第6章 软件项目的进度计划制订与团队组织

BACK

6.1 基本概念与工作流程

软件项目的开发，作为一个产品的“生产”或“制造”，人们(尤其是项目经理与企业管理人员)主要关心的产品目标是质量、成本、进度和团队，这四个目标构成了一个如图6.1所示的产品生成要素四边形，在此四边形中的四个生产要素是相互影响、相互制约的，任何一个生产要素的变动都将对其他三个生产要素产生重要的影响。国内、外大量的软件工程实践告诉我们，由于缺乏一个科学的项目进度计划或高效、协调的开发团队，将直接导致软件成本的无限膨胀和质量目标的无法完成。

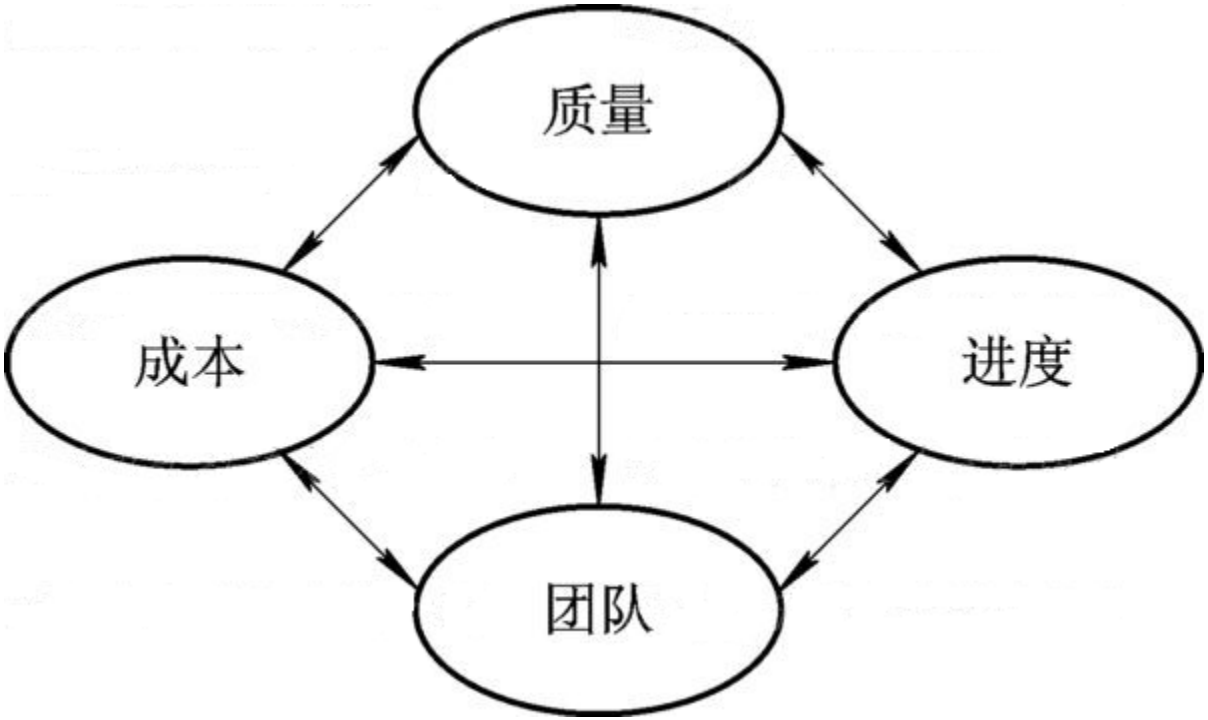


图6.1 生产要素四边形

所谓软件项目的进度计划，是指为执行软件项目的各项活动(任务)和里程碑所制订的工作计划日程表，它是项目组工作进度安排的出发点、管理人员跟踪和监控项目进展状态是否异常的判断标准和跟踪变更(进度、人力、设备)对项目影响的依据。软件项目的进度计划安排一般有如下两种状况：一种是项目交付日期已规定(如招标方和投资方的要求)，然后来安排进度计划。另一种是软件企业或项目组根据自身已有的资源(人力或资金等)来计划交付日期(工期)和安排进度计划。软件项目的进度计划制订的基础是项目工作(任务)分解与计划网络图。以下作简要介绍。

6.1.1 项目工作(任务)分解结构

软件项目的规划、分析、设计、开发与测试等任务的实施通常是由一系列的项目活动(Project Activity)或项目任务构成，为了更好地完成软件项目的计划与控制，将这一系列项目活动组成一定的层次结构是十分有用的。这种由一系列软件项目活动所组成的层次结构称为工作(任务)分解结构(Work Breakdown Structure，WBS)。利用WBS，项目经理或企业管理人员可以将整个软件项目任务进行分解，并落实到各项目团队与个人，并以此为基础来制订软件项目的时间进度计划。

需要说明的是，WBS是一种分层的树形结构，树中的每一结点是对软件项目的一项活动的表述，而该结点下的分支则是对该活动的更细致的描述，上述树形结构的最底层结点一般是可交付的工作包(Work Package)，即可交付的软件模块，或其他计划、测试、文档等任务集合。这些工作包一般应由唯一的一个团队小组或个人负责完成。上述工作分解的活动(任务)数量与层次划分应视具体情况而定，分解不能太粗，但也要避免过细，根据管理跨度的理论，通常要求WBS不超过七层，而每一层底层的工作量为每周40小时。在第1章中，表1.10给出了软件生存周期中按照需求分析、概要设计、详细设计与编码、集成与测试四个阶段划分，且每个阶段又分解为八项活动的软件开发WBS任务表。对于任何一个特定的软件项目还可在表1.10的基础上作进一步的分解，例如图6.2给出了ERP项目按阶段分解的WBS图，而图6.3则给出了ERP项目按目标或功能属性分解的WBS图。

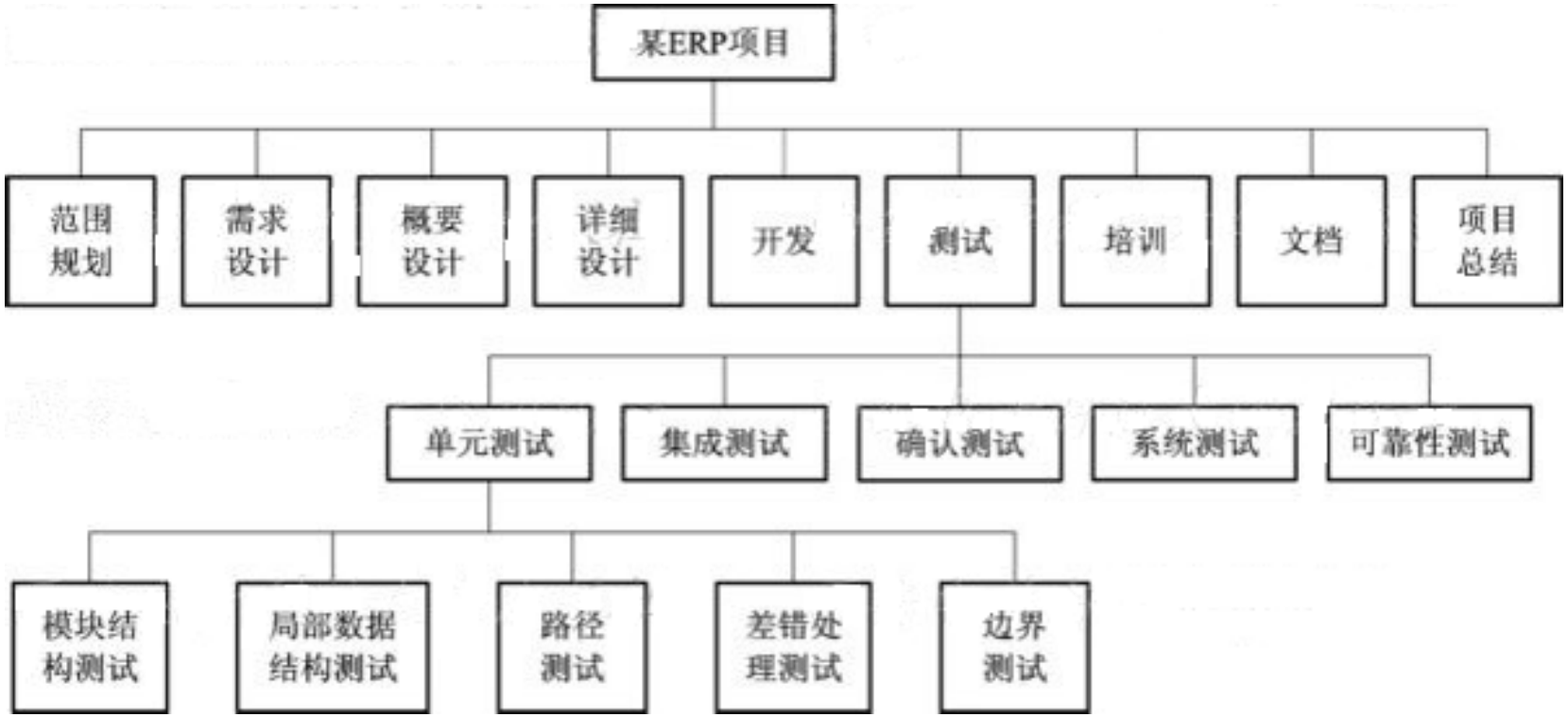


图6.2 按阶段分解的WBS图

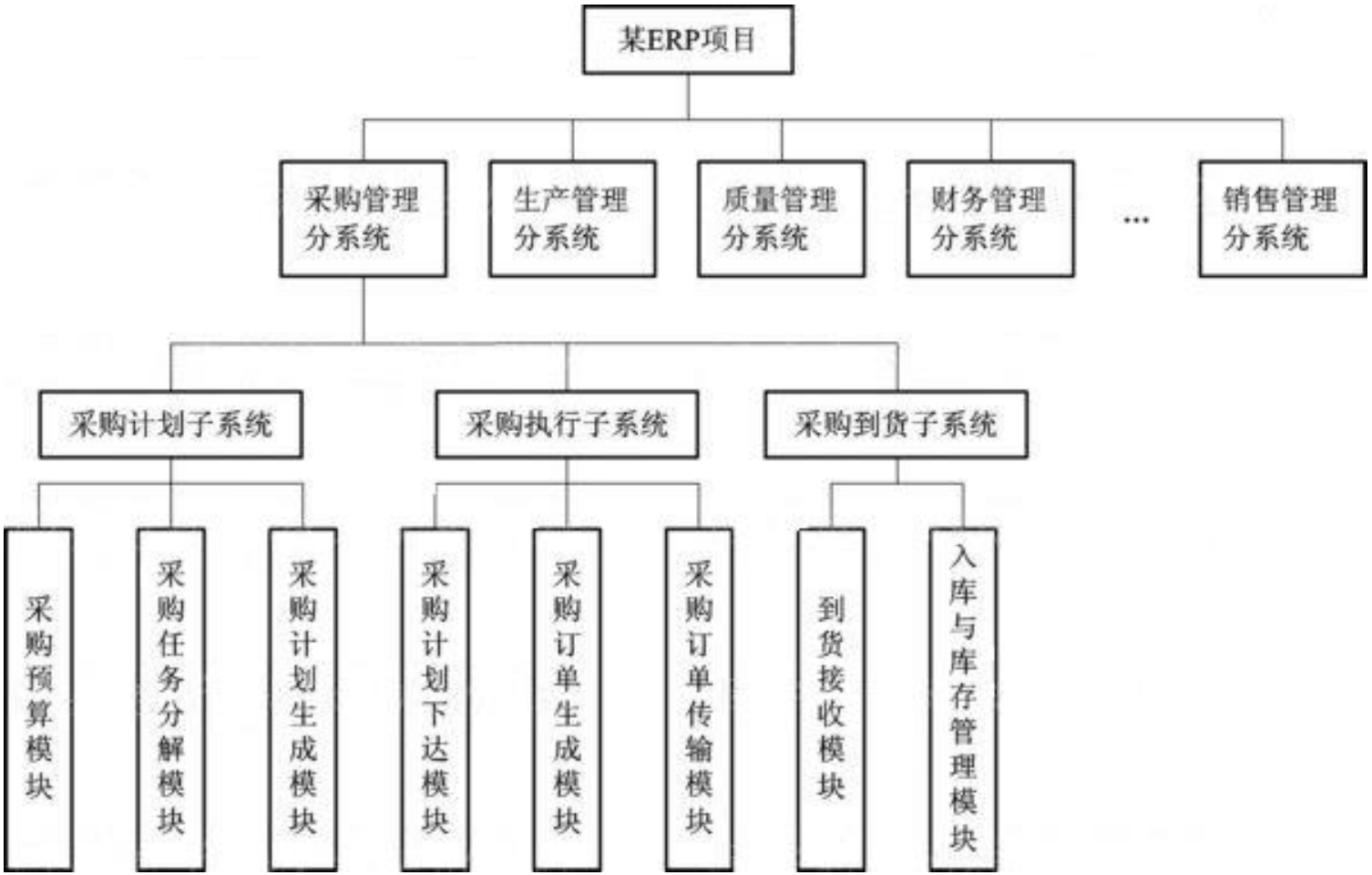


图6.3 按目标或功能属性分解的WBS图

6.1.2 活动的逻辑顺序与计划网络图

1. 活动的逻辑顺序

WBS给出了每个特定软件项目的活动(任务)及其层次结构，这种活动之间的层次结构反映了不同活动之间的地位或重要性的不同以及相互依赖与制约(支配)关系。然而，要使一个软件项目按时(工期)保质(目标)地顺利完成开发任务，仅有上述这种依赖与制约关系是不够的，它还需要各项活动实施过程中的逻辑顺序先后关系。一般来说，任何两项活动A与B存在着如下四种逻辑顺序关系：紧前关系，紧后关系，先行关系，后行关系。

若活动B的开始必须在活动A结束以后才能执行，则称B是A的后行活动，A是B的先行活动；若活动A结束后紧接着可以允许实施B活动，则称A是B的紧前活动，B是A的紧后活动。在图6.4中以箭线(有向弧)表述活动，以结点表述事项(活动的开始或结束事项)，则图6.4(a)表述了活动A与B的先行后行逻辑顺序关系，而图6.4(b)表述了活动A与B的紧前与紧后逻辑顺序关系。

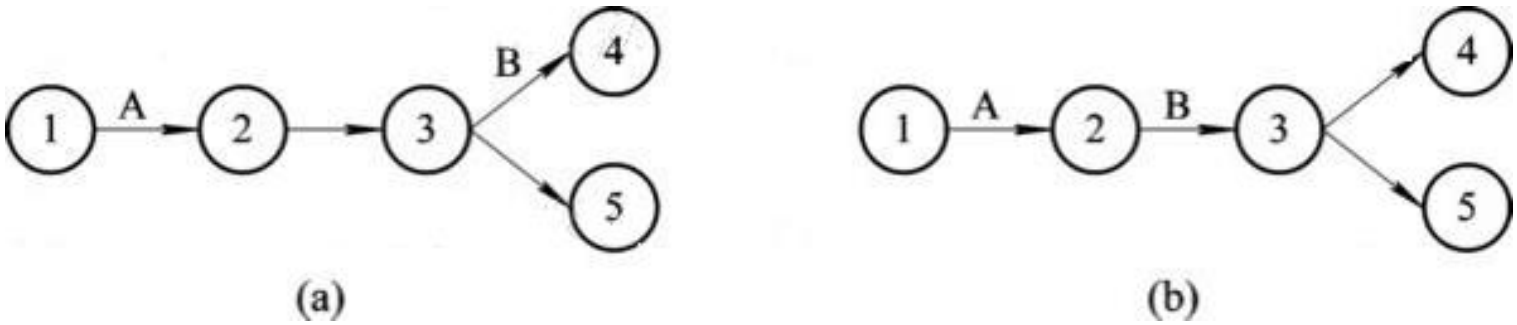


图6.4　活动间的逻辑顺序关系

需要说明的是，这种逻辑顺序关系通常是由如下四种原因所造成的：

① 两活动间的执行逻辑顺序是由客观规律和内部物质条件的限制所造成，是人们无法违背的事实。如需求分析活动必须在系统设计活动前完成，而测试活动又必须在设计与编码活动完成后方能开始。上述这种逻辑顺序关系称为刚性逻辑关系。

