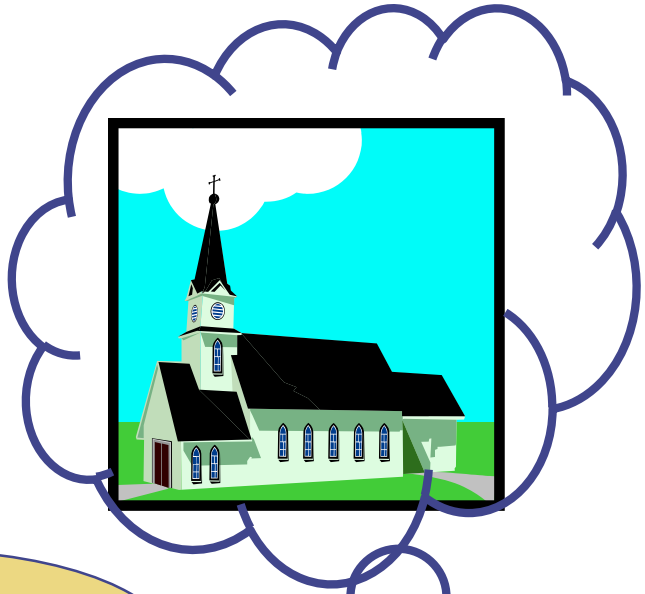
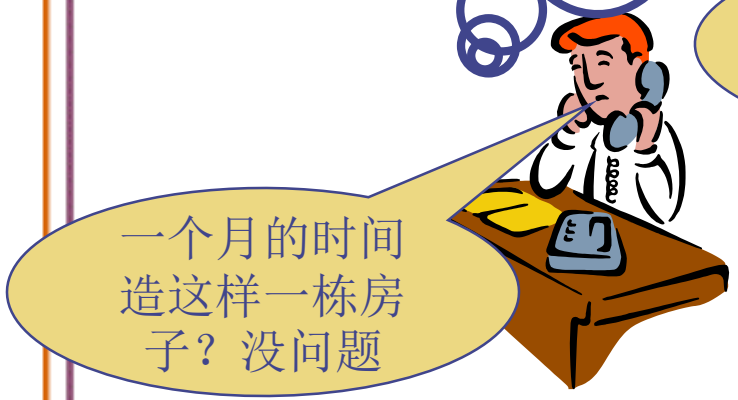
项目管理

软件项目工作量估算





你当初计划10万元造的房屋可能最终的实际造价为50万元。

从造房子中学到的

- 除非你确切知道“它”是什么？否则无法说明它的确切花费。
- 盖房子时，可以盖梦想中的房子（不考虑花费），也可以按估算盖，但是功能必须具有一定的灵活性

不确定性问题

- 客户会要求X功能吗?
- 客户要的是X功能的便宜版本还是昂贵版本呢？同一功能的不同版本的实施难度至少有10%左右的差别。
- 如果实施了X功能的便宜版本，客户会不会以后又想要昂贵的版本。
- X功能如何设计？同一功能的不同设计，在复杂度方面会有10%左右的差别。
- X功能的质量级别是什么？依据实施过程的不同，首次提交的X功能的缺陷数量会有10%的差异。
- 调试和纠正X功能实施过程中的错误要花多少时间？研究发现调试和纠正同样的错误，不同程序员所花时间会有10%左右的差异。
- 把X功能和其它功能结合起来要花多少时间?
- ……

软件工作量估算的渐进性

阶段	工作量和规模		工作量和规模	
	乐观	悲观	乐观	悲观
初始产品定义	0.25	4.0	0.60	1.60
批准的产品定义	0.50	2.0	0.80	1.25
需求说明书	0.67	1.5	0.85	1.15
产品设计说明书	0.80	1.25	0.90	1.10
详细设计说明书	0.90	1.10	0.95	1.05
摘自 Cost Models for Future Life Cycle Process: COCOMO 2.0,				

软件工作量估算困难的原因

- 估算困难是由于软件的本质带来的，特别是其复杂性和不可见性。
- 软件开发是人力密集型工作的，因而不能以机械的观点来看待
- 传统的工程项目经常会议相近的项目做参考，不同的只是客户和地点，而绝大部分软件项目是独一无二的。
- 新技术的不断出现和应用。
- 缺少项目经验数据，许多组织无法提供原有项目数据，而即使提供了这些项目数据，也未必非常有用。

软件估算的基础

- 历史数据的需要
  - 在参考历史数据时需要考虑不同的环境，如编程语言，软件工具，标准和人员的经验。
- 工作度量
  - 直接计算真正的成本或时间是不可能的。编写程序的时间不同的人将有显著的区别。
  - 通常将工作量表达为工作量，如源代码的数量（source line of code，SLOC），或者千行代码量（KLOC）
- 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。

软件工作量估计技术

- 算术模型
- 专家判断
- 对比法
- Albrecht功能点分析
- 自顶向下：首先定义整个项目的工作量，然后分解到各个部分
- 自下而上：各个部分的工作量先估算出来，然后进行合成

自底向上方法

- 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

自顶向下方法

- 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
- 形式：
- effort=（系统规模）*（生产率）
  - 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

专家判断

- 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由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。

类比估计

- 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。
- 如何选择与待预测的项目相近的项目？
  - 欧几里的距离（Euclidean Distance）公式
  - 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

Albrecht功能点分析

- 该方法是由Allan Albrecht在IBM工作时发明的自顶向下方法。

- 功能点的计算公式为：FP = UFP×TCF，

UFP为统计未调整的功能点计数

TCF称为技术复杂度因子，是由总影响度TDI计算出来的：TCF = 0.65 + 0.01×TDI。因此功能点的计算公式也可以表示为：

- FP = UFP×（0.65 + 0.01×TDI)

