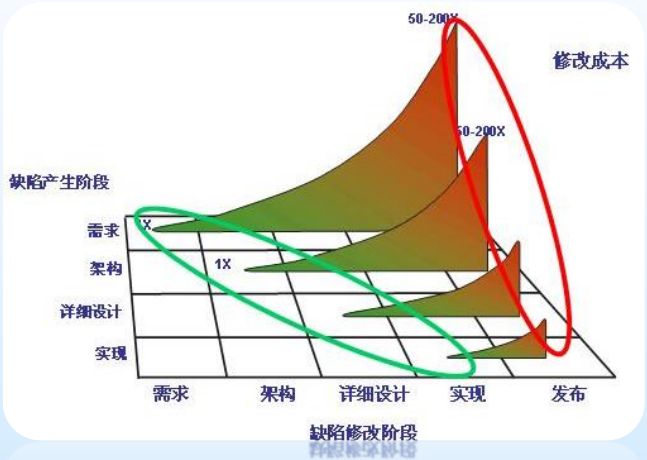
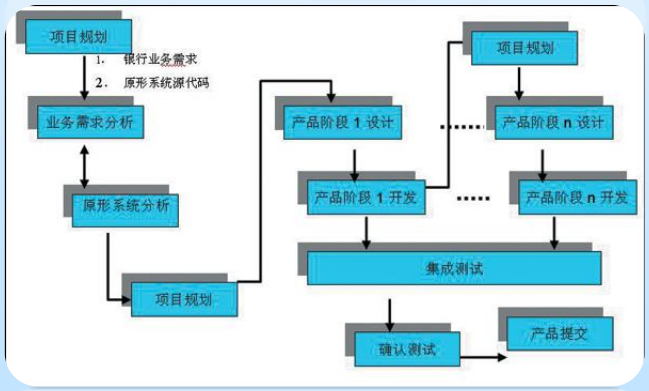
软件项目管理





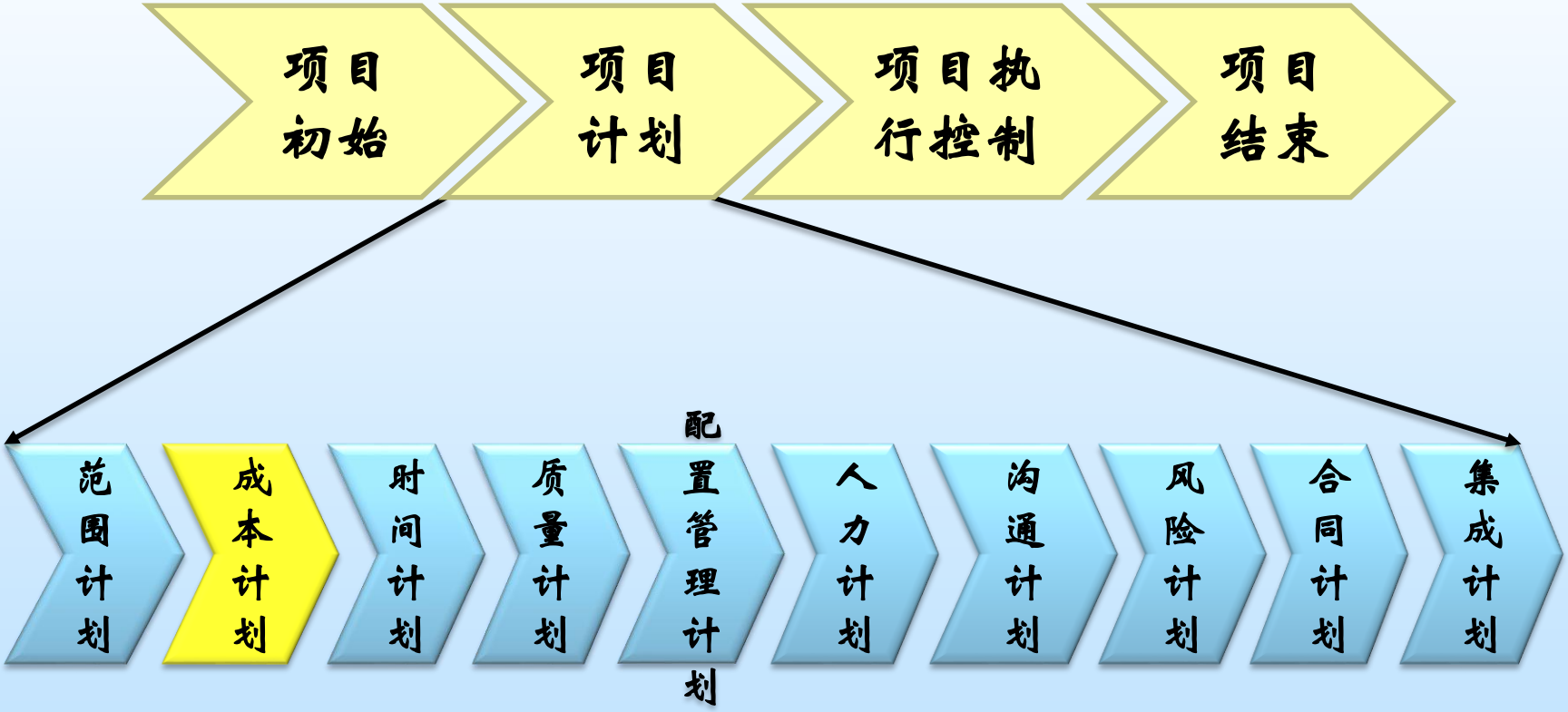
中国科学技术大学

信息科学技术学院自动化系

王子磊

zlwang@ustc.edu.cn

RoadMap


项目初始
项目计划
项目执行控制
项目结束
范围计划
成本计划
时间计划
质量计划
配置管理计划
人力计划
沟通计划
风险计划
合同计划
集成计划

软件项目管理

第 5 章

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

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
  - 从软件项目范围中抽出的软件功能
  - 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
  - ……

软件的规模和成本的关系

- 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

成本管理过程

- 成本管理计划
  - 规划软件项目成本管理的总体内容
- 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