② 两活动间的逻辑顺序之先后带有一定的机动性，可由人的主观意志来决定。如设备管理活动与人力资源管理活动二者由于彼此的关联较少，因而在软件开发过程中哪一个活动先执行，哪一个后执行可由企业管理人员或项目经理来决定。上述这种逻辑顺序关系可称为软逻辑关系。

③ 某些活动能否可以执行依赖于外部环境或条件，如环境测试或传感器测试将依赖于外部提供的环境设备或传感器测试设备等，这样的逻辑顺序关系又称为外部依赖关系。

④ 里程碑(Milestone)。软件项目分阶段完成与进行考评的活动或活动完成时刻称为里程碑。显然，里程碑的内容决定了某些活动应是其先行活动与紧前活动，而另一些活动又应是其后行活动或紧后活动。

2. 计划网络图及其绘制规划与特性

为了系统、全面、直观、形象地反映软件项目所有活动的逻辑顺序关系，人们引进了计划进度网络图(Network Diagramming)或统筹图这一工具。计划进度网络图(简称计划网络图)是由一系列结点和有向边(有边弧)组成的反映软件项目各活动(任务)执行内在逻辑关系的赋权有向图。此中的“内在逻辑关系”即为前述的刚性逻辑顺序关系、软逻辑顺序关系、外部依赖关系和里程碑关系，它是由软件项目本身的外部环境或内部条件等因素的限制所决定。常用的计划网络图(统筹图)有结点法网络图(单代号网络图)、箭线法网络图(双代号网络图)和条件箭线图法等。本书主要介绍箭线法网络图的有关内容。

箭线法计划网络图是由一系列结点和箭线(有向弧)所构成的赋权有向图。此中箭线表述项目活动(任务)，在统筹图中它又称为作业、工序等，每一箭线的始端和终端均有两个结点，分别表示该活动的开始事项和终止事项(如图6.4(a)或(b)所示)。

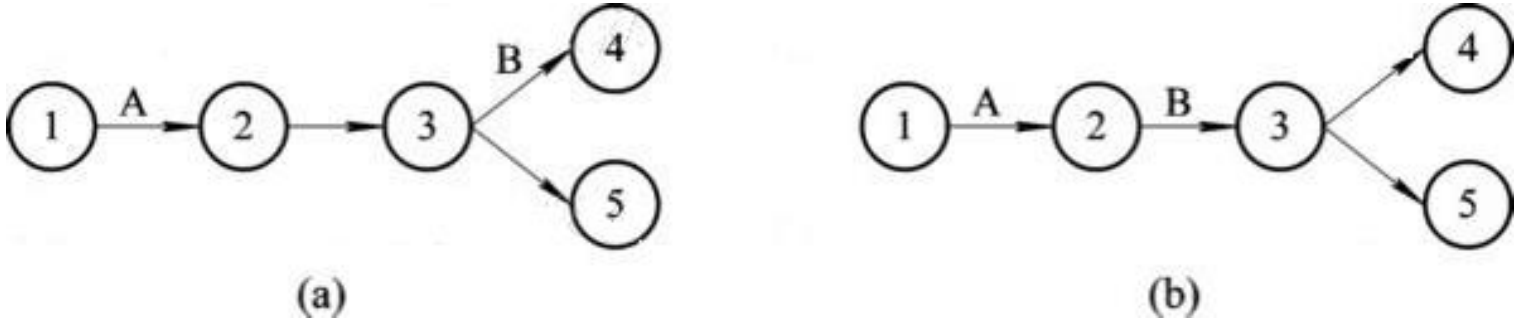


图6.4　活动间的逻辑顺序关系

以下简述箭线法计划网络图的绘制规则。

(1) 每一活动用一箭线及其前后两个结点连结来描述，箭线的上方表明活动代号(可用英文字母表述)，箭线的下方标明该活动的完成所耗费的时间长度(单位：周、月或年)，两端的结点圆圈内注明结点编号。

(2) 一对结点间只能有一条箭线，也不允许出现回路(如图6.5(a)所示)，这是由于回路的存在意味着该活动要按正反顺序执行两次，这在计划上是无意义的。

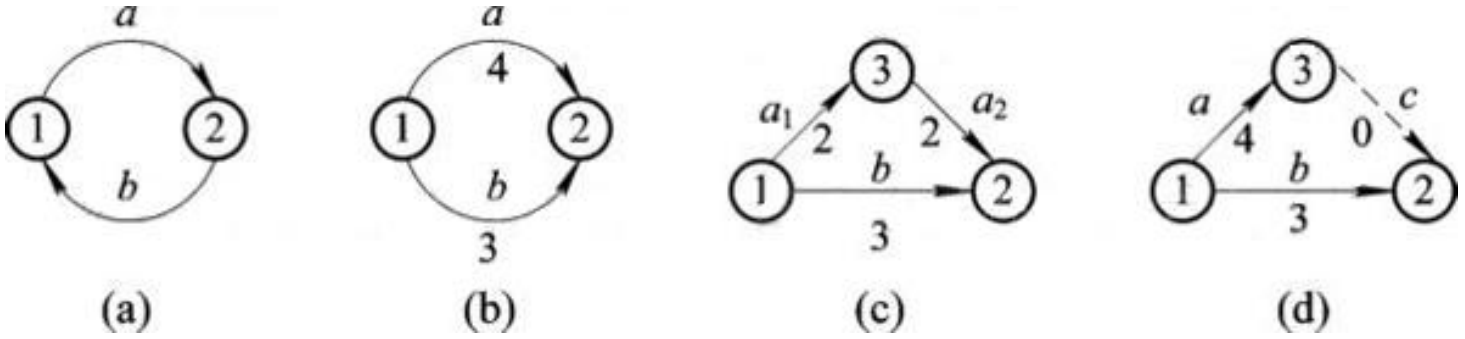


图6.5 活动规则示意图

(3) 一对结点间若出现两项以上的并行活动(如图6.5(b)所示)时，可人为地将其中之一活动一分为二或引入虚工序。此中，虚工序用虚线构成的箭线表述，如图6.5(b)中由于结点1和结点2间出现了并行活动a与b， 为符合规则(2)，人为地将活动a分成两个子活动a_1和a_2，(如图6.5(c)所示)，也可引入虚活动c(如图6.5(d)所示)，以满足规则2的要求。

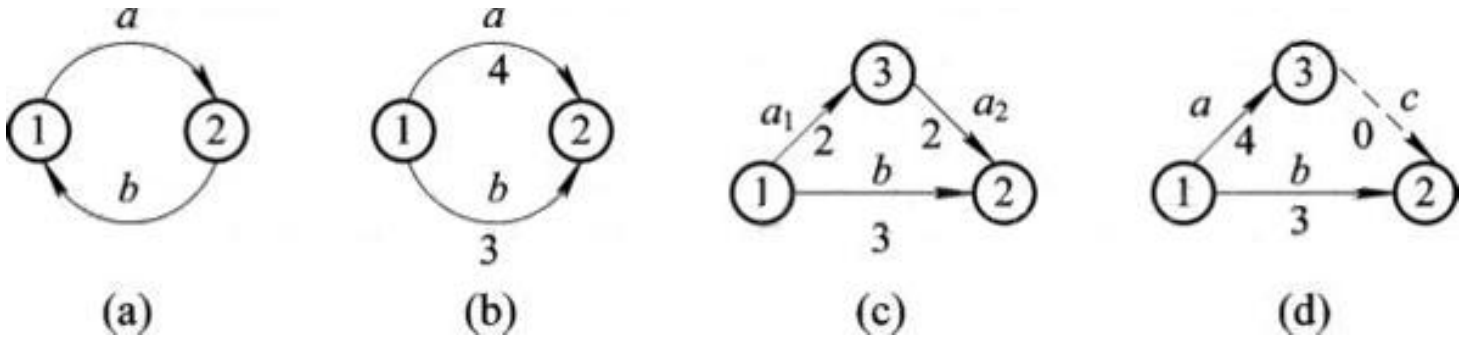


图6.5　活动规则示意图

(4) 在软件开发过程中，若出现必要的反复过程，应将活动的过程拉长或采用等效活动处理(如图6.6所示)。

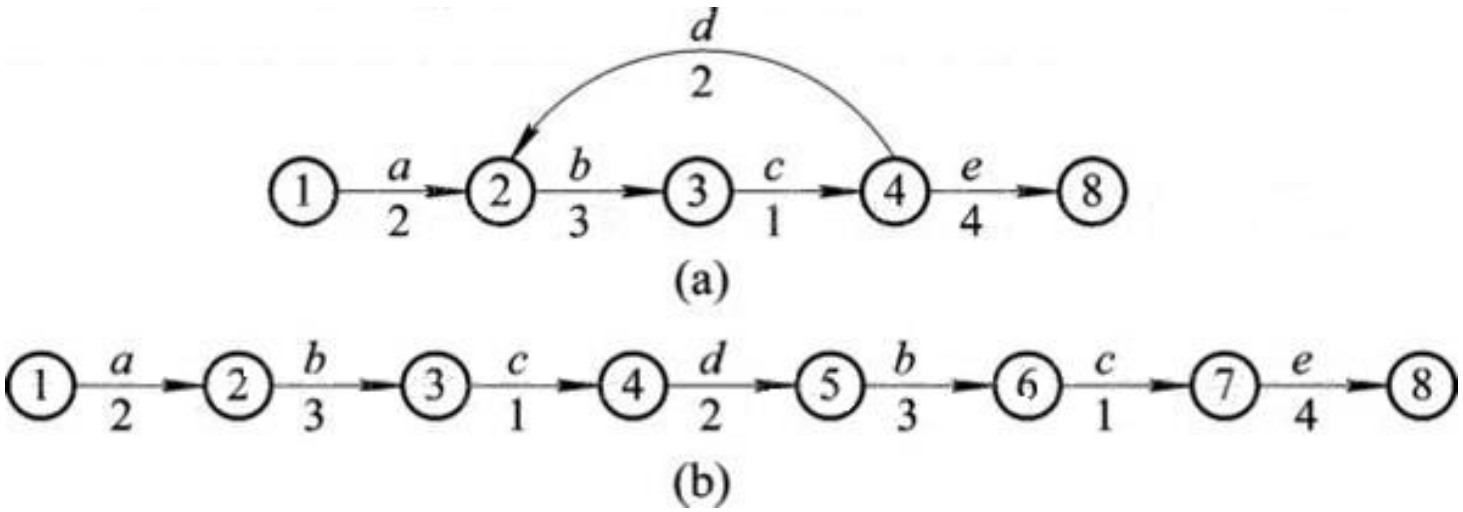


图6.6 活动出现反复过程

(5) 为加快工程进度，有时可引入交叉活动，如在图6.7(a)中，b是a的紧后活动，但为了加快工程进度，可将活动a分成三段a_1、a_2、a_3进行，这些子活动间出现了如图6.7(b)所示的交叉过程，然而由于引进了虚工序c，从而既满足了b是a的紧后活动的要求，又满足了规则(3)的要求。

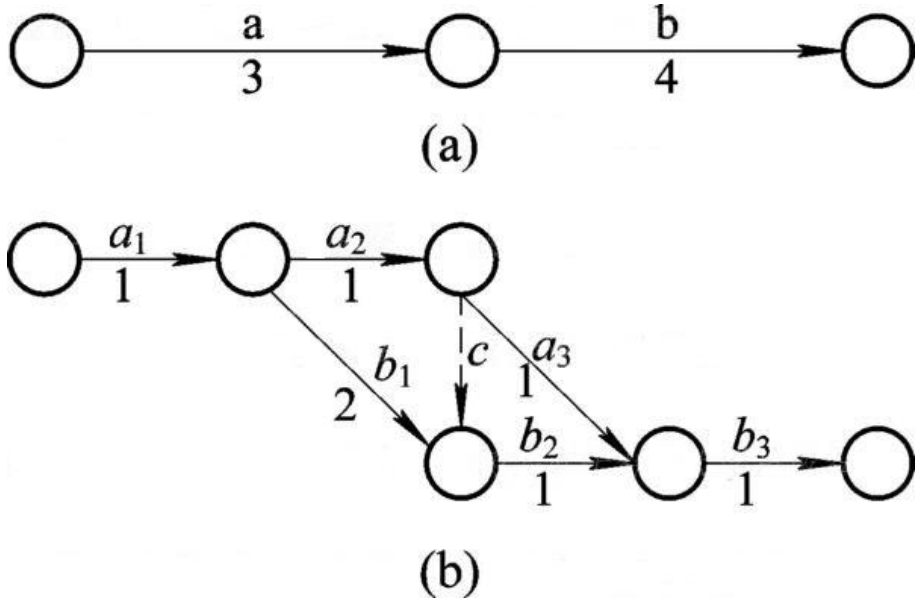


图6.7 交叉活动过程

计划网络图具有如下特性：

(1) 有向性和不可逆转性。这是由于计划网络图是一种有向无回路的赋权网络图，其每一条箭线表述一次活动，活动的完成是需要耗费时间的，而时间又是不可逆的，这就是计划网络图的有向性和不可逆转性。

(2) 连通性(连续性)。由于任何一次活动(或活动集合)总是从最初的事项(某一活动的开始事项)开始，依次经历过若干个活动及其事项，直到到达某一活动的终止事项为止，并标志着此项活动(或活动集合)的完成。这就是计划网络图的连通性(连续性)的内涵。显然，由于图的连通性的要求，因此计划网络图中不允许中断的活动或无关联的箭线和结点。

(3) 封闭性。一个计划网络图只允许有一个起始结点和一个终止结点，这就是计划网络图封闭性的含义。当计划网络出现多个起始结点或多个终止结点时，应引入虚活动(活动时间长度为0)，以保证该计划网络的封闭性。例如，图6.8(a)为非封闭的具有两个终止结点的计划网络，而经引入虚工序或合并输出结点(终止结点)时所得到的图6.8(b)和(c)所示的计划网络图则满足了封闭性的要求。

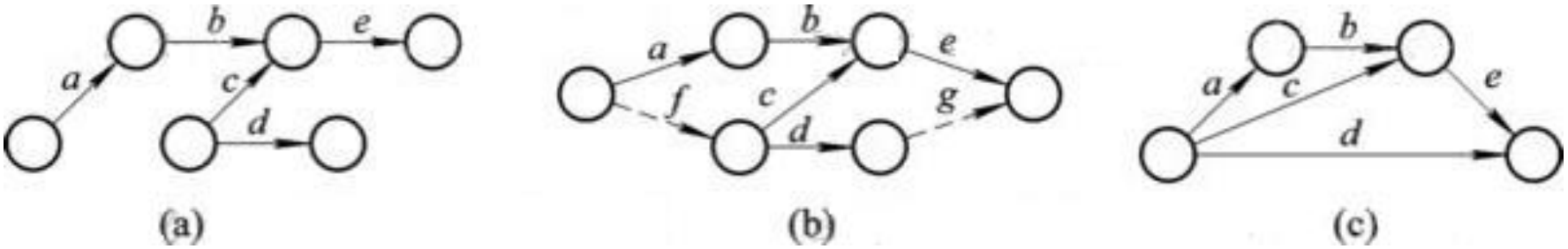


图6.8 封闭性示意图

3. 活动明细表与活动时长估计

表6.1所示的内容称为某软件项目的活动明细表，又称作业(任务、工序)明细表。该活动明细表一般有五列，分别表述了该软件项目所分解的各活动要素的编号，活动代号、活动内容、紧前(或紧后)活动、活动执行延续时间(简称活动时长)。表6.1各列记录了某软件项目各活动要素执行的逻辑顺序关系及各活动完成需耗费的时间(活动时长)。

表 6.1 活 动 明 细 表

编号	活动代号	活动内容	活动时长/(单位：月)	紧前活动
1	*a*	需求分析	60	—
2	*b*	文档	45	*a*
3	*c*	测试概要	10	*a*
4	*d*	概要设计	20	*a*
5	*e*	系统管理	40	*a*
6	*f*	测试准备	18	*c*
7	*g*	详细设计与编码Ⅰ	30	*d*
8	*h*	详细设计与编码Ⅱ	15	*d*、*e*
9	*i*	配置管理与质量保证	25	*g*
10	*j*	系统集成与测试	35	*b*、*i*、*f*、*h*

