软件工作量估算

软件项目的规模估算

确定了软件项目开发的生命周期模型，进行了工作任务分解，就建立了一个项目任务整体的框架结构。

另外一方面，一个良好的软件项目计划的建立，还必须估算准备开发的软件项目的任务大小、资源情况、投入的成本、限制因素等，进行充分的估算，最后，根据估算，才能制定出合理的项目开发计划。

具体来说，要估算的内容包括：

- 软件工作产品的规模
- 软件项目的工作量和成本
- 软件项目的进度
- 项目所需要的人员、计算机等资源

什么是软件项目的规模

在一个软件项目中，项目组要完成的工作产品，是规模评估的对象，那么，项目组要完成的工作产品包括些什么？是最后要交付的（向用户和向组织二个方面）程序、文档。

但是，项目组并不是只要完成最后交付的程序和文档，就可以了。在交付前，要进行确认和验证测试，为此，要进行质量控制有关的工作。再往前追述，项目组还必须做配置管理、需求管理，以及项目管理。这些都有工作量。那么，软件规模如何估算？

现在，常用的办法，是通过对软件程序的规模进行估算的办法，来间接反映软件项目的规模。规模是工作量的一个方面，并不能说规模大，工作量就大。在这方面，并不一定是完全等同的。显然，接口控制程序的程序量可能并不大，但是程序量比较大的报表处理程序的工作量就大。这种不合理性，一般通过相关的程序复杂度、难度，加以调节。这个问题，在相应的评估算法中，采用加权因子的方法，加以调整。同样，程序规模的增长，会带来支持和管理工作成指数规模的增长。因此，这也是需要注意的地方。

用什么来估算软件项目的规模

软件的规模计算，从有软件的一天开始，就是一个没有解决的问题。

没有解决的难题是，现在越来越没有办法给出评价程序量多少的统一尺度。在程序设计的早期，直接的编码量（字节数）是度量程序量的简单办法。但是，没有多久，这个办法就受到了挑战。因为有一个好的算法（例如：好的循环控制），可以节省大量的程序编码，但工作量（设计所花的时间、测试的复杂度）等，反而并没有节省开发的精力和时间。因此，程序量作为工作量的度量标准，显然是不正确的。那么现在，在完全不同的系统、应用环境下，提出统一和易于运用的度量标准，是非常困难的。

为了解决问题，在CMM2的计划管理中，已经提出了一些度量的实例，包括：功能点数、特征点数、编码行数（LOC）、需求数或页数等。还可以有：模块数目，表格数，用户界面屏数，及数据结构等，作为规模评估的参考。

度量软件项目规模的尺度，是一个相对值，而不存在绝对值。

软件项目规模的估算方法——LOC法

LOC(Line of Code)——

一个衡量软件项目规模最常用的方法：

LOC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（JCL：Job Control Language）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。

单位编码行（1LOC）的价值和人月均编码行数可以体现一个软件生产组织的生产能力。组织可以根据对历史项目的审计来核算组织的单行编码价值。

概念说明

- LOC
- NCLOC
- CLOC
- LOC=NCLOC+CLOC

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，则可估计该项目源编码大约为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（包括人均工资、福利、办公费用公滩等），则该项目中单位LOC的价值为：

（240×10000）/150000＝16元/LOC

该项目的人月均编码行数为：

150000/240=625LOC/人月

软件项目规模的估算方法——Delphi法

Delphi法是最流行的专家评估技术，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，但专家“专”的程度及对项目的理解程度是工作中的难点，尽管Delphi技术可以减轻这种偏差，专家评估技术在评定一个新软件实际成本时通常用得不多，但是，这种方式对决定其它模型的输入时特别有用。

Delphi法的步骤是：

1、协调人向各专家提供项目规格和估计表格；
2、协调人召集小组会各专家讨论与规模相关的因素；
3、各专家匿名填写迭代表格；
4、协调人整理出一个估计总结，以迭代表的形式返回专家；
5、协调人召集小组会，讨论较大的估计差异；
6、专家复查估计总结并在迭代表上提交另一个匿名估计；
7、重复4-6， 直到达到一个最低和最高估计的一致。

软件项目规模的估算方法——类比法

类比法适合评估一些与历史项目在应用领域、环境和复杂度的相似的项目，通过新项目与历史项目的比较得到规模估计。类比法估计结果的精确度取决于历史项目数据的完整性和准确度。因此，用好类比法的前提条件之一是组织建立起较好的项目后评价与分析机制，对历史项目的数据分析是可信赖的。

类比法的基本步骤是：

1、整理出项目功能列表和实现每个功能的编码行数；
2、标识出每个功能列表与历史项目的相同点和不同点，特别要注意历史项目做得不够的地方；
3、通过步骤1和2得出各个功能的估计值；
4、产生规模估计。

