第13章 软件项目管理与计划

本章内容：

第13章 软件项目管理与计划

据统计，就是在软件业发达的美国，每年有差不多50%的软件项目在开发完成交付使用后遇到巨大的困难。包括程序出错、成本超支和没能完成用户的要求等。所有的这一切，都可以归结到软件项目管理太弱的原因。

13.1 软件项目管理概述

13.1.1 软件管理的对象

在软件项目管理中，重要的是人、问题和过程三者。其中人是最重要的管理对象，因为软件工程是人的智力密集的劳动。

从15世纪60年代起，培养有创造力、技术水平高的软件人员就是一个重要的话题。以至于在软件工程的项目管理中，有一个专用于衡量管理软件人员水平的模型-人员管理成熟度模型（PM-CMM）。目的是“通过吸引、培养、鼓励和留住改善其软件开发能力所需要的人才来增强软件组织承担日益复杂的应用程序开发的能力”。在现实中，在人员管理成熟度较高的组织中，更有可能成功实现软件工程开发。

组成一个软件工程的开发项目的人员有以下几类：

（1）高级管理者：负责确定软件的问题。

（2）项目技术管理者：管理软件开发人员。

（3）开发人员：软件开发的专门的技术人员。

（4）客户：负责说明软件需求的人员。

（5）最终用户：最终使用软件的人员。

作为项目负责人的目标之一就是使得上面的几类人可以高效地合作，发挥每个人的能力。

软件开发人员碰到的一个两难问题是：一开始就需要制定计划，需要定量的估算成本，但是却没有可靠的信息使用。对软件项目的详细需求分析可以得出基本上可靠和足够的信息，但是在时间上来说太晚，制定一个计划仍然是必需的。

问题就是指软件工程的目的和范围。根据该目的和范围，选择可能的方案，定义技术规范。没有这些信息，就不能进行合理的成本估算，也不能进行有效的风险评估，对项目也就不能进行适当的划分，无法安排开发进度。定义软件的问题是在软件工程的第一阶段开始的，直到软件的需求分析完成。

一旦解决了项目的目的和范围的问题，就可以得到一个合理的解决方案，根据该方案可以对各种资源进行估算。

对过程来说，就是制定一个软件开发的综合计划。对于每一个任务集合（都是由任务、里程碑、交付物以及质量保证点组成）都可以适应软件项目的特点。现在已经有了多个过程模型（原型模型、瀑布模型、螺旋模型、增量模型等），对管理者来说，困难的是如何根据具体的情况，选择一个合适的过程模型。

13.1.2 软件开发中的资源

软件项目计划的第二个任务是对完成该软件项目所需的资源进行估算。如图15-1所示把软件开发所需的资源画成一个金字塔，在塔的底部必须有现成的用以支持软件开发的工具——硬件工具及软件工具，在塔的高层是最主要的资源——人。

通常，对每一种资源，应说明以下四个特性：资源的描述、资源的有效性说明、资源在何时开始需要、使用资源的持续时间。最后两个特性统称为时间窗口。对每一个特定的时间窗口，在开始使用它之前就应说明它的有效性。

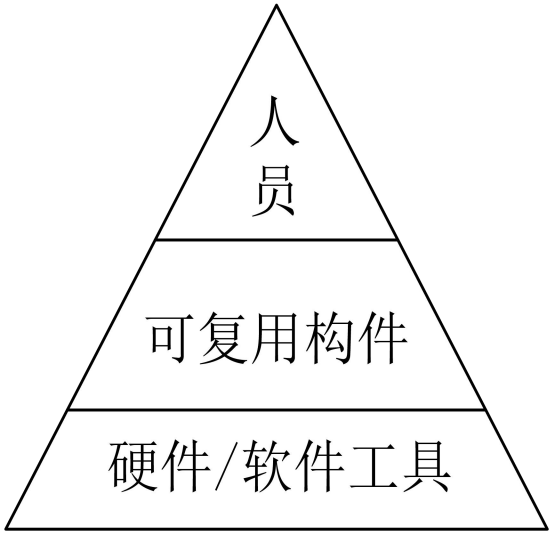

图13-1 软件开发所需的资源

1. 人力资源是最重要的资源

在安排开发活动时必须考虑人员的技术水平、专业、人数以及在开发过程各阶段中对各种人员的需要。

对一些规模较大的项目，在整个软件生存期中，各种人员的参与情况是不一样的。如图15-2所示画出了各类不同的人员随开发工作的进展在软件工程各个阶段的参与情况的典型曲线。

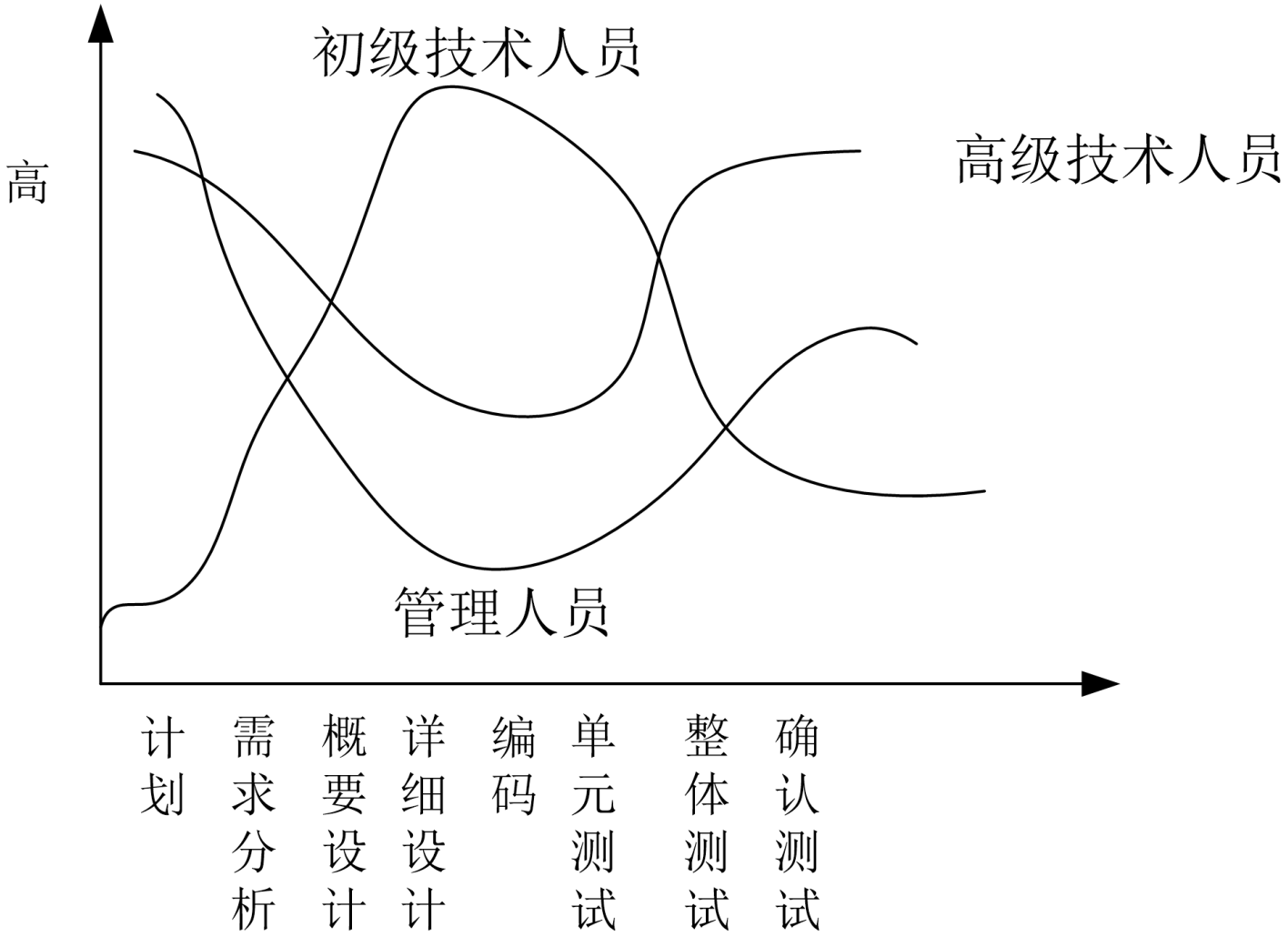

图13-2 管理人员与技术人员的参与情况

2. 硬件/软件资源

硬件是作为软件开发项目的一种工具而投入的。在软件项目计划期间，考虑三种硬件资源：

（1）宿主机（Host machine）——软件开发时使用的计算机及外围设备。

（2）目标机（Target machine）——运行已开发成功软件的计算机及外围设备。

（3）其他硬件设备——专用软件开发时需要的特殊硬件资源。

宿主机连同必要的软件工具构成软件开发系统。

软件资源包括用于开发的运行平台、各种CASE工具可以帮助分析和设计软件、开发程序所有的编程语言等。

3. 可复用构件资源

为了促成软件的复用，以提高软件的生产率和软件产品的质量，可建立可复用的软件部件库。根据需要，对软件部件稍做加工，就可以构成一些大的软件包。这要求这些软件部件应加以编目，以利于引用，并进行标准化和确认，以利于应用和集成。

遗憾的是，在计划阶段，人们往往忽视软件资源。直到软件工程过程的开发阶段，软件资源成为一个重大问题时才引起人们的重视。最好是尽早确定软件资源的需求，这样可以对各种候选方案进行技术评价，并及时地获取这些软件。

13.1.3 分解技术

当一个待解决的问题过于复杂时，可以把它进一步分解，直到分解后的子问题变得容易解决为止。然后，分别解决每一个子问题，并将这些子问题的解答综合起来，从而得到原问题的解答。

软件项目估算是一种解决问题的形式，在多数情况下，要解决的问题（对于软件项目来说，就是成本和工作量的估算）非常复杂，想一次性整体解决比较困难。因此，对问题进行分解，把其分解成一组较小的接近于最终解决的可控的子问题，再定义它们的特性。

分解技术可以分为问题分解和过程分解。

1.问题分解，也叫划分。它关心两个方面的问题：软件的功能和交付使用的过程。比如对一个字处理软件项目。客户要求可以使用语音输入、有页面布局和自动编辑功能等。对于这些问题，可以进一步分解如下：

