丛书第4分册

MW

MICROCOMPUTER MARKET IN THE WORLD

世界微型计算机市场

微型计算机软件

电子计算机工业信息交流中心

世界微型计算机市场

微型计算机软件分册

电子计算机工业信息交流中心

编 辑 说 明

一、当今，微型计算机技术的发展极为迅速，市场销售的产品种类繁多。如何选型，不仅是用户和经销单位普遍关注的问题，也是从事计算机科研开发、研制生产和应用的各级管理部门、各级决策者面临的重要问题。《世界微型计算机市场》丛书正是作为一套完整的参考资料，使读者对国际微型机市场行情和产品技术特点有一个比较全面的了解。

二、本丛书是有重要价值的信息咨询资料，而不是一般的出版物。本册资料来源主要是根据美国DATAPRO RESERCH公司一九八五年十月发表的有关报告，并参考其它有关资料汇编而成的。DATAPRO公司创建于一九六八年，自一九七〇年开始将其高技术研究和咨询报告提供给订户，包括电子数据处理、办公室系统等新产品编辑成有意义而又可管理的信息，是迄今获取有关数据处理、电子计算机产品、通信产品和网络方面最广泛、富有权威性公司。

三、本丛书共分四册，即微型计算机系统分册，办公室自动化、局部网络和CAD系统分册，微型机外部设备分册，以及微型机软件分册。

四、本分册主要有微型计算机系统软件、实用程序，以及应用软件等产品概况，并撰写有微型计算机软件费用估价、集成软件综述，书的最后一部分是通信软件、应用软件部分，包括：财会、建筑/土木工程、电气/电子工程、化学/燃料工程、事务图形、电子数据表、数据库、政府用行政管理、数学和统计等。内容充实，数据可靠，取材广泛，注重实用。

五、本分册的主要翻译、编审、编辑人员有刘侃、张百顺、张鹏飞、钱承德、金绮、龚继宴、陆忠恒等同志。本书曾得到电子部雷达局情报室、华北计算技术研究所情报室、中国科学院计算技术研究所情报室等单位的大力支持，在此一并致谢。本书的编审工作由于时间仓促，疏漏之处难免，请读者指正。

电子计算机工业信息交流中心
中国计算机用户协会
中国计算机动态信息网
一九八六年一月

目　　录

软件费用估价

在软件工程中，一个软件产品费用的估价是件非常棘手而又易错的工作。在软件开发的规划阶段，由于大量的未知因素存在，因此很难作出准确的费用估算。而按照签订合同的惯例，在可行性研究中经常要求有一个严格的费用保证。随着商业上的竞争，对软件费用的估价已成为影响软件项目的各种费用和进度的主要因素。

由于这个问题，某些机构使用了一系列费用估算方法。在规划阶段需要有初步的费用估算并写在课题可行性的论证报告中。在软件需求的论证报告中要有进一步完善的估算。在初步设计论证报告中要有最终的费用估算。每一次估算都是对前一次估算的进一步完善。随着工作的深入，利用不断完善的信息使得估算更加准确。有时也将一些可选产品及相应的价格写在论证报告中，从而使顾客在可能的范围内选择费用效益较好的方式。

有时顾客将分析阶段和初步设计阶段分开签订合同，以便获得精确的费用和计划估算。有时顾客将分析和初步设计的合同交给多个软件开发机构，这样顾客可利用分析和初步设计的相互竞争的有利条件，来选择某个机构开发所需的软件产品

表1列出了影响软件费用的主要因素，下面对这些因素加以详细的讨论。

表1　　影响软件费用的主要因素

程序员能力
产品复杂程度
产品规模
可用时间
要求的可靠性
技术等级

软件费用因素

影响一个软件产品的费用有多种因素。这些因素对开发和维护工作的影响较大，因而开发和维护工作的费用也较难估算。主要的费用因素是参加这项工程的每个人员的能力及他们熟悉的应用领域；产品的复杂程度；产品的规模；可用时间；所需求的可靠性；使用的技术等级；以及开发这个产品使用的系统的可用性、稳定性和熟悉程度。

程序员能力

1968年，Harold Sackman和他的同事们作了一个著名的试验，目的是测定一下批作业方式和分时方式对程序员生产率的相对影响。12个有经验的程序员每人要解决2个编程问题，一些人用批处理手段，而另外一些人用分时手段。结果表明，程序员本身的技术素质对结果的影响，要大于使用成批或分时机器访问所产生的影响。性能最好和最差之间的差别因素为：程序规模6比1，执行时间8比1，开发时间9比1，编码时间18比1，调试时间28比1。这些结果归纳在表2中。对同一个程序，编得最好和编得最差的两个程序员，他们都具有11年的经验。

表 2　　程序员生产率的变化（Sackman，12个程序员）

性能测试	比率
调试时间(小时)# 1	28：1
调试时间(小时)# 2	26：1
DVMT的CPU执行时间(秒)# 1	8：1
DVMT的CPU执行时间(秒)# 2	11：1
编码时间<小时># 1	16：1
编码时间<小时># 2	25：1
程序规模# 1	6：1
程序规模# 2	5：1
执行时间# 1	5：1
执行时间# 2	13：1