- 功能点法（Function Points）的基本点是计算机信息系统包括五个主要部件或者外部用户类型，它们是：
  - 外部输入：应用数据
  - 外部输出：提供给用户的面向应用的信息
  - 内部逻辑文件：逻辑主文件
  - 外部接口文件：与其它系统交换信息
  - 外部查询：在线的输入以获得立即的结果

功能点方法

- 加权因子的确定

Table 5.2 *Albrecht complexity multipliers*

External user type	*Multiplier*		
	Low	*Average*	*High*
External input type	3	4	6
External output type	4	5	7
Logical internal file type	7	10	15
External interface file type	5	7	10
External inquiry type	3	4	6

功能点方法计算实例

银行信息系统

该系统应能增加新客户，并能从客户文件中删除。系统支持客户的存款和提款业务。当出现透支时，系统应给出警告信息。客户可通过终端查询自己的账户余额，可以根据要求给出透支客户报告。

外部输入：增加新客户，删除客户，存款业务，取款业务，给出透支报告要求
外部输出：透支的警告信息，透支客户报告
外部查询：客户查询存款余额
内部文件：客户文件
外部接口文件：无

练习

- 在学院工资系统项目中需要开发一个程序，该程序将从会计系统中提取每年的工资额，并从两个文件中分别提取课程情况和每个老师教的每一门课的时间，该程序将计算每一门课的老师的成本并将结果存成一个文件，该文件可以输出给会计系统，同时该程序也将产生一个报表，以显示对于每一门课，每个老师教学的时间和这些工时的成本。
- 假定报表是具有高度复杂性的，其它具有一般复杂性

练习

- 外部输入：无
- 外部输出：报告，1
- 内部逻辑文件：财务输入文件，1
- 外部接口文件：工资文件，人员文件，课程文件，财务输入文件，4
- 外部查询：无

- 考虑加权：
  - 外部输入：无；外部输出：1×7＝7；内部逻辑文件：1×10＝10；外部接口文件：4×7＝28；外部查询：无；共：45

功能点方法：复杂性判定

- 如何判定功能的复杂性?
- 国际功能点用户小组（IFPUG)
  - 内部逻辑文件、外部接口文件

Table 5.3 *IFPUG file type complexity*

Number of record types	*Number of data types*		
	<20	*20 to 50*	*>50*
1	low	low	average
2 to 5	low	average	high
> 5	average	high	high

  - 外部输入文件

Table 5.4 *IFPUG External input complexity*

Number of file types accessed	*Number of data types accessed*		
	<5	*5 to 15*	*>15*
0 or 1	low	low	average
2	low	average	high
> 2	average	high	high

功能点方法：复杂性判定

- 外部输出文件

Table 5.5 *IFPUG External output complexity*

Number of file types	*Number of data types*		
	<6	*6 to 19*	*>19*
0 or 1	low	low	average
2 or 3	low	average	high
> 3	average	high	high

- 如何确定记录个数和数据个数
  - 如某系统内部逻辑文件：订单文件，包含订单信息（包括订单号，供应商名称，订单日期）和订单项（包括商品号，价格和数目），则记录个数为2，数据个数为6，在表中可以确定该功能点复杂性为低。

某个计算机辅助设计（CAD）应用为例，估算开发的软件包的输入、输出、查询、文件及外部接口。为了达到这个估算目的，我们假设复杂度加权因子都是平均的。根据对软件范围的叙述，对软件功能进行分解，识别出主要的几个功能：用户界面和控制功能、二维几何分析、三维几何分析、数据库管理、计算机图形显示功能、外设控制以及设计分析模块。最后可得到如下所示的估算表。

UFP=318

信息域值	乐观值	可能值	悲观值	估算数	权值	计数值
输入数	20	24	30	24	4	96
输出数	12	15	22	16	5	80
查询数	16	22	28	22	4	88
主控文件数	4	4	5	4	10	40
外部接口数	2	2	5	2	7	14
总计数值						318

因子i	所提问题	F_i取值
1	备份和恢复	4
2	数据通信	2
3	分布式处理	0
4	关键性能	4
5	现有的操作环境	3
6	联机数据登录	4
7	多屏幕输入切换	5
8	信息域值复杂度	5
9	输入、输出、文件、查询复杂度	3
10	内部处理复杂度	5
11	设计成可复用的代码	4
12	设计中的转换及安装	3
13	多次安装	5
14	方便修改的应用设计	5
$\sum F_i$		52

$TCF=0.65+0.01*\Sigma fi=0.65+0.01*52=1.17$

FP估算值=UFP*TCF=318*1.17=372

根据类似项目估计平均FP生产率：6.5FP/PM

劳动力价格8000元/月

FP的成本：8000/6.5=1230元

总项目成本：1230*372=457560元

工作量估算：372/6.5=58个人月

功能点方法：转换为代码行

- 通过定义各个功能点对应各种语言的代码行数，则功能点可以转化为代码行
- 一些数据：
  - Cobol: 91
  - C: 128
  - Quick Basic: 64
  - Object Oriented Languages: 30

L = 372×30 = 11160行 = 11.160KLOC

COCOMO: 参数化模型

- COCOMO：Constructive Cost Model
- 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。
- 基本的公式为：
- $Effort = c \times size^{k}$
  - 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)

COCOMO系数

- 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：
  - 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。
  - 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。
  - 半分离模式（Semi-detached Mode)两者之间
  - 信息系统是有机模式，而实时系统是嵌入式模式。

COCOMO系数

- 系数表：

Table 5.10 *COCOMO constants*

System type	*c*	*k*
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

- K的值反映了项目越大，则工作量成指数增加，因为大项目需要更多的协调和安排。

Table F.6 shows a comparison of the actual work-months from Table 5.1 and the COCOMO estimates. It illustrates the need for the calibration of COCOMO to suit local conditions.

