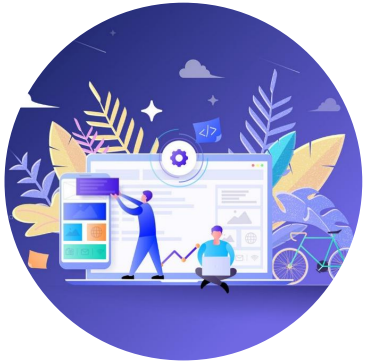
软件项目管理

第二篇 项目计划



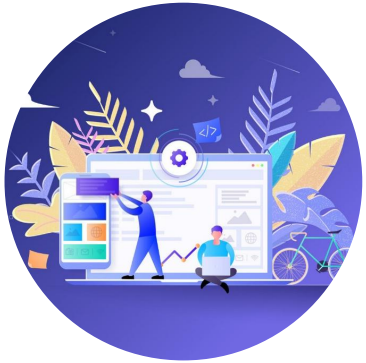
软件学院
罗昕

luoxin@sdu.edu.cn

软件学院办公楼-425

软件项目管理

第 6 章 软件项目成本计划



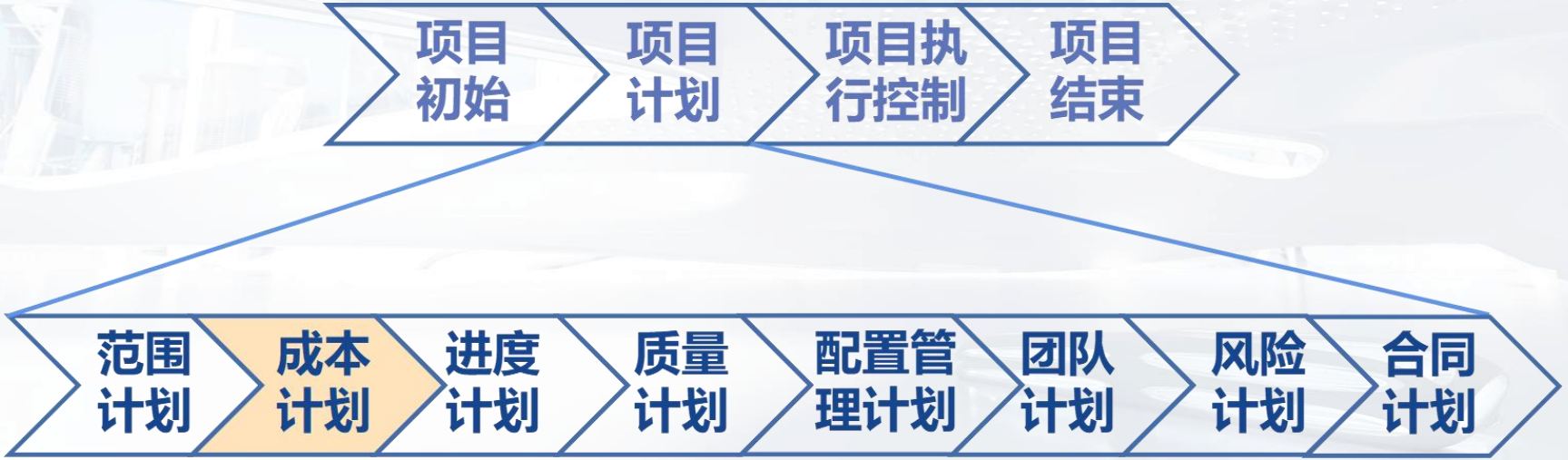
软件学院
罗昕

luoxin@sdu.edu.cn

软件学院办公楼-425

《软件项目管理》 - 路线图

MIMA



本章要点

MIMA

- 成本估算概述
- 估算过程
- 估算方法
- 敏捷项目成本估算
- 成本预算
- 案例分析
- 课程实践

本章要点

MIMA

- **成本估算概述** 
- 估算过程
- 估算方法
- 敏捷项目成本估算
- 成本预算
- 案例分析
- 课程实践

成本估算概述

MIMA

- 成本估算是对完成项目所需费用的估计和计划

- 重要性和意义
  - 软件成本估算是成本管理的核心
  - 成本估算是项目计划中的一个重要组成部分
  - 要实行成本控制，首先要进行成本估算
  - 软件成本估算，一直是软件工程和软件项目管理中最具挑战、最为重要的问题

成本

MIMA

直接成本

指与项目有直接关系的成本费用，是与项目直接对应的，包括人员的工资、材料费用、外包外购成本等，包括开发成本、管理成本、质量成本等。

间接成本

（如房租、水电、员工福利、税收等） 不能归属于一个具体的项目，是企业的运营成本，可以分摊到各个项目中。

软件项目规模与成本的关系

MIMA

- 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。

- 软件成本估算，预测开发一个软件系统所需要的总工作量的过程。

软件项目成本，指软件开发过程中所花费的工作量及相应的代价。

- 代价是待开发的软件项目需要的资金
  - 成本一般采用货币单位：人民币元、美元……

软件规模单位

MIMA

- LOC (Lines of Code) 代码行
  源代码长度的测量
- FP (Function Point) 功能点
  用系统的功能数量来测量
- 人月
- 人天
- 人年

- 例如，一个软件项目的规模是20人月，而某企业的人力成本是3万元/人月，成本是20*3=60万元

成本估算

MIMA

- 人的劳动的消耗所需要的代价是软件产品的主要成本，即开发成本。是以一次性开发过程所花费的代价来计算的。
- 开发成本的估算，应以软件项目管理，需求分析，设计，编码，测试，等这些过程所花费的代价作为依据。
- 成本估算贯穿于软件的生存期。

成本估算

MIMA

- 估算不是很准确，有误差
- 项目经验数据非常重要
- 不要太迷信某些数学模型

成本估算概述 -- 软件项目成本

MIMA

- 影响IT项目成本的因素
  - **(1) 耗用资源的数量和价格**
  - **(2) 项目工期**
  - **(3) 项目质量**
  - **(4) 项目管理水平**
  - (5) 其他

