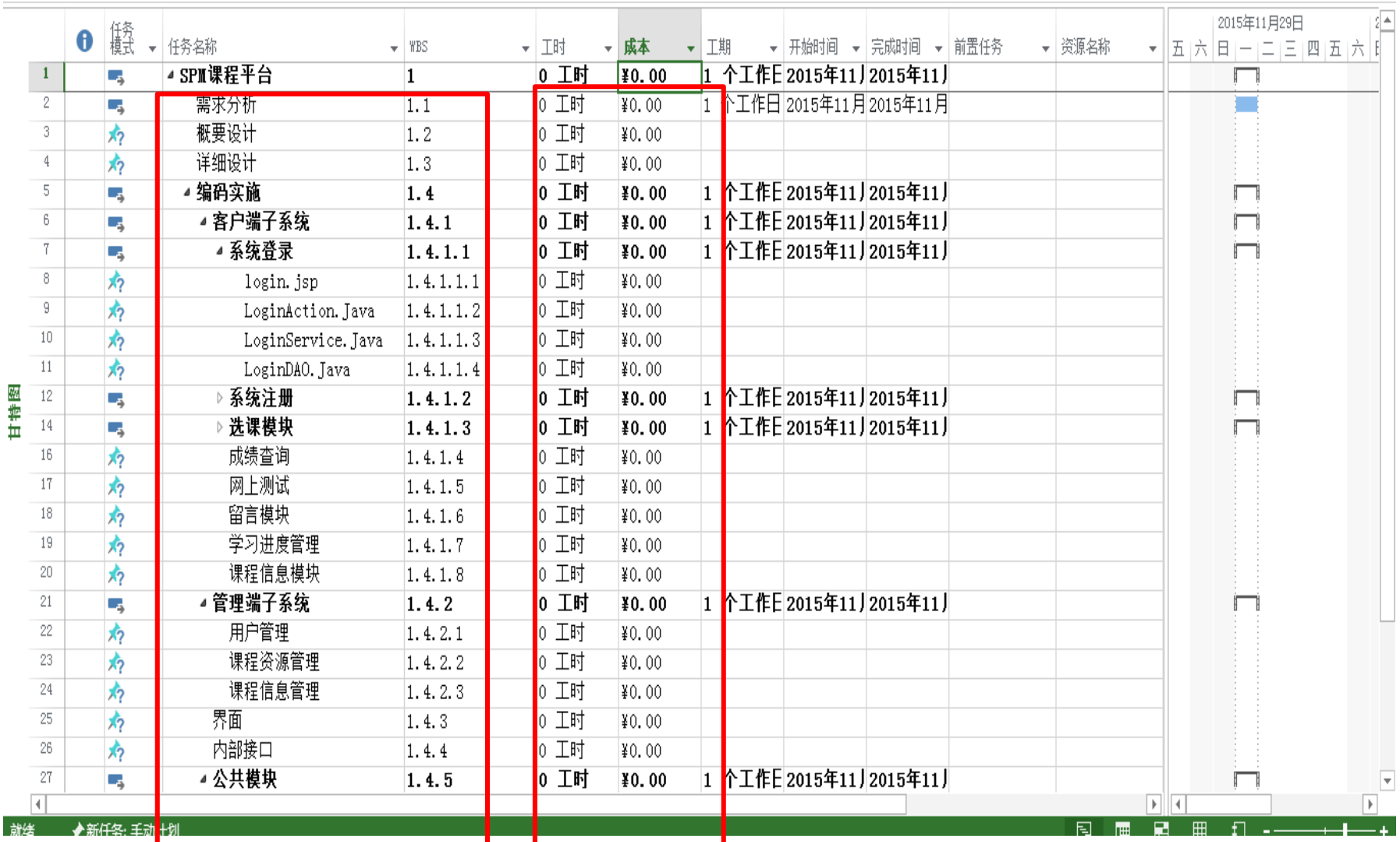
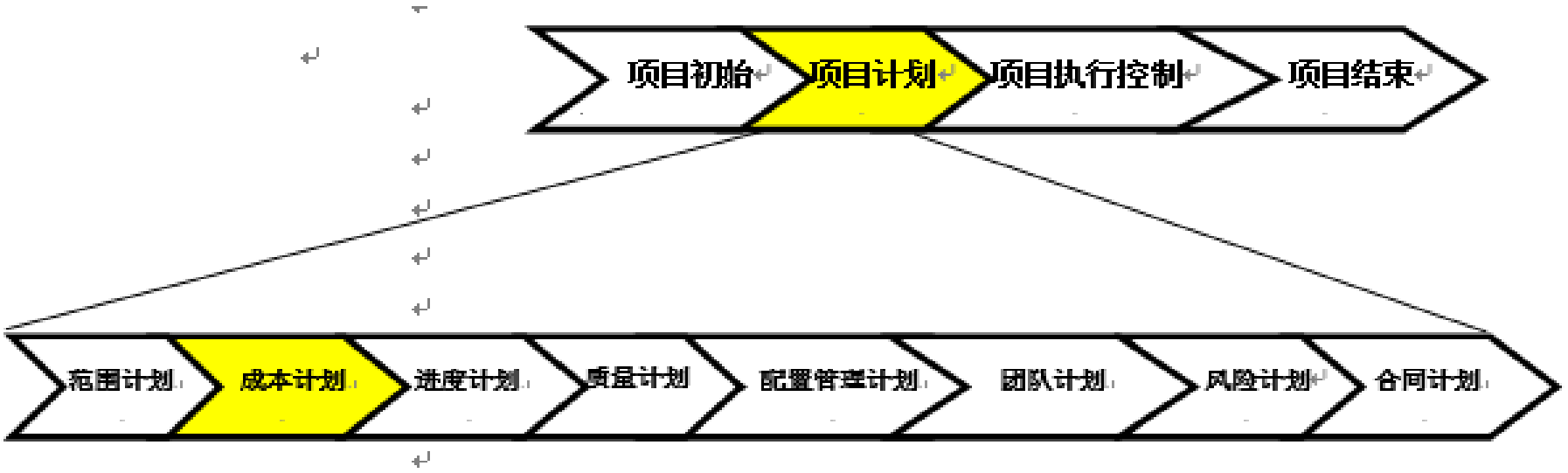
情景引入



甘特图

	任务模式	任务名称	WBS	工时	成本	工期	开始时间	完成时间	前置任务	资源名称
1		SPM课程平台	1	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
2		需求分析	1.1	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
3		概要设计	1.2	0 工时	¥0.00					
4		详细设计	1.3	0 工时	¥0.00					
5		编码实施	1.4	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
6		客户端子系统	1.4.1	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
7		系统登录	1.4.1.1	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
8		login.jsp	1.4.1.1.1	0 工时	¥0.00					
9		LoginAction.Java	1.4.1.1.2	0 工时	¥0.00					
10		LoginService.Java	1.4.1.1.3	0 工时	¥0.00					
11		LoginDAO.Java	1.4.1.1.4	0 工时	¥0.00					
12		系统注册	1.4.1.2	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
14		选课模块	1.4.1.3	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
16		成绩查询	1.4.1.4	0 工时	¥0.00					
17		网上测试	1.4.1.5	0 工时	¥0.00					
18		留言模块	1.4.1.6	0 工时	¥0.00					
19		学习进度管理	1.4.1.7	0 工时	¥0.00					
20		课程信息模块	1.4.1.8	0 工时	¥0.00					
21		管理端子系统	1.4.2	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
22		用户管理	1.4.2.1	0 工时	¥0.00					
23		课程资源管理	1.4.2.2	0 工时	¥0.00					
24		课程信息管理	1.4.2.3	0 工时	¥0.00					
25		界面	1.4.3	0 工时	¥0.00					
26		内部接口	1.4.4	0 工时	¥0.00					
27		公共模块	1.4.5	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		

路线图：成本计划


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
进度计划
质量计划
配置管理计划
团队计划
风险计划
合同计划

软件项目管理 第二篇

第 6 章 软件项目成本计划

本章要点

一 估算过程概念

二 传统估算方法

三 敏捷估算方法

四 成本预算

五 案例分析

关于估算

软件项目规模

甘特图

	任务模式	任务名称	WBS	工时	成本	工期	开始时间	完成时间	前置任务	资源名称
1		**SPM课程平台**	**1**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
2		需求分析	1.1	0 工时	¥0.00	1 个工作日	2015年11月	2015年11月		
3		概要设计	1.2	0 工时	¥0.00					
4		详细设计	1.3	0 工时	¥0.00					
5		**编码实施**	**1.4**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
6		**客户端子系统**	**1.4.1**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
7		**系统登录**	**1.4.1.1**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
8		login.jsp	1.4.1.1.1	0 工时	¥0.00					
9		LoginAction.Java	1.4.1.1.2	0 工时	¥0.00					
10		LoginService.Java	1.4.1.1.3	0 工时	¥0.00					
11		LoginDAO.Java	1.4.1.1.4	0 工时	¥0.00					
12		**系统注册**	**1.4.1.2**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
14		**选课模块**	**1.4.1.3**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
16		成绩查询	1.4.1.4	0 工时	¥0.00					
17		网上测试	1.4.1.5	0 工时	¥0.00					
18		留言模块	1.4.1.6	0 工时	¥0.00					
19		学习进度管理	1.4.1.7	0 工时	¥0.00					
20		课程信息模块	1.4.1.8	0 工时	¥0.00					
21		**管理端子系统**	**1.4.2**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		
22		用户管理	1.4.2.1	0 工时	¥0.00					
23		课程资源管理	1.4.2.2	0 工时	¥0.00					
24		课程信息管理	1.4.2.3	0 工时	¥0.00					
25		界面	1.4.3	0 工时	¥0.00					
26		内部接口	1.4.4	0 工时	¥0.00					
27		**公共模块**	**1.4.5**	**0 工时**	**¥0.00**	**1 个工作日**	**2015年11月**	**2015年11月**		

软件项目规模

软件项目规模即工作量

例如：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。

软件规模单位

- LOC(Loc of Code)
  - 源代码长度的测量
- 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
- 货币单位

软件规模和软件成本的关系

成本估算结果

直接成本

与具体项目相关的成本，例如：参与项目的人员成本

间接成本

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

本章要点

一 估算过程概念

二 传统估算方法

三 敏捷估算方法

四 成本预算

五 案例分析

传统估算方法

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

传统估算方法-代码行估算法

```
var chart1 = Ext.create("Ext.chart.Chart", {
    ctrid: "chart_col",
    store: me.stores,
    axes: [{
        type: 'Numeric',
        position: 'left',
        fields: me.chartY,
        label: {
            renderer: Ext.util.Format.numberRenderer('0,0')
        },
        grid: false,
        minimum: 0
    }, {
        type: 'Category',
        position: 'bottom',
        fields: [me.chartX]
    }],
    series: [{
        type: 'column',
        axis: 'left',
        xField: me.chartX,
        yField: me.chartY
    }]
});
```