- 前面介绍的代码行技术是比较简单的定量估算软件规模的方法。
- 这种方法依据以往开发类似产品的经验和历史数据，估算实现一个功能所需要的源程序行数。

当有以往开发类似产品的历史数据可供参考时，这种方法估算出的数值还是比较准确的。

代码行技术

- 代码行技术的优点：
  - ✓ 代码是所有软件项目开发都包含的“产品”，而且代码行数很容易计算。

- 代码行技术的缺点：
  - ✓ 源程序仅是软件配置的一个成分，用它的规模代表整个软件规模不太合理；
  - ✓ 用不同语言实现同一个软件所需的代码行数并不相同，即它依赖于所使用的编程语言；
  - ✓ 不适合于非过程语言。

功能点技术

- 是为克服代码行技术缺点提出的；
- 它涉及多种因素的度量方法；
- 功能点技术根据对软件信息域特性和软件复杂性的评估结果，估算软件规模，所以在系统设计初期就能够估算出软件项目的规模。

信息域特性

- 产品信息域的5个特性：
  - 输入项数(Inp)

    用户向软件输入的项数，这些输入为软件提供了面向应用的数据
  - 输出项数(Out)

    软件向用户输出的项数，它们向用户提供面向应用的信息
  - 查询数(Inq)

    查询，即一次联机输入，它导致软件以联机输出方式产生某种即时响应。
  - 主文件数(Maf)

    逻辑主文件（数据的一个逻辑集合）的数目。
  - 外部接口数(Inf)。

    机器可读的全部接口的数量。

功能点技术基本原理

- 使用5个信息域的“加权和”CT与14个技术因素的“复杂性调节值”F_i来计算功能点FP:

$$FP = CT \times (0.65 + 0.01 \times \sum_{i=1}^{14} F_i)$$

TCF

估算功能点的步骤

1、计算未调整的功能点数CT

- 首先把信息域的每个特性都分类为简单级、平均级或复杂级，根据等级按下表分配功能点数：

估算功能点的步骤

特性系数 \ 复杂级别	简单	平均	复杂
输入系数a_1	3	4	5
输出系数a_2	4	5	7
查询系数a_3	3	4	6
文件系数a_4	7	10	15
接口系数a_5	5	7	10

然后用下式计算未调整的功能点数UFP：

$CT=a_1\times Inp+a_2\times Out+a_3\times Inq+a_4\times Maf+a_5\times Inf$

估算功能点的步骤

2、计算技术复杂因子TCF

- ✓软件复杂度的估算基于14个问题，逐一对各问题做出复杂度估计，其取值范围是0-5。

$$DI = \sum_{i=1}^{14} F_i$$

$$TCF = 0.65 + 0.01 \times DI$$

估算功能点的步骤

“没有影响”取值0
“偶然的”取值1
“适中的”取值2
“普通的”取值3
“重要的”取值4
“极重要的”取值5

系统需要可靠的备份和复原吗？
需要数据通信吗？
有分布处理功能吗？
性能很关键吗？
系统是否在一个已有的、很实用的操作环境中运行？
系统需要联机入口吗？
联机入口需要在多屏幕或多操作之间切换以完成输入？
系统联机需要更新主文件吗？
系统的输入、输出、文件或查询很复杂吗？
系统内部处理复杂吗？
代码需要被设计成可复用的吗？
设计中要包括转换及安装吗？
系统的设计支持不同组织的多次安装吗？
系统的设计有利于用户修改和使用吗？

功能点技术

- 功能点技术没有涉及系统本身的算法复杂性。
- 因此，它仅仅适合算法比较简单的事务处理软件的规模度量；对算法较复杂的大型软件系统并不适应。

工作量估算

- 软件估算模型使用由经验导出的公式来预测软件开发工作量，其中，工作量是软件规模(LOC或FP)的函数，工作量的单位通常是人月（pm）。
  - ✓静态单变量模型
  - ✓动态多变量模型
  - ✓COCOMO2模型

支持大多数估算模型的经验数据，都是从有限个项目的样本集中总结出来的，因此，没有一个估算模型可以适应所有类型的软件和开发环境。

静态单变量模型

- 静态单变量的总体结构形式如下：

$$E = A + B \times (ev)^C$$

其中，A、B、C是由经验数据导出的常数，E是以人月为单位的工作量，ev是估算变量（KLOC或FP）。

下面给出典型的静态单变量模型：

1) 面向KLOC的估算模型
2) 面向FP的估算模型

面向KLOC的估算模型

- ✓**Walston_Felix (IBM模型)**

$$E = 5.2 \times KLOC^{0.91}$$

- ✓**Bailey_Basili模型**

$$E = 5.5 + 0.73 \times KLOC^{1.16}$$

- ✓**Boehm简单模型**

$$E = 3.2 \times KLOC^{1.05}$$

- ✓**Doty模型(KLOC>9时适合)**

$$E = 5.288 \times KLOC^{1.047}$$

面向FP的估算模型

- **Albrecht&Gaffney模型**

$$E = -13.39 + 0.0545FP$$

- **Maston，Barnett和Mellichamp模型**

$$E = 585.7 + 15.12FP$$

静态单变量的估算模型

- 从上面可以看出，对于相同的KLOC或LP，用不同的模型估算的结果各不相同。
- 主要原因：
  这些模型都仅仅依据若干领域中有限个项目的经验数据推导出来的，适应范围有限。