Table F.6 *Comparison of COCOMO estimates and actual effort*

SLOC	*Actual (work-months)*	*COCOMO estimates*	*Difference (work months)*	*Difference (%)*
6050	16.7	15.9	−0.8	−4.9
8363	22.6	22.3	−0.3	−1.2
13334	32.2	36.4	4.2	13.1
5942	3.9	15.6	11.7	299.7
3315	17.3	8.4	−8.9	−51.2
38988	67.7	112.4	44.7	66.0
38614	10.1	111.2	101.1	1001.5
12762	19.3	34.8	15.5	80.2
26500	59.5	74.9	15.4	25.9

COCOMO修正

- 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：
  - $Pm=Pm_{nom}\times dem$

COCOMO成本因素

- dem的计算

Table 5.11 *COCOMO81 intermediate cost drivers*

Driver type	*Code*	*Cost driver*
Product attributes	RELY	required software reliability
	DATA	database size
	CPLX	product complexity
Computer attributes	TIME	execution time constraints
	STOR	main storage constraints
	VIRT	virtual machine volatility – degree to which the operating system changes
	TURN	computer turn around time
Personnel attributes	ACAP	analyst capability
	AEXP	application experience
	PCAP	programmer capability
	VEXP	virtual machine (i.e. operating system) experience
	LEXP	programming language experience
Project attributes	MODP	use of modern programming practices
	TOOL	use of software tools
	SCED	required development schedule.

Very low	1.46
Low	1.19
Nominal	1.00
High	0.80
Very High	0.71

练习

- 在某企业中，绝大多数系统技术上，产品，计算机和项目等属性都是类似的。只有人员的属性有所差异。该企业制定了下表：

Personnel attributes	ACAP	analyst capability
	AEXP	application experience
	PCAP	programmer capability
	VEXP	virtual machine (i.e. operating system) experience
	LEXP	programming language experience

Attribute	*Very low*	*Low*	*Nominal*	*High*	*Very high*
ACAP	1.46	1.19	1.00	0.86	0.71
AEXP	1.29	1.13	1.00	0.91	0.82
PCAP	1.42	1.17	1.00	0.80.	0.70
VEXP	1.21	1.10	1.00	0.90	—
LEXP	1.14	1.07	1.00	0.95	—

- 分析员非常优秀，编程人员也很优秀但是对该项目面向的领域不熟悉并准备用新的编程语言。他们对操作系统很熟悉。请计算dem。如果名义工作量是4人月，则估算的工作量是多少?

练习

Table F.7 *Calculating the development multiplier*

Factor	*Rating*	*Multiplier*
ACAP	very high	0.71
AEXP	low	1.13
PCAP	high	0.80
VEXP	high	0.90
LEXP	low	1.07