语音输入是否需要培训？如果要，则可以分解该功能。对页面布局的要求到底是怎么样的？而自动编辑功能进一步应该具有：

（1）拼写检查。

（2）句子的语法检查。

（3）大型文档的参考书目关联检查。

这样就可以将问题逐步分解了，也就可以提高项目的估算精度。

2.过程分解。不同的项目开发模型有不同的适应情况，在选择一个软件开发模型时，可以相对灵活。比如要开发的项目的某部分同以前曾经开发的项目相似时，可以选择线性模型。因此，可以将一个项目的过程进行分解，找到它最适合的开发过程模型。

13.2 项目管理过程

软件项目管理的对象是软件工程项目。它所涉及的范围覆盖了整个软件工程过程。

为使软件项目开发获得成功，必须对软件开发项目的工作范围、可能遇到的风险、需要的资源（人、软/硬件）、要实现的任务、经历的里程碑、花费的工作量（成本），以及进度的安排等进行非常规范、科学的管理。这种管理开始于技术工作开始之前，在软件从概念到实现的过程中持续运行，最后终止于软件工程过程结束。包括以下几个方面：

1. 启动一个软件项目

通常，软件人员和用户是在软件工程的可行性分析阶段确定项目的目标和范围。目标标明了软件项目的目的。范围标明了软件要实现的基本功能，并寻求解决的方案。虽然涉及方案细节不多，但有了方案，管理人员和技术人员就能够据此选择一种“好的”方法，确定合理、精确的成本估算，实际可行的任务分解以及可管理的进度安排。

2. 成本估算

在软件项目管理过程中一个关键的活动是制定项目计划。在做计划时，必须就需要的人力、项目持续时间、成本作出估算。这种估算大多是参考以前的花费，凭经验作出的。但是现在已经有了一些软件开发中模块数量、软件的复杂度等为参数的成本估算模型，可以帮助我们对新开发的软件作出成本估算。

3. 风险分析

每当新建一个计算机程序时，总是存在某些不确定性。这些不确定性就是软件开发中的风险。风险分析对于软件项目管理的成功与否是决定性的，然而现在还是有许多项目不考虑风险就着手进行。Tom Gilb在他的有关软件工程管理的书中写道：“如果谁不主动地攻击（项目和技术）风险，它们就会主动地攻击谁”。风险分析实际上就是贯穿在软件工程过程中的一系列风险管理步骤，其中包括风险识别、风险估计、风险管理策略、风险解决和风险监督，它能让人们去主动“攻击”风险。

4. 进度安排

要想软件能够按时完成，则进度安排是必不可少的。进度安排的思想，首先识别一组项目任务，再建立任务之间的相互关联，然后估算各个任务的工作量，分配人力和其他资源，制定进度时序。比如要考虑进度如何计划？工作怎样就位？如何识别定义好的任务？管理人员对结束时间如何掌握，如何识别和监控关键路径以确保结束？以及如何建立分隔任务的里程碑。

5. 追踪和控制

建立了开发进度安排，并不能高枕无忧，项目管理人员负责对项目开发的追踪和控制活动。如果任务实际完成日期滞后于进度安排，则管理员可以使用一种自动的项目进度安排工具来确定在项目的中间里程碑上进度误期所造成的影响。此外，还可对资源重新定向，对任务重新安排，或者（作为最坏的结果）可以修改交付日期以调整已经暴露的问题。用这种方式可以较好地控制软件的开发行为。

13.3 软件开发成本估算

软件开发和任何的商业活动一样，都是希望通过投资得到更大的回报，因此对成本的估算就是非常重要的，甚至关系到项目的成败。

软件开发成本主要是指软件开发过程中所花费的工作量及相应的代价，不包括原材料和能源的消耗，主要是人的劳动的消耗。它的开发成本是以一次性开发过程所花费的代价来计算的。因此软件开发成本的估算，应是从软件计划、需求分析、设计、编码、单元测试、组装测试到确认测试，整个软件开发全过程所花费的人工代价作为依据的。

13.3.1 软件开发成本估算方法

由于软件开发的特殊性，开发成本的估算不是一件简单的事，往往不到最后时刻，是很难得到准确的成本的。对软件成本的估算，主要靠分解和类推的手段进行。基本估算方法分为三类：

1. 自顶向下的估算方法

这种方法的想法是从项目的整体出发，进行类推。即估算人员根据以前已完成项目所耗费的总成本（或总工作量），推算将要开发的软件的总成本（即总工作量），然后按比例将它分配到各开发任务中去，再检验它是否能满足要求。Boehm给出一个参考例子，如表15-1所示。

软件库存情况更新　　开发者W.Ward　　日期			
阶段	项目任务	工作量分布（1/53）	小计（1/53）
计划和需求划内	软件需求定义	5	6
	开发计划	1	
产品设计	产品设计	6	10
	初步的用户手册	3	
	测试计划	1	
详细设计	详细PDL描述	4	12
	数据定义	4	
	测试数据及过程设计	2	
	正式的用户手册	2	
编码与单元测试	编码	6	16
	单元测试结果	10	
组装与联合测试	按实际情况编写文档	4	9
	组装与测试	5	
总计			53

表13-1 软件开发各阶段工作量的分配

这种方法的优点是估算工作量小，速度快。缺点是对项目中的特殊困难估计不足，估算出来的成本盲目性大。

2. 自底向上的估算法

这种方法的想法是把待开发的软件细分，直到每一个子任务都已经明确所需要的开发工作量，然后把它们加起来，得到软件开发的总工作量。

这是一种常见的估算方法。它的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，必须用其他方法进行检验和校正。

3. 差别估算法

这种方法综合了上述两种方法的优点，其想法是把待开发的软件项目与过去已完成的软件项目进行类比，从其开发的各个子任务中区分出类似的部分和不同的部分。类似的部分按实际量进行计算，不同的部分则采用相应的方法进行估算。这种方法的优点是可以提高估算的准确度，缺点是不容易明确“类似”的界限。

13.3.2 软件开发成本估算的经验模型

开发成本估算模型通常采用经验公式来预测软件项目计划所需要的成本、工作量和进度。这些经验数据都是从有限的一些项目样本中得到的。还没有一种估算模型能够适用于所有的软件类型和开发环境，每一种模型得到的数据都是不准确的。因此从这些模型中得到的结果必须慎重使用。

1. IBM模型

1977年，Walston和Felix总结了IBM联合系统分部（FSD）负责的60个项目的数据。其中各项目的源代码行数从400行到467000行，开发工作量从12PM到11758PM，共使用29种不同语言和66种计算机。利用最小二乘法拟合，得到如下估算公式：

E = 5.2×L0.19　　D = 4.1×L0.36 = 17.47×E0.35

S = 0.54×E0.6　　DOC = 49×L1.01

其中，L是源代码行数（以KLOC计），E是工作量（以PM计），D是项目持续时间（以月计），S是人员需要量（以人计），DOC是文档数量（以页计）。因此估算出了源代码的数量，就可以对工作量、文档数量等进行估算了。

IBM模型是一个静态单变量模型，它利用已估算的特性，例如源代码行数，来估算各种资源的需要量。IBM模型是一个静态单变量模型，但不是一个通用公式。在应用中有时要根据具体实际情况，对公式中的参数进行修改。

2. Putnam模型

这是1978年Putnam提出的模型，是一种动态多变量模型。该模型的基础是假定在软件开发的整个生存期中工作量有特定的分布。它把项目的资源需求当做时间的函数。根据对一些大型项目的统计分析，软件开发工作量分布可用如图13-3所示的曲线表示。

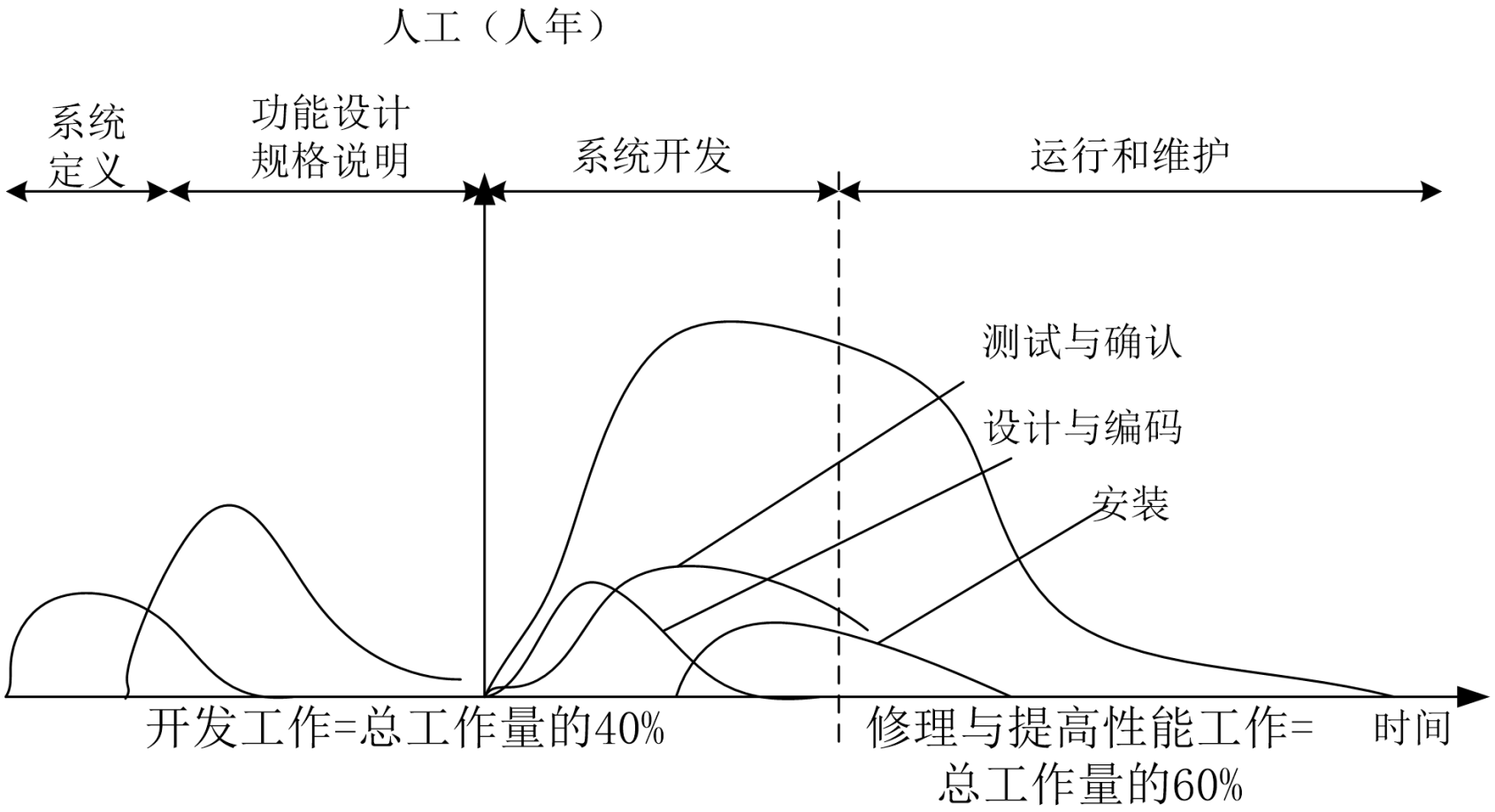

图13-3 大型项目的工作量分布情况

如图13-3所示中的曲线被称为Rayleigh-Norden曲线。利用该曲线得到如下的经验公式：

$$L=Ck\cdot K^{1/3}\cdot td^{4/3}$$

其中，td是开发持续时间（以年计），K是软件开发与维护在内的整个生存期所花费的工作量（以人年计），L是源代码行数（以LOC计），Ck是技术状态常数，它反映出“妨碍程序员进展的限制”，并因开发环境而异。

