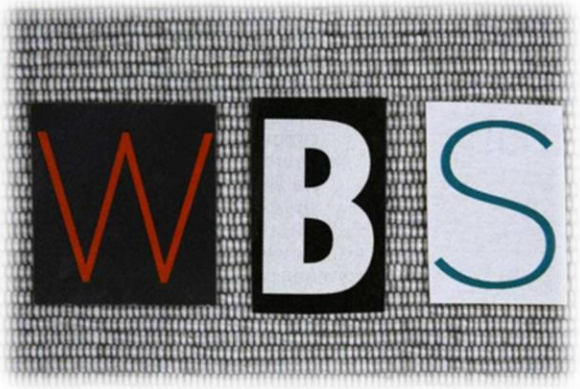
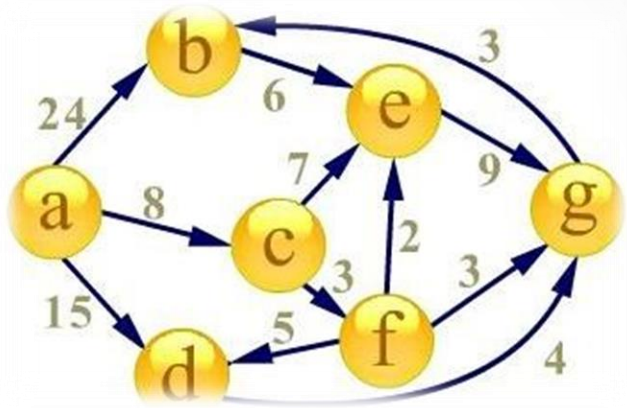
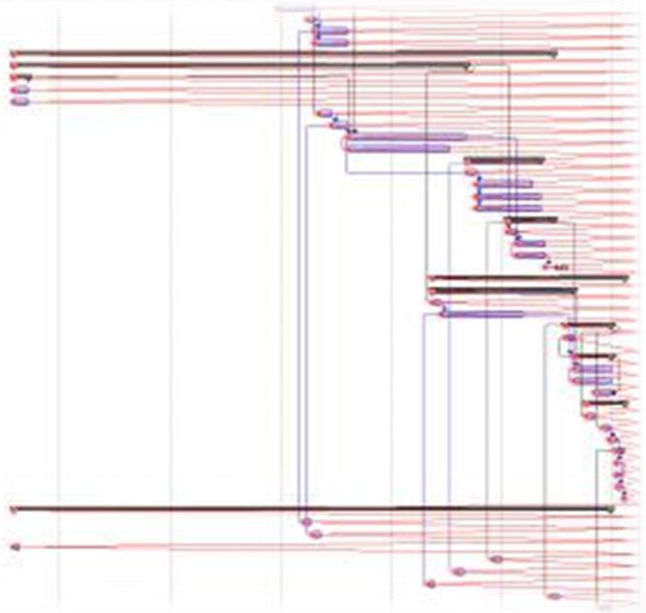
软件工程与实践

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com，微信电话: 13679081552







8.1.软件项目管理概念

- 软件项目管理的定义
- 软件项目管理的4P要素

8.1.1.软件项目管理的定义

- 计划、协调、度量、监控、控制及报告等管理方法在软件开发和维护中的具体应用，以保证整个过程是系统的、有原则的、可量化的(IEEE610.12-90)。
- 软件项目管理是为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对人员（People）、产品（Product）、过程（Process）和项目(Project)进行分析和管理的活动。

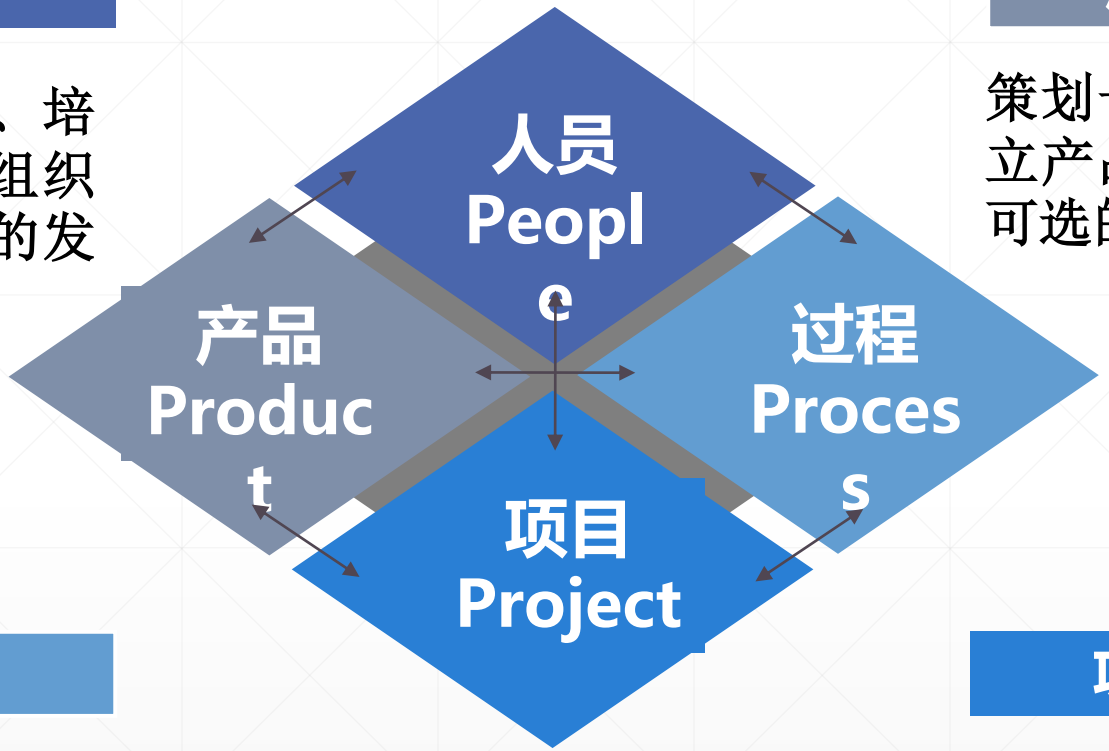
软件项目管理内容


内容
计划
实施
控制
范围
时间
成本
质量
沟通
风险
人力
采购
项目范围定义
活动定义与排序
执行进度计划
资源估算
成本估算
质量计划编制
信息计划
风险识别与量化
风险应对措施
团队组织计划
员工招聘计划
采购计划
招标计划
范围核实
计划执行
质量保证
信息发布
团队建设
招标/合同
范围变更控制
进度控制
成本控制
质量控制
执行情况发布
风险控制
团队激励与管理