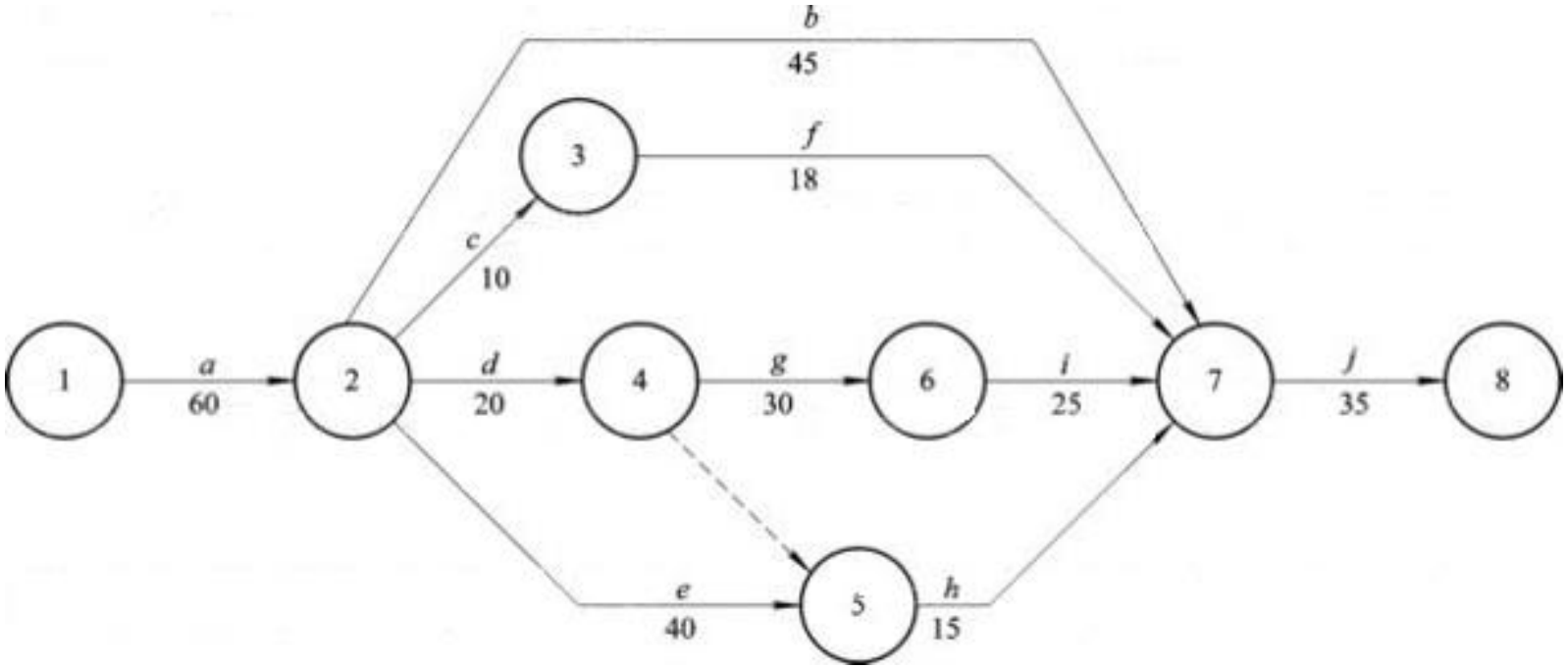


图6.13 计算网络图

一般来说，当一个中、大规模的软件项目要直接给出其对应的计划网络图往往是较为困难的，通常的做法是先给出其活动明细表(活动明细表中每项活动的紧前活动可用人工来做出判断)，然后再将活动明细表转换成计划网络图。这种转换工作可以由人工来完成，亦有专门的软件工具来自动或半自动地完成上述转换工作。

软件项目经工作任务分解后，各项目活动的延续时间或活动时长可通过如下两种方式之一来完成：

(1) 经验法(专家法)。它适用于不少软件项目的公共模块。开发人员曾多次实践过，不确定因素较少。如数据库模块、报表生成等办公自动化模块等，其模块时长估计可采用若干专家(或有经验人员)的经验，估计并取算术平均的方法来解决。即若设 T_e 表示某活动 e 的活动时长估值(单位：周、月或年)，t_j 表示第 j 个专家(或有经验人员)对活动 e 的时长估计值，则有

$$T_e = \frac{1}{n}\sum_{j=1}^{n} t_j \tag{6.1}$$

(2) 三点估计法。它适用于一些开发人员对其功能与性能或环境属性了解不多，或不确定因素较多的模块，此时可将活动 e 时长 T_e 视作服从 β 分布的随机变量，并用如下公式计算：

$$E(T_e)=\frac{a+4M+b}{6}\ ,\qquad \mathrm{var}(T_e)=\frac{(b-a)^2}{6} \tag{6.2}$$

其中，a 表示对活动 e 时长 T_e 的最乐观完成时间(顺利情况下活动 e 的完成时间)估计；b 表示对活动 e 时长 T_e 的最悲观完成时间(最不顺利情况下活动 e 的完成时间)估计；M 表示对活动 e 时长 T_e 的最可能时间(正常情况下活动e的完成时间)估计，并有$0\le a\le M\le b$。

4. 计划网络图生成流程

计划网络图可通过如图6.9所示的流程来完成。

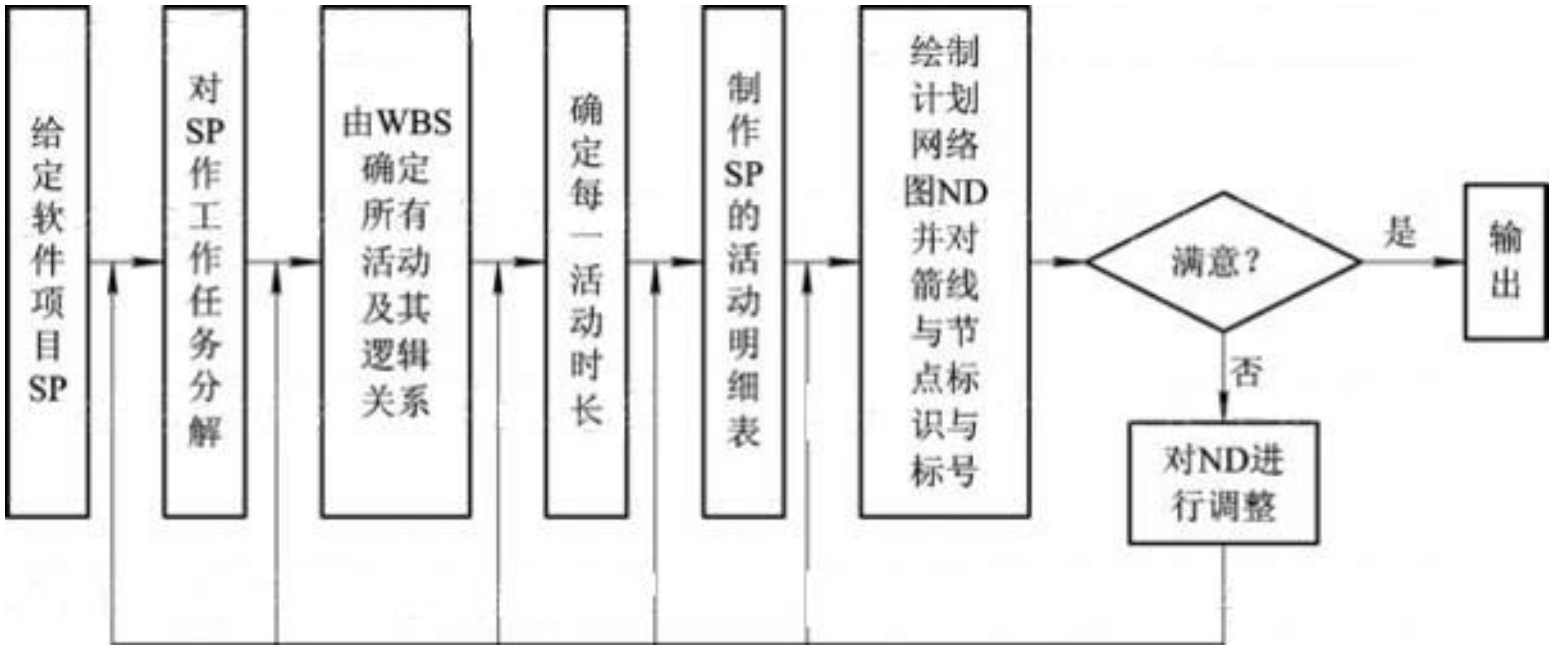


图6.9 计划网络图求解流程

6.1.3 进度计划与团队组织的工作流程

介绍软件项目为制订时间进度计划以及完成项目开发团队的组织与建设的基本流程，如图6.10所示。

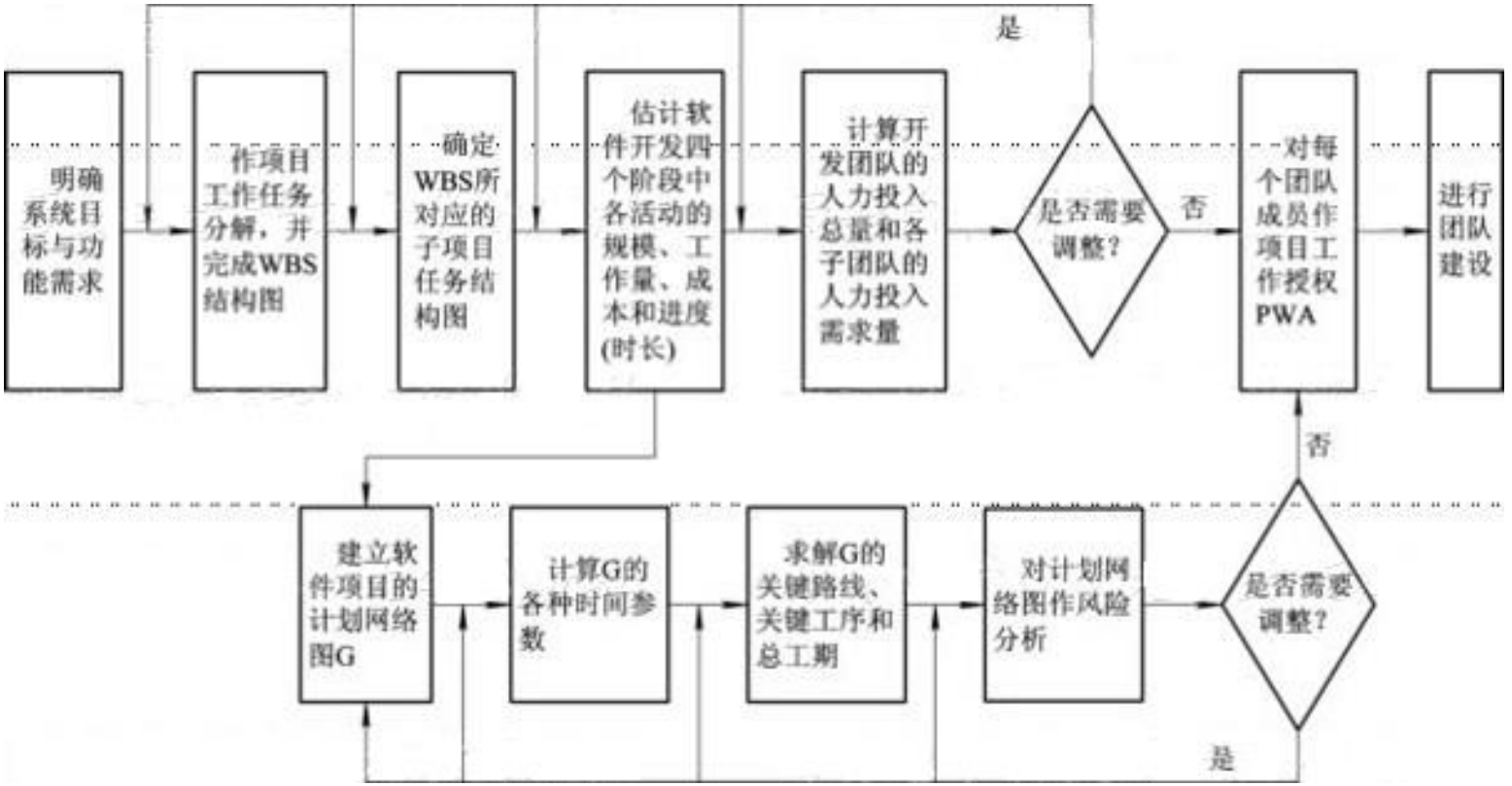


图6.10 进度计划与团队组织工作流程

由图6.10可知，在明确了软件项目的系统目标与功能、性能需求后，项目计划人员可以首先作工作任务分解，并完成对应于特定软件项目的WBS，并在此WBS的基础上完成此WBS对应的子项目任务结构图，从而获得该软件项目开发过程中的“最小单元”——活动及其具体内涵和进行活动规模、工作量、成本、进度的估计；

然后可作两个分支的工作，第一个分支的工作是计划网络图及其关键路线、关键工序、总工期的求解与系统风险分析，另一个分支的主要任务是根据特定软件项目的任务结构图中的各项活动的具体要求，计算进入软件项目开发团队及各种子团队的人员数量与工种；

最后在两个分支工作完成后，对开发团队及其成员进行项目工作授权和进一步考虑软件项目团队的建设问题。

本章6.2节将详细介绍图6.10中各单元的有关内容。

6.2 进度计划的分析与求解

求得软件项目的时间计划网络图只是求解软件项目进度计划的第一步，作为项目管理人员，为了更好地对该软件项目的开发进程进行管理与控制，还需要解决如下五个问题：

(1) 确定每个活动的开始时间和结束(完成)时间，且这样的活动开始时间与结束(完成)时间不应是硬性规定的，应允许其有一定的机动余地。

(2) 在开发方已有的资源投入下求解该软件项目的交付日期(工期)，或给定工期(投资方要求)条件下来安排各活动的开始时间和结束时间。

(3) 为了完成工期T_d目标，在整个软件项目开发过程中哪些活动是关键的？此中所谓关键活动，是指由于这些活动完成的耽误或更改，将直接影响项目工期目标的完成。

(4) 由于开发过程中各相关活动是延续进行的，因此前一活动的耽误必将影响紧后活动的完成，从而构成了一条关键线路。显然关键线路及其中的每一关键活动是项目管理人员管理与控制的重点。

(5) 对于给定的工期目标T_d，在已有项目各活动时长的条件下能否顺利完成的风险分析。

6.2.1 进度计划中关键线路的分析与求解

以下介绍计划网络图中的时间参数及其关联，关键活动与关键线路求解的基本理论以及软件项目关键线路求解的应用案例。所采用的方法称为规划评审技术(Program Evaluation and Review Technique，PERT)与关键线路法(Critical Path Method，CPM)。

1. 时间参数及其关联

由图6.4可知，计划网络图的任一箭线表述活动，而箭线的左右各有一个结点分别表示此活动的开始事项和结束事项(如图6.11所示)，故此活动又可用(*i*，*j*)来表示，其中*i*为箭线(活动)的起始事项(结点*i*)，*j*为箭线(活动)的终止或结束事项(结点*j*)。

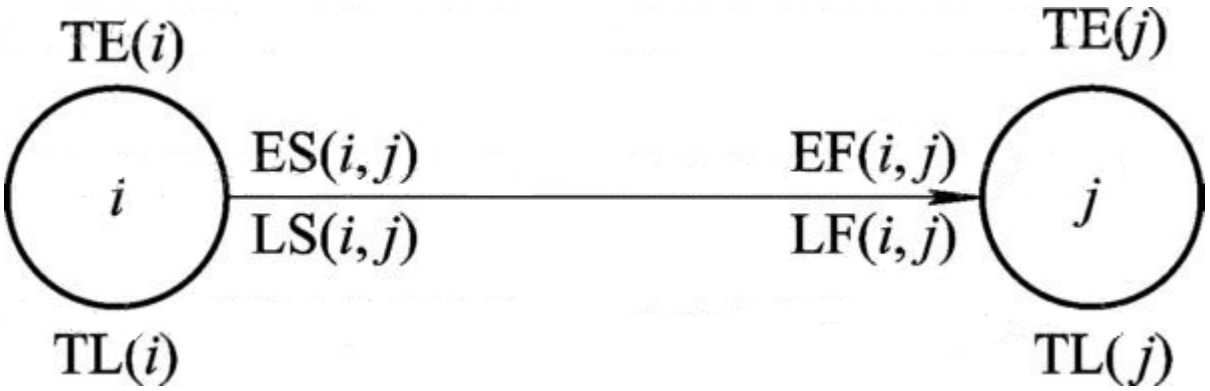


图6.11　时间参数关联图

活动(*i*，*j*)的8个时间参数，它们的标示符与对应的物理含义如下：

ES(*i*，*j*)表示活动(*i*，*j*)的最早开始时间(Earliest Start，ES)；

EF(*i*，*j*)表示活动(*i*，*j*)的最早完成时间(Earliest Finish，EF)；

LS(*i*，*j*)表示活动(*i*，*j*)的最晚开始时间(Lastest Start，LS)；

LF(*i*，*j*)表示活动(*i*，*j*)的最晚完成时间(Lastest Finish，LF)；

TE(*i*)表示结点*i*的最早开始时间；

TL(*i*)表示结点*i*的最晚完成时间；

R(*i*，*j*)表示活动(*i*，*j*)的时差(反映活动(*i*，*j*)的机动时间)；

R(*i*)表示结点*i*的时差(反映结点的机动时间)。

若活动$(i，j)$的实际开始时刻为t_S，实际完成时刻为t_F，活动$(i，j)$时长为$t(i，j)$，则显然有

$$\mathrm{ES}(i,j)\le t_S\le \mathrm{LS}(i,j)，\ t_S+t(i,j)=t_F，\ \mathrm{EF}(i,j)\le t_F\le \mathrm{LF}(i,j)$$