3. COCOMO模型（Constructive Cost Model）

Barry Boehm提出的一种软件估算模型的层次体系，称为结构型成本估算模型。是一种比较精确、易于使用的综合成本估算方法。

该模型分为三个层次：

基本的COCOMO模型：只是将工作量（成本）作为程序规模的函数进行计算。

中级的COCOMO模型：除了工作量以外，还将对产品、硬件、人员及项目属性的主观评价作为“成本驱动因子”加入估算模型中。

高级的COCOMO模型：除了中级模型的因素外，还加入了成本驱动因子对软件开发的每一个过程的影响的评估。

对于项目属性来说，COCOMO规定了三种项目属性：

（1）组织型（Organic）：较小、较简单的软件项目。项目组人员经验丰富，对软件的使用环境很熟悉，受硬件的约束较少，程序的规模不是很大（＜5万行）。

（2）嵌入型（Embadded）：此种软件要求在紧密联系的硬件、软件和操作的限制条件下运行的软件。比如航天用控制系统属此种类型。

（3）半独立型（Semidetached）：对此种软件的要求介于上述两种软件之间，但软件规模和复杂性都属于中等以上，最大可达30万行。例如，大多数事务处理系统属此种类型。

基本的COCOMO模型的估计方式由如下公式确定：

E = a*KLOCb

D = c*Ed

E是以人月为单位的工作量，D是以月表示的开发时间，KLOC是项目的代码行（以千行为单位），a、b、c、d是系数，如表15-2所示。

软件项目	a	b	c	d
组织型	2.4	1.05	2.5	0.38
半独立型	3.0	1.12	2.5	0.35
嵌入型	3.6	1.15	2.5	0.32

表13-2 基本COCOMO模型系数表

在软件开发中，人们提出了很多的经验模型。但所有的经验模型都是从已有的软件项目中进行回归分析得到的，都带有极大的经验的成分。对同一个项目，使用不同的经验模型，得到的软件开发的成本不同。

对于预测的结果，预测成本和实际成本相差不到15%，开发时间的估计相差不到30%以内，就足以给软件工程提供很大的帮助了。

13.4 风险分析

在软件工程领域，也应该考虑风险。Robert Charette关于风险的定义是：“首先，风险关系到未来发生的事情。今天收获的是以前的活动播下的种子。问题是，能否通过改变今天的活动为自身的明天创造一个完全不同的充满希望的美好前景。其次，风险会发生变化，就像爱好、意见、动作或地点会变化一样……。第三，风险导致选择，而选择本身将带来不确定性。

因此，风险就像死亡那样，是一个其生命很少确定性的东西。”当在软件工程的环境中考虑时，风险的含义一是关心未来，风险是否会导致软件项目失败；二是关心变化，在用户需求、开发技术、目标机器，以及所有其他与项目有关的实体中会发生的变化？三是必须解决选择问题：应当采用什么方法和工具，应当配备多少人力，在质量上强调到什么程度才满足要求？风险分析实际上是4个不同的活动：风险识别、风险估算、风险评价和风险驾驭。

13.4.1 风险识别

可用不同的方法对风险进行分类。从宏观上来看，可将风险分为项目风险、技术风险和商业风险。项目风险包括潜在的预算、进度、个人（包括人员和组织）、资源用户和需求方面的问题，以及它们对软件项目的影响。技术风险包括潜在的设计、实现、接口、检验和维护方面的问题。此外，规格说明的多义性、技术上的不确定性、技术陈旧、最新技术（不成熟）也是风险因素。之所以存在技术风险是由于软件开发中总会意想不到的技术问题出现。

商业风险主要有以下几种：

（1）建立的软件虽然很优秀但不是真正所想要的（市场风险）。

（2）建立的软件不适用整个软件产品战略。

（3）销售部门不清楚如何推销这种软件。

（4）失去上级管理部门的支持。

（5）失去预算或人员的承诺（预算风险）。

（6）最终用户的水平。

13.4.2 风险估算

风险估算，又叫风险预测。使用两种方法来估计每一种风险发生的可能性和概率。通常，项目计划人员与管理人员、技术人员一起，进行4种风险估算活动：

（1）建立一个尺度或标准来表示一个风险的可能性。

（2）描述风险的结果。

（3）估计风险对项目和产品的影响。

（4）确定风险估计的正确性。

可以通过检查风险表来度量各种风险。尺度可以用布尔值、定性的、或定量的方式定义。一种比较好的方法是使用定量的概率尺度，它具有下列的值：极罕见的、罕见的、普通的、可能的、极可能的。还可以将多个开发人员对某个项目的风险估计进行平均后作为评估结果。

最后，根据已掌握的风险对项目的影响，可以给风险加权，并把它们安排到一个优先队列中。造成影响的因素有三种：风险的表现、风险的范围和风险的时间。风险的表现指出在风险出现时可能的问题。风险的范围则组合了风险的严重性（即它严重到什么程度）与其总的分布（即对项目的影响有多大，对用户的损害又有多大）。风险的时间则考虑风险的影响什么时候开始，要影响多长时间。

如图13-4所示，风险影响和出现概率对驾驭参与有不同的影响。一个具有较高影响权值但出现概率极低的风险因素应当不占用很多有效管理时间。然而，具有中等到高概率的高影响的风险和具有高概率的低影响的风险，就必须进行风险的分析。

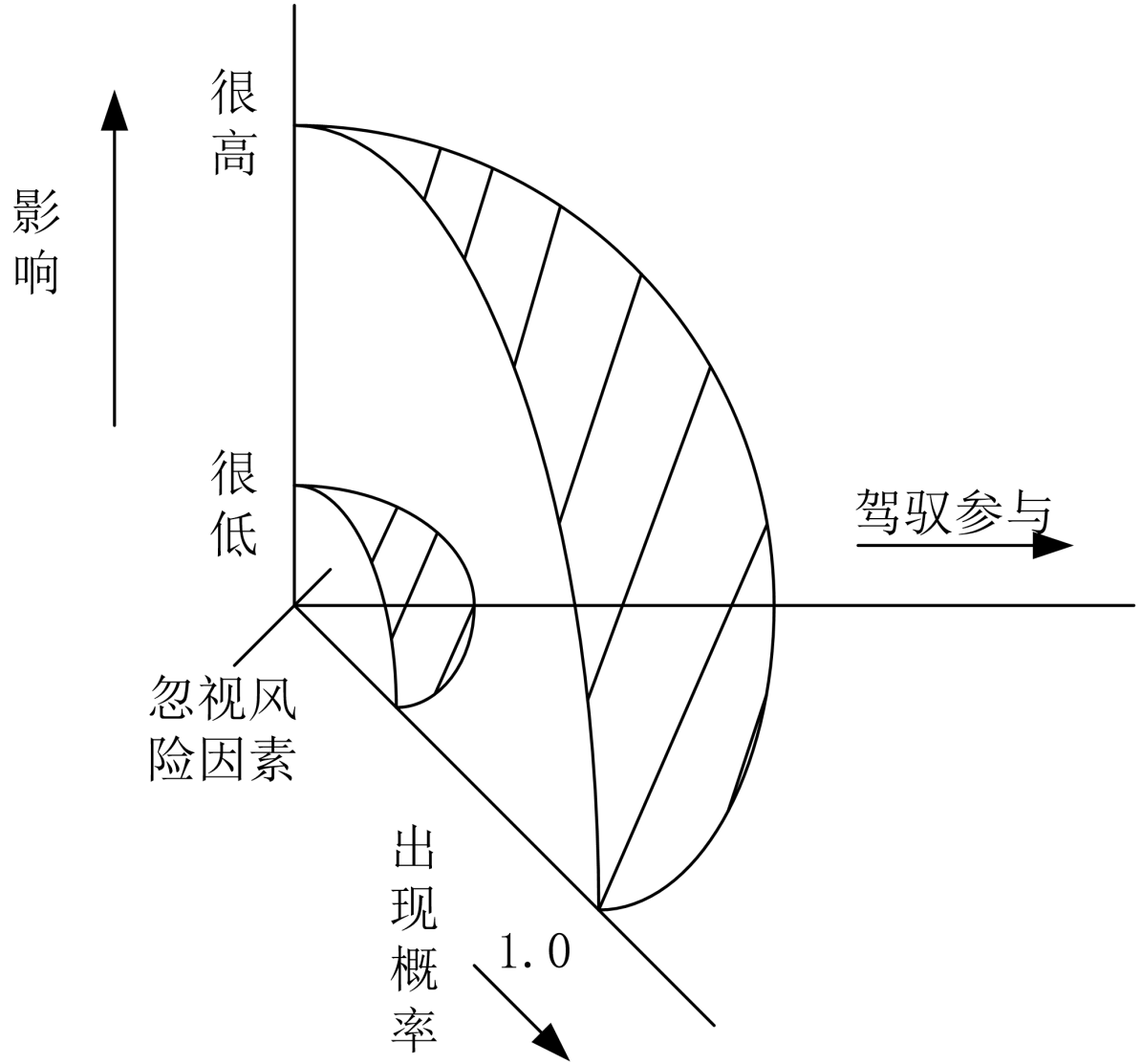

图13-4 风险与驾驭参与

13.4.3 风险评价

在风险分析过程中进行风险评价的时候，应当建立一个三元组：

[ri，li，xi]

