**《软件项目管理》补充作业**

**王超颖 2022141461190**

1. **简述**

## **1.1软件工作量**

### **1.1.1 简述**

软件开发工作量评估是指对软件开发项目进行工作量的量化估计，用于确定开发项目所需的资源和时间投入。估计是在软件项目的各个不同阶段进行的。每个阶段，估计的动机和采用的方法是不同的。

### **1.1.2 常见问题**

我们对大多数项目的估计都过低，工作量估计过低的原因是：人员要求和风险要求过低。如果职员能够完成目标，那么他们将受到鼓舞，如果他们发现目标根本不能完成，那么他们的激情将受到极大损害。因而，估计不是一种简单的预测行为，而是一种管理目标。

### **1.1.2 基础**

1.需要历史数据；

2.工作的度量标准（源代码的行数、功能点）；

3.复杂性（相同KLOC的两个程序花费的时间将会不同。因而不能简单地应用KLOC或SLOC，而要根据复杂性进行修正，但是复杂性的度量通常是主观而定的。）

1. **分类**

## **2.1由低向上估计**

### **2.1.1 简述**

该方法首先将项目分成部件任务，然后估算每个任务所需的工作量。在大型的项目中，分解任务的过程是一个叠代的过程，直到最下面的任务不可分解，产生WBS。该方法适合于项目规划的后期。如果应用在前期，那么必须对最终的系统作出一些假设，例如对软件模块的数量和大小进行假设。如果项目是全新的或者没有历史数据，建议用该方法。

## **2.2自顶向下和参数模型**

### **2.2.1 简述**

自顶向下的方法和参数化模型一般采用对比方法确定总的工作量。对比是建立在一系列参数的基础上的，通过这些参数可以计算出新系统的工作量。它的形式为：effort=（系统规模）\*（生产率）。例如系统规模可以用KLOC来计算，生产率以40天/KLOC。预测软件开发工作量的模型有两个部分，第一部分为估算软件大小，第二部分为估算工作效率。

## **2.3** 专家判断

## **2.3.1 简述**

专家判断是具有应用领域或者开发环境知识的人员对任务的评估。该方法特别是在对原有系统进行替换时有用，评估者对影响的代码的比例进行分析，从而得到工作量评估。

Delphi估算法：Delphi法是一种专家评估技术，在没有历史数据的情况下，这种方式适用于评定过去与将来，新技术与特定程序之间的差别。对于需要 预测和深度分析的领域，依赖于专家的技术指导，可以获得较为客观的估算。通过专家们的互相讨论，还可以博取众长。源于“三个臭皮匠能抵一个诸葛亮”的原则。DELPHI估算法是开始于1948年南德公司，这种方法首先让一小组相关专家无记名地给出自己对问题的估算，然后经过多次反复讨论以达成一致的估算结果。

## **2.4 类比估计**

## **2.4.1 简述**

类比方法又被称为基于案例的推理（Case-based reasoning）。评估者寻找已经完成的项目，这些项目与需要开发的新项目在许多特征上必须是类似的。我们用欧几里的距离（Euclidean Distance）公式选择与待预测的项目相近的项目：

distance=（（目标系统参数1-原系统参数1）2+（目标系统参数2-原系统参数2）2+……）的平方根。

## **2.5 功能点方法**

## **2.5.1 Albrecht功能点方法**

功能点（Allan Albreht先生于20世纪70年代末提出以来）计算方法的基本思想为首先计算软件的五个基本信息量：外部输入数(External Input.，EI)、外部输出数(External Output.，EO)、外部查询数(External Query.，EQ)、内部逻辑文件数(Internal Logical File.，ILF)、外部接口文件数(External Interface File.，EIF)的加权和CT，然后对其通过14个环境复杂性因子作如下修正。