本章要点

一、软件项目规模成本的概念

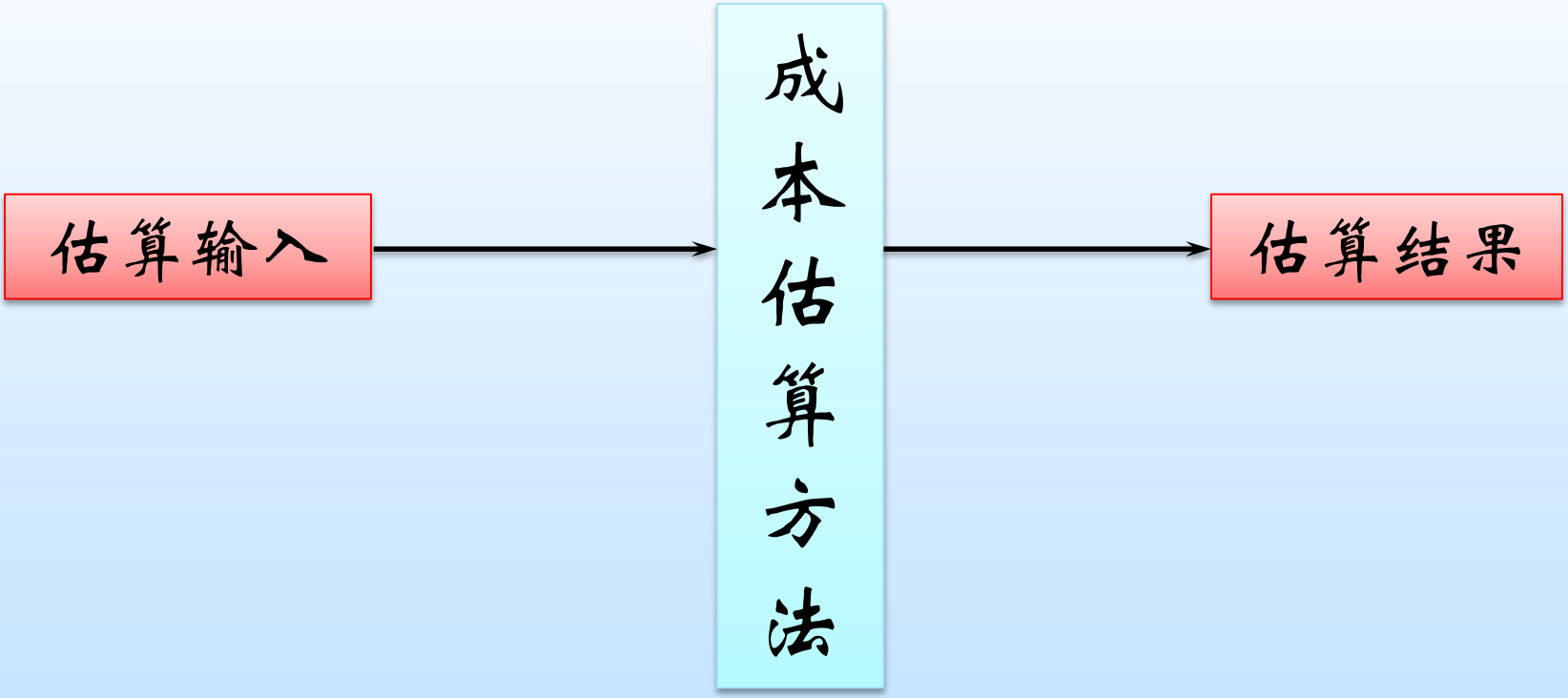
二、成本估算过程

三、成本估算方法

四、成本预算

五、案例分析

成本估算过程

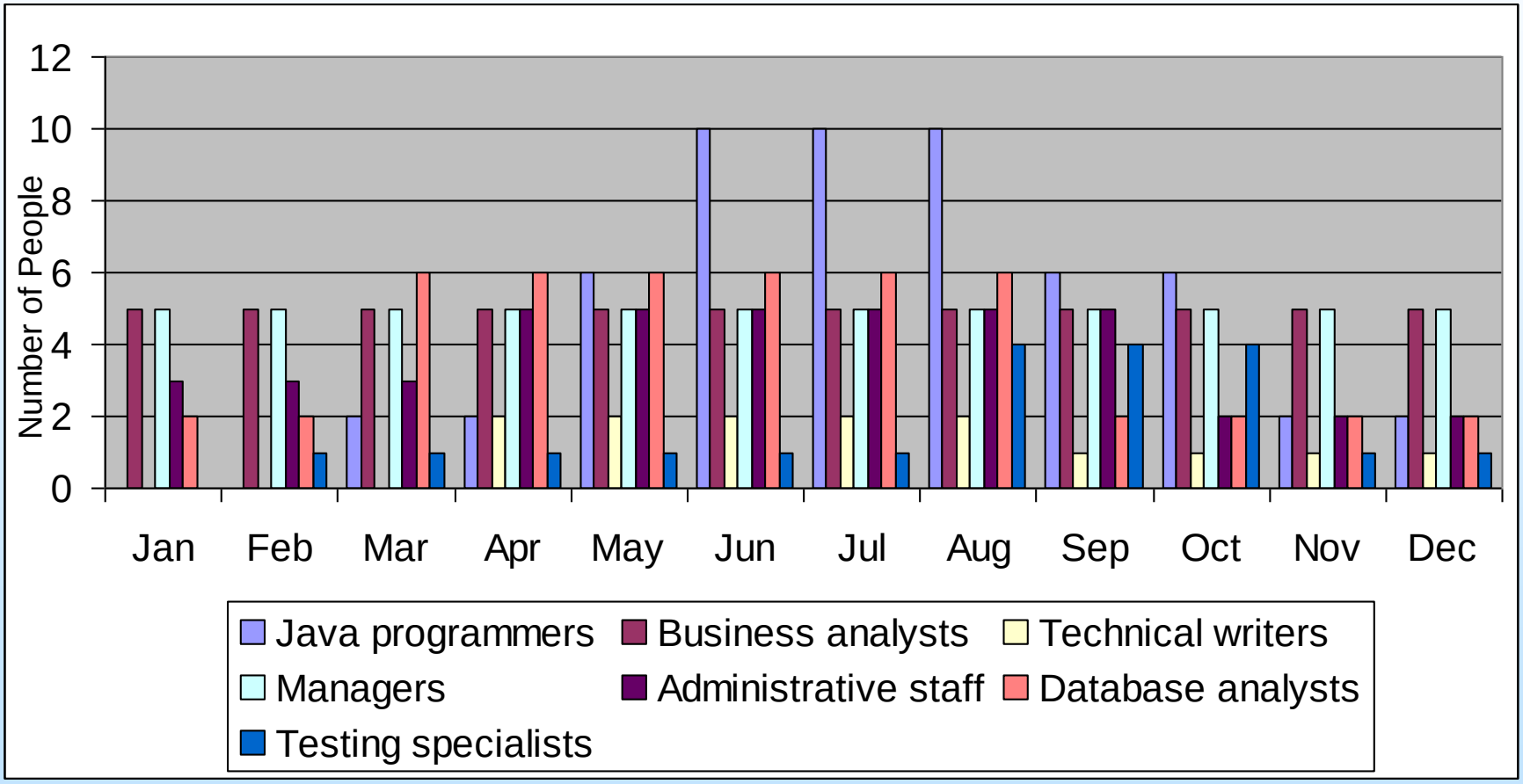


成本估算输入

- 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



成本估算

- 直接成本
  - 与具体项目相关的成本
- 间接成本

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

代码行技术的主要优点

- 代码是所有软件开发项目都有的“产品”
- 且很容易计算代码行数

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，只是项目实现阶段的一部分

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