8.1.2.软件项目管理的4P要素

人员

招聘、选拔、绩效管理、培训、薪酬、职业发展、组织和工作设计、团队/文化的发展



产品

策划一个项目以前，应当建立产品的目标和范围，考虑可选的解决方案

过程

软件过程提供一个框架，在此框架下可以制定项目开发的综合计划。

项目

理解成功项目管理的关键因素，掌握项目计划、监控和控制的一般方法

8.1.2.软件项目管理的4P要素：人员

高级管理者

负责定义业务问题

01

项目管理者

计划、激励、组织和控制软件开发人员

02

03

开发人员

拥有开发产品或应用软件所需技能的人员

客户

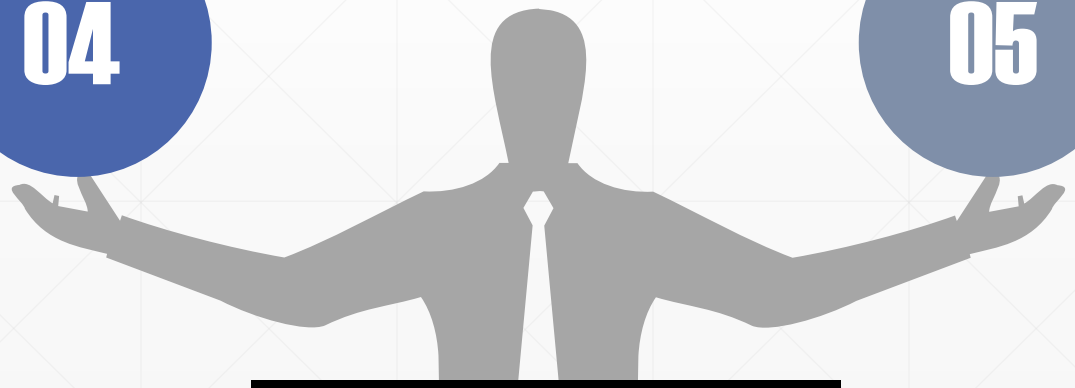
阐明软件需求的人员

04

05

最终用户

直接使用或者与软件产品交互的人



人员（People）

8.1.2.软件项目管理的4P要素：团队/人员

按照权利层次来组织团队

做过去类似项目有优势

难以承担创新型项目

封闭式

随机式

松散、专家组合型团队

有创新优势

难以承担有次序执行的项目

开放式

同步式

封闭式+随机式

适合解决有次序又有创新的复杂项目

效率可能不是太高

根据项目分解进行分工

适合松散耦合子系统项目

项目集成可能会遇到问题

团队（Team）/人员（People）

8.1.2.软件项目管理的4P要素：产品

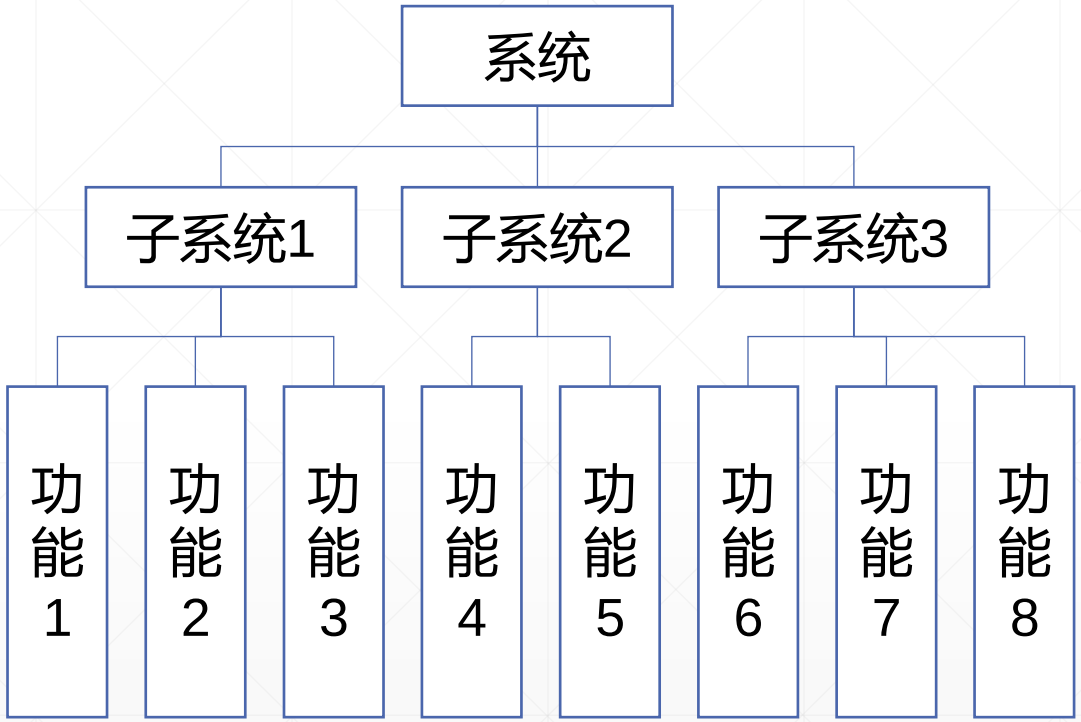
产品（Product）

项目环境
- 大系统环境/业务环境
- 环境的约束是什么

信息目标
- 输出哪些客户可见对象
- 需要什么数据作为输入

功能性能
- 如何将输入变成输出
- 要满足哪些特殊性能

确定软件范围



功能分解

8.1.2.软件项目管理的4P要素：过程

过程（Process）

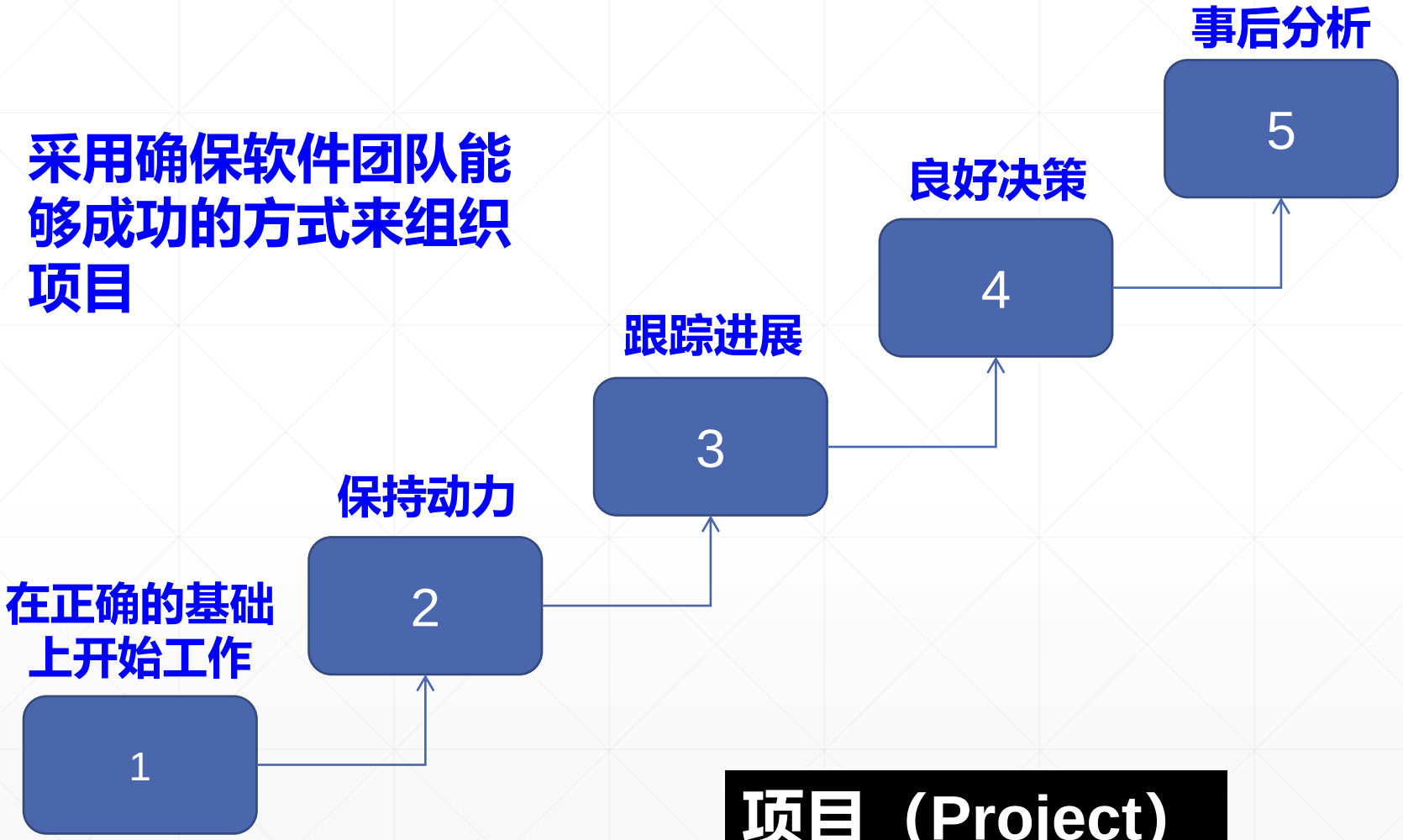
功能 \ 过程活动	沟通	策划	建模	构造	部署	
文本输入						
编辑与格式设计						
自动复制编辑						
页面布局能力						
自动生成索引						
文档生成						

根据项目特征选择合适的过程模型

根据过程模型进行项目分解

8.1.2.软件项目管理的4P要素：项目

采用确保软件团队能够成功的方式来组织项目

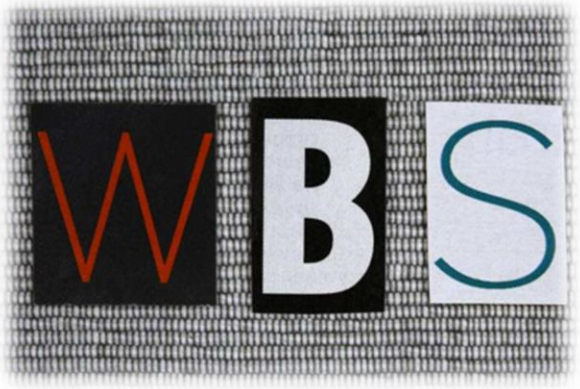


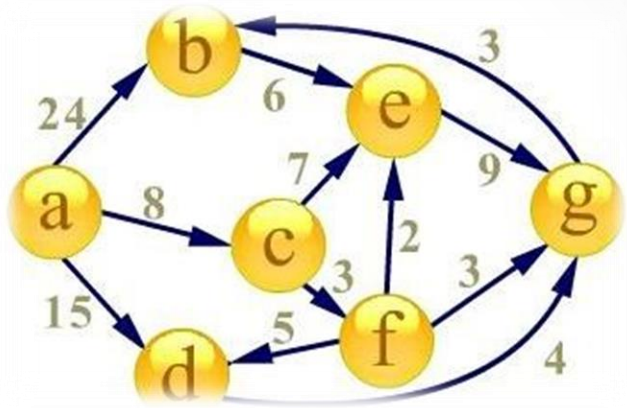
项目（Project）

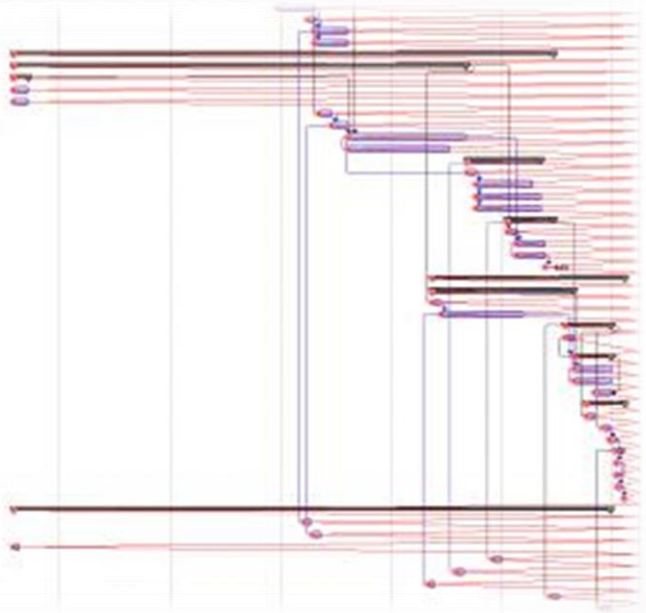
项目处于危险的信号：

- 不了解客户需求
- 产品范围定义不清楚
- 变更管理不好
- 最后期限不切实际
- 客户抵制

……







8.2.软件度量概念及面向规模的度量

- 软件度量的目的
- 软件度量的内容
- 软件度量的方法
- 面向规模的度量

8.2.1.软件项目度量的目的

- 当你能够度量你所说的事物，并能用数字表达它时，你就对它有了一定了解；反之，如果不能测量他，也不能用数字表达，就说明你对它的了解还不深入，不能令人满意
- 软件项目管理的成熟化也需要度量与数字化，目的是持续改进软件过程，并用于项目估算、质量控制、生产率评估等。