成本估算概述 -- 软件项目成本

MIMA

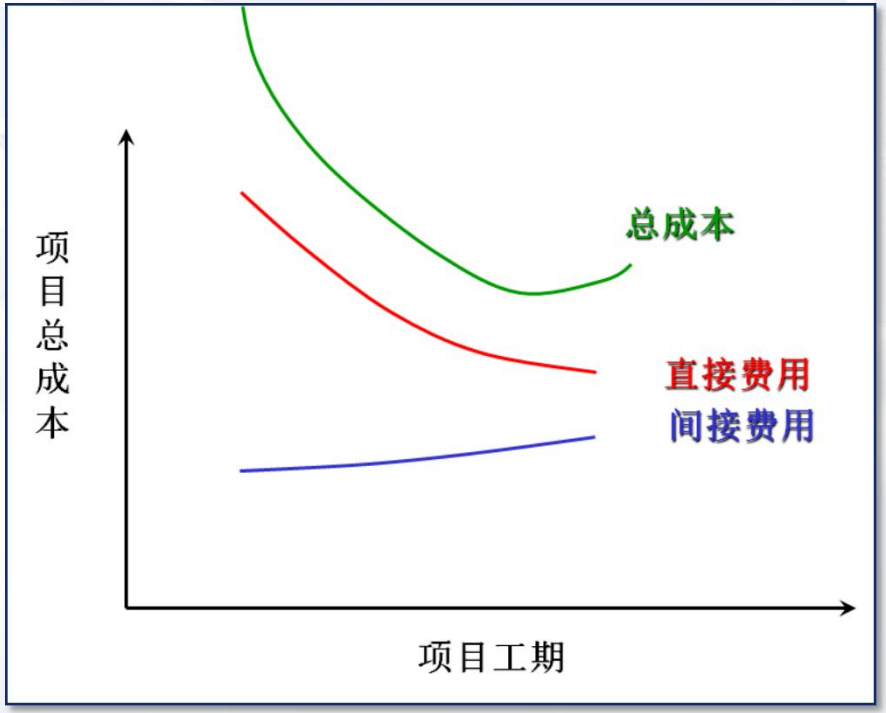
- 影响IT项目成本的因素 - **耗用资源的数量和价格**

- 中间产品和服务、市场人力资源、硬件、软件的价格也对成本产生直接的影响。价格对项目预算的估计影响很大。

成本估算概述 -- 软件项目成本

MIMA

- 影响IT项目成本的因素 - **项目工期**



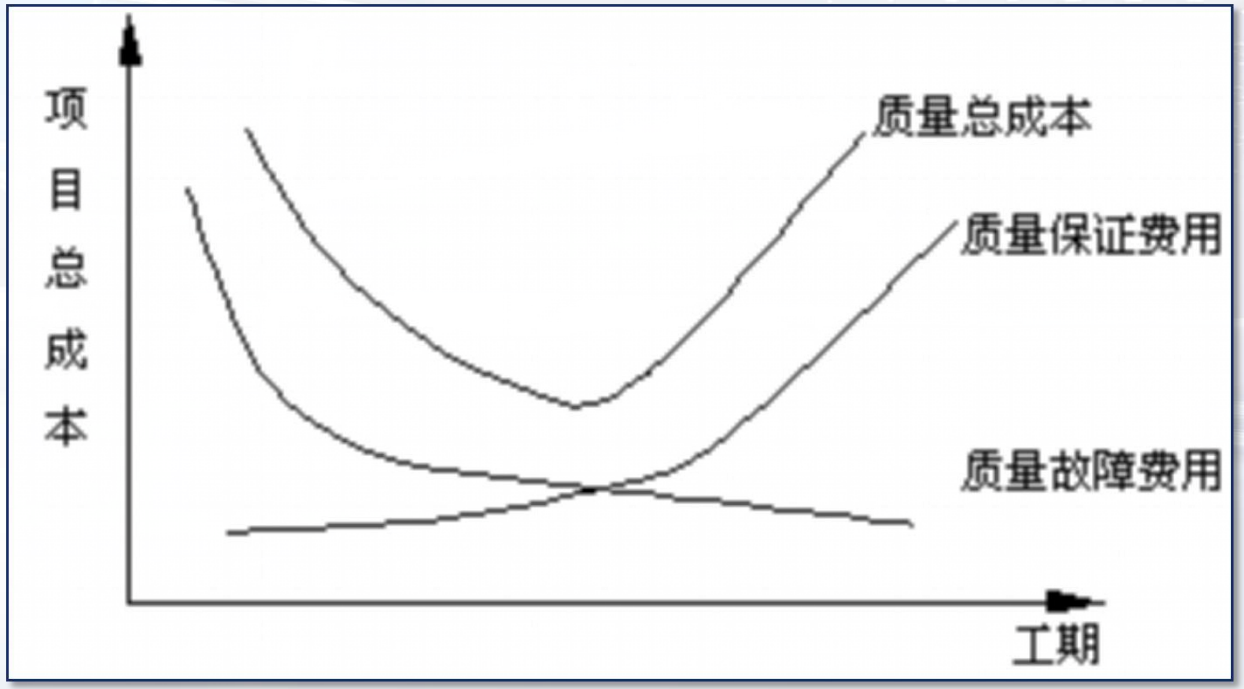
项目的费用由直接费用和间接费用组成，一般工期越长，项目的直接费用越低，间接费用越高；工期越短，直接费用越高，间接费用越低。