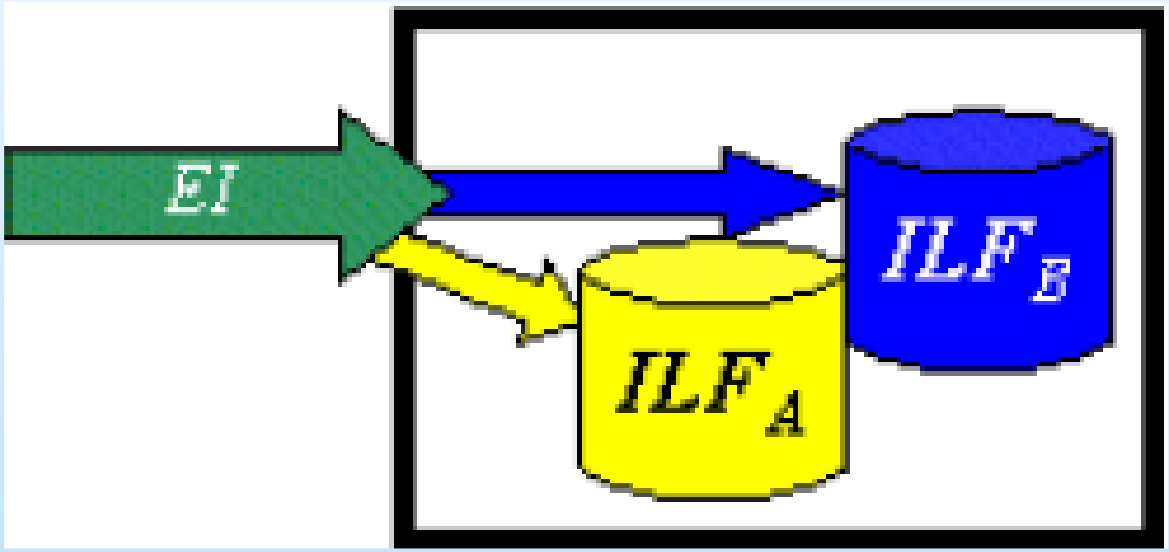
UFC-未调整功能点计数

功能计数项：

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

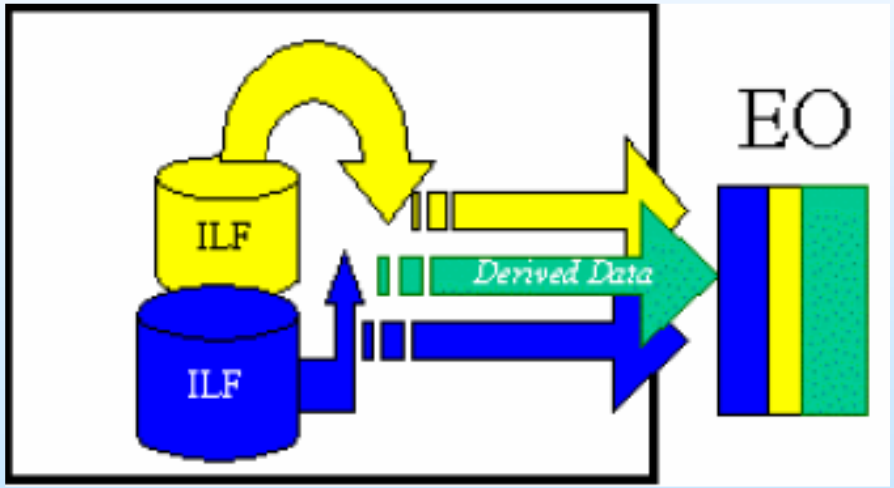
外部输入 (External Inputs, EI)

- 给软件提供面向应用的数据的项 (如屏幕、表单、对话框、控件，文件等)
- 在这个过程中，数据穿越外部边界进入到系统内部



外部输出 (External Outputs, EO)

- 向用户提供(经过处理的)面向应用的信息
  - 例如，报表和出错信息等

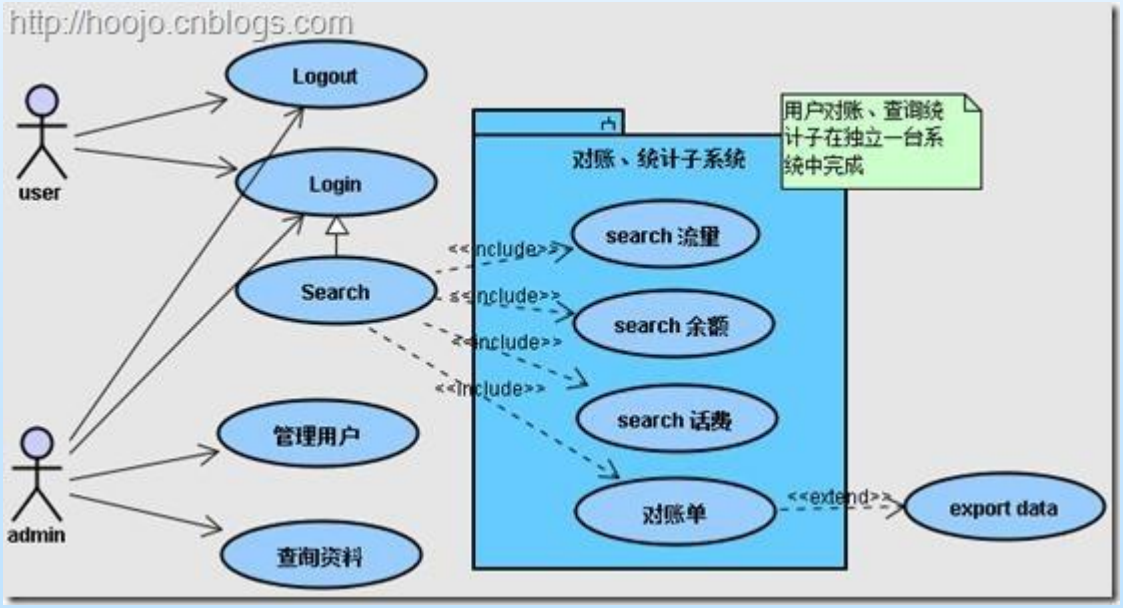


外部查询 (External Inquiry, EQ)

- 外部查询即是一次联机输入，它导致软件以联机输出方式产生某种即时响应

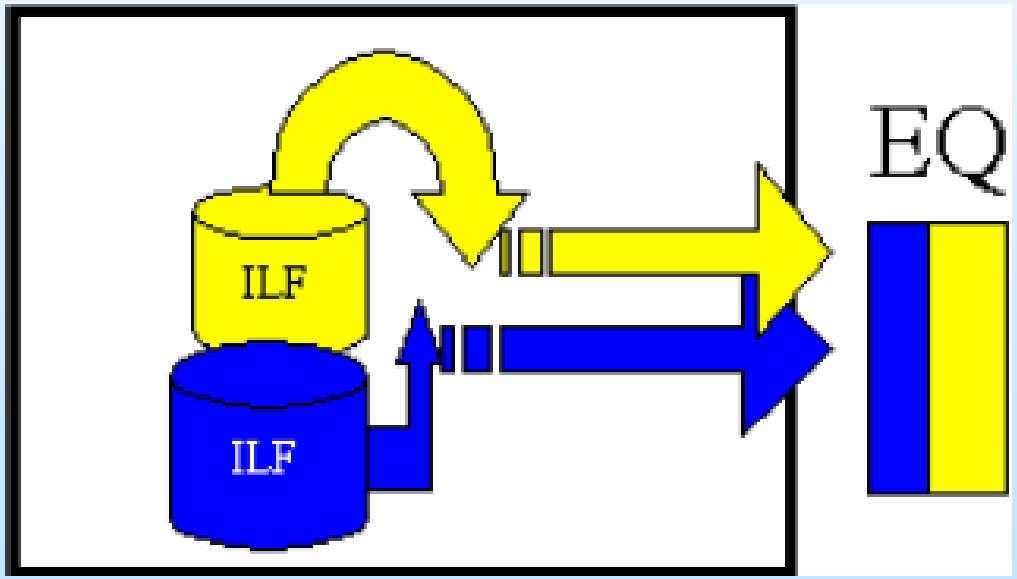
外部接口文件 (External Interface Files, EIF's)

