《软件开发过程与项目管理》

Software Development Process and Project Management

任课教师：范国祥

电　　话：0451-86418876-811(O)

13199561265(Mobile)

邮　　箱：fgx@hit.edu.cn

哈工大计算学部

国家示范性软件学院

软件工程教研室

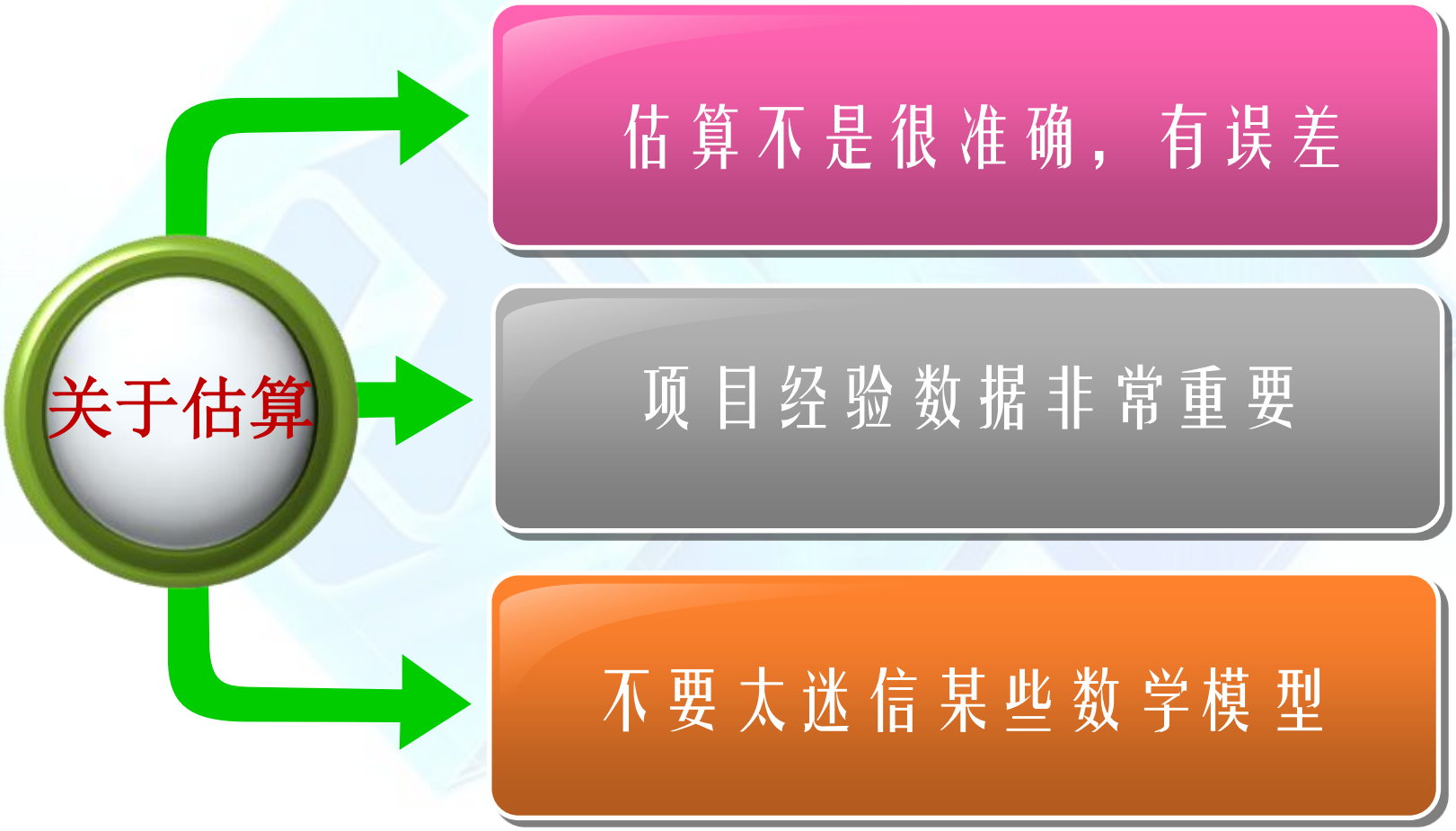
2023. 10

软件成本估算是老大难问题

成本度量——

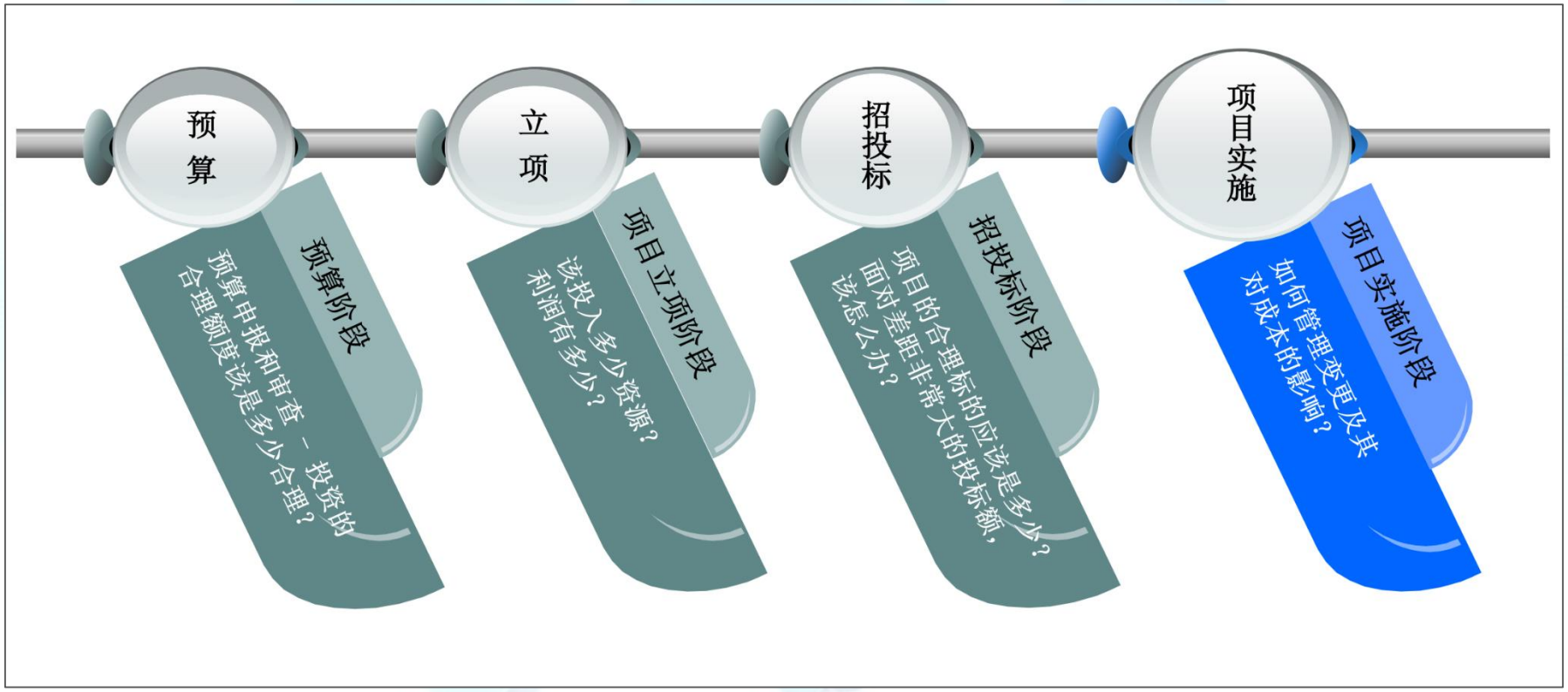
软件产业老大难问题

关于估算



软件项目管理-成本估算

何时进行估算



估算过程

规模估算:	【功能点】	1000		
生产率PE:	【小时/功能点】	10	15	20
工作量估算:	【人月】	41.67	62.50	83.33
人工单价:	【万元/人月】	2.60	1.80	1.50
成本估算:	【万元】	108.33	112.50	125.00

国标《软件开发成本度量规范》核心内容

1. 软件研发成本构成

本标准中依据财务惯例将软件研发成本分为**直接成本和间接成本**，同时考虑到软件行业的特性，将直接成本和间接成本分为人力成本和非人力成本，同时明确了各种成本的定义和计算方式。

2. 估算过程

本标准中定义的软件研发成本估算过程包括**规模估算、工作量估算和成本估算**三部分。其中，估算软件规模时采用国际标准的功能点方法，而工作量则根据不同情况，可选择采用方程法、类比法或类推法进行估算。