缩短工期需要更多的、技术水平更高的人员，直接成本费用就会增加。

成本估算概述 -- 软件项目成本

MIMA

- 影响IT项目成本的因素 - **项目质量**



质量总成本由质量故障成本和质量保证成本组成。

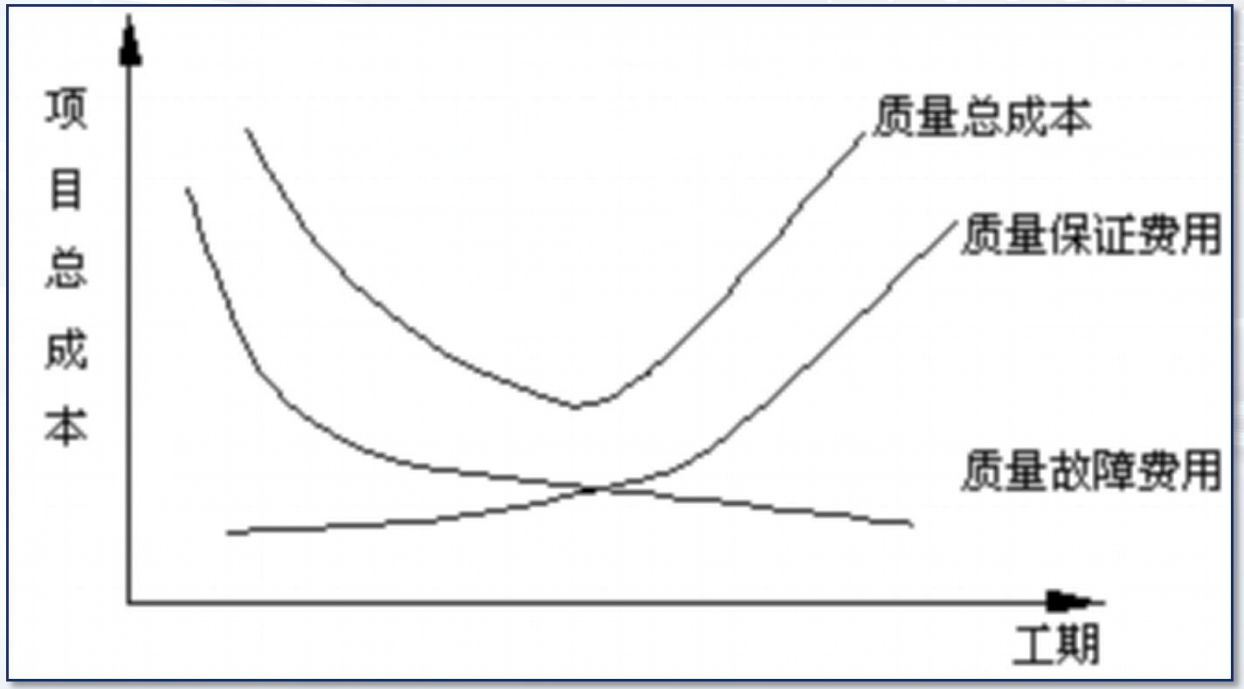
质量故障成本指为了排除产品质量原因所产生的故障，保证产品重新恢复功能的费用。

质量保证成本指为了保证和提高产品质量而采取的技术措施所消耗的费用。

成本估算概述 -- 软件项目成本

MIMA

- 影响IT项目成本的因素 - **项目质量**



项目产品的质量越低，由于质量不合格引起的损失就越大，即故障成本增加。

质量越高，相应的质量保证成本也越高，故障就越少，由故障引起的损失也相应减少。

因此需要建立一个动态平衡关系。

成本估算概述 -- 软件项目成本

MIMA

- 影响IT项目成本的因素 - **项目管理水平**

- 高的管理水平可以提高预算的准确度，加强对项目预算的执行和监管，对工期的控制严格限制在计划许可的范围之内，对设计方案和项目计划更改造成的成本增加、减少和工期的变更，可以较为有效地控制，减少风险的损失等。

- 对软件成本事先估计的有效控制，项目成本预算和估算的准确度
  - 预算过粗会使项目费用的随意性较大，准确度降低；
  - 预算过细会使项目控制的内容过多，弹性差，变化不灵活，管理成本加大。

成本估算概述 -- 软件项目成本

MIMA

- 影响IT项目成本的因素 - **人力资源对成本的影响**

- 高技术能力、高技术素质的人才，本身的人力资源成本比较高，但可以产生高的工作效率、高质量的产品、较短的工期等间接效果，从而总体上会降低成本。

- 相反一般人员，需要技术培训，对项目的理解及工作效率相对低下，工期会延长，造成成本的增加。

本章要点

MIMA

- 成本估算概述
- **估算过程**
- 估算方法
- 敏捷项目成本估算
- 成本预算
- 案例分析
- 课程实践

成本估算过程

MIMA

- 软件企业的经济性基础是利润，而利润的最直接决定因素是成本。项目结束时的最终成本应控制在预算内。
- 成本管理是确保项目在预算范围之内的管理过程，包括成本估算、成本预算、成本控制等过程。

- 估算输入
- 估算处理
- 估算输出

成本估算过程 -- 估算输入

主要依据包括：

- 项目需求
- 工作分解结构WBS
- 资源计划
- 资源单位价格
- 进度计划
- 历史信息

成本估算过程 -- 估算处理

MIMA

- 一定的估算方法进行
- 代码行估算法、功能点估算法、用例点估算法、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
- 成本

直接成本和间接成本

成本估算过程 -- 估算输出

MIMA

- 规模成本估算的结果可以以简略或详细的形式表示。
- 估算通常以货币单位表达，如元、法郎、美元等，这个估算便是成本估算的结果；也可用人月、人天 或人小时这样的单位，这个估算结果便是项目规模估算的结果。有时用复合单位。
- 成本估算是一个不断优化的过程。
- 估算说明 ①工作范围的描述 ②说明估算的基础和依据

本章要点

MIMA

- 成本估算概述
- 估算过程概念
- **估算方法**
- 敏捷项目成本估算
- 成本预算
- 案例分析
- 课程实践

估算基本方法

MIMA

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

代码行估算法

MIMA

- **面向规模的度量**

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（类比和经验方法）

- LOC(Lines of Code) 代码行
- 通常采用非注释的源代码行

代码行估算法

MIMA

- 代码行(LOC)是衡量软件项目规模最常用的概念，指所有的可执行的源代码行数，包括可交付的工作控制语言语句、数据定义、数据类型声明、等价声明、输入/输出格式声明等。

- 例如，某软件公司统计发现该公司每一万行C语言源代码形成的源文件（.c和.h文件）约为250K。某项目的源文件大小为3.75M，则可估计该项目源代码大约为15万行，该项目累计投入工作量为240人月，每人月费用为10000元（人均工资、福利、办公费用公摊等）。

则该项目中1LOC的价值为：
（240×10000）/150000＝16元/L。

代码行估算法

MIMA

- 每千行代码(KLOC)的错误数。
- 每千行代码行(KLOC)的缺陷数。
- 每千行代码行(KLOC)的成本。
- 每千行代码行(KLOC)的文档页数。
- 每人月错误数。
- 每页文档的成本。

- 代码行数依赖选择的硬件和软件，因此并不被认为是软件度量的最优方法。

代码行估算法

MIMA

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。

缺点

对代码行没有公认的可接受的标准定义。

代码行数量依赖于所用的编程语言和个人的编程风格。

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。

代码行强调编码的工作量，只是项目实现阶段的一部分。

估算的基本方法

MIMA

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

功能点估算法

MIMA

- 面向功能点（FP,Function Point）的度量
- 面向功能点法是1979年由IBM公司的 Alan Albrecht最先提出的。
- 功能点估算是对程序规模的一个综合量度，经常用于项目早期阶段。
- 从需求说明书确定功能点比确定代码行容易。
- 功能点分析中，系统分为5个组件：
  - 外部输入、外部输出、外部查询 （因为这些组件的每一个都处理文件，因此被称为事务）
  - 内部逻辑文件、外部接口文件 （构成逻辑信息的存储地）

功能点估算法

MIMA

- 功能点技术是依赖软件信息域的基本特征和对软件复杂性的估计，估算出软件规模。
- 它是一种按照统一方式测定软件功能的方法，最后的结果是一个数。这个结果数可以用来估计代码行数、成本和工期。