- 外部接口文件是用户可以识别的一组逻辑相关数据，这组数据只能被引用
  - 是机器可读的全部接口 (例如，磁盘或磁带上的数据文件)的数量
  - 用这些接口把信息传送给另一个系统

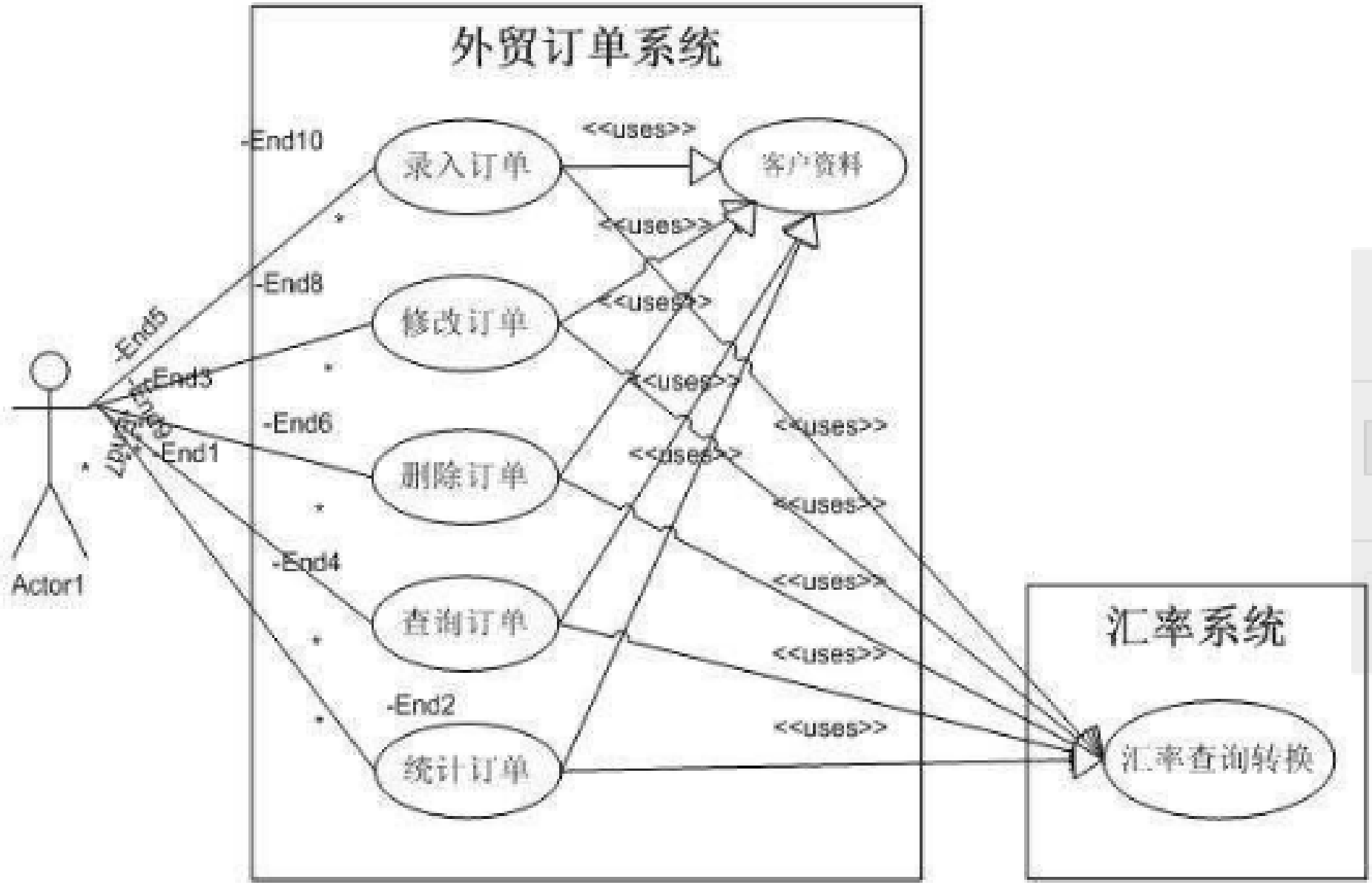


内部逻辑文件(Internal Logical Files, ILF'S)

- 用户可以识别的一组逻辑相关的数据，而且完全存在于应用的边界之内，并且通过外部输入维护，是逻辑主文件的数目



FP估算方法举例


外贸订单系统
录入订单
修改订单
删除订单
查询订单
统计订单
客户资料
汇率系统
汇率查询转换
Actor1
-End10
-End8
-End5
-End3
-End6
-End1
-End4
-End2
<<uses>>

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		

TCF-技术复杂度因子

TCF=0.65+0.01(sum(Fi))： Fi: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	6

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—理论举例

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

类比——主观判断举例

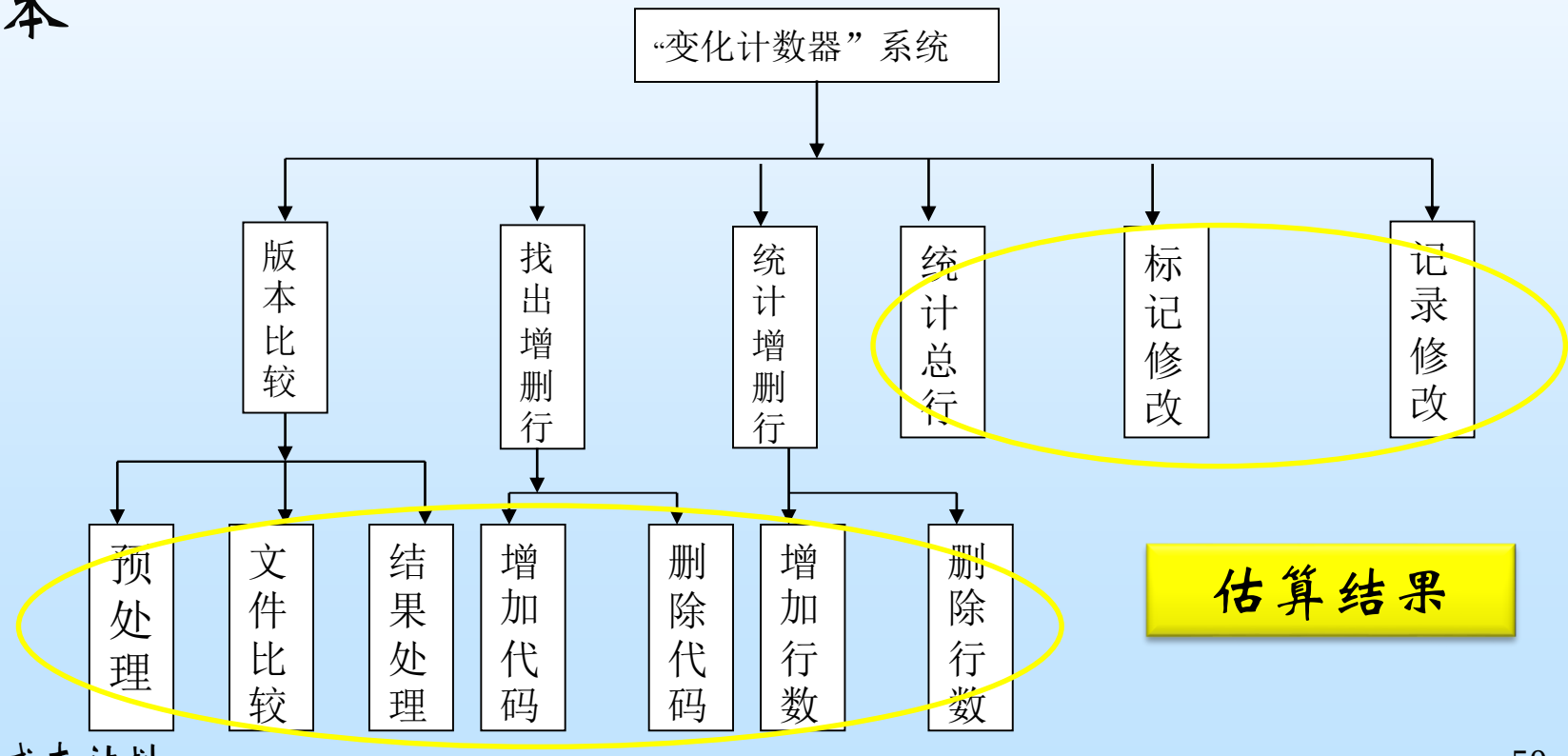
- 证券交易网站
  - 需求类似
  - 历史数据：10万
  - 类比估算：10万