上述各时间参数的物理内涵,容易证明它们有如下数量关系：

$$\begin{cases}(1)\ \mathrm{TE}(i)=\mathrm{ES}(i，j)\\ (2)\ \mathrm{TL}(i)=\min\limits_{j}\{\mathrm{TL}(j)-t(i，j)\}=\min\limits_{j}\mathrm{LS}(i，j)\\ (3)\ \mathrm{TE}(j)=\max\limits_{i}\{\mathrm{TE}(i)+t(i，j)\}=\max\limits_{i}\mathrm{EF}(i，j)\\ (4)\ \mathrm{TL}(j)=\mathrm{LF}(i，j)\\ (5)\ \mathrm{EF}(i，j)=\mathrm{ES}(i，j)+t(i，j)=\mathrm{TE}(i)+t(i，j)\\ (6)\ \mathrm{LF}(i，j)=\mathrm{LS}(i，j)+t(i，j)=\mathrm{TL}(j)\end{cases}\tag{6.3}$$

$$\begin{cases} (7)\ R(i，j) = \mathrm{LS}(i，j) - \mathrm{ES}(i，j) = \mathrm{LF}(i，j) - \mathrm{EF}(i，j) \\ (8)\ R(i) = \mathrm{TL}(i) - \mathrm{TE}(i) \end{cases} \tag{6.4}$$

需要说明的是：所谓一个结点i的最早开始时间TE(i)，是指它所代表的开始事项最早可在什么时刻进行，或者说以i为起始结点的后续活动(i，j)最早可在什么时刻开始；所谓一个结点 i 的最晚完成时间TL(i)，是指无论该结点 i 有多少项与其关联的先行活动，都必须在该时刻 i 之前完工，否则的话，必然影响结点 i 有关联的后行活动的按期开工。对于一个起始结点为1，终止结点为 n 的计划网络图，为计算方便起见，可定义TE(1)=0，TL(n)=T_d 或 T_d 待求。

2. 关键线路、关键活动及其特性

在一个给定的计划网络图*G*中，由于只允许有一个起点和一个终止结点，故在*G*中从起始结点开始经过一系列活动直到终止结点为止的一个结点、边(箭线)序列称为*G*的一条通路或路线*P*。*P*中所经由的各活动时长之和称为该通路的路长。在*G*的所有通路中，路长最大时对应的通路即称为关键路线或关键通路(Critical Path)。在关键路线上所途经的活动称为关键活动(Critical Activity)。

由上述关键路线的定义易知，在计划网络图中，关键路线是影响能否达到工期目标的关键部分，是项目管理与控制的重点对象。在较为简单的计划网络图中，人们可以通过所有通路的穷举并作路长比较来求解关键路线和关键活动，然而在一个复杂的计划网络图中，采用穷举法显然是不可行的，这就需要通过活动时差或结点时差的方法来求解关键活动或关键通路。利用活动时差和结点时差的(6.4)式定义，容易证得如下几个性质：

(1) 设G为计划网络图，通路CP为G的关键路线的充分必要条件为：对G上CP的任何活动$(i，j)$有$R(i，j)=0$或有

$$\sum_{(i,j)\in CP} R(i,j)=0 \tag{6.5}$$

(2) 设G为计划网络图，通路CP为G的关键路线，则对G上CP的任何结点i有$R(i)=0$ 或有

$$\sum_{i \in CP} R(i) = 0 \tag{6.6}$$

(3) 若G为有限计划网络图，则G至少有一条关键路线，至多有有限条关键路线。

(4) 计划网络图中所有关键活动时长的总和即为该计划网络图G的总工期T_d，或有

$$T_d = \sum_{(i,j) \in CP} t(i, j) \tag{6.7}$$

事实上，各活动的时差或单时差、自由时差(Free Float)反映了该活动实施时的机动时间。时差为零，说明该活动不可能提前完成，因而该活动必为影响*G*工期完成的关键环节；若时差不为零，说明该活动有提前完成的机动余地，也说明了该活动并非是影响*G*工期完成的关键环节。由于在关键路线*CP*上的各活动时差为0，这就导致这些活动的起始事项与终止事项无机动余地，从而有*CP*上各结点的时差亦为0。由此可知，结点时差为零时对应通路是关键路线的必要条件(注意：但并非充分条件，因为非关键路线上的结点亦可能出现时差为零的状况)。此外，由定义知：关键路线*CP*是*G*中所有通路中路长最大的一条，故只要关键路线上的各关键活动实施完毕，则其他非关键活动必然应在关键活动结束前实施完成，这就是性质(4)或(6.7)式的由来。

3. 关键路线求解

由上述各时间参数的定义及关键路线、关键活动的特性可知，只须在给定的计划网络图G中首先计算G中各活动、结点的时间参数ES(i，j)、EF(i，j)、LS(i，j)、LF(i，j)、TE(j)、TL(i)，然后在此基础上计算各活动的时差R(i，j)，最后通过性质(1)或(6.5)式即可寻找出关键路线和关键工序。当然，为了使上述求解过程更为准确起见，在所寻找出来的关键路线上可再计算其各结点的时差，并以此来验证上述求解的正确性。

图6.12给出了一种被称为“标号算法”的关键路线求解算法，该算法包括三部分：第I部分为正向(自左向右)求解计划网络图的结点与活动的最早时间参数，其算法见(6.8)式，第II部分为逆向(自右向左)求解计划网络图的结点与活动的最晚时间参数，其算法见(6.9)式，第III部分为计算活动与结点的时差，算法见(6.10)式，并由性质(1)及(6.5)式来寻找关键路线和关键结点。(6.8)、(6.9)和(6.10)式实际上是(6.3)式算法的分解，读者不难从中验证。

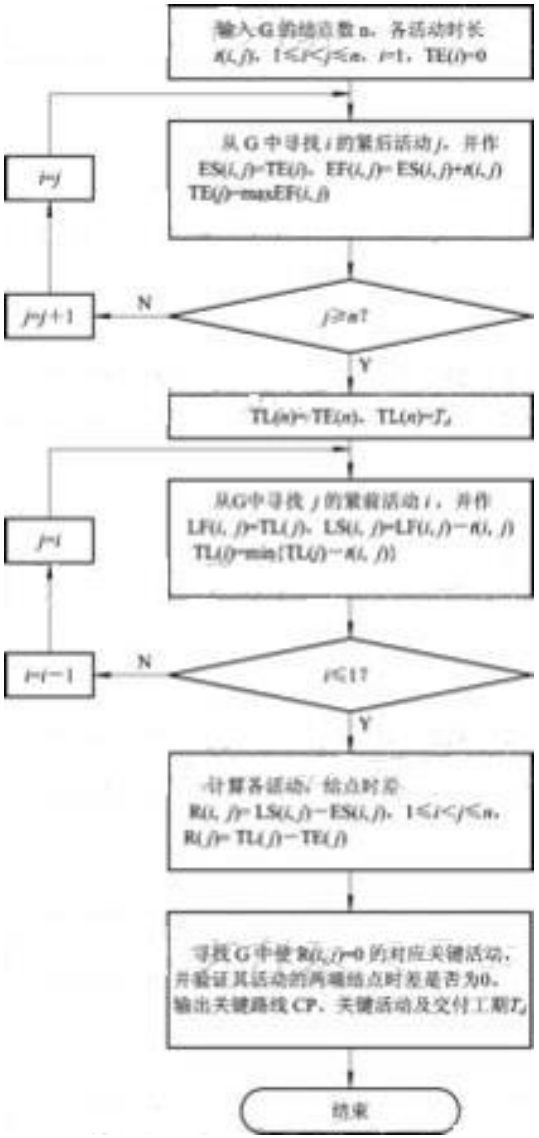


图6.12　标号算法流程图

$$\begin{cases} \mathrm{TE}(1)=0 \\ \mathrm{ES}(i，j)=\mathrm{TE}(i) \\ \mathrm{EF}(i，j)=\mathrm{ES}(i，j)+t(i，j) \end{cases} \tag{6.8}$$

$$\begin{cases} \mathrm{TE}(j)=\max\limits_{i}\mathrm{EF}(i，j) \\ \mathrm{TE}(n)=\mathrm{TL}(n)=T_d \\ \mathrm{LF}(i，j)=\mathrm{TL}(j) \\ \mathrm{LS}(i，j)=\min\limits_{j}\mathrm{LF}(i，j)-t(i，j) \end{cases} \tag{6.9}$$

$$\begin{cases} \mathrm{TL}(i)=\min\mathrm{LS}(i，j) \\ \mathrm{R}(i，j)=\mathrm{LS}(i，j)-\mathrm{ES}(i，j) \\ \mathrm{R}(i)=\mathrm{TL}(i)-\mathrm{TE}(i) \end{cases} \tag{6.10}$$

[例6.1] 已知某软件项目经工作任务分解后，得到活动明细表如表6.1所示。

(1) 绘出表6.1对应的计划网络图G；

表 6.1 活 动 明 细 表

编号	活动代号	活动内容	活动时长/(单位：月)	紧前活动
1	*a*	需求分析	60	—
2	*b*	文档	45	*a*
3	*c*	测试概要	10	*a*
4	*d*	概要设计	20	*a*
5	*e*	系统管理	40	*a*
6	*f*	测试准备	18	*c*
7	*g*	详细设计与编码Ⅰ	30	*d*
8	*h*	详细设计与编码Ⅱ	15	*d*、*e*
9	*i*	配置管理与质量保证	25	*g*
10	*j*	系统集成与测试	35	*b*、*i*、*f*、*h*

(2) 求解计划网络图G的关键路线CP和关键活动，并给出在给定活动明细表6.1状况下的软件项目交付工期T_d。

解 (1) 由该软件项目的明细表中所给出的各活动逻辑关系，容易绘制出该软件项目的计划网络图G如图6.13所示。此时为作以后的关键路线求解准备，可在G中每一箭线(活动)上方注明该活动的代号，在箭线的下方注明该活动所耗费的时长。显然，活动时长可以在活动明细表6.1中得到。亦即在对该软件项目作工作任务分解后，运用前述的经验法或三点估计法得到。

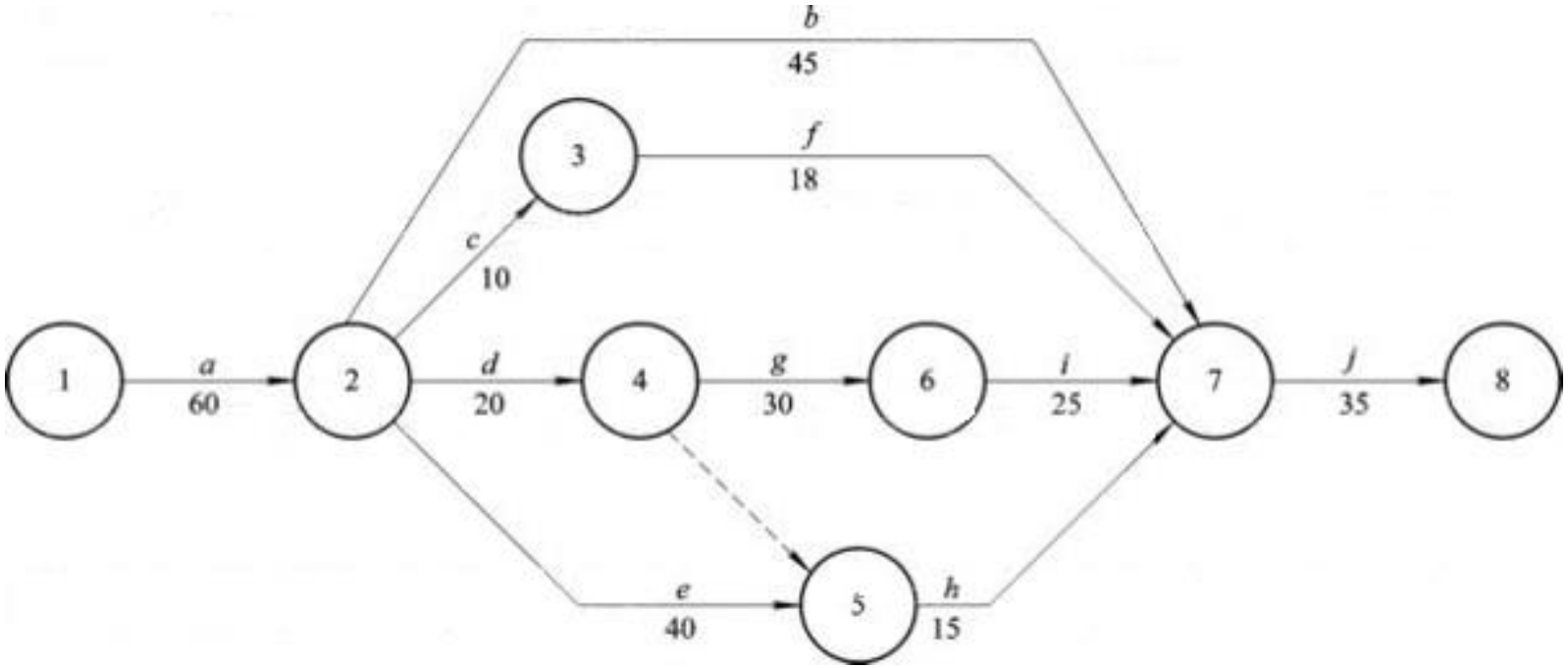


图6.13 计算网络图

(2) 利用图6.12所示的算法流程，可以求得计划网络图G的关键路线、关键工序和交付工期。图6.14给出了求解过程中正向计算所得到的G中各活动、结点的最早时间参数，图6.15给出了求解过程中逆向计算所得的G中各活动、结点的最晚时间参数。

需要说明的是，在正向计算过程中，每得到一个活动的最早开始时间ES(*i*，*j*)和最早完成时间EF(*i*，*j*)，就用形如[ES(*i*，*j*)，EF(*i*，*j*)]的形式标注在箭线上方，同样结点的最早开始时间TE(*i*)亦写在结点的上方；

类似地，在逆向计算过程中，每得到一个活动的最晚开始时间LS(*i*，*j*)和最晚完成时间LF(*i*，*j*)，就用形如[LS(*i*，*j*)、LF(*i*，*j*)]的形式标注在箭线(活动)的下方，而得到的结点最晚完成时间TL(*j*)亦写在结点的下方。