其中，ri是风险，li是风险出现的可能性（概率），而xi是风险的影响。在做风险评价时，应当进一步检验在风险估计时所得到的估计的准确性，尝试对已暴露的风险进行优先排队，并着手考虑控制和（或）消除可能出现风险的方法。

一个对风险评价很有用的技术就是定义风险参照水准。对于大多数软件项目来说，成本、进度和性能就是三种典型的风险参照水准。就是说，对于成本超支、进度延期、性能降低（或它们的某种组合），有一个表明导致项目终止的水准。如果风险的某种组合造成一些问题，从而超出了一个或多个参照水准，就要终止工作。在做软件风险分析的环境中，一个风险参照水准就有一个单独的点，叫做参照点或崩溃点。在这个点上，就要决定看是继续执行项目工作，还是终止它们（出的问题太大）。

图13-5表示了这种情况。如果因为风险的一个组合引出造成项目成本和进度超出的问题，将有一个水准（在图中用曲线表示），当超出时，将导致项目终止（图中封闭区域）。

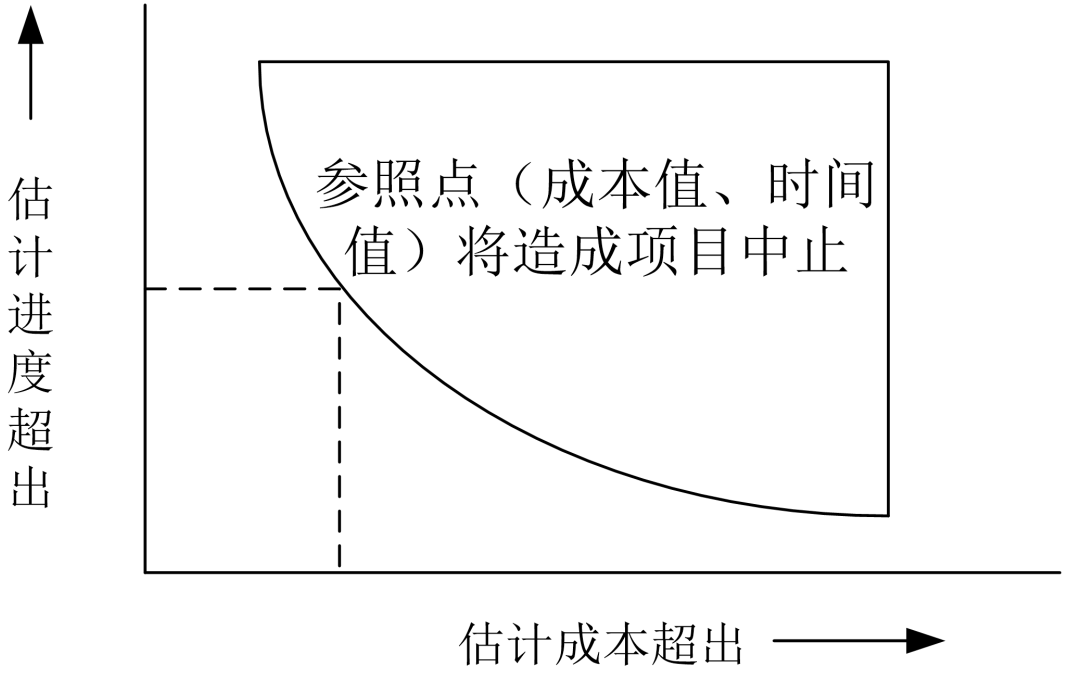

图13-5 风险参照水准

在多数情况中，参照点不是一条平滑的曲线，而是一个区域，这个区域可能是易变动的区域，在这些区域内想要作出基于参照值组合的管理判断往往是不准确的。因此，在做风险评价时，按以下步骤执行：

（1）为项目定义风险参照水准。

（2）尝试找出在每个[ri，li，xi]和每个参照水准之间的关系。

（3）预测参照点，定义一个终止区域，用一条曲线或一些易变动区域来界定。

（4）努力预测复合的风险组合将如何形成一个参照水准。

支持这些步骤的更详细的数学讨论已超出本书的范围，有需要的读者请参看有关书籍。

13.4.4 风险驾驭和监控

所有的风险分析活动都只有一个目的——建立处理风险的策略。风险驾驭是指利用某些技术，如原型化、软件自动化、软件心理学、可靠性工程学以及某些项目管理方法等设法避开或转移风险。与每一风险相关的三元组（风险描述，风险可能性，风险影响）是建立风险驾驭（风险消除）步骤的基础。

例如，假如人员的频繁流动是一项风险ri，基于过去的历史和管理经验，频繁流动可能性的估算值li为0.70（70%相当高），而影响xi的估计值是：项目开发时间增加15%，总成本增加12%，给出了这些数据之后，建议可使用以下风险驾驭步骤：

（1）与现在在职的人员协商，确定人员流动的原因（如工作条件差、收入低、人才市场竞争等）。

（2）在项目开始之前，把缓解这些原因（避开风险）的工作列入已拟定的驾驭计划中。

（3）当项目启动时，做好人员流动会出现的准备。采取一些办法以确保人员一旦离开时项目仍能继续（削弱风险）。

（4）建立项目组，以使大家都了解有关开发活动的信息。

（5）制定文档标准，并建立一种机制以保证文档能够及时产生。

（6）对所有工作组织细致的评审（以使更多的人能够按计划进度来完成自己的工作）。

（7）对每一个关键性的技术人员，要培养后备人员。

这些风险驾驭步骤带来了额外的项目成本。例如，花费时间来培养关键技术人员的后备需要花钱。因此要对风险驾驭带来的成本/效益进行分析。

对于一个大型的软件项目，可能识别30~40项风险。如果每一项风险有3～7个风险驾驭步骤，那么风险驾驭也可能成为一个项目。在风险管理中也可以应用Pareto 80/15规则。

经验表明，所有项目风险的80%（即是使项目失败的潜在因素的80%）能够通过15%的已识别风险来说明。

由于这个原因，对某些不属于关键的15%（具有最高项目优先级的风险）的风险可进行识别、估算、评价，但可以不写进风险驾驭计划中。

图13-6表示风险驾驭步骤要写进风险驾驭与监控计划RMMP（Risk Management and Monitoring Plan）。RMMP记叙了风险分析的全部工作。

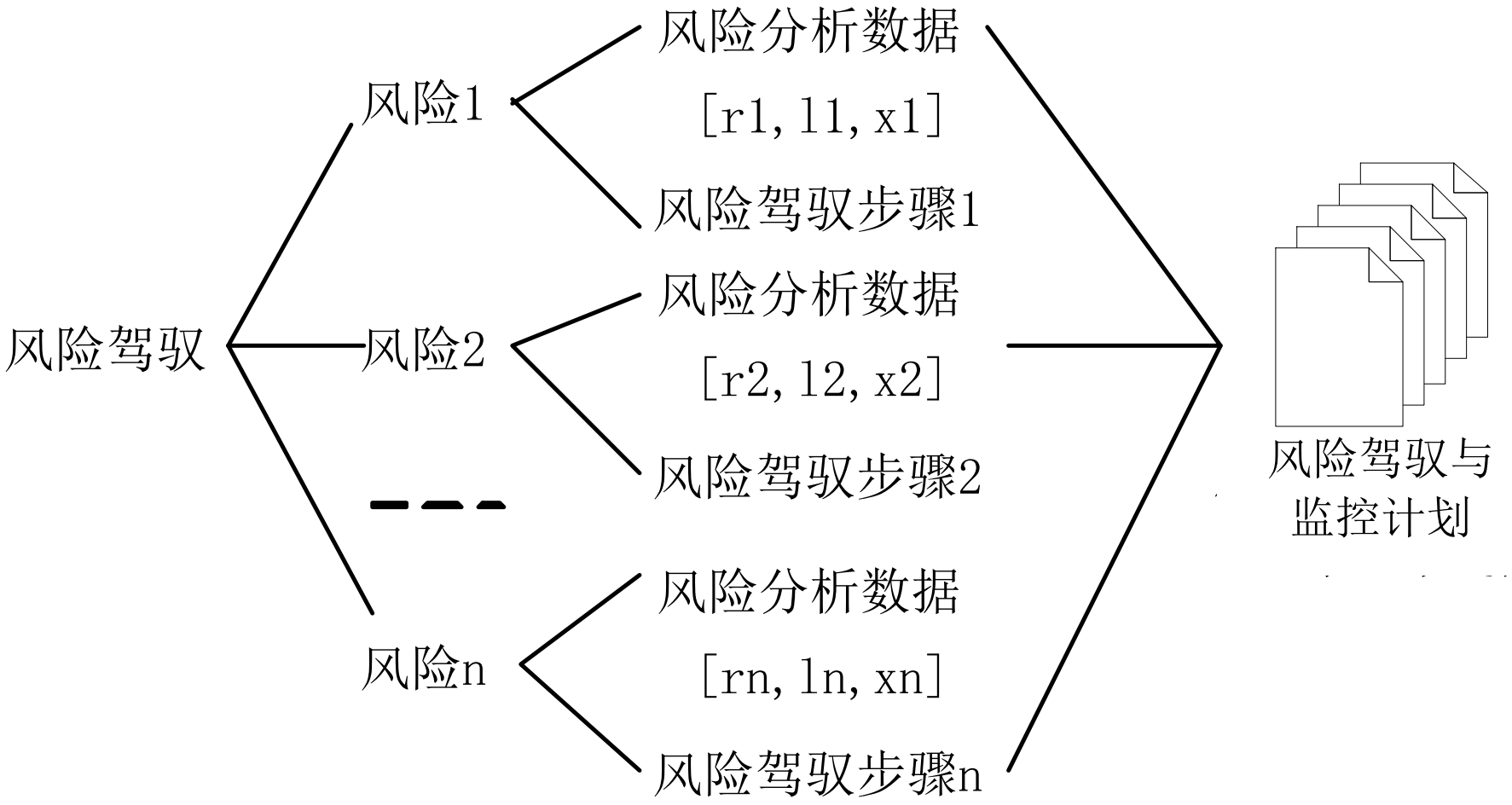

图13-6 风险驾驭与监控

1.引言
1.1 本文档的范围和目的
1.2 概述　a.目标　b.风险消除优先级
1.3 组织　a.管理　b.职责　c.作业描述
1.4 消除过程描述　a.进度安排　b.主要里程碑和评审　c.预算
2.风险分析
2.1 识别　a.风险概述　（i）风险源　b.风险分类
2.2 风险估计　a.估算风险概率　b.估算风险后果　c.估算规则
d.可能的估算错误源
2.3 评价　a.评价所使用的方法　b.评价方法的假设和限制
c.评价风险参照　d.评价结果
3.风险驾驭
3.1 劝告
3.2 风险消除的选项
3.3 风险消除的劝告
3.4 风险监控过程
4.附录
4.1 风险位置的估算
4.2 风险排除计划