工作量=软件规模*软件因素调整因子*开发因素调整因子*生产率

- 软件因素包括规模、质量要求、应用类型、业务领域等
- 开发因素包括采用技术、过程、团队经验等

软件研发成本=工作量*人月费率+直接非人力成本

国标《软件开发成本度量规范》软件研发成本构成

直接人力成本估算是软件研发成本估算中的难点！

软件项目管理-成本估算

软件规模度量单位

- LOC（Lines of Code）
  - 代码行：用源代码长度的测量
- FP（Function Points）
  - 功能点：用系统的功能点数量来测量
- UCP（Use Case Points）
  - 用例点：用系统的用例点数量来测量

软件工作量度量单位

- 人天/人周/人月/人年
  - 开发团队每个人工作天数/周数/月数/年数的累计数

软件成本度量单位

- RMB（元或万元），$（元或万元），等等

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投入
- 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：RMB（元或万元），$（元或万元），等等

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. 类比估算法
5. 自下而上估算法
6. 三点估算法
7. 专家估算法

传统估算方法-代码行估算法

aaa.bat | Unit1.pas

```
299
300     //显示抽取到的学生照片。说明：如果抽取的不是1个班级，或者不是1个学生，则只显示
301     //抽到的第1个班级的第1个学生的照片
302     SName:=StudentName[ClassResultNo[1],StudentResultNo[1]];
303     if SName='XXX' then www:='JPGblank'  //www为资源文件的关键字，资源文件JPGblank意思为blank.jpg
304     else
305           begin
306             if(StudentOrderInClass[iii]>99)then
307                 www:='JPG'+StudentNoCommonPart+IntToStr(StudentClassNo[iii])+inttostr(
                    StudentOrderInClass[iii])
308             else
309                 if(StudentOrderInClass[iii]>9)then
310                     www:='JPG'+StudentNoCommonPart+IntToStr(StudentClassNo[iii])+'0'+inttostr(
                        StudentOrderInClass[iii])
311                 else
312                     www:='JPG'+StudentNoCommonPart+IntToStr(StudentClassNo[iii])+'00'+inttostr(
                        StudentOrderInClass[iii])
313           end;
314     DisplayPicture(www);
315
316 end;
317
318 procedure TForm1.DisplayPicture(PictureFileName:string);
319 //显示给定资源文件名的照片
320 var Stream:TStream;
321 var MyJpg:TJpegImage;
322 begin
323
324     Application.ProcessMessages();  //使其他控件响应消息，从而解决了程序由于连续循环显示界面内容
325                                     //或者是连续点击功能按钮导致的卡顿现象
326                                     //经验之谈：当界面元素如编辑框、图片等控件中内容反复刷新动态
327                                     //显示时，会出现死机现象
328
329     Stream:=TResourceStream.Create(HINSTANCE,PictureFileName,'JPEG');
330     Try
331       MyJpg:=TJpegImage.Create;
332       Try
333         MyJpg.LoadfromStream(Stream);
```

Pascal source file | length : 31,540 lines : 958 | Ln : 115 Col : 19 Pos : 3,830 | Windows (CR LF) | GB2312 (简体中文) | INS

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，其他阶段无代码产生

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. 类比估算法
5. 自下而上估算法
6. 三点估算法
7. 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
- 估算软件规模，即软件规模度量单位

传统估算方法- Albrecht功能点估算

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

功能点公式

$$FP = UFC \times TCF$$

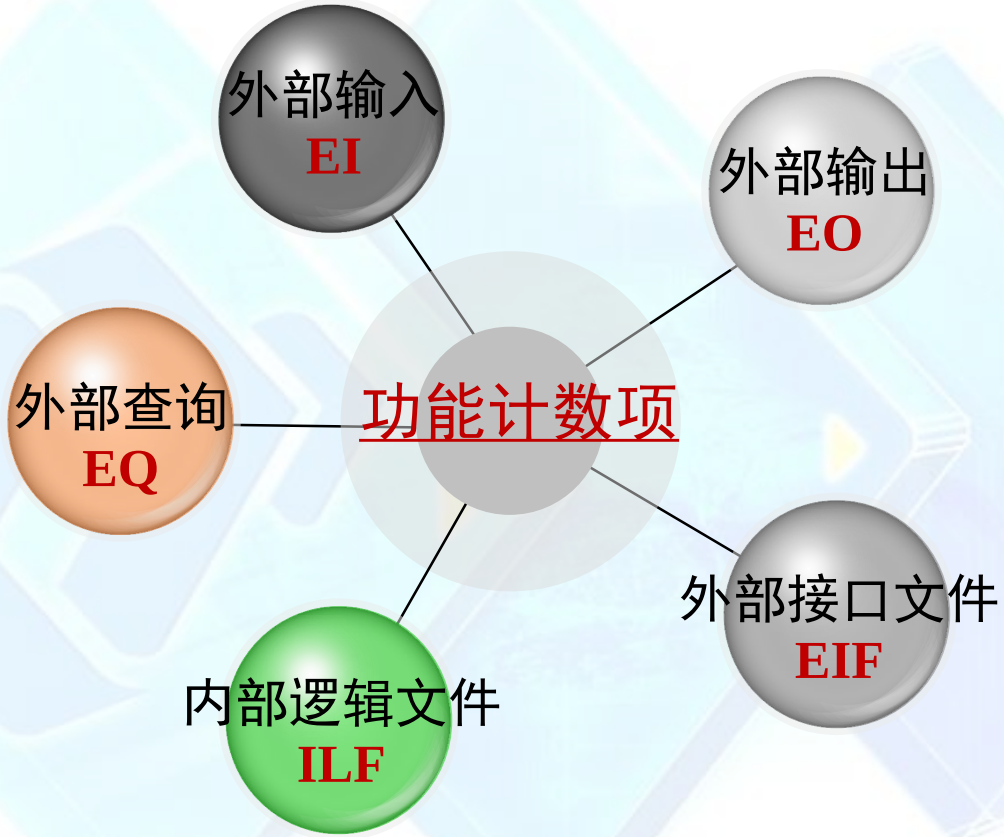
其中：FP – Function Points 功能点

UFC – Unadjusted Function Component 未调整的功能点计数)

TCF – Technical Complexity Factor 技术复杂度因子

注：功能点≠软件系统中的可以运行的功能模块！

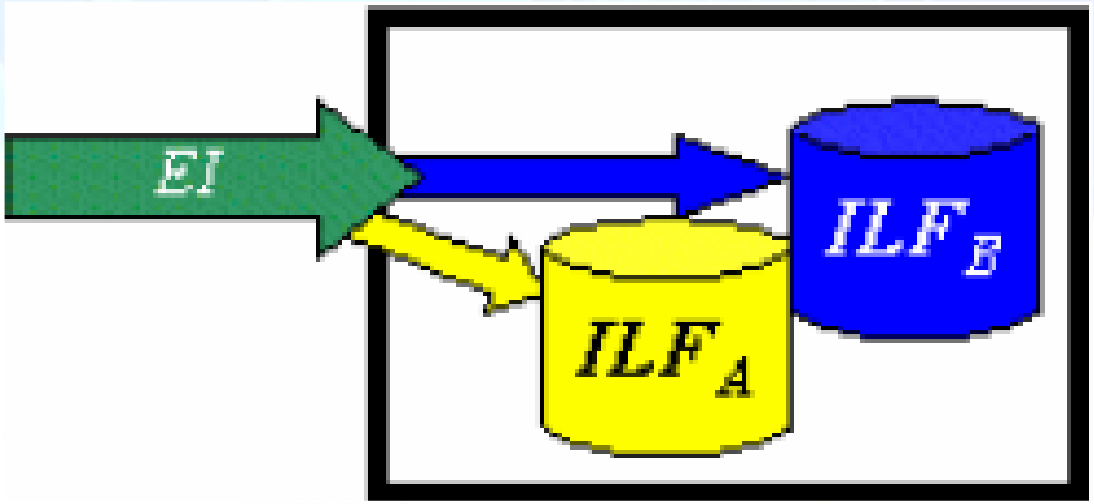
UFC-未调整的功能点计数项



- 不同类型的系统功能，其开发复杂度（TCF）是不同的
- 针对未来系统，需要划分一个“系统边界”
- Albrecht将功能点计数项分为上图所示5大类
- 根据经验，一般来说：$TCF_{ILF} > TCF_{EIF} > TCF_{EO} > TCF_{EI} \approx TCF_{EQ}$