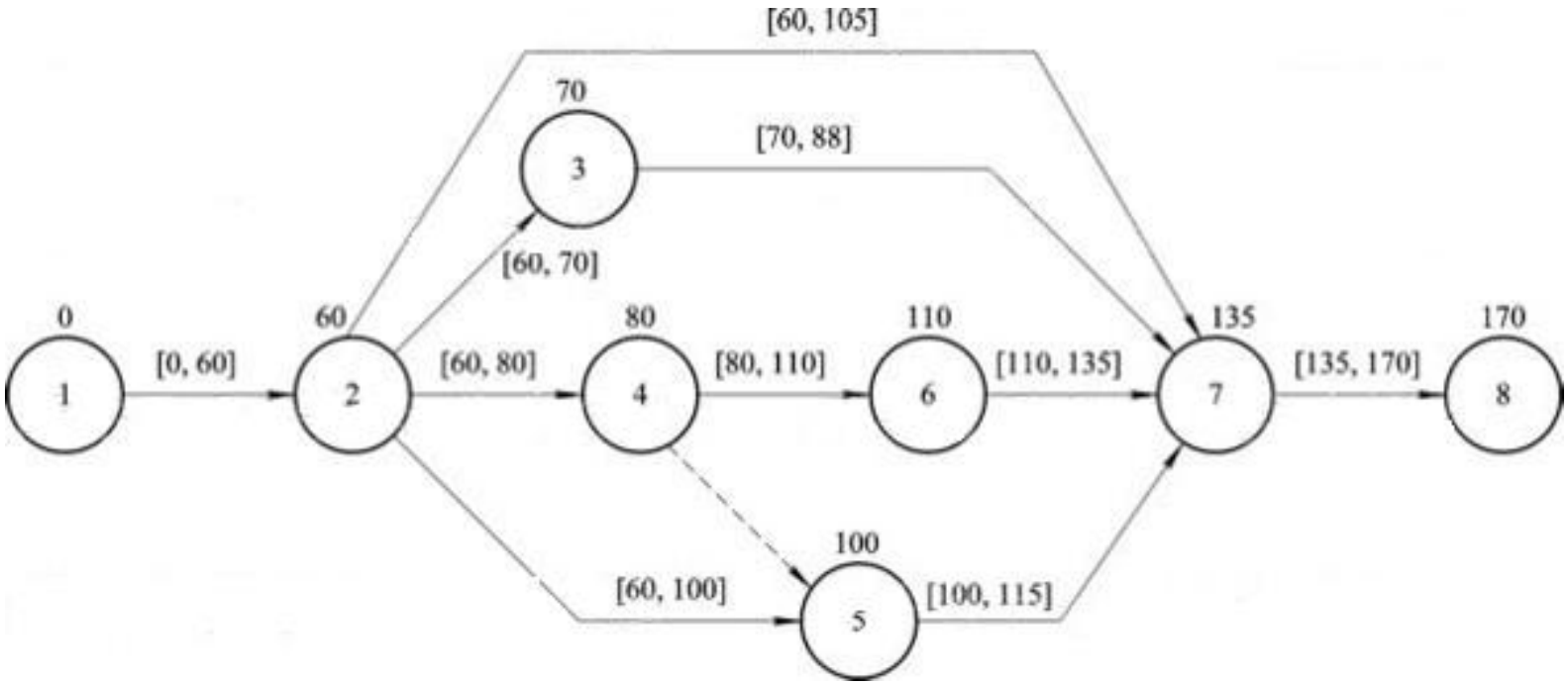

[60, 105]
70
[70, 88]
3
[60, 70]
0
60
80
110
135
170
1
[0, 60]
2
[60, 80]
4
[80, 110]
6
[110, 135]
7
[135, 170]
8
100
[60, 100]
[100, 115]
5

图6.14 正向计算过程

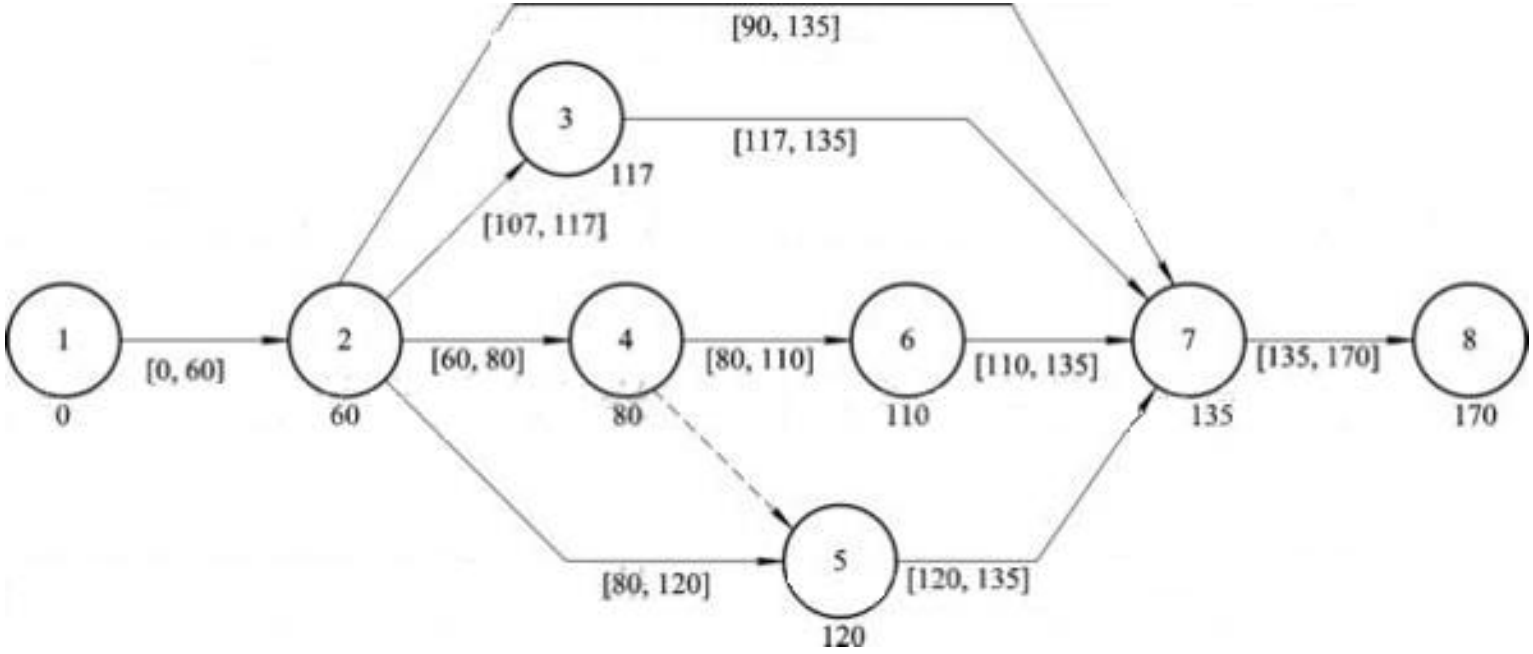

[90, 135]
3
117
[117, 135]
[107, 117]
1
[0, 60]
2
[60, 80]
4
[80, 110]
6
[110, 135]
7
[135, 170]
8
0
60
80
110
135
170
[80, 120]
5
[120, 135]
120

图6.15 逆向计算过程

例如根据算法(6.8)式可求得如下最早时间参数：

TE(1)=0，ES(1，2)=0，EF(1，2)=60

$TE(2)=\max_{i} EF(i，2)=EF(1，2)=60$

ES(2，3)=TE(2)=60，EF(2，3)= ES(2，3)+t(2，3)=60+10=70

…

$TE(7)=\max_{i} EF(i，7)=\max\{EF(2，7)，EF(3，7)，EF(6，7)，EF(5，7)\}$

=max{105，88，135，115}=135

同样，根据算法(6.9)式可求得如下最晚时间参数：

$TL(8)=TE(8)=170$，$LF(7，8)=TL(8)=170$

$LS(7，8)=LF(7，8)-t(7，8)=170-35=135$

$TL(7)=\min\limits_{j} LS(7，j)=LS(7，8)=135$

$LF(2，7)=TL(7)=135$，$LS(2，7)=LF(2，7)-t(2，7)=135-45=90$

…

$TL(2)=\min\limits_{j} LS(2，j)=\min\{LS(2，7)，LS(2，3)，LS(2，4)，LS(2，5)\}$

$=\min\{90，107，60，80\}=60$

表6.2给出了计划网络图G根据流程图6.12求得的各活动的时间参数。表6.3给出了求得的各结点时间参数，由(6.10)式容易求得各活动时差和各结点时差分别列在表6.2的第六列和表6.3的第四列。观察表6.2各活动时差，可以得到计划网络图G的关键路线为$a\rightarrow d\rightarrow g\rightarrow i\rightarrow j$，关键活动为{$a$，$d$，$g$，$i$，$j$}，上述各活动为关键活动的事实还可通过观察表6.3中各对应结点时差为0而得到证实。

表 6.2 活动时间参数表

活动	ES	LS	EF	LF	$R(i, j)$	关键活动
a	0	0	60	60	0	√
b	60	90	105	135	30	×
c	60	107	70	117	47	×
d	60	60	80	80	0	√
e	60	80	100	120	20	×
f	70	117	88	135	47	×
g	80	80	110	110	0	√
h	100	120	115	135	20	×
i	110	110	135	135	0	√
j	135	135	170	170	0	√

注：“√”表示是；“×”表示否。

表 6.3 结点时间参数表

结点 i	TE(i)	TL(i)	$R(i)$	关键活动端点
1	0	0	0	√
2	60	60	0	√
3	70	117	47	×
4	80	80	0	√
5	100	120	20	×
6	110	110	0	√
7	135	135	0	√
8	170	170	0	√

注：“√”表示是；“×”表示否。

需要说明的是，上述时间参数的求解过程对于小规模的软件项目完全可用手工在计划网络图上进行作业(故上述标号算法又可称为图上作业法)。其计算过程分为三个阶段，第一阶段为正向计算，亦即依据结点的自左向右顺序逐个进行结点时间参数计算→活动时间参数计算→结点时间参数计算，当一个结点的紧后活动有多个时，其活动时间参数可按照自上而下的顺序逐个进行活动时间参数进行，每得到一个时间参数，可标注在箭线与结点上方(如图6.14所示)。当最后一个结点最早开始时间得到后即进行第二阶段的逆向计算过程，逆向计算其计算过程与第一阶段相仿，只是对结点的计算是依自右向左的方向进行，而所得到的时间参数应标注在箭线的下方和结点的下方；对于第三阶段关键活动的判断只须观察G中每一箭线(活动)的上方时间参数和下方时间参数即可，当箭线的上方时间参数与下方时间参数完全相同时，显然此时该箭线所对应的活动时差必为0，从而该活动为关键活动；反之，当上、下方时间参数不同时，该箭线对应的活动就不是关键活动，至于结点时差是否为0的判断亦只须观察结点上、下方时间参数是否相同即可，上、下方时间参数相同，说明该结点时差为0。

6.2.2 进度计划的风险分析与网络优化

对于一个给定的软件项目，当依据前述的工作原理，绘制了计划网络图和完成了各活动的时间参数求解和关键活动、关键路线求解后，即可得到该软件项目的时间进度计划方案如表6.4所示。需要指出的是，时间进度计划方案只是一个初步的方案，该方案在项目开发的过程中能否按照预定的轨迹前进并达到工期目标还是未知的，因为它还将受到各种未来的不确定因素的影响。这些不确定性因素有系统调查时对用户目标、功能与性能要求的错误理解或不充分的理解，系统开发时对某些功能模块的困难程度认识不足，开发人员的变动与开发效率的不确定性，用户对系统目标、功能与性能、工期的变更要求，即将购置的开发工具、测试设备性能的不确定性，以及开发人员对各项目任务工时估计的不确定性等。而这些不确定性因素随着软件规模的增大将越来越多，因此对于一个给定的中、大型软件项目，研究开发机构按工期完工的可能性以及给定软件项目执行进度计划的难度等问题是必要的。

表 6.4 项目进度计划方案

序号	活动 i	功能、性能、要求	ES(i)	EF(i)	LS(i)	LF(i)	负责人
1	a	…	…	…	…	…	张三
2	b	…	…	…	…	…	李四
3	c	…	…	…	…	…	…
…	…						
n	k	…	…	…	…	…	…

1. 大、中型软件项目总工期的概率特性

对于一个给定的大、中型软件项目，由以上分析可知，影响开发过程能否按预期进度计划实施的影响因素十分众多，且工作任务分解下的底层活动数量 n 也很大，我们可粗略地认为这些影响因素虽然数量众多，但彼此相互独立且对总工期的影响均匀地微小。于是，由概率论的中心极限定理可知，总工期 T_d 服从正态分布，并有数学期望和方差如下：

$$\left.\begin{aligned} E(T_d) &= \sum_{(i,j)\in CP} E(t(i,j)) = \sum_{(i,j)\in CP} \frac{a_{ij} + 4m_{ij} + b_{ij}}{6} = \mu_{CP} \\ \mathrm{var}(T_d) &= \sum_{(i,j)\in CP} \mathrm{var}(t(i,j)) = \sum_{(i,j)\in CP} \left(\frac{b_{ij} - a_{ij}}{6}\right)^2 = \sigma_{CP}^2 \end{aligned}\right\} \tag{6.11}$$

从而有$T_d \sim N(\mu_{CP}, \sigma_{CP}^2)$。其中，$CP$为给定的一个大、中型软件项目计划网络图$G$的关键路线，$a_{ij}$、$m_{ij}$、$b_{ij}$分别为$G$的活动$(i, j)$时长的最乐观完成时间、最可能完成时间、最悲观完成时间。利用上述结论，可以推断给定一个工期目标T_0，该软件项目的计划网络完工的概率为

$$P_r(T_d \le T_0) = P_r\left(\frac{T_d - \mu_{CP}}{\sigma_{CP}} \le \frac{T_0 - \mu_{CP}}{\sigma_{CP}}\right) = P_r\left(\eta \le \frac{T_0 - \mu_{CP}}{\sigma_{CP}}\right) = \Phi\left(\frac{T_0 - \mu_{CP}}{\sigma_{CP}}\right) \tag{6.12}$$

其中由于 $\eta = \frac{T_d - \mu_{CP}}{\sigma_{CP}}$ 服从标准正态分布，故$\Phi(\cdot)$即为标准正态分布表中的拉普拉斯函数(Laplace)。(6.12)式的现实意义在于：对于一个大、中型软件项目，在求得了该项目计划网络图中关键路线CP及其关键活动时长的最乐观完成时间、最可能完成时间和最悲观完成时间的估计值，则对于任何一个给定的工期目标要求(例如用户或投资方提出的要求)T_0，可以计算按T_0要求完工的可能性，并且开发机构还可以据此来决定是否同意用户要求。

[例6.2] 某软件项目经工作任务分解后，给出了活动明细表如表6.5所示。

(1) 绘制对应计划网络图G，求解G的关键路线CP和期望总工期μ_{CP}。

(2) 给定总工期目标26月，计算能按此总工期目标完工的概率。

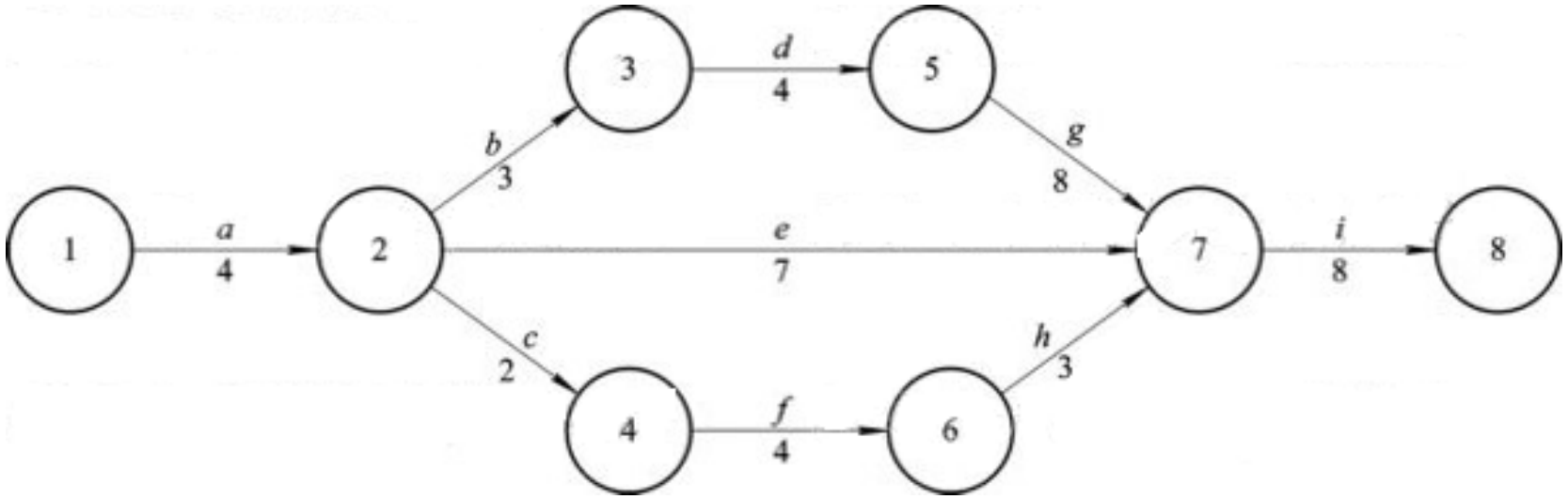
(3) 若要求该计划网络G完工的可能性达到95%，则应确定此软件项目的总工期T_d为多少月。