$dem = 0.71 \times 1.13 \times 0.80 \times 0.90 \times 1.07 = 0.62$

final estimate = 4 person-months × 0.62 = 2.48 staff months

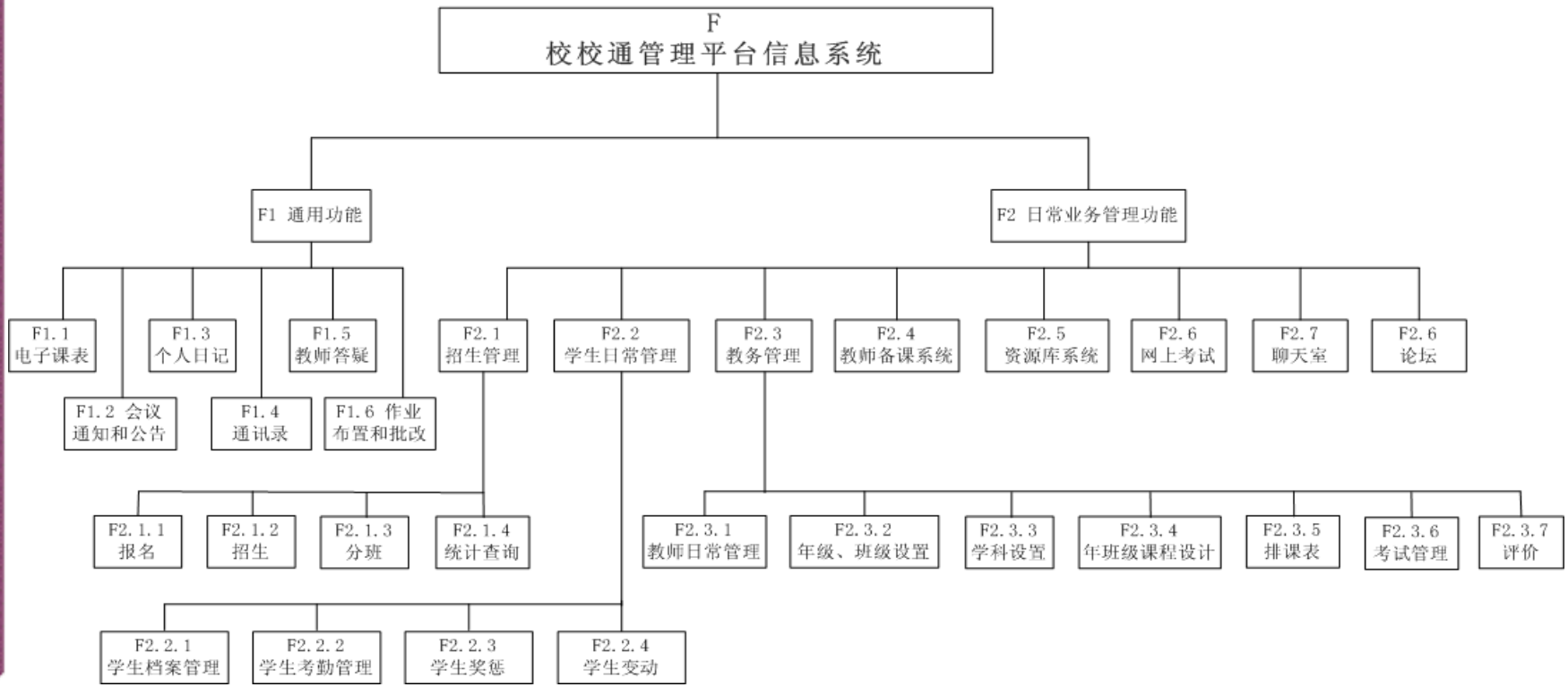
软件开发项目最短时间下限速查表

系统规模（代码行数）	系统软件类产品 时间（月）	系统软件类产品 工作量（人月）	商务类软件产品 时间（月）	商务类软件产品 工作量（人月）	紧包装类产品 时间（月）	紧包装类产品 工作量（人月）
10,000	6	25	3.5	5	4.2	8
15,000	7	40	4.1	8	4.9	13
20,000	8	57	4.6	11	5.6	19
25,000	9	74	5.1	15	6	24
30,000	9	110	5.5	22	7	37
35,000	10	130	5.8	26	7	44
40,000	11	170	6	34	7	57
45,000	11	195	6	39	8	66
50,000	11	230	7	46	8	79
60,000	12	285	7	57	9	98
70,000	13	350	8	71	9	120
80,000	14	410	8	83	10	140
90,000	14	480	9	96	10	170
100,000	15	540	9	110	11	190
120,000	16	680	10	140	11	240
140,000	17	820	10	160	12	280
160,000	18	960	10	190	13	335
180,000	19	1,100	11	220	13	390
200,000	20	1,250	11	250	14	440
250,000	22	1,650	13	330	15	580
300,000	24	2,100	14	420	16	725
400,000	27	2,900	15	590	19	1,000
500,000	30	3,900	17	780	20	1,400

估算的技巧

- 避免无准备的估算
  - 不要随口说出一个估算
- 留出估算的时间，并做好计划
  - 估算本身也是一个项目
- 开发人员参与估算
- 不要忽略普通任务
- 使用几种不同的估算技术，并比较它们的结果
- 估算的群体讨论

案例(校校通管理平台)


F
校校通管理平台信息系统
F1 通用功能
F2 日常业务管理功能
F1.1 电子课表
F1.3 个人日记
F1.5 教师答疑
F1.2 会议通知和公告
F1.4 通讯录
F1.6 作业布置和批改
F2.1 招生管理
F2.2 学生日常管理
F2.3 教务管理
F2.4 教师备课系统
F2.5 资源库系统
F2.6 网上考试
F2.7 聊天室
F2.6 论坛
F2.1.1 报名
F2.1.2 招生
F2.1.3 分班
F2.1.4 统计查询
F2.2.1 学生档案管理
F2.2.2 学生考勤管理
F2.2.3 学生奖惩
F2.2.4 学生变动
F2.3.1 教师日常管理
F2.3.2 年级、班级设置
F2.3.3 学科设置
F2.3.4 年班级课程设计
F2.3.5 排课表
F2.3.6 考试管理
F2.3.7 评价

WBS	名 称	估 计 值(人天)	小 计(人天)	总 计(人天)
1	通用功能		31	103
1.1	电子课表	8		
1.2	会议通知和公告	3		
1.3	个人日记	5		
1.4	通讯录	2		
1.5	教师答疑	5		
1.6	作业布置和批改	8		
2	日常业务管理功能			
2.1	招生管理		26	
2.1.1	报名	3		
2.1.2	招生	5		
2.1.3	分班	10		
2.1.4	统计查询	8		
2.2	学生日常管理		10	
2.2.1	学生档案管理	4		
2.2.2	学生考勤管理	2		
2.2.3	学生奖惩	2		
2.2.4	学生变动	2		
2.3	教务管理		31	
2.3.1	教师日常管理	2		
2.3.2	年级、班级设置	4		
2.3.3	学科设置	2		
2.3.4	年班级课程设计	5		
2.3.5	排课表	9		
2.3.6	考试管理	4		
2.3.7	评价	5		
2.4	教师备课系统	（外包5000元）	1	
2.5	资源库系统	（外包3000）	1	
2.6	网上考试	（外购3000元）	1	
2.7	聊天室	(已存在)	1	
2.8	论坛	(已存在)	1	

估算步骤如下:
1 获取项目分解结果WBS
2 计算开发成本
2.1项目规模是103人天，开发人员成本参数=480元/天，
则内部的开发成本=480元/天*103天=49440元
2.2外包外购的部分软件成本5000+3000+3000=11000元
开发成本 =49440+11000=60440元
3 计算管理、质量成本
项目的管理和质量成本=开发成本*20%=12088元
4 直接成本=60440+12088=72528元
5 计算间接成本
间接成本包括前期合同费用、房祖水电、培训、员工福利、客户服务等
间接成本=25%直接成本=18132元
6项目总估算成本=72528+18132=90660元
7重新评估项目的报价
项目的利润是30%，其中风险基金10%，利润15%，税费5%。
项目的总报价=90660*1.3=117858元