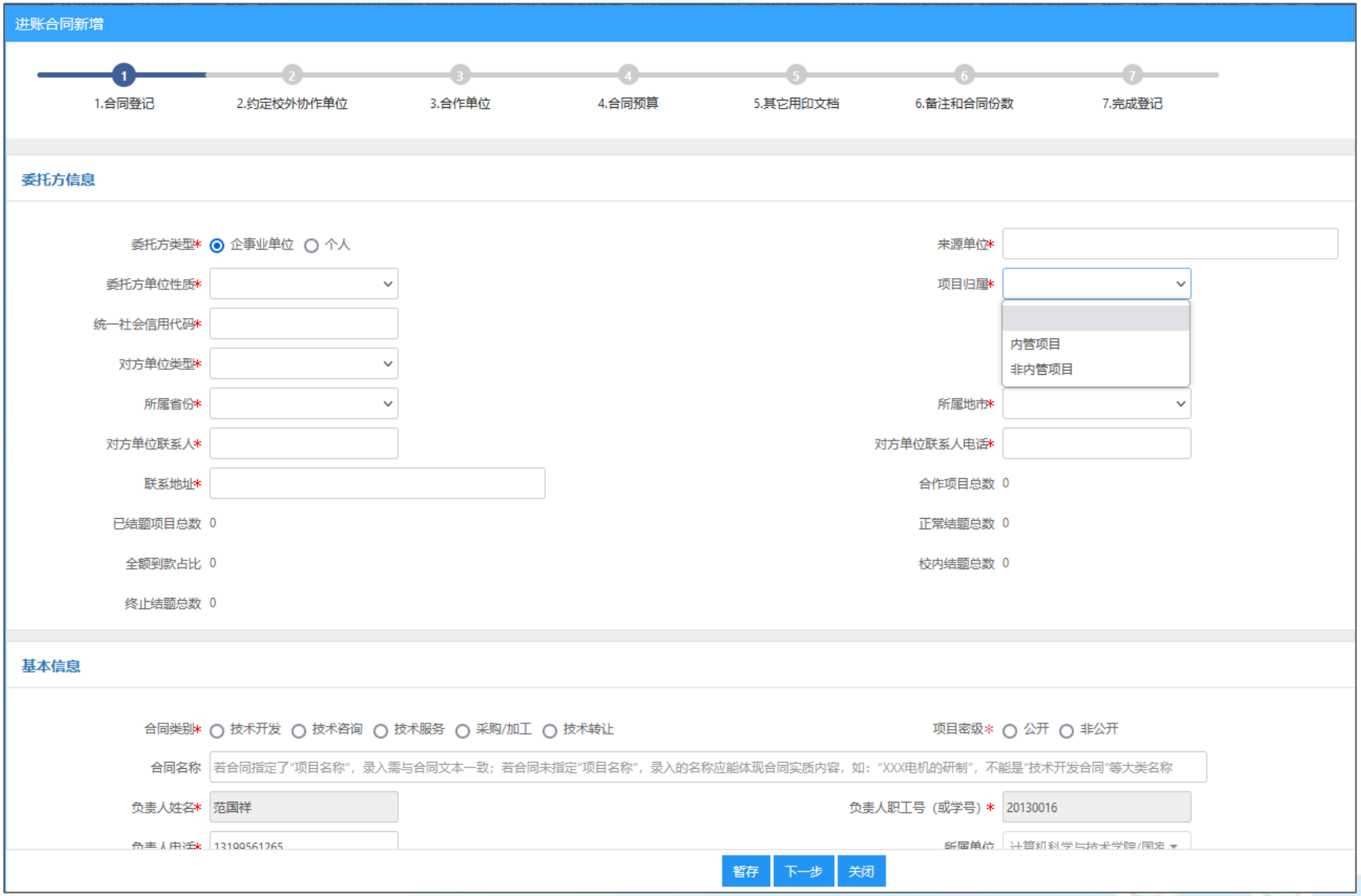
外部输入(External Inputs: EI)

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



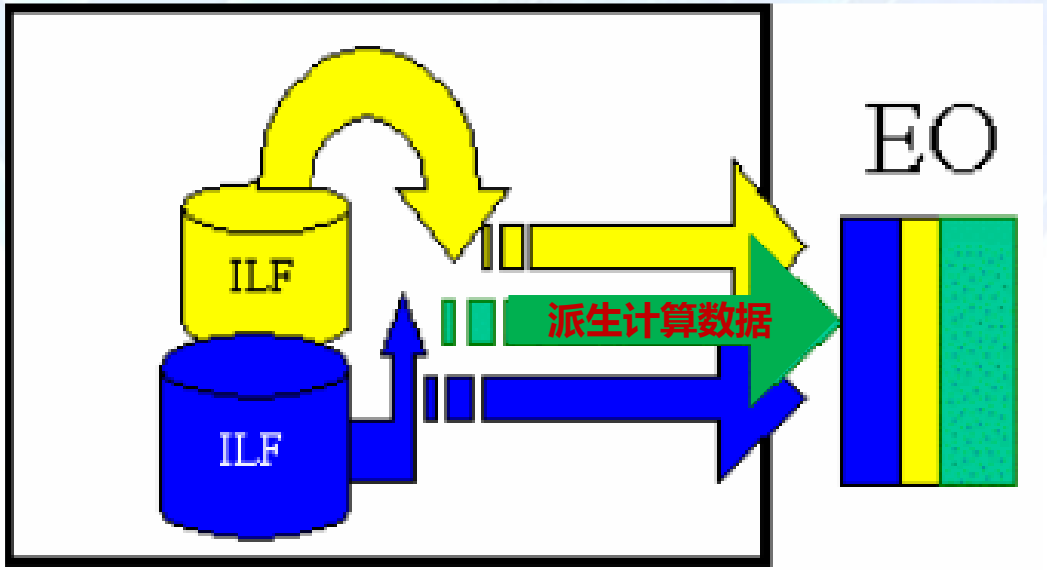
软件项目管理-成本估算

外部输入EI示例



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

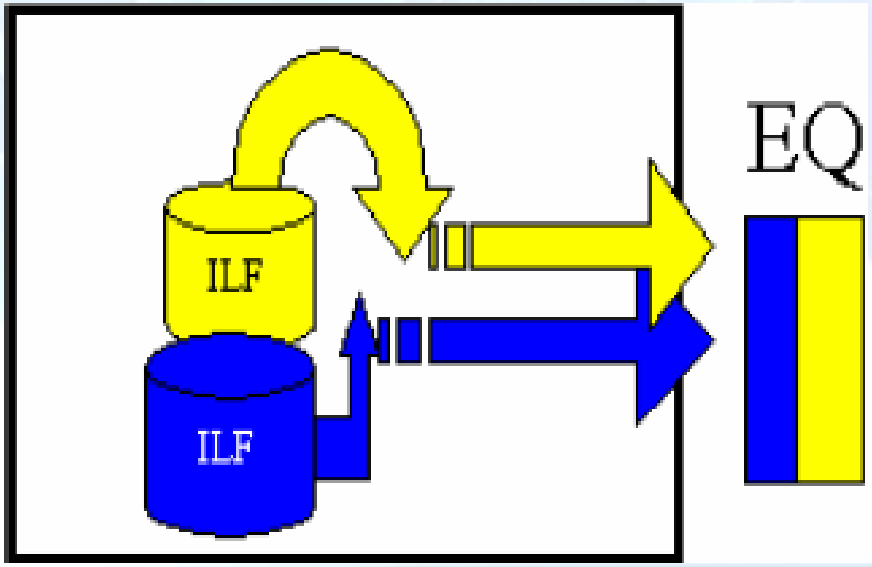


软件项目管理-成本估算

外部输出EO示例

外部查询(External Query EQ)

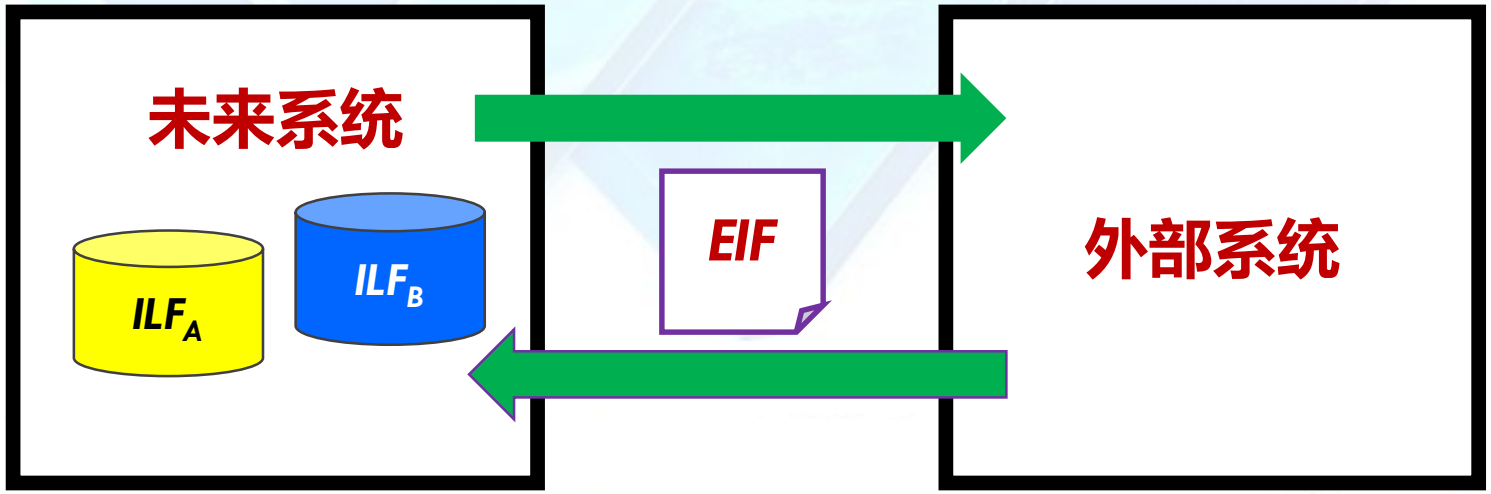
- 外部查询是一个输入引出一个即时的**简单输出**
- **没有处理过程**，即不会产生任何新的数据