表 6.5 某软件项目活动明细表 单位：月

活动代号	a_{ij}	m_{ij}	b_{ij}	μ_{ij}	σ_{ij}^2	紧后活动	关键活动
a	3	4	5	4	1/9	b，c，e	√
b	2	3	4	3	1/9	d	√
c	1	2	3	2	—	f	×
d	2	4	6	4	4/9	g	√
e	3	7	11	7	—	i	×
f	3	4	5	4	—	h	×
g	6	7	14	8	16/9	i	√
h	2	3	4	3	—	i	×
i	6	7	14	8	16/9	—	√

注：“√”表示是；“×”表示否。

解 (1) 由活动明细表容易绘制对应的计划网络图G如图6.16所示。



根据表6.5中各活动时长的期望值μ_{ij}，$(i，j)\in G$，容易求解G的关键路线$CP=(1，2，3，5，7，8)$，此向量中的各分量代表CP所历经的结点标号。由此容易求得G的期望总工期与方差有

$$\begin{aligned}\mu_{CP} &= \sum_{(i,j)\in CP} E(t(i,j)) = 4+3+4+8+8 = 27 \quad \text{月} \\ \sigma_{CP}^2 &= \sum_{(i,j)\in CP} \sigma_{ij}^2 = \frac{1}{9}+\frac{1}{9}+\frac{4}{9}+\frac{16}{9}+\frac{16}{9} = \frac{38}{9} \quad \text{月}^2\end{aligned} \tag{6.13}$$

因此 $\sigma_{CP} = 2.055$

(2) 对于给定的总工期目标T_0=26，其按此工期目标完工的概率有

$$P_r(T_d \le T_0) = P_r(T_d \le 26) = \Phi(\frac{T_0 - \mu_{CP}}{\sigma_{CP}}) = \Phi(\frac{26-27}{2.055}) = 0.314 = 31.4\%$$

(3) 欲使$P_r(T_d \le T_0)$=0.95，或即有 $\Phi(\frac{T_0 - 27}{2.055}) = 0.95$。查$N(0，1)$表知有

$$\frac{T_0 - 27}{2.055} = 1.64$$

从而有

T_0=27+1.64×2.055=30.37月=2.53年

亦即为使该计划网络G能按期完工的概率达到95%，则应确定G的总工期为30.37月或2.53年。

由软件项目总工期的概率特性(6.12)式，还可推出下述两个性质：

(1) 设计划网络G的工期为T_d，$E(T_d)=\mu_{CP}$，则对任意$x\geq 0$，有

$$F(x)=\Pr(T_d\leq x)=\begin{cases}>0.5, & x>\mu_{CP}\\ =0.5, & x=\mu_{CP}\\ <0.5, & x<\mu_{CP}\end{cases} \qquad (6.14)$$

事实上，由分布函数$F(x)$的单调不减性和$F(\mu_{CP})=\Phi(\frac{\mu_{CP}-\mu_{CP}}{\sigma_{CP}})$ $=\Phi(0)=0.5$，容易得到(6.14)式之结论。

(2) 若计划网络G有两条关键路线CP_1和CP_2，执行CP_1和CP_2时的工期分为T_1和T_2，并有

$$E(T_1)=\mu CP_1=E(T_2)=\mu_{CP2},\ \mathrm{var}(T_1)=\sigma^2_{CP1}>\sigma^2_{CP2}=\mathrm{var}(T_2) \qquad (6.15)$$

则对任何工期目标$T_0\geq 0$，有

$$P_r(T_1\leq T_0)\leq P_r(T_2\leq T_0) \qquad (6.16)$$

事实上，由题设知有$\mu_{CP1}=\mu_{CP2}$，$\sigma^2_{CP1}>\sigma^2_{CP2}$，从而有$\frac{T_0-\mu_{CP1}}{\sigma_{CP1}}<\frac{T_0-\mu_{CP2}}{\sigma_{CP2}}$。再注意到Laplace函数$\Phi(x)$的单调不减性，容易证得(6.16)式结论。

上述两个性质的现实意义在于：性质(1)给了我们一个拒绝工期目标的基本准则，若给定工期目标小于μ_{CP}，则开发机构应予拒绝；反之，则可以考虑有无条件接纳此目标要求。性质(2)告诉我们，若计划网络G有两条关键路线时，管理人员只须跟踪方差大的那条关键路线上的活动执行状况，因为只要方差较大的那条关键路线上的关键活动能顺利执行完成时，其保证软件项目能按期完工的可能性就大。此外，性质(2)对于G有m条($m\geq 2$)关键路线的情况下仍然成立，读者可自行给出结论。

2. 关键路线与计划难度系数

设有一个计划网络 G，其关键路线为 CP，CP 路长的期望和方差分为 μ_{CP} 和 σ^2_{CP}。对于一个给定的工期目标T_0，可以定义一个计划难度系数 δ_0 来度量G执行给定工期目标 T_0 的难易程度，即

$$\delta_0 = 2 \cdot \frac{T_0 - \mu_{CP}}{\sigma_{CP}} \tag{6.17}$$

对于上述定义的计划难度系数δ_0，容易得到如下性质：

$$P(T \le T_0) = \Phi\left(\frac{T_0 - \mu_{CP}}{\sigma_{CP}}\right) = \Phi(\frac{\delta_0}{2}) = \begin{cases} 0 & \delta_0 \in (-\infty, -6) = \Delta_1 \\ \in (0, \quad 0.308), & \delta_0 \in (-6, -1) = \Delta_2 \\ \in (0.308, \quad 0.692), & \delta_0 \in (-1, 1) = \Delta_3 \\ \in (0.692, \quad 1), & \delta_0 \in (1, 6) = \Delta_4 \\ 1 & \delta_0 \in (6, +\infty) = \Delta_5 \end{cases} \tag{6.18}$$

事实上，由于$P(T\leq T_0)=\Phi\left(\frac{\delta_0}{2}\right)$，故有

当$\delta_0=-6$时，有$\Phi\left(\frac{\delta_0}{2}\right)=\Phi(-3)=0$。

当$\delta_0=-1$时，$\Phi\left(\frac{\delta_0}{2}\right)=\Phi\left(-\frac{1}{2}\right)=0.308$。

当$\delta_0=1$时，$\Phi\left(\frac{\delta_0}{2}\right)=\Phi\left(-\frac{1}{2}\right)=0.692$。

当$\delta_0=6$时，$\Phi\left(\frac{\delta_0}{2}\right)=\Phi(3)\approx 1$。

再由分布函数$\Phi\left(\frac{\delta_0}{2}\right)$的单调不减性(详见图6.17)容易得到(6.18)式的结论。当计划难度系数δ_0在Δ_1区域内时，由(6.18)式知执行工期T_0的计划根本不能完成；当δ_0在Δ_2区域内时，(6.18)式说明执行工期T_0的计划完成的可能性较小；当δ_0落在Δ_3区域内时，(6.18)式说明执行工期T_0的计划完成可能性较大；当δ_0落在Δ_4区域内时，(6.18)式说明执行工期T_0的计划完成的可能性很大；当δ_0落在Δ_5区域内时，(6.18)式说明执行工期T_0的计划有压缩工期的可能。

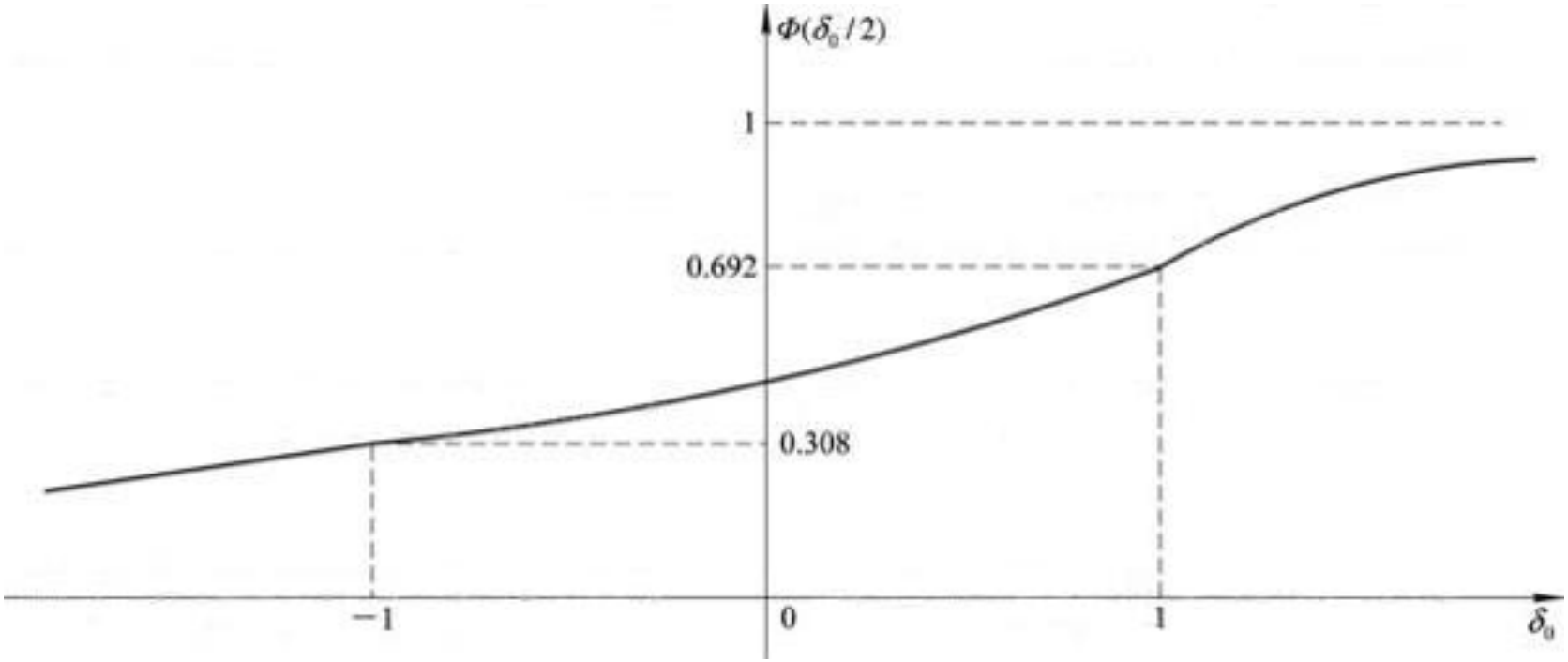


图6.17 $\Phi\left(\frac{\delta_0}{2}\right)$ 曲线

[例6.3] 对于例6.2的计划网络图G，若工期T_0分别取23月、25月、27月、29月、31月，求该软件项目能按期完工的可能性。

解 由(6.17)式知，计划难度系数有

$$\delta_0 = 2 \cdot \frac{T_0 - \mu_{CP}}{\sigma_{CP}} = 2 \cdot \frac{T_0 - 27}{2.055}$$

以题设T_0的五个值分别代入上式可得结果如表6.6所示。由此表中的数据可知，当T_0取23月或25月时，该软件项目能按期完工的概率小；当T_0取27月时，该软件项目能按期完工的概率较大；当T_0取29月或31月时，该软件项目能按期完工的概率大。

表 6.6 计划难度系数表

T_0	23	25	27	29	31
δ_0	−3.892	−1.946	0	1.946	3.892
δ_0 所属的区间	Δ_2	Δ_2	Δ_3	Δ_4	Δ_4

3. 网络优化

对于一个给定的计划网络G，它是否就是一个在时间进度、资源投入、成本耗费等方面最佳的方案？若不是，又如何来调整与优化此计划网络G？ 解决上述问题的研究称为网络优化问题。

网络优化问题从数学上来看，可分为单目标优化(如时间优化、资源优化、成本或费用优化)和多目标优化两大类问题，其中多目标优化又有时间—资源优化问题，时间—费用优化问题和时间—资源—费用优化问题等。限于篇幅，以下仅介绍时间优化问题的一般思想。

计划网络G的时间优化的一般原理有：① 向关键路线上要时间，其采取的措施包括将串行活动在允许的情况下，调整为并行活动；使用先进的软件开发工具；调用经验丰富的技术人员；采用性能较高的计算机(服务器)等等。② 将富裕路线(非关键路线)上的资源(主要是人力资源)调整到关键路线上来。③ 从项目团队外增拨资源(开发工具、测试设备、开发人员、测试人员)来加强团队，提高开发效率。④ 优化活动间的逻辑顺序关系，以缩短交付工期。⑤ 通过对所建立的单目标或多目标规划模型，采用运筹学的有关方法如线性规划、非线性规划(无约束或有约束)和整数规划的有关方法来作结构优化、或时间优化等。需要了解上述知识的读者可参阅相关的运筹学文献。

6.3 软件项目开发团队的组织与建设

任何一个软件项目都必须由一个开发群体来负责完成，这个群体中的每一个人员均赋予了不同层次的权利和责任，为了同一个确定的目标，互相分工，相互协作，共同来完成软件项目的开发任务。这样的群体，人们通常称为团队(Team)。由于软件项目管理的需要，软件开发团队逐渐小型化，因而人们也将为完成软件开发项目不同任务的各种工作小组统称为团队(或子团队)。一般来说，软件行业的团队，其内涵类似于一般行业的组织或部门(Organization)； 一般来说，一个完整的软件项目开发任务需要如下的团队(子团队)：

(1) 项目经理团队；

(2) 编程开发团队；

(3) 软件测试团队；

(4) 产品可用性团队；

(5) 客户培训与文档团队；

(6) 硬件使用维护测试团队；

(7) 系统安装调试运行团队；

(8) 本地化团队。

需要说明的是：所谓可用性，通常是指人机界面的友好性，用户操作的简便性等等要求。而本地化则是负责将原版软件的语言翻译成某一国家或地区的语言，然后向这一国家或地区进行专门的发行工作的统称。在上述各种团队(子团队)中，编程开发团队无疑是最重要的组织，它与软件测试团队、项目经理团队构成软件项目开发的关键性中心团队(Core Team)。因为少了这三者中的任何一个，整个软件开发项目的质量、效率就无法保证，甚至导致软件项目开发无法进行下去。除了这三个“中心团队”之外，其他的各种团队人们常称为外围性支援团队，它们同样是保障软件项目开发高效、保质的必要组织。

作为一个软件企业，仅有上述各种软件开发团队仍然是不够的，因为产品生产出来了，如果不能推向市场，那么企业将得不到丝毫利润。因此任何一个软件企业还需要组织一支市场营销和市场开发团队。这些团队通常有：

(1) 产品销售团队；

(2) 产品售后服务团队；

(3) 客户咨询团队；

(4) 法律团队；

(5) 人力资源管理团队；

(6) 员工培训团队；

(7) 财务管理团队；

……

以下我们将介绍这些项目团队的特点、生成，组织与建设的有关内容，重点将介绍开发团队(Development Team)及其所包括的三个中心团队。

6.3.1 开发团队的特点

软件项目的开发团队是为适应软件产品设计、生产的高效、优质而建立的，因此不同的软件项目其开发团队的组织结构、人员配备与职责分工应该有所不同。然而，作为一个有组织的群体，不同的团队间仍然有着一些共同的特点，这些共同的特点包括如下四个方面。

1. 共同的系统目标

任何一个组织都有其自己的目标，项目开发团队也不例外。正是在这一共同的系统目标的感召下，项目开发团队的每一个成员凝聚在一起，并为之共同奋斗。这样的系统目标具有下述要求：

(1) 系统目标必须明确化，让每个团队成员知道自己努力追求的方向；

(2) 系统目标必须是可度量的，以便将系统总目标分解到每一个团队成员使其成为自己的工作目标，并让每个团队成员明了必须完成什么任务，何时完成，按照什么样的逻辑顺序去完成，与其他成员应如何配合；