8.2.2. 软件项目度量的内容

- 生产率度量
  - 项目工作量
  - 项目周期
  - 项目成本
- 质量度量
  - 产品发布之前发现的缺陷数
  - 产品发布后用户报告的缺陷数
  - 产品的运行速度
  - ……

行业及组织的历史数据是软件项目度量的基础

8.2.3. 软件项目度量的方法

- 面向规模的度量
- 面向功能点的度量
- 面向对象的度量
- 面向用例的度量

8.2.4.面向规模的度量：概念

- **通过对质量和（或）生产率的测量进行规范化而得到的，这些测量是根据开发过的软件的规模得到的。**
- **千行代码（KLOC）：** 这些代码指的是源代码，通过源代码的行数来直观度量一个软件程序有多大规模
- **生产率（PM）**：PM = L / E, L表示代码总量(单位：KLOC)，E表示软件工作量(单位：人月)

8.2.4.面向规模的度量：概念

- **每千行代码的平均成本（CKL）**：CKL = S / L，S为软件项目总开销， L表示代码总量(单位：KLOC)
- **代码出错率（EQRl）**：EQRl = Ne / L，Ne表示代码出错的行数，L表示代码总量(单位：KLOC)
- **文档与代码比（Dl）**：Dl = Pd / L，Pd表示文档页数， L表示代码总量(单位：KLOC)

8.2.4. 面向规模的度量：示例

项目	代码行数(KLOC)	工作量(人月)	成本(万元)	缺陷代码行数	文档页数	人员
A	12.1	24	168	134	29	3
B	27.2	62	440	321	1224	5
C	20.2	43	314	256	1050	6

以项目A为例：

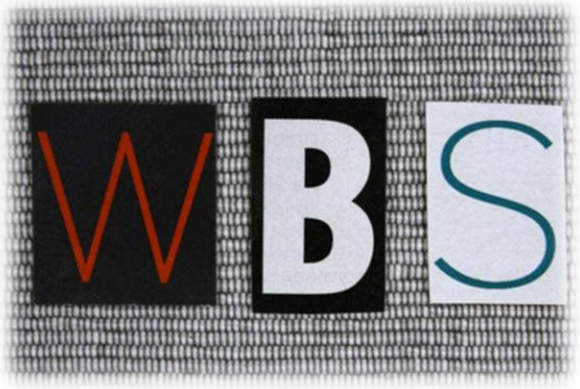
生产率（PM）= 12.1/24 = 0.51

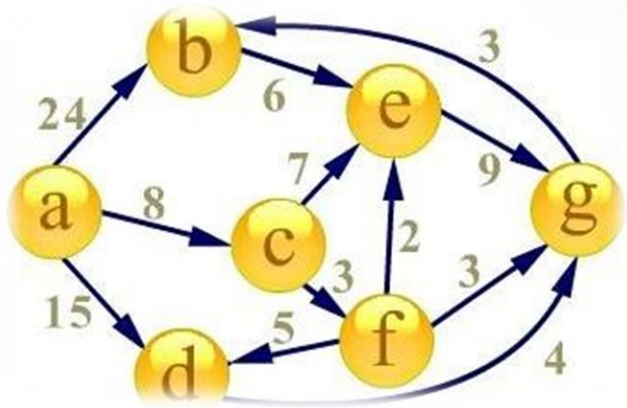
每千行代码的平均成本（ CKL ）= 168/12.1=13.9

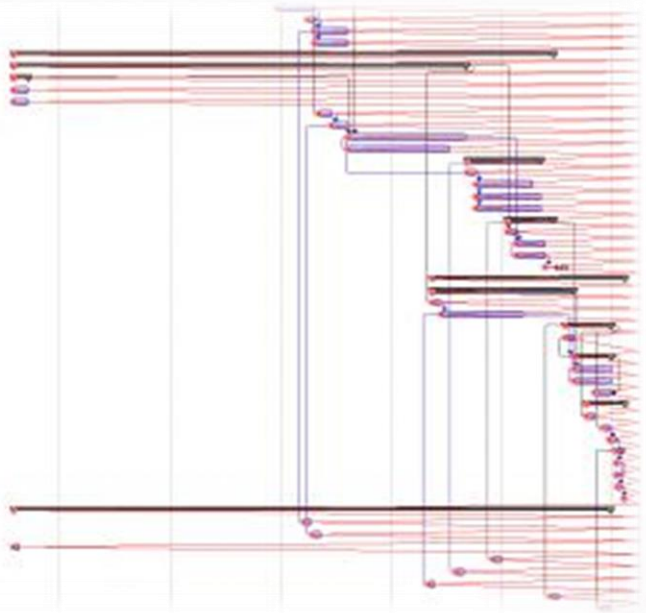
代码出错率（EQRI）=134/12.1= 11.1， 文档与代码比（DI）=29/12.1=2.4

8.2.4. 面向规模的度量：优缺点

- **优点**
  - 简单易行，自然直观
- **缺点**
  - 依赖于程序设计语言的表达能力和功能
  - 软件开发初期很难估算出最终软件的代码行数
  - 对精巧的软件项目不合适







8.3.面向功能的度量

- 面向功能度量的概念
- 功能点法计算公式
- UFC相关五类组件
- 复杂性调节因素
- 面向功能的度量优缺点

8.3.1. 面向功能的度量概念

- 用软件的功能表示软件的规模
- 应用最广泛的是功能点（Function Point, FP）法
- 项目开发初期就可估算出
- 功能点计算目前主要基于经验公式

8.3.2. 功能点计算方法

- $FP = UFC \times TCF = UFC \times (0.65 + 0.01 \times \Sigma Fi)$
  - UFC (Unadjusted Function Component) : 未调整功能点计数, 5个信息量的“加权和”
  - TCF (Technical Complexity Factor): 技术复杂度因子
  - Fi: 14个因素的“复杂性调节值” (i =1..14)
  - 0.65, 0.01都是经验常数，现在由国际组织根据大量项目跟踪分析获得。

8.3.3. UFC相关的五类组件

- 内部逻辑文件（ILF, Internal Logical Files）
  - 一个用户可识别的逻辑相关的数据组，它在应用程序边界内，由用户输入来维护
  - 它可能是某个大型数据库的一部分或是一个独立的文件
- 外部接口文件（EIF, External Interface Files）
  - 一个用户可识别的逻辑相关的数据组，但只能被引用，且数据完全存于软件外部，由另一个应用程序进行维护
  - 是机器可读的全部接口（如磁盘或磁带上的数据文件）
  - 是另一个应用程序的内部逻辑文件

8.3.3. UFC相关的五类组件

- 外部输入（EI, External Input）
  - 来自于软件外部的数据输入
  - 控制信息(不更新ILF) / 业务逻辑信息（更新ILF）
  - 可来自于一个数据输入屏幕或其他应用程序。
- 外部输出（EO, External Output）
  - 经过处理的数据，由程序内部输出到外部
  - 从ILF、EIF中取出数据经过一定的组合、计算后得出的输出数据, 如生成报表, 派生数据, 可能更新ILF
- 用户查询（EQ, External Query）
  - 一个输入输出的组合过程，从一个或多个ILF、EIF中取出数据输出到程序外部
  - 输入过程不更新ILF，输出过程不进行任何数据处理