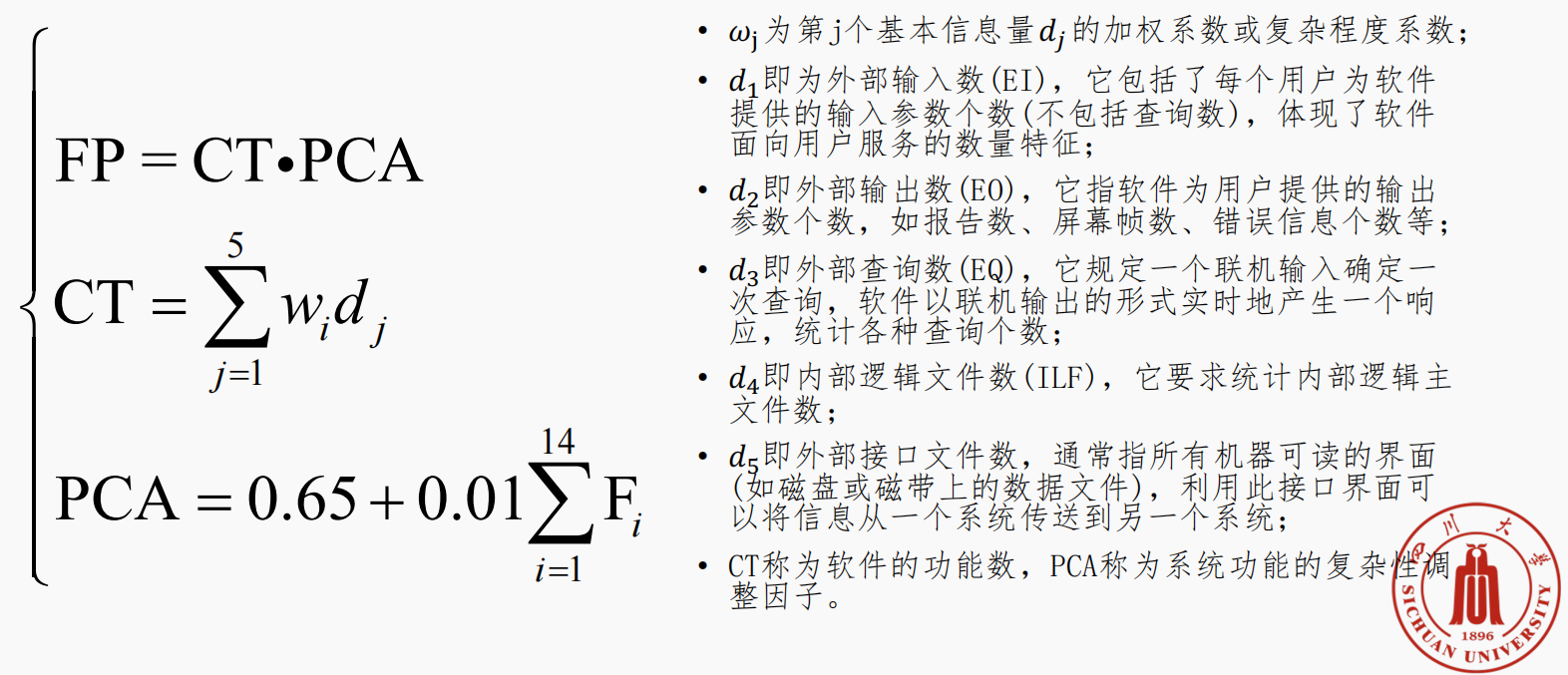


图2.1 功能点方法

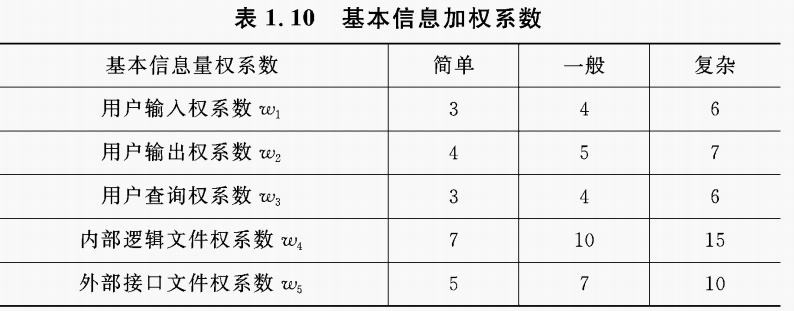


图2.2 基本信息加权系数

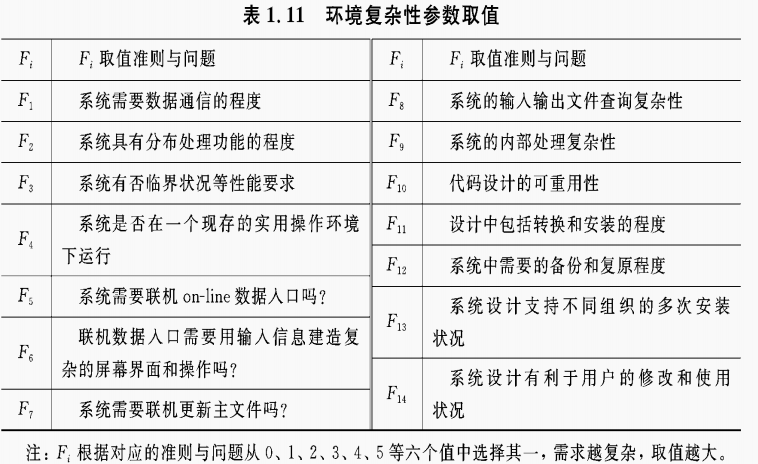


图2.3 环境复杂性参数取值

根据国际功能点用户小组（IFPUG)相关文件判定功能的复杂性，如1.4，1.5所示。

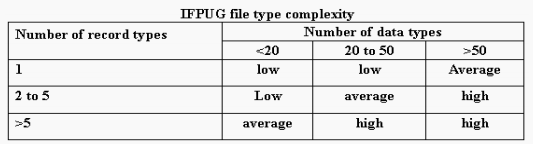


图2.4 内部逻辑文件、外部接口文件

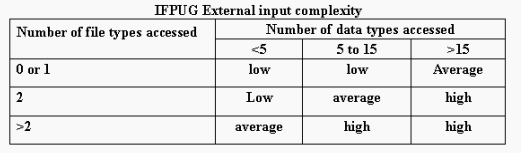


图2.5 外部输入文件

## **2.5.2 COSMIC-FFP功能点方法**

对于IFPUG方法适合于信息系统，但是不适合于实时系统或则嵌入式系统，所以产生了另外一种功能点方法COSMIC-FFP，全功能点方法。这个方法由位于加拿大、魁北克的两个相互关联的小组开发，由通用软件度量协会（Common Software Measurement Consortium,COSMIC）推动，但是其中没有美国专家。

它的关注点是数据组：COSMIC功能点方法是一种主流的软件功能规模度量方法，其原理是通过识别“数据移动”的个数来度量软件规模，每个数据移动计为一个COSMIC功能规模单位（CFP），四种数据移动类型分别为：输入（Entry）、输出（eXit）、读（Read）和写（Write）。功能点方法与技术平台无关，与实现的人员无关，系统之间的规模具有可比性，它是客观度量，不依赖于参与估算的人员。