小结

- 软件工作量估算的特点
- 软件工作量估算困难的原因
- 估算的基本方法
- 功能点方法和COCOMO模型
- 估算技巧

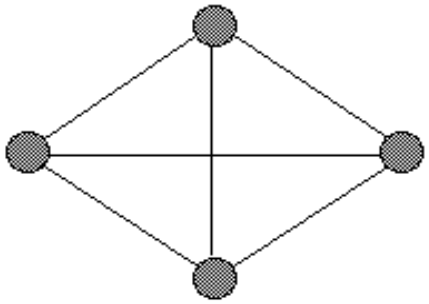
进度安排

- 软件开发项目的进度安排有两种方式：
  （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，最後交付日期由软件开发部门确定。

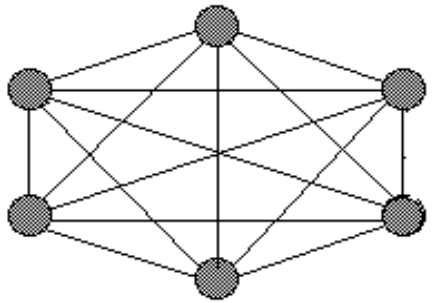
开发人数与软件生产率的关系

- 当几个人共同承担软件开发项目中的某一任务时，人与人之间必须通过交流来解决各自承担任务之间的接口问题，即所谓通信问题。通信需花费时间和代价，会引起软件错误的增加，降低软件生产率。

- 若两个人之间需要通信，则称在这两个人之间存在一条通信路径。如果一个软件开发小组有 n 个人，每两人之间都需要通信，则总的通信路径有 n(n-1)/2 (条)。



（a）四人之间所有通信路径



（b）六人之间所有通信路径

- 设一个人单独开发软件，生产率是5000行 / 人年。若 4 个人组成一个小组共同开发这个软件，则需要 6条通信路径。若在每条通信路径上耗费的工作量是 250 行 / 人年。则小组中每个人的软件生产率降低为

  5000 - 6×250 / 4 =

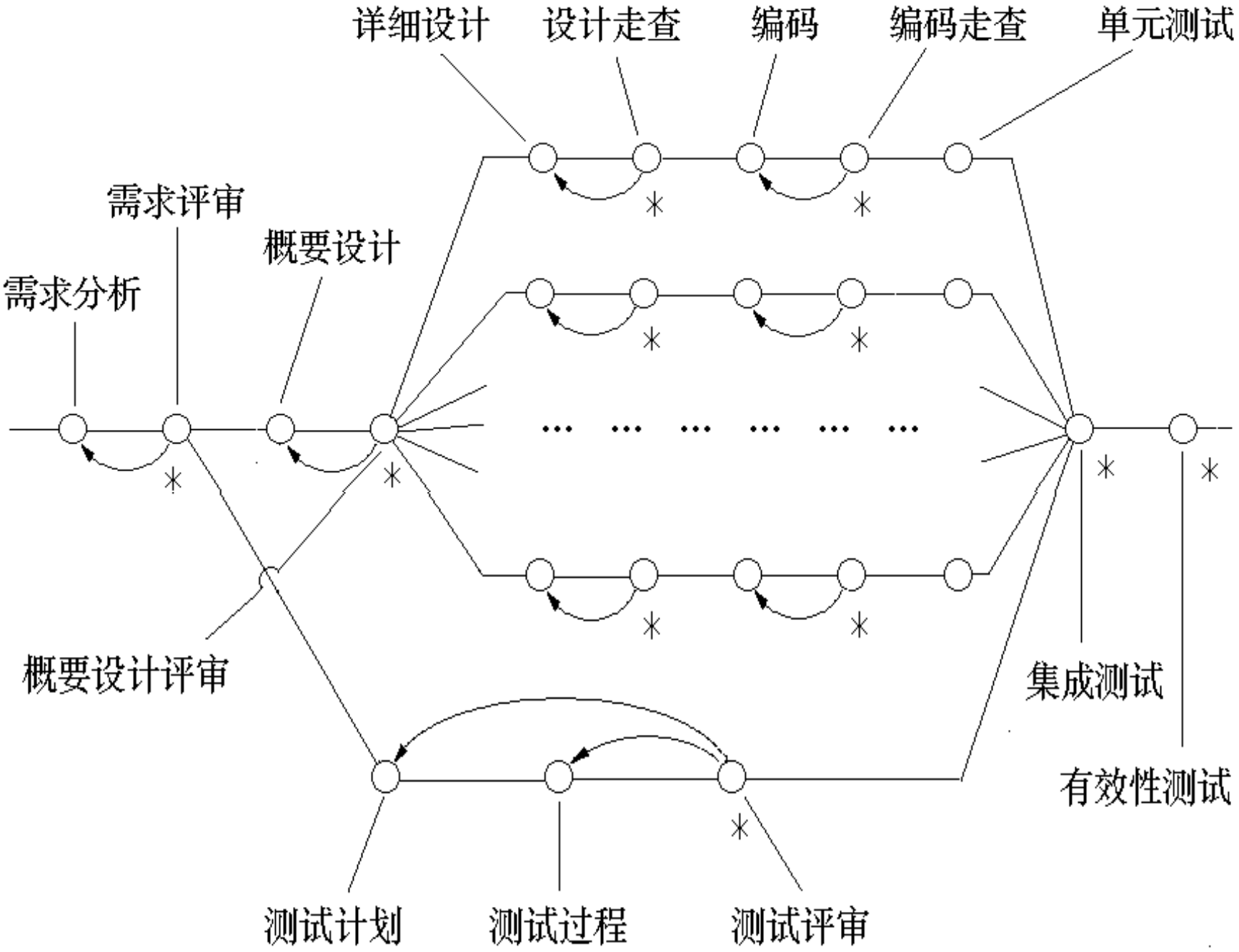
  = 5000 - 375 =

  = 4625 行 / 人年。

- 从上述分析可知，一个软件任务由一个人单独开发，生产率最高；而对于一个稍大型的软件项目，一个人单独开发，时间太长。因此软件开发小组是必要的。
- 但是，开发小组不宜太大，成员之间避免太多的通信路径。
- 在开发进程中，切忌中途加人，避免太多的生产率损失。