8.3.3. UFC相关的五类组件

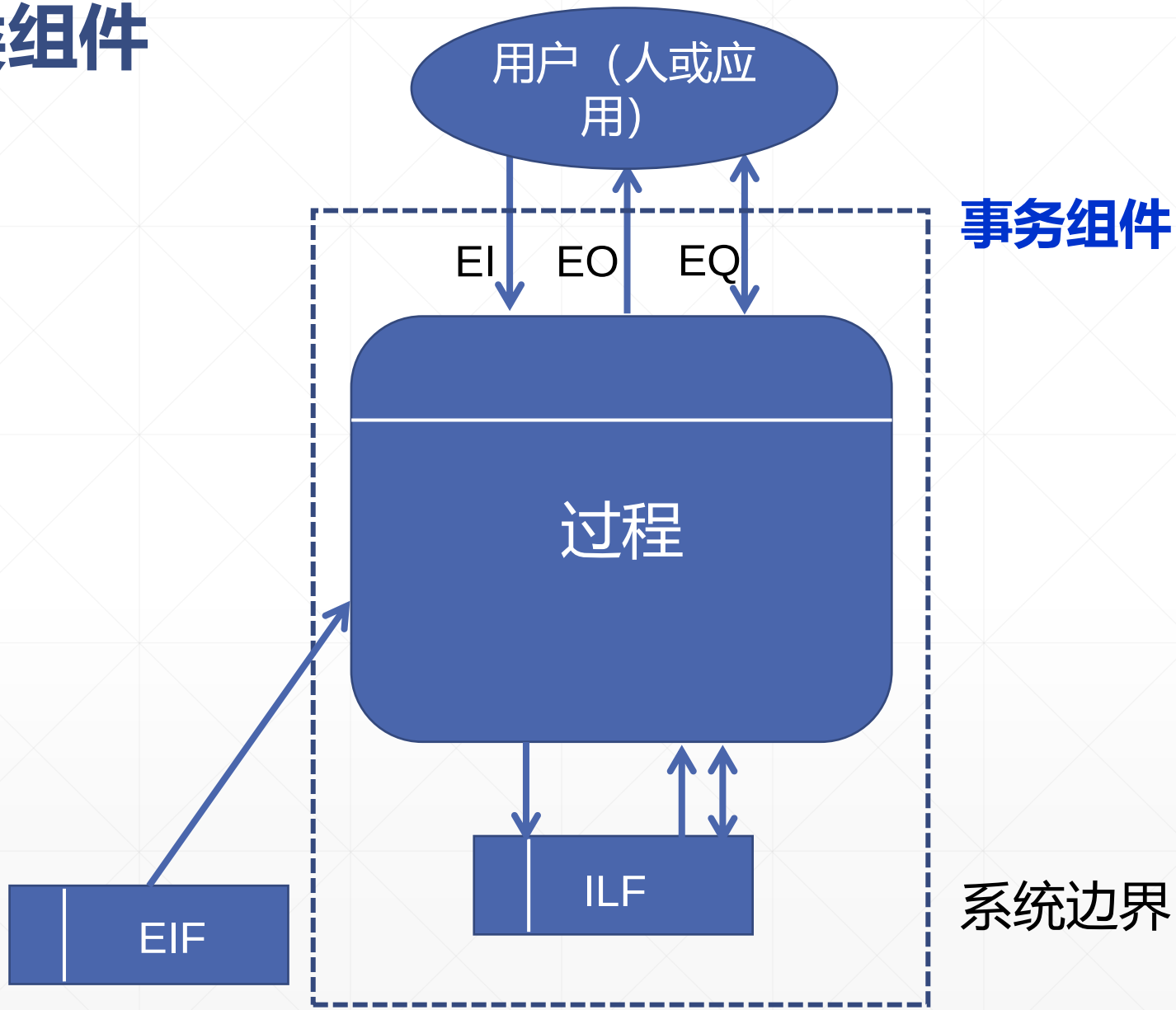
EI：External Input（外部输入）

EO：External Output（外部输出）

EQ：External Query（外部查询）

ILF：Internal Logical File（内部逻辑文件）

EIF：External Interface File（外部逻辑文件）



逻辑信息存储组件

8.3.3. UFC相关的五类组件

功能组件复杂度加权因子表

功能组件类型	简单	中等	复杂
外部输入数（EI）	3	4	6
外部输出数（EO）	4	5	7
外部查询表（EQ）	3	4	6
内部逻辑文件数（ILF）	7	10	15
外部接口文件数（EIF）	5	7	10

8.3.4. UFC的计算方法

- 外部输入EI数×加权因子(简单=3,平均=4,复杂=6)
- 外部输出EO数×加权因子(简单=4,平均=5,复杂=7)
- 外部查询EQ数×加权因子(简单=3,平均=4,复杂=6)
- 内部逻辑文件ILF数×加权因子(简单=7,平均=10,复杂=15)
- 外部接口EIF数×加权因子(简单=5,平均=7,复杂=10)

UFC = 上述计算值的总和（加权和）

8.3.5. UFC计算例子

- 假设每个功能要素的复杂度都是平均的。若有一个由25个数据登记表、15个报告、10个外部查询、 20个逻辑内部表单和5个接口文件组成的学生管理系统，其未调整功能点计数为：

UFC = (25x4) + (15x5) + (10x4) + (20x10) + (5x7) = 450（个功能点）

8.3.6. 14个复杂性调节因素F_i

1. 系统需要可靠的备份和复原吗?
2. 系统需要数据通信吗?
3. 系统有分布处理功能吗?
4. 性能很关键吗?
5. 系统是否运行在既存的、高度实用化的操作系统环境中?
6. 系统需要联机数据项吗?
7. 联机数据项是否在多屏幕或多操作之间进行切换?

复杂性调节因素值F_i
0-没有影响
1-偶有影响
2-轻微影响
3-平均影响
4-较大影响
5-严重影响

8.3.6. 14个复杂性调节因素F_i

8. 需要联机更新主文件吗?
9. 输入、输出、查询和文件很复杂吗?
10. 内部处理复杂吗?
11. 代码需要被设计成可重用吗?
12. 设计中需要包括转换和安装吗?
13. 系统的设计支持不同组织的多次安装吗?
14. 应用的设计方便用户修改和使用吗?

复杂性调节因素值F_i
0-没有影响
1-偶有影响
2-轻微影响
3-平均影响
4-较大影响
5-严重影响

$$FP = UFC \times (0.65 + 0.01 \times \Sigma Fi)$$

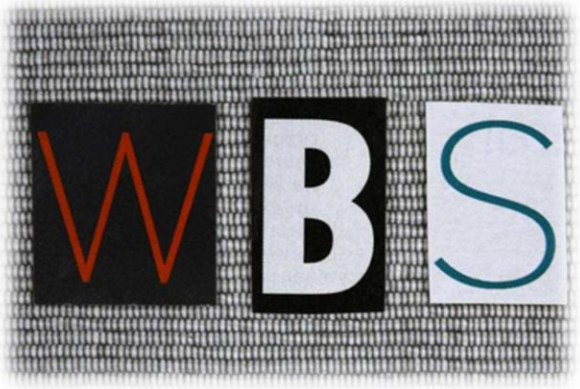
8.3.7. 面向功能的度量：优缺点

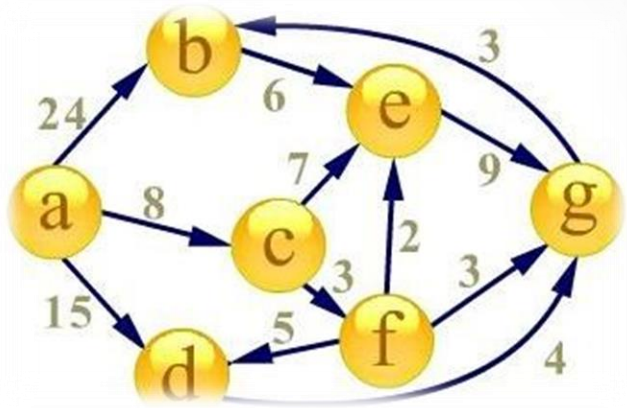
- 优点
  - 与程序设计语言无关, 在开发前就可以估算出软件项目的规模
- 不足
  - 没有直接涉及算法的复杂度，不适合算法比较复杂的软件系统；
  - 功能点计算主要靠经验公式，主观因素比较多

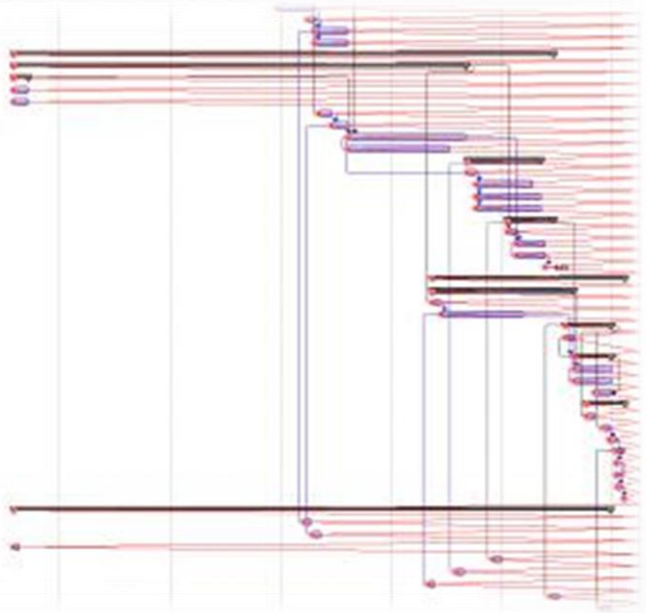
8.3.8. 面向功能的度量：和代码行的转换

一个功能点所需的代码行粗略估算

程序语言	平均值	中值	低值	高值
C	162	109	33	704
C++	66	53	29	178
ASP	86	83	20	184
HTML	43	42	35	53
Java	63	53	9	214
Javascript	58	63	42	75
SQL	40	37	7	110







8.4. 软件项目估算

- 软件项目估算概念
- 三点期望值法
- 基于LOC的项目估算
- 基于功能点的项目估算
- 基于过程分解的项目估算

8.4.1. 软件项目估算

概念

项目启动之前，软件团队应该估算将要做的工作、所需要的资源、成本、从开始到完成的时间，也即是对这些内容进行预测

策略

项目度量方法为项目估算提供了依据与有效输入

尽量把估算推迟到项目的后期进行

根据已经完成的项目进行估算

8.4.2. 项目估算方法

- 基于问题分解的估算
  - 基于LOC
  - 基于功能点FP
- 基于过程分解的估算

基于分解技术的项目估算方法

- 基于回归分析的经验估算模型
  - 基于LOC
  - 基于功能点FP
- COCOMO模型

基于经验的项目估算方法

8.4.3. 三点期望值法

- 在基于问题的分解估算方法中，通过估计最大值、最小值、最可能值的加权平均值作为期望值来估算
- 估计期望值=(最大值＋4×最可能值＋最小值) / 6
- 例如：如果估计系统X规模的最大值为100KLOC，最小值为50KLOC，最可能值为60KLOC，则其估计期望规模为(100+4×60+50)/6 = 65KLOC