在后来的试验中，Sackman观察到生产率的变化为16比1。

通过观察可以看到，程序员能力之间的差异是由于在两个方向上的最终性能数目很少所致。取消这些情况后得到的程序员生产能力的实际变化值，是费用估算的一个重要因素。在大工程项目中，单个程序员能力的差异将趋于平均值，而在仅有五个以下程序员完成的工程中，单个程序员的能力不同就影响较大。

产品复杂程度

一般公认有三类软件产品：应用程序，它包括数据处理和科学运算程序；实用程序，如编译程序、连接编辑程序和编目系统；系统程序，如数据库管理系统、操作系统和实时系统。应用程序在由Fortran和Pascal等语言编译程序所提供的环境中加以开发。与操作系统的交互作用限制在作业控制语句，而运行时支持的功能由语言处理程序提供。实用程序为用户提供处理环境并建立对操作系统较为完善的应用。系统程序直接与硬件相互作用，主要管理并行处理和时间控制。

Brooks指出，编写实用程序要比写应用程序难三倍，而写系统程序又比写实用程序难三倍。因此，他将应用程序—实用程序—系统程序的产品复杂程度定为1—3—9。

Boehm用产品复杂程度的三个级来表示产品并给出了方程式来预测工作项目的全部程序员一月(PM)，用在产品中已交货的千条源指令(KDSI)来表示。软件课题中程序员的费用为付出全部程序员人月与每个程序员人月费用的乘积。方程可以由大量实际工程的历史数据推导出来。由于工程足够大，因而程序员生产率间的个体差异趋于一个期望值。Boehm用有机程序，半分离程序和嵌入式程序专门术语来表示产品复杂程度的三级。这三级大致对应着应用程序、实用程序和系统程序：

应用程序：$PM=2.4*(KDSI)^{1.05}$

实用程序：$PM=3.0*(KDSI)^{1.12}$

系统程序：$PM=3.6*(KDSI)^{1.20}$

这些方程的图形如图1所示。从图1可以看出，开发60,000行应用程序、实用程序和系统程序的程序员人月比率大致为1:1.7:2.8。因此，这些方程指出，对于同样的60K行程序，实用程序的工作量约是应用程序的2倍，而系统程序的工作约是应用程序的3倍。从图1中还可看出，程序越大，比率就

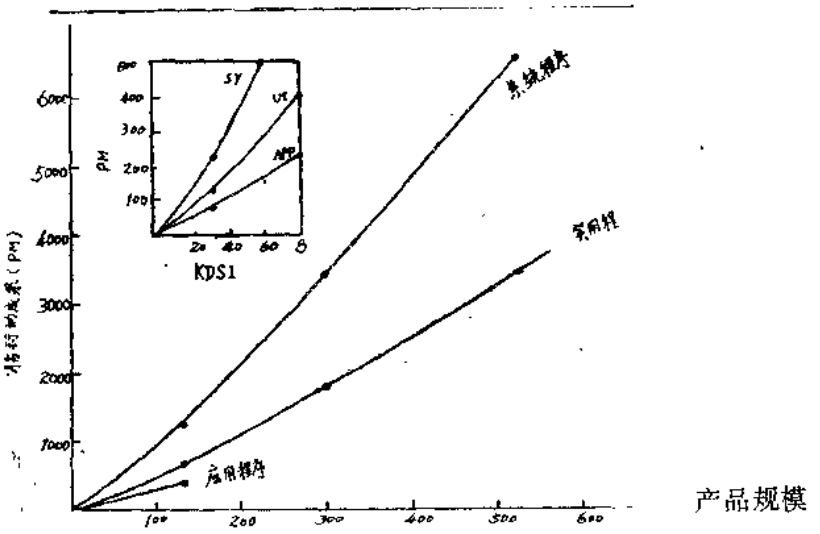


图 1 COCOMO工作估算(Boehm)

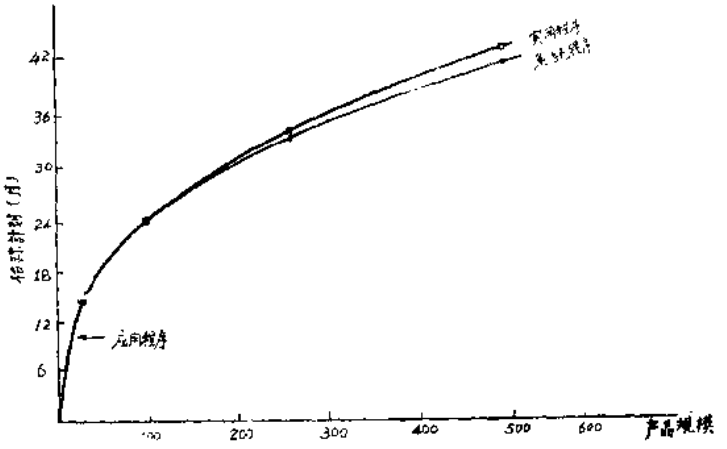


图 2 COCOMO模式估算(Boehm)