图13-7 风险驾驭与监控计划概要

一旦制定出RMMP，软件项目也开始时，风险监控就开始了。风险监控的三个主要目标是：

（1）判断一个预测的风险事实上是否发生了。

（2）确保针对某个风险而制定的风险消除步骤正在合理地实施。

（3）收集可用于将来的风险分析的信息。

实际上，项目中发生的问题总能追踪到许多风险。风险监控的另一项工作就是要把“责任”（什么风险导致问题发生）分配到项目中去。虽然风险分析会增大成本，但是相对于因为严重的风险发生而没有采取有效的措施造成的项目损失来说，这些工作量花得值得。

13.5 进度安排

软件开发项目的进度安排有两种考虑方式：

（1）系统最终交付日期已经确定，软件开发部门必须在规定期限内完成。

（2）系统最终交付日期只确定了大致的年限，最后交付日期由软件开发部门确定。

对于前一种情形，只能从交付日期开始往前推，安排软件开发周期中的每一个阶段的工作。如果时间太紧，就要想办法增加资源来节省时间。后一种安排能够对软件开发项目进行细致的分析，最好地利用资源，合理地分配工作，而最后的交付日期则可以在对软件进行仔细地分析之后再确定下来。

进度安排的准确程度非常重要。因为进度安排落空，导致的后果可能非常严重。比如会导致市场机会的丧失，使用户不满意，而且也会导致成本的增加。

在制定进度安排时，要考虑人员数量和生产率的关系，要考虑如何追踪进度，要关注开发的关键路径等。

13.5.1 软件开发小组人数与软件生产率

在软件开发中，生产率和人数往往是成反比的。在通常情况下，一个人的开发小组效率是最高的。主要的原因是减少了人员之间的通信和理解工作量。但是，不可能让一个人工作10年，而是让10个人工作1年去进行软件开发。因此，需要多人组成开发小组共同参加一个项目的开发。多个人开发同一个项目会产生通信问题，对接口、设计的理解。通信需花费时间和代价，降低软件生产率。

但是小组这种软件开发形式便于开展质量保证活动，可以获得更完善的软件分析与设计，从而减少因为软件分析引起的错误数，从而降低测试工作量。

一个估计有33000LOC，需要花费12个人年的软件可以用8个人工作1.3年完成。如果把完成时间延长到1.75年，根据Putnam软件方程，可得：

$K = L3/(Ck3 \times td4) \approx 3.8$人年

这表明，如果把完成时间延长6个月，就可以把人员数从8个减少到4个。这种结果不一定可靠，但可以帮助我们在制定进度时作出定性的分析。经验表明，软件开发小组的规模在2~8人左右为宜。

13.5.2 任务的确定与并行性

当参加同一个软件工程项目的人数不止一人的时候，开发工作就会出现并行情形。如图13-8所示表示了一个典型的由多人参加的软件工程项目的任务图（*表示项目阶段任务的里程碑）。

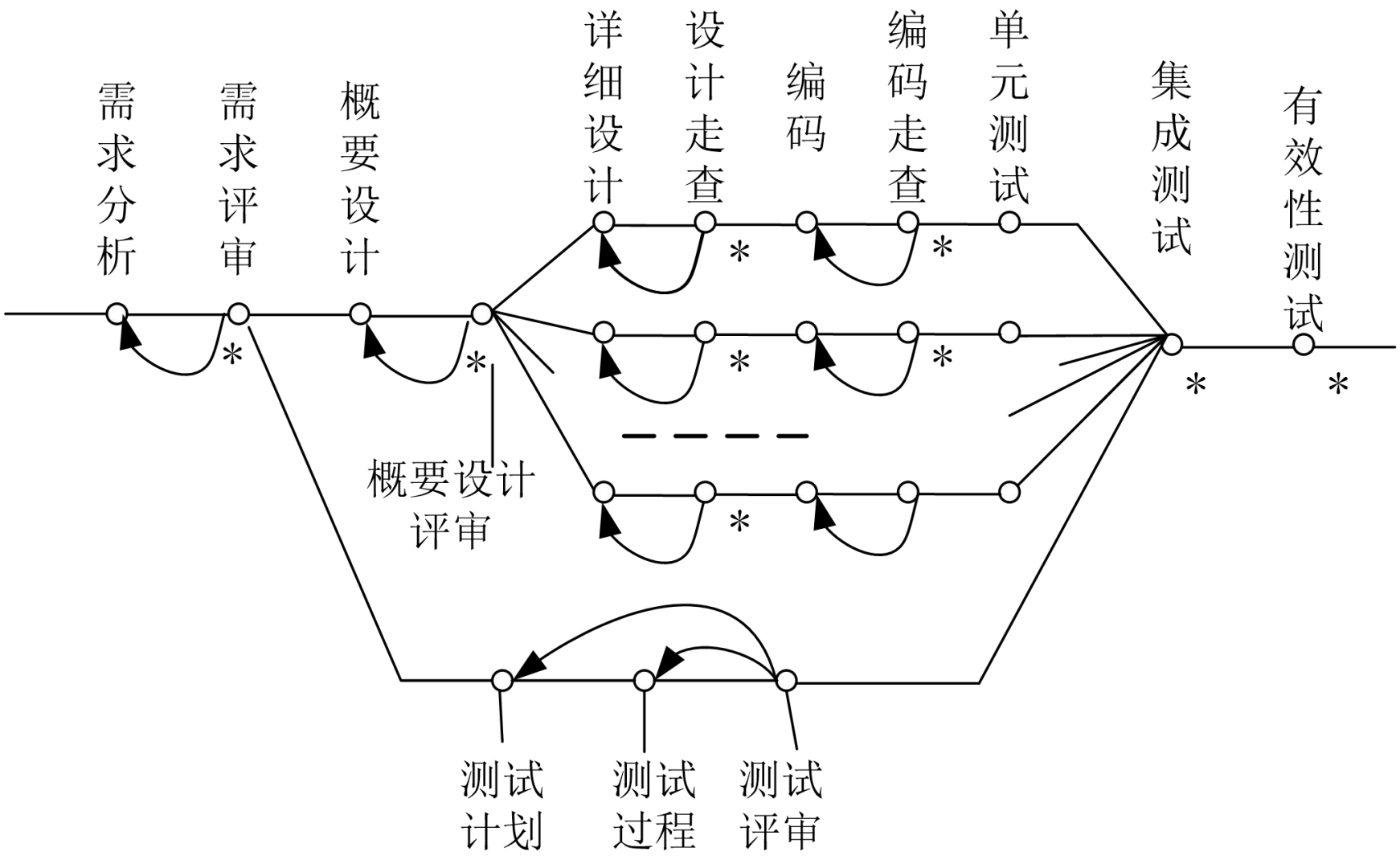

图13-8 软件项目的并行性

在软件项目的各种活动中，在完成了软件的需求分析，并通过了评审后，概要设计（系统结构设计和数据设计）工作和测试计划制定工作就可以并行进行。

在随后的各个模块的详细设计、编码、单元测试等工作又可以并行进行。待到每一个模块都已经调试完成，就可以对它们进行组装，并进行组装测试，最后进行确认测试，为软件交付进行确认工作。

在图13-8中可以看到，软件开发进程中设置了许多里程碑。里程碑为管理人员提供了指示项目进度的可靠依据。当一个软件工程任务成功地通过了评审并产生了文档之后，就完成了一个里程碑。

软件工程项目的并行性提出了一系列的进度要求。按统筹的思想，对并行的任务制定进度时，必须决定任务之间的从属关系，确定各个任务的先后次序和衔接，确定各个任务完成的持续时间。

因此，项目负责人的一个重要任务就是要时刻注意关键路径的任务，即若要保证整个项目能按进度要求完成，就必须保证这些任务要按进度要求完成。这样就可以确定在进度安排中应保证的重点。

13.5.3 制定开发进度计划

在制定软件的开发进度计划时，有一种常用来估计在整个定义与开发阶段工作量分配的简单方案，称为40—15—40规则。该规则表明在开发过程中，编码的工作量仅占15%，编码前的工作量占40%，编码后的工作量占40%。该规则过于简单，只能是一个粗略的指导。有许多更好的制定方法。

对实际开发的软件项目进行统计，发现花费在计划阶段的工作量很少超过总工作量的2%~3%。需求分析可能占项目工作量的10%~25%。花费在分析或原型化上面的工作量应当随项目规模和复杂性成比例地增加。

通常用于软件设计的工作量在15%~25%之间。而用在设计评审与反复修改的时间也必须考虑在内。由于软件设计已经投入了工作量，因此其后的编码工作相对来说困难要小一些，用总工作量的15%~15%就可以完成。测试和随后的调试工作约占软件开发工作量的30%~40%。

所需要的测试量往往取决于软件的重要程度。如果软件与人命相关，在测试阶段花费的工作量可能达到其余各个阶段的3～5倍。

在制定进度计划时，可以参照前面介绍的经验模型中对工作量的估计。如果利用基本COCOMO模型或其他公式，可参照表15-3所示的进度分配百分比表。

阶段	需求分析	设计	编码与单元测试	组装与测试
占开发时间的百分比	10~30	17~27	25~60	16~28

表13-3 进度分配百分比表

按此比例确定各个阶段工作量的分配，从而进一步确定每一阶段所需的开发时间，然后在每个阶段，进行任务分解，对各个任务再进行工作量和开发时间的分配。一直到项目负责人认为较好控制的水平。表15-4是利用基本COCOMO模型，给出的一个较为准确的进度分配表。

总体类型	阶段分配	规模（KDSI）				
		微型 ＜2	小型 8	中型 32	大型 128	特大型 512
组织型	计划与需求	10	11	12	17	
	设计	19	19	19	19	
	编码与单元测试	63	59	55	51	
	组装与测试	18	22	26	30	
半独立型	计划与需求	16	18	15	22	24
	设计	24	25	26	27	28
	编码与单元测试	56	52	48	44	40
	组装与测试	15	23	26	29	32
嵌入型	计划与需求	24	28	32	36	40
	设计	30	32	34	36	38
	编码与单元测试	48	44	40	36	32
	组装与测试	22	24	26	28	30