8.4.4. 案例：基于问题分解的估算 (1/8)

- 软件描述(CAD软件)
  - CAD图形软件可接受来自用户的二维和三维几何数据，用户通过界面与CAD软件进行交互，并控制它，该软件具有良好的人机界面设计的特征。所有的几何数据及其支持信息存放在数据库中。开发设计分析模块，以产生所需的输出，这些输出将显示在各种不同的图形化设备上。软件在设计中要考虑与外设进行交互并控制它们，包括鼠标、数字化仪、打印机等。

8.4.4. 案例：基于问题分解的估算 (2/8)

- 软件子系统划分
  - 图形用户界面及其控制机制（UICF）
  - 二维几何分析（2DGA）
  - 三维几何分析（3DGA）
  - 数据库管理（DBM）
  - 图形显示（CGDF）
  - 外设控制(与打印机、数字化仪、扫描仪的接口）（PCF）
  - 设计分析子系统（DAM）

8.4.4. 案例：基于问题分解的估算(3/8)

- 基于LOC的估算：估算出各个子系统的代码行，例如三维几何分析功能的代码行估算范围为：
  - 乐观值a：4600
  - 可能值m：6900
  - 悲观值b：8600
  - 估算值： e = (a + 4m + b)/6 = 6800

8.4.4. 案例：基于问题分解的估算(4/8)

子系统	代码行
图形用户界面及其控制机制	2 300
二维几何分析	5 300
三维几何分析	6 800
数据库管理	3 350
计算机图形显示	4 950
外设控制(与打印机、扫描仪等的接口)	2 100
设计分析子系统	8 400
合计	**33200**

8.4.4. 案例：基于问题分解的估算(5/8)

- 历史数据
  - 平均生产率PM: 620 LOC/PM(620行代码/人月)
  - 每个人月的成本 C = 8000￥
- 估算项目成本和工作量
  - 估算工作量 = 总代码行/PM= 33200/620=54人月
  - 估算成本 = 估算工作量 ×每个人月的成本 = 54人月× 8000 = 43 2000￥

8.4.4. 案例：基于问题分解的估算(6/8)

- 基于功能点估算：

Step1: 估算功能点

信息域	乐观值	可能值	悲观值	估算计数	加权因子	FP计数
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
文件数	4	4	5	4	10	40
接口数	2	2	3	2	7	14
总计						318

8.4.4. 案例：基于问题分解的估算(7/8)

Step2: 计算复杂度调整因子

因子	值
备份和复原	4
数据通信	2
分布式处理	0
关键性能	4
操作环境	3
联机数据输入	4
多屏幕输入切换	5

因子	值
联机更新主文件	3
信息域值复杂性	5
内部处理复杂性	5
软件重用	4
转换和安装	3
多次安装	5
方便修改	5

复杂度调节因子	1.17

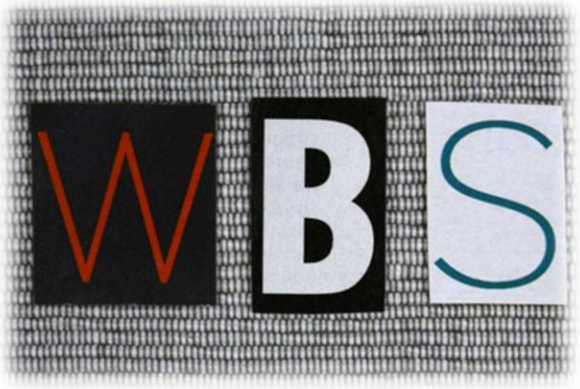
8.4.4. 案例：基于问题分解的估算(8/8)

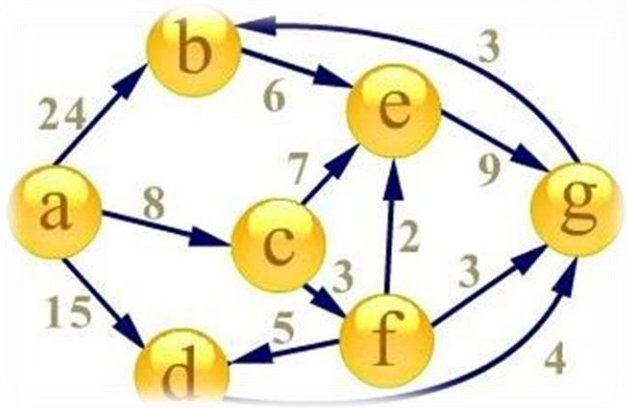
Step3: 计算成本与工作量

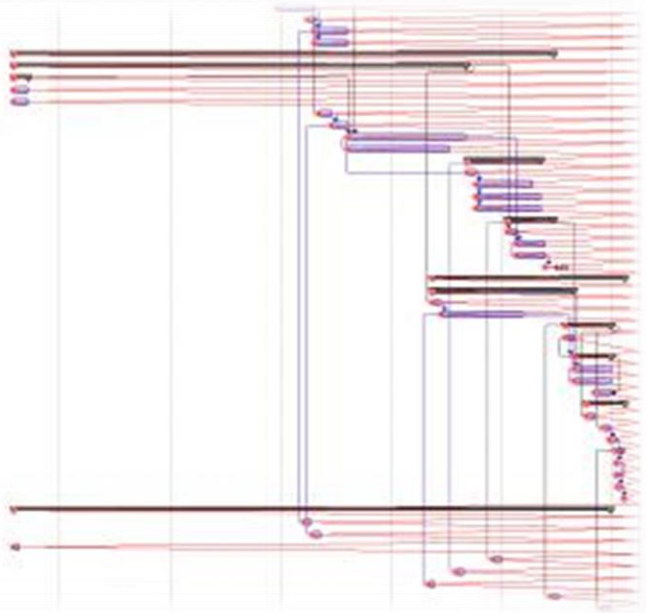
- 计算出FP的估算值
  - FP = (0.65 + 0.01×ΣFi)×CT = 372
- 历史数据
  - 平均生产率 6.5 FP/PM
  - 每个人月的成本 C = 8000￥(平均月薪)
- 估算成本和工作量
  - 工作量 58人月（**基于LOC的估算值54人月**）
  - 成本 457000￥（**基于LOC的估算值43 2000￥**）

8.4.5. 基于过程分解的估算

活动	客户沟通	策划	风险分析	工程		构造发布		客户评估	合计(人月)
任务→				分析	设计	编码	测试		
功能↓									
UICF				0.5	2.5	0.4	5.0	n/a	8.4
2DGA				0.75	4.0	0.6	2.0	n/a	7.35
3DGA				0.5	4.0	1.0	3.0	n/a	8.5
CGDF				0.5	3.0	1.0	1.5	n/a	6.0
DBM				0.5	3.0	0.75	1.5	n/a	5.75
PCF				0.25	2.0	0.5	1.5	n/a	4.25
DAM				0.5	2.0	0.5	2.0	n/a	5.0
合计	0.25	0.25	0.25	3.5	20.50	4.5	16.5		46.0
工作量	1%	1%	1%	8%	45%	10%	36%		







8.5. 基于经验的软件估算

- 基于回归分析的经验估算模型
- 基本COCOMO模型公式及计算方法
- 中间COCOMO模型公式及计算方法

8.5.1.基于回归分析的经验估算模型

通过对以往软件项目中搜集的数据进行回归分析而导出

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

其中A、B、C是经验常数，E是工作量（人月），e_v是估算变量（LOC或功能点）

面向规模的回归分析经验估算模型

$E=5.2\times(KLOC)^{0.91}$ Walston-Felix模型

$E=5.5+0.73\times(KLOC)^{1.16}$ Bailey-Basili模型

$E=3.2\times(KLOC)^{1.05}$ Boehm简单模型

$E=5.288\times(KLOC)^{1.047}$ Doty模型，用于KLOC>9的情况

8.5.1. 基于回归分析的经验估算模型

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

其中A、B、C是经验常数，E是工作量（人月），e_v是估算变量（LOC或功能点）

面向功能点的回归分析经验估算模型

E=-91.4+0.355FP　Albrecht和Gaffney模型

E=-37+0.96FP　Kemerer模型

E=-12.88+0.405FP　小型项目回归模型

8.5.2. COCOMO经验估算模型——概述

- COCOMO模型?
  - COCOMO是指COnstructive COst MOdel，构造性成本模型，Boehm于1981年提出，用于对软件开发项目的规模、成本、进度等方面进行估算
  - COCOMO模型是一个综合经验模型，模型中的参数取值来自于经验值，并且综合了诸多的因素、比较全面的估算模型
  - 在欧盟国家应用较为广泛

8.5.2. COCOMO经验估算模型——模型层次

- COCOMO模型的层次 - 支持不同的阶段
  - 基本COCOMO模型
    - 系统开发的初期，估算整个系统的工作量(包括维护)和软件开发和维护所需的时间
  - 中间COCOMO模型
    - 估算各个子系统的工作量和开发时间
  - 详细COCOMO模型
    - 估算独立的软构件，如各个子系统的各个模块的工作量和开发时间

8.5.3. COCOMO经验估算模型——基本模型

- 基本COCOMO模型
  - $E = a * (KLOC)^b$; E是工作量(人月)， a和b是经验常数
  - $D = c * E^d$;D是开发时间(月)， c和d是经验常数
  - 其中， a,b,c,d为经验常数，其取值见下表

软件类型	a	b	c	d	适用范围
组织型	2.4	1.05	2.5	0.38	各类应用程序
半独立型	3.0	1.12	2.5	0.35	各类编译程序等
嵌入型	3.6	1.20	2.5	0.32	实时软件、OS等