越大。

Boehm给出的程序开发时间是：

应用程序：$TDEV=2.5*(PM)^{0.38}$

实用程序：$TDEV=2.5*(PM)^{0.35}$

系统程序：$TDEV=2.5*(PM)^{0.32}$

这些方程的图形如图 2 所示。从图 2 可以看出，这三类系统的开发时间大致相同。

例如，一个60K源程序的程序不管它属于那一个复杂级都可用约18个月的时间开发完成。我们举此例是想说明，对于同样规模的程序，实用程序或系统程序有较多的机会采用并行工作完成，而应用程序则较少，甚至也许应用课题的工作计划就完全不同。

当有了一个课题的整个程序员人—月数和所要求的额定开发时间，用简单除法便可

得到工作人员数目。例如对于60 KDSI程序，可得如下结果：

应用程序：176.6PM/17.85MO=9.9个程序员

实用程序：294PM/18.3MO=16个程序员

系统程序：489.6PM/181.1MO=27个程序员

这些数字表示平均人员数，在分析和总体设计阶段只需较少的人员，而在实现阶段则需要平均人员数的125%至150%。人员数目的估算要在后面讨论。

需要强调一点，这些结果仅用作说明。只有当仔细读过Boehm的文章并知道了这些方程式的假设前题和限制条件后，才可用来估算软件成果。而且，这些方程的独立变量被引来源指令。因此，这种估算不会比我们自己去估算程序中的最终指令数好。这是规划阶段中较难的估算。

在估算一个软件产品的源指令数目时常出的一个错误，是低估了所要求的内务操作码的数目。内务操作码是源代码的一部分，用来处理输入/输出、交互用户通讯、人机接口工程、错误检查和处理。内务操作码常超过软件产品中源代码的50%，有时达90%。通常，估算操纵数据和完成计算的代码次比估算内务代码数更容易。当估算整个源代码行数时，若根据前面的值估算则容易估错。

产品规模

一个大的软件产品开发费用显然高于小产品的开发费用。Boehm的方程指出，所要求的软件项目的增长率是按源指令数目的指数递增的，指数系数略大于1。其它研究者也开发了类似于Boehm的方程式。表3给出了一些不同的项目和开发计划的估算者，它们都是从Boehm方程式演变而来的。如表3所示，一些研究者认为项目增长率是按指数略小于1而增长的，但大多数研究者使用的指数系数为1.05至1.83。

表3　　项目和规划估算者（Boehm）

项目方程	计划方程	参照
MM=5.2(KDSI)**0.91	TDEV=2.47(MM)**0.35	Walston and Felix(11)
MM=4.9(KDSI)**0.98	TDEV=2.04(MM)**0.36	Nelson(12)
MM=1.5(KDSI)**1.02	TDEV=4.38(MM)**0.25	Freburger and Basili(13)
MM=2.4(KDSI)**1.05	TDEV=2.50(MM)**0.38	Boehm(3)
MM=3.0(KDSI)**1.12	TDEV=2.50(MM)**0.35	Boehm(3)
MM=3.6(KDSI)**1.20	TDEV=2.50(MM)**0.32	Boehm(3)
MM=1.0(KDSI)**1.40—		Jones(14)
MM=0.7(KDSI)**1.50—		Halstead(15)
MM=28(KDSI)**1.83—		Schneider(16)

用指数0.91和1.83导致对一个产品的成果估值为1.88和3.5，为已知产品的2倍，得出一些产品的系数为8.1和67.6，为已知产品的10倍。这些估算出现差异是由于产品系数为1.86（3.5/1.88），差大约2倍，产品为8.3（67.6/8.1），相差大约10倍。我们假定所有影响费用的其它因素保持不变，在用上述的指数因子进行估算时，对是已知产品规模2倍的产品工作的估算可用因子2去除，对是已知产品规模10倍的产品工作的估算用因子10去除。

注意，表3中的开发时间估算者与项目

估算者相差不大，但他们和我们使算已交付的源指令数的能力是极相一致的。

可用时间

整个课题项目对于完成课题的所用时间是很敏感的。一些调查者研究了优化开发时间的问题，其中多数人认为：无论开发时间比优化时间多还是少，软件课题都需要完整的工作成果。这些研究结果归纳在图3中。

图3中最明显的特性是Putnam曲线。按照Putnam的看法，课题项目与开发时间的四次幂成反比，E=K/（Td＊＊4）。这条曲线指出对计划压缩的严励惩罚和对将课题计划扩展的高度奖励。例如：对100个程序员一大一月课题的开发计划时间加倍将会减少所需要的100/2＊＊4=6.25个程序员人一月的总的工作计划。该公式会带来不可能的结果，即如果开发时间趋于无穷则成果为零。实际上，由Putnam开发的SLIM费用估算模式用线性规划方法将四次幂曲线限制在相对于正常开发时间的一个小区间内。尽管如此，许多研究者认为Putnam曲线还是较灵敏的，而且当在优化时间上增加开发时间，将使得整个成果增加。