- 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。

功能点估算法

MIMA

- 功能点估计法要评估产品所需要的内部基本功能和外部功能。然后根据技术复杂度因子（权）对它们进行量化，产生产品规模的最终结果。

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

功能点估算法

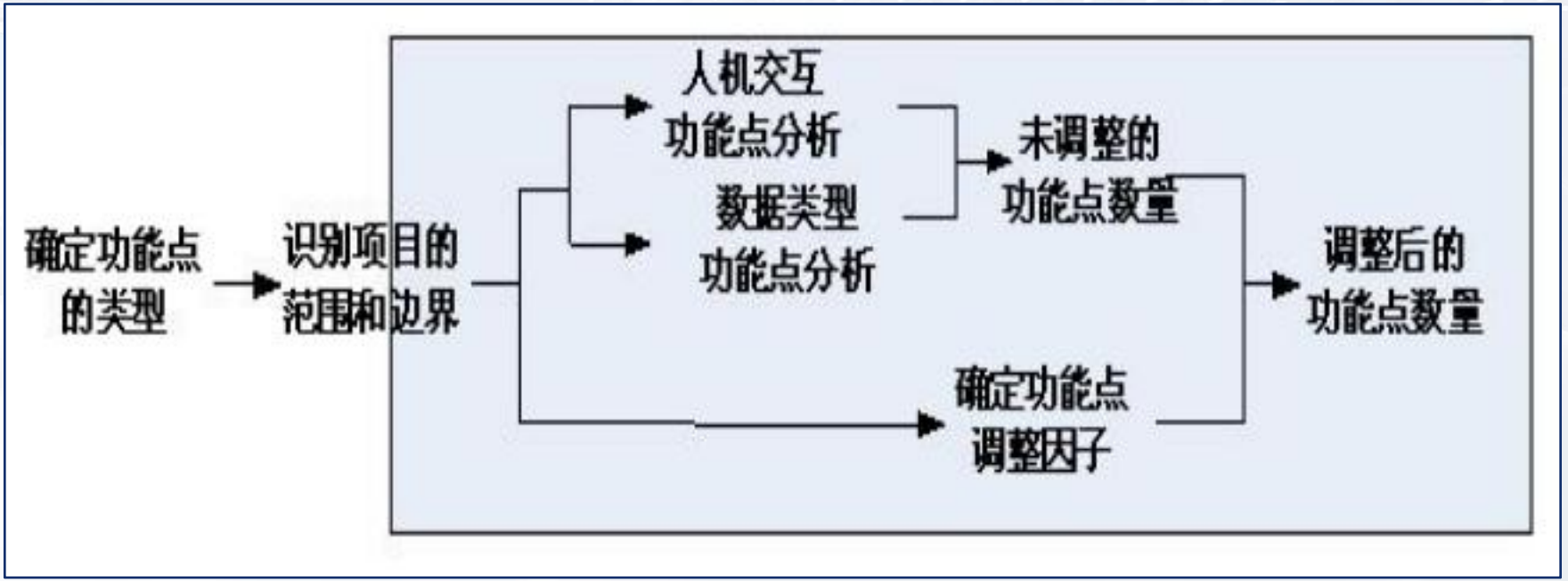
MIMA

- 国际标准IFPUG（International Function Point Users Group）组织提供的功能点估算法V4.1.1为基础进行讲解。如下图所示，为功能点估算法的使用步骤。

- **具体步骤**包括：

① 识别功能点的类型。
② 识别待估算应用程序的边界和范围。
③ 计算数据类型功能点所提供的未调整的功能点数量。
④ 计算人机交互功能所提供的未调整的功能点数量。
⑤ 确定调整因子。
⑥ 计算调整后的功能点数量。

功能点估算法


确定功能点
的类型
识别项目的
范围和边界
人机交互
功能点分析
数据类型
功能点分析
未调整的
功能点数量
确定功能点
调整因子
调整后的
功能点数量

功能点估算法 -- 功能点计算步骤

MIMA

- 1）确定应用程序必须包含的功能（例如，“回溯”、“显示”）。国际功能点用户组（International Function Point Users Group，IFPUG）已经公布了相关标准，说明哪些部分组成应用的一个功能。一个功能等价于处理显示器上的一屏显示或者一个表单。

- 2）对每一项功能，通过计算**4类系统外部行为或事务**的数目，以及**1类内部逻辑文件**的数目来估算由一组需求所表达的功能点数目。

- 这5类功能计数项分别是：

UFC-未调整功能点计数

MIMA

- **功能计数项:(从处理逻辑的角度) 组件**

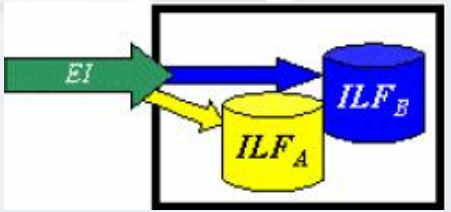
- 外部输入 External Inputs EI
- 外部输出 External Outputs EO
- 外部查询 External Inquiry EQ
- 外部接口文件 External Interface Files EIF

- 内部逻辑文件 Internal Logical Files

外部输入(External Inputs EI)

MIMA

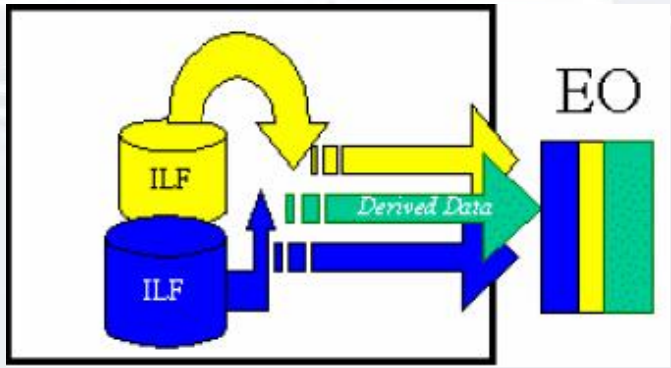
- 外部输入是最终用户或其他程序用来增加、删除或改变程序数据的屏幕（screen）、表单 （form）、对话框（dialog）或控制信号（control signal）等。
- 外部输入给软件提供面向应用的数据项（如屏幕、表单、对话框、控件、文件等）。
- 在这个过程中，数据穿越外部边界进入到系统内部。



外部输出(External Outputs EO)

MIMA

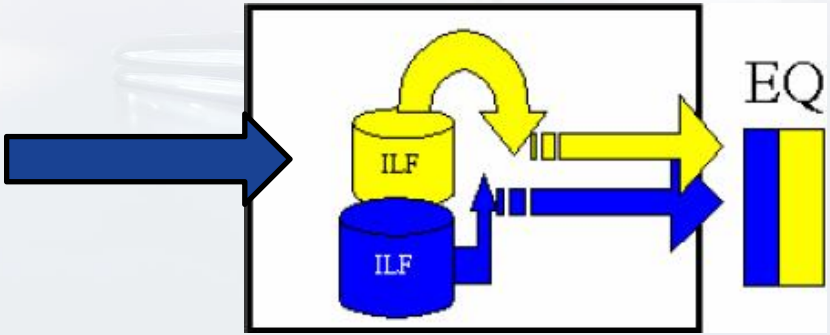
- 外部输出是程序生成供最终用户或其他程序使用的屏幕、报表（report）、图表（graph） 或控制信号等。
- 向用户提供(经过处理的)面向应用的信息，例如，报表和出错信息等。
- 派生数据由内部穿越边界传送到外部。