8.5.4. COCOMO经验估算模型——中间模型

- 中间COCOMO模型
  - $E = a * (KLOC)^b * EAF$
  - 其中，E表示工作量(人月)，EAF表示工作量调节因子，a,b为经验常数，其取值见下表

软件类型	a	b
组织型	3.2	1.05
半独立型	3.0	1.12
嵌入型	2.8	1.20

8.5.2. COCOMO经验估算模型——影响因子

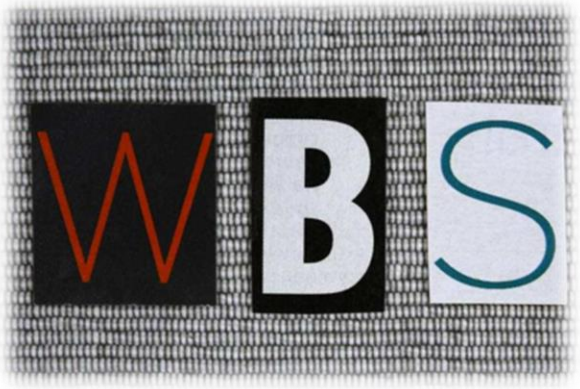
- EAF的取值(考虑15个因素)
  - 软件产品属性(3)：软件可靠性，软件复杂性，数据库的规模
  - 计算机属性(4)：程序执行时间，程序占用内存大小，软件开发环境的变化，软件开发环境的响应速度
  - 人员属性(5)：分析员能力，程序员能力，领域经验，开发环境的经验，程序设计语言的经验
  - 项目属性(3)：软件开发方法的能力，软件工具的数量和质量，软件开发的进度要求

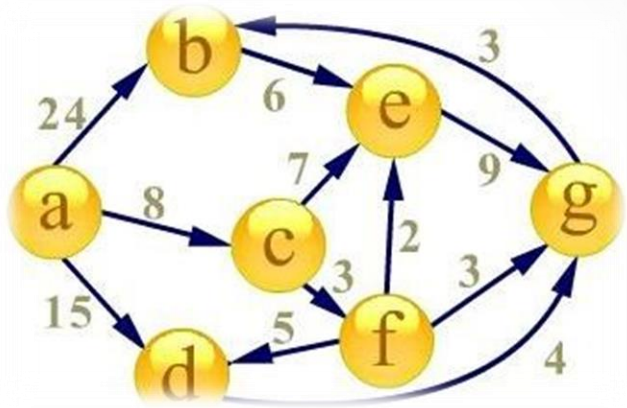
8.5.2. COCOMO经验估算模型——影响因子的取值

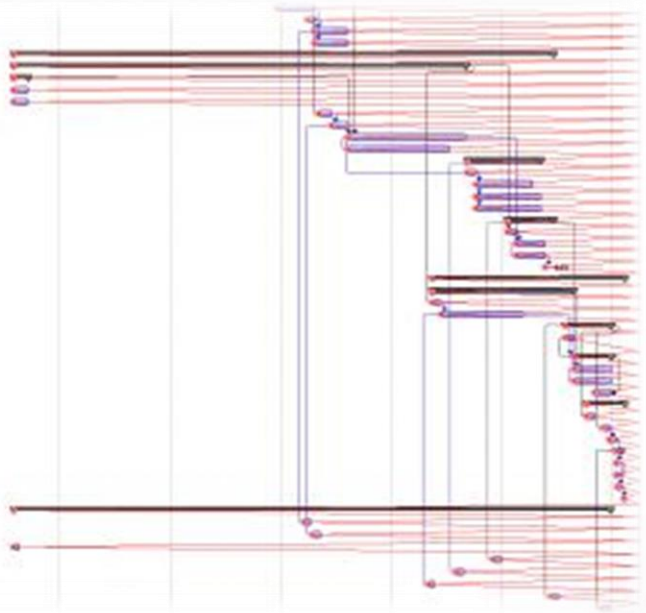
- EAF的取值(范围)
  - 很低、低、正常、高、很高、极高
  - Boehm建议取值范围[0.70-1.66]
  - EAF的计算 = $\prod F_i$ (i=1..15)
- 调节因子及其取值由统计结果和经验决定，不同的软件开发组织在不同的时期可能会有不同的取值

8.5.2. COCOMO经验估算模型——案例分析

- 案例分析：用基本COCOMO模型估算项目的工作量、开发时间和参加项目开发的人数
  - CAD软件：目标代码行33.2KLOC，属于中等规模，半独立型，因而a = 3.0, b = 1.12, c = 2.5, d = 0.35
  - $E = 3.0*(33.2)^{1.12}$ =152 PM
  - $D = 2.5*(152)^{0.35}$ = 14.5 (月)
  - 参加项目人数N = E/D = 152/14.5 = 11(人)