Putnam还指出，无论用多少人工或资源，也不可将开发计划时间压缩到低于一般

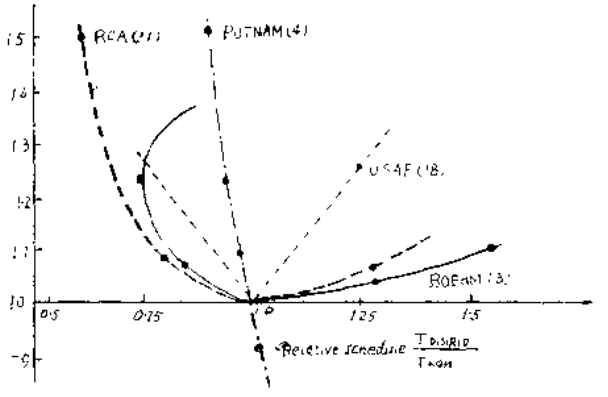


图3 计划外的相对成果（Boehm③）

计划时间的86%。四次幂倒数的方程式反示：计划时间压缩到0.86，则人员数需要增加到1.82倍。

图3中的其它曲线都较接近，特别是有关计划的压缩。Boehm研究了63个软件设计后发现，利用他的费用估算模式预测，仅有4个设计的压缩因子低于开发时间的75%。在这个研究基础上他指出："除了软件工程不能通过增加人员和购买设备来减少计划时间外，还有一个限制，即限定值为正常计划时间的75%。

需要的可靠性等级

软件可靠性可以定义为：一个程序在指定的时间周期、在指定的条件下完成所要求功能的概率。可靠性可以用源代码的准确性、耐久性、完整性和一致性等术语来表示。软件产品可以建立可靠性特性，通过提高对项目的分析、设计、完成、验证和确认等级，可以确保高可靠性，但也需要增加相应的费用。

需要的可靠性等级可在计划阶段通过考虑软件失效的费用来建立。在某些情况下，产品故障对用户仅有轻微的影响，而另外一些产品故障将会导致大的财政损失或对人类生活产生较大的危害。

Boehm描述了五类可靠性并且为每一类推荐了一个开发项目因子。表4给出了Boehm

表 4　　软件可靠性的开发项目因子

类　别	保障影响	项目因子
最低	轻微的影响	0.75
低	有损失但易恢复	0.88
一般	恢复损失有些困难	1.00
高	高财政损失	1.15
最高	危害人类生活	1.40

可靠性分类及每一类的项目因子。注意：因子范围从最低可靠性的0.75到最高可靠性的1.4，因此，项目比率为1.87：(1.4/0.75)。

技术等级

软件开发工程中的技术等级由编程语言、抽象计算机（硬件加软件）、编程习惯和使用的软件工具等反映出来。众所周知，每日所写的源指令数目与所用的语言完全无关，而且用高级语言如Fortran和Pascal等所写的程序语句一般要扩展成若干机器级语句。因此，使用高级语言而不是汇编语言，可提高程序员的效率，其因子为5—10。另外，高级语言的类型检查法则及自文件系统等方面可完善高级语言程序的可靠性和可修改性。现代编程语言（如Ada）提供了提高程序员生产率和软件可靠性的附加特性。这些特性包括类别检查、数据抽象、分别编译、异常处理、中断处理和并行处理方式。

抽象机是一组在开发过程中使用的软、硬件功能的集合。对抽象机的通晓、抽象机的稳定性及易访问性，都影响程序员的生产率，因而也就影响软件工程的费用。如果程序员必须学习一种新的机器环境作为开发过程的一部分，或者机器正与软件并行开发，或者限制程序员仅可访问某机器，则生产率将下降。

现代编程实践包括系统分析及设计技术、结构设计表示法、整体检查、结构编码、系统测试、程序开发库的使用。软件工具包括初等工具（如汇编程序和基本调试辅助程序等），编译程序和连接编辑程序，交互式正文编辑程序和数据库管理系统，程序设计语言处理程序和要求说明分析程序以及全集中式开发环境，包括配置管理和自动验证工具。

Boehm为现代编程习惯提供了从1.24（非现代方式）到0.82(全部使用现代方式)的成果因子，为软件工具提供了从1.24（最基本工具）到0.83（高级开发工具）的成果因子。因此，使用现代方式和现代开发工具可分别将编程项目减少到用汇编和原始开发工具编程工作的0.67（0.82/1.24）。按Boehm的观点，同时使用现代方式和现代开发工具可将开发工作项目减少到使用原始工具和技术所需开发工作的0.45（0.67*0.67）。

软件费用估算技术

在多数机构中，软件费用估算是依据已

有的性能经验。历史数据常用来验证费用因子并且决定在该机构环境中各种因子的相对重要性。这当然也意味着必须对当前的工程采集费用和生产率数据，以便对未来的工程进行估算。