外部查询(External Inquiry EQ)

MIMA

- 外部查询是输入/输出组合，其中一个输入引出一个即时的简单输出。
- 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。

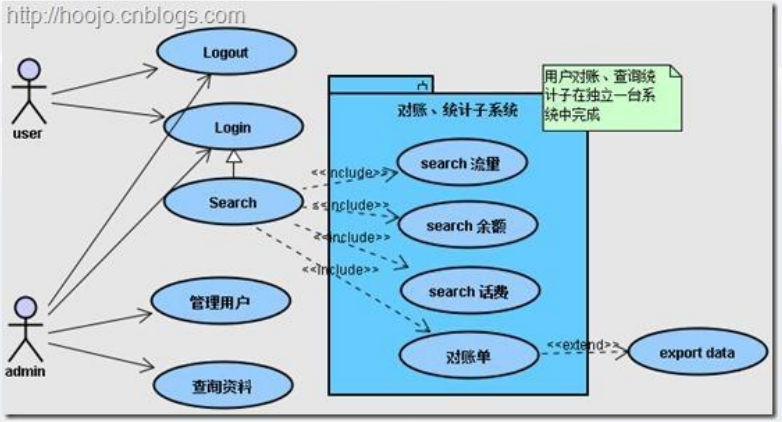


外部接口文件（External Interface Files EIF’s）

MIMA

- 外部接口文件是受其他程序控制的文件，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。

- 数据完全存在于应用的外部，并且由另一个应用维护。外部接口文件是另外一个应用的内部逻辑文件。

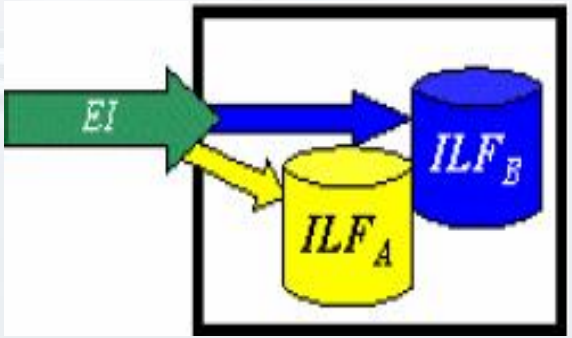


内部逻辑文件 (Internal Logical Files ILF’S)

MIMA

- 内部逻辑文件是用户可确认的、在应用程序内部维护的、逻辑上相关联的最终用户数据或控制信息，如一个平面文件，或者关系数据库中的一个表。

- 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。



功能点估算法 -- 功能点计算步骤

MIMA

- 3）在估算中对 5类功能计数项中的每一类功能计数项按其复杂性的不同分为简单（低）、一般（中）和复杂（高）3个级别。

- 功能复杂性是由某一功能的数据分组和数据元素共同决定的。计算数据元素和无重复的数据分组个数后，将数值和复杂性矩阵对照，就可以确定该功能的复杂性属于高、中、低。下表是5类功能计数项的复杂等级。

功能计数项的复杂度等级

MIMA

	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

功能点估算法 -- 功能点计算步骤

MIMA

- 3）在估算中对 5类功能计数项 中的每一类功能计数项按其复杂性的不同分为简单（低）、一般（中）和复杂（高）3个级别。

- 功能复杂性是由某一功能的数据分组和数据元素共同决定的。计算数据元素和无重复的数据分组个数后，将数值和复杂性矩阵对照，就可以确定该功能的复杂性属于高、中、低。下表是5类功能计数项的复杂等级。

- 产品中所有功能计数项加权的总和，就形成了该产品的未调整功能点计数（UFC）。

计算UFC的结果（例子）

MIMA

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	134	65	102
UFC	301		

功能点估算法 -- 功能点计算步骤

MIMA

- 4）这一步是计算项目中14个**技术复杂度因子（TCF）**。下表是14个技术复杂度因子，每个因子的取值范围是0~5。

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

功能点估算法 -- 功能点计算步骤

MIMA

调整各项权重值——Fi 的 取 值					
0	1	2	3	4	5
没有影响	偶有影响	轻微影响	平均影响	较大影响	严重影响

技术复杂度因子 $TCF=0.65+0.01\times\sum Fi$

- 根据计算所得功能点可能存在偏差，因此需要调整所计算的功能点。
- 通过对14个技术复杂度因子的情况来计算调整功能点系数。
- 权重调整取值Fi见表。

调整功能点时考虑问题的回答情况

序号	问 题（根据问题回答情况确实 *Fi* 值）	回 答	*Fi*
1	系统是否需要可靠的备份和恢复		
2	是否需要数据通信		
3	是否有分布处理功能		
4	系统是否很关键		
5	系统是否在一个已有的、很实用的操作环境中运行		
6	系统是否需要联机处理		
7	联机数据项是否需要在多屏幕或多操作之间切换以完成操作		
8	是否需要联机更新主文件		
9	输入、输出及文件查询是否很复杂		
10	内部处理是否复杂		
11	内部处理是否需要设计成可复用的		
12	设计中是否需要包装转换及安装		
13	系统的设计是否支持不同组织的多次安装		
14	应用的设计是否方便用户修改及使用		

统计∑*Fi* 值

功能点估算法 -- 功能点计算步骤

MIMA

- 5）调整所计算的功能点（FP）：

FP = 功能点总数UFC×调整系数TCF

- 其中：调整系数TCF=0.65+0.01×∑Fi

- Fi是根据对调整功能点时需考虑问题的回答结果而得出的权重调整值。取值范围0-5。
- 常数和参数的加权因子是根据经验确定的。
- 调整系数一般在0.65~1.35之间变化。（0.65+0.01*5*14=1.35）

功能点计算步骤

MIMA

- 5）最后根据功能点计算公式FP=UFC×TCF计算出调整后的功能点总和。

- 其中：UFC表示未调整功能点计数，TCF表示技术复杂因子。功能点计算公式的含义是：如果对应用程序完全没有特殊的功能要求（即综合特征总值=0），那么功能点数应该比未调整的（原有的）点数降低35%（这也就是“0.65”的含义）。

功能点估算法

MIMA

- 1）确定应用程序必须包含的功能
- 2）对每一项功能，通过计算4类系统外部行为或事务的数目，以及1类内部逻辑文件的数目来估算由一组需求所表达的功能点数目
- 3）简单（低）、一般（中）和复杂（高）3个级别。所有功能计数项加权的总和，就形成了该产品的未调整功能点计数（UFC）
- 4）计算项目中14个技术复杂度因子（TCF）。取值范围是0~5
- 5）根据功能点计算公式FP=UFC×TCF计算出调整后的功能点总和