8.6. 项目进度计划

- 项目进度计划概念
- 甘特图
- 里程碑

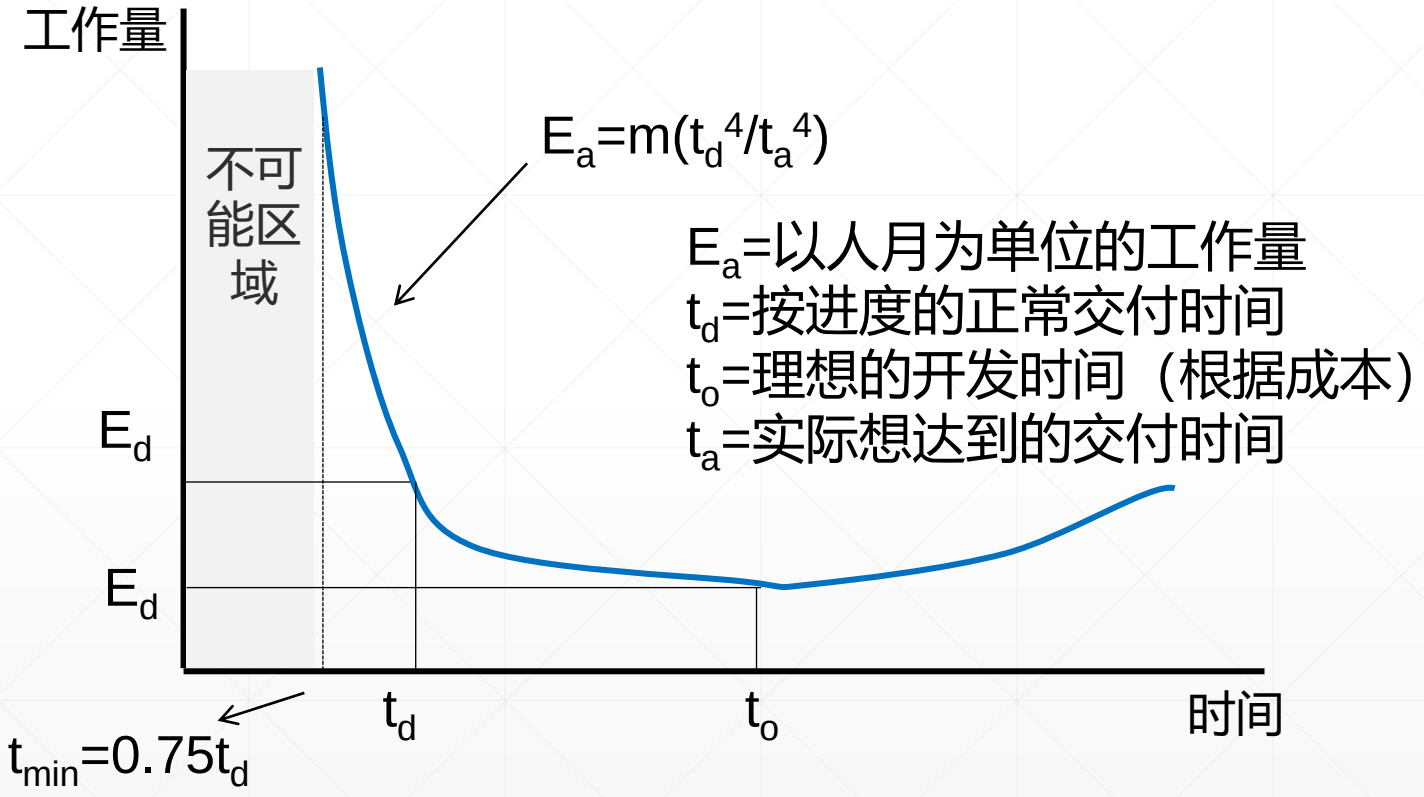
8.6.1. 项目进度计划概念

定义：对项目进行任务划分，定义任务之间的依赖关系，并进行时间估算和资源分配，确保以最佳的时间与成本输出满足质量要求的产品。

编制项目计划本质是一个优化问题。

8.6.1. 项目进度计划概念

表示软件项目工作量(成本)与开发时间之间的PNR曲线



项目进度与成本之间是一个非线性关系

最低成本的交付时间应该为正常交付时间的两倍左右

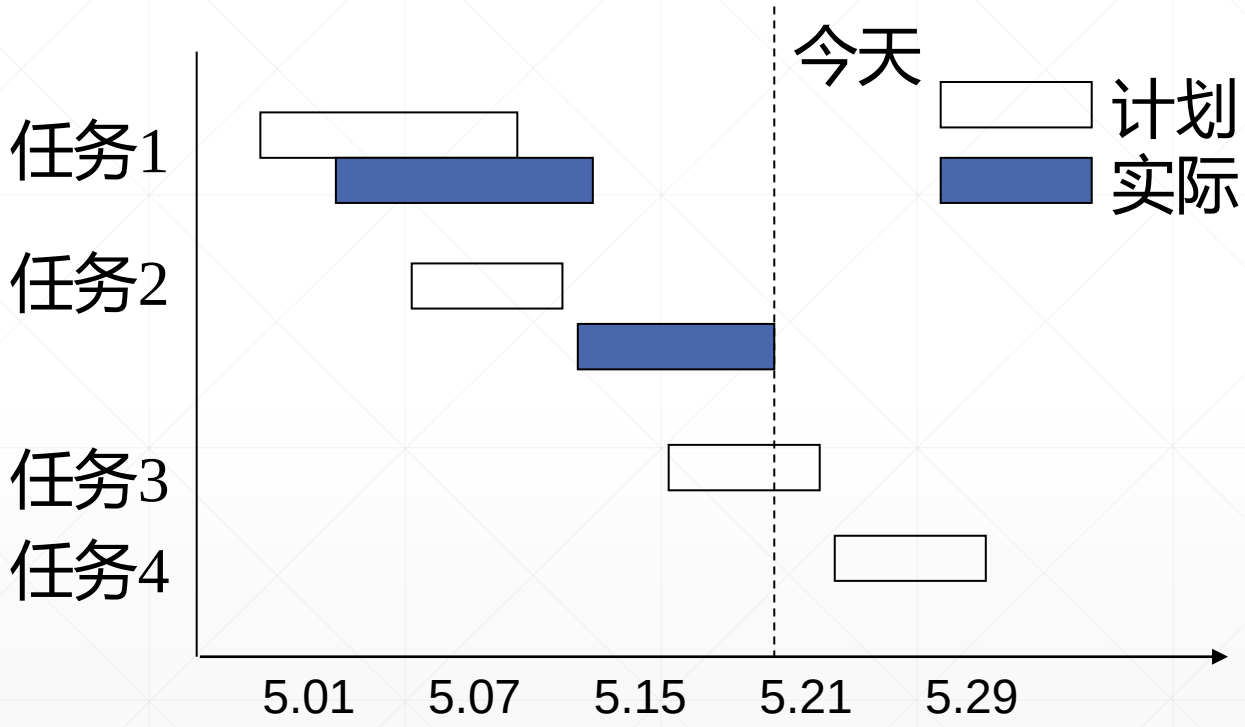
8.6.2. 项目进度计划的价值

- 有序、可控制地对软件项目进行管理
- 确保员工保持高生产率
- 及时交付软件产品
- 降低软件开发成本
- 提高客户满意度
- 及时发布产品新版本

8.6.3. 项目进度计划的可视化

甘特图

- 显示基本的任务信息
- 定义并查看任务的工期、开始时间和结束时间
- 定义并查看任务所分配的资源的信息
- 可定义任务间的前后关系



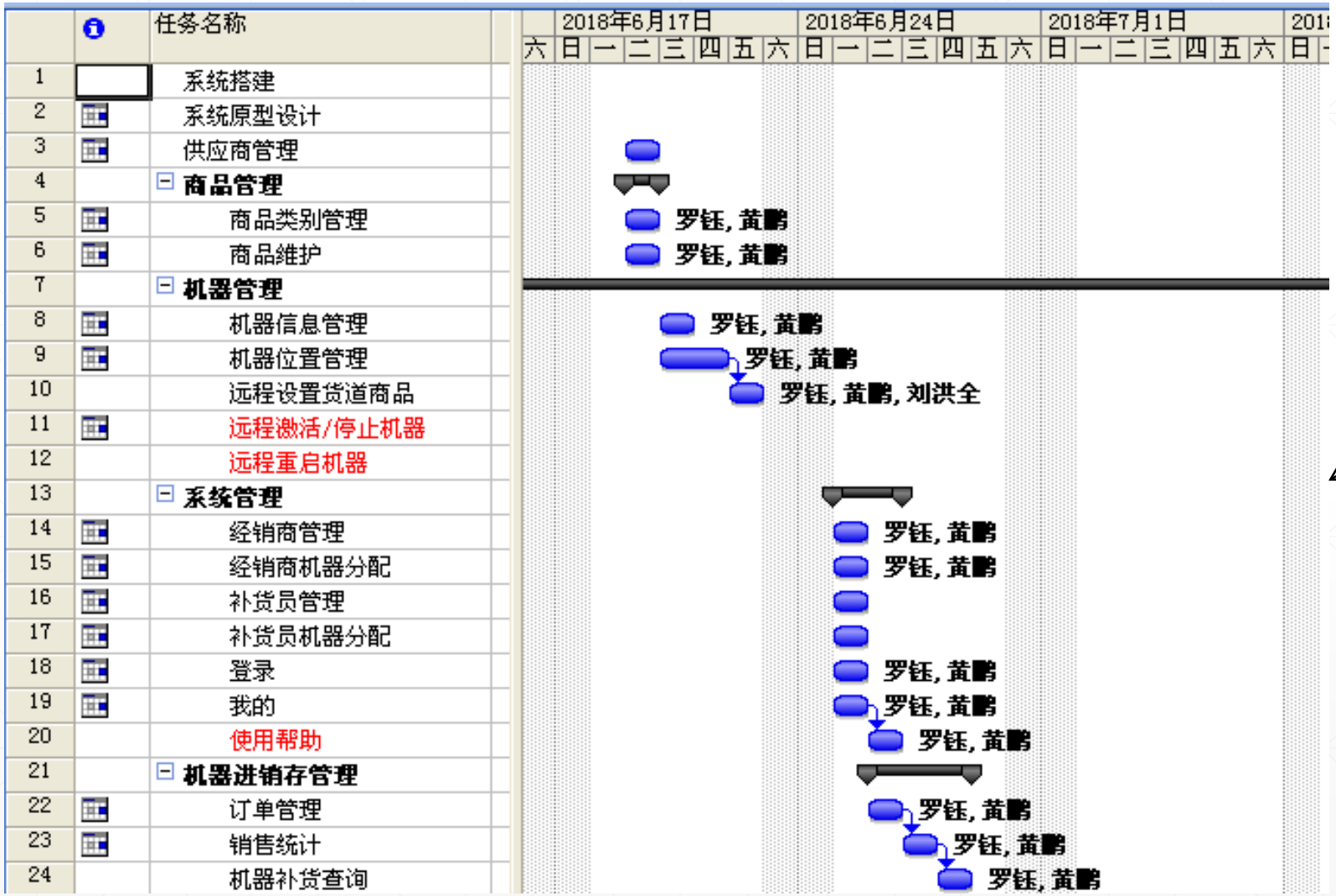
纵轴表示任务，横轴表示时间

8.6.4. 项目进度计划的可视化-微软的Project软件

任务名称　　工期　　开始时间　　结束时间　前置任务　资源名称

		任务名称	工期	开始时间	完成时间	前置任务	资源名称
1		系统搭建	1 工作日	2018年6月4日	2018年6月4日		
2		系统原型设计	9 工作日	2018年6月5日	2018年6月15日	1	
3		供应商管理	1 工作日	2018年6月19日	2018年6月19日		
4		**商品管理**	**1 工作日**	**2018年6月19日**	**2018年6月19日**		
5		商品类别管理	1 工作日	2018年6月19日	2018年6月19日		罗钰, 黄鹏
6		商品维护	1 工作日	2018年6月19日	2018年6月19日		罗钰, 黄鹏
7		**机器管理**	**35 工作日**	**2018年6月4日**	**2018年7月19日**		
8		机器信息管理	1 工作日	2018年6月20日	2018年6月20日		罗钰, 黄鹏
9		机器位置管理	2 工作日	2018年6月20日	2018年6月21日		罗钰, 黄鹏
10		远程设置货道商品	1 工作日	2018年6月22日	2018年6月22日	9	罗钰, 黄鹏, 刘洪全
11		远程激活/停止机器	3 工作日	2018年7月17日	2018年7月19日		罗钰, 黄鹏, 刘洪全
12		远程重启机器	2 工作日	2018年6月4日	2018年6月5日		罗钰, 黄鹏, 刘洪全
13		**系统管理**	**2 工作日**	**2018年6月25日**	**2018年6月26日**		
14		经销商管理	1 工作日	2018年6月25日	2018年6月25日		罗钰, 黄鹏
15		经销商机器分配	1 工作日	2018年6月25日	2018年6月25日		罗钰, 黄鹏
16		补货员管理	1 工作日	2018年6月25日	2018年6月25日		
17		补货员机器分配	1 工作日	2018年6月25日	2018年6月25日		
18		登录	1 工作日	2018年6月25日	2018年6月25日		罗钰, 黄鹏
19		我的	1 工作日	2018年6月25日	2018年6月25日		罗钰, 黄鹏
20		使用帮助	1 工作日	2018年6月26日	2018年6月26日	19	罗钰, 黄鹏
21		**机器进销存管理**	**3 工作日**	**2018年6月26日**	**2018年6月28日**		
22		订单管理	1 工作日	2018年6月26日	2018年6月26日		罗钰, 黄鹏
23		销售统计	1 工作日	2018年6月27日	2018年6月27日	22	罗钰, 黄鹏
24		机器补货查询	1 工作日	2018年6月28日	2018年6月28日	23	罗钰, 黄鹏