(3) 系统目标应有一定的挑战性，从而激发团队成员高效率地去完成任务；

(4) 系统目标必须能被跟踪，并让团队小组的每个成员都能看到他们向其总目标前进的轨迹，从而激发每个团队成员的积极性。

2. 合理的分工与协作

项目团队中的每一个成员都应明确自己在整个团队中的角色、任务、职责和权利，例如项目计划经理的任务或职责是制定项目开发的进度计划、管理项目的进程，并进行跟踪，负责各子团队间的协调，进行开发成本、进度的统筹管理；项目设计经理的主要任务或职责是将用户(如企业领导)的远期目标和对软件项目的目标、功能和性能需求转变成目标软件的功能、结构、设计方案，并将这样的设计方案撰写成软件项目的具体设计规范书；软件设计工程师的任务或职责是做好软件测试的准备(测试工具、测试用例等的准备)，并对所开发的软件作功能、性能测试，寻找软件中的一切瑕疵和缺陷并寻找产生的根源，它包括所有的设计问题、功能问题、性能问题、可用性问题，甚至产品说明书中的问题等，亦即寻找软件中的一切“害虫”(Bug)…… 在明确了每一个团队成员的角色、任务、权力和职责后，还必须明确各成员之间的相互关系，这种相互关系包括彼此任务完成间的逻辑关系、协作关系等。

3. 高度的凝聚力

所谓凝聚力，是指为维系项目团队为完成共同目标时所有团队成员之间的共同协作精神。 一般来说，较小规模的团队，由于彼此交往与合作的机会较多，就容易产生凝聚力，而随着团队规模的增大，要使所有团队成员都具有高度的凝聚力，这就需要项目经理和其他管理人员的大量工作，包括最大限度地设法满足团队成员的合理需求，关心和帮助团队成员存在的问题和困难，大力宣传团队凝聚力的重要性，等等。

4. 团队成员的相互信任和有效沟通

任何一个优秀团队在完成任务的进程中都不可避免地会有不同的认识和见解，因此作为团队的管理者，要鼓励团队成员自由地发表自己的见解，大胆地提出一些可能产生争议或冲突的问题，然后通过相互讨论与协商来解决冲突，形成共识。而要达到上述氛围的前提显然是团队成员的相互信任和建立有效的沟通机制，而这种有效的沟通方式应是全方位的，它包括各种各样的、正式和非正式的信息渠道的建立，如运用通信网络进行沟通，采用会议与座谈进行面对面的讨论与沟通，以及个人与个人间的沟通等，只要团队形成了一种开放的、坦诚的沟通气氛，每个团队成员都能在沟通过程中充分表述自己的观点与意见，虚心倾听和接纳他人的有益建议，任何冲突都会得到解决并可能变成推动团队前进的动力。

6.3.2 开发团队的生成与组织

软件项目开发团队的组织一般要解决如下三个问题：

(1) 确定项目团队的任务结构和相应的组织结构。

(2) 确定进入开发团队的相当于全职的软件人员(Full time equivalent Software Personnel，FSP)总数和分布到各团队小组中的人员数。这里所谓的相当于全职的软件人员，是指该人员将全力负责本开发团队所承担的有关项目任务而不兼顾其他的软件项目任务。

(3) 选择项目开发团队的项目经理和负责软件设计、开发、测试、管理等骨干人员，以形成合理的技术结构。

以下介绍其相关内容。

1. 开发团队的任务结构

软件项目开发团队任务结构的确定是开发团队组织结构和进入团队的FSP总数求解的基础。图6.18为一个通用的软件开发团队任务结构。

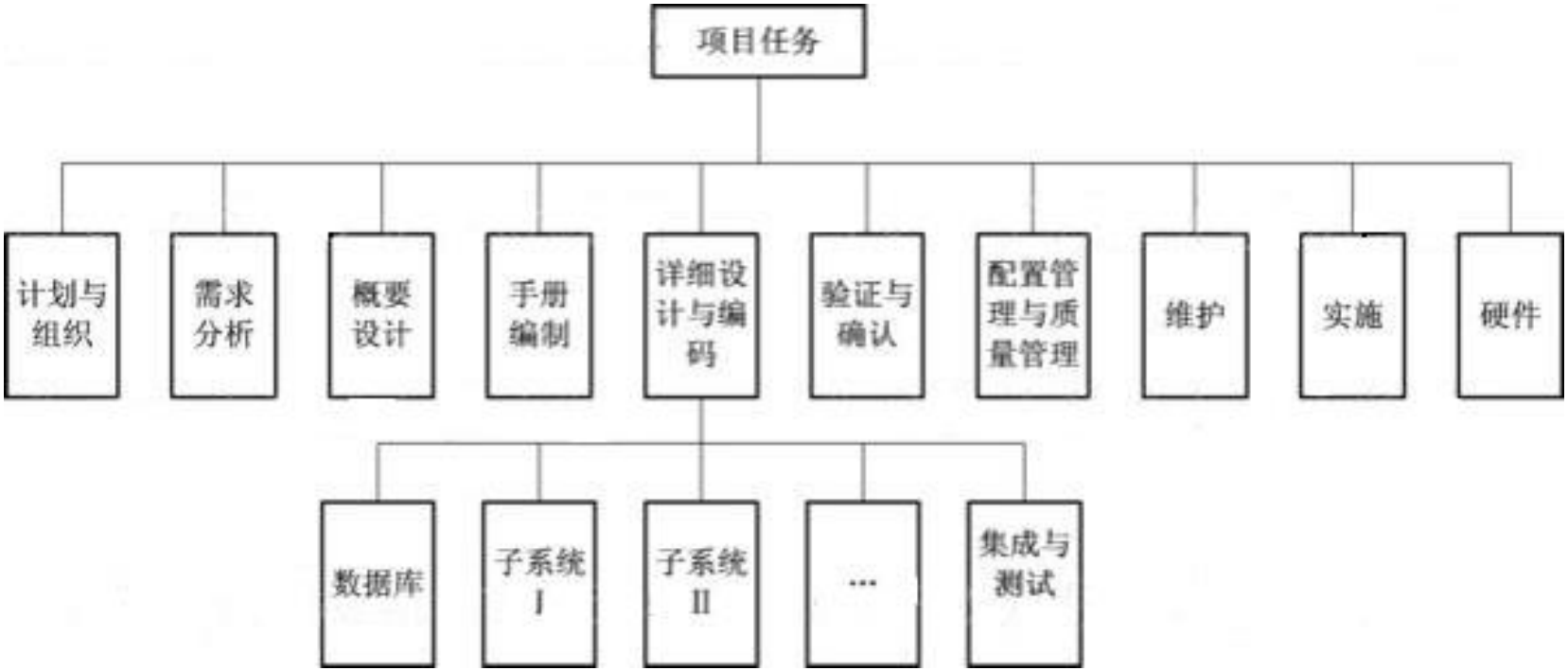


图6.18 项目团队任务结构

对于一个给定的软件项目，可以依据其自身的目标与功能需求，并参考图6.18的模式写出对应的项目团队任务结构。此中每一个方框中的活动内容可以交付一个团队小组来完成，此时上述的项目团队任务结构即可成为项目任务的组织结构，但对于一个中、大规模的软件项目来说，由于部分方框内的活动过于复杂或工作量较大，且考虑到各方框内活动所需的人力并非在开发过程中自始至终均参与工作，而是随着时间的推移、人力投入密度呈瑞利分布曲线的特征(详见第5章)。亦即具有人力投入密度开始自0递增到最高峰，然后又自最高峰递降下来的规律，因此人们认为按照软件开发过程的四个阶段来作项目任务细分更适合实际情况。

表6.7给出了上述各项活动按照开发过程的计划与需求I、概要设计II、详细设计与编码III、集成与测试IV等四个阶段的任务细分。若第 *i* 项活动在第 *j* 阶段有执行需要，则就在第*i*行第 *j* 列对应的方格中打 √。对于已打 √ 的阶段活动可采用功能点法、Delphi法、层次分析法(AHP)等方法中的一种估计其规模，然后即可据此来计算各阶段活动需投入的人力数，进而完成该软件项目的团队组织结构设计。

需要说明的是，从团队任务结构去求解团队组织结构的方法有自上而下的分解法与自下而上的汇总法。以下介绍的方法是自下而上的汇总法，即先求解出各阶段活动的规模，进而计算出相应的人力需求及系统需求总量，然后来设计相应的组织结构。

表 6.7 阶段任务细分表

阶段 / 活动	Ⅰ	Ⅱ	Ⅲ	Ⅳ
1. 需求分析与更新	√	√	√	√
2. 计划组织与控制	√	√	√	√
3. 概要设计		√		
4. 详细设计与编码	√	√	√	√
5. 验证与确认		√	√	√
6. 手册编制		√	√	√
7. 配置管理与质量保障			√	√
8. 硬件			√	√

2. 软件项目的人力需求与团队组织结构

设L_{ij}表示j阶段i活动的规模(单位：KLOC)，M_{ij}表示j阶段 i 活动的工作量(单位：人月)，t_{ij}表示 j 阶段 i 活动完成的工作长度(单位：月)，FSP_{ij} 表示 j 阶段 i 活动所需的全职人员数(单位：人)，C_{ij} 表示 j 阶段 I 活动所需成本(单位：千元)。则 L_{ij}、M_{ij}、t_{ij} 可由价值工程法、Delphi法、COCOMO法来确定。例如，若采用中级COCOMO模型，则有

$$
\begin{cases}
M_{ij} = U(rL_{ij}^{k}) \\
t_{ij} = h(M_{ij})^{d} \\
C_{ij} = M_{ij} \cdot F_{ij} \\
\mathrm{FSP}_{ij} = \dfrac{M_{ij}}{t_{dj}} \\
\mathrm{FSP}_{S} = \sum_{i=1}^{8}\sum_{j=1}^{4}\mathrm{FSP}_{ij}
\end{cases}
\tag{6.19}
$$

其中，U为由软件开发环境所确定的工作量乘数，F_{ij}为 j 阶段 I 活动的工时费用率(可通过历史数据或调查获得)。

在确定各阶段活动的全职人员数时，应注意如下几个问题：

(1) 应尽量使 FSP_{ij} 为整数，其不足之处可由同一阶段的其他活动全职人员数合并。

(2) 软件程序规模较大时，编程或测试阶段可将程序员再分成若干个小组，为管理有效起见，每个团队小组不宜超过7人。

(3) 在安排各团队小组的人员配置时，应尽量使投入的每个全职人员在完成任务时在时间上具有连续性，不宜将人员频繁地调动与更换工作任务。

表6.8给出了通过上述各步骤获得的各阶段活动的全职人员数，由表6.8所示的各阶段有关数据及上述注意事项，容易得到如图6.19所示的软件项目团队组织结构图。图中圆括号内的数字代表该团队小组的人数。

表 6.8　各阶段活动人数

阶段 / 活动	Ⅰ	Ⅱ	Ⅲ	Ⅳ
1. 需求分析与更新	7	2	7	3
2. 计划组织与控制	3	2	7	4
3. 概要设计		12		
4. 详细设计与编码	2	3	26	12
5. 验证与确认		2	6	14
6. 手册编制		2	3	3
7. 配置管理与质量保障			4	5
8. 硬件			1	3
共计	12	23	54	44

由图6.19所示的项目团队组织结构可知该项目的人力投入密度的时间分布呈瑞利分布曲线特征，每个阶段组成一个团队，承担该阶段中的若干项活动。对于那些工作量较大的活动，如概要设计阶段的设计任务(12人)、详细设计与编码阶段的设计与编码任务(26人)、集成与测试阶段中的编码修改与更新任务(12人)集成与测试任务(14人)可再作分解，使每个最终分解成的基本单元(团队小组)的人数不超过7人。

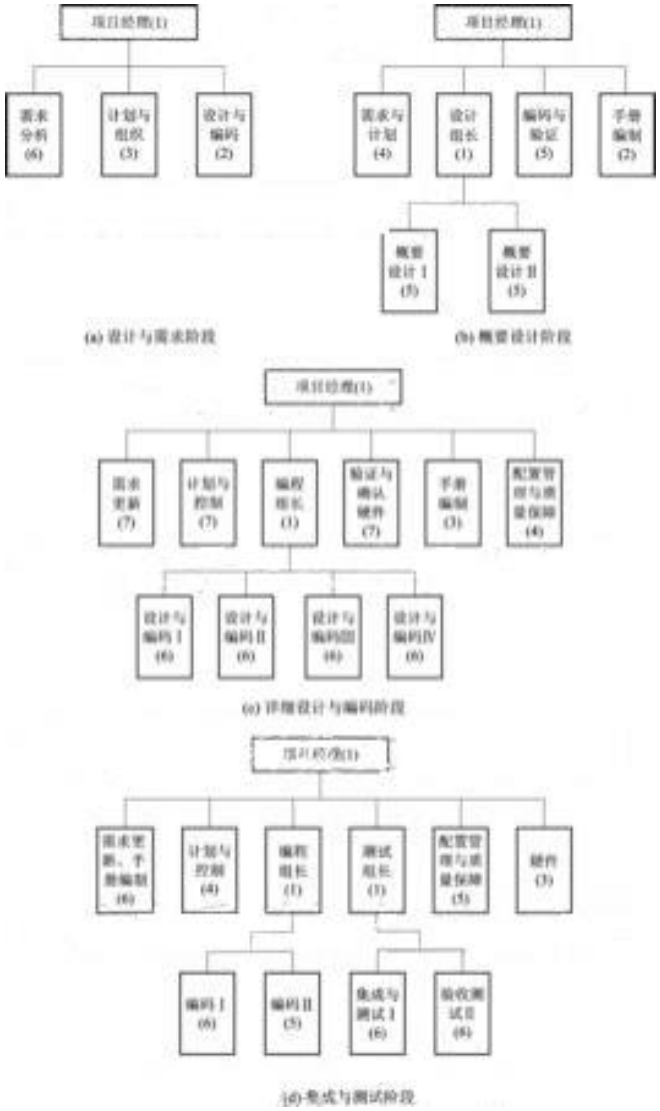


图6.19 项目团队组织结构图

3. 团队人员的选择

软件项目的开发工作不仅仅是计算机程序的编写，计划编制、成本、质量、进度的控制、手册的编制、硬件人员的支持等均不可缺少。因此，一个组织有序、行动高效的开发团队对团队人员素质的要求除必要的技术素质要求外，下列两个条件是必不可少的：① 开发团队必须有一个具备领导才能的项目经理；② 开发团队必须有一个相对合理的人员技术结构。项目经理作为项目的负责人、管理者和领导者，其基本职责是负责项目的组织、计划和实施全过程，以保证系统目标的成功实现。

因此，在考虑项目经理人选时，其具备的领导才能应具有下述特征：

(1) 完成任务显示出主动性、灵活性和适应性；

(2) 具有系统的思维能力和丰富的技术知识，能将大多数时间用于计划与控制；

(3) 具有娴熟的管理能力(包括决策能力、计划能力、组织能力、协调能力和人机交互能力)、丰富的项目管理经验和积极的创新能力；

(4) 具有一定的人格魅力和个人感召力，并能和上级、下级进行有效的沟通，对团队的每个成员的才能和个性有着敏锐的判断力，善于鼓舞士气，充分调动团队成员的积极性；

(5) 能根据项目的成本、工期和人力资源投入等因素来综合权衡技术方案的优劣。

项目开发团队的合理的人员技术结构要求团队具有如下的各类人员：

(1) 系统分析师；

(2) 构架设计师；

(3) 数据库管理员；

(4) 程序设计员；

(5) 测试设计员；

(6) 测试员；

(7) 项目复审师；

(8) 操作/支持工程师；

(9) 质量保证工程师；

(10) 用户界面设计师；

(11) 主题专家(用户)；……

6.3.3 开发团队的建设

1. 加强团队整合，建立有效的团队激励机制与约束机制

项目开发团队是一个由个人组成的集合，这些人员是由于其所具备的技能和能力被挑选出来以执行即将来临的项目任务。要使这些分散的人员为了共同的团队目标而工作，就必须采取各种有效的团队整合措施，如团队成员要各尽其才、各司其职，做到分工明确，权责分明；要建立规范的操作规程和科学的绩效考评制度，使每个团队成员明了什么能做，什么不能做，谁做得好，谁做得差，团队要定期召开会议，一方面传达项目的进度、成本进展信息，另一方面要反复强调团队的整体性；团队要建立有效的奖励机制，保证优秀的团队成员有工作任务安排的选择权，利用进度的控制权、稀缺设备的使用权和允许他们外出参加各种技术交流会议以开阔他们的视野，邀请他们共进晚餐，提供休假、旅游以及发放奖金、股权等措施。