费用估算可以采取自上而下或自下而上的方法。自上而下方式首先估算系统费用，如开发该系统所需的计算机资源和人员以及配置管理、质量保证、系统集成、培训和公布推广等费用。通过调查已完成的类似工程的费用可以获得人员的费用标准。

自下而上费用估算法首先估算开发每个模块或子系统的费用。合成这些费用便得到总体估算。自上而下估算具有在系统级估算费用的优点，但可能会忽略要被开发的某些模块的一系列技术因素。自下而上估算强调了开发单独系统模块的相应费用，但可能没有顾计到系统级费用，如配置管理和质量控制。实际上，这两种费用估算方式都应并发，通过反复比较删去那些不同之处。

专家判断

相当广泛使用的费用估算技术是专家判断，这是一种固有的自上而下的估算技术。专家判断依赖于经验、背景和该机构中一个或若干关键人物的商业观点。

专家从如下方式得出费用估算：

要被开发的系统是一个过程控制系统，它与去年开发的系统相似，去年用了十个月时间，花费1百万美元。我们没有因为此项工程而变富，但却获得了预期的利益，因此就不需要调整工程基线。新系统有类似控制功能，但要控制25%以上的活动范围，因此应增加25%的时间和费用估算。前一个系统是我们开发的这一类系统中的第一个，然而，我们将用同样的计算机和外部显示和（或）控制设备，并可用很多同样的人类开发新系统，这样可把估算值减少20%。而且，我们可以借用前一个产品的低级代码，从而减少时间和费用估算值25%。考虑这些以后的最终结果是时间和费用减少20%，因此得出估算值为80万美元和8个月的开发时间。我们知道用户预算该系统1百万美元和1年的交货期，因此加上一个小的安全裕量，可对该系统投标为85万美元和9个月的开发时间。

由专家进行判断的最大好处即经验，这也可能是一种不利的条件。专家可以确定该工程是与前一个相类似，但也会忽略掉使新工程完全不同的因素。或者，进行估算的专家并没有与当前工程类似的经验。

为了弥补这些因素，有时是由一个专家小组准备出一个一致的估算，使个人的忽略和对某个专门工程的不熟悉减少到最低程度，这样还可中和个人偏见，通过对估算进行整体优化来赢得合同（有时是潜意识的）。小组估算的主要不利之处是组内人员之间的关系是动态的，而且可能在组内形不成一致意见。小组成员也会因为政治考虑、组内权威人士的出现或比较武断的小组成员的优势而不公正。Delphi技术可用来克服这些不利因素。

Delphi 费用估算法

Delphi技术是在1948年由美国兰德公司开发的，它使专家们获得了一致意见而不会产生小组会议的反向边缘效果。Delphi技术可用下列方式用于软件费用估算：

1. 一位协调人为每一位估算者提供系统定义文件和记录费用估算的方式。

2. 估算者不隐各地研究定义和完成估算。他们可以向协调人询问问题，但互相之间不准讨论。

3. 协调人准备并分发估算者答案的摘要，包括估算者指出的任何独特的原理阐述。

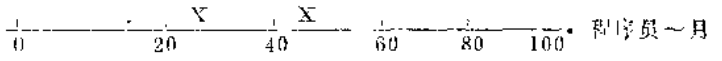
4. 估算者用前次估算结果来完成另一个估算，并且仍然采取无记名方式。在小组中具有尖锐不同估算值的估算者用隐名方法

工程：操作系统　　日 期：6/6/84

第三轮估算的范围

你的估算

中间估算



下一轮你的估算：35 P M

你估算的原理阐述：

看上去象一个标准过程控制操作系统。

我们的人员对这样一个系统已有了许多经验。

他们这样做不会有困难。

我们算增加估算，由于另一个估算者提到的新DMA通道问题。

图 4　Delphi 费用估算的方式

给出他们估算的正确性证明。

5．这样一个过程按需要可反复进行多次。在整个过程中不允许小组讨论。

图 4 给出了Delphi费用估算的方式。

下面的方法是对标准Delphi技术进行了改进，它在保证隐名的情况下增加了组内信息交换：

1．协调者为每一位估算者提供一份系统定义和估算方式。

2．估算者研究定义，并由协调者召开一个小组会议，估算者和协调者可以互相讨论估算问题。

3．估算者用隐名方式完成自己的估算。

4．协调者准备估算的总结，但并不记录任何原理阐述。

5．协调者召开小组会集中讨论估算分歧比较大的地方。

6．估算者再次隐名完成另一个估算。这个过程按需要可重复多次。

可能经过多轮估算仍得不出一致的估算值。在这种情况下，协调者必须与每一位估算者讨论这个问题，从而找出分歧的原因。协调者还应收集其他信息并告诉估算者，从而解决观点分歧问题。

工作分类结构

专家判断法和小组统一方式是自上而下的估算技术。工作分类结构方法是一种自下而上的估算工具。工作分类结构是一个说明系统每个部分的层次图。一个工作分类结构WBS图，既可以指出产品层次，又可以指出过程层次。