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—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—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—特点

- 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

综合业务系统成本估算表如下：

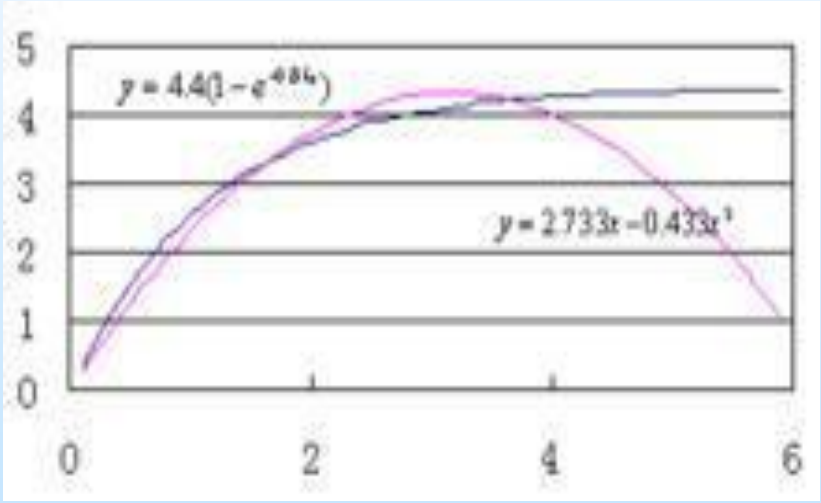
阶段	人力	时间（月）	成本（万）	总计（万）
项目准备阶段	M:2/D:8/Q:1	0.5	16.5	
设计阶段	M:2/D:8/Q:2/ S:1	1	39	
基础模块开发：				
公共控制子系统		1.5	90	
中央会计子系统	M:2/D:15/Q:2/ S:1			
客户信息子系统				145.5
基本功能模块开发：				
帐户管理子系统	M:2/D:12/Q:2/ S:1	0.25	12.75	
出纳管理子系统	M:2/D:18/Q:2/ S:1	0.5	34.5	
凭证管理子系统	M:2/D:8/Q:2/ S:1	0.25	9.75	126
会计核算子系统	M:2/D:18/Q:2/ S:1	0.5	34.5	
储蓄子系统	M:2/D:18/Q:2/ S:1	0.5	34.5	
扩展功能模块开发：				
同城业务子系统	M:2/D:15/Q:2/ S:1	0.25	15	
联行业务子系统	M:2/D:18/Q:2/ S:1	0.5	34.5	
内部清算子系统	M:2/D:12/Q:2/ S:1	0.25	12.75	
固定资产管理子系统	M:2/D:10/Q:2/ S:1	0.25	11.25	
信贷管理子系统	M:2/D:18/Q:2/ S:1	0.5	34.5	189.75
一卡通业务子系统	M:2/D:18/Q:2/ S:1	0.5	34.5	
中间业务子系统	M:2/D:12/Q:2/ S:1	0.25	12.75	
金卡接口模块	M:2/D:18/Q:2/ S:1	0.5	34.5	
现场联调	M:2/D:10/Q:2	1	42	42
	503.25			

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
- 参数估算法
- 专家估算法

参数估算法—定义

- 通过过去项目数据,进行回归分析,得出的回归模型
- 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



参数估算法——使用情况

- 存在成熟的项目估算模型
- 应该具有良好的项目数据为基础

参数估算法－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

参数模型：规模(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)

参数模型：规模(成本)模型（续）

面向LOC驱动的

- Walston-Felix (IBM)
  - E= 5.2*(KLOC)^0.91
- Balley-Basili
  - E=5.5+0.73*(KLOC)^1.16
- COCOMO
  - E=3.2*(KLOC)^1.05
- Doty
  - E=5.288*(KLOC)^1.047

参数模型：规模(成本)模型（续）

面向FP驱动的

- Albrecht and Gaffney
  - E=-12.39+0.0545FP
- Matson,Barnett
  - E=585.7+15.12FP

建议掌握模型

- IBM模型 – (Walston-Felix)
- COCOMO模型 – (Boehm)

IBM模型

1977年，IBM的Walston和Felix提出了如下的估算公式

- E = 5.2×L^0.91，L是源代码行数(以KLOC计)，E是工作量(以PM计)
- D = 4.1×L^0.36，D是项目持续时间(以月计)
- S = 0.54×E^0.6，S是人员需要量(以人计)
- DOC = 49×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 = 366×46 = 16386行 = 16.386KLOC
- E = 5.2×L^0.91 = 5.2×16.386^0.91 = 66人月
- DOC = 49×L^1.01 = 49×16.386^1.01 = 826页

COCOMO (Constructive Cost model)

- 结构化成本模型
- 是世界上应用最广泛的参数型软件成本估计模型
- 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基本原理

- 将开发所需要的工作量表示为软件规模和一系列成本因子的函数，基本估算公式:

$$PM = A \times S^{E} \times \prod_{i=1}^{n} EM_i$$

A: 可以校准的常量

S: 软件规模;

E: 为规模的指数,说明不同规模软件具有的相对规模经济和不经济性

EM: 为工作量乘数,反映某个项目特征对完成项目开发所需工作量的影响程度

n: 为描述软件项目特征的成本驱动因子的个数

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

COCOMO 81 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

COCOMO 81 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、软件和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-81

- $E = a\,(KLOC)^b$
- 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-81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：