表13-4 更精确的进度分配表

13.5.4 进度安排的图形方法

在具体的制定软件项目的进度安排时，有很多的方法。甚至可以将任何一个多重任务的安排方法直接应用到软件项目的进度安排上。常用的有表格法和图形法。采用图示的方法比使用语言叙述在表现各项任务之间进度的相互依赖关系上更清楚。

在图示法中，必须明确标明各个任务的计划开始时间、完成时间；各个任务完成的标志（即○文档编写和△评审）；各个任务与参与工作的人数、各个任务与工作量之间的衔接情况；完成各个任务所需的物理资源和数据资源。

甘特图（Gantt Chart）是常用的多任务安排工具。用水平线段表示任务的工作阶段，用垂直线表示当前的执行情况；线段的起点和终点分别对应着任务的开工时间和完成时间；线段的长度表示完成任务所需的时间。

在甘特图中，任务完成的标准是以应交付的文档与通过评审为标准。因此在甘特图中，文档编制与评审是软件开发进度的里程碑。甘特图的优点是标明了各任务的计划进度和当前进度，能动态地反映软件开发进展情况。缺点是难以反映多个任务之间存在的复杂的逻辑关系。如图13-9所示给出一个具有5个任务的甘特图（任务名分别为A、B、C、D、E）。从甘特图上可以很清楚地看出各子任务在时间上的对比关系。

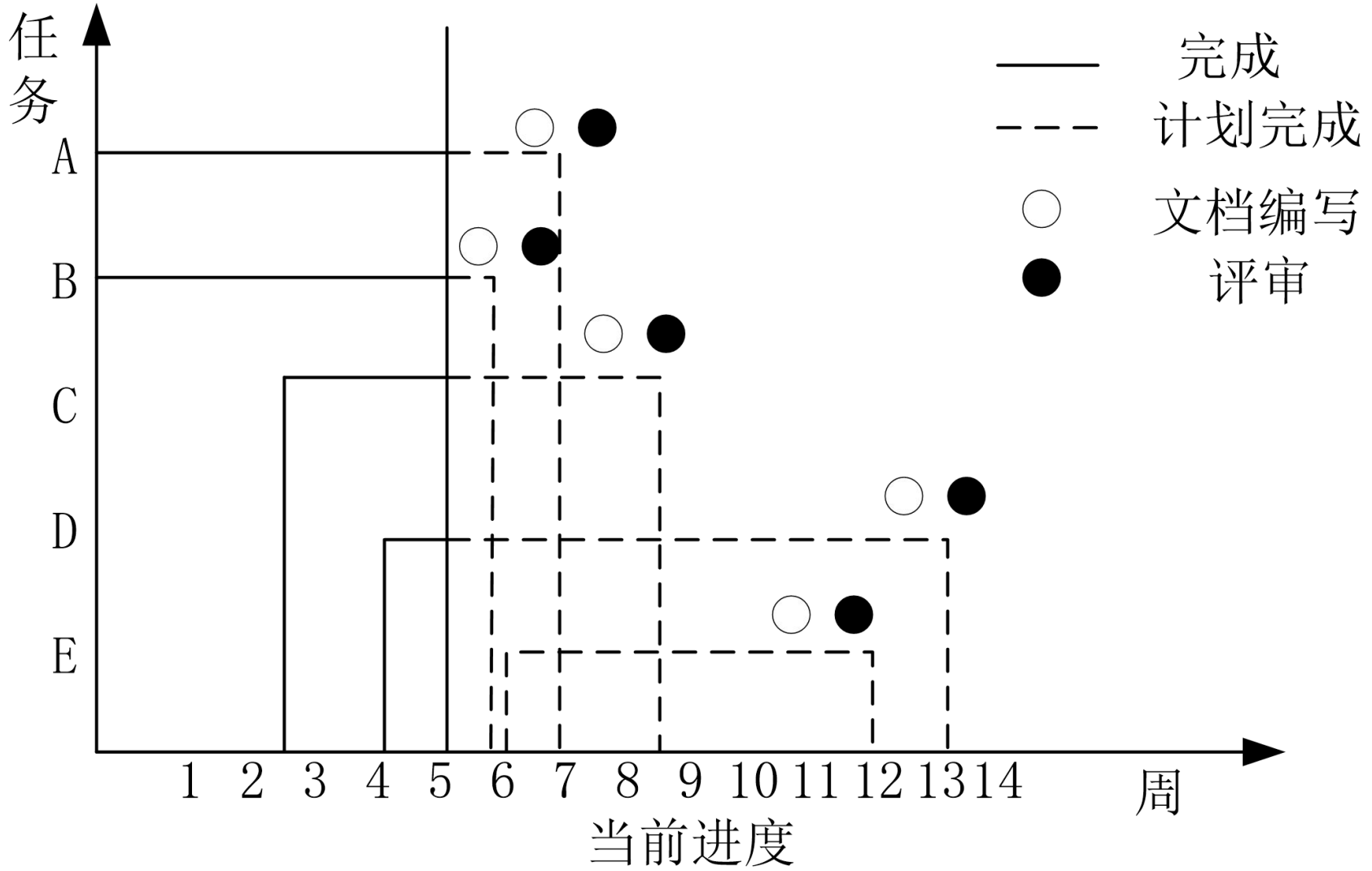

图13-9 甘特图

有成熟的软件工具来绘制甘特图，并给出各个任务之间的联系，制定进度安排表。

最后，在软件工程项目中必须处理好进度与质量之间的关系。在软件开发实践中常常会遇到这样的事情，当任务未能按计划完成时，只好设法加快进度赶上去。但事实说明，在进度压力下赶任务，其成果往往是以牺牲产品的质量为代价的。

13.5.5 项目的追踪和控制

一件事情，无论计划做得多完美，如果没有严格的过程管理，执行不力，失败的可能性非常大。软件项目管理的一项重要工作就是在项目实施过程中进行追踪，对过程进行严格的控制。可以用以下不同的方式进行追踪：

（1）定期举行项目状态会议。在会上，每一位项目成员报告他的进展和遇到的问题。

（2）评价在软件工程中所产生的所有评审的结果。

（3）确定由项目的计划进度所安排的可能选择的正式的里程碑。

（4）比较在项目资源表中所列出的每一个项目任务的实际开始时间和计划开始时间。

（5）非正式地与开发人员交谈，以得到他们对开发进展和刚冒头的问题的客观评价。

13.6 软件项目的组织

13.6.1 软件项目管理的特点

解决软件项目管理的问题，要涉及到社会的因素、精神的因素、人的因素，比单纯的技术问题要复杂得多。仅靠技术、工程或科研项目的效率、质量、成本和进度分析，很难较好的解决软件项目管理中的问题。

因此，单纯形式地照搬曾经成功的经验不一定奏效。此外，管理技术的基础是实践，为取得管理技术的成果必须反复实践。很显然，管理能够带来效率，能够赢得时间，最终将在技术前进的道路上取得领先地位。

1. 软件项目的特点

软件产品与其他任何产业的产品不同，没有重要、明显的物理属性，但在软件产品中融合了软件开发人员的思想、概念、算法、流程、组织、效率。软件产品开发中首先要了解的是用户的要求，但是常常因为用户不了解计算机而造成叙述的困难。文档编制工作在软件开发中占有非常重要的地位。软件的特点体现在以下的几个方面：

1）智力密集，可见性差

软件工程过程充满了大量高强度的脑力劳动。软件开发的成果是不可见的逻辑实体，软件产品的质量难以量化。对于不深入掌握软件知识或缺乏软件开发实践经验的人员，是不可能领导做好软件管理工作的。软件开发完成后的质量难以准确评价。

2）单件生产

软件的开发环境都是在特定的软硬件环境中，再加上软件项目特定的目标，使得软件具有独一无二的特色，几乎找不到与之完全相同的软件产品。这种建立在内容、形式各异的基础上的研制或生产方式，与其他领域中大规模现代化生产有着很大的差别，也自然会给管理工程造成许多实际困难。

3）劳动密集，自动化程度低

软件项目经历的各个阶段都渗透了大量的手工劳动，这些劳动十分细致、复杂和容易出错。尽管近年来开展了软件工具和CASE的研究，但总体来说，仍远未达到自动化的程度，使得软件的正确性难以保证。

4）软件工作渗透了人的因素

在软件开发的各阶段中，人的因素始终处在一个特殊和重要的位置。为高质量地完成软件项目，需要经验丰富的开发人员，还要充分发掘人员的智力才能和创造精神。软件人员的情绪和他们的工作环境，对他们工作有很大的影响。与其他行业相比，它的这一特点十分突出，必须给予足够的重视。

2. 软件管理的主要职能

软件管理的主要职能包括：

（1）制定计划：规定待完成的任务、要求、资源、人力和进度等。

（2）建立组织：为实施计划，保证任务的完成，需要建立分工明确的责任制机构。

（3）配备人员：任用各种层次的技术人员和管理人员。

（4）指导：鼓励和动员软件人员完成所分配的工作。

（5）检验：对照计划或标准，监督和检查实施的情况。

15.6.2 软件项目组织的建立

建立一个好的组织来进行软件开发，是一切软件项目开发能够顺利进行的必要条件之一。无法想象松散、责任不明确的一伙人在一起可以高效地开发软件。针对软件项目的特点，开发人员的工作习惯、素质甚至民族特性来决定开发组织采用什么形式。人的因素是不容忽视的参数。

在建立软件开发的组织时，要注意：

（1）尽早落实责任：在软件项目工作的开始，就要指定专人负责。使他有权进行管理，并对任务的完成负全责。

（2）减少接口：在开发过程中，人与人之间的联系是必不可少的，组织应该有合理的分工、好的组织结构，以减少不必要通信。

（3）责权均衡：软件经理人员所负的责任不应比委任给他的权力还大。

通常有三种组织的模式可供选择：

1. 按课题划分的模式（Project Format）

把软件人员按软件项目中的课题组成小组，小组成员自始至终参加所承担课题的各项任务，包括完成软件产品的定义、设计、实现、测试、复查、文档编制，甚至包括维护在内的全过程。