传统估算方法-代码行估算法

从软件程序量的角度定义项目规模

- 与具体的编程语言有关
- 分解足够详细
- 有一定的经验数据

代码行技术的主要优缺点

主要优点

代码是所有软件开发项目都有的“产品”，而 且很容易计算代码行数。

主要缺陷

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

传统估算方法-

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

传统估算方法-功能点估算

功能点估算

- 与实现的语言和技术没有关系
- 用系统的功能数量来测量其规模
- 通过评估、加权、量化得出功能点

传统估算方法- Albrecht功能点估算

- 1979年, Alan Albrecht 提出
- 也称为IFPUG(国际功能点用户组织)功能点
- 适用于信息系统

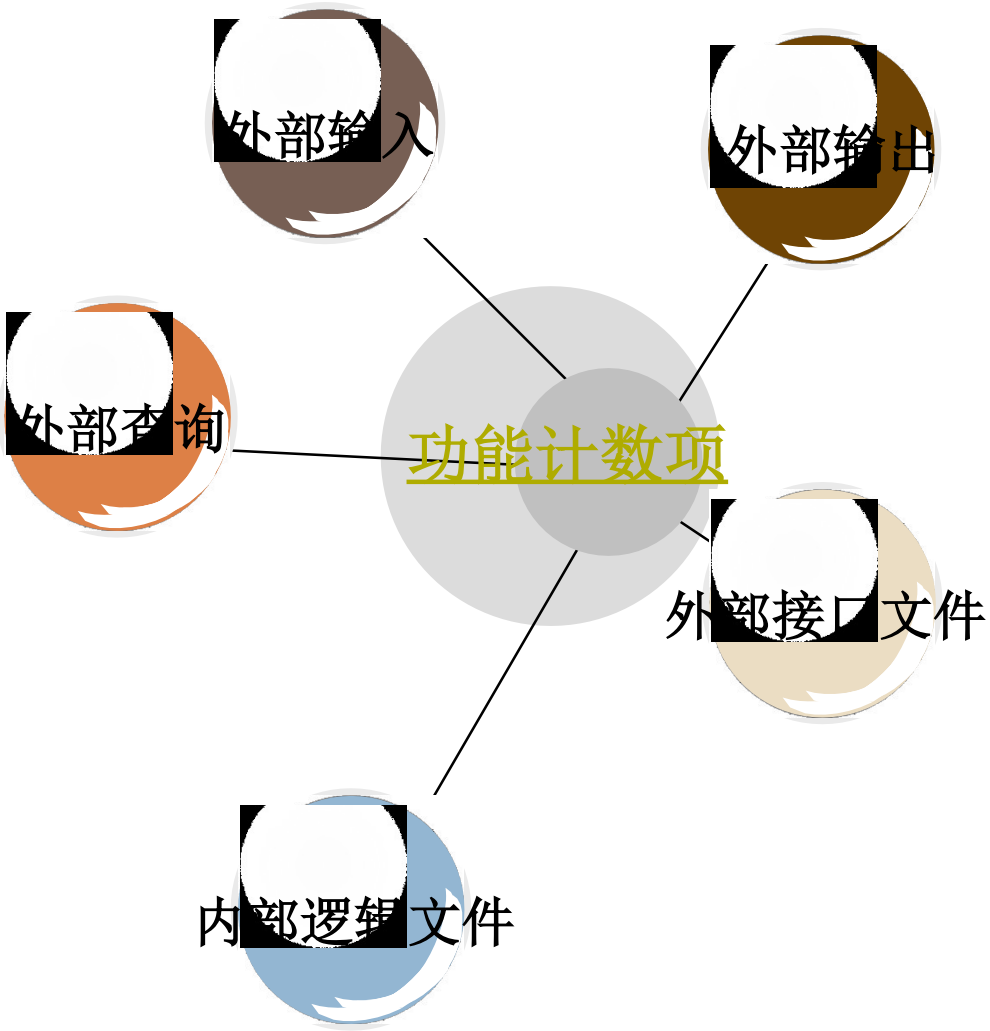
功能点公式

FP =UFC*TCF

- UFC：未调整功能点计数
- TCF：技术复杂度因子

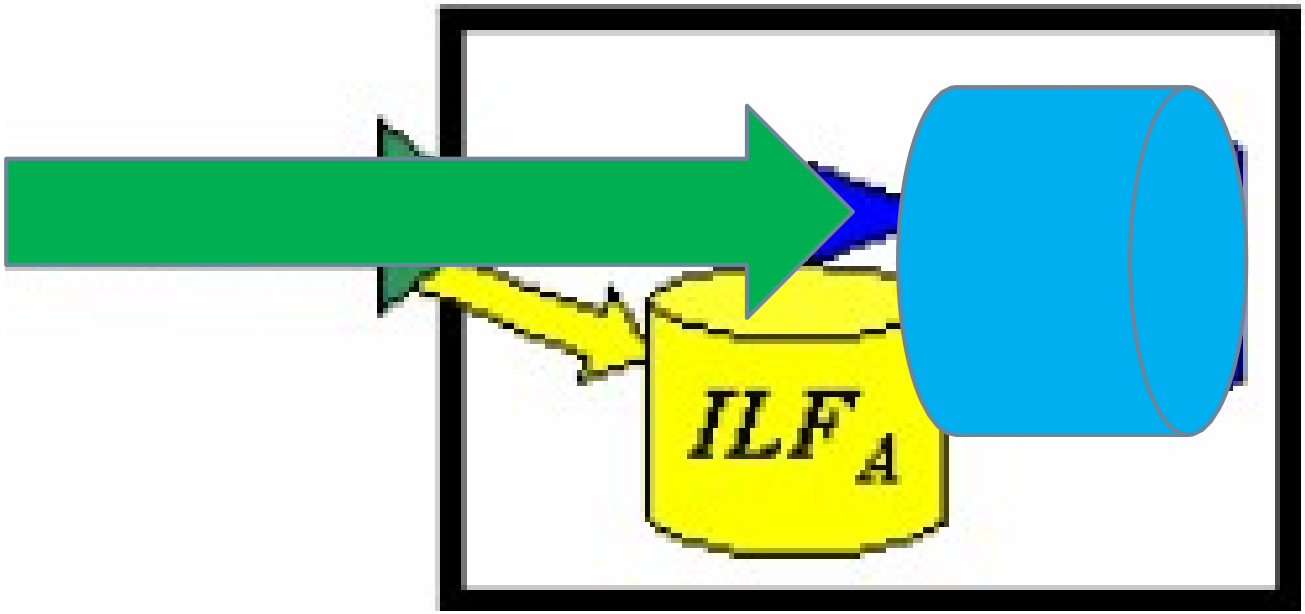
。

UFC-未调整功能点计数

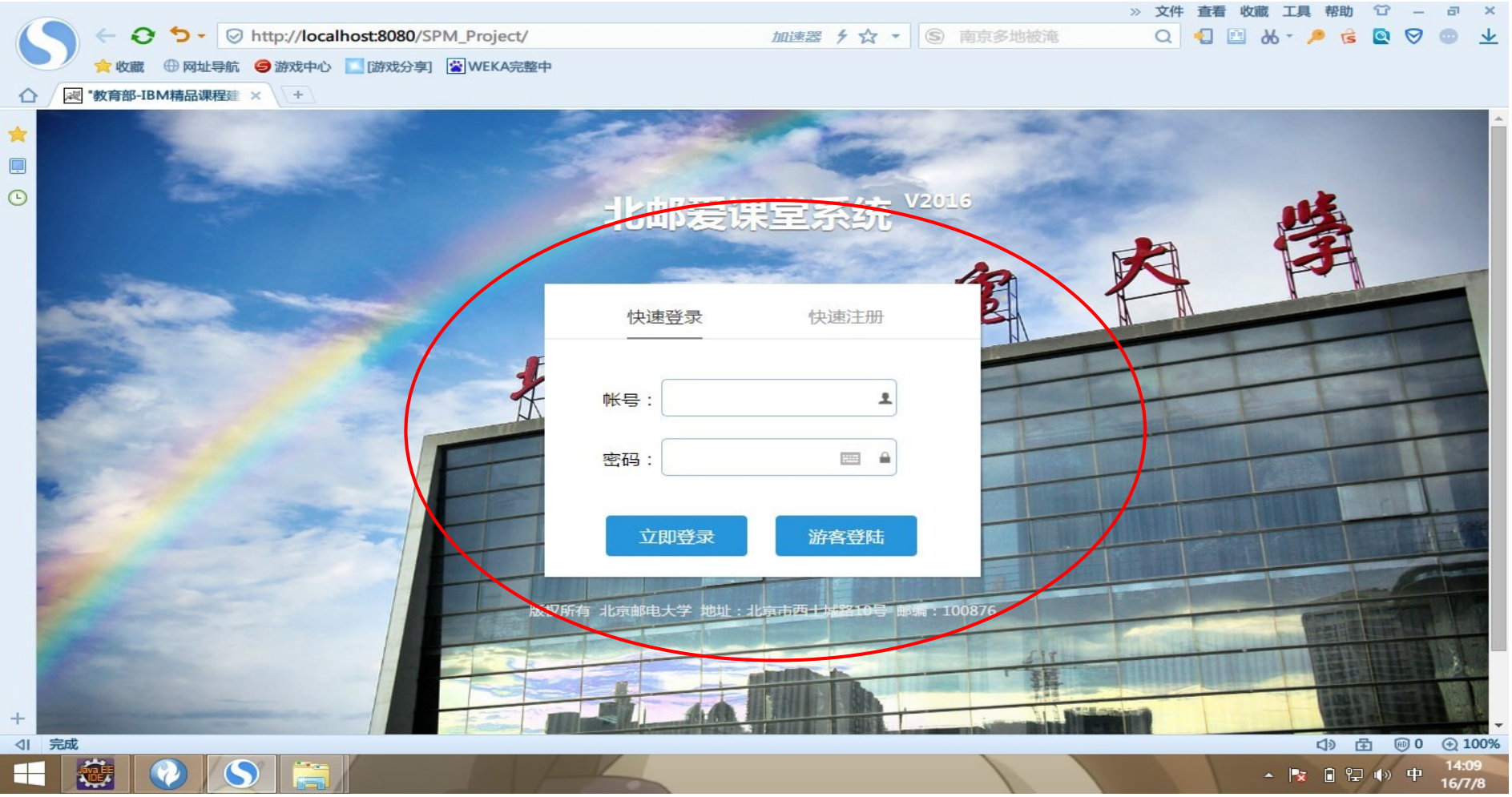


外部输入(External Inputs： EI)

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。

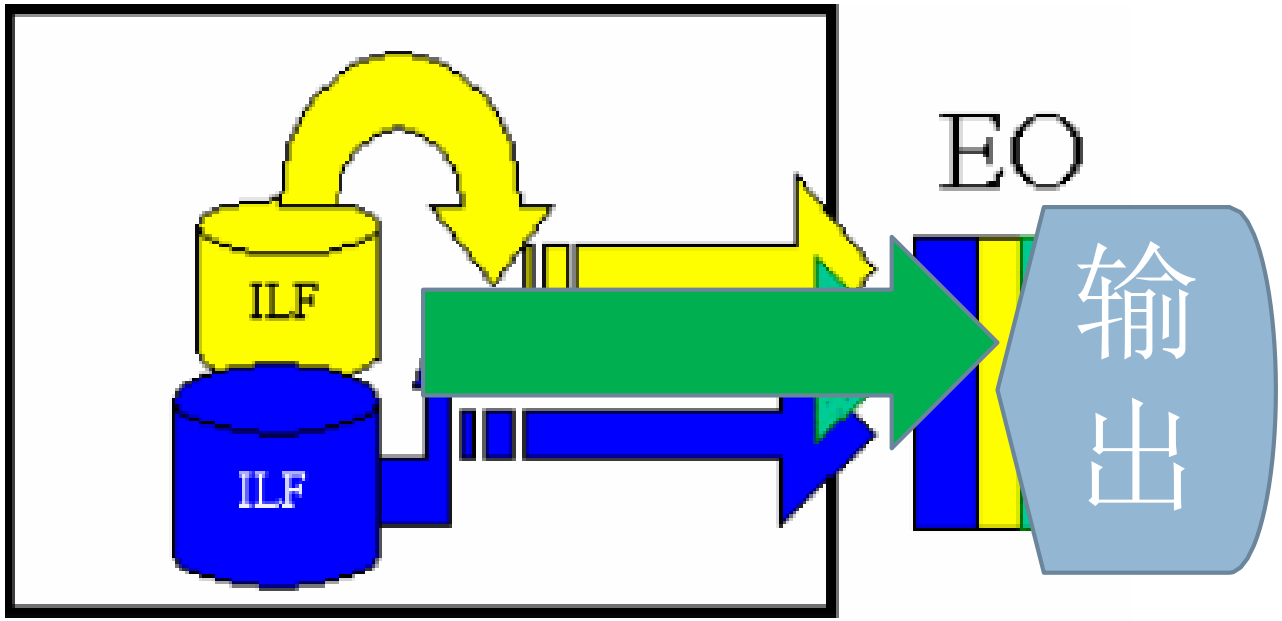


外部输入例子


http://localhost:8080/SPM_Project/
北邮爱课堂系统 V2016
快速登录
快速注册
帐号：
密码：
立即登录
游客登陆
版权所有 北京邮电大学 地址：北京市西土城路10号 邮编：100876

外部输出(External Outputs EO)

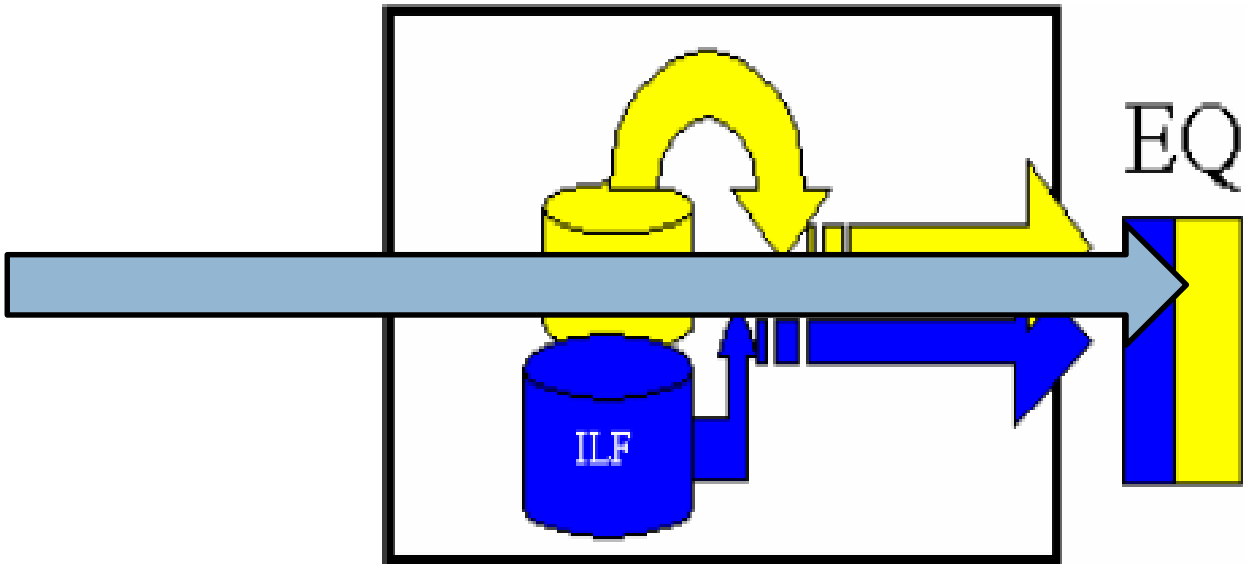
向用户提供(经过处理的)面向应用的信息，例如，报表和出错信息等。



外部输出例子

外部查询(External Inquiry EQ)

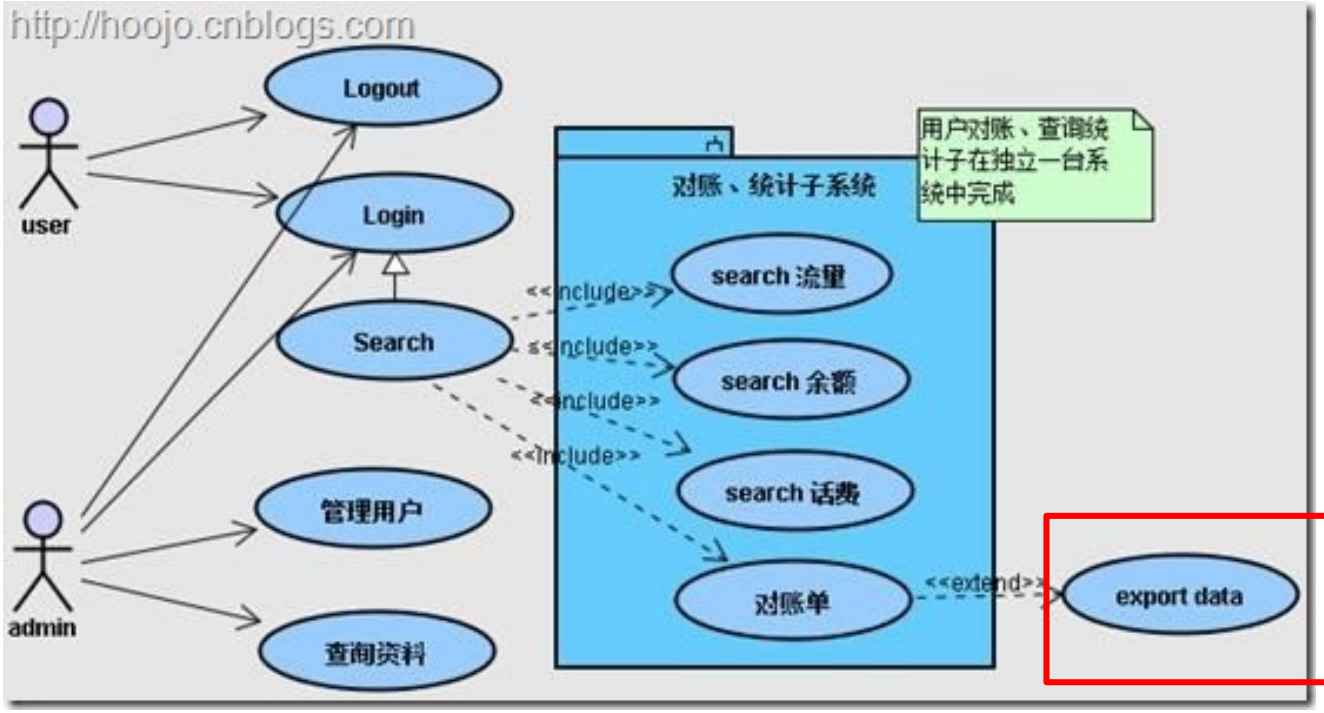
外部查询是一个输入引出一个即时的简单输出。
没有处理过程。



外部查询例子

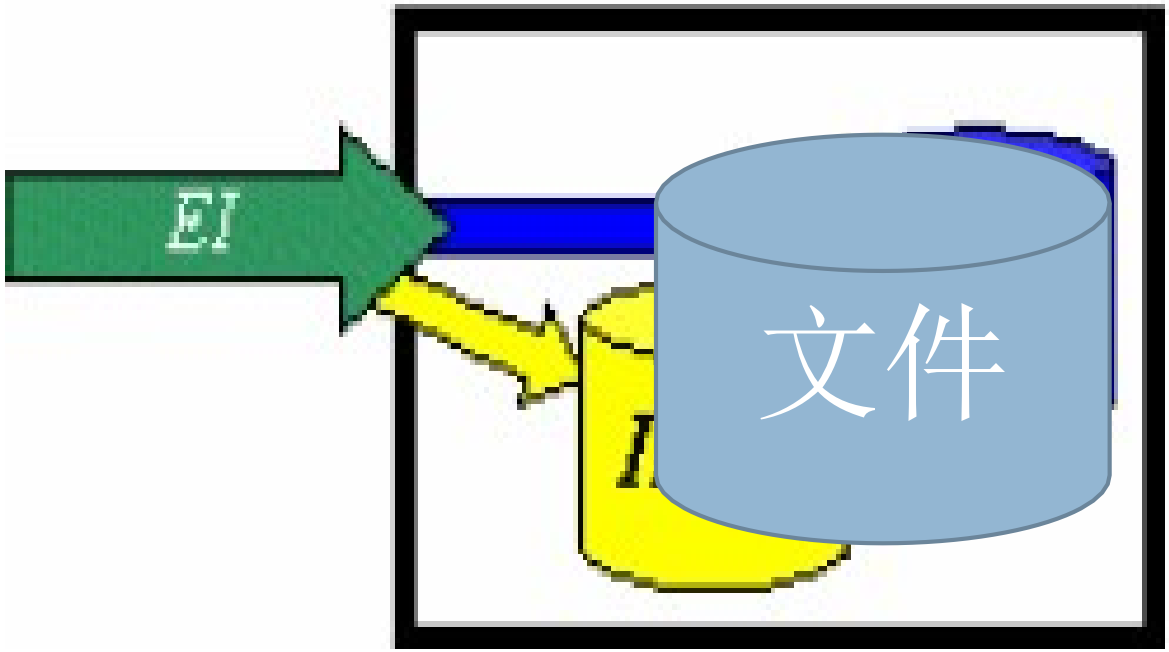
外部接口文件（External Interface Files EIF’s）

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。用这些接口把信息传送给另一个系统。

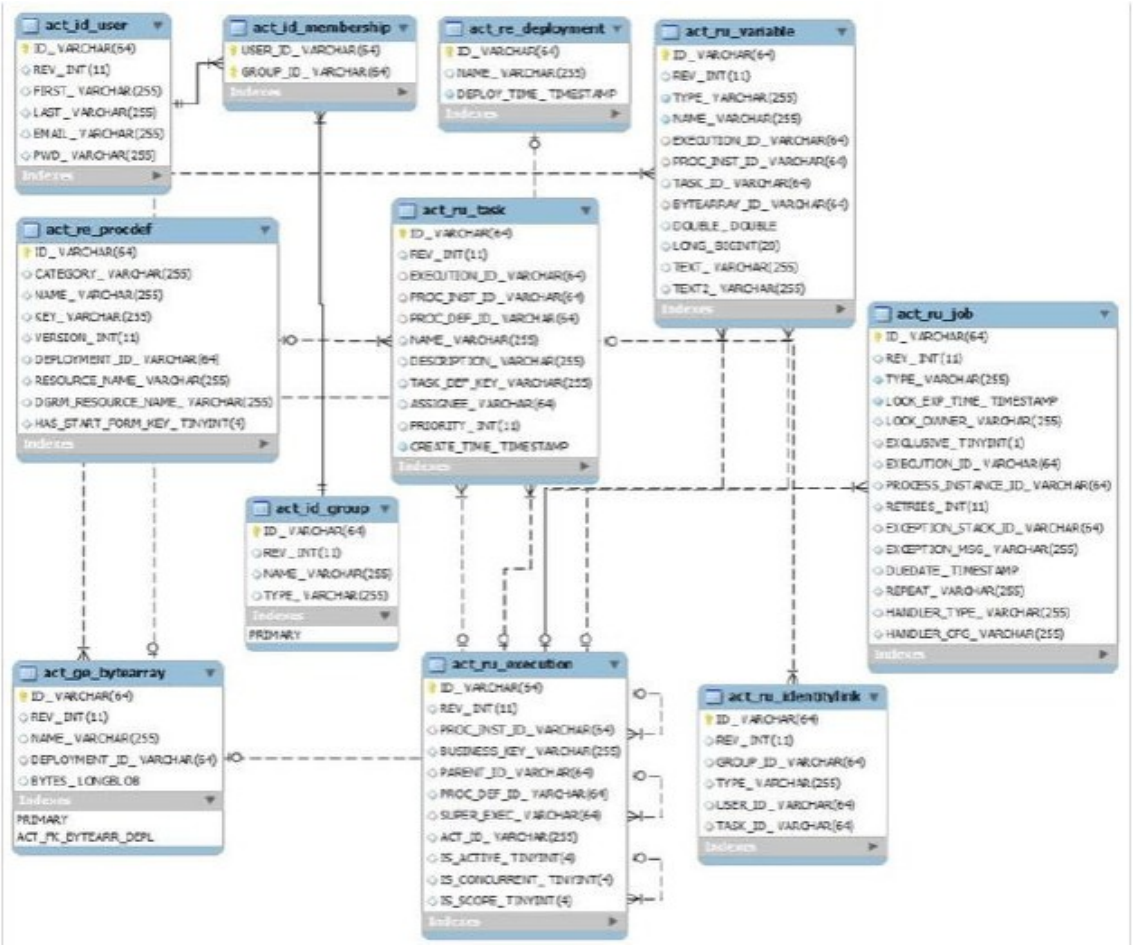


内部逻辑文件(Internal Logical Files： ILF’S)