- a=3.0，b=1.12
- E = 3.0 * L ^1.12 = 3.0 * 33.3 ^1.12 = 152 PM

中等COCOMO-81

- $E=a(KLOC)^{b}$*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-81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

乘法因子属性

1. 产品属性
2. 平台属性
3. 人员属性
4. 过程属性

乘法因子

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Fi的取值范围为：
  很低、低、正常、高、很高、极高，共六级
  正常情况下 Fi=1

- Boehm推荐的Fi取值范围
  0.70, 0.85, 1.00, 1.15, 1.30, 1.65
- 当每个Fi的值选定后，乘法因子的计算如下
  乘法因子 = F1*F2*… Fi …* Fn

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 ×1.09＝166 PM

高级(详细)COCOMO-81

- 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-81

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	--

COCOMO II

- 应用组装模型---规划阶段
- 早期设计模型---设计阶段
- 后体系结构模型---开发阶段

COCOMO II-后体系结构模型

- A: 可以校准，目前设定A=2.94
- B: 可以校准，目前设定B=0.91

$$PM = A \times S^E \times \prod_{i=1}^{17} EM_i$$

$$\text{其中 } E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

表 4　COCOMO II 比例因子

PREC 先例性	TEAM 团队凝聚力
FLEX 开发灵活性	PMAT 过程成熟度
RESL 体系结构与风险化解	

表 5　COCOMO II 后体系结构模型成本驱动因子

产品因子	人员因子
RELY, DATA, CPLX, RUSE(可复用开发), DOCU(匹配生命周期需求的文档)	ACAP, PCAP, APEX, PCON(人员连续性), PLEX(平台经验), LTEX (语言和工具经验)
项目因子	平台因子
TOOL, SCED, SITE(多点开发)	TIME, STOR, PVOL

模型研究例子——项目数据

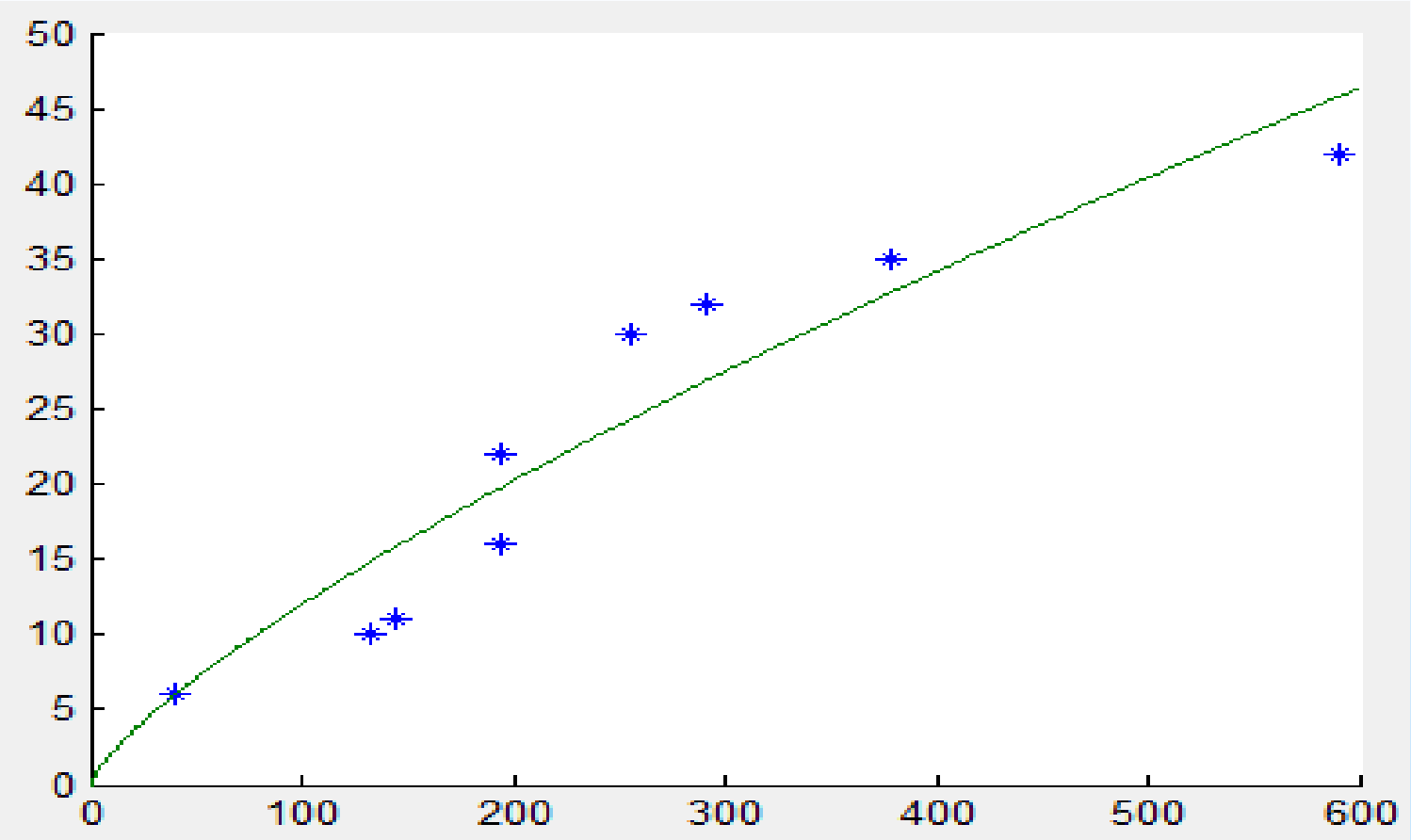
步骤:

xx=[41 132 144 194 194 291 255 378 591];

时间:

yy=[6,10,11,16,22,32,30,35,42];

模型研究例子——项目数据图式



模型研究例子--拟合算法

- 算法：

```
function n = cocomo(m)
    xx=[41 132 144 194 194 291 255 378 591];
    yy=[6,10,11,16,22,32,30,35,42];
    fun=@(c,x)[c(1) * x.^c(2)];
     abc0=[1 1];
     c= lsqcurvefit(fun,abc0,xx,yy);
     n=fun(c,m);
end
```