外部查询EQ示例

外部接口文件（External Interface Files EIF）

- 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外部系统，或从外部系统取回接口信息

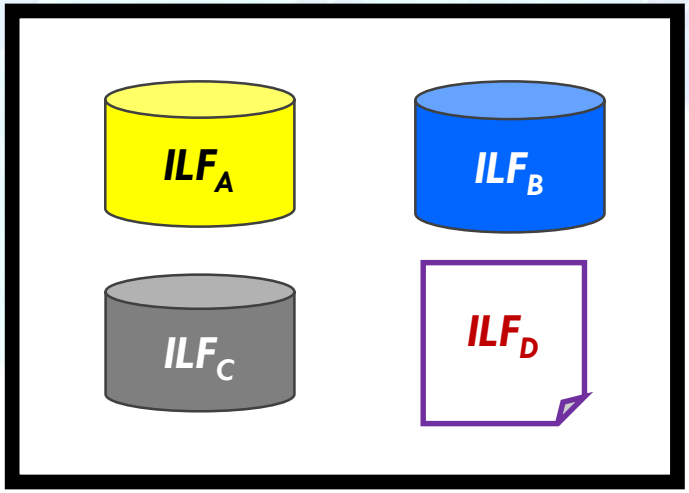


软件项目管理-成本估算

外部接口文件EIF示例

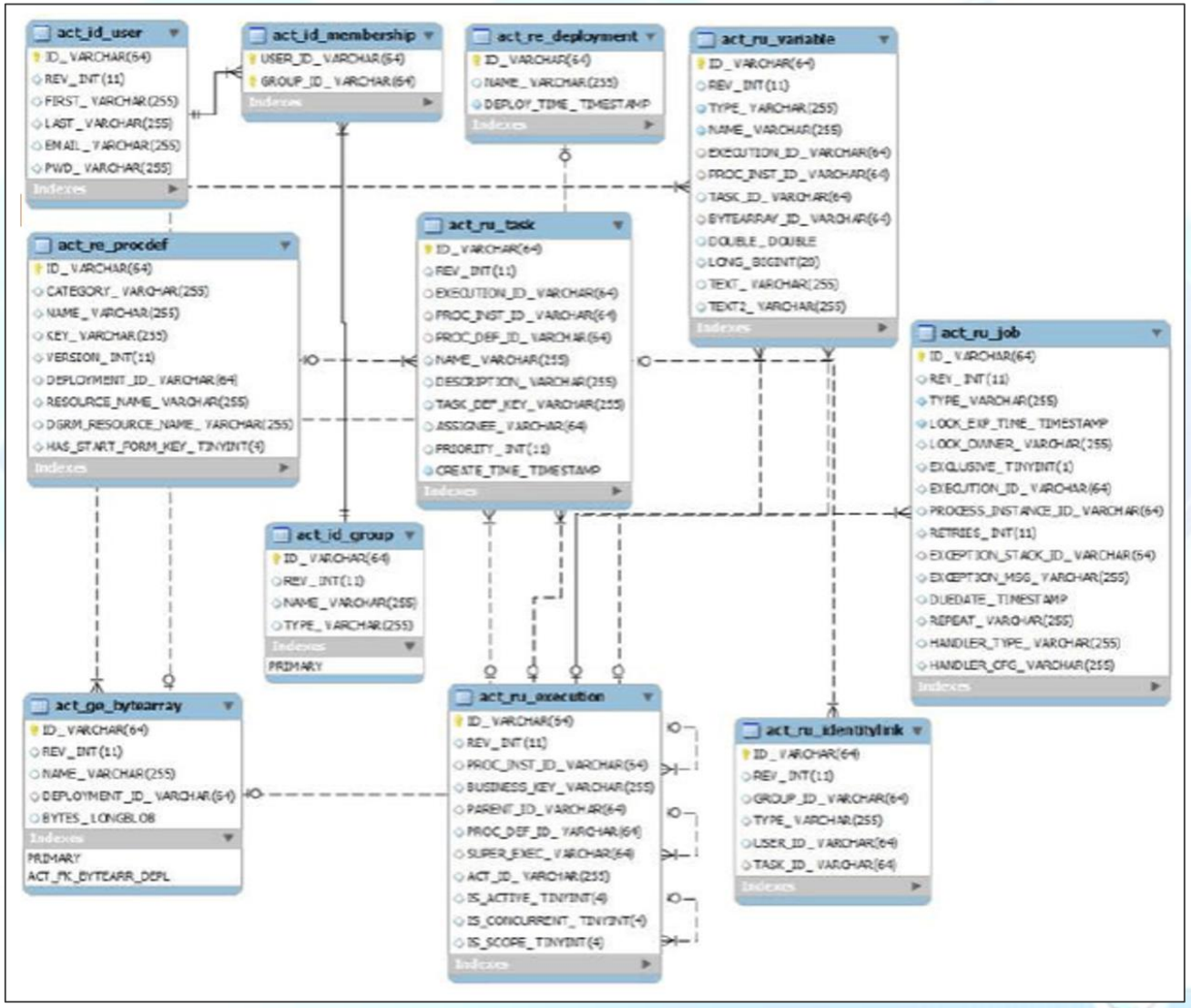
内部逻辑文件(Internal Logical Files： ILF)

- 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
- 通过外部输入进行改变，在系统内部进行维护的数据
- 例如：数据库表、数据结构等



chapter__6

内部逻辑文件ILF示例



功能点(FP)计数的规则和标准

- 国际功能点用户组织(IFPUG：International Funciton Point User Group)发布FP计数标准
- 《IFPUG功能点估算方法使用指南》
  - 未调整功能点UFC的计数项复杂度定级标准
  - 软件项目技术复杂度TCF因子及取值标准

未调整功能点UFC的计数项复杂度定级标准

EI(外部输入)的定级表

引用的文件类型个数	数据元素		
	1～4	5～15	> 15
0～1	低	低	低
2	低	中	高
≥3	中	高	高

EO(外部输出)和 EQ(外部查询)公用的定级表

引用的文件类型个数	数据元素		
	1～5	6～19	> 19
0～1	低	低	中
2～3	低	中	高
> 3	中	高	高

EI(外部输入)、EO(外部输出)和 EQ(外部查询)的定级取值表

级数	值(计算功能点数的权值)		
	EO(外部输出)	EQ(外部查询)	EI(外部输入)
低	4	3	3
中	5	4	4
高	7	6	6

未调整功能点UFC的计数项复杂度定级标准

ILF(内部逻辑文件)或者 EIF(外部接口文件)的定级表

记录元素类型个数	数据元素		
	1～19	20～50	＞50
1	低	低	中
2～5	低	中	高
＞5	中	高	高

ILF(内部逻辑文件)或者 EIF(外部接口文件)的定级取值表

级数	值(计算功能点数的权值)	
	ILF（内部逻辑文件）	EIF（外部逻辑文件）
低	7	5
中	10	7
高	15	10

未调整功能点UFC的计数项复杂度定级标准

- 前述表合并得到各类型功能计数项复杂度级别与系数取值表
- 数值为功能点计数项对UFC(未调整功能点)计算的功能复杂度加权系数

不同类型功能计数项复杂度级别与系数取值表

功能类型	功能复杂度定级取值(计算功能点数的权值)		
	低（简单）	中（一般）	高（复杂）
外部输入 EI	3	4	6
外部输出 EO	4	5	7
外部查询 EQ	3	4	6
外部接口文件 EIF	5	7	10
内部逻辑文件 ILF	7	10	15