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。



内部逻辑文件例子


act_id_user
ID_ VARCHAR(64)
REV_ INT(11)
FIRST_ VARCHAR(255)
LAST_ VARCHAR(255)
EMAIL_ VARCHAR(255)
PWD_ VARCHAR(255)
Indexes
act_id_membership
USER_ID_ VARCHAR(64)
GROUP_ID_ VARCHAR(64)
Indexes
act_re_deployment
ID_ VARCHAR(64)
NAME_ VARCHAR(255)
DEPLOY_TIME_ TIMESTAMP
Indexes
act_ru_variable
ID_ VARCHAR(64)
REV_ INT(11)
TYPE_ VARCHAR(255)
NAME_ VARCHAR(255)
EXECUTION_ID_ VARCHAR(64)
PROC_INST_ID_ VARCHAR(64)
TASK_ID_ VARCHAR(64)
BYTEARRAY_ID_ VARCHAR(64)
DOUBLE_ DOUBLE
LONG_ BIGINT(20)
TEXT_ VARCHAR(255)
TEXT2_ VARCHAR(255)
Indexes
act_re_procdef
ID_ VARCHAR(64)
CATEGORY_ VARCHAR(255)
NAME_ VARCHAR(255)
KEY_ VARCHAR(255)
VERSION_ INT(11)
DEPLOYMENT_ID_ VARCHAR(64)
RESOURCE_NAME_ VARCHAR(255)
DGRM_RESOURCE_NAME_ VARCHAR(255)
HAS_START_FORM_KEY_ TINYINT(4)
Indexes
act_ru_task
ID_ VARCHAR(64)
REV_ INT(11)
EXECUTION_ID_ VARCHAR(64)
PROC_INST_ID_ VARCHAR(64)
PROC_DEF_ID_ VARCHAR(64)
NAME_ VARCHAR(255)
DESCRIPTION_ VARCHAR(255)
TASK_DEF_KEY_ VARCHAR(255)
ASSIGNEE_ VARCHAR(64)
PRIORITY_ INT(11)
CREATE_TIME_ TIMESTAMP
Indexes
act_ru_job
ID_ VARCHAR(64)
REV_ INT(11)
TYPE_ VARCHAR(255)
LOCK_EXP_TIME_ TIMESTAMP
LOCK_OWNER_ VARCHAR(255)
EXCLUSIVE_ TINYINT(1)
EXECUTION_ID_ VARCHAR(64)
PROCESS_INSTANCE_ID_ VARCHAR(64)
RETRIES_ INT(11)
EXCEPTION_STACK_ID_ VARCHAR(64)
EXCEPTION_MSG_ VARCHAR(255)
DUEDATE_ TIMESTAMP
REPEAT_ VARCHAR(255)
HANDLER_TYPE_ VARCHAR(255)
HANDLER_CFG_ VARCHAR(255)
Indexes
act_id_group
ID_ VARCHAR(64)
REV_ INT(11)
NAME_ VARCHAR(255)
TYPE_ VARCHAR(255)
Indexes
PRIMARY
act_ge_bytearray
ID_ VARCHAR(64)
REV_ INT(11)
NAME_ VARCHAR(255)
DEPLOYMENT_ID_ VARCHAR(64)
BYTES_ LONGBLOB
Indexes
PRIMARY
ACT_FK_BYTEARR_DEPL
act_ru_execution
ID_ VARCHAR(64)
REV_ INT(11)
PROC_INST_ID_ VARCHAR(64)
BUSINESS_KEY_ VARCHAR(255)
PARENT_ID_ VARCHAR(64)
PROC_DEF_ID_ VARCHAR(64)
SUPER_EXEC_ VARCHAR(64)
ACT_ID_ VARCHAR(255)
IS_ACTIVE_ TINYINT(4)
IS_CONCURRENT_ TINYINT(4)
IS_SCOPE_ TINYINT(4)
Indexes
act_ru_identitylink
ID_ VARCHAR(64)
REV_ INT(11)
GROUP_ID_ VARCHAR(64)
TYPE_ VARCHAR(255)
USER_ID_ VARCHAR(64)
TASK_ID_ VARCHAR(64)
Indexes

FP计数的规则

国际功能点用户组织（IFPUG）发布FP计数的规则

<IFPUG功能点估算方法使用指南>

事务组件进行定级

表 6-1 外部输入的定级表

引用的文件类型个数（FTR's）	数据元素(Data Elements)		
	1-4	5-15	>15
0-1	低	低	低
2	低	中	高
>=3	中	高	高

表 6-2 外部输出和外部查询共用的定级表

引用的文件类型个数（FTR's）	数据元素(Data Elements)		
	1-5	6-19	>19
0-1	低	低	中
2－3	低	中	高
>3	中	高	高

6　3 外部输入、外部输出和外部查询的定级取值表

级数(Rating)	值		
	EO	EQ	EI
低	4	3	3
中	5	4	4
高	7	6	6

内部逻辑文件和外部接口文件

表 6-4 内部逻辑文件或者外部接口文件的定级表

记录元素类型 (RET's)	数据元素(Data Elements)		
	1-19	20-50	>50
1	低	低	中
2-5	低	中	高
>5	中	高	高

表 6　5 内部逻辑文件或者外部接口文件的定级取值表

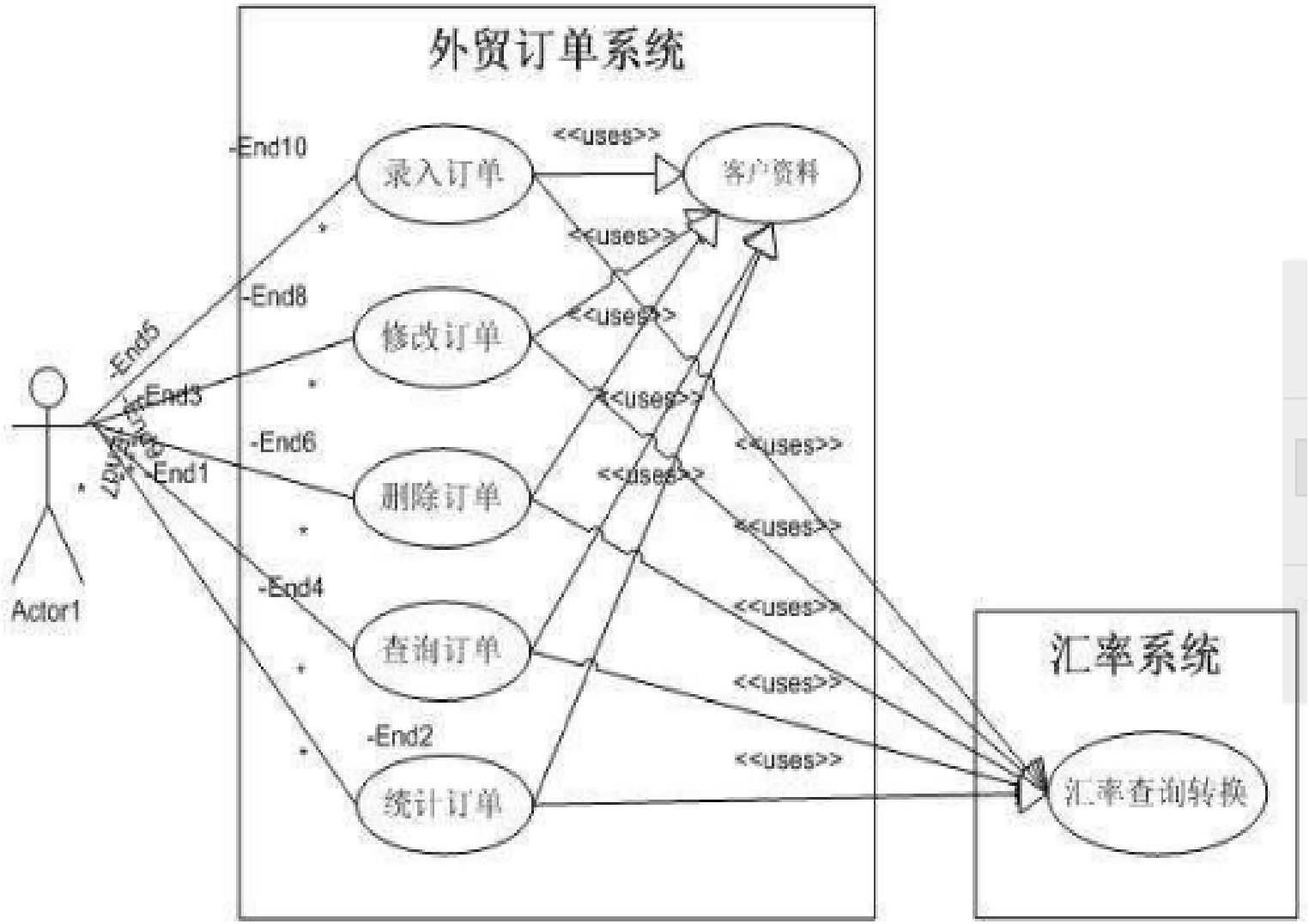
级数(Rating)	值	
	ILF	EIF
低	7	5
中	10	7
高	15	10

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(低)	一般（中）	复杂（高）
外部输入	3	4	6
外部输出	4	5	7
外部查询	3	4	6
外部接口文件	5	7	10
内部逻辑文件	7	10	15

功能点估算方法举例


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>>

外贸订单:--UFC

外部输入:3项;外部输出:1项;外部查询:1项;

外部接口文件:1项;内部逻辑文件:2项

	功能点		
项	简单	一般	复杂
外部输入	2 * 3	1 * 4	0 * 6
外部输出	0 * 4	0 * 5	1* 7
外部查询	0 * 3	1 * 4	0 * 6
外部接口文件	0 * 5	1 * 7	0 * 10
内部逻辑文件	1 * 7	1 * 10	0* 15
总计			
UFC	45		

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

外贸订单项目：功能点计算

因为：UFC=45

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

所以：FP=UFC*TCF=45*1.07=48

如果：PF=15工时/功能点

则：Effort=48*15=720工时

传统估算方法-功能点估算

其他功能点方法

- Mark II 功能点(主要应用在英国)
- COSMIC – FFP 功能点(适用实时系统或者嵌入式系统)

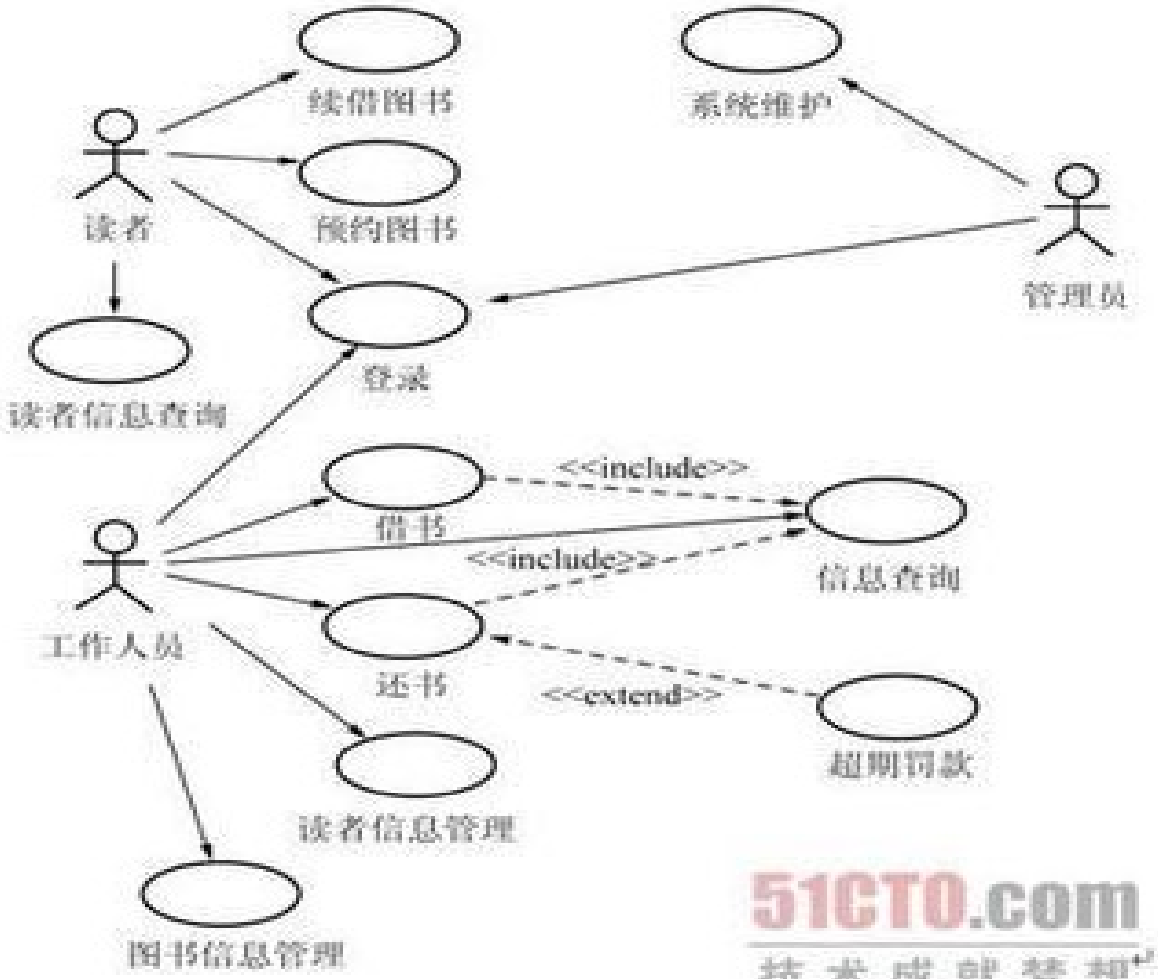
功能点与代码行的转换

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

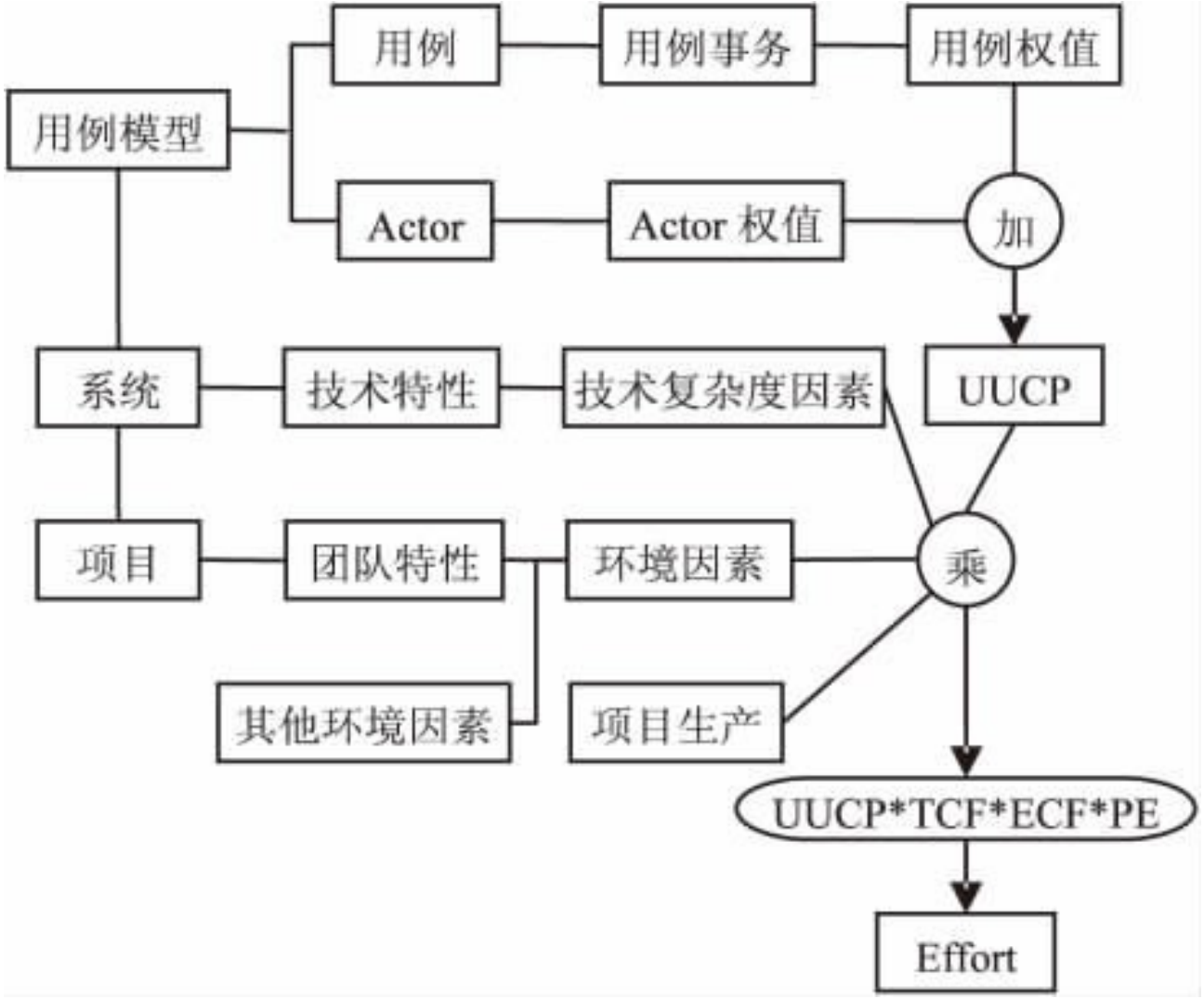
传统估算方法

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

传统估算方法-用例模型



传统估算方法-用例点估算模型



用例点估算方法的基本步骤

1. 计算未调整的角色权值UAW;
2. 计算未调整的用例权值UUCW；
3. 计算未调整的用例点UUCP;
4. 计算技术和环境因子TEF;
5. 计算调整的用例点UCP；
6. 计算工作量(man－hours)。