功能点计算

信息域特征	估算值	加权因子				单项总和
		简单	中等	复杂		
外部输入		× 3	4	6	=	
外部输出		× 4	5	7	=	
外部查询		× 3	4	6	=	
内部逻辑文件		× 7	10	15	=	
外部接口		× 5	7	10	=	
				未调整功能点总计 UFP		

计算公式 $FP = 总计数 \times [0.65 + 0.01 \times \sum F_i]$

TCF计算公式的含义是：如果对应用程序完全没有特殊的功能要求（即综合特征总值**=0**），那么功能点数应该比未调整的点数降低**35%**（这也就是“**0.65**”的含义）。

5）根据功能点计算公式**FP=UFC×TCF**计算出调整后的功能点总和。

其中：**UFC**表示未调整功能点计数，**TCF**表示技术复杂因子

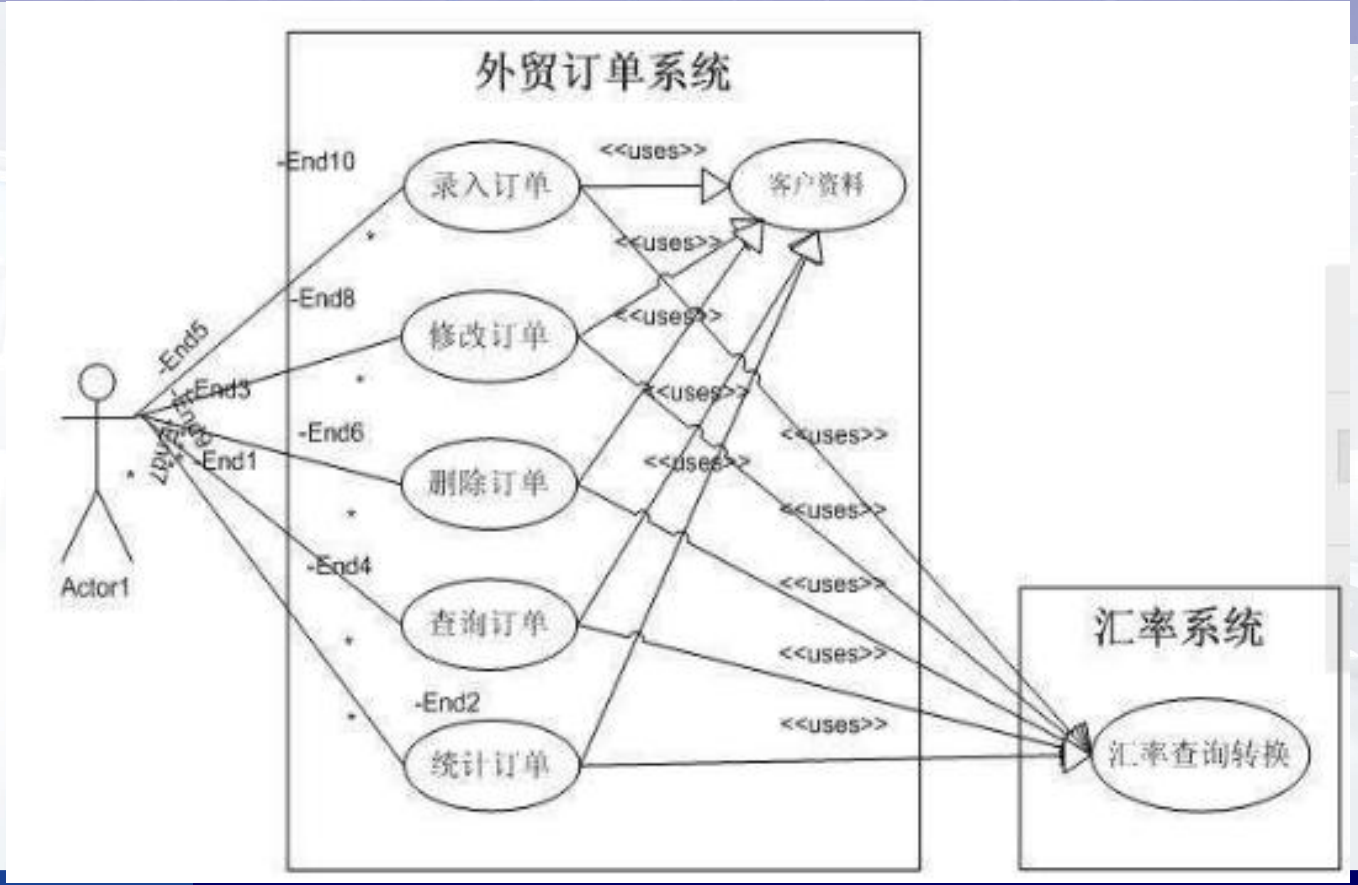
功能点到代码行的转换表

MIMA

功能点可以按照一定的条件转换为软件代码行（LOC）。右表就是一个转换表，它是针对各种语言的转换率，这个表是根据业界的经验研究得出的。

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

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>>

功能点计算实例-UFC

MIMA

根据上面的外贸订单项目的需求评估:

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	13	25	7
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

MIMA

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：功能点计算实例

MIMA

- FP=UFC*TCF
  - UFC=45
  - TCF=0.65+0.01(14*3)=1.07
- FP=45*1.07=48

- 如果：项目的生产率 PE=15工时/功能点
- 则项目的规模为 48*15=720工时

外贸订单项目：功能点计算实例

MIMA

- FP=UFC*TCF
  - UFC=45
  - TCF=0.65+0.01(14*3)=1.07
- FP=45*1.07=48

- 功能点可按照一定的条件转换为软件代码（LOC）
- LOC=AVC×功能点的数量
- AVC：指该语言在实现一个功能点时所要用的平均代码行
- 业界标准：开发软件产品的生产率不低于0.86千行等价 汇编语言代码0.86KLOC/人月