软件项目技术复杂度TCF因子及取值标准

- 未来系统的各种特性，决定了开发的技术复杂度
- IFPUG制定了技术复杂度TCF的系统特性及取值标准

(功能点估算法）技术复杂度因子

通用特性		描述
F1	数据通信	多少个通信设施应用或系统之间辅助传输和交换信息
F2	分布数据处理	分布的数据和过程函数如何处理
F3	性能	用户要求响应时间或者吞吐量吗？
F4	硬件负荷	应用运行在的硬件平台工作强度如何？
F5	事务频度	事务执行的频率(天、周、月)如何？
F6	在线数据输入	在线数据输入率是多少？
F7	终端用户效率	应用程序设计考虑到终端用户的效率吗？
F8	在线更新	多少内部逻辑文件被在线事务所更新
F9	处理复杂度	应用有很多的逻辑或者数据处理吗？
F10	可复用性	被开发的应用要满足一个或者多个用户需要吗？
F11	易安装性	升级或者安装的难度如何？
F12	易操作性	启动、备份、恢复过程的效率和自动化程度如何？
F13	跨平台性	应用被设计、开发和支持被安装在多个组织的多个安装点（不同的安装点的软硬件平台环境不同）吗？
F14	可扩展性	应用被设计、开发以适应变化吗？

(功能点估算法)
技术复杂度因子的取值

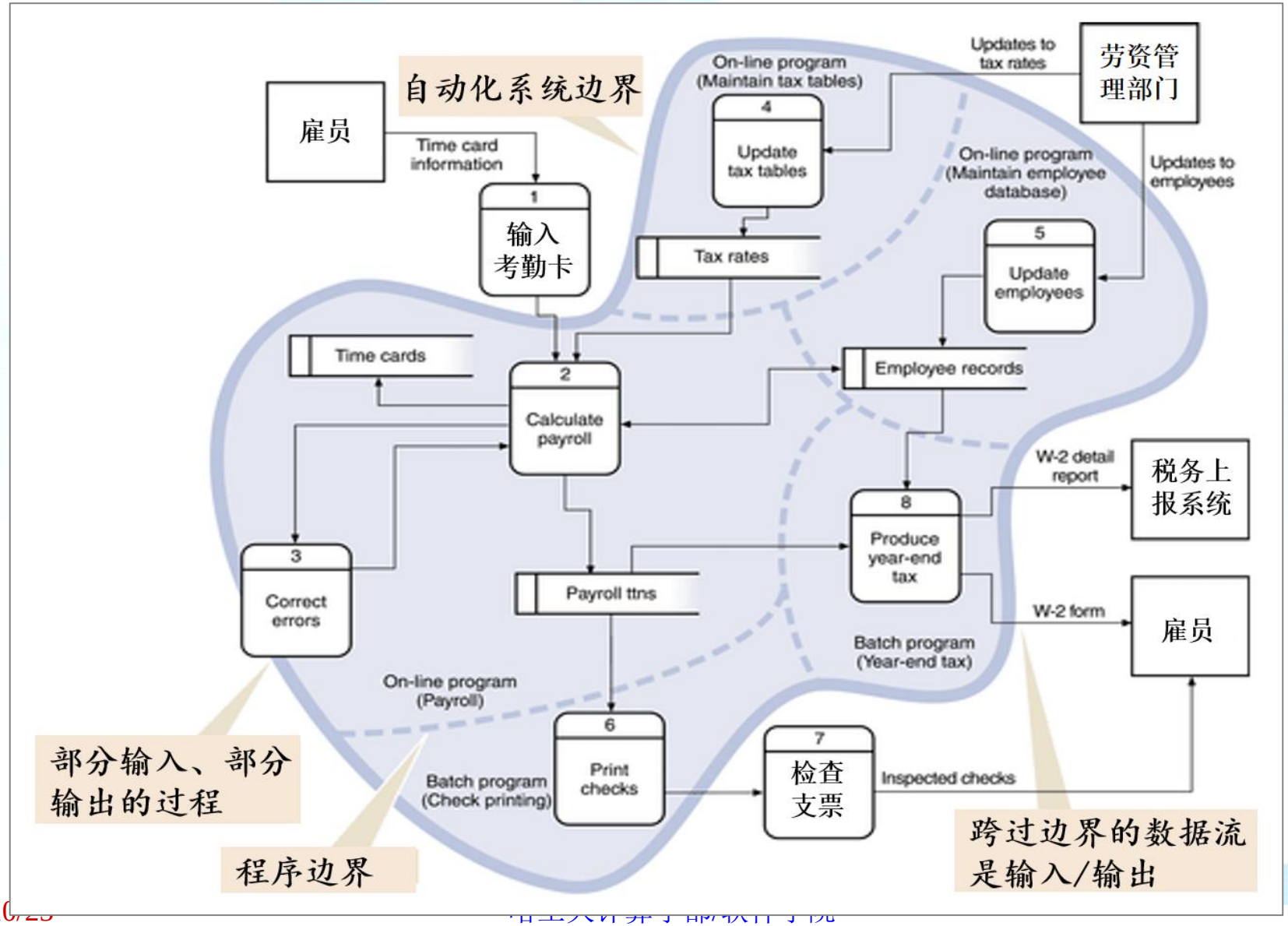
调整系数	描述
0	不存在或者没有影响
1	不显著的影响
2	相当的影响
3	平均的影响
4	显著的影响
5	强大的影响

显然，Fi的取值越小越好
Fi越小，表明技术复杂度越低

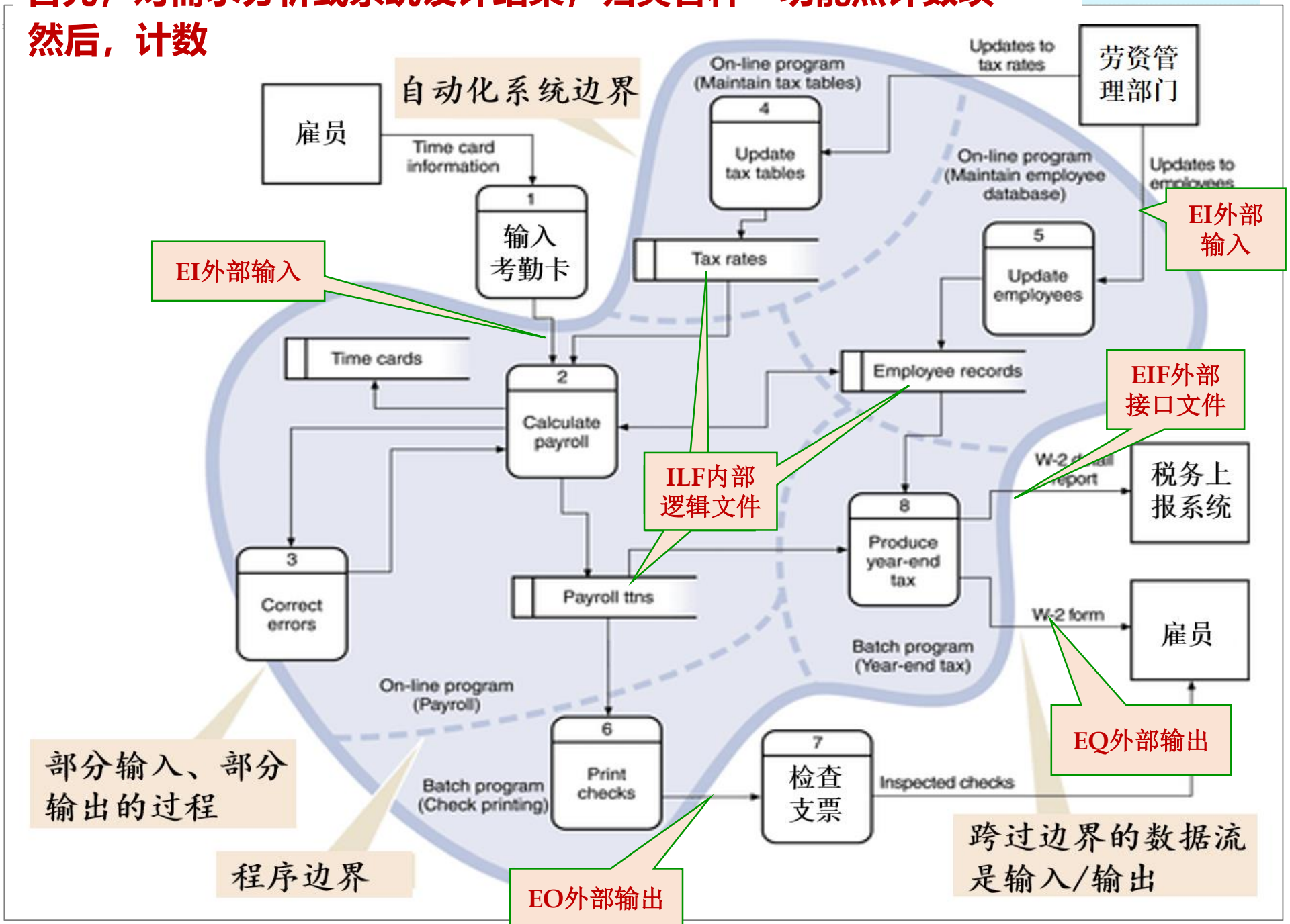
- 给出了TCF计算公式： $TCF = 0.65 + 0.01 \times \sum_{i=1 \text{ to } 14} F_i$
- 其中F_i取值范围[0, 5]，显然TCF的计算结果范围[0.65, 1.35]