2. 按职能划分的模式（Functional Format）

软件开发的周期是按阶段来划分的，在每个阶段都有不同的特点，对人员的技术和经验也有不同的要求。按职能划分的模式就是把参加开发项目的软件人员按任务的工作阶段划分成若干个专业小组。例如，分别建立计划组、需求分析组、设计组、实现组、系统测试组、质量保证组、维护组等。各种文档资料按软件开发的阶段在各组之间传递。这种模式的缺点是小组之间的接口较多，但便于软件人员熟悉小组的工作，进而变成这方面的专家。

3. 矩阵形模式（Matrix Format）

结合前面两种模式的优点，就形成了矩阵形模式。一方面，按工作性质，成立一些专门组，如开发组、业务组、测试组等；另一方面，每一个项目又有它的经理人员负责管理。每个软件人员属于某一个专门组，又参加某一项目的工作。例如，属于测试组的一个成员，他参加了某一项目的研制工作，因此他要接受双重领导（一是测试组，一是该软件项目的经理）。如图13-10所示。

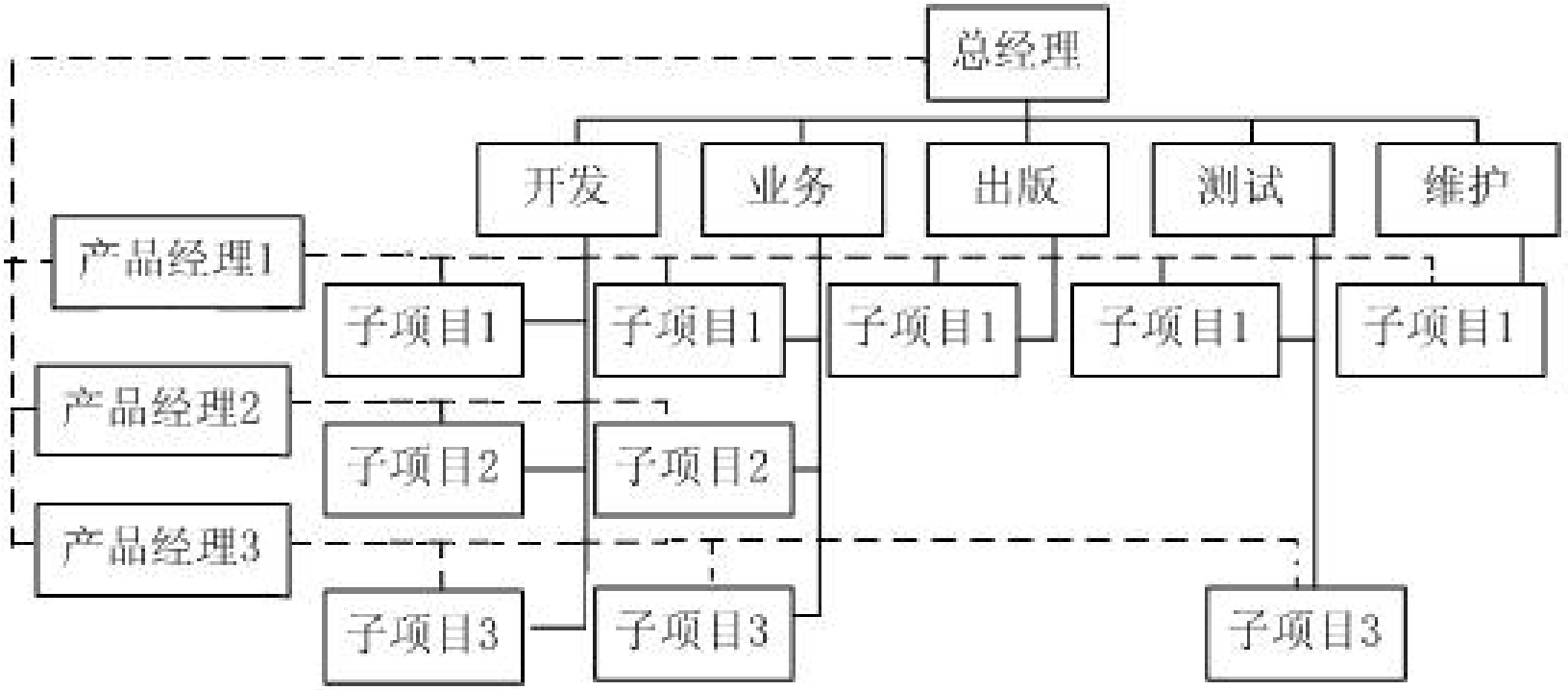

图13-10 软件开发组织的矩阵形模式

矩阵形结构组织的优点是：参加专门组的成员可在组内交流在各项目中取得的经验，这更有利于发挥专业人员的作用。而且各个项目有专人负责，有利于软件项目的完成。

程序设计小组的组织形式：在程序设计小组中典型的有三种组织形式。如图15-11所示，上排的三种为结构形式，下排的三种为通信路径。

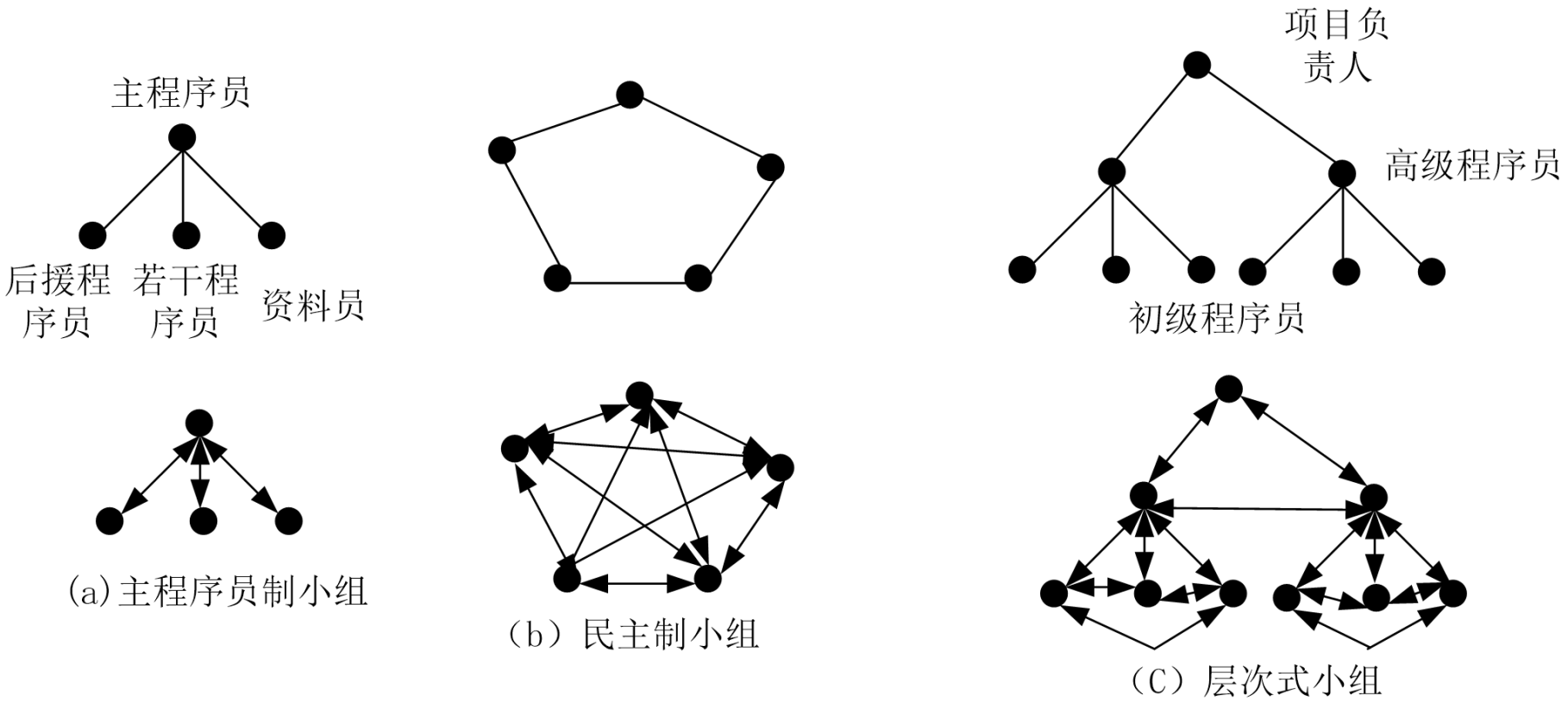

图13-11 三种不同的小组结构

（1）主程序员制小组（Chief Programmer Team）。

小组的核心由1位主程序员（高级工程师）、2至5位技术员、1位后援工程师组成，另外还可以有部分辅助人员（资料员）。主程序员负责小组全部技术活动的计划、协调与审查工作，还负责设计和实现项目中的关键部分。技术员负责项目的具体分析与开发，以及文档资料的编写工作。后援工程师协助和支持主程序员的工作，为主程序员提供咨询，也做部分分析、设计和实现的工作。如图13-12所示。

主程序员制的开发小组强调主程序员与其他技术人员的直接联系，简化了技术人员之间的横向通信，如图13-11（a）所示。这种组织制度的工作效果的好坏很大程度上取决于主程序员的技术水平和管理才能。

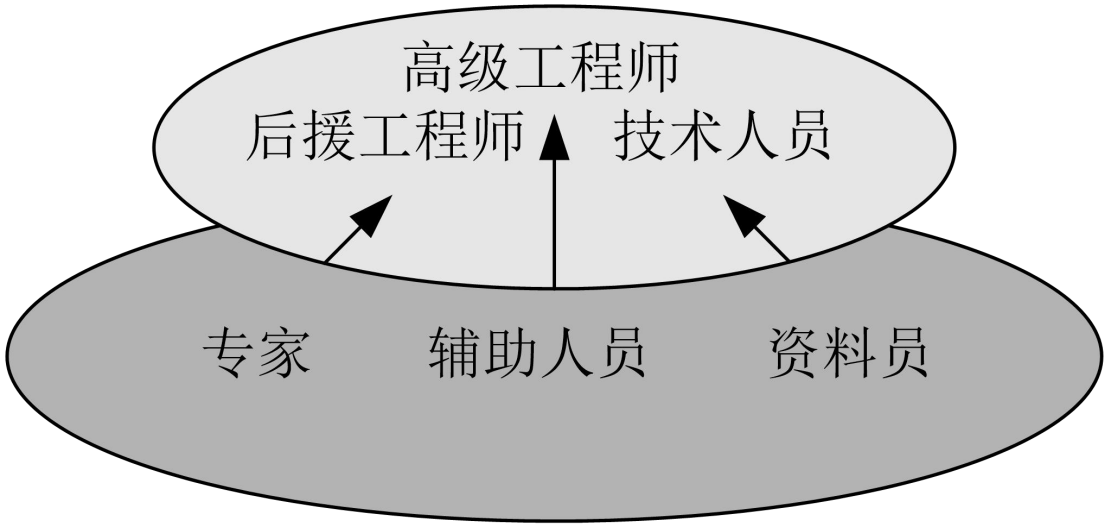

图13-12 主程序员小组的组织

（2）民主制小组（Democratic Team）。

在民主制小组中也设置一位组长，但是每当遇到问题，组内成员之间可以平等地交换意见，如图13-11（b）所示。

工作目标的制定及做出决定都由全体成员参加。这种组织形式强调发挥小组每个成员的积极性、主动精神和协作精神。缺点是会削弱了个人责任心和必要的权威作用。这种组织形式适合于研制时间长、开发难度大的项目。