1、计算未调整的角色权值UAW

$$UAW = \sum_{c=1}^{x} aWeight(c) \times aCardinality(c)$$

表6-12 Actor 权值定义

序号	复杂度级别	复杂度标准	权值
1	simple	角色通过API 与系统交互	1
2	average	角色通过协议与系统交互	2
3	complex	用户通过GUI 与系统交互	3

2、计算未调整的用例权值UUCW

$$UUCW = \sum_{c=1}^{c} uWeight(c) \times uCardinality(c)$$

表6-13 Use Case 权值定义

序号	复杂度级别	事务/场景个数	权值
1	simple	1—3	5
2	average	4—7	10
3	complex	>7	15

3、计算未调整的用例点UUCP

UUCP =UAW+UUCW ：例如

表6-16： Actor 权值

序号	复杂度级别	权值	参与角色数	UAWi
1	simple	1	2	2
2	average	2	4	8
3	complex	3	5	15

表6-17：Use Case 权值

序号	复杂度级别	权值	用例数	UUCWi
1	simple	5	5	25
2	average	10	2	20
3	complex	15	3	45

UAW= = 1×2+2×4+3×5 = 2+8+15 = 25

UUCW= = 5×5+10×2+15×3 = 25+20+45 = 85

UUCP =UAW+UUCW= 25+85 = 110

4、计算技术因子TCF

$$TCF = 0.6 + (0.01 \times \sum_{i=1}^{13} TCF\_Weight_i \times Value_i)$$

序号	技术因子	说明	权值
1	TCF1	分布式系统	2.0
2	TCF2	性能要求	1.0
3	TCF3	最终用户使用效率	1.0
4	TCF4	内部处理复杂度	1.0
5	TCF5	复用程度	1.0
6	TCF6	易于安装	0.5
7	TCF7	系统易于使用	0.5
8	TCF8	可移植性	2.0
9	TCF9	系统易于修改	1.0
10	TCF10	并发性	1.0
11	TCF11	安全功能特性	1.0
12	TCF12	为第三方系统提供直接系统访问	1.0
13	TCF13	特殊的用户培训设施	1.0

TCF = 0.6+0. 01×(2.0×3+1.0×5+1.0×3+1.0×5+1.0×0 +0.5×3+0.5×5+2.0×3+1.0×5+1.0×3+1.0×5+1.0×0+1.0×0)= 1.02

4、 计算环境因子ECF

$$ECF = 1.4 + (-0.03 \times \sum_{i=1}^{8} ECF\_Weight_i \times Value_i)$$

表6-15　环境因子的定义

序号	环境因子	说明	权值
1	ECF1	UML 精通程度	1.5
2	ECF2	系统应用经验	0.5
3	ECF3	面向对象经验	1.0
4	ECF4	系统分析员能力	0.5
5	ECF5	团队士气	1.0
6	ECF6	需求稳定度	2.0
7	ECF7	兼职人员比例高低	1.0
8	ECF8	编程语言难易程度	1.0

ECF = 1.4+〔−0.03×（1.5×3+0.5×3+1.0×3+0.5×5+1.0×3+2.0×3+1.0 ×0+1.0×0）〕

= 0.785

5、计算调整的用例点UCP

$$UCP = UUCP \times TCF \times ECF$$

UCP =UUCP×TCF×ECF = 110×1.02×0.785 = 88

6、计算工作量

$$Effort = UCP \times PF$$

如果：PF = 20工时/用例点

则：Effort =UCP×PF = 88×20 = 1760h=220人天

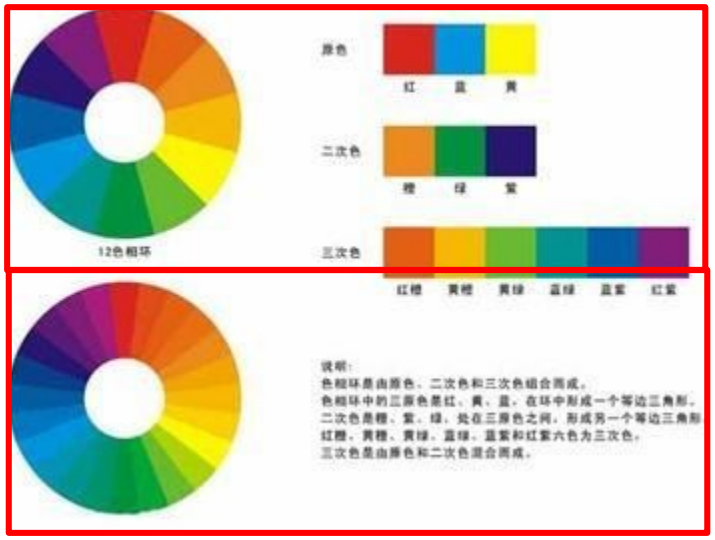
传统估算方法

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

传统估算方法-类比估算-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）。

是一种自上而下的估算形式



类比估算—使用情况

- 有类似的历史项目数据
- 信息不足（例如市场招标）的时候
- 要求不是非常精确估算的时候

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

类比估算—主观判断举例

证券交易网站

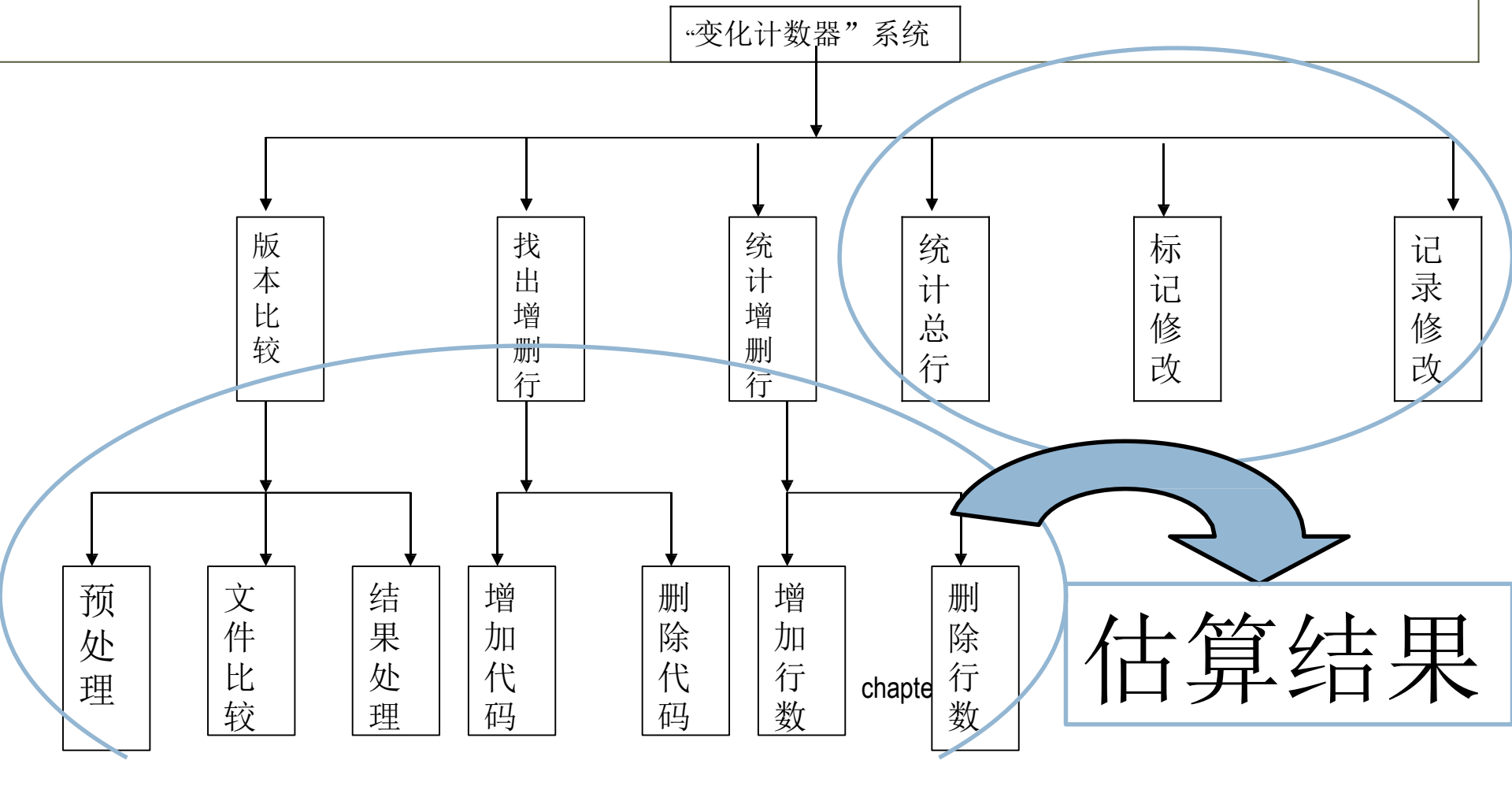
- 需求类似
- 历史数据：10万
- 类比估算：10万

传统估算方法

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

自下而上估算--定义

利用任务分解图(WBS),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。



自下而上估算--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
- 花费时间

自下而上估算举例

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			

传统估算方法

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

三点估算

基于任务成本的三种估算值来计算预期成本的方法.

三点估算 --三种估算值

- **最可能成本（C_M）:**比较现实的估算成本。
- **最乐观成本（C_O）:**最好情况所得到的估算成本。
- **最悲观成本（C_P）:**最差情况所得到的估算成本。

三点估算结果

- 三角分布: $C_E = (C_O + C_M + C_P) / 3$
- 贝塔分布: $C_E = (C_O + 4C_M + C_P) / 6$

三点估算结果-举例

- 三角分布: $C_E = (C_O + C_M + C_P) / 3$
- 贝塔分布: $C_E = (C_O + 4C_M + C_P) / 6$

例如: $C_O = 7$, $C_P = 12$, $C_M = 9$

三角分布: $C_E = 9.33$

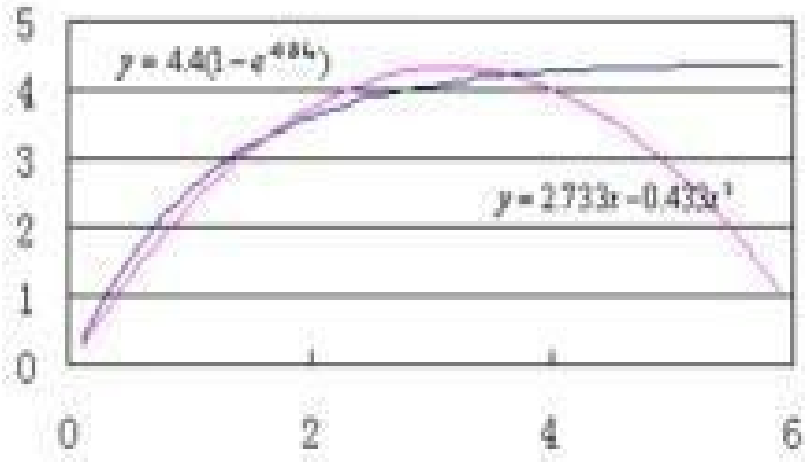
贝塔分布: $C_E = 10.67$

传统估算方法

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算
7. 参数估算法
8. 专家估算法

参数估算法—定义

- 通过项目数据,进行回归分析,得出回归模型
- 通过参数模型估算(规模)成本的方法。



参数模型: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:面向FP驱动的

Albrecht and Gaffney

- E=-12.39+0.0545FP

Matson,Barnett

- E=585.7+15.12FP

参数模型:整体公式

$E = a + b * S^c * M$

- E:以人月表示的工作量
- a,b,c:经验导出的系数
- M: 调节因子
- S:主要的输入参数(通常是LOC,FP等)

建议掌握模型

Walston-Felix模型

COCOMO模型

Walston-Felix模型

1977年，IBM的Walston和Felix提出了如下的估算公式

- $E = 5.2 \times (KLOC)^{0.91}$， KLOC是源代码行数，E是工作量（以PM计）
- $D = 4.1 \times (KLOC)^{0.36}$，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)
- $DOC = 49 \times (KLOC)^{1.01}$，DOC是文档数量(以页计)

Walston-Felix模型举例

采用java 完成项目，估计有366功能点，则

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

COCOMO（Constructive Cost model）

- 结构化成本模型
- 是目前应用最广泛的参数型软件成本估计模型
- 由Barry Boehm团队开发的

COCOMO模型

- COCOMO 81
- COCOMO II

COCOMO基本原理

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

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

A:可以校准的常量; S为KLOC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

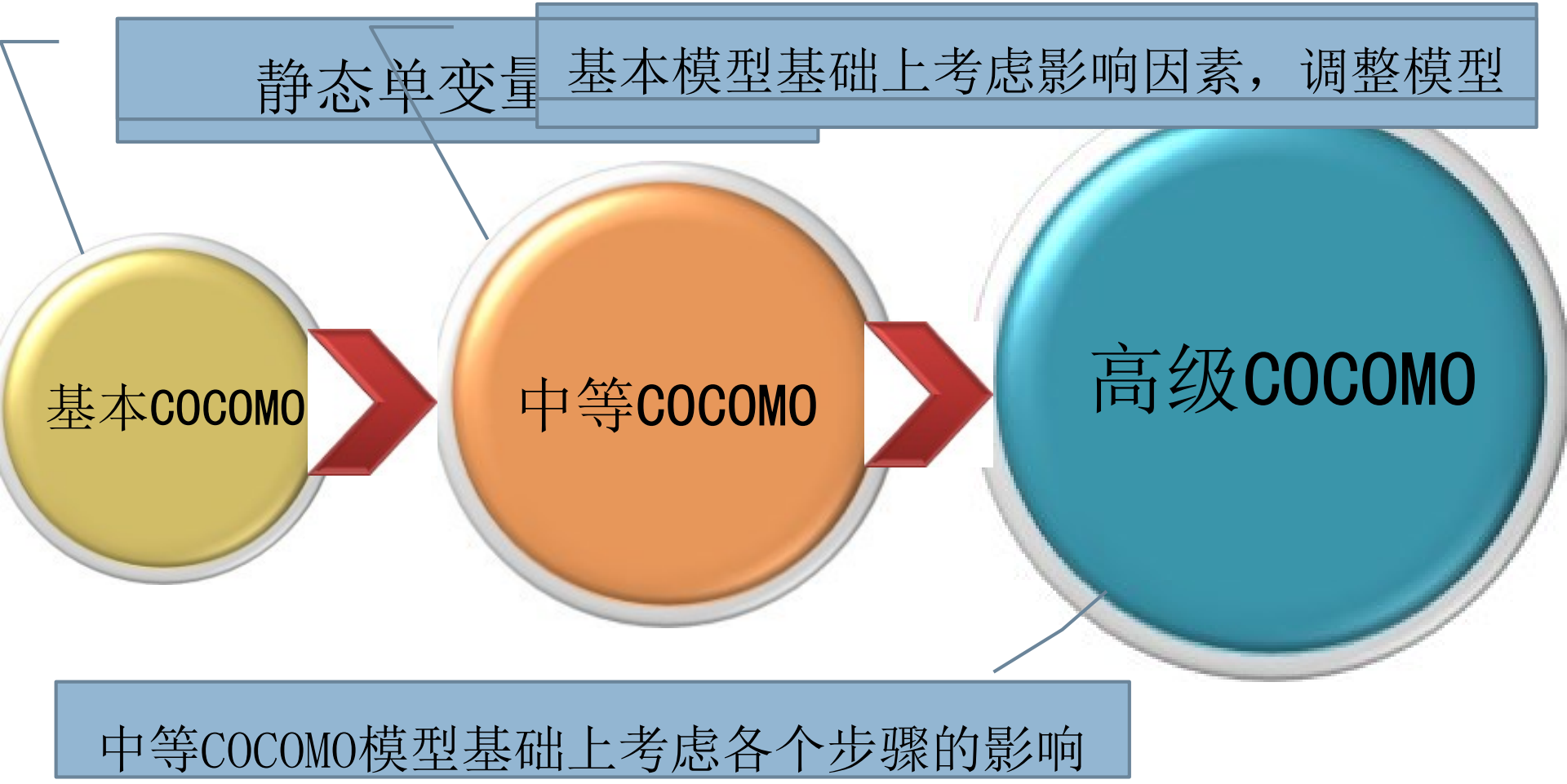
模型级别：

- 基本COCOMO
- 中等COCOMO
- 高级COCOMO

项目类型：

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

COCOMO 81模型的级别



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
- 半嵌入式: 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

$$E = a \times (KLOC)^b$$

E：工作量（人月）

KLOC：是交付的代码行

a，b：依赖于项目自然属性的系数

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，计算项目规模。

a=3.0，b=1.12。

E = 3.0 * L ^1.12 = 3.0 * 33.3 ^1.12 = 152 PM

中等COCOMO-81

$E = a*(KLOC)^b * 乘法因子$

- E：工作量（人月）
- KLOC：是交付的代码行
- a，b：依赖于项目自然属性的系数
- 乘法因子是对公式的校正系数

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

乘法因子的成本驱动属性

乘法因子

乘法因子＝F1XF2 X 。 。 。 X F15

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

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，采用中等COCOMO，计算项目规模。

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 ＊乘法因子＝ 3.0＊33.3 ^1.12 ×１.０９＝１６６ PM、