它的功能规模是通过“数据移动(Data movement)”的个数来度量。原则是每一个功能必须有一个输入，一个输出或一个写，即至少2个CFP。

它能识别数据移动。从用户功能需求入手，分析功能过程的执行过程，分解

出每个分过程，识别输入、输出、读、写这4类数据移动，一个数据移动记为一个COSMIC功能规模单位(CFP)。

功能性用户需求的来源：

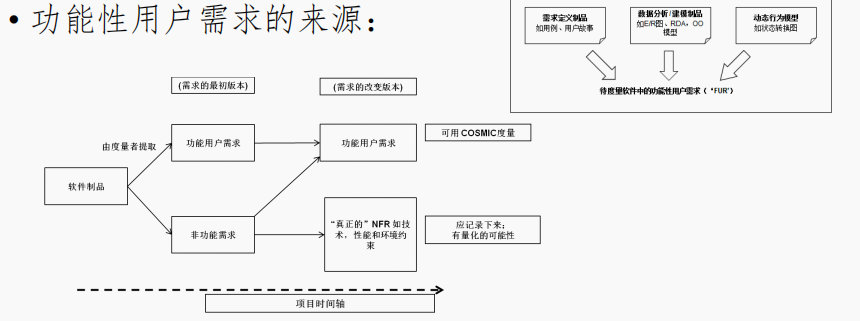


图2.6 功能性用户需求的来源

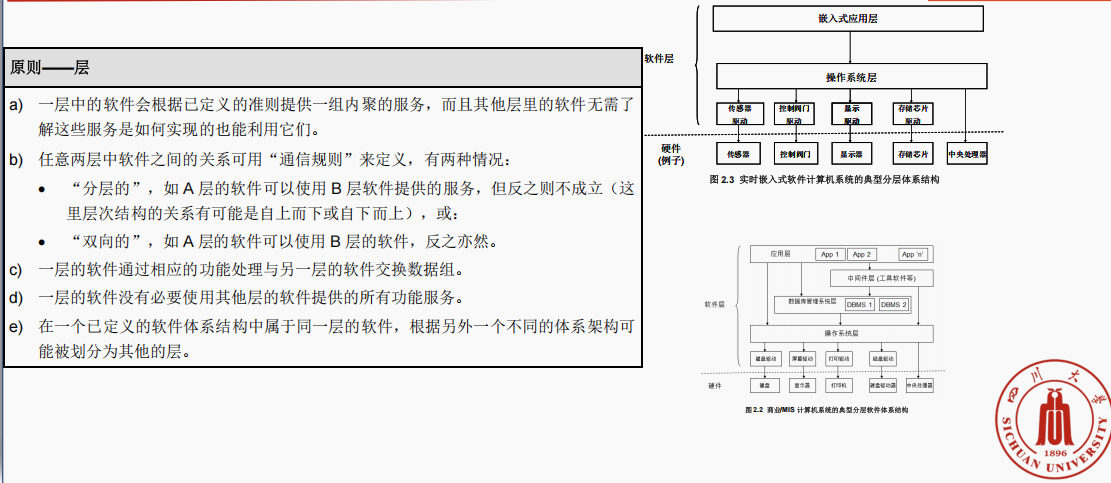


图2.7 COSMIC-FFP功能点方法

COSMIC的映射阶段是把已有的软件制品中的功能性用户需求（FUR）映射为 COSMIC通用软件模型所需格式的过程。

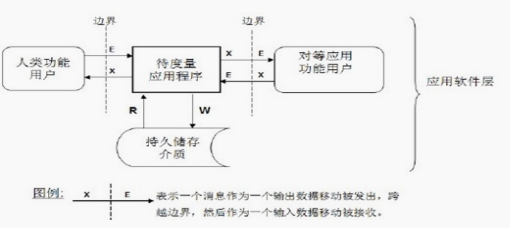


图2.8 COSMIC-FFP映射阶段

待度量软件的功能性用户需求中可识别的一个事件，此事件使得一个或多个软件功能用户产生一个或多个数据组，每个数据组随后被一个触发输入所移动。一个触发事件不可再拆分，并且要么已经发生，要么没有发生。注：时钟和定时事件可以作为触发事件。



图2.9 COSMIC-FFP的触发事件

## **2.5.3 Mark II功能点方法**

该方法被作为英国政府项目实施中采用的标准，本来是对IFPUG的改进，主要是英国在使用。基本原理：对于一个处理事务的计算方法为

wi×输入数据元素＋we×实体＋wo×输出数据元素

系数总和为2.5，标准设置为0.58，1.66，0.26。系数调整的考虑因素为：与其它应用的接口、特殊的安全特征、与第三方的直接交互、用户训练特征、文档需求。

**2.5.4 功能点转化为工作量**

（1）对于原来的项目，计算生产率：生产率＝功能点数目/工作量（人日）；

（2）对于新项目，功能点计算出来后工作量为工作量＝功能点数目/生产率；

（3）更复杂的方法：最小二乘法，即工作量＝系数1＋功能点数×系数2。

**2.5.5 扩展**

功能点方法起源于业务信息系统应用，因而强调了数据方面的因素而没有考虑功能和行为（控制）方面的因素。特征点（Feature Points):除了考虑普通功能点的内容外，还考虑了算法的特征（矩阵转换，字符串解析，处理中断等都是算法的例子）。Boeing提出了一个三维功能点方法（3D）其中三维为数据维，功能维（输入转化为输出的步骤）和控制维（状态之间的转换数）。

## **2.6 对象点**

## **2.6.1 简述**

Object Points起源于纽约大学的Leonard N.Stern商学院，它类似于功能点方法，但是更容易计算。对象点方法与面向对象方法并无直接联系。该方法计算应用所需要处理的屏幕，报告和部件，这些都被称为对象。每一对象需要被确定为简单的，中等的，困难的三个层次。

首先考虑已经存在的对象应该排除在工作量计算内，即计算新的对象点（NOP）。根据原来从事过的项目计算在不同情况下的项目的生产率，例如下表：

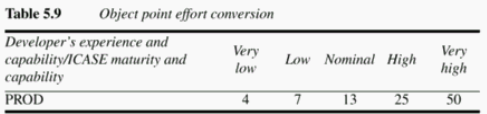


图2.10 对象点生产率

假定有672个对象点要开发，开发者的经验和工具使用都是一般性的，则需要672/13＝52个月。

## **2.6.2 扩展-NESMA**

NESMA（ Netherland Software Measurement Association，荷兰软件度量协会）功能点估算方法，它基于IFPUG发展而来。2003年，NESMA功能点估算法升级为ISO/IEC国际标准，目前版本是ISO/IEC 24570:2018。2016年，我国电子行业标准《软件工程 功能规模测量NESMA方法》正式发布。

（1）指示法：一般用于计划阶段，因为此阶段需求文件多不完善，故而只需关注逻辑文件即可。（2）估算法：一般用于执行阶段，此时需求文件较为完善，故需要关注逻辑文件和相应的操作；（3） 详细法：主要用于事后评估阶段，此时功能需求非常详细，可关注逻辑文件、相应操作和复杂度。