产品层次结构用于对产品成分进行分类，并指出这些成分间内部联系的方式。过程层次的WBS图指出工作活动及它们之间的关系。在图5a和5b中分别给出了典型的产品WBS图和过程WBS图。使用WBS技术时，先对图中每个成分进行费用估算，然后求和。

某些计划人员用产品及过程工作分类结构图进行费用估算。WBS技术的主要优点是能指出并说明不同的过程和产品因子，并且可在估算中严格确定包括哪些费用。

专家判断法、小组统一方式和工作分类结构都是被广泛使用的费用估算技术。许多组织是同时使用这三种方法进行反复估算，直至解决了不同之处为止。

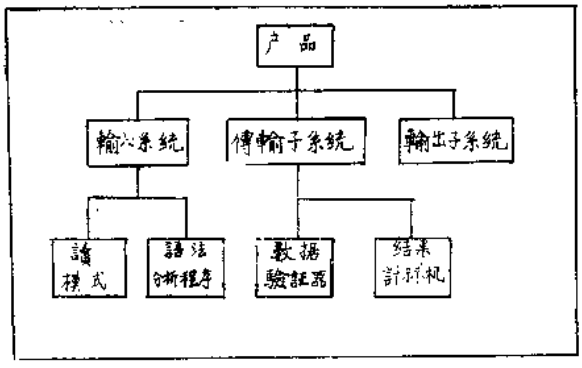


图 5a　产品工作分类结构

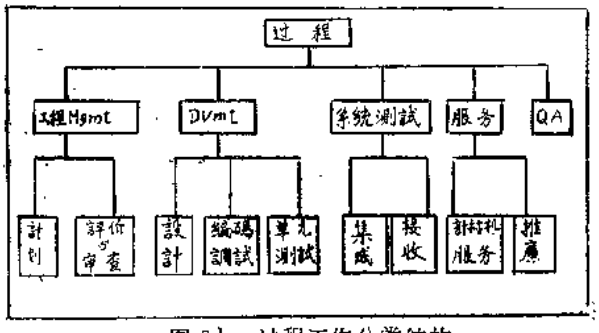


图 5b　过程工作分类结构

算法费用模式

算法费用估算者计算软件系统的估算费用，作为组成系统的那些模块和子系统的总费用。因此，算法模式是自下而上的估算程序。

结构费用模式（COCOMO）是Boehm描述的一个算法费用模式。下面简述COCOMO法。

当使用COCOMO法时，前面所描述过的方程式可用来对程序员人—月及一个程序单位的开发计划表进行一般估算，即依据程序单位已估算的交货源指令数（DSI）进行。这时，用成果乘式对产品属性、计算机属性、人员属性和工程属性的估算进行调整。表5归纳了COCOMO成果乘式和它们的值域。通过对63个软件开发题目的数据进行验证，并且在一个软件专家组中使用DelPhi技术，从而得出此方程式及成果乘式。

COCOMO方程式允许一些假定条件。例如，普通的组织模式（应用程序）方程适用于如下环境：

小到中等规模的计划(2K到32K DSI)，

熟悉的应用领域

稳定的、众所了解的虚拟机

室内开发项目

项目乘式用来修正这些假定条件，估算包括下面的活动：

通过可接收测试的设计

文件和评价的费用

题目管理人员及程序管理人员的费用

项目估算不包括计划和分析费用、安装机培训费用以及秘书、门卫和计算机操作员的费用。DSI估算包括工作控制语句和源语句，但是不包括解释语句和未被修改的实用程序。每个DSI可以看作是一行或者是一个卡片映象，一个程序员人—月包括152个程序员-小时。

由COCOMO结算的软件工程的特性，有如下其它假设：

表 5　　COCOMO 项目乘式

乘　式	值　域
产品属性	
所需可靠性	0.75至1.40
数据库规模	0.94至1.16
产品复杂性	0.70至1.65
计算机属性	
执行时间限制	1.00至1.66
主存限制	1.00至1.56
虚拟机易变性	0.87至1.30
计算机同转时间	0.87至1.15
人员属性	
分析员能力	1.46至0.71
程序员能力	1.42至0.70
应用经验	1.29至0.82
虚拟机经验	1.21至0.90
编程语言经验	1.14至0.95
工程属性	
现代编程实践的利用	1.24至0.82
软件工具的使用	1.24至0.83
所需开支计划表	1.23至1.10

- 对要求的详细定义机有效性检查由少数有能力的人完成。
- 整个工程的要求要保持稳定。
- 对体系结构设计的详细定义和有效性检查由少数有能力的人承担。
- 具体设计、编码、单元测试由许多组程序员并行实现。
- 在初期测试计划基础上进行集中测试。
- 大部分接口错误在集中测试前由单元测试、检验和预演时发现。
- 作为开发过程的一部分，记录文本被逐渐增加。