高级（详细）COCOMO

- 将项目分解为一系列的子系统或者子模型
- 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]

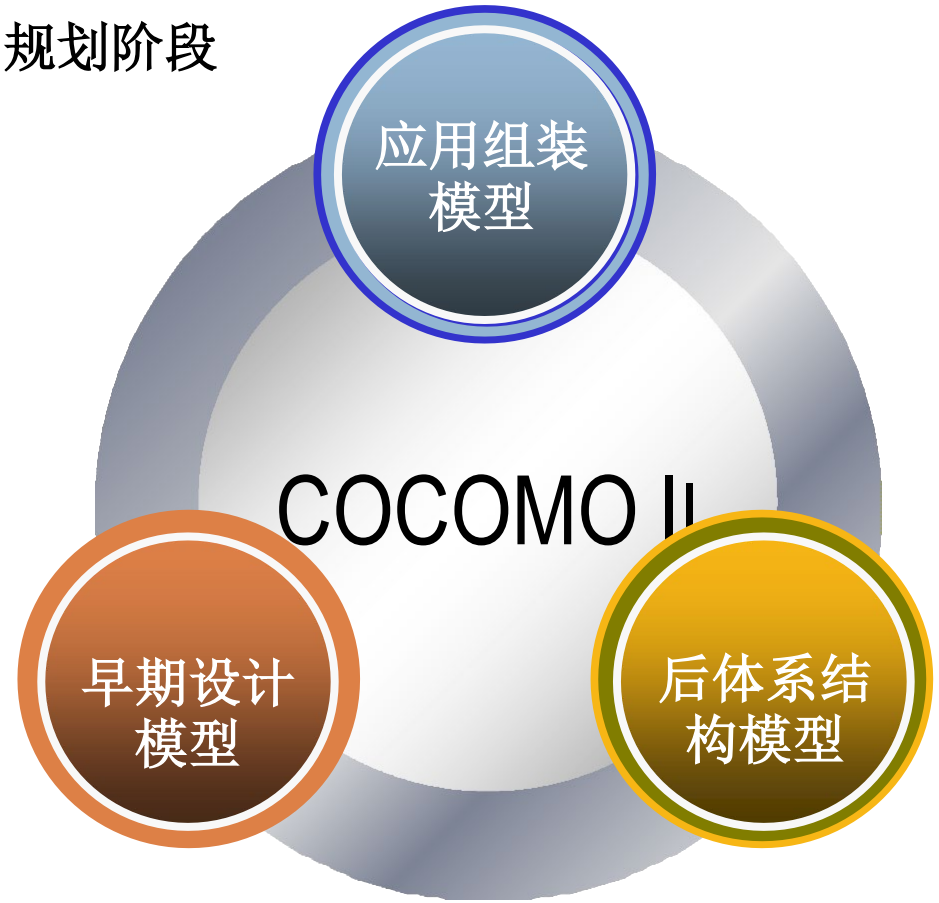
Cost drivers	Development phase	Rating levels					
		Very low	Low	Nominal	High	Very high	Extra high
AEXP	RPD (requirement and product design)	1.40	1.20	1.00	0.87	0.75	--
	DD (detailed design)	1.30	1.15	1.00	0.90	0.80	--
	CUT (code and unit test)	1.25	1.10	1.00	0.92	0.85	--
	IT (integration and test)	1.25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

COCOMO II

- COCOMO II是95年左右，Ｂoehm在81模型的基础上，结合了软件工程技术的发展提出来的，

COCOMO II组成

规划阶段



设计阶段

开发阶段

COCOMO II-应用组装模型

- 规划阶段
- 原型构造或者复用构件组合项目
- 公式： PM=（NAP*（1-%reuse/100））/PROD
  - PM：以人月为单位的工作量
  - NAP：应用点总数
  - %reuse：重用代码量估计
  - PROD：生产率

开发者的经验和能力	非常低	低	一般	高	非常高
软件工具的成熟度和能力	非常低	低	一般	高	非常高
PROD（NOP/月）	4	7	13	25	50

COCOMO II-早期设计模型

早期设计(early design)模型适用于项目初期，需求已经确定，系统设计的初始阶段

$$PM = A \times S^{E} \times \prod_{i=1}^{7} EM_{i}$$

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_{j}$$

PM是工作量；A是常数，；S是LOC规模；E是指数比例因子，B可以校准, SF是指数驱动因子，

COCOMO II-早期设计模型

-因子和系数

COCOMO II 比例因子值

驱动因子	很很低	低	正常	高	很高	极高	说明
PREC	6.2	4.96	3.72	2.48	1.24	0.00	有先例
FLEX	5.07	4.05	3.04	2.03	1.01	0.00	开发灵活性
RESL	7.07	5.65	4.24	2.83	1.41	0.00	架构风险解决方案
TEAM	5.48	4.38	3.29	2.19	1.10	0.00	团队凝聚力
PMAT	7.80	6.24	4.68	3.12	1.56	0.00	过程成熟度

COCOMO II 早期设计的工作量系数

驱动因子	级别(Rating levels)						
	非常低	很低 (Very low)	低(Low)	正常(Normal)	高(High)	很高(Very high)	极高(Extra high)
产品可靠性和复杂度:RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
需求的可重用性:RUSE			0.95	1.00	1.07	1.15	1.24
平台难度：PDIF			0.87	1.00	1.29	1.81	2.61
人员的能力：PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
人员的经验：PREX	1.59	1.33	1.12	1.00	0.87	0.74	0.62
设施的可用性：PCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
进度压力：SCED		1.43	1.14	1.00	1.00	1.00	

COCOMO II-后体系结构模型

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

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

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

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

复用模型

ESLOC=（ASLOC*（1-AT/100）*AAM）

- ESLOC：新源代码的等价行数
- ASLOC：必须修改的复用构件的代码行数
- AT/100：可以自动修改的复用代码所占百分比
- AAM：改写调整因子，反映了构件复用时所需的额外工作量

$$PM = A \times S^{E} \times \prod_{i=1}^{7} EM_{i}$$

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

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

参数模型综述

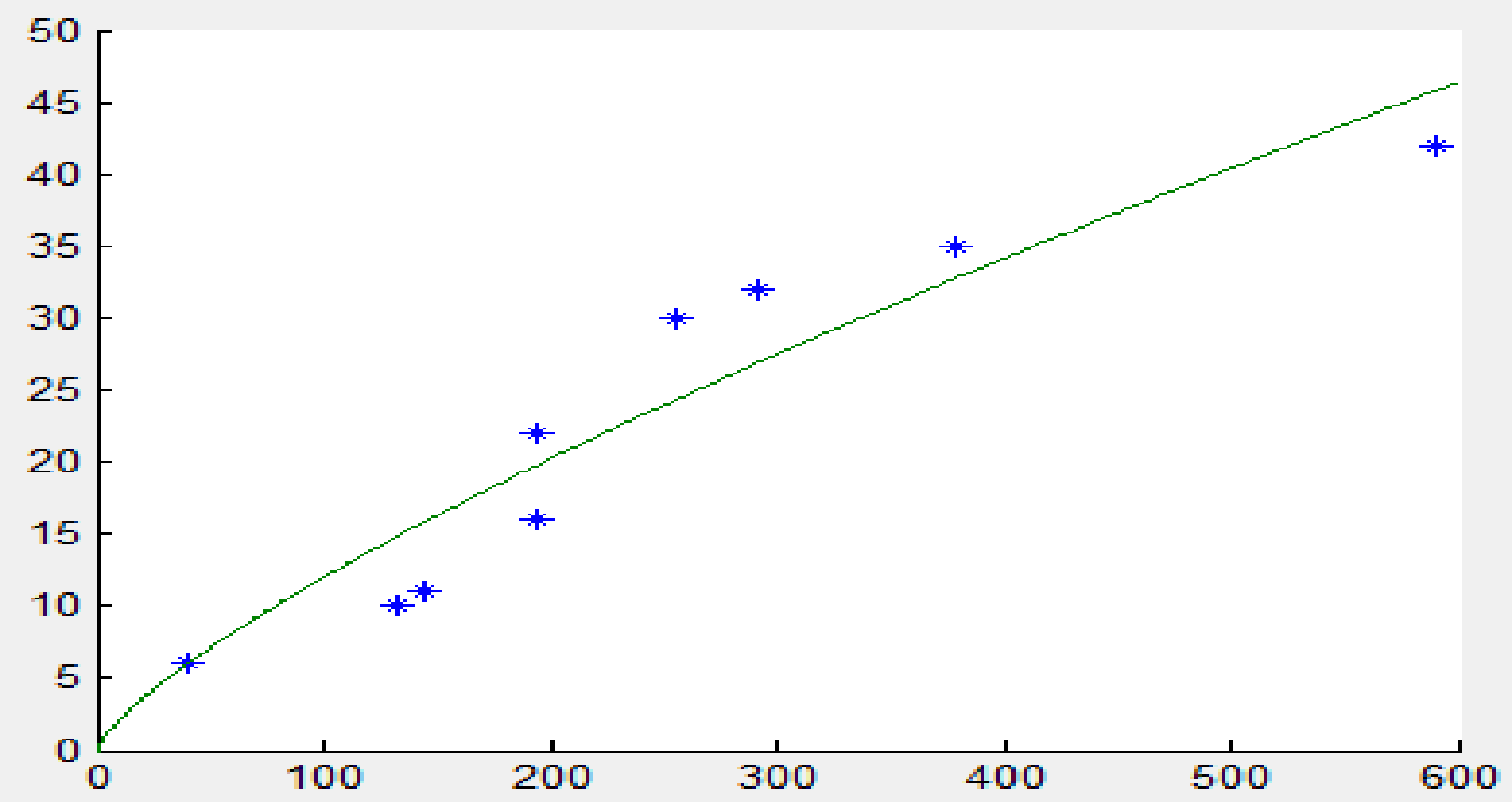
- 根据项目数据进行回归分析，得出回归模型作为参数模型。
- 回归分析方法：线性回归，多项式回归，逻辑回归，神经网络，集成方法等。
- 参数模型可以是线性也可以是非线性的。