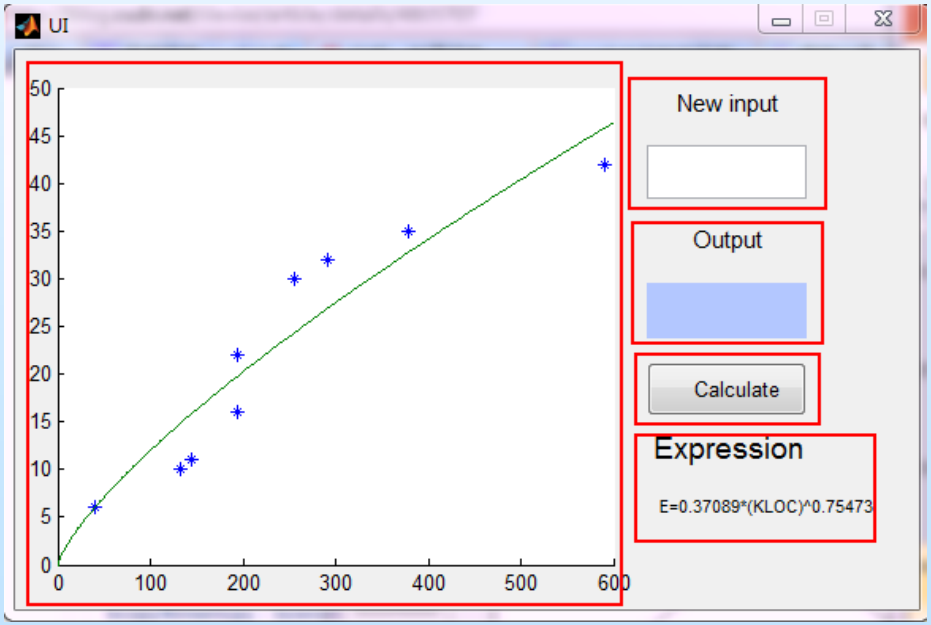
模型研究例子--结果输出

- 模型输出

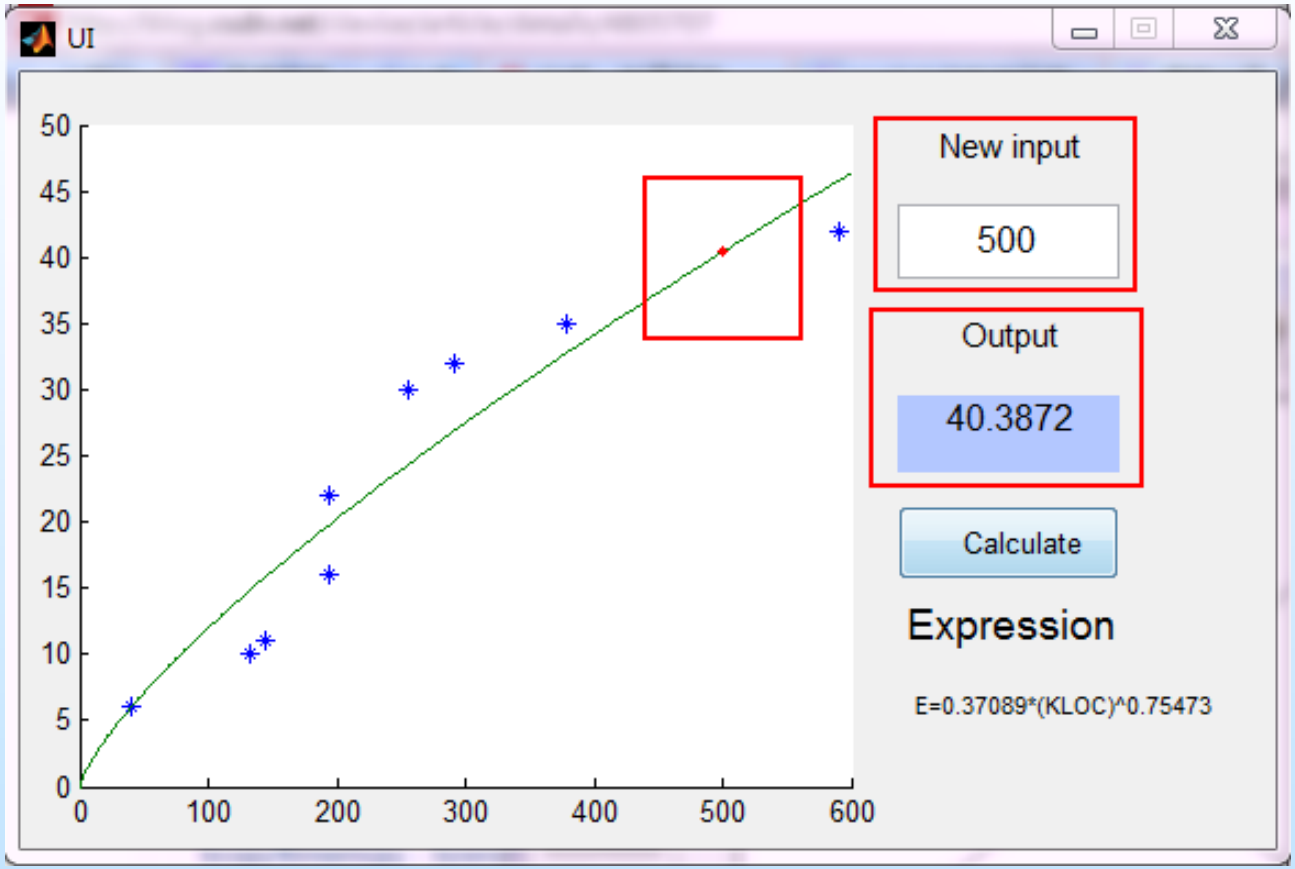
Expression

E=0.37089*(KLOC)^0.75473

- 图形输出



模型研究例子--模型应用


UI
50
45
40
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600
New input
500
Output
40.3872
Calculate
Expression
E=0.37089*(KLOC)^0.75473

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-Delphi

- 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 ai
  - 最可能的 mi
  - 最大 bi
- 组织者对专家的表格中的答复进行整理
- 计算每位专家的 Ei=(ai+4mi+bi)/6

专家估算法-Delphi (续)

- 综合结果后:E=E1+E2+…En/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 =› (1+9+4*8)/6=7 (万元)
  - 专家2： 4， 6 ， 8 =› (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WBS:1,2,…,i…
2. 估算每个任务的成本E_i
3. 直接成本=$E_1+E_2+\ldots\ldots+E_i+\ldots\ldots+E_n$
4. 间接成本估算
5. 项目总估算成本= 直接成本+间接成本
6. 项目总报价

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$E_i = Q_i$ *人力成本参数
  - 唯一估计值：$Q_i = Avg$
  - PERT算法: $Q_i = (Max + 4Avg + Min)/6$

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=5万/人月，30人月规模的项目的直接成本是150万

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* Scale(Dev)

[a为比例系数：例如：20%--25%]

直接成本= Scale(Dev) + a* Scale(Dev)