- 因此，必须根据当前项目特点选择适应的估算模型，并依据需要对相应模型作出调整。

动态多变量模型

- 动态多变量模型，即软件方程式，它是根据4000多个当代软件项目中收集的生产率数据推导出来的。
- 该模型把工作量看作是软件规模和开发时间的函数，其形式如下：

$$E = (LOC \times B^{0.333} / P)^3 \times (1/t)^4$$

动态多变量模型

其中：

- ✓ E是以人月或人年为单位的工作量；
- ✓ t是以月或年为单位的项目持续时间；
- ✓ B是特殊因子，它随着对测试、质量保证、文档及管理技术的需求的增加而缓慢增加。对于较小程序(KLOC=5~15)，B=0.16；而对于超过70KLOC的程序，B=0.39；
- ✓ P是生产率参数，它反映下属因素对工作量的影响：
  - 总体过程成熟度及管理水平；
  - 使用良好的软件工程实践的程度；
  - 使用程序设计语言的级别；
  - 软件环境状态；
  - 软件项目组的经验和技术；
  - 应用系统的复杂性等。

动态多变量模型

P的取值：

开发实时嵌入式软件时，P的典型值为2000；

开发电信系统时，P=10000；

对于商业软件来说，P=28000。

从上式可以看出，开发同一个软件产品（即LOC固定）的时候，如果把项目持续时间延长，则可降低完成项目所需要的工作量。

COCOMO模型

COCOMO模型—构造性成本模型

- 1981年Boehm在《软件工程经济学》中首次提出了COCOMO模型。
- 1997年Boehm等人提出的COCOMO2模型，是原始的COCOMO模型模型的修正版，它反映了十多年来人们在软件成本估算方面所积累的经验。

COCOMO模型

COCOMO模型给出了3个层次的工作量估算模型。这3个层次的模型在估算工作量时，对软件细节考虑的详尽程度逐级增加。这些模型既可以用于不同类型的项目，也可以用于同一个项目的不同开发阶段。这三个层次的估算模型分别是：

- ✓ 基本COCOMO模型
- ✓ 中间COCOMO模型
- ✓ 详细COCOMO模型

COCOMO模型基本原理

- COCOMO模型具有以下形式：

$$MM = C \times KLOC^{a} \times \prod_{i=1}^{15} f_i$$

其中，MM是开发工作量；
KLOC是估计的源代码行数（以千行为单位）；
a是模型系数；
f_i是成本因素

COCOMO模型基本原理

- 每个成本因素根据它的重要程度和影响大小赋予一定的值，可把这些成本因素划分为：
  - 产品因素
  - 计算机因素
  - 人员因素
  - 项目因素。

成本因素	级别					
	很低	低	正常	高	很高	极高
软件可靠性(RELY)	0.75	0.88	1.00	1.15	1.40	
数据库规模(DATA)		0.94	1.00	1.08	1.16	
软件复杂性(CPLX)	0.70	0.85	1.00	1.15	1.30	1.65
执行时间约束(TIME)			1.00	1.11	1.30	1.66
内存约束(STOR)			1.00	1.06	1.21	1.56
软件开发环境的变化(VIRT)		0.87	1.00	1.15	1.30	
开发环境的响应速度(TURN)		0.87	1.00	1.07	1.15	
系统分析员的能力(ACAP)	1.46	1.19	1.00	0.86	0.71	
应用经验(AEXP)	1.29	1.13	1.00	0.91	0.82	
程序员水平(PCAP)	1.42	1.17	1.00	0.86	0.70	
开发环境的经验(VEXP)	1.21	1.10	1.00	0.90		
程序设计语言的经验(LEXP)	1.14	1.07	1.00	0.95		
程序设计实践(MODP)	1.24	1.10	1.00	0.91	0.82	
软件工具的使用(TOOL)	1.24	1.10	1.00	0.91	0.83	
开发进度的要求（SCED)	1.23	1.08	1.00	1.04	1.10	