软件项目管理-成本估算

功能点估算方法举例



首先，对需求分析或系统设计结果，归类各种“功能点计数项”
然后，计数



功能点估算方法举例

假设，经过对某软件的需求分析，得到如下关于功能计数项的数据：
外部输入EI：11个；外部输出EO：14个；外部查询EQ：6个；
外部接口文件EIF：10个；内部逻辑文件ILF：11个。
计数项的复杂度分布情况如下表：

软件需求的功能计数项

功能类型 \ 复杂度	低（简单）	中（一般）	高（复杂）
外部输入 EI	6	2	3
外部输出 EO	7	7	0
外部查询 EQ	0	2	4
外部接口文件 EIF	5	2	3
内部逻辑文件 ILF	9	0	2

软件项目管理-成本估算

功能点估算方法举例

计算“未调整的功能点UFC”

方法：查表【*各类型功能计数项复杂度级别与系数取值表*】，用已知的功能计数项个数×复杂度系数，最后累加

详细计算过程如下表：

不同类型功能计数项复杂度级别与系数取值表

功能类型	功能复杂度定级取值(计算功能点数的权值)		
	低（简单）	中（一般）	高（复杂）
外部输入 EI	3	4	6
外部输出 EO	4	5	7
外部查询 EQ	3	4	6
外部接口文件 EIF	5	7	10
内部逻辑文件 ILF	7	10	15

功能类型	不同复杂度功能点计算结果									
	低（简单）			中（一般）			高（复杂）			UFC合计
	复杂度系数	功能数	UFC	复杂度系数	功能数	UFC	复杂度系数	功能数	UFC	
外部输入 EI	3	6	18	4	2	8	6	3	18	44
外部输出 EO	4	7	28	5	7	35	7	0	0	63
外部查询 EQ	3	0	0	4	2	8	6	4	24	32
外部接口文件 EIF	5	5	25	7	2	14	10	3	30	69
内部逻辑文件 ILF	7	9	63	10	0	0	15	2	30	93
UFC =			134	+		65	+		102	= 301

软件项目管理-成本估算

功能点估算方法举例

计算“项目技术复杂度TCF”

方法：对照【*技术复杂度因子通用特性表*】，逐一分析对本例的影响情况，确定影响系数F_i

假设：本例Fi均为3，如右表

则：TCF计算过程及结果如下：

技术复杂度因子对本例的影响系数

通用特性		对本项目影响程度系数
F1	数据通信	3
F2	分布数据处理	3
F3	性能	3
F4	硬件负荷	3
F5	事务频度	3
F6	在线数据输入	3
	终端用户效率	3
F8	在线更新	3
F9	处理复杂度	3
F10	可复用性	3
F11	易安装性	3
F12	易操作性	3
F13	跨平台性	3
F14	可扩展性	3

$$TCF = 0.65 + 0.01 \times \sum_{i=1 \text{ to } 14} Fi = 0.65 + 0.01 \times (3 \times 14) = 1.07$$

计算“项目功能点FP”

$FP = UFC \times TCF = 301 \times 1.07 \approx 322$个功能点

假设：项目开发生产率PE=15工时/功能点，按国家法律规定，每天工作8小时，每周工作5天，每月工作22天，则：

项目工作量 $Effort = FP \times PE = 322 \times 15 = 4830$工时 ≈ 121人周 ≈ 27人月

假设人工成本：1.5万元/人月，则项目人工成本 $= 1.5 \times 27 = 40.5$万元

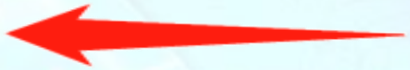
功能点与代码行的转换标准

功能点到代码行的转换系数表（Garmus & David，1996）

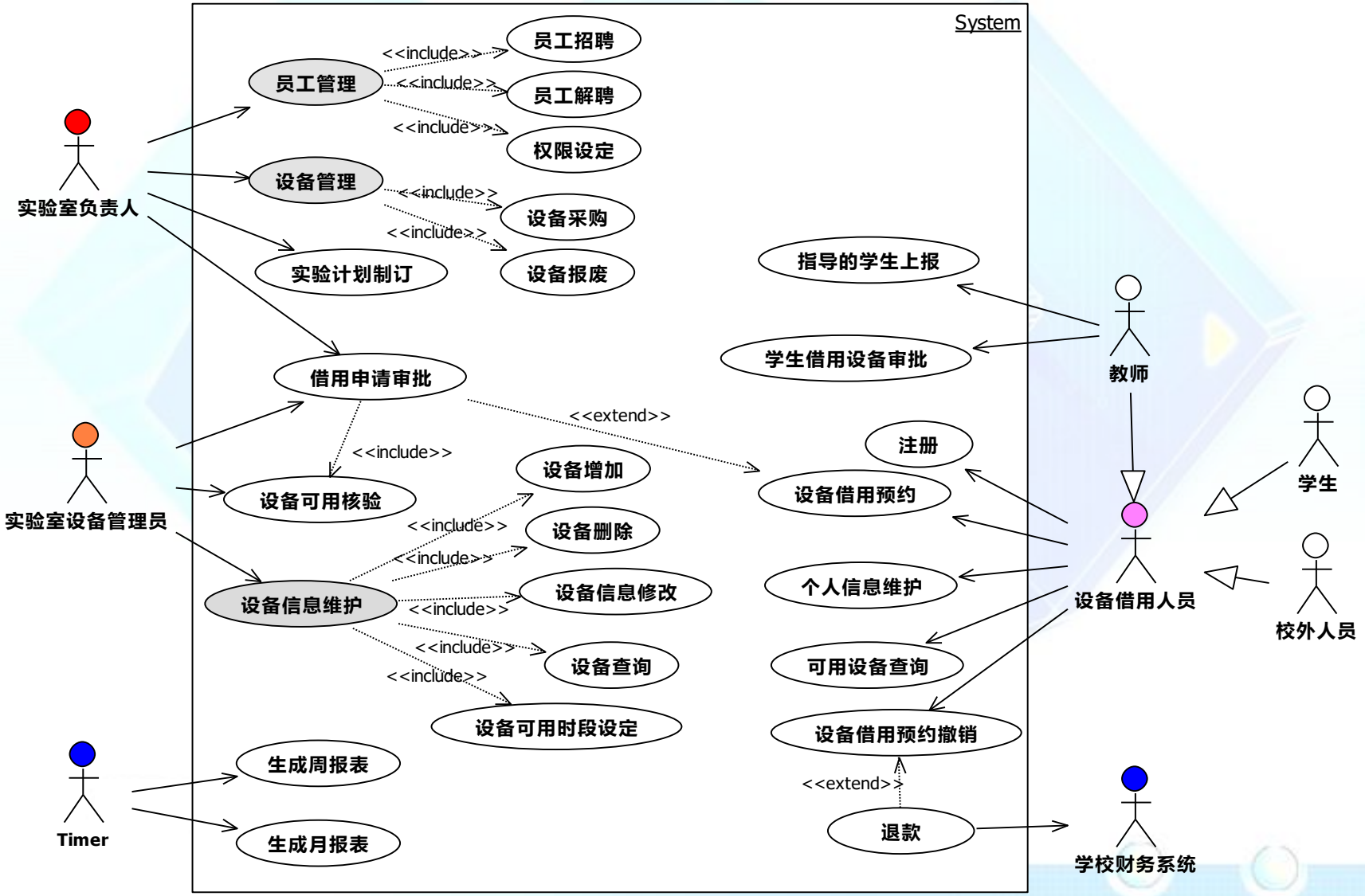
编程语言	代码行/功能点	编程语言	代码行/功能点
Assembly	320	ADA	71
C	150	PL/1	65
COBOL	105	Prolog/LISP	64
FORTRAN	105	Smalltalk	21
Pascal	91	Spreadsheet	6