2. 建立有效的沟通机制，创建相互信任、团结互助的团队精神

人们普遍认为IT项目成功的三大因素是：用户的积极参与、明确的需求表示和管理层的大力支持。而这三个要素全部依赖于良好的沟通技巧。因此开发团队与用户建立明确的沟通方式(口头、书面、网络、文档)，确定具体的沟通组织，建立良好的沟通氛围是十分必要的；作为团队内部的成员之间同样需要建立一种有冲突就应加强对话和沟通，通过平等协商以争取协调一致的工作环境；让团队成员形成尊重他人，学会宽容，承认错误，注意必要的妥协的工作作风，并创建相互信任、团结互助、为了共同的团队目标而共同努力的团队精神。

3. 加强技术培训与交流，注意知识创新，不断提高团队成员的创新能力

团队中的每个成员在以往的工作中都或多或少地积累有一定的经验和教训，并形成了自己对一些问题的特定看法和解决思路。他们在新的开发项目中，通过“干中学”和对话交流、模仿、理解和吸收他人的开发经验并结合自己的历史经验和知识，从而对软件项目开发有了进一步的认识和理解，完成了从个人隐性知识向显性知识的转化；每一个项目开发结束后，定期公开项目事后分析和回顾，总结收获和体会，形成新的经验与知识，从而完成团队隐性知识向团队显性知识的转化…… 有关团队知识创新过程的相互关系详见图6.20。此外，通过企业内部实现成员间的有效沟通和交流，建立知识联盟，与高校、科研院所建立产学研合作机制和机构等措施来为提高团队成员的创新能力打下良好的基础。

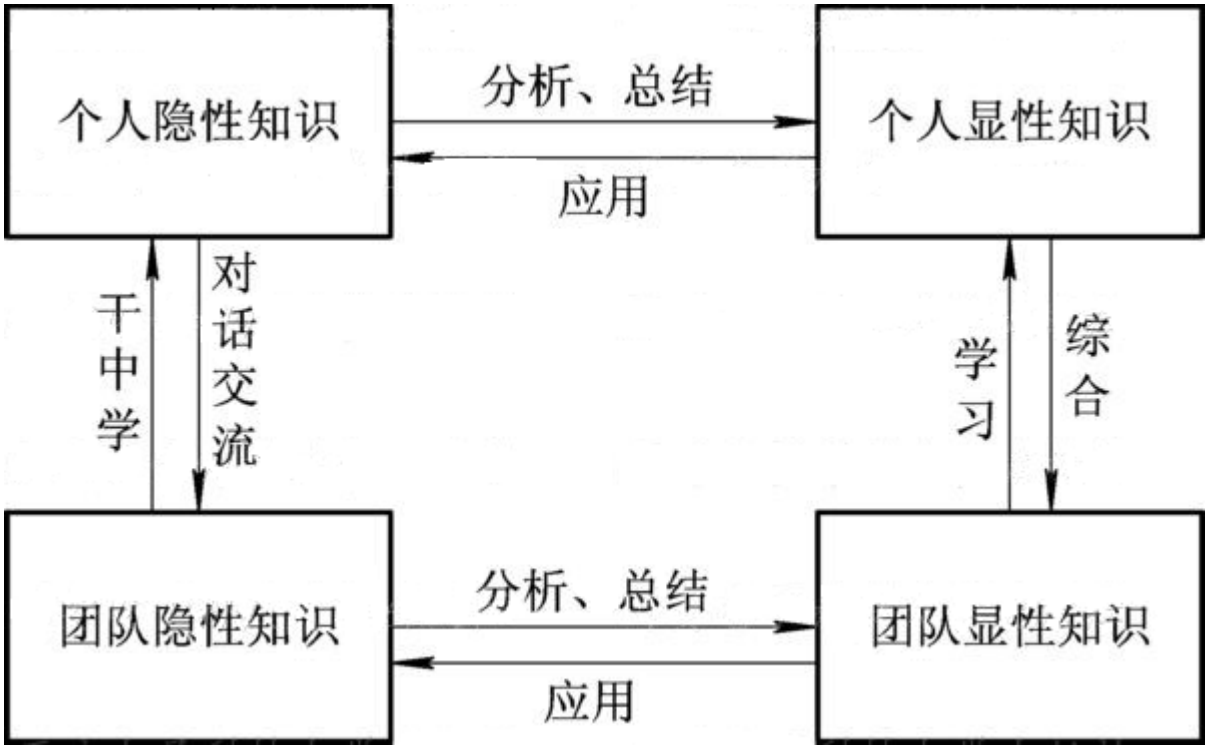


图6.20 团队知识创新过程

4. 重视软件项目开发团队的管理，聘用优秀人才，重视对人才的不断培养等

由于软件自身的特点，致使软件生产过程至今未能实现全自动或半自动机械化生产方式，而只能采用依赖于人的手工方式来生产。因此软件项目的管理除一般的时间进度管理、成本管理等外，更注重对开发团队的人力资源管理。考虑到开发团队的成员一般均为知识型员工，他们具有智力劳动的创建性，业务上的追求性，工作上的自主性和人力资本的流动性等特点，因此聘用优秀的软件人才，最大限度地发挥他们的工作积极性尤为重要。为此，建立适合于知识型员工的岗位绩效考评体系，恰当运用物质激励与精神激励相结合的激励手段，重视对人才的不断培养等应是对开发团队进行科学管理的主要方向。

作业

习题5、7、9

习 题 六

1. 软件产品的生产目标有哪些? 这些目标之间有何相关关系?

2. 什么是软件项目的进度计划? 如何来编制软件项目的进度计划?

3. 什么是软件项目的工作(任务)分解结构(WBS)? 任举一软件项目，写出其WBS。

4. 什么是软件项目的计划网络图? 计划网络图有何特性? 如图6.21所示的网络图是否满足计划网络图的要求? 若不满足要求，则说明不满足计划网络图的哪些要求? 并给予改正。

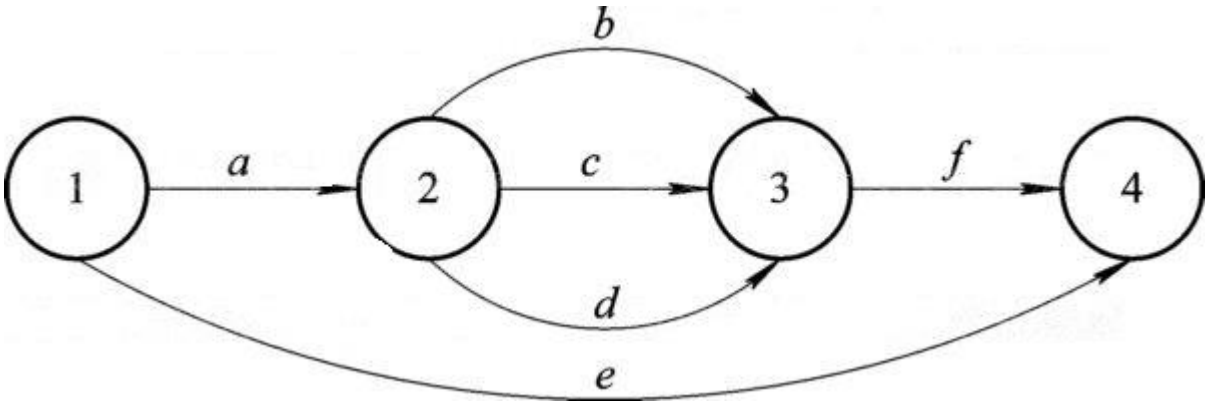


图6.21　某网络图

5. 某计划网络图的活动明细表如表6.9所示，画出与表6.9对应的计划网络图。

表 6.9 活动明细表 工序长度单位：月

工序名称	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*
工序长度	3	1	2	4	3	2	2	3	5	2	4	5	3
紧前工序	—	*a*	*a*	*b*	*a*	*c*	*d*	*f*	*d*	*i*	*e*、*h*、*g*	*f*	*j*、*k*、*l*

6. 某计划网络图如图6.22所示，求解此计划网络图G的关键路线与关键工序和工期。

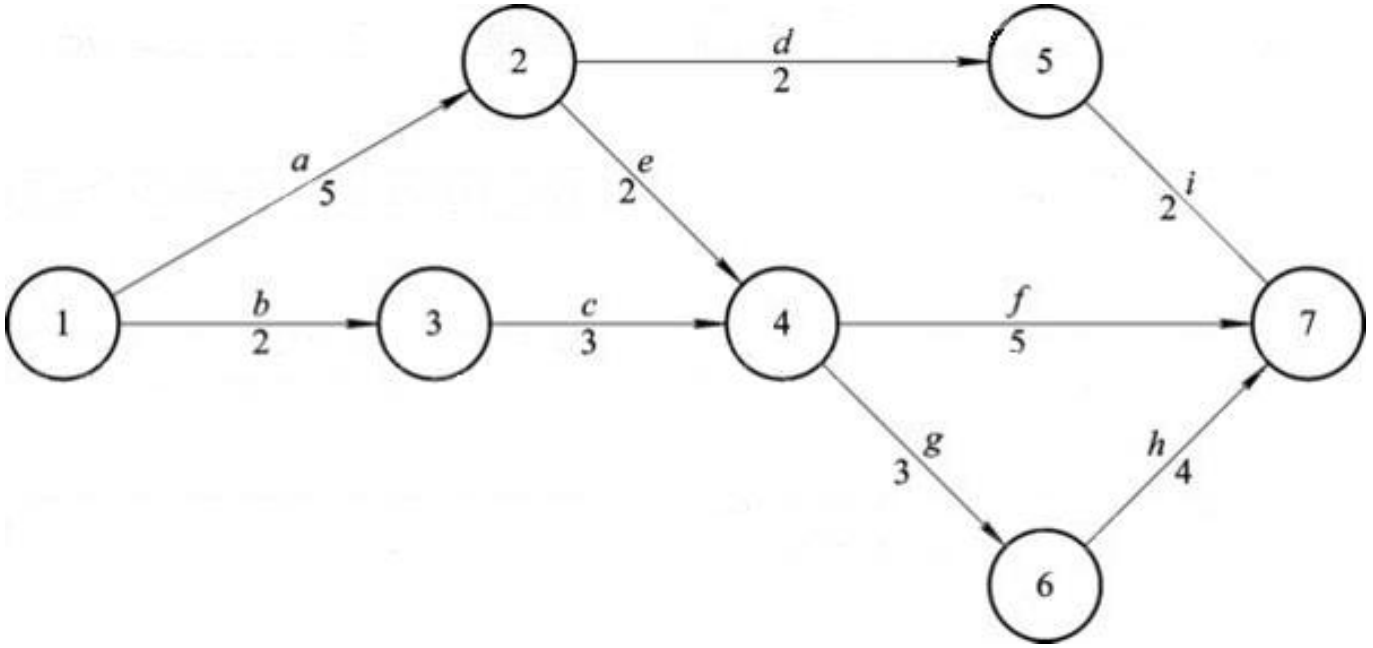


图6.22　某计划网络图

7. 求解表6.9活动明细表对应的计划网络图G之关键路线与关键工序和工期。

8. 某软件工程，其工序与工序长度见表6.10，根据工序间的执行逻辑顺序建立如图6.23所示的计划网络图G。

(1) 计算各工序的时间参数ES、EF、LS、LF和各结点的时间参数TE、TL。

(2) 求解该计划网络G的关键路线、关键工序和工期。

表 6.10 工 序 长 度 表

工序名	工序任务	工序长度/月	工序名	工序任务	工序长度/月
a	需求调查	3	*j*	设计与编码Ⅲ	2
b	测试计划	2	*k*	用户手册	1
c	概要设计	2	*l*	虚工序	0
d	虚工序	0	*m*	文档Ⅰ	2
e	测试数据采集	2	*n*	集成测试	3
f	测试驱动	1	*o*	用户培训	1
g	虚工序	0	*p*	文档Ⅱ	2
h	设计与编码Ⅰ	3	*q*	虚工序	0
i	设计与编码Ⅱ	4			

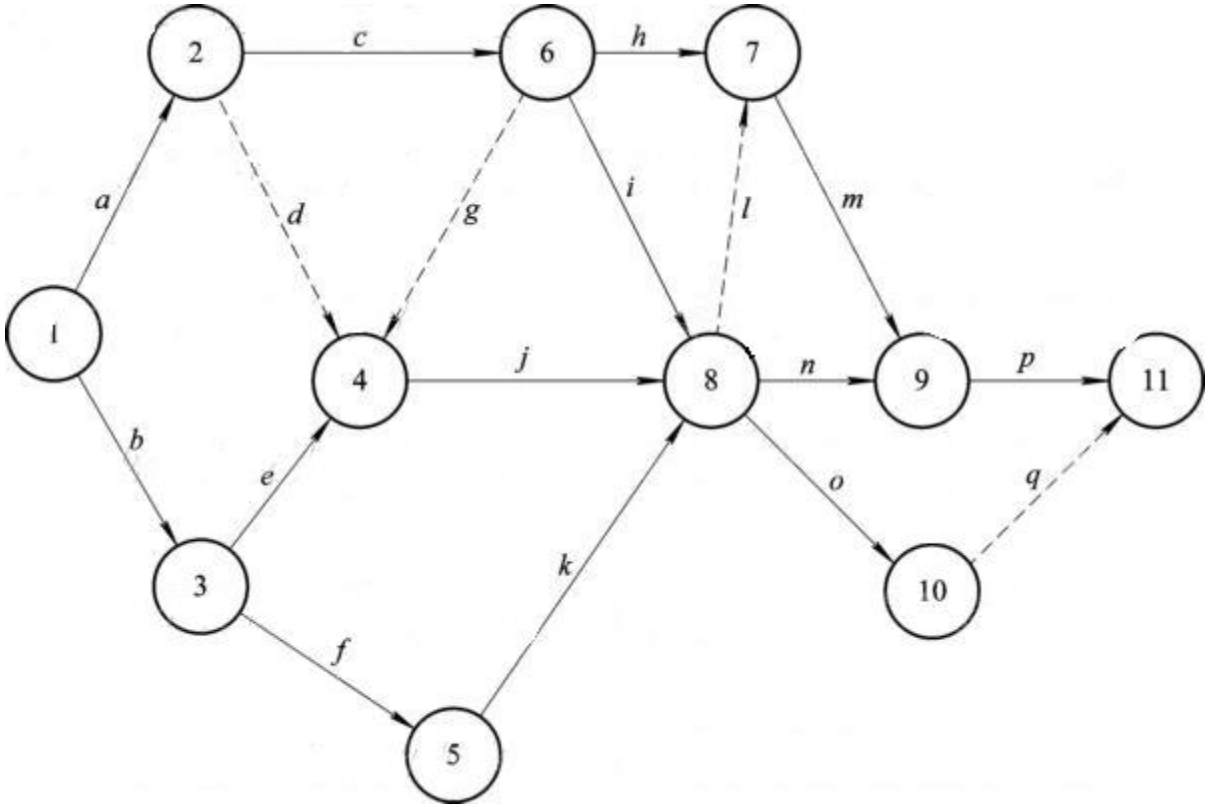


图6.23　计划网络图G

9. 对于如图6.22所示的网络图G(工序长度单位: 月)，

(1) 若给定总工期目标T_d=11.5月，求按此目标完工的概率。

(2) 若要求该计划网络图G完工的可能性达到95%, 此时该软件项目的总工期应确定为多少?

10. 某软件项目之计划网络图如图6.22所示。若工期T_d分别取9个月、10个月、11个月、12个月、13个月，

(1) 计算不同T_d对应的计划难度系数;

(2) 计算取不同T_d时该项目能按期完工的概率。

11. 一个中、大型软件项目开发任务，需要哪些团队? 软件项目开发团队有哪些特点? 如何进行开发团队的组织和建设?

12. 某软件项目根据其工作任务分解，获得生存周期内五个阶段(阶段V为运行与维护阶段)各活动的人数如表6.11所示。试画出相应的软件项目团队的组织结构图。

表 6.11 各阶段活动人数表

阶段 / 人数/人 / 活动	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ
1. 需求分析与更新	6	3	2	1	
2. 计划组织与控制	3	2	7	3	2
3. 概要设计		14	4		
4. 详细设计与编码	3	3	36	10	10
5. 验证与确认		2	4	20	10
6. 手册编制		3	4	6	2
7. 配置管理与质量保障	1	2	4	7	5
8. 硬件			2	3	3
合 计	13	29	63	50	32