基本COCOMO模型

- 基本COCOMO模型是一个静态单变量模型，它把软件系统所需要的成本看作是程序大小单一变量的函数，用于系统级的粗略估算。

工作量 $MM=C\times KLOC^{a}$ （人月|人年）
开发时间 $D=cE^{d}$ （月|年）

中间COCOMO模型

- 中间COCOMO模型是一个静态多变量模型，它把软件系统所需要的成本看作是程序大小和一系列成本驱动属性的函数。该模型将软件系统区分为系统和部件两个层次；系统是由部件组成。

$$MM = C \times KLOC^{a} \times \prod_{i=1}^{15} f_i$$

详细COCOMO模型

- 详细COCOMO模型除了包括中级COCOMO模型中所考虑的因素以外，并对工作量调节因子在软件过程中每个步骤的影响作出详细评估。
- 该模型则将软件系统分为系统、子系统、模块三个层次。

对规模估算的修正

为了补充单一算法的不足，实际上，软件项目经理常采用其他方法，来进行“校正”和补充。这些方法是：

（1）把大块的任务分解为小规模的任务。例如：使用WBS对任务分解，然后对分解到的最“末梢”功能，进行规模估算，然后，在进行相加。这种方法的好处是化复杂为简单。当我们无法把握大系统的规模时，我们可以比较有把握地估算小模块的规模。但是，问题是，各单独模块的规模之合，一般（从工作量上）并不等于系统的工作量之合。这就是我们说到的系统集成的工作量。

（2）采用与历史数据比较、修正的方法。

（3）采用最大值、最小值和最可能值折算的方法。

（4）对同一项目，至少使用二种以上工具和方法进行测算，避免一种方法的局限性。

（5）使用同行专家评审、评估小组集体投票取折中值的方法，博采众长。

（6）逐步逼近的方法；不把评估结果作为最终值，而是看成是一个逐步逼近的近似值，在以后的再评估中，可以进行调整。当项目刚刚启动，做第一次测算时，可以允许有30%以上的波动范围。在完成需求分析阶段，希望能达到波动控制在30%的范围内的目标。当基本完成详细设计的时候，估计的编码规模，应只能有20%、甚至更低的误差。

对规模估算的迭代修正

随着软件项目进程的逐步展开，需求被逐步地精细描述和定义，任务被依据需求进行分配，使用任务分解结构(WBS)的方法来建立需求（产品架构）与任务（软件开发过程）之间直接的、准确的联系，并实现详细化、文档化，工作产品被分解成各个工作元素。

产品架构必然要反映出项目的整体设计概念、层次结构，并从分解后的组件和模块功能分层中体现出来。

随着产品架构的精化，进一步定义出了开发各产品元素所需的过程任务。再把这些任务分配给不同的工程师。其目的是使任务分解架构中每一项任务，可以由个人或几个人的小组在较短的时间内完成。总之，任务分解架构越详细，则任务分配越明确，产品估算越准确，项目计划越完善，项目跟踪也越精确。

在这个基础上，经过一段时间的运行，一些典型的模块已经完成，所没有完成的只是相类似的模块等待接着继续开发。这时，对每一个产品元素的规模进行再估算和精细化调整的条件已经具备，根据一段时间的经验，重新调整软件项目规模估算，逐步求精，将获得更接近实际的估算结果。

序号	系数	因素	因素取值准则					得分
			1	2	3	4	5	
1	1	创造性要求	没有-在不同设备上重编程序	很少-具有更严格的要求	有限-具有新的接口	相当多-具有相当多的技巧	重大的-应用先进的技巧	
2	1	通用程度（数据库、中间平台、操作系统等）	很强的限制-单一目标	有限制-功能的范围是参量化的	有限的灵活性-允许格式上有变化	多用途格式有一个主题领域	很灵活能多设备范围广泛	
3	1	工作范围	同一办公室	同一栋楼	同一城市	跨城市	跨国	
4	1.2	目标范围的变化	没有	极少	偶尔有	经常	不断	
5	1.2	体系结构	单一结构	C/S B/S	B/A/S	C/A/S	三层以上跨语言平台	
6	1	设备复杂性	单机、常规处理	单机、常规处理，扩充的外设系统	多机，标准外设系统	多机，复杂的外设系统	一机控制多机自动I/O和显示	
7	1.2	人员	1~2人	3~5人	5~10人	10~18人	18人以上	
8	5	开发投资	6人月以下	6人月至3人年	3人年至10人年	10人年至30人	30人年以上	

9	1.2	重要程度	数据处理	对单一用户关系有影响	对用户群关系有影响	单位成败	国家安危	
10	1	对程序改变的完成时间要求	2周以上	1~2周	3~7天	1~3天	24小时以内	
11	1	对数据输入的响应时间要求	1天以上	60分钟~1天	1~60分钟	3秒~1分钟	3秒钟以内	
12	1.2	编程环境	4GL（PB、Delphi 、ASP等）	4GL(VC、CBuilder、JBuilder)	专业软件平台	小中大型机平台	汇编语言	
13	1	并行的软件开发	没有	有限	中等程度	很多	完全的并行开发	
14	1.2	代码复用程度	80%~100%	60%~80%	40%~60%	20%~40%	0%~20%	
15	1	终端用户数	1	2~5	6~20	21~50	51~	
				总分				