在IFPUG的基础上，NESMA增加了指示法（Indicative）和估算法（Estimated），使得用户可以在需求不完善的情况下，快速估算软件规模。因翻译的差异，上述三种方法可能在名称上有所不同，但表达的意思是一致的。

## **2.6.3 扩展-FiSMA**

FiSMA1.1由芬兰软件测量协会(Finnish Software MetricsAsociation, FiSMA)的工作组开发有些FSM方法是面向过程的, 而FiSMA1.1是面向服务的。它的基本功能部件类为basefunctional componentclas;BFC。

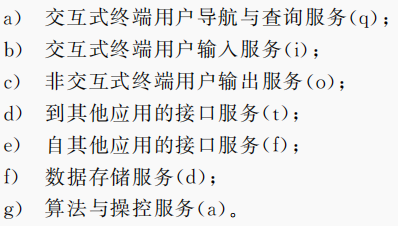


图2.11 FiSMA1.1方法

## **2.7 参数化模型**

## **2.6.1 COCOMO**

COCOMO（Constructive Cost Model）是Boehm在二十世纪70年代采用他的模型对63个项目进行了研究，由于其中只有7个是商务系统，因而它们不仅仅能被用于信息系统。它的基本的公式为：Effort=c∗sizek,其中effort采用“人月(152个工作小时）”pm来度量，size采用kdsi即千行交付源代码指令(thousands of delivered source code instructions)。

C,k的取值根据系统的分类而定：根据系统的技术特性和开发环境可以分为：

（1）有机模式（organic mode): 相对小的团队在一个高度熟悉的内部环境中开发规模较小，接口需求较灵活的系统。

（2）嵌入式模式（Embedded Mode)开发的产品在高度约束的条件下进行，对系统改变的成本很高。

（3）半分离模式（Semi-detached Mode)两者之间

其中，信息系统是有机模式，而实时系统是嵌入式模式。

图2.12为参数化模型的系数表，其中，K值反映了项目越大，则工作量成指数增加，因为大项目需要更多的协调和安排。

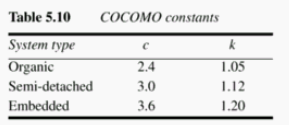


图2.12 系数表

事实上，基本COCOMO模型对工作量的衡量不稳定，Boehm本人也发现了此问题，因而提出名义成本估算的概念。首先从基本模型得到名义成本，然后采用开发成本乘法算子（development effort multiplier,dem)进行修正，即：Pm=Pmnom×dem。dem的计算如图2.13所示。

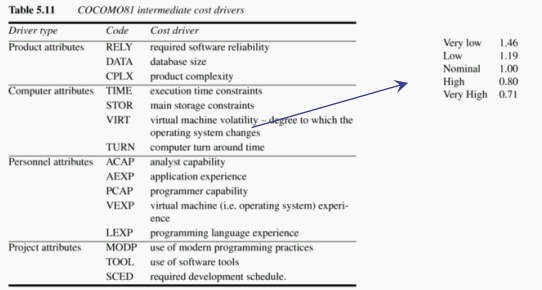


图2.13 dem的计算方法

## **2.6.1 COCOMO II**

针对COCOMO的不足，Boehm开始和其合作者发展了新的模型。该方法利用多种乘法算子和指数。一个明显的特点是该模型适应了项目在执行过程中变得越来越确定的状态，因而是一种渐进性评估。COCOMOII被分为三个阶段性模型：应用构成阶段（Application Composition)：（1）系统的外部特征被设计。在该阶段经常可以采用原型。（2）早期设计（Early Design): 基本的软件结构被设计。（3）构造阶段（Post architecture): 构造满足要求的系统。

在应用构成阶段，采用对象点计算的方法。在早期设计阶段，采用功能点计算的方法。功能点可以转换为SLOC。

Pm=A×sizesf×em1×em2×……emn。

其中，Pm为“人月”工作量，A是一个常数，size以SLOC为单位，sf是规模指数。（Sf＝1.01+0.01×因素指数的和）

计算规模因素的质量：（1）先前经验（Precedentedness): 是否有先前的经验；（2）开发的灵活性（Development Flexibility): 是否需求能够以多种方式来满足；（3）体系结构/风险解决(Architecture/Risk Resolution)：是否方案已经被确定和解决的程度；（4）团队的凝聚性（Team cohesion)；（5）过程的成熟性（Process Maturity)。