任务的分解与并行性

- 当参加同一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。
- 软件开发进程中设置许多里程碑。里程碑为管理人员提供了指示项目进度的可靠依据。
- 软件工程项目的并行性提出了一系列的进度要求。


详细设计
设计走查
编码
编码走查
单元测试
需求评审
概要设计
需求分析
概要设计评审
集成测试
有效性测试
测试计划
测试过程
测试评审

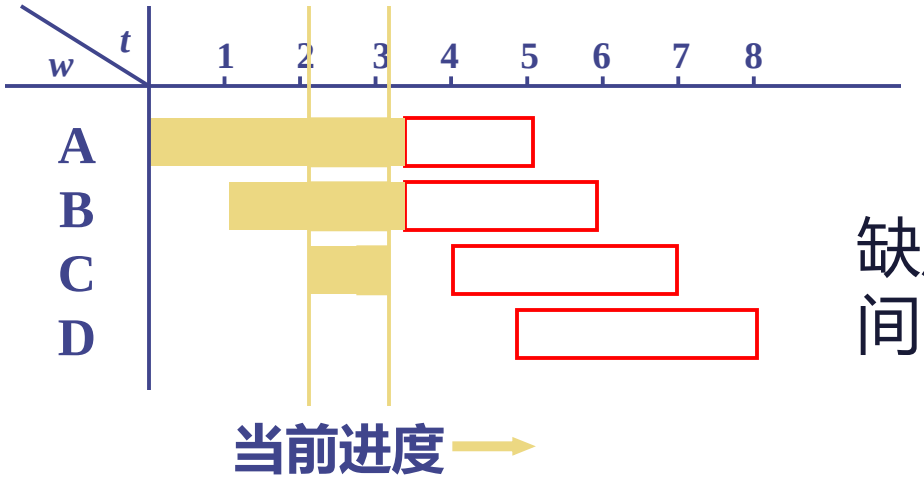
- 因为并行任务是同时发生的，所以进度计划表必须决定任务之间的从属关系，确定各个任务的先后次序和衔接，确定各个任务完成的持续时间。
- 项目负责人应注意构成关键路径的任务，即若要保证整个项目能按进度要求完成，就必须保证这些任务要按进度要求完成。

进度安排的方法

- 可以把用于一般开发项目的进度安排的技术和工具应用于软件项目。
- 为监控软件项目的进度计划和工作的实际进展情况，为表现各项任务之间进度的相互依赖关系，需要采用图示的方法。

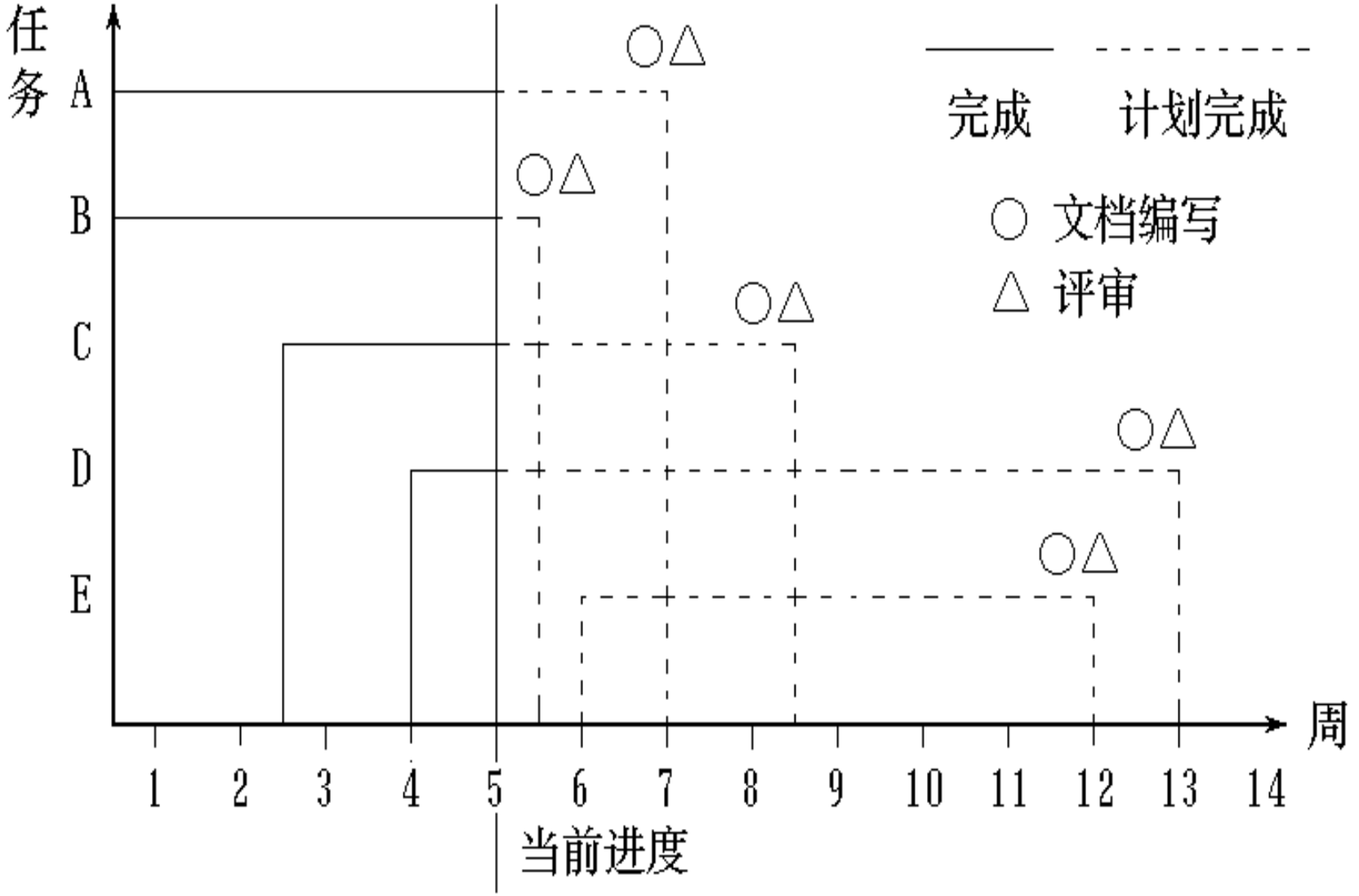
进度计划 (Software Plan)