功能点计算实例

MIMA

FP=48

则该项目若用C语言编写，查表得C的功能点置换为150行，则代码量为：

LOC=48*150

若用PASCAL语言编写，查表得PASCAL的功能点置换为91行，则代码量为：

LOC=48*91

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

功能点分析总结

MIMA

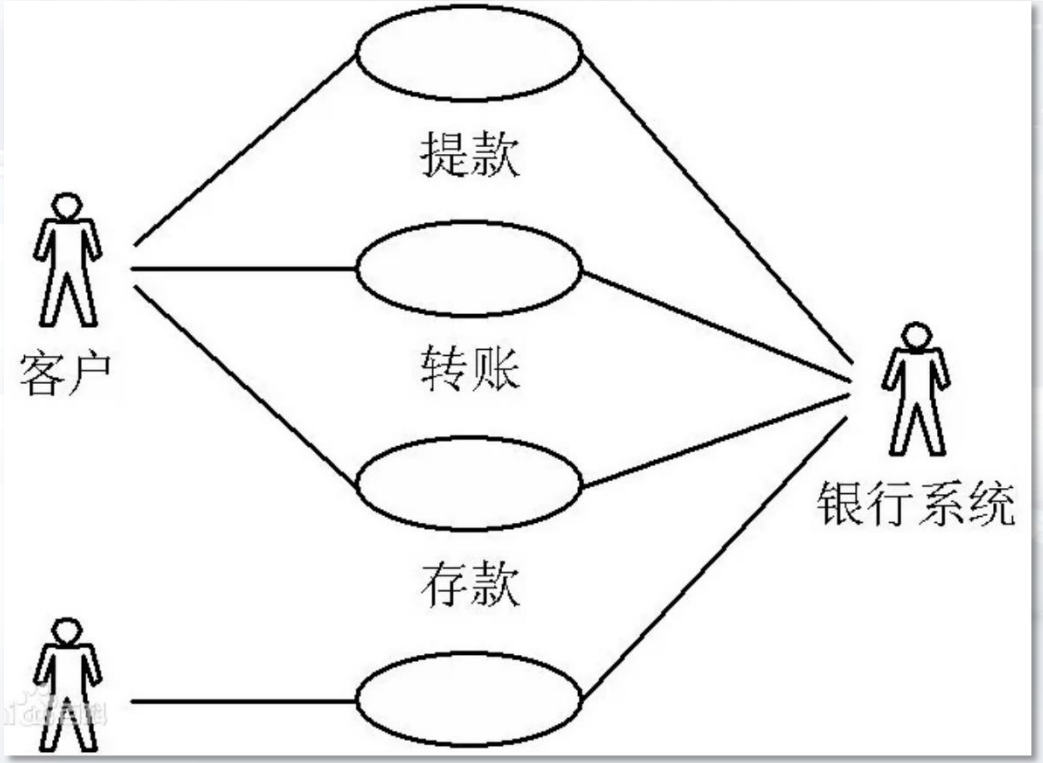
- 功能点作为度量软件规模的方法已经被越来越广泛地接受

估算的基本方法

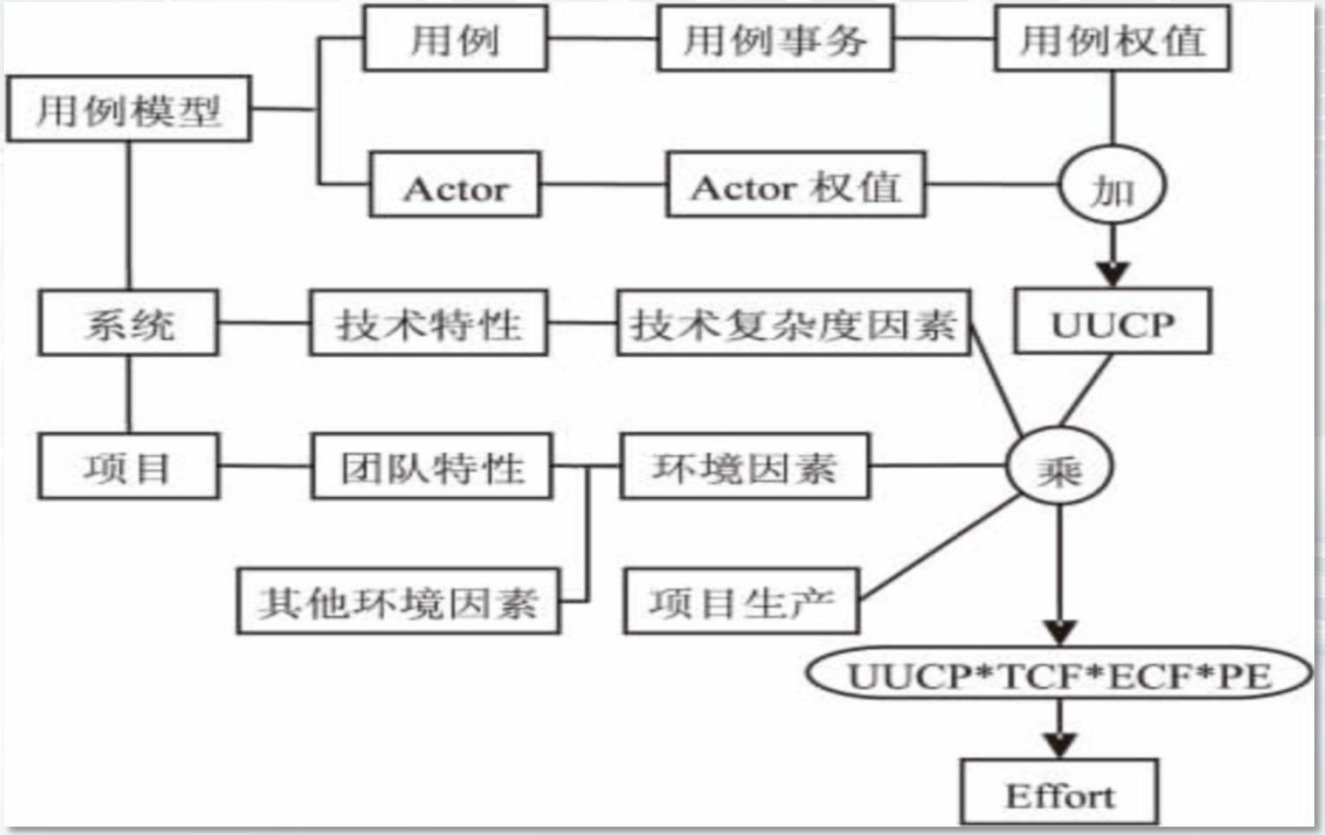
MIMA

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

用例

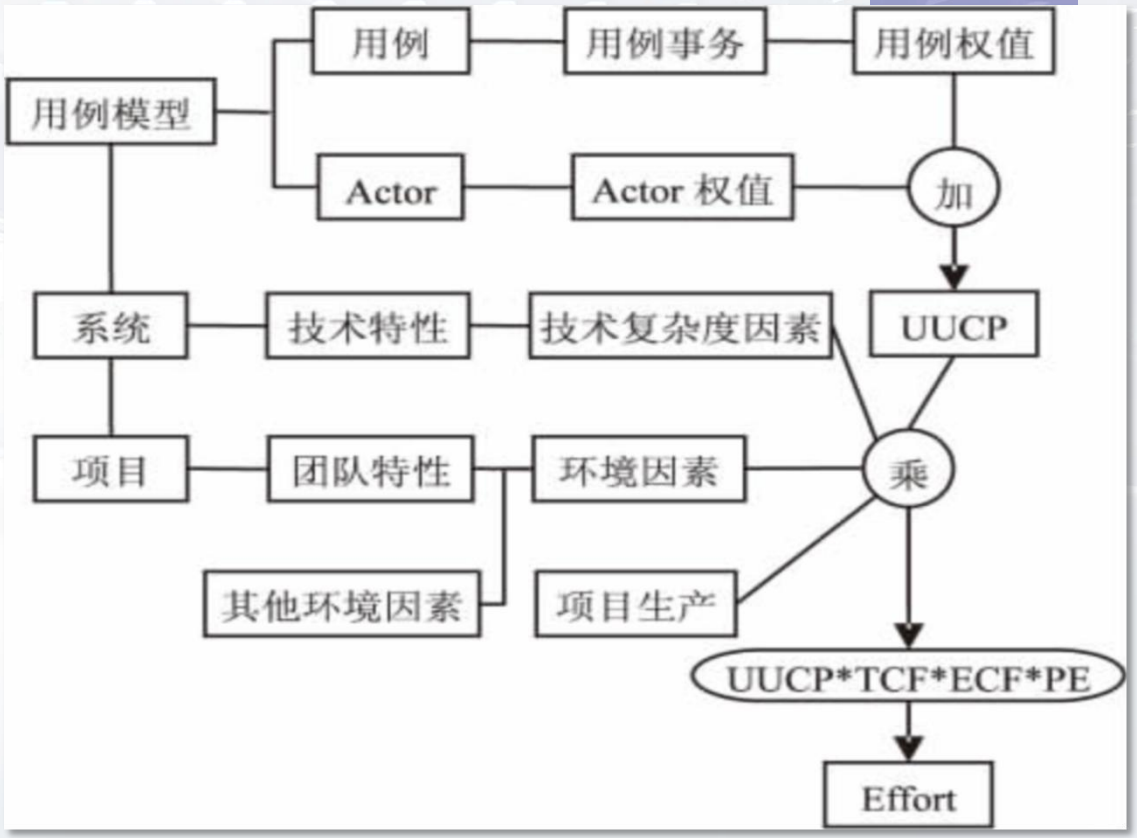


用例点估算模型


用例
用例事务
用例权值
用例模型
Actor
Actor 权值
加
系统
技术特性
技术复杂度因素
UUCP
项目
团队特性
环境因素
乘
其他环境因素
项目生产
UUCP*TCF*ECF*PE
Effort

用例点估算方法的基本步骤

- 计算未调整的角色的权值UAW;
- 计算未调整的用例的权值UUCW；
- 计算未调整的用例点UUCP;
- 计算技术和环境因子TEF;
- 计算调整的用例点UCP；
- 计算工作量(man-hours)。



1、计算未调整的角色的权值UAW

MIMA

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

- Unadjusted Actor Weight
- C={simple, average, complex}
- aCardinality(c)是参与的角色数目

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、计算未调整的用例的权值UUCW

MIMA

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

- Unadjusted User Case Weight
- C={simple, average, complex}
- uCardinality(c)是参与的用例数目

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

MIMA

UUCP =UAW+UUCW

例如

Unadjusted User Case Point

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、计算技术复杂度因子TCF - 13个

表6-14 技术复杂度因子的定义

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

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

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 - 8个

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

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

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

MIMA

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

User Case Point

- 计算未调整的角色的权值 UAW
- 计算未调整的用例的权值 UUCW
- 计算未调整的用例点 UUCP= UAW+UUCW
- 计算技术和环境因子 TEF=TCF×ECF
- 计算调整的用例点UCP

6、计算工作量

MIMA

- Effort =UCP×PF
- PF是Productivity Factor，是项目生产率，其默认值为20
- 如果：PF = 20工时/用例点
  则：Effort =UCP×PF = 88×20 = 1760h=220人天

估算的基本方法

MIMA

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