传统估算方法

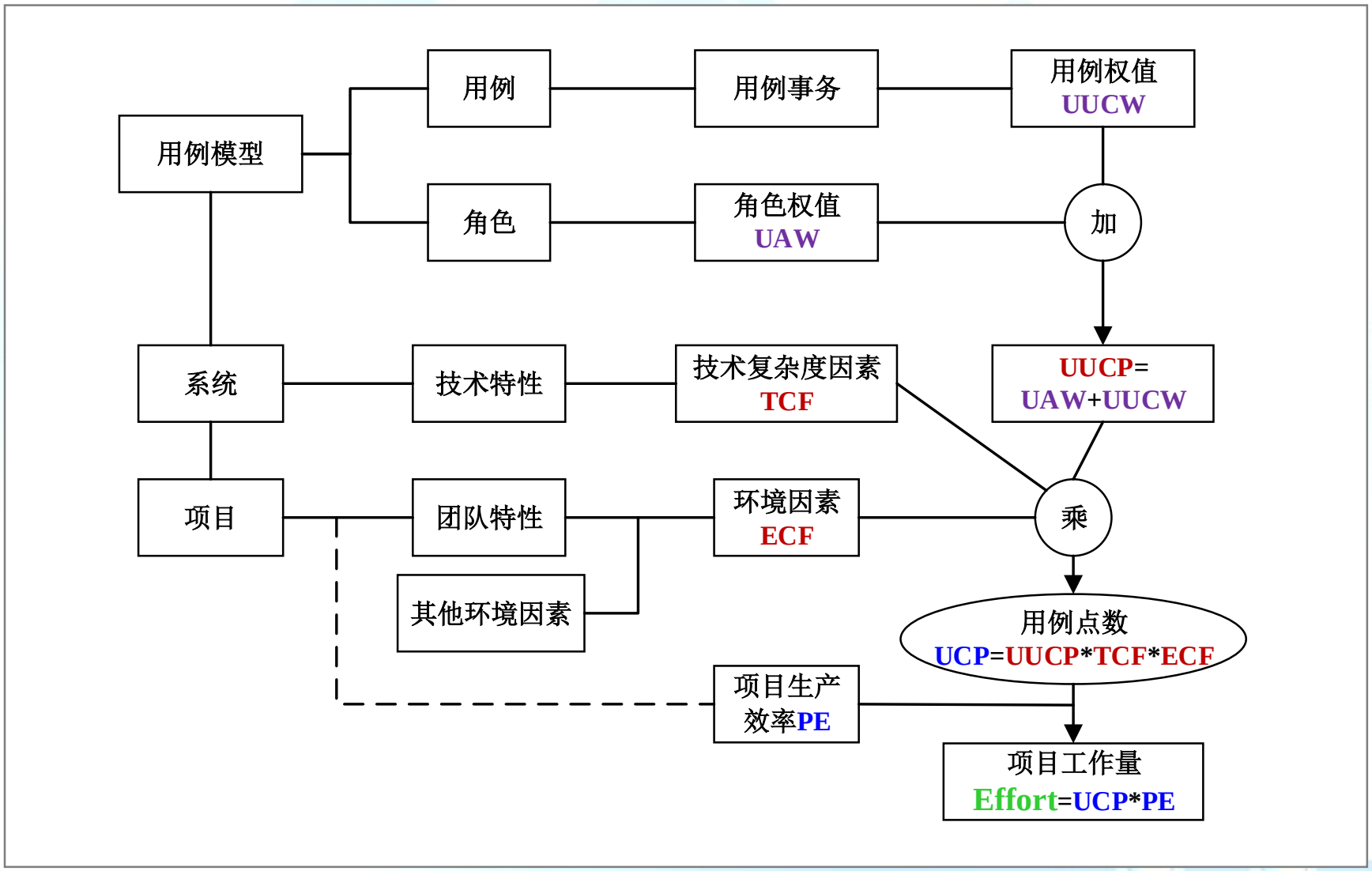
1. 代码行估算法
2. 功能点估算法
3. 用例点估算法
4. 类比估算法
5. 自下而上估算法
6. 三点估算法
7. 专家估算法



传统估算方法-用例模型


System
员工招聘
员工管理
员工解聘
权限设定
设备管理
设备采购
实验计划制订
设备报废
实验室负责人
借用申请审批
指导的学生上报
学生借用设备审批
教师
注册
设备借用预约
学生
实验室设备管理员
设备可用核验
设备增加
设备删除
设备信息维护
设备信息修改
个人信息维护
设备借用人员
校外人员
设备查询
可用设备查询
设备可用时段设定
设备借用预约撤销
生成周报表
Timer
生成月报表
退款
学校财务系统
<<include>>
<<extend>>

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. 计算技术复杂度因子TCF
5. 计算环境复杂度因子ECF
6. 计算调整的用例点UCP
7. 计算工作量Effort

1.计算未调整的角色权值UAW

$$UAW = \sum_{C=c} aWeight(c) \times aCardinality(c)$$

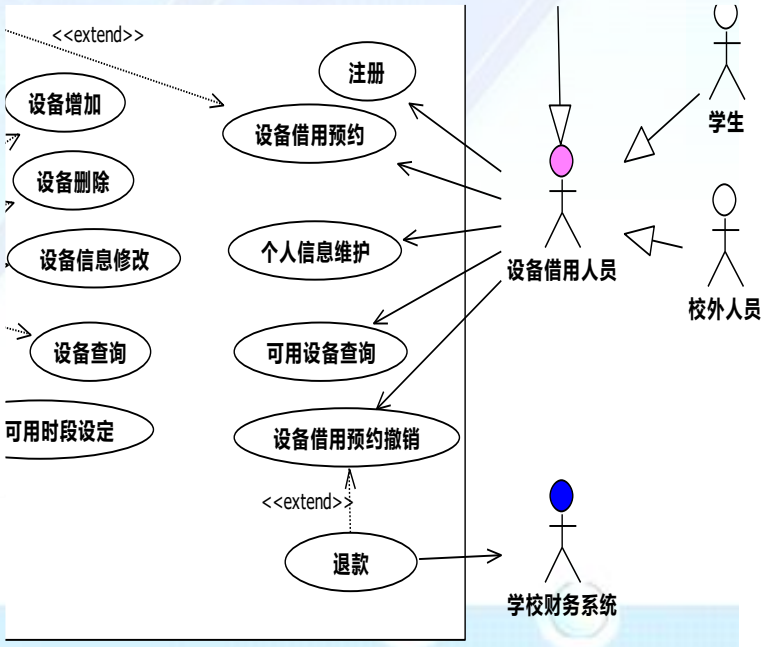
C = {simple, average, complex}

aWeight(c) – actor Weight：复杂度级别c的角色权值

aCardinality(c) – actor Cardinality：复杂度级别c的角色基数

（用例点估算法）角色权值定义

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

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

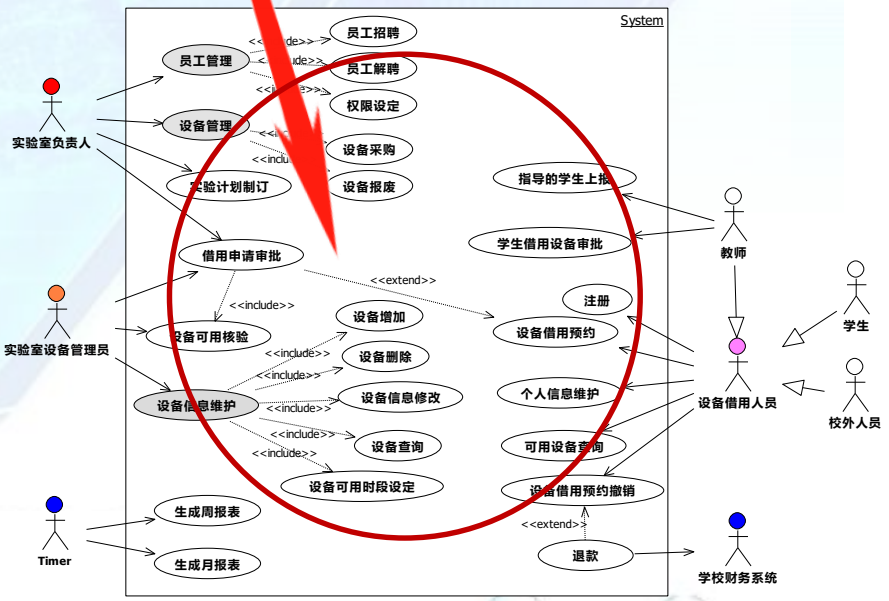
C = {simple, average, complex}

uWeight(c) – use case Weight 复杂度级别c的用例权值

uCardinality(c) – use case Cardinality 复杂度级别c的用例基数

（用例点估算法）用例权值定义

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：UUCP = 25 + 90 = 115

（用例点估算法实例）Actor 权值计算表

序号	复杂度级别	权值 $aWeight_i$	参与角色数 $aCardinality_i$	未调整角色权值 $UAW_i = aWeight_i \times aCardinality_i$
1	Simple	1	2	2
2	Average	2	4	8
3	Complex	3	5	15
$UAW = \sum UAW_i =$				25