1、Gantt Chart



优点：简单，能动态地反映开发进展。

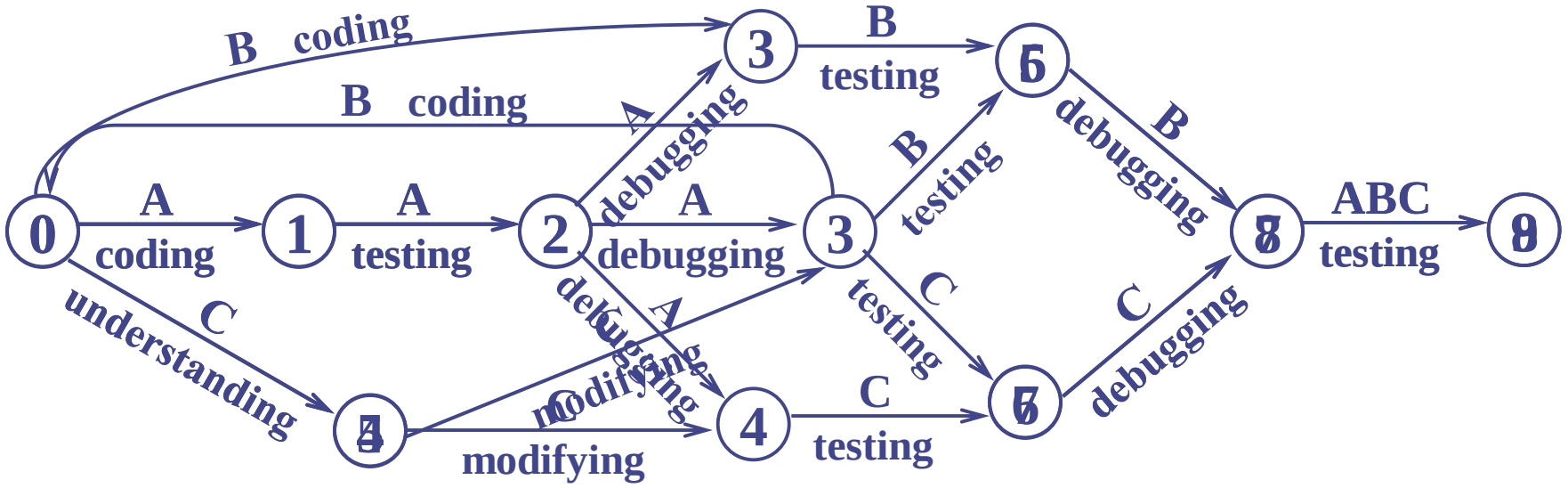
缺点：难以反映多个任务间的逻辑关系。

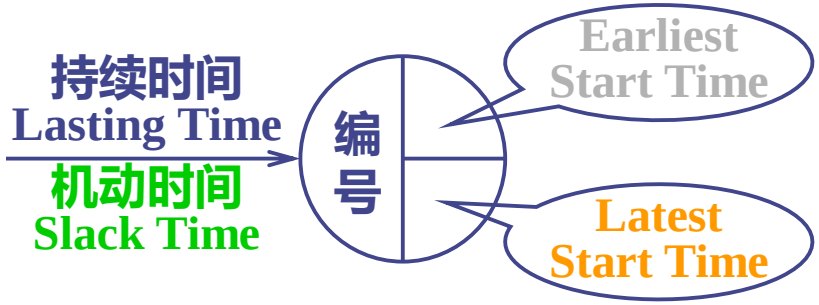

任务
A
B
C
D
E
完成
计划完成
文档编写
评审
周
1
2
3
4
5
6
7
8
9
10
11
12
13
14
当前进度

2、PERT (Program Evaluation & Review Technique)和CPM (Critical Path Method)

例：开发三个模块A、B、C。

A为公用模块，B、C的测试须等A的调试完成后进行。A的编码需6天，测试8天，调试6天。B的编码需7天，测试8天，调试6天。C利用已有的模块，须先理解原模块8天，再修改8天，测试9天，调试7天。最后三模块集成测试需5天完成。





