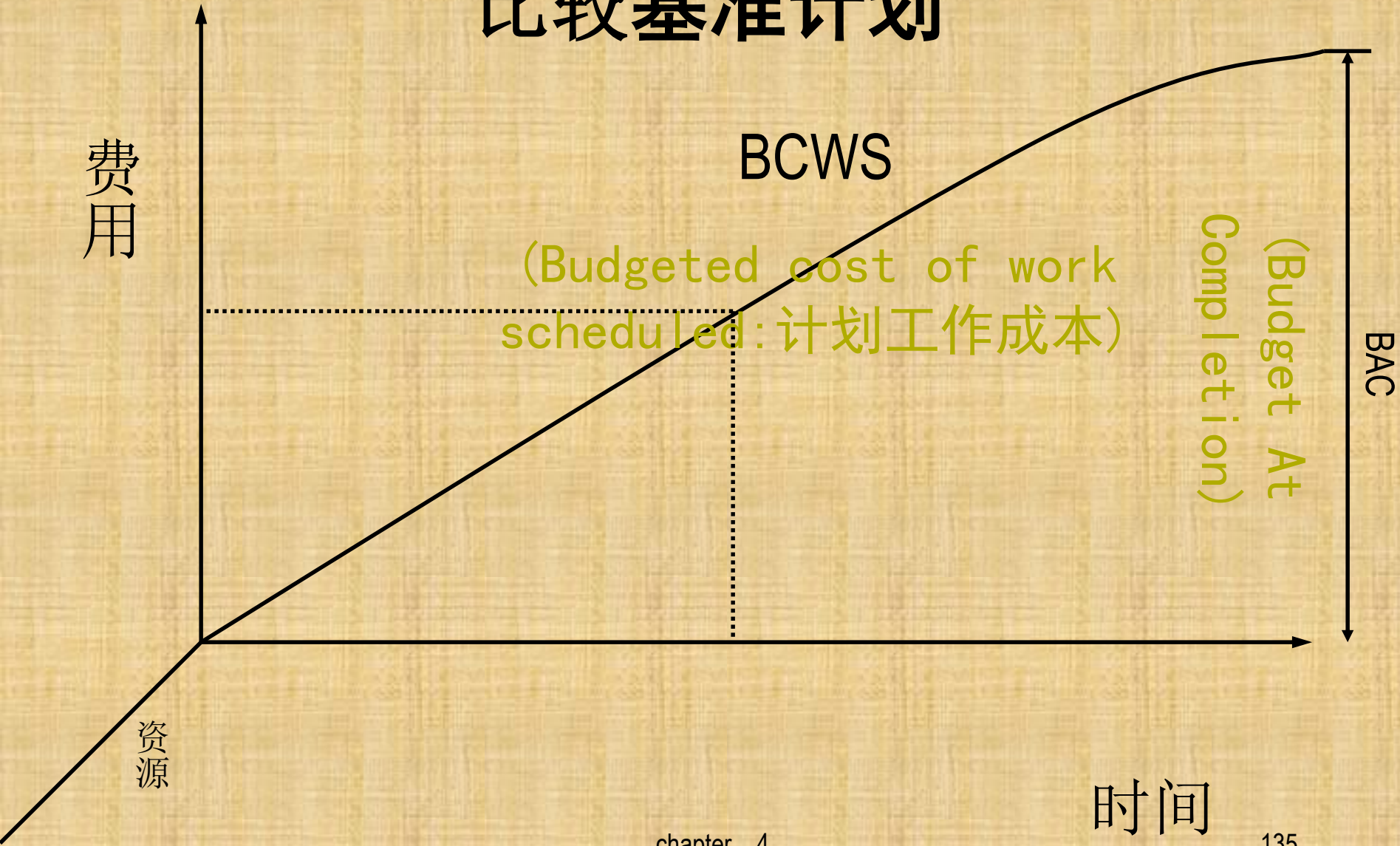
项目计划的渐进性



编制项目进度计划步骤

1. 进度编制
2. 资源调整
3. 成本预算
4. 计划优化调整
5. 计划基线

比较基准计划



本章要点

- ❑ 一、软件项目规模成本的概念
- ❑ 二、成本估算过程
- ❑ 三、成本估算方法
- ❑ 四、成本预算
- ❑ 五、案例分析

案例说明

School 项目估算

▣ 项目估算结果

小结

- ❑ 成本估算的过程
- ❑ 成本估算的方法
- ❑ 掌握成本估算的技巧

项目的总成本估算

核心计划小结