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—多项式回归算法

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
```

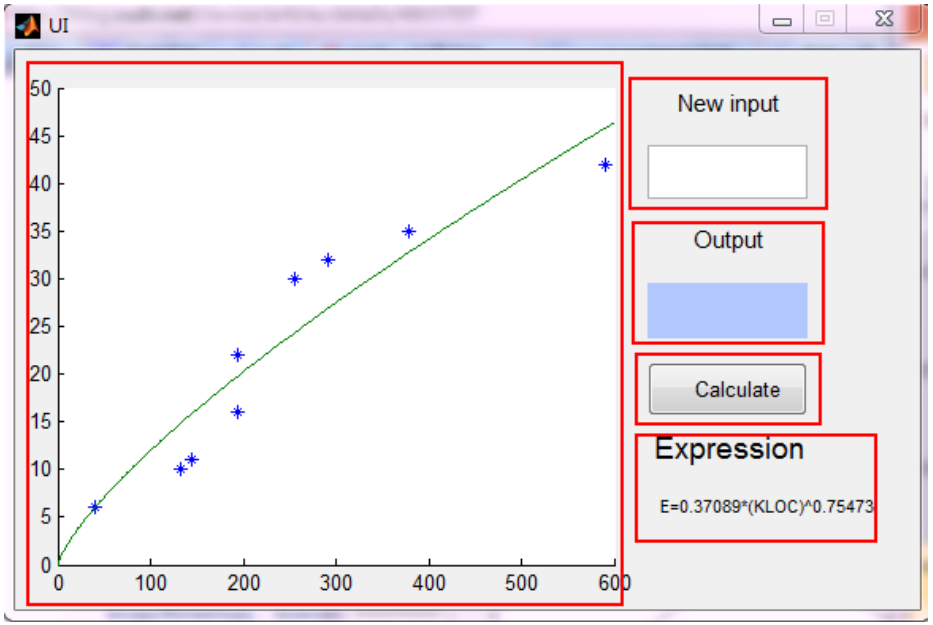
多项式回归模型例子--结果输出

模型输出

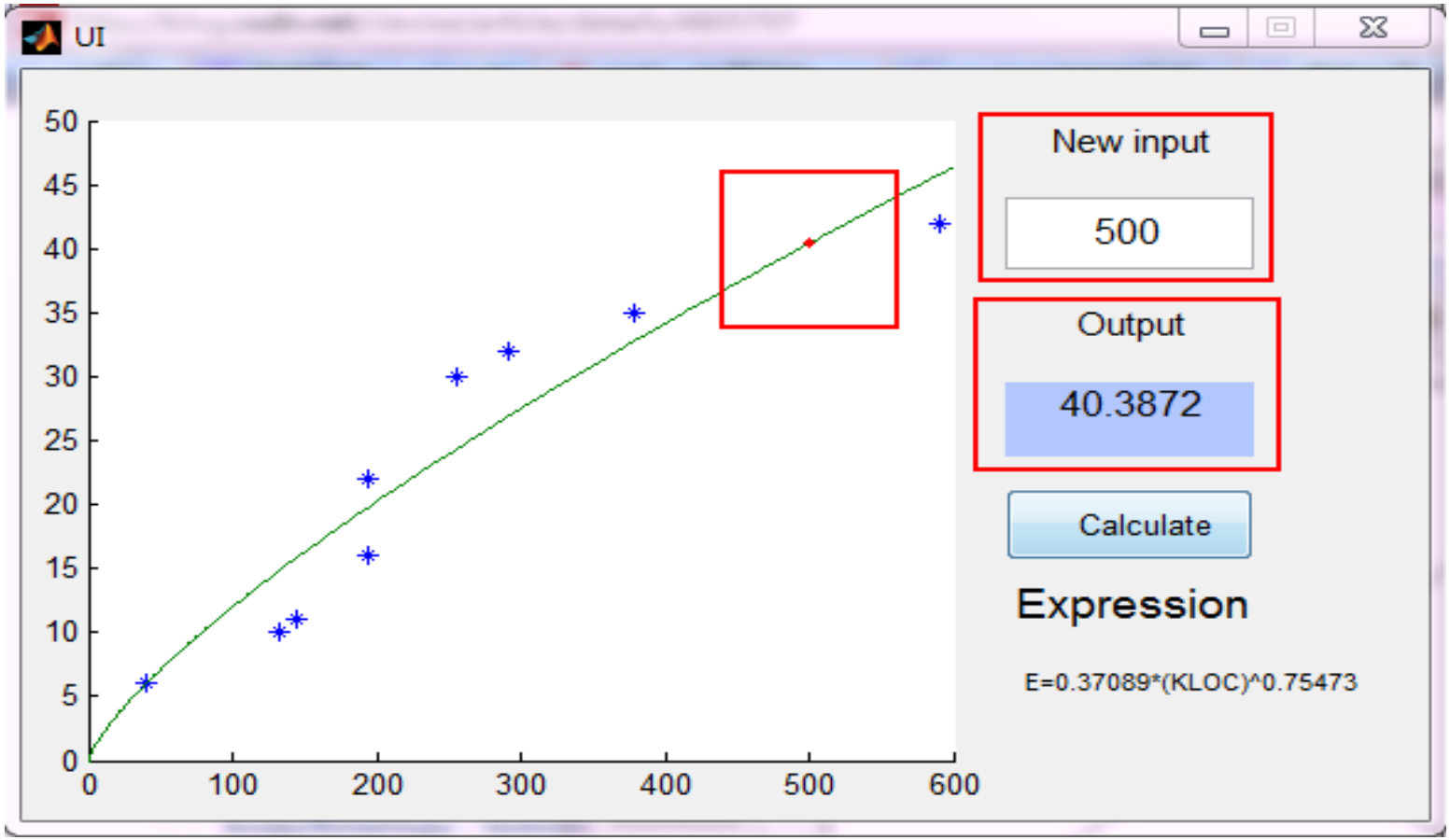
Expression

E=0.37089*(KLOC)^0.75473

图形输出



模型研究例子--多项式回归模型应用



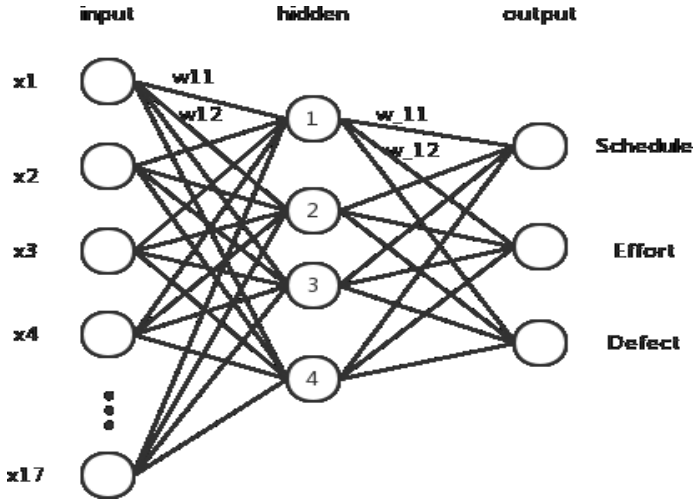
基于神经网络估算模型-项目数据

	A	B	C	D	E	F	G	H	I	J	K	L
1	功能点数	38	54	31	47	58	1	51	45	39	34	3
2	可靠性	很高	高	很低	很低	很低	低	低	很低	低	很低	高
3	数据规模	高	很高	很低	极高	极高	正常	低	很高	很低	低	很低
4	复杂性	高	低	高	正常	很高	很低	正常	低	很低	高	正常
5	文档	低	正常	极高	低	极高	高	极高	很高	很高	很低	高
6	复用	正常	很低	极高	很低	极高	低	低	低	很高	正常	极高
7	技术难度	很低	很高	低	极高	极高	高	正常	高	低	很高	正常
8	平台难度	低	很低	低	很低	极高	极高	低	极高	高	很高	很低
9	需求易变性	很低	很低	高	很高	很低	极高	很低	高	低	低	高
10	分析能力	低	极高	正常	正常	高	很高	正常	低	极高	正常	正常
11	应用经验	高	正常	正常	低	高	很高	极高	很低	极高	高	高
12	程序员水平	极高	高	极高	高	正常	高	很高	正常	正常	正常	很高
13	语言与工具经验	高	低	极高	高	低	很低	低	低	很低	正常	很高
14	团队凝聚力	极高	正常	很低	很低	低	高	高	高	正常	高	高
15	过程成熟度	正常	高	高	很高	低	极高	低	极高	低	高	正常
16	使用软件工具的水平	正常	极高	低	低	高	很低	正常	很低	低	低	正常
17	进度管理	很高	低	正常	正常	极高	极高	极高	极高	极高	低	正常
18												
19	时间（月）	17.6	25	14.4	24.6	29	0.4	24.6	26.1	18	15	1.7
20	成本（人/月）	52.8	75	43.2	73.8	87	1.2	73.8	78.3	54	45	5.1
21	缺陷数（个）	95	135	79	125	156	4	125	117	89	75	7

项目进行数字化和归一化

	A	B	C	D	E	F	G	H	I	J	K	L
1	功能点数	0.88	0.34	0.35	0.63	0.90	0.01	0.44	0.14	0.52	0.35	0.04
2	可靠性	0.65	0.13	0.51	0.43	0.43	0.40	0.20	0.13	0.07	0.05	0.63
3	数据规模	0.42	0.84	0.97	0.09	0.30	0.14	0.57	0.60	0.94	0.58	0.75
4	复杂性	0.90	0.50	0.23	0.83	0.42	0.32	0.75	0.33	0.26	0.06	0.56
5	文档	0.97	0.21	0.71	0.19	0.34	0.95	0.77	0.96	0.04	0.38	0.50
6	复用	0.45	0.74	0.21	0.29	0.32	0.93	0.43	0.90	0.49	0.25	0.42
7	技术难度	0.13	0.03	0.42	0.64	0.04	0.61	0.96	0.41	0.87	0.09	0.83
8	平台难度	0.34	0.90	0.46	0.10	0.92	0.72	0.06	0.69	0.22	0.80	0.43
9	需求易变性	0.62	0.53	0.17	0.69	0.85	0.92	0.10	0.62	0.16	0.84	0.47
10	分析能力	0.12	0.48	0.60	0.73	0.96	0.65	0.42	0.09	0.59	0.70	0.59
11	应用经验	0.49	0.25	0.07	0.16	0.00	0.28	0.40	0.40	0.65	0.57	0.58
12	程序员水平	0.82	0.06	0.71	0.37	0.74	0.66	0.06	0.57	0.04	0.15	0.47
13	语言与工具经验	0.15	0.57	0.32	0.09	0.27	0.62	0.94	0.08	0.64	0.52	0.01
14	团队凝聚力	0.07	0.89	0.15	0.19	0.46	0.63	0.75	0.05	0.75	0.81	0.11
15	过程成熟度	0.26	0.01	0.46	0.17	0.19	0.79	0.27	0.29	0.06	0.02	0.07
16	使用软件工具的水平	0.20	0.42	0.41	0.60	0.60	0.84	0.04	0.16	0.10	0.04	0.99
17	进度管理	0.29	0.10	0.47	0.41	0.94	0.17	0.24	0.75	0.84	0.36	0.85
18												
19	时间（月）	0.28	0.80	0.31	0.64	0.86	0.06	0.77	0.07	0.83	0.34	0.14
20	成本（人/月）	0.71	0.95	0.53	0.65	0.37	0.53	0.68	0.20	0.73	0.68	0.18
21	缺陷数（个）	0.95	0.96	0.71	0.85	0.87	0.88	0.96	0.32	0.08	0.29	0.54

采用三层的BP神经网络建模



隐含层节点 i 的输入为

$$h_i = \sum w_{ij} x_j + b_i$$

隐含层节点 i 的输出为

$$y_i = \theta(h_i) = \theta\left(\sum w_{ij} x_j + b_i\right)$$

隐含层的所有节点的输出就是输出层节点的输入信号，类似地可得输出层节点 k 的输入为

$$out\_in_k = \sum w_{ki} y_i + a_k = \sum w_{ki}\theta\left(\sum w_{ij} x_j + b_i\right) + a_k$$

输出层节点 k 的输出为

$$o_k = \beta(out\_in_k) = \beta\left(\sum w_{ki} y_i + a_k\right) = \beta\left(\sum w_{ki}\theta\left(\sum w_{ij} x_j + b_i\right) + a_k\right)$$

模型建立

$$\Delta w_{ki} = \eta \cdot e_k \cdot \beta' \cdot y_i$$

$$\Delta a_k = \eta \cdot e_k \cdot \beta'$$

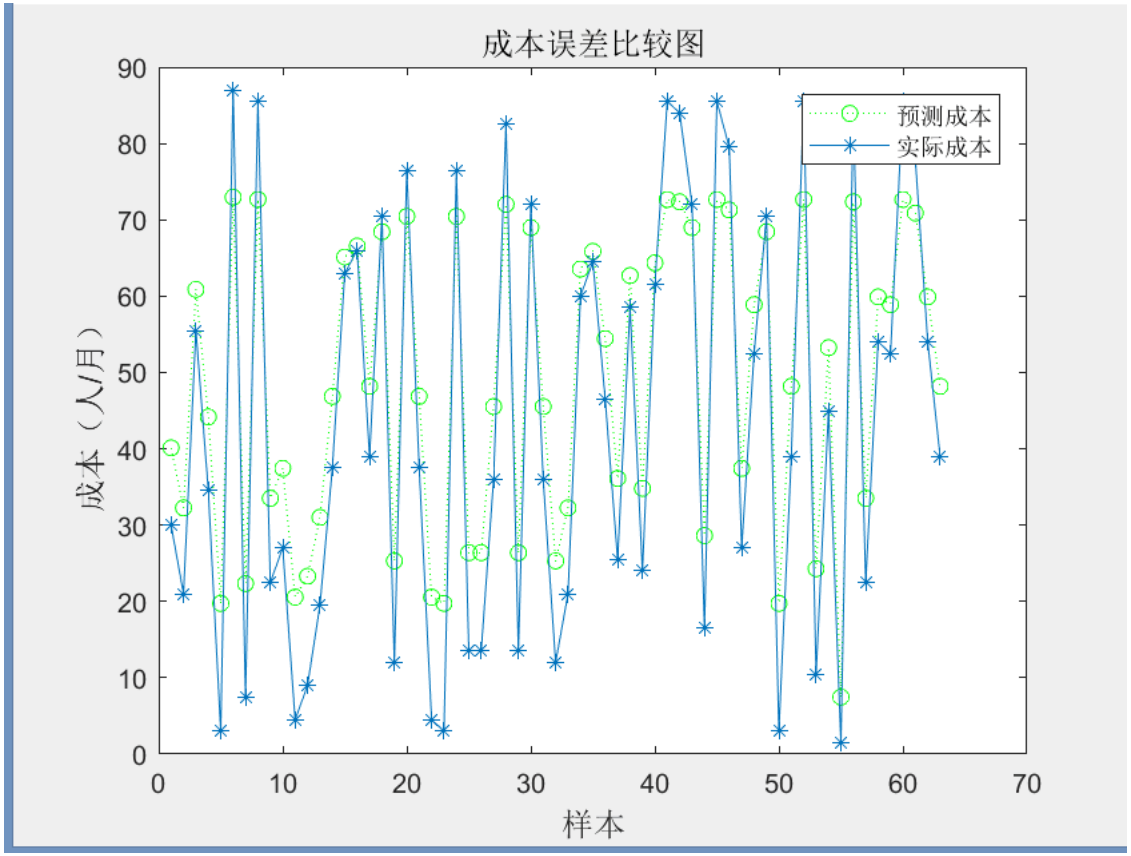
$$\Delta w_{ij} = \eta \cdot \beta' \cdot \theta' \cdot x_j \cdot e_k \cdot w_{ki}$$