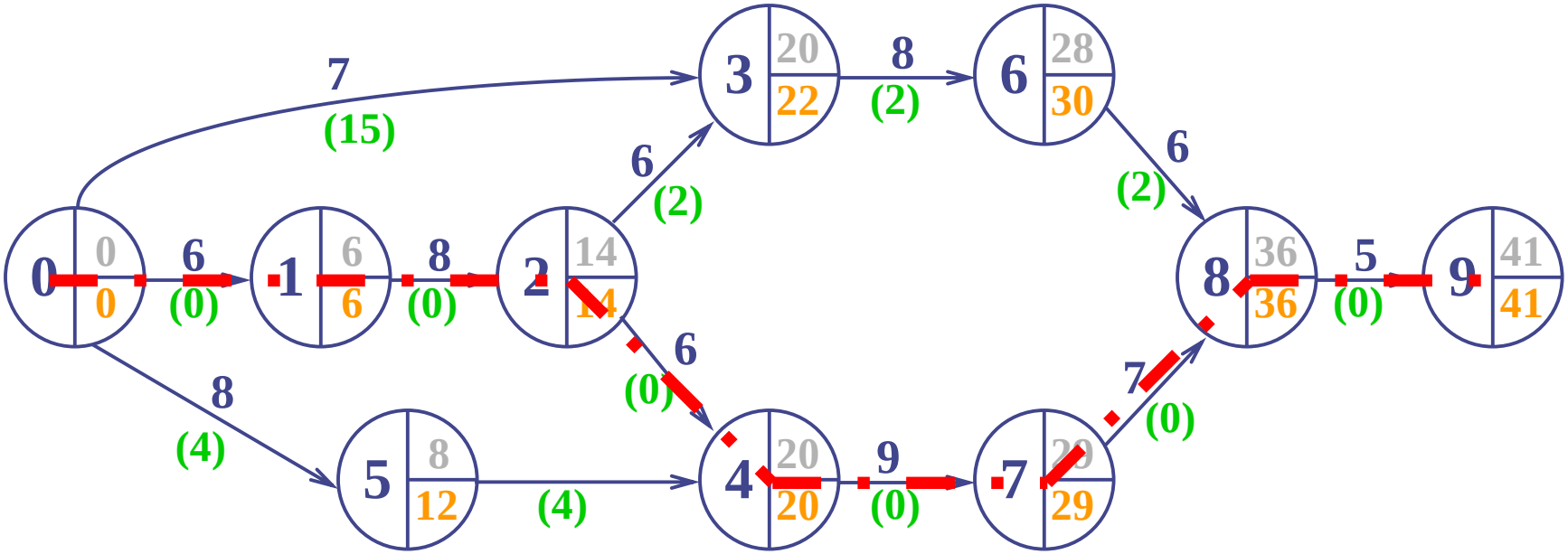
(1) 标出 Lasting Time

(2) 标出 EST: = 从起点始，所有进入事件的 EST+LT 中最大的

(3) 标出 LST: = 从终点(EST = LST)始，所有离开事件的 LST–LT 中最小的

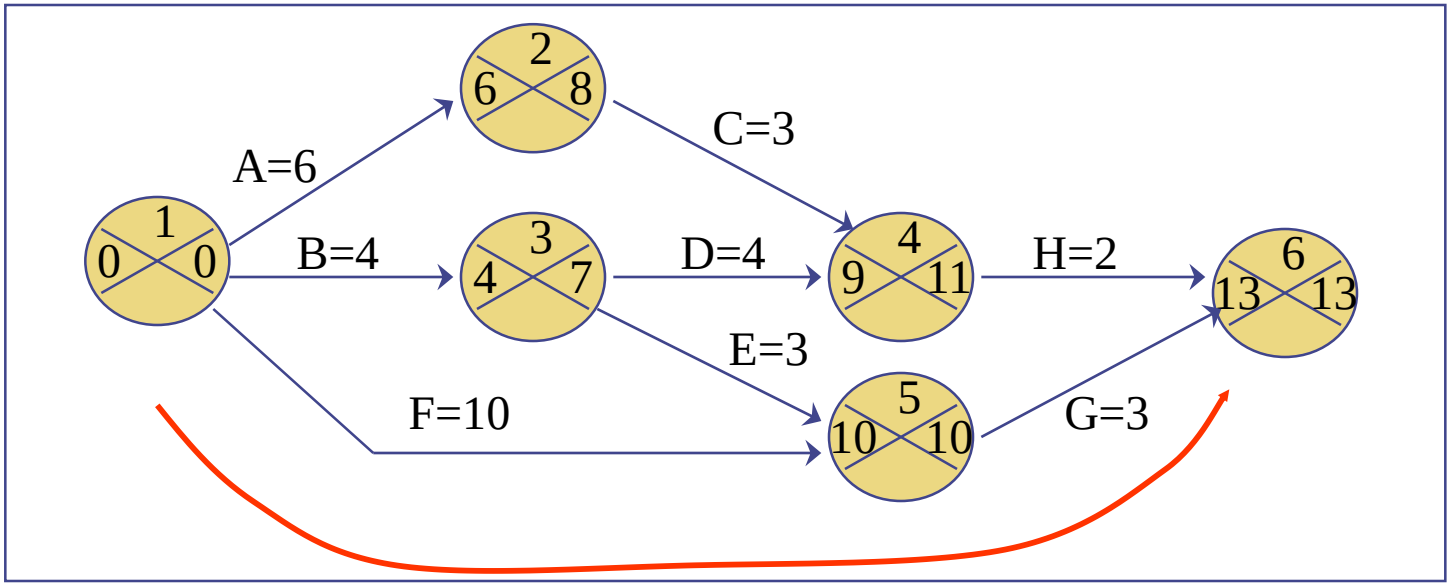
(4) 标出 ST: = 终点LST – 起点EST – LT

(5) 标出Critical Path: 即EST = LST 的所有事件组成的路径



练习

Activity	Duration(weeks)	Precedents
A Hardware selection	6	
B Software design	4	
C Install hardware	3	A
D Code & test Software	4	B
E File take-on	3	B
F Write user manuals	10	
G User training	3	E, F
H Install & test system	2	C, D


1
0
0
2
6
8
3
4
7
4
9
11
5
10
10
6
13
13
A=6
B=4
C=3
D=4
E=3
F=10
G=3
H=2