（用例点估算法实例）用例权值计算表

序号	复杂度级别	权值 $uWeight_i$	用例数 $uCardinality_i$	未调整用例权值 $UUCW_i = uWeight_i \times uCardinality_i$
1	Simple	5	5	25
2	Average	10	2	20
3	Complex	15	3	45
$UUCW = \sum UUCW_i =$				90

4.计算技术复杂度因子TCF

考虑未来系统开发技术相关因素

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

TCF_Weight_i – 共考虑13个技术复杂度因素，以及对系统开发影响系数

$Value_i$ – 对具体的未来系统开发的影响度权值范围为[0-5]

（用例点估算法）技术复杂度因子的定义

技术复杂度因子	说明	权值
TCF1	分布式系统	2.0
TCF2	性能要求	1.0
TCF3	最终用户使用频率	1.0
TCF4	内部处理复杂度	1.0
TCF5	复用程度	1.0
TCF6	易于安装	0.5
TCF7	系统易于使用	0.5
TCF8	可移植性	2.0
TCF9	系统易于修改	1.0
TCF10	并发性	1.0
TCF11	安全功能特性	1.0
TCF12	为第三方系统提供直接系统访问	1.0
TCF13	特殊的用户培训设施	1.0

权值	技术复杂度因子的影响程度说明
0	某项技术复杂度因子与本项目无关
3	某项技术复杂度因子对本项目影响一般
5	某项技术复杂度因子对本项目有很强的影响

4.计算技术复杂度因子TCF

例如：经过分析可知，未来系统开发的技术影响度取值情况，以及通过公式计算结果如果下表所示，得到TCF = 1.02

技术复杂度因子	说明	权值 TCF_Weight_i	复杂度因子影响等级值 $Value_i$	$TCF\_Weight_i \times Value_i$
TCF1	分布式系统	2.0	3	6.0
TCF2	性能要求	1.0	5	5.0
TCF3	最终用户使用频率	1.0	3	3.0
TCF4	内部处理复杂度	1.0	5	5.0
TCF5	复用程度	1.0	0	0.0
TCF6	易于安装	0.5	3	1.5
TCF7	系统易于使用	0.5	5	2.5
TCF8	可移植性	2.0	3	6.0
TCF9	系统易于修改	1.0	5	5.0
TCF10	并发性	1.0	3	3.0
TCF11	安全功能特性	1.0	5	5.0
TCF12	为第三方系统提供直接系统访问	1.0	0	0.0
TCF13	特殊的用户培训设施	1.0	0	0.0
			$\sum TCF\_Weight_i \times Value_i =$	42.0
TCF	$=0.6+(0.01\times\sum TCF\_Weight_i \times Value_i)=0.6+0.01\times42.0 =$			1.02

5.计算环境复杂度因子ECF

ECFi的取值越大越好，ECFi越大，表明环境复杂度对项目开发影响越低

考虑项目开发团队的相关因素

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

ECF_Weight_i – 共考虑8个环境复杂度因素，以及对系统开发影响系数

$Value_i$ – 对具体的开发团队的影响度权值范围为[0-5]

环境复杂度因子定义表

环境复杂度因子	说明	权值 ECF_Weight_i
ECF1	UML 精通程度	1.5
ECF2	系统应用经验	0.5
ECF3	面向对象经验	1.0
ECF4	系统分析员能力	0.5
ECF5	团队士气	1.0
ECF6	需求稳定度	2.0
ECF7	兼职人员比例高低	1.0
ECF8	编程语言难易程度	1.0

（用例点估算法）$Value_i$ 取值表

权值	环境复杂度因子的影响程度说明
0	本项目团队成员都不具备某项环境复杂度因子描述的因素，或者说对本项目的开发负面影响很大
3	本项目团队具备某项环境复杂度因子描述的因素的程度一般，或者说对本项目的开发负面影响程度一般
5	本项目团队具备某项环境复杂度因子描述的因素的程度很高，或者说对本项目的开发负面影响程度很小

5.计算环境复杂度因子ECF

例如：经过分析可知，项目开发团队的整体情况，环境复杂度取值以及通过公式计算结果如果下表所示，得到ECF = 0.785

环境复杂度因子	说明	权值 ECF_Weight_i	复杂度因子影响等级值 $Value_i$	$ECF\_Weight_i \times Value_i$
ECF1	UML 精通程度	1.5	3	4.5
ECF2	系统应用经验	0.5	3	1.5
ECF3	面向对象经验	1.0	3	3
ECF4	系统分析员能力	0.5	5	2.5
ECF5	团队士气	1.0	3	3
ECF6	需求稳定度	2.0	3	6
ECF7	兼职人员比例高低	1.0	0	0
ECF8	编程语言难易程度	1.0	0	0
			$\sum ECF\_Weight_i \times Value_i =$	20.5
ECF	$=1.4+[-0.03\times\sum ECF\_Weight_i \times Value_i)=1.4-0.03\times20.5 =$			0.785

6.计算调整的用例点UCP

根据（1）未调整的用例点UUCP、（2）技术复杂度因子TCF、（3）环境复杂度因子ECF，可以计算出调整后的用例点UCP

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

根据前面的计算结果，得到本例的UCP：

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

$$= 115 \times 1.02 \times 0.785$$

$$\approx 92$$

7.计算工作量

假设生产率PE = 20h/用例点，则：

该项目工作量Effort = UCP × PE = 92 × 20 = 1840h

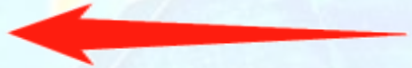
如果团队成员8人，每人每周工作35h，则：

团队每周工作量 = 35h × 8人 = 280h

所以项目工期 = 1840 ÷ 280 = 6.57周 ≈ 7周

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. 类比估算法
5. 自下而上估算法
6. 三点估算法
7. 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）

类比估算 — 适用情况

- 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 — 计算公式

$$distance(P_i, P_j) = \sqrt{\frac{\sum_{k=1}^{n} \delta(P_{ik}, P_{jk})}{n}}$$

$$\delta(P_{ik}, P_{jk}) = \begin{cases} \left(\frac{|P_{ik},\ P_{jk}|}{max_k - min_k}\right)^2 & k\text{是连续的} \\ 0 & k\text{是分散的且}P_{ik} = P_{jk} \\ 1 & k\text{是分散的且}P_{ik} \neq P_{jk} \end{cases}$$

$$distance(P_i, P_j) = \sqrt{\frac{\sum_{k=1}^{n} \delta(P_{ik}, P_{jk})}{n}}$$

$$\delta(P_{ik}, P_{jk}) = \begin{cases} \left(\frac{|P_{ik},\ P_{jk}|}{max_k - min_k}\right)^2 & k\text{是连续的} \\ 0 & k\text{是分散的且}P_{ik} = P_{jk} \\ 1 & k\text{是分散的且}P_{ik} \neq P_{jk} \end{cases}$$

类比估算 — 举例

项目 P0 与项目 P1、P2 的相似点比较

项目	项目类型	编程语言	团队规模	项目规模	工作量
P_0	MIS	C	9	180	??
P_1	MIS	C	11	200	1000
P_2	实时系统	C	10	175	900

结论：项目P_0更接近于项目P_2
因此认为：
项目P_0的工作量为900

项目间的相似度计算过程

P_0 对比 P_1		P_0 对比 P_2	
$\delta(P_{01}, P_{11})$	$= \delta(MIS, MIS) = 0$	$\delta(P_{01}, P_{21})$	$= \delta(MIS,$ 实时系统$) = 1$
$\delta(P_{02}, P_{12})$	$= \delta(C, C) = 0$	$\delta(P_{02}, P_{22})$	$= \delta(C, C) = 0$
$\delta(P_{03}, P_{13})$	$= \delta(9, 11)=[\|9-11\|/(11-9)]^2 = 1$	$\delta(P_{03}, P_{23})$	$= \delta(9, 10)=[\|9-10\|/(11-9)]^2 = 0.25$
$\delta(P_{04}, P_{14})$	$= \delta(180, 200)$ $= [\|180-200\|/(200-175)]^2 = 0.64$	$\delta(P_{04}, P_{24})$	$= \delta(180, 175)$ $= [\|180-175\|/(200-175)]^2 = 0.04$
$distance(P_0, P_1)$ $=[(0+0+1+0.64)/4]^{0.5} \approx 0.64$		$distance(P_0, P_2)$ $=[(1+0+0.25+0.04)/4]^{0.5} \approx 0.57$	

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