规模评分 必须具备	20～30	31～48	49～69	70～88	89～100
可行性研究报告	√	√	√	√	√
项目开发计划	√	√	√	√	√
风险管理计划表			√	√	√
技术白皮书			√	√	√
软件需求说明书			√	√	√
数据要求说明书		据 实 际 决 定	据实际决定	据实际决定	据实际决定
系统概要设计说明书			√	√	√
详细设计说明书				√	√
数据库设计说明书		据实际决定	据实际决定	据实际决定	据实际决定
用户手册或使用说明	√	√	√	√	√
操作手册			√	√	√
代码	√	√	√	√	√
测试计划（单元）			√	√	√
测试分析报告（单元）			√	√	√
测试计划（集成）		√	√	√	√
测试分析报告（集成）		√	√	√	√
测试计划（系统）					√
测试分析报告（系统）					√40
项目开发总结报告	√	√	√	√	√

附录：关于COCOMO模型

COCOMO模型

- **COCOMO模型包括基本模型、中级模型和详细模型三个子模型**
- **COCOMO基本模型是一个静态单变量模型，它利用以最终交付的源代码行数作为自变量的函数来计算软件的开发工作量和开发工期。**
- **COCOMO中级模型是在用源代码行数作为自变量的函数计算软件开发工作量的基础上，再用涉及产品、硬件、人员、项目等多方面属性的影响因素来调整工作量的估算。**
- **COCOMO详细模型包括了COCOMO中级模型的所有特性，但在用上述影响因素来调整工作量估算时，还需要考虑对软件工程过程中的每一个步骤的影响。**

•COCOMO基本模型成本测算的基本原理：根据最终交付的行数来计算开发工作量，并进而估计开发工期。

•COCOMO基本模型成本测算的公式如下：

$$PM=C1*(KDSI)^{K1}$$

$$TDEV=C2*(MM)^{K2}$$

PM—代表开发该软件所需要的人月数；

KDSI—所交付的千行源代码数

TDEV—该软件所需的开发时间，以月为单位

C1、C2、K1、K2—常量参数，采用不同的开发方式取值不同

COCOMO基本模型的常量参数值

开发方式	C1	K1	C2	K2
组织型方式	2.4	1.05	2.5	0.38
半分离方式	3.0	1.12	2.5	0.35
嵌入方式	3.6	1.20	2.5	0.32

COCOMO基本模型从软件规模和开发方式的特征出发，将开发工作分成组织型(有机式)、半分离型（半相连式）、嵌入型三种开发方式。

软件项目的工作量估算

软件开发项目的工作量，主要指软件开发各过程中所花费的工作量。与传统的制造业不同，软件的成本基本可以不考虑原材料和能源的消耗，主要是人的劳动的消耗。另外，软件也没有一个明显的制造过程，它的开发过程具有明显的一次性过程特征。因此，软件开发工作量的估算，应是从软件计划、需求分析、设计、编码、单元测试、集成测试到认证测试，整个开发过程所花费的工作量，作为工作量测算的依据。

我们在前一章已经介绍了软件的生命周期模型，采用什么样的生命周期模型，对使用什么样的工作量估算算法，有很大的影响。实际上，有些算法适合某一模型，有些则适合另一模型。采用面向对象技术为主的开发与传统的开发技术，在工作量的估算上，当然也不同。这些，都需要我们根据软件开发的具体特点，进行选择。

软件项目的工作量估算与进度估算的关系

- 软件项目工作量估算的结果是任务的人力和需时。

在工作量估算时，度量的任务需时是讨论以任务元素、子任务、项目任务为单位（我们称为：单位任务）的需时，它是计算成本、制定进度计划的依据。而在进度估算时，单位任务的需时，又是时间进度计划安排的基本数据来源。

- 二者最主要的区别是：

工作量估算时对时间的测算，注重的是最后获得的时间总量，或者是不同阶段、不同工作性质、不同成本因素下的时间量。例如：工作量估算结果会按人力资源层次的不同，进行分类：系统设计师需要多少人月、一般编程人员需要多少人月等。或者，在需求分析阶段，需要多少人月、在设计阶段需要多少人月等。这样，可以比较容易地获得人力资源需求和成本估算结果。

在进度估算时，注重的是任务单元的时间长度、任务之间的时间先后关系和联系关系。在做计划进度安排时，重点考虑单位任务的时间历时，不考虑由谁和完成什么样的单位任务（当然不会完全不考虑，例如，需要调整或协调的时候）

软件项目工作量的估算方法

软件工作量的估算，可以采取不同的操作方法，以下是几种常用的方法：

（1）自顶向下估算法：首先对整个系统进行总工作量估算，再考虑子系统，把总工作量逐步分解为各组成部分的工作量，并考虑到开发该软件所需要的资源、人员、质量保证、系统集成安装等的工作量。