（3）层次式小组（Hierarchical Team）。

如图13-11（c）所示，在层次式小组中，组内人员分为三级：组长（项目负责人）1人负责全组工作。他直接领导2～3名高级程序员，每位高级程序员通过基层小组，管理若干位程序员。这种组织结构特点比较适合的项目就是层次结构状的课题，可以按组织形式划分课题，然后把子项目分配给基层小组，由基层小组完成。对于大型项目，可以通过层次式划分将项目需要划分成若干层，因此，大型软件项目的开发比较适合于这种组织方式。

在实际应用中，组织形式并不是一成不变的，可以根据问题的特点调整。但是调整的幅度过大，会造成成员之间的交流不顺。

总之，软件开发小组的主要目的是发挥集体的力量进行软件研制。因此，小组培养从“全局”的观点出发进行程序设计，消除软件的“个人”性质，并促进更充分的复审，小组提倡在共同工作中互相学习从而改善软件的质量。

13.6.3 人员配备

在不同的软件开发阶段，应该合理地配备人员，这也是成功地完成软件项目的切实保证。在各个阶段对人员的数量、技术水平的需求是不同的。

如图15-13所示的就是各阶段对人员的需求情况。一个软件项目完成的快慢，取决于参与开发人员的多少。在实际的软件开发过程中，多数软件项目是以恒定人力配备的。恒定的人力配备会带来各种问题。

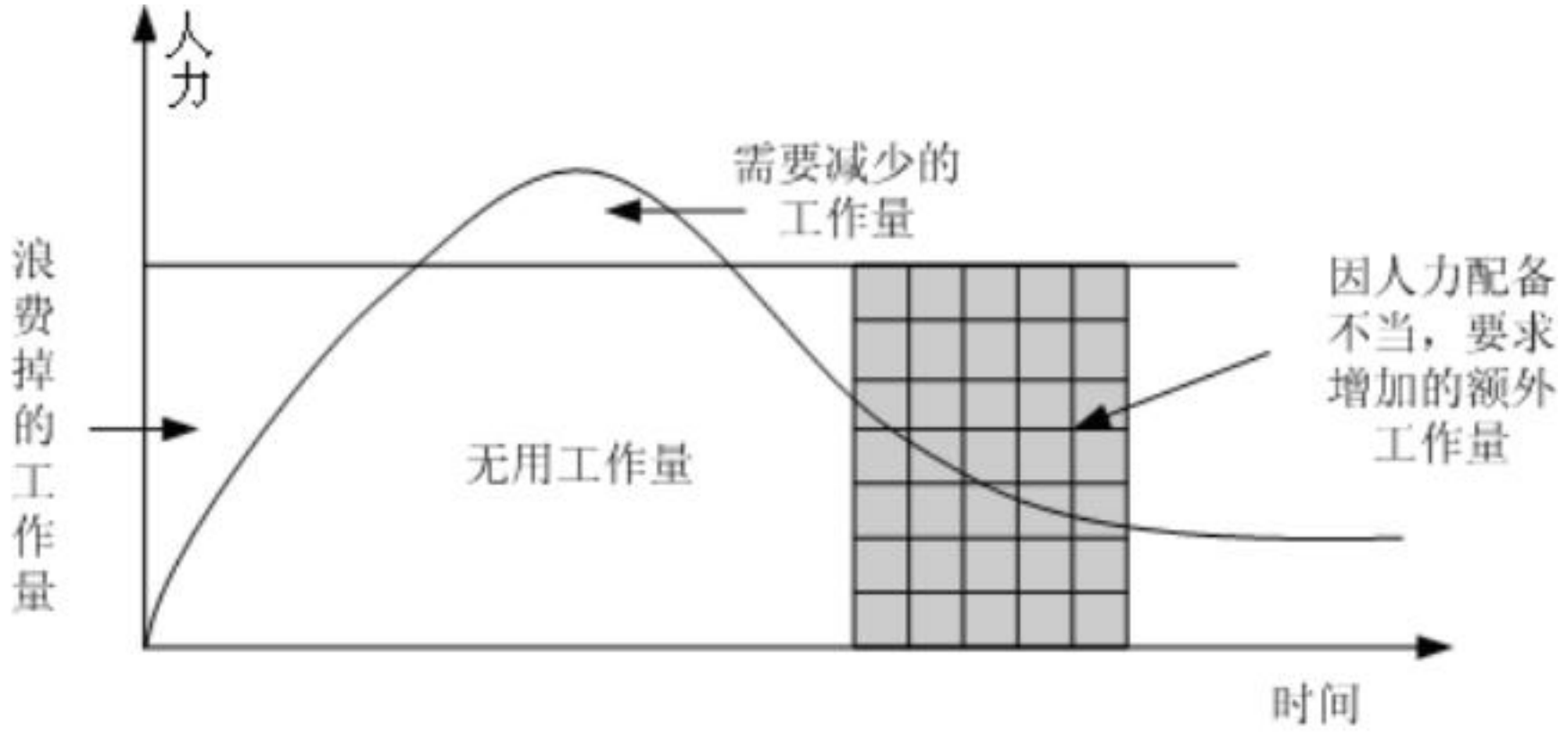

图13-13 软件项目人力配备情况

图13-13中的人力需求曲线就是在前面介绍Putnam模型时曾引入的Rayleigh-Norden工作量分布曲线。该曲线表示需求定义结束后的软件生存期（包括运行和维护）内的工作量分布情况，它也是投入人力与开发时间的关系曲线。按此曲线，需要的人力随开发的进展逐渐增加，在编码与单元测试阶段达到高峰，以后又逐渐减少。

如果恒定地配备人力，在开发的初期，将会有部分人力资源用不上而浪费掉。在开发的中期（编码与单元测试），需要的人力又不够，造成进度的延误。这样在开发的后期就需要增加人力以赶进度。因此，恒定地配备人力，对人力资源是比较大的浪费。

为合理地安排人力，可利用Rayleign-Norden曲线计算开发时间t所需要的人力资源：

dY/dt = (Kt/td2)·exp (－t2/2td2)　　（K＞0，td＞0）

其中，Y为在达到时刻t以前需要投入的全部人力数（人年），K为整个生存期内需要投入的总人力数（人年）：

td = dY/dt

为开发时刻的最大值，根据以往的开发经验，td大约与软件开发的收尾期相重合。

由上式可以推导出：

$Y = K\{1 - \exp(-t^2/2td^2)\}$

其中的K（总工作量）和td（开发时间）可以由下面的软件方程得到：

$L = Ck \cdot K^{1/3} \cdot td^{4/3}$　　（$Ck > 0$）

其中，L是前面介绍的开发规模（LOC），Ck是技术状态常数。

根据软件开发各阶段对人力的需求情况，配备人员时应该遵循如下的原则：

（1）重质量：软件项目是技术性很强的工作，任用少量有实践经验、有能力的人员去完成关键性的任务，常常要比使用较多的经验不足的人员要更有效。

（2）重培训：花力气培养所需的技术人员和管理人员是有效解决人员问题的好方法。

（3）双阶段提升：人员的提升应分别按技术职务和管理职务进行，不能混在一起。

1. 对项目经理人员的要求

软件经理人员是工作的组织者，他的管理能力的强弱是项目成败的关键。除去一般的管理要求外，他应具有以下能力：

（1）把用户提出的非技术性要求加以整理提炼，以技术说明书形式转告给分析员和测试员。

（2）能说服用户放弃一些不切实际的要求，以便保证合理的要求得以满足。

（3）能够综合用户的需求，归结为“需要什么”，“要解决什么问题”。

（4）同用户和上级有很强的沟通能力，能清晰明白地了解对方的思想和表达自己的思想。

2. 评价人员的条件

软件项目中人的因素越来越受到重视，因此对开发人员的评价也显得越来越重要。需要有一个好的方法来了解开发人员的管理能力和技术水平。人员素质的优劣常常影响到项目的成败，一个好的开发人员应该在以下的几方面有较好的表现：

（1）牢固掌握计算机软件的基本知识和技能。

（2）善于分析和综合问题，具有严密的逻辑思维能力。

（3）工作踏实、细致，不靠碰运气，遵循标准和规范，具有严格的科学作风。

（4）工作中表现出有耐心、有毅力、有责任心。

（5）善于听取别人的意见，善于与周围人员团结协作，建立良好的人际关系。

（6）具有良好的书面和口头表达能力。

小结

本章主要介绍了软件项目管理。一个好的项目管理方法对软件开发的成功是至关重要的。软件的项目管理有它自己的特点，主要体现在软件开发是一个智力密集、单件生产、劳动密集的活动，并且人的因素在其中有着重要的地位。

小结

软件项目管理的过程主要包括启动一个软件项目，对软件开发做风险估计，并采取各种策略驾驭和避免风险，保证软件的成功，准确估计软件的开发成本，制定软件开发计划的进度安排，在开发后对进度进行严格的控制，保证软件能按计划进行。

小结

到目前为止，对软件的成本的估算都是建立在各种经验模型上的，只要估计的成本和实际成本相差不过15%，估计的工作量和实际工作量的差别不到30%，就已经对软件的项目管理有很大的帮助了。

要驾驭风险，首先要识别风险。通过对风险分类，并检查各种风险项目来识别风险，对风险的重要性要进行评估，并对重要的15%的风险要建立风险驾驭策略。

小结

对进度的控制也是项目管理的重要内容，制定计划后，用图形制定进度表是一种直观的方法，项目开始后，要经常使用各种方法对进度进行追踪、控制，并进行各种资源的调配，保证项目按计划完成。

谢谢