间接成本

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

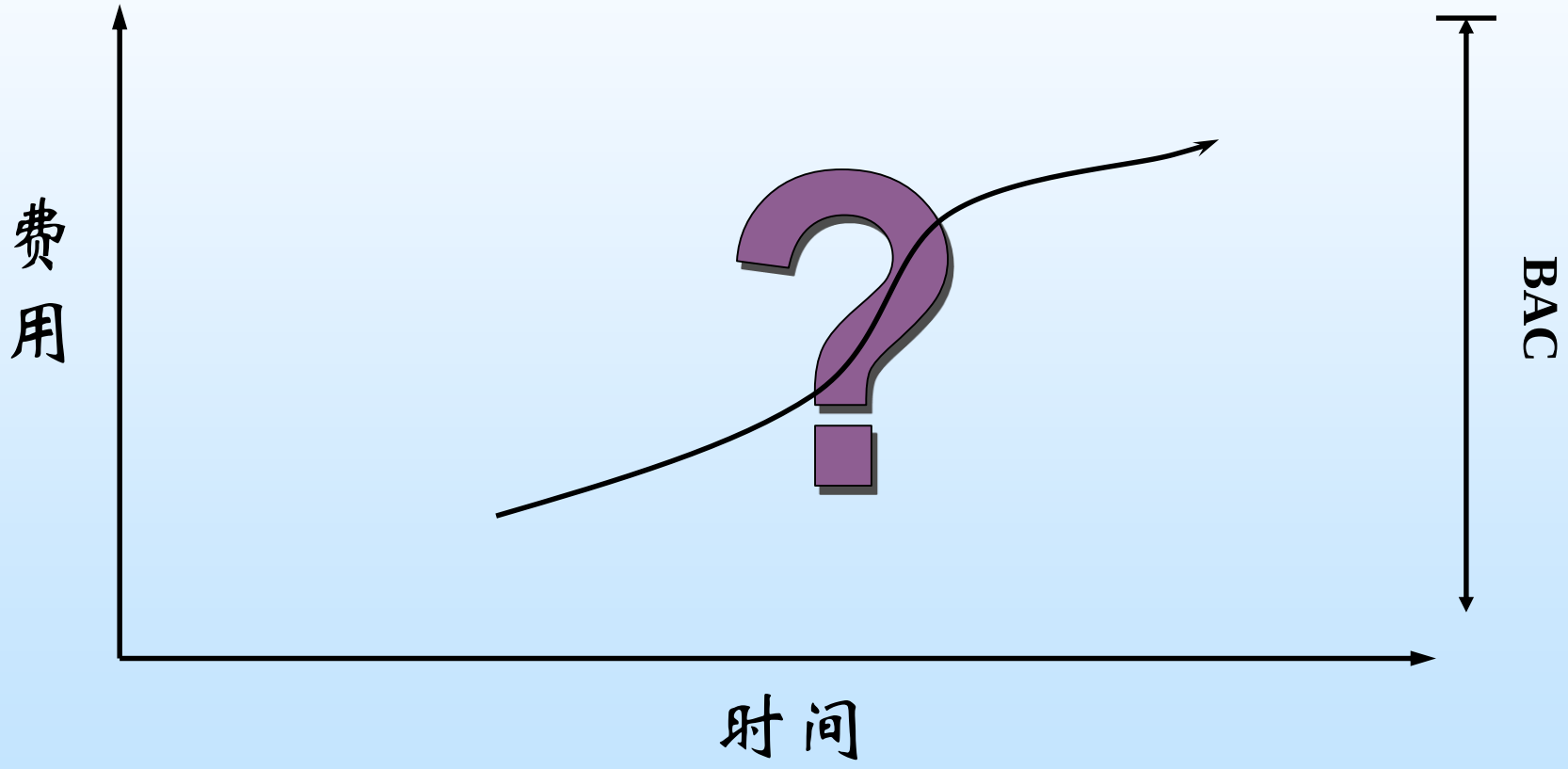
- 估算成本=直接成本+间接成本
- 估算成本=直接成本+直接成本*间接成本系数
- 估算成本=直接成本(1+间接成本系数)
- 估算成本=规模*人力成本参数（1+间接成本系数）
- 成本系数=人力成本参数*（1+间接成本系数）

- 简易算法：
  - 估算成本=规模*成本系数
  - 例如：成本系数=8万/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
3. 项目总报价=(1+(a+b+c)%)*项目总估算成本

总估算成本(BAC)



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

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
  - 。。。。。。

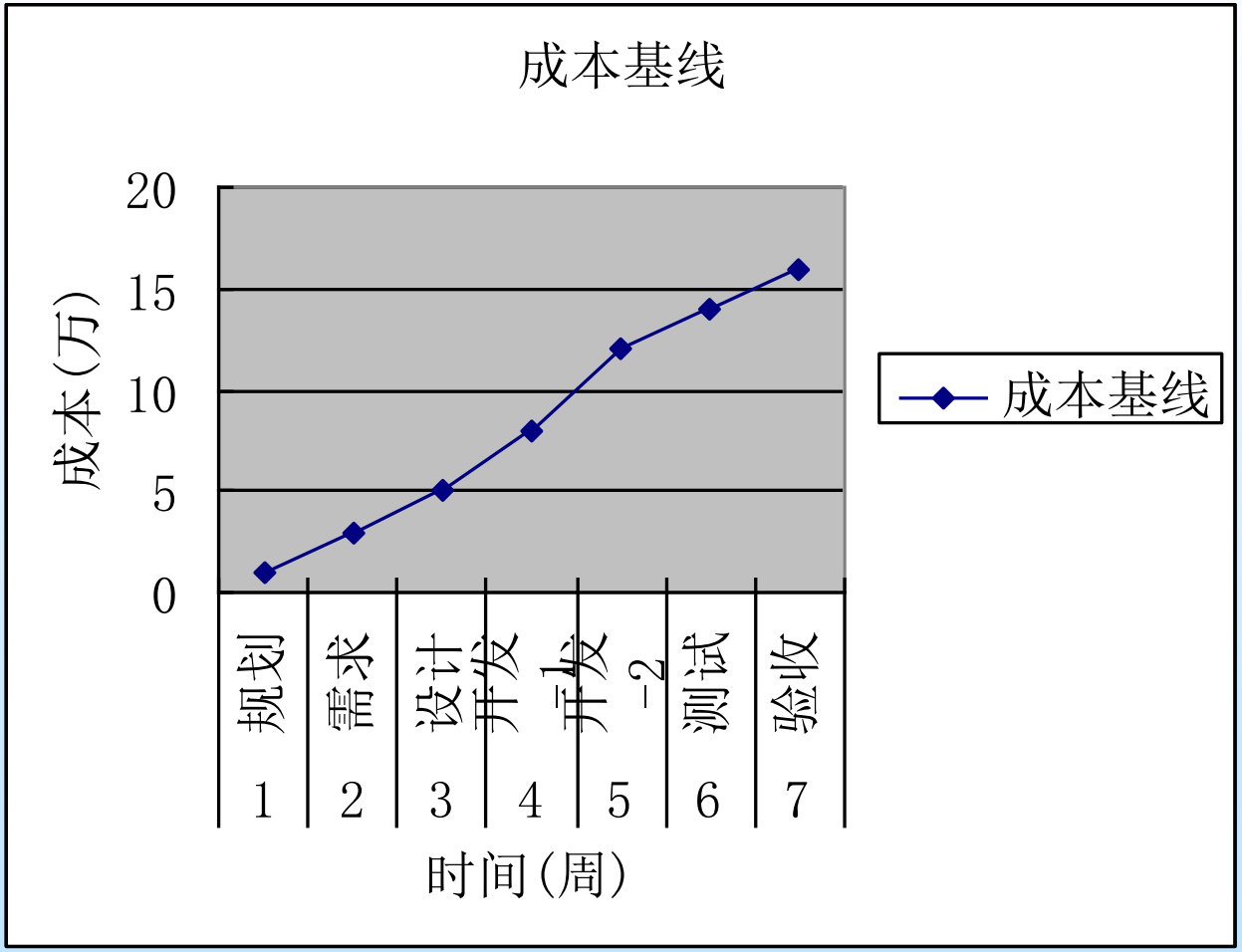
分配固定资源成本

- 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

成本基线



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
- 人为因素，如：签约前后不连贯
- 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

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

“校务通系统”项目成本估算

- 项目估算结果

小结

- 成本估算的过程
- 成本估算的方法
- 成本预算的方法

项目的总成本估算