优点是估算的工作量小，速度快。缺点是对项目中的特殊困难估计不足，估算出来的工作量盲目性大，有时会遗漏被开发软件的某些部分。

（2）自底向上估算法：先对开发各个子系统或每个模块的工作量进行估算，再逐步相加。这是一种常见的估算方法。

优点是估算各个部分的准确性高。缺点是缺少各项子任务之间相互联系所需要的工作量，还缺少许多与软件开发有关的系统级工作量（配置管理、质量管理、项目管理）。所以往往估算值偏低，必须用其它方法进行检验和校正。

（3）相似比较估算法：把开发项目的工作分割到一定的程度，和过去的工作进行比较，对其中相同的或相近的部分，用已有的数据进行估算，对不同的部分再用其它的方法估算。

优点是可以提高估算的准确程度，缺点是不容易明确“类似”的界限。

（4）Delphi 估算法：请多位项目经理、系统分析员或其他专家，利用专家的经验来评估软件的开发成本。

优点是可以摈弃无根据的估算值，缺点是一些组员可能会受权威或政治因素的影响。

软件项目工作量的计算

工作量评估评估是对项目有关工作的、以人时、人月、人年为单位进行的计算，它是成本和预算的依据。工作量估计的结果，一般是多少个人的多少工作时间。

项目工作量估算的来源，是项目任务的WBS分解。因此，在获得工作量的度量值之前，首先是对任务的分解。当然，在项目的不同阶段，对任务的认识和理解的程度不同，所能分解的“粒度”不同，获得的工作量估算的准确性也会不同，这是不用再说明的。