类比估算法

MIMA

- 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

类比估算法-自上而下

MIMA

- 自上而下的估算，又称类比估算，通常在项目的初期或信息不足时进行，此时只确定了初步的工作分解结构，分解层次少，估算精度较差。

- 自上而下的成本估算实际上是以项目成本总体为估算对象，在收集上层和中层管理人员的经验判断，以及可以获得的关于以往类似项目的历史数据的基础上，将成本从工作分解结构的上部向下部依次分配、传递，直至WBS的最底层。

类比估算法

MIMA

- 类推估算法的特点：
  - 类比估算法通常比其他方法简便易行，成本低。
  - 这种估算是基于实际经验和实际数据的。
  - 有两种情况可以使用这种方法：其一是以前完成的项目与新项目非常相似，其二是项目成本估算专家或小组具有必需的专业技能。
  - 这种方法的局限性在于很多时候没有真正类似项目的成本数据，因为项目的独特性和一次性使多数项目之间不具备可比性。

类比估算法

MIMA

- 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

- 类比估算中，需要评价不同项目间的相似程度。两种常用求距离方式来度量差距
  - 不加权的欧氏距离
  - 加权的欧氏距离

类比估算

MIMA

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

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

类比估算 - 举例

? ?

MIMA

表 6-18　项目 P_0 与项目 P_1 和 P_2 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	1 000
P_2	实时系统	C	10	175	900

类比估算 - 举例

? ?

MIMA

表 6-18　项目 P_0 与项目 P_1 和 P_2 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	1 000
P_2	实时系统	C	10	175	900

表 6-19　项目间的相似度计算过程

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

类比估算 - 举例 - 其他解法

? ?

MIMA

表 6-18　项目 P_0 与项目 P_1 和 P_2 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	1 000
P_2	实时系统	C	10	175	900

1）可以直接取最相似的项目的工作量，例如，如果认为 P_0 与 P_1 达到相似度要求，则 P_0 工作量估算值可以取 1 000；如果认为 P_0 与 P_2 达到相似度要求，则 P_0 工作量估算值也可以取 900。

2）可以取比较相似的几个项目的工作量平均值，即可以取 P_1、P_2 两个相似项目的工作量平均值，则 P_0 工作量估算值可以取 950 作为估算值。

3）可以采用某种调整策略，例如，用项目的规模做调整参考，对应到例子中，可采用如下调整：Size (P_0)/Size (P_1) = Effort (P_0)/Effort (P_1)，得到 P_0 工作量估算值为 1 000 × 180/200 = 900。

类比估算-主观推断

MIMA

- 由于相似度计算比较麻烦，类比估算基本上采用主观推断，很少采用相似度计算方法。

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

类比估算-主观推断

MIMA

- 由于相似度计算比较麻烦，类比估算基本上采用主观推断，很少采用相似度计算方法。

- 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

类比估算法

MIMA

- 优点
  - 直观
  - 能够基于过去实际的项目经验来确定与新的类似 项目的具体差异以及可能对成本产生的影响
- 缺点
  - 一是不能适用于早期规模等数据都不确定的情况；
  - 二是应用一般集中于已有经验的狭窄领域，不能跨领域应用；
  - 三是难以适应新 的项目中约束条件、技术、人员等发生重大变化的情况。

感谢！

软 件 学 院　罗 昕
luoxin@sdu.edu.cn