$$\Delta b_i = \eta \cdot \beta' \cdot \theta' \cdot e_k \cdot w_{ki}$$

分析结果：

- 隐层节点数目为4，
- 网络学习率为0.54，
- 隐层激活函数为logsig，
- 输出层激活函数为purelin。
- 确定动量因子mc为0.9

模型应用



参数估算法

使用条件

- 具有良好的项目数据为基础
- 存在成熟的项目估算模型

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

传统估算方法-

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 三点估算法
7. 参数估算法
8. 专家估算法

专家估算法

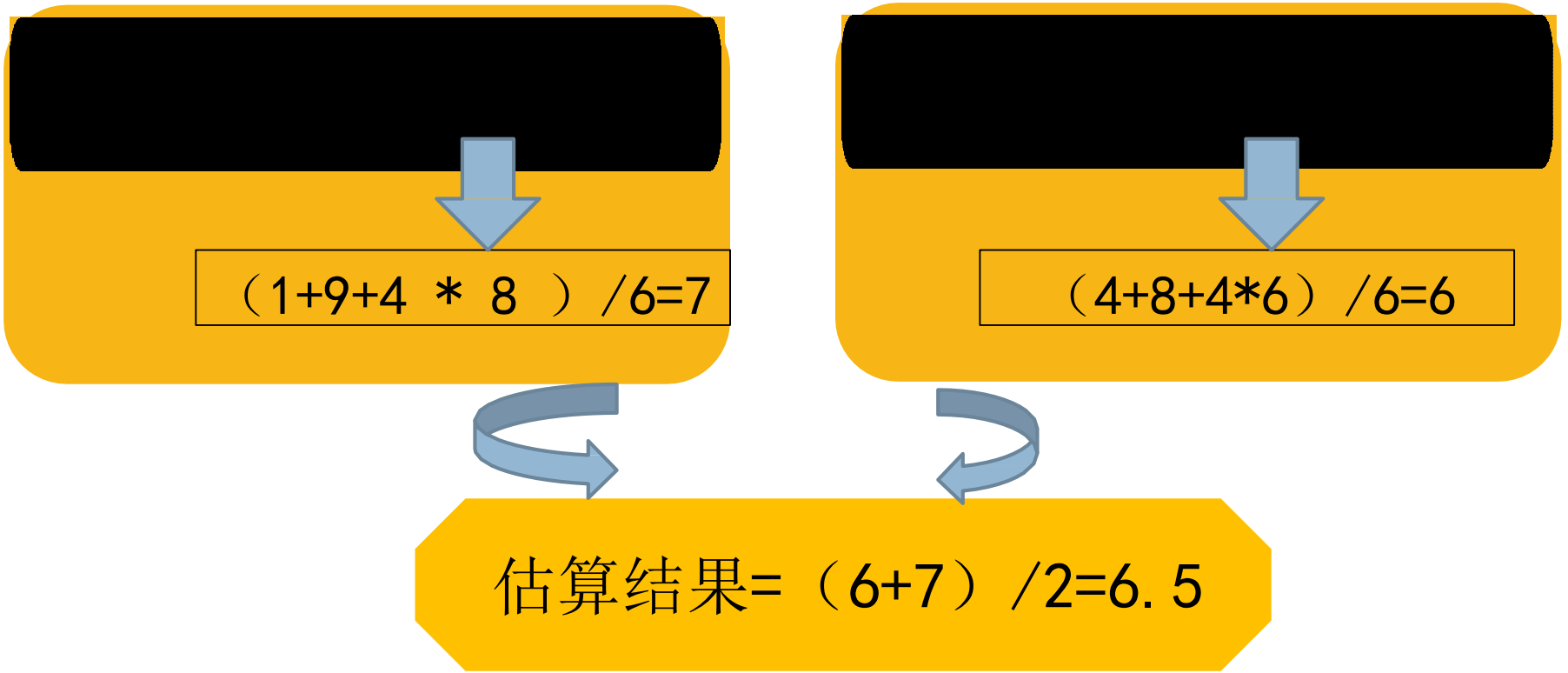
由多位专家进行成本估算，一个专家可能会有偏见，最好由多位专家进行估算，取得多个估算值，最后得出综合的估算值。

专家估算法-Deiphi

1. 组织者确定专家，这些专家互相不见面
2. 组织者发给每位专家一份软件规格说明
3. 专家以无记名对该软件给出3个规模的估算值
   ① 最小ai
   ② 最可能的mi
   ③ 最大bi
4. 组织者计算每位专家的Ei=(ai+4mi+bi)/6
5. 最终可以获得一个多数专家共识的软件规模：E=E1+E2+…En/n（n:表示n 个专家）
6. 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

Deiphi专家估算法-举例

某多媒体信息查询系统，采用专家估算方法



本章要点

一 估算过程概念

二 传统估算方法

三 敏捷估算方法

四 成本预算

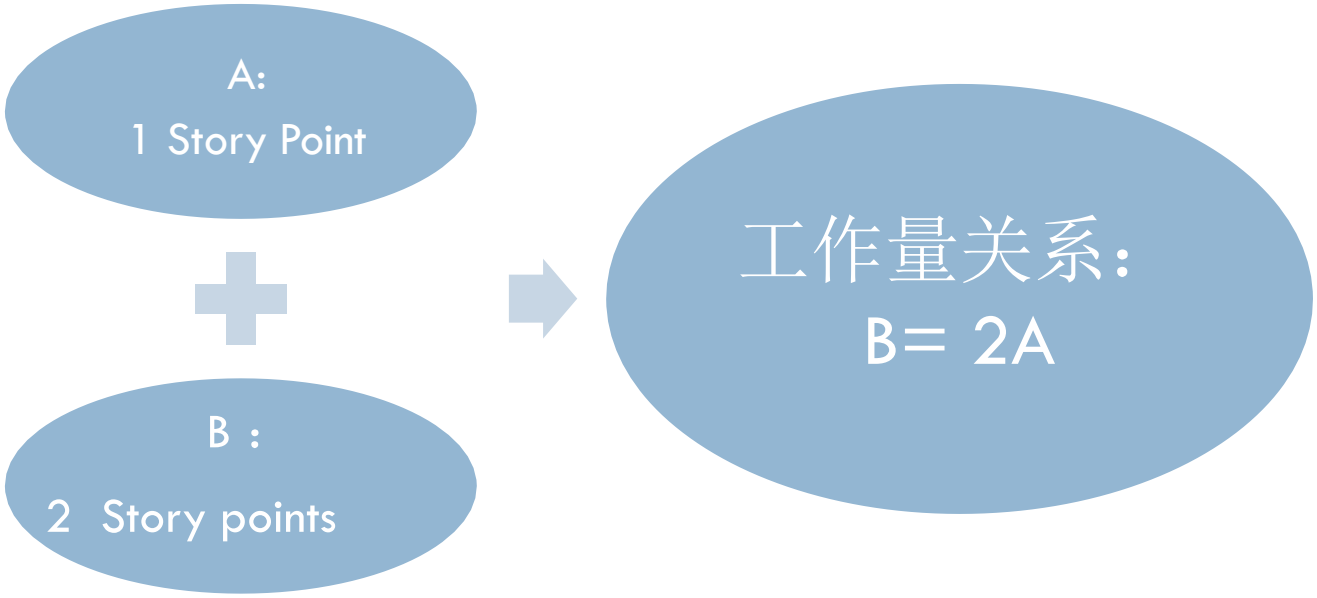
五 案例分析

敏捷估算思维

- 采用轻量级估算方法快速生成高层级估算
- 短期规划可以进行详细的估算

Story point估算方法

Story point（故事点）用来度量实现一个Story 需要付出的工作量的相对估算。



Story point估算-常用的两个标准

- Fibonacci: 0, 1, 2, 3, 5, 8, 13, 21,34, 55, 89
- Power of 2: 0, 1, 2, 4, 8, 16, 32, 64,128

Story point估算-Fibonacci 七个等级

0、1、2、3、5、8、13七个等级

1. 选取预估为3 story points 的Story
2. 将需要预估的story与选取的Story进行比较，
   2.1 如果两个工作量差不多,设置该story 的story point为3
   2.2 如果工作量略少，则为2，
3. 如果工作量更少，则为1，
4. 如果该story 不需要完成, 则设置为0。
5. 同理,如果略多/更多/再多,可以相应的设置为5/8/13。
6. 如果该story 超过13 story point,可以认为是Epic,可以再分解

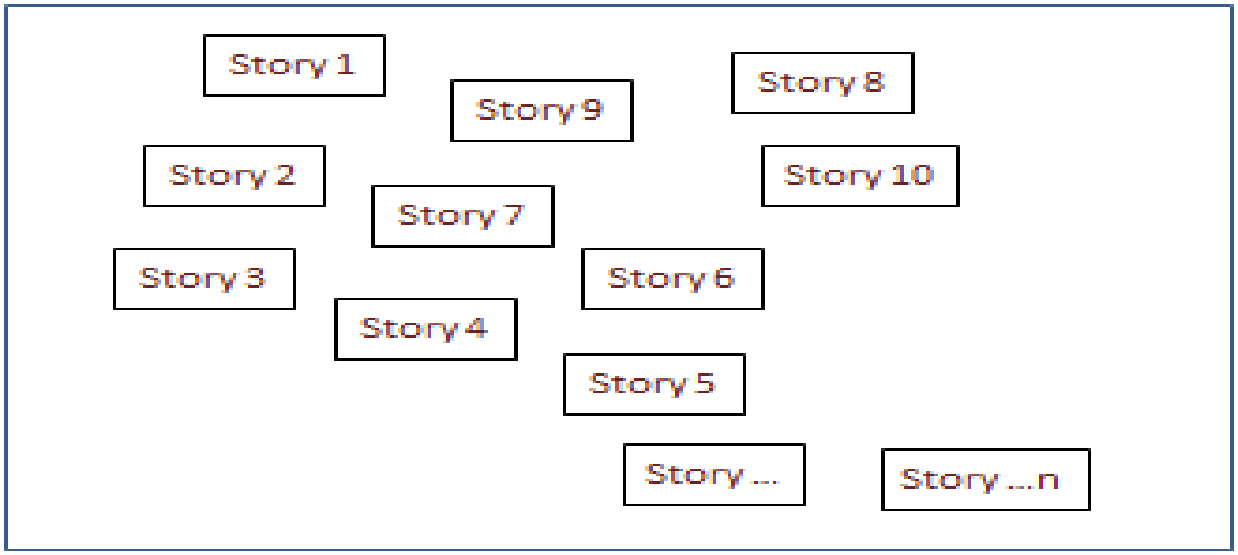
Story point估算- 举例

针对《SPM需求规格》：

- 预估：
  - 注册功能为3个story points
- 则估算：
  - 登录功能为2个story points
  - 人员管理功能为5个story points.

A Fast Story Point Estimation Process

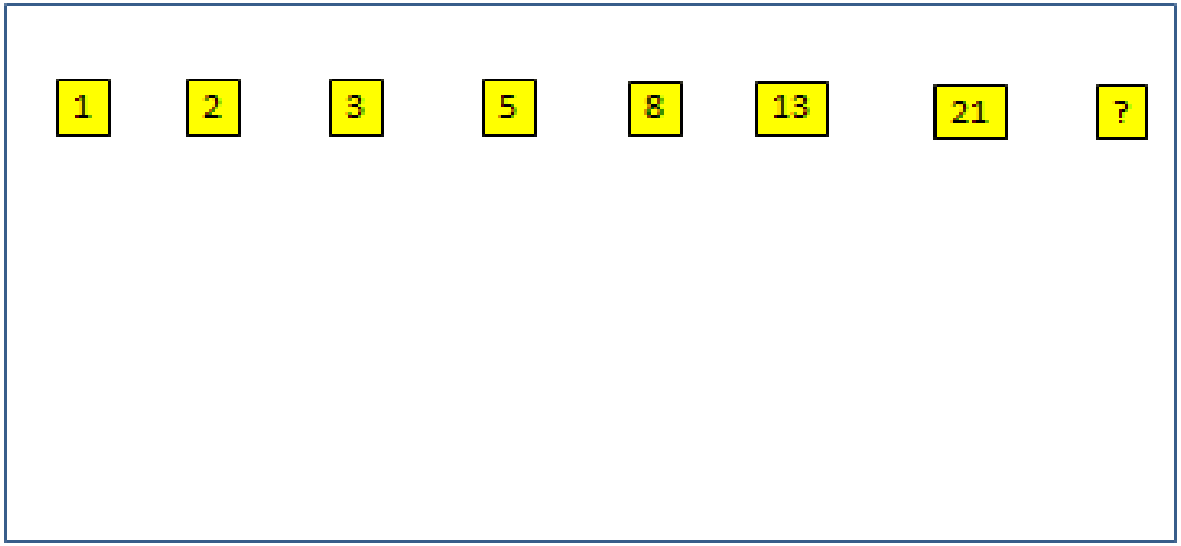
每个用户故事被独立打印，贴在墙上。



Stories are posted on part of the wall in advance of the meeting

A Fast Story Point Estimation Process

在墙上写下斐波那契数列：1-2-3-5-8-13-21并且加上一列“？”



A Fibonacci scale is posted on a section of the wall

A Fast Story Point EstimationProcess

① 团队人员排成一排

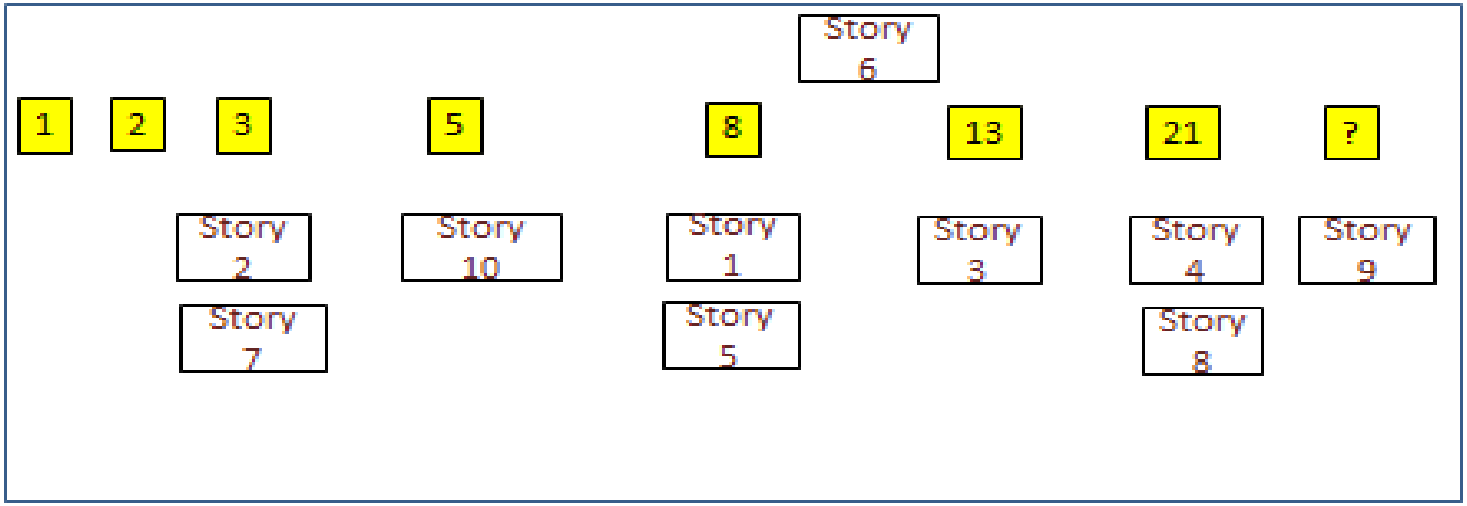
② 要求第一名成员把一个用户故事放到他认为可以正确反应故事点值的那一列上

③ 第一名成员做完后排团队成员的最后一个位置

④ 下一个团队成员可以挪动已经摆好的用户故事，也可以选择另外的用户故事，把它挪到他认为可以正确反应故事点值的那一列

⑤ 继续这个过程，直到所有用户故事都摆放完毕。

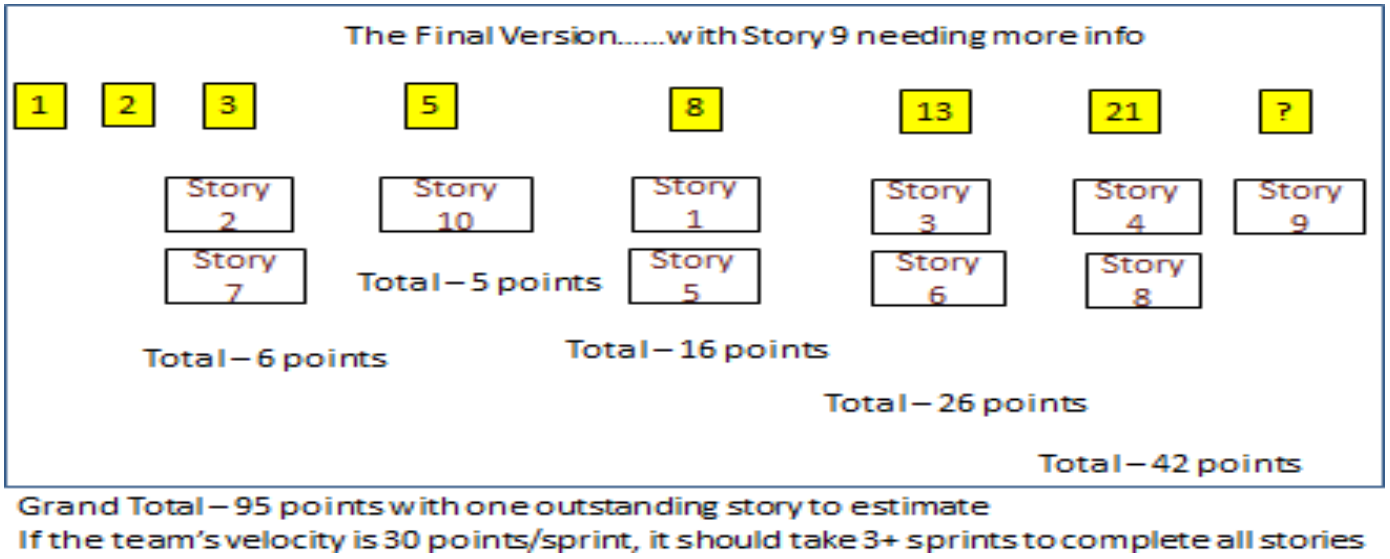
A Fast Story Point Estimation Process



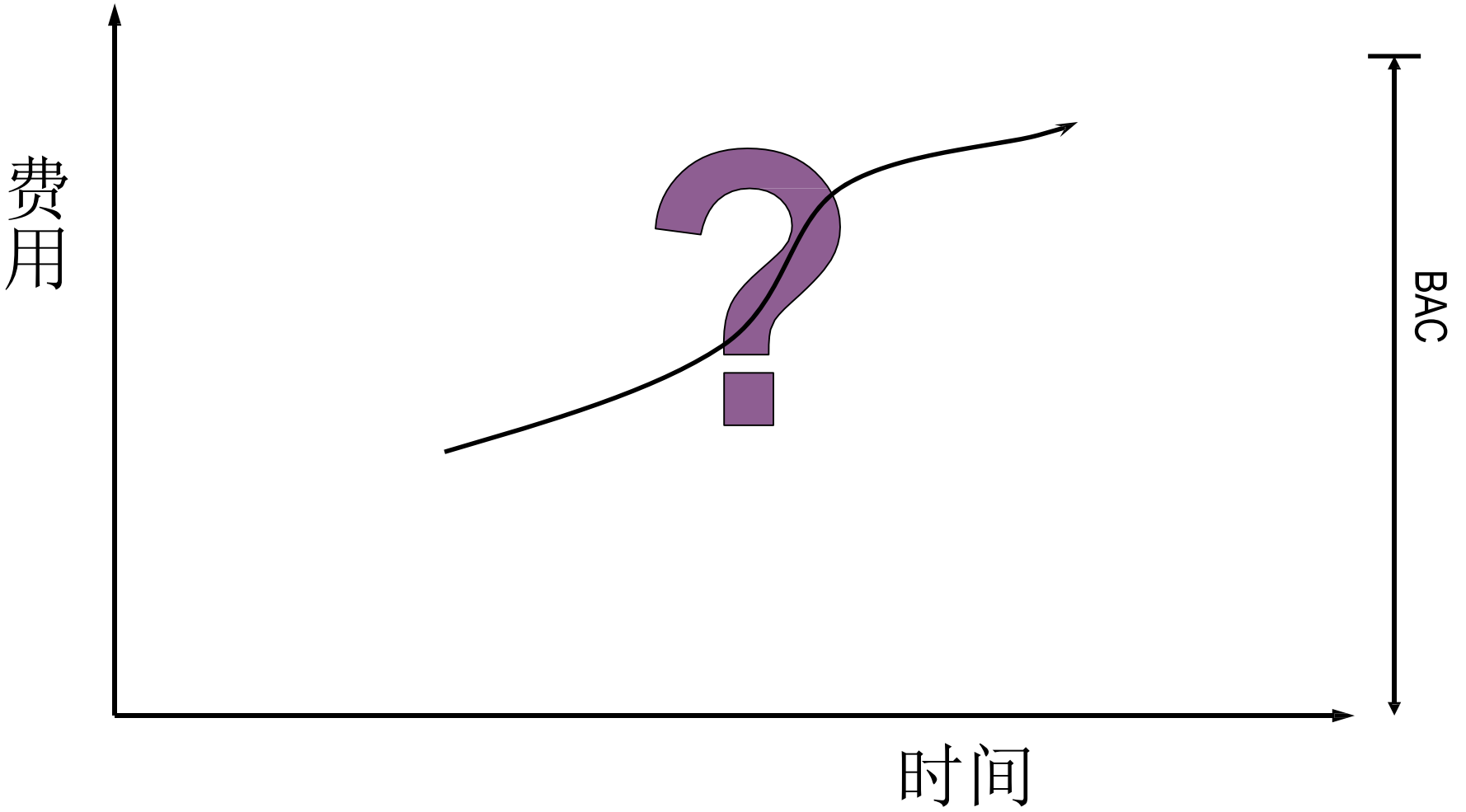
- 在此循环过程中，会有用户故事在不同的估值点列中来回挪动，引导师可以把这些用户故事挪到列表的上方，用于最后的时候讨论。
- 也会有一些故事需要更多的信息，把他们放到“？”一列

A Fast Story Point Estimation Process

- 一旦团队成员对放置的故事都满意，计算每一列故事的个数，并且乘以故事点，从而得到所有的故事点。



总估算成本（BAC)



本章要点

一 估算过程概念

二 传统估算方法

三 敏捷估算方法

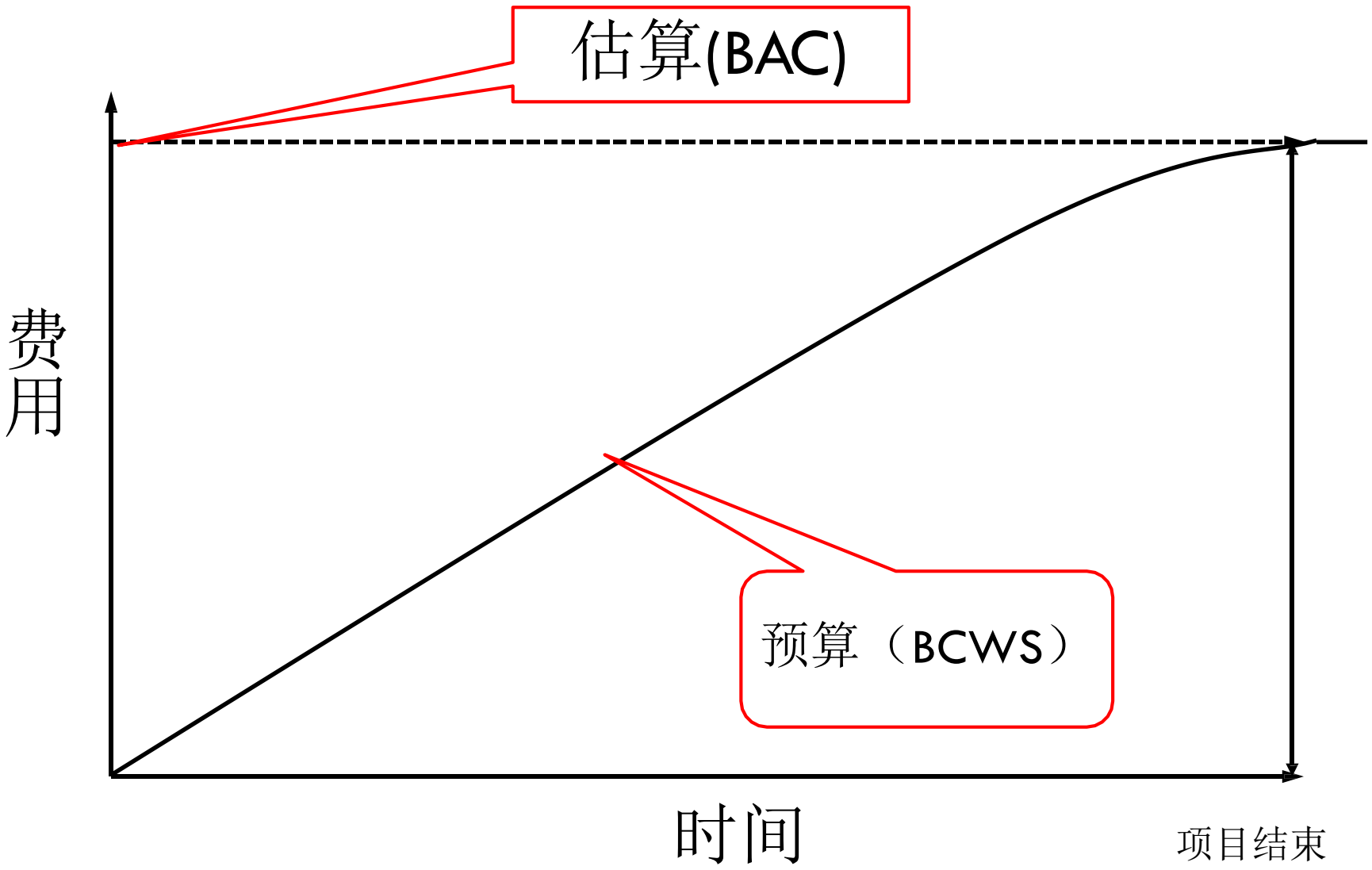
四 成本预算

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
- 成本预算的目的是产生成本基线

估算（BAC）与预算（BCWS）



项目成本预算

分配项目成本预算包括三种情况：

1. 给任务分配资源成本
2. 给任务分配固定资源成本
3. 给任务分配固定成本

给任务分配资源成本:常规方法

与资源的费率相关

- 标准费率
- 加班费率
- 每次使用费率
- 。 。 。 。 。 。

给任务分配资源成本：例子

	任务模式	任务名称	WBS
1		**SPM课程平台**	**1**
2		需求分析	1.1
3		概要设计	1.2
4		详细设计	1.3
5		**编码实施**	**1.4**
6		**客户端子系统**	**1.4.1**
7		**系统登录**	**1.4.1.1**
8		login.jsp	1.4.1.1.1
9		LoginAction.Java	1.4.1.1.2
10		LoginService.Java	1.4.1.1.3
11		LoginDAO.Java	1.4.1.1.4
12		**系统注册**	**1.4.1.2**
14		**选课模块**	**1.4.1.3**
16		成绩查询	1.4.1.4
17		网上测试	1.4.1.5
18		留言模块	1.4.1.6
19		学习进度管理	1.4.1.7
20		课程信息模块	1.4.1.8
21		**管理端子系统**	**1.4.2**
22		用户管理	1.4.2.1
23		课程资源管理	1.4.2.2