在前面，我们已经介绍了WBS分解方法。有了任务分解（暂时不考虑集成的相关工作），就可以对分解后获得的任务单元的性质，进行定义。

例如：是概要设计、架构设计，还是接口设计等。定义的目的是分配不同级别的人力资源，并估计在这样（或不是这样）的人力资源条件下的任务历时时间。

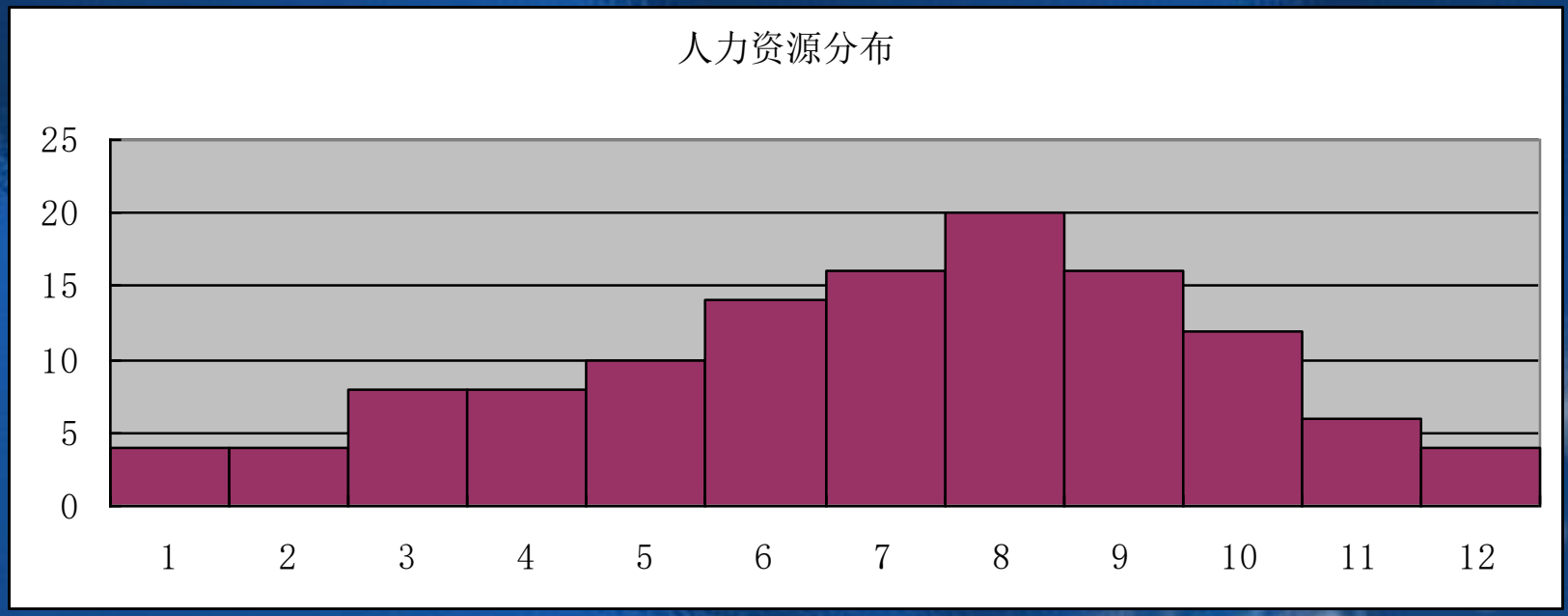
软件项目工作量估算的案例

任务名称	人力资源名称	工作量（人月）	资源数量（人）	工期（月）
项目管理	项目经理	10	1	10
系统需求分析	系统设计师	4	2	2
系统概要设计	系统设计师	2	2	1
系统详细设计	系统设计师	6	3	2
系统架构设计	系统架构师	1	1	1
核心模块编码	高级程序员	12	4	3
业务模块编码	高级程序员	15	5	3
一般模块编码	初级程序员	32	8	4
单元测试	测试工程师	16	2	8
集成测试	高级测试师	4	2	2
文档编写	文档编辑	20	2	10
合计		122		

经过测算，开发电信综合营业系统的工作量估计是122个人月。

软件项目工作量估算的案例

下图是该项目的人力资源分布（按月）曲线。根据公司的人力资源情况，可以分析这个分布是否可以得到满足。



软件项目工作量的其他影响因素

在做软件项目工作量的估计时，应考虑到以下几个其他因素：
项目复杂度、人为因素和工程因素、以外因素：

- 复杂度包括：问题领域、算法复杂性、程序设计语言、软件复用量、可靠性等性能要求、系统平台复杂性、资源的限制等。
- 人为因素包括：开发人员的能力、经验、稳定性，开发的组织管理能力、用户的配合等。
- 工程因素:包括开发技术的难度、进度的紧迫度、项目团队的凝聚力、多地点开发等。

意外事件也是一个影响因素，意外事件可以有以下几种：

- 工程师的事假、产假和病假天数；
- 工程师参加与项目无关的培训的天数；
- 工程师需要为其它项目做软件质量保证工作的时间；
- 工程师需要做非项目有关的工作，例如参加员工会议，软件工程过程小组的工作等；

软件项目工作量的其他构成因素

在我们进行软件项目工作量估算的时候，我们可能会比较集中地考虑需求分析、设计、编码、测试等的工作量，但往往会忽略以下的一些工作量：

- 用于各模块、子系统、软件系统与硬件/网络系统等之间集成的测试、调试等的工作量；
- 用于编写用户文档和设计文档的工作量；
- 用于需求管理、配置管理、质量管理、风险管理等支持过程的工作量；
- 用于项目管理的工作量。

有关这些部分工作量的度量，目前还没有看到比较系统地、适用的度量和估算方法的介绍。

因为作为项目管理，本身就是全过程、随时性和零碎的，因此，对于它们的工作度量，确实更为困难。但是，这部分的工作又是非常关键和不可或缺，工作量也是比较大的。因此，如何规范地测量，是项目管理的一个新课题。

软件项目工作量的其他估算

软件项目的成本估算，是一个非常重大的课题。

在CMM2软件项目计划管理活动的第7项，是对关键计算机资源的评估。所有项目必须进行关键计算机资源评估，并且必须将填写完成的关键计算机资源评估调查表存入项目文件夹中。对计算机资源的评估，可以借助一些软件工具。

在CMM2的计划管理活动中，第9个活动是对软件项目的风险，进行识别和评估。在制定软件项目计划的时候，对所有项目都必须进行风险评估，并且必须将填写完成的风险评估调查表存入项目文件夹中。为了便于对软件风险管理能有一个系统的认识和理解，我们把软件风险的识别、评估，统一放在《软件项目的风险管理》中集中讨论。

软件项目的进度估算

在前面的介绍中，我们已经讨论了软件项目工作量估算中的时间和软件项目进度估算中时间概念的不同。在进行项目的进度估算时，我们要做的工作是：

- ✓ 估算要完成的单位任务（系统、子系统、模块、单元）的时间长度；
- ✓ 确定标志阶段任务完成的里程碑事件；
- ✓ 确定重要阶段点的评审日期。

这些工作的结果，一般表现在一张项目进度表里。

软件项目的项目进度表

软件项目进度表是组织规范项目组进度计划管理最基本的工具，除非在客户需求中有特殊指明，否则，组织应要求组织内的所有项目，都应该采用公司所建议使用的工具和项目进度表模板，来制定进度表。