换句话说，软件工程的系统技术被用于整个开发过程。

下面是使用COCOMO的算法费用估算的例子。

要开发的产品是一个10 KDSI，是在一个商业用微处理机上做的用于远程通讯过程的嵌入式软件产品。对一个10kDSI嵌入式产品的额定项目平衡，预测需要44.4个程序员人—月机8.4个月来开发产品。

$$PM=2.8*(10)**1.20=44.4$$

$$TDEV=2.5*(44)**0.32=8.4$$

项目乘式被用来调整对工程超额点的估算。例如，软件要求是高度复杂的，但是，这可以由计划使用高质量的分析员和程序员来平衡。有关这项工程的项目乘式如表6所示。

项目调节因子1.17是该项目乘式的结果。当提供于定额估算时，项目调节因子产生一个51.9个程序员人—月和8.8个月开发时间的估算，假定程序员和分析员每个人—月花费6,000美元，工程人员的总费用将是：

美元总数＝(51.9PM)*(6,000美元/每个PM)

＝311,400美元

表 6　　嵌入式远程通讯项目乘式举例

乘式	理由	值
可靠性	仅本地使用，没有严重恢复问题（定额）	1.00
数据库	20,000 字节（低）	0.94
复杂性	远程通讯处理（非常高）	1.30
计时	将用 70% 的处理时间（高）	1.11
存储	64K 可利用45K(高)	1.06
机器	稳定，商用微机（定额）	1.00
转向	平均 2 小时（定额）	1.00
分析员	有经验的高级工作人员（高）	0.86
程序员	有经验的高级工作人员（高）	0.86
阅历	干过 8 年远程通讯工作（定额）	1.00
阅历	用过 6 个月微机(低)	1.10
阅历	使用语言达 12 个月（定额）	1.00
实践	有 1 年以上的现代技术实践（高）	0.91
工具	Basic 微型机软件（低）	1.10
计划	9 个月：估计 8.4 个月（定额）	1.00
项目调整因子=1.17		

COCOMO也可以通过完成费用估算的灵敏度分析来用于开发过程的折衷方案。例如，能力低一些的人员可以以每个人—月1,000美元的费用来实施远程通讯处理程序，然而，分析员和程序员能力乘式必须都以0.86增加到1.00，以反映其变化。在这些情况下，项目调节因子变成1.58，并且美元账目变成350,760美元，净增费用39,360美元。再就是，估算的开发计划增加到9.7个月。因此，最好使用较高技术水平和较高费用的人员。

另一方面，假定我们能把微处机内存从46k增加到96k，而加价10,000美元，增加的内存将减少存储乘式从1.06减到1.00，结果是有一个新的费用调节因子1.10和一个新估算费用293,000美元，净节省费用18,400美元。另外，项目缩减也使开发时间估算产生一个微弱的减少。因此，购买附加内存是一项明智的决定，不但在开发期间对节省费用有利，而且为维护期间的产品提高和修改，在主存上提供了一个裕量。

表 7 列出了使用COCOMO 完成一个费用估算所需要的步骤。COCOMO最大的优点是该模型可以在某个组织结构内洞悉费用因子。数据可以集中采集和分析，新因子可以被识别，并且在需要校准COCOMO到本地环境时可以调整项目乘式。或许COCOMO的最大弱点是乘式项目调节因子的使用要假定各种各样的项目乘式是独立的。事实上，变化一个因子经常意味着其它因子也应该被调整。通常，不容易搞清楚一个因子对其它因子的影响是如何变化的。

这里列出的COCOMO版本是Boehm的中间模型。在他的正文里，Boehm也描述了一个详细的模型，它解释说明了在开发阶段各种项目乘式的不同情况。

最后要注意的是：大部分费用估算模型不包括对当前代码重新使用的费用估算。有时，重用码的费用是开发相等量新码费用的20%。

雇用人员水平估算

一项软件开发工程所需人员的数量不是一个常数。一般，计划和分析由一小组人就可完成，总体设计需要一大组人而不是一小组人，具体设计需要更多的人，工具和系统测试需要最多的人员。早期维修维护需要大批的工作人员，但是过一段时间后这一大批人应该减下来。在不需要大的提高或改进的情况下，用于维护的人员应该是少量的。

1958年，Nordeu观察到：研究和开发工程遵循一个周期规律，即计划、设计、初始定型、开发和使用，其相应的人员利用情况如图 6 所示。

表 7　　使用 COCOMO 的费用估算过程

1.识别产品的所有子系统和模块
2.估算每块模块的规模，设计每个子系统和整个系统的规模
3.为每个模块指定模块级项目乘式，模块级乘式是：产品复杂性、程序员能力、虚拟机实践和编程语言实践
4.为每个模块计算模块项目和开发时间的估算，使用定额估算方程和模块级项目乘式
5.为每个子系统指定保持11个项目乘式
6.从第4.5步对每个子系统计算估算的项目和开发时间
7.从第 6 步，计算总的系统项目和开发时间
8.在估算的基础上进行灵敏度分析，以建立折衷方案
9.如其它开发费用，例如在估算中未包括的计划和分析费用
10.把估算同用从顶而下Delphi估算法开发的估算相比较。识别和矫正估算中的不同点