24		课程信息管理	1.4.2.3
25		界面	1.4.3
26		内部接口	1.4.4
27		**公共模块**	**1.4.5**

甘特图

就绪 新任务: 手动计划

3人天*1000元/人天=3000元

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，可以向任务分配固定资源成本。

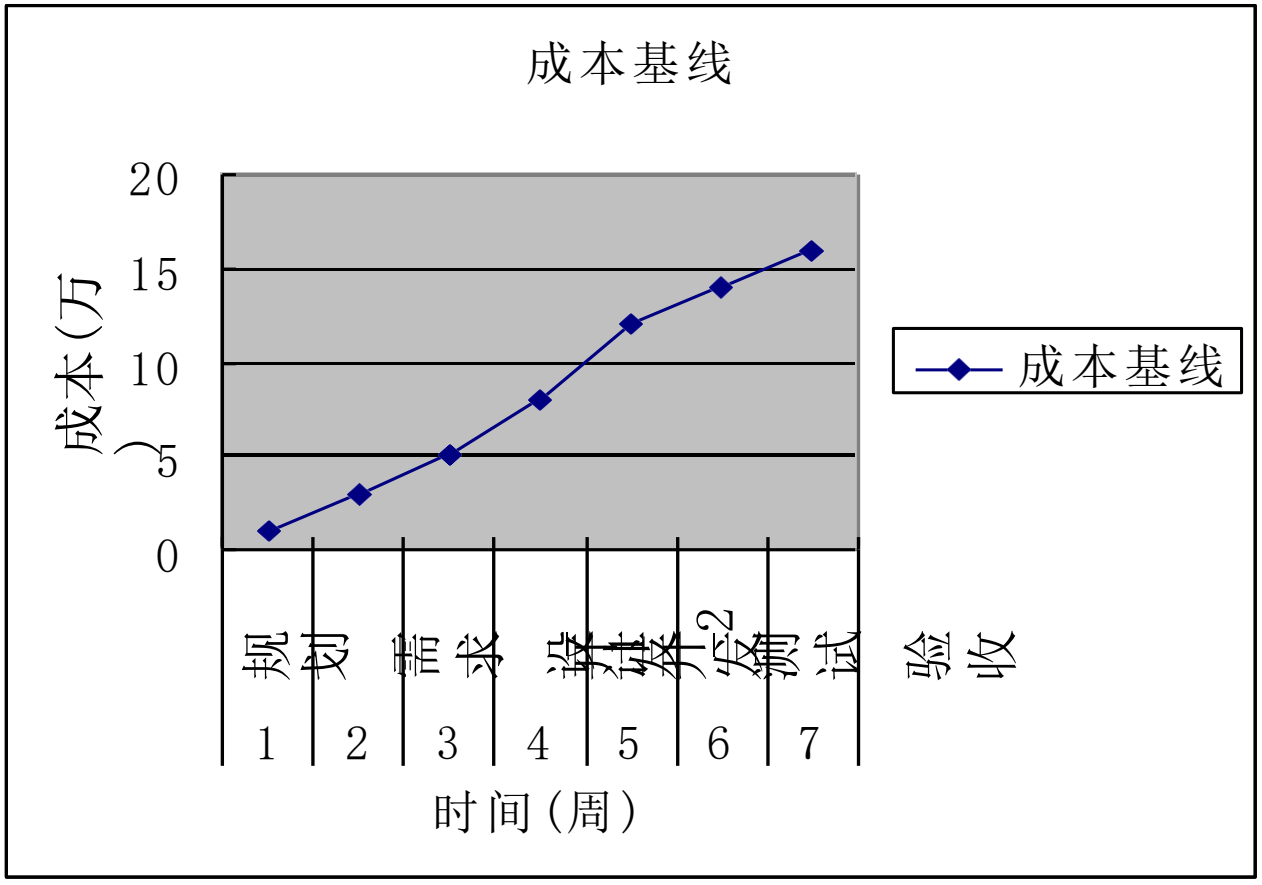
例如：项目中的一个兼职人员成本

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，管理者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，管理者向任务直接分配成本。

例如：某外包任务、培训任务

成本基线



本章要点

一 估算过程概念

二 传统估算方法

三 敏捷估算方法

四 成本预算

五 案例分析

医疗信息商务平台成本估算

工作量成本估算案例

- 自下而上的估算
- 用例点估算

MED自下而上的估算

表 6A-5：自下而上的估算

医疗信息商务平台			人天	小计	总计
F1：用户					396
	F1.1：注册			15	
		个人注册	4		
		组织注册	4		
		协会/学会注册	4		
		登录	3		
	F1.2：管理			26	
		用户信息	3		
		用户权限	8		
		统计分析	15		
F2：产品信息					
	F2.1：编辑			77	
		离线产品	23		
		在线产品	25		
		产品状态管理	29		
	F2.2：浏览			27	
		按厂商浏览	12		
		按产品分类浏览	6		
		按医院科别浏览	9		
	F2.3：查找			14	
F3：网上交易					
	3.1 售前			24	
		3.1.1 客户的分级优惠	12		
		3.1.2 购买数量的优惠	12		
	3.2 售中			30	
		3.2.1 询价	15		
		3.2.2 接受订单	15		
	3.3 售后			40	
		3.3.1 统计分析	20		
		3.3.2 查看采购记录	20		
F4：分类广告				50	
	4.1 与产品有关		9		
	4.2 与产品无关		10		
	4.3 匹配		15		
	4.4 招标		16		
F5：协会/学会				33	
	5.1 编辑		8		
	5.2 浏览		9		
	5.3 管理		16		
F6：医院管理				49	
	6.1 编辑		9		
	6.2 浏览		8		
	6.3 检索		16		
	6.4 管理		16		
F7：Email 管理（购买）		2.4 万	1	1	
F8：Chat 管理（现成）			2	2	
F9：护士排班表（修植）			5	5	
F10：联机帮助			3	3	

计算开发成本

1. 通过自下而上的计算，得知项目开发规模是396人天，开发人员成本参数=1000元/天，则内部的开发成本=1000元/天*396天=39.6万元
2. 加上外包部分软件成本2.4万元，则开发成本=39.6万+2.4万=42万元

计算直接、间接成本

管理成本＝开发成本*10%＝＝42万元*10%=4.2万元。

直接成本＝开发成本+管理成本＝42万元+4.2＝46.2万元

间接成本＝直接成本*20%=46.2万元*20%=9.24万元

计算总估算成本

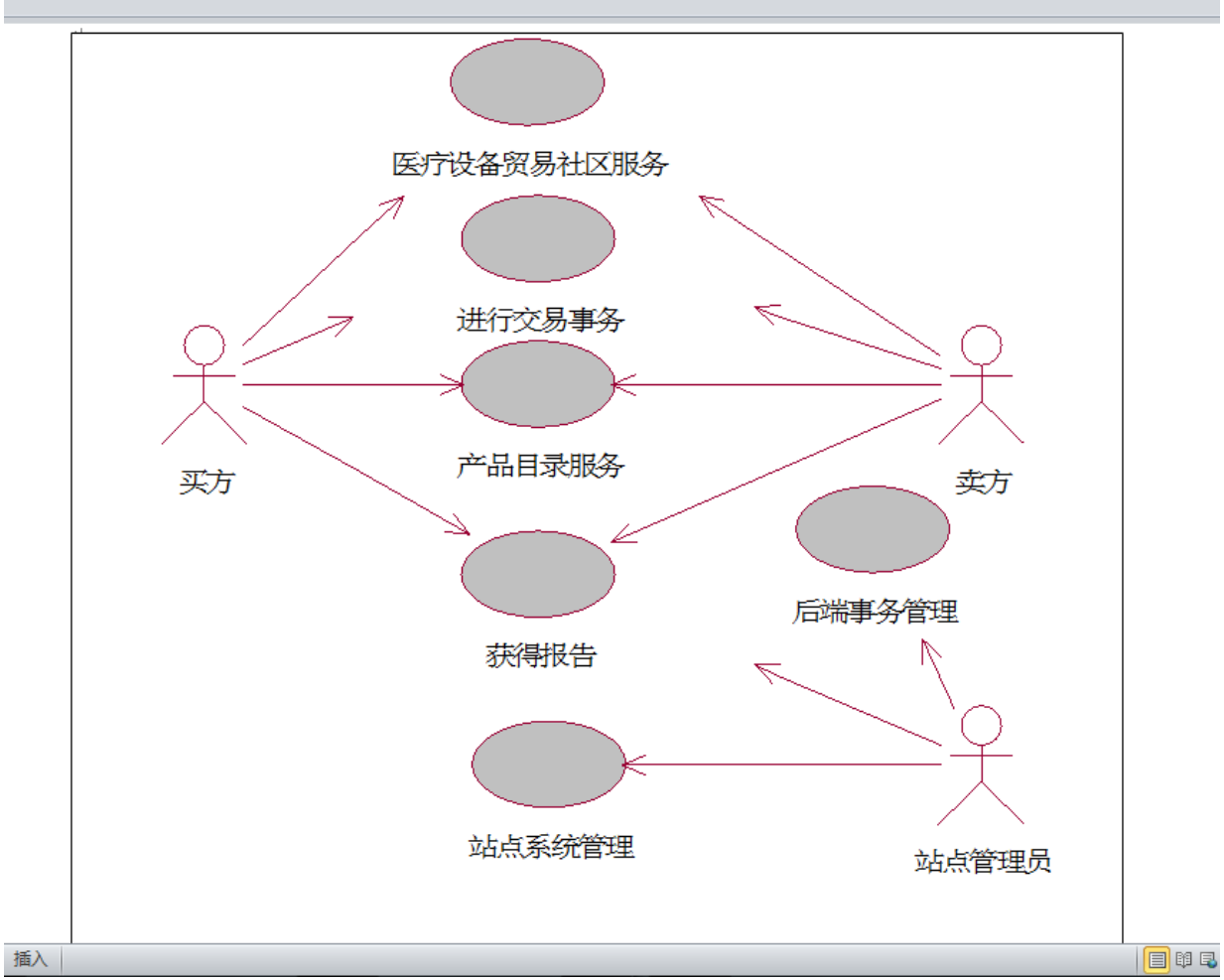
项目总估算成本=直接成本+间接成本=
46.2万元+9.24万元=55.44万元。

医疗信息商务平台成本估算

工作量成本估算案例

- 自下而上的估算
- 用例点估算

MED用例点估算



角色：User

表 4-1 用户类型及角色表

用户类型	用户子类	角色	子角色
Medeal.com		Webmaster	
		内容管理经理	
		市场部经理	
组织	交易组织	厂商	管理者
			内容管理经理
			内容管理者
			销售经理
			销售人员
			决策人
			市场分析员
			合同履行人员
			售后服务经理
			售后服务人员
			一般人员
		经销商	管理者
			内容管理经理
			内容管理者
			销售经理
			销售人员
			决策人
			市场分析员
			合同履行人员
			售后服务经理
			售后服务人员
			采购经理
			采购员
			一般人员
		医院	管理者
			决策人
			采购经理
			采购员
			护士长
			护士
			一般人员
	非交易组织	协会/学会	管理者
			编辑者
			会员
个人		成员	
		非成员	

计算未调整的角色的权值：UAW

UAW=18

表6A-1　UAW计算过程

序号	Actor 复杂度级别	权值	参与角色 Actor 数	UAWi
1	simple	1	1	1
2	average	2	7	14
3	complex	3	1	3
总计				18

计算未调整的用例权值：UUCW

UUCW=240

表6A-2　UUCW计算过程

序号	Use case 复杂度级别	权值	Use case 数量	UUCWi
1	simple	5	15	75
2	average	10	12	120
3	complex	15	3	45
总计				240

计算未调整的用例点：UUCP

UUCP=UAW+UUCW=18+240=258

技术复杂度因子TCF

序号	技术因子	权值	Value 值	TCFi
1	TCF1	2.0	3	6.0
2	TCF2	1.0	5	5.0
3	TCF3	1.0	3	3.0
4	TCF4	1.0	5	5.0
5	TCF5	1.0	3	3.0
6	TCF6	0.5	3	1.5
7	TCF7	0.5	5	2.5
8	TCF8	2.0	3	6.0
9	TCF9	1.0	5	5.0
10	TCF10	1.0	3	3.0
11	TCF11	1.0	5	5.0
12	TCF12	1.0	3	3.0
13	TCF13	1.0	0	0.0
	TCF	0. 6+(0.01×ΣTCFi) =1.08		

环境因子ECF

序号	环境因子	权值	Value值	ECFi
1	ECF1	1.5	3	4.5
2	ECF2	0.5	3	1.5
3	ECF3	1.0	3	3.0
4	ECF4	0.5	5	2.5
5	ECF5	1.0	3	3.0
6	ECF6	2.0	3	6.0
7	ECF7	1.0	0	0.0
8	ECF8	1.0	0	0.0
	ECF	$1.4+(-0.03\times\Sigma ECFi)=0.785$		

计算用例点UCP

UCP =UUCP×TCF×ECF =258×1.08×0.785 =218.7

规模： Effort

如果：PF=20工时/用例点

则：Effort =UCP×PF = 218.7×20 =4374工时。

因为1人天=8工时，则项目规模：4374/8 =547人天

如果1000元/人天，则成本54.7万

本章小结

成本估算

1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比 (自顶向下)估算法
5. 自下而上估算法
6. 参数估算法
7. 专家估算法
8. 三点估算法
9. 敏捷方法

成本预算