作为一个大型的综合性项目，计划可能涉及到许多方面，例如：供货计划、硬件和网络安装调试计划、软件开发计划、项目管理计划等。它们时间又是相互配合、相互衔接的。

我们的计划模板，包括以下内容：

（1） 项目实施总计划
（2） 项目管理计划
（3） 交货计划
（4） 软件技术实施计划
（5） 硬件技术实施计划

下表是一个硬件实施进度计划的例子：

项目名称　　编号

项目技术实施（硬件）计划

制订日期　　版本：1.0

序号	工作内容	里程碑描述	开始	完成	责任	配合	备注
1	项目正式启动						
1.1	提交系统解决方案						
1.2	系统解决方案修改、确认						
1.3	详细设计（路由/IP地址规划）						
2	环境平台						
2.1	提交《机房环境安装条件表》						
2.2	准备机房安装环境						
3	系统安装						
3.1	交货验收						
3.2	设备安装调试						
3.3	系统调试						
3.4	提交测试大纲						
3.5	系统测试						

4	系统割接						
4.1	提交割接计划						
4.2	割接计划确认、修改						
4.3	割接						
5	系统验收						
5.1	提交竣工文档						
5.2	系统验收						
5.3	签署系统验收报告						
6	培训						
6.1	制定培训计划						
6.2	编制培训材料						
6.3	使用人员培训						
6.4	维护人员培训						

审批　　拟制

对每一项任务估算时间进度

模板实际上已经为我们分解和定义了项目任务。按照模板，项目进度的估算可以按以下步骤进行：

（1）按分解的任务确定责任人

在计划进度表中，模板已经分解了项目任务，并为估算过程中的每一项任务，标识出了一条记录。每一个任务记录，需要我们填写的，就是任务进度估算值：什么时间开始，什么时间结束，里程碑标志是什么，责任人、配合人是谁等。

任务进度表中，必须标明任务责任人或小组，根据这个任务表，任务将会分配到一个个人或一个小组。比较粗的计划可能以小组为单位，或不确定的个人。但详细计划将明确到具体的个人。因为，不同的人，担负相同的任务，其时间历时，有可能会差距巨大。人员的不确定，使计划的真实和准确性，没有保证。

为了让项目组成员各负其责，以文字形式确定他们在项目组里分担的责任是很重要的。比较有效的方法是绘制责任矩阵表。在项目开始时就要恰当地搭配好人员、技术及工作任务。随着项目的进展，有可能必须把已分的工作再细分或进行新的调整，为此，项目经理应该清楚地了解项目组成员各自掌握的技术。

对每一项任务估算时间进度

首先，项目经理可以打开项目组人员编制表，然后，为项目组每一个人员打分，其方法是按照对专业领域的熟悉程度打分。例如，将专业领域分为五个：系统分析员、程序员、测试工程师、硬件工程师、数据库管理员，并将最高分定为5分。随后根据每个成员对上述专业领域的熟悉程度打分，熟悉程度越高，打分越高。如此一来，就可以对项目组人员及技术状况一目了然，并据此分配工作。

打分表绘制完成之后，项目经理就可以根据项目的实际需求来绘制责任矩阵。该矩阵是项目经理与项目组成员之间的工作合同文件，也是进行人员任用或让其承诺某项工作的重要手段。

（2）估算任务单元的可支配时间

根据具体任务承担人的能力、资源、条件等，通过估算，输入完成每一项已分配任务所需时间的估算值，也是该任务责任人的可支配时间。

对于具体任务单元的历时估算，没有统一标准的计算方法。也不可能给出统一的计算方法。例如：安装和调试一台网络交换机的时间，从交换机上架开始，到网络调通、测试完成。不同厂家、不同型号、不同设备配置、不同用户环境和线路环境、不同网络结构、不同测试要求等，给出一个标准的完成时间，是不可能的。

因此，在没有其他制约因素（可以有项目组自主决定实施时间）的前提下，历时时间的估算，采用任务责任人提出，技术和项目经理根据经验调整、确定的方法，是最常见的估算方法。

对每一项任务估算时间进度

（3）考虑任务估算的制约和影响因素

实际上，没有实施时间制约是不可能的。

不单是时间制约，还有其他最常见的制约有：

用户对完成任务的时间要求：这个要求通常是倒计时的，即限定什么时间内必须完成。

项目内部配合上对时间的要求：任务是相互衔接的，因此，上一任务的完成可能是下一任务开始的前提条件，因此，项目整体对具体一个任务的完成时间是有要求的。

资源的限制：这是非常容易理解的，因为资源永远是有限的，因此，资源必须是分享。以上谈到的用户要求和项目组配合的要求，都是整体项目任务时间资源在项目组内部任务之间的分配和利用。硬件资源（设备的轮流公用）、人力资源（工程师的调配）、支持和管理资源（测试计划的安排）等，都是资源限制的例子。

质量标准和规范的限制：文档种类、规范要求等，要求必须达到的标准等。

应付变动和不确定因素：时间估算必须留有应对意外变化的余地，因此，应对不确定因素的考虑，也是估算历时时间时，必须考虑的因素。

这些制约，有些，就直接反映到项目任务的时间进度上。

（4）对任务单元，进行计划安排

在以下的小节中，我们将详细讨论计划安排，包括：并行、交叉、关键路径等计划编排技术。