他同时也观察到：图 6 中曲线之下面积的总和可以逼近Rayleigh方程。Rayleigh方程如图 7 所示。Rayleigh曲线上任何特殊的点都表示在那一时刻相应需要的全工作日等价人员的数目。注意Rayleigh曲线有两个参数：td为时间，在该时刻曲线达到最大值；K为曲线下的总面积，它表示该工程所需的总工作量。

在1976年，Putnam作报告时说：软件产品的生命期所需的项目人员的工作水平有一个相似的框架。Putnam随后研究了50 个军用软件工程和150个其它的工程，以确定Rayleigh曲线如何被用来描述软件生命周期。

对其它事情，Putnam还观察到：Rayligh曲线达到其最大值的时间td，与许多软

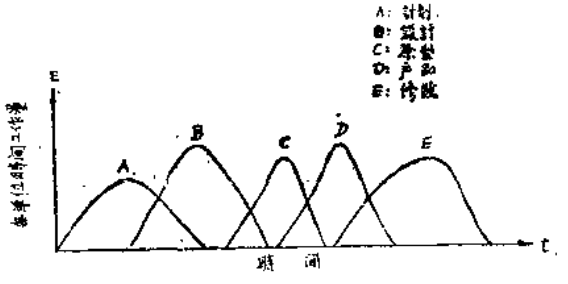


图 6　研究和开发工程中的周期(Nardeu)

件产品的系统调试和产品交付的时间是相对应的。Rayleigh曲线下的任何间隔的面积都表示在那段间隔里总的工作量。Rayleigh曲线下的面积大约有40%是在td的左边，60%是在td的右边。这是对许多产品生命周期的开发和维护间的工作分布的合理估算。Putnam对Rayleigh曲线的解释如图8所示。注意：计划需求分析和功能设计（外部和总体结构设计）都不包括在该工程曲线内。

Boehm观察到Rayleigh曲线是从体系结构设计到实现和系统调试的开发周期中对人员需求的合理的准确的估算方法，如果使用的是0.3td和1.7td之间的曲线部分，Rayleigh曲线是下面的公式：

$$FSP = PM\left(\frac{0.15TDEV + 0.7t}{0.25(TDEV)^2}\right)$$

$$e^{-\frac{(0.15TDEV+0.7t)^2}{0.5(TDEV)^2}}$$

这里PM是对产品开发（不包括计划、分析）的程序员人—月的估算数，TDEV是估算的开发时间。给定这两个因子后，在任何特定时间t所需的全工作日软件人员数FSP可以计算出来。h的变化范围从0.3 td到1.7 td。注意：td仍是雇用人员需求量的高峰时间，但是，它不再被解释成过去的开发时间。一个32KDSL、91PM和14个月的工程作为时间函数的工作人员需求曲线如图9所示。

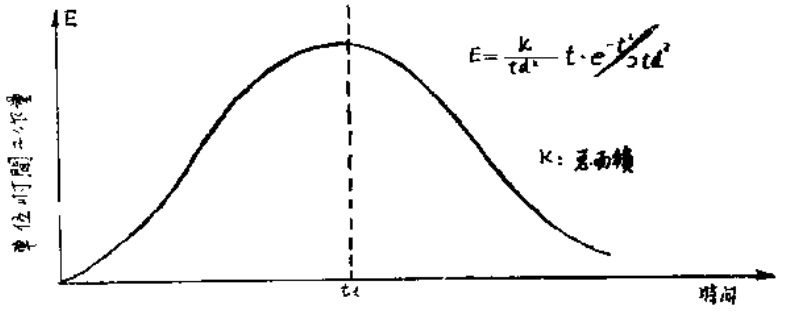


图 7　工作量对时间的Rayleigh曲线

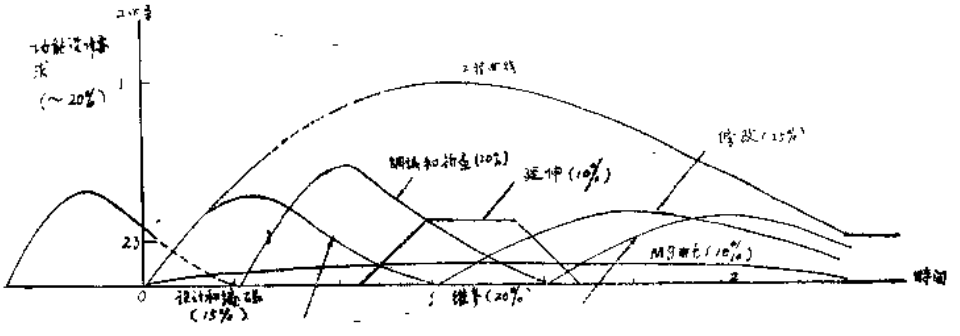


图 8　Putnam的Rayleigh曲线解释(Putnam)