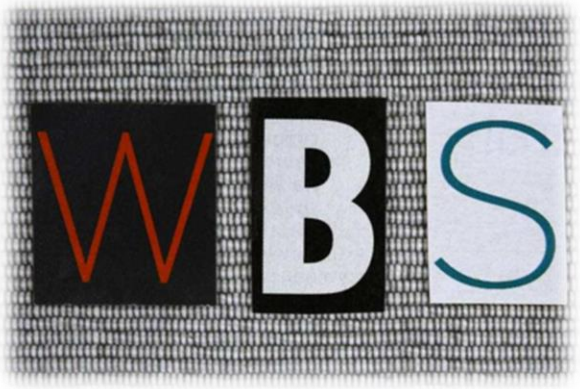
8.6.4. 项目进度计划的可视化-微软的Project软件

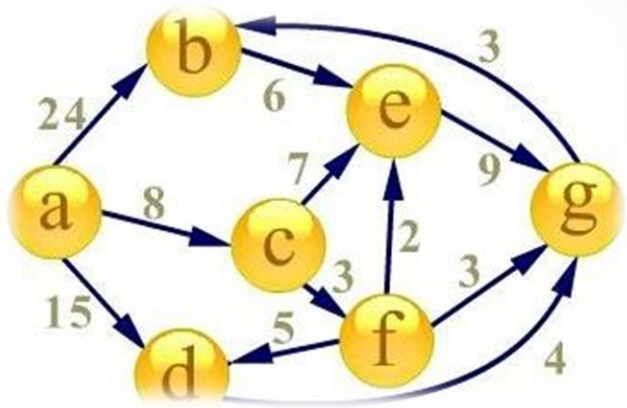


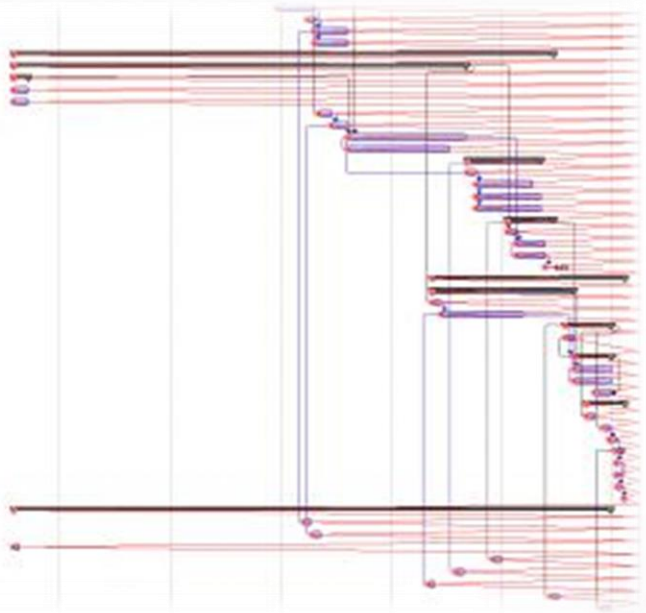
进度条模式

8.6.5. 里程碑

- 里程碑显示项目进展中的重大工作完成
- 里程碑不同于活动
  - 活动是需要消耗资源的
  - 里程碑仅仅表示事件的标记





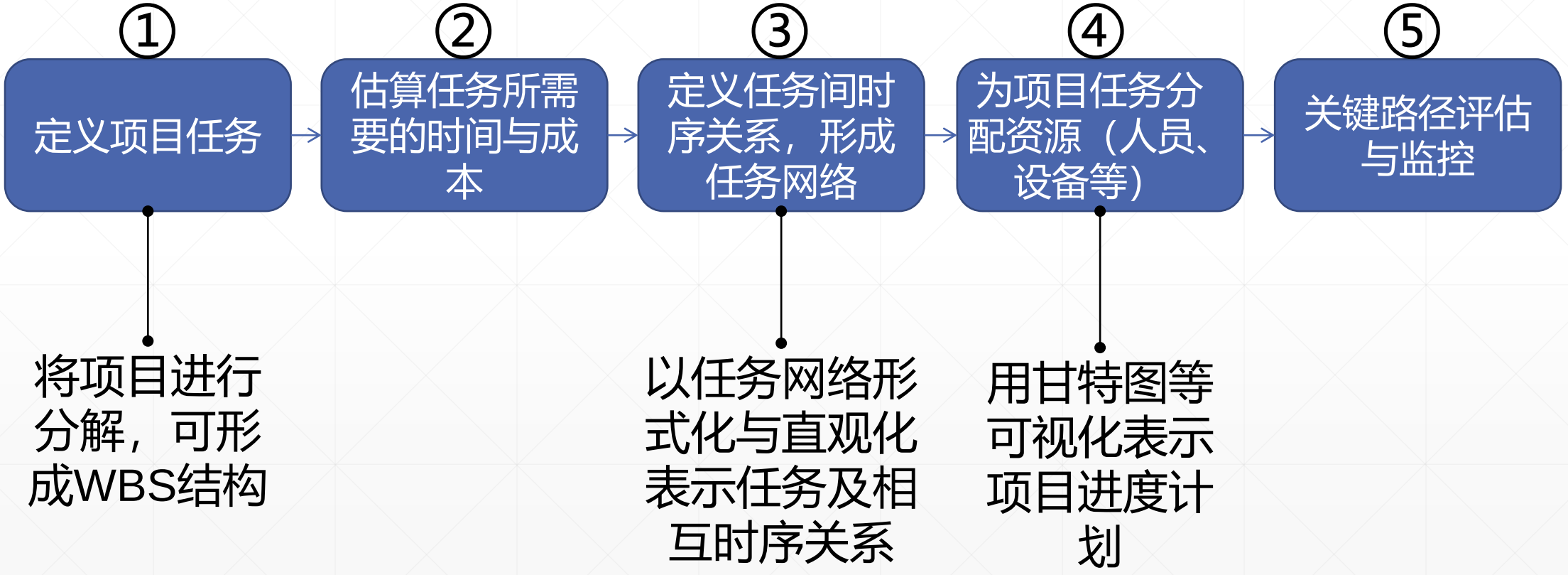


8.7. WBS分解与任务网络图

- 项目进度计划编制过程
- WBS分解
- 任务网络图

8.7.1. 项目进度计划编制过程

编制项目进度计划的步骤



8.7.2. 工作分解结构WBS——定义与作用

定义：

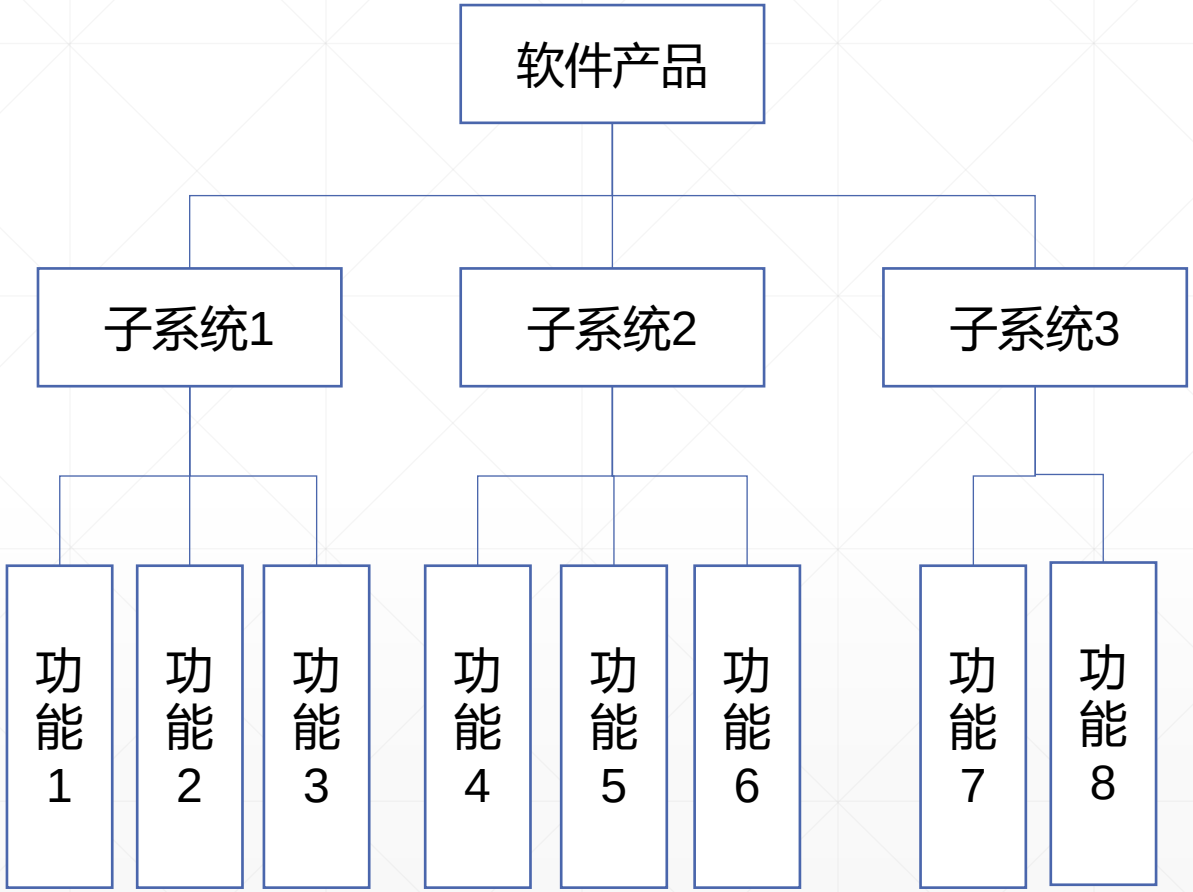
工作分解结构（Work Breakdown Structure, WBS）是将项目按照功能或过程进行逐层分解，直到划分为若干内容单一、便于组织管理的单项工作，最终形成的树形结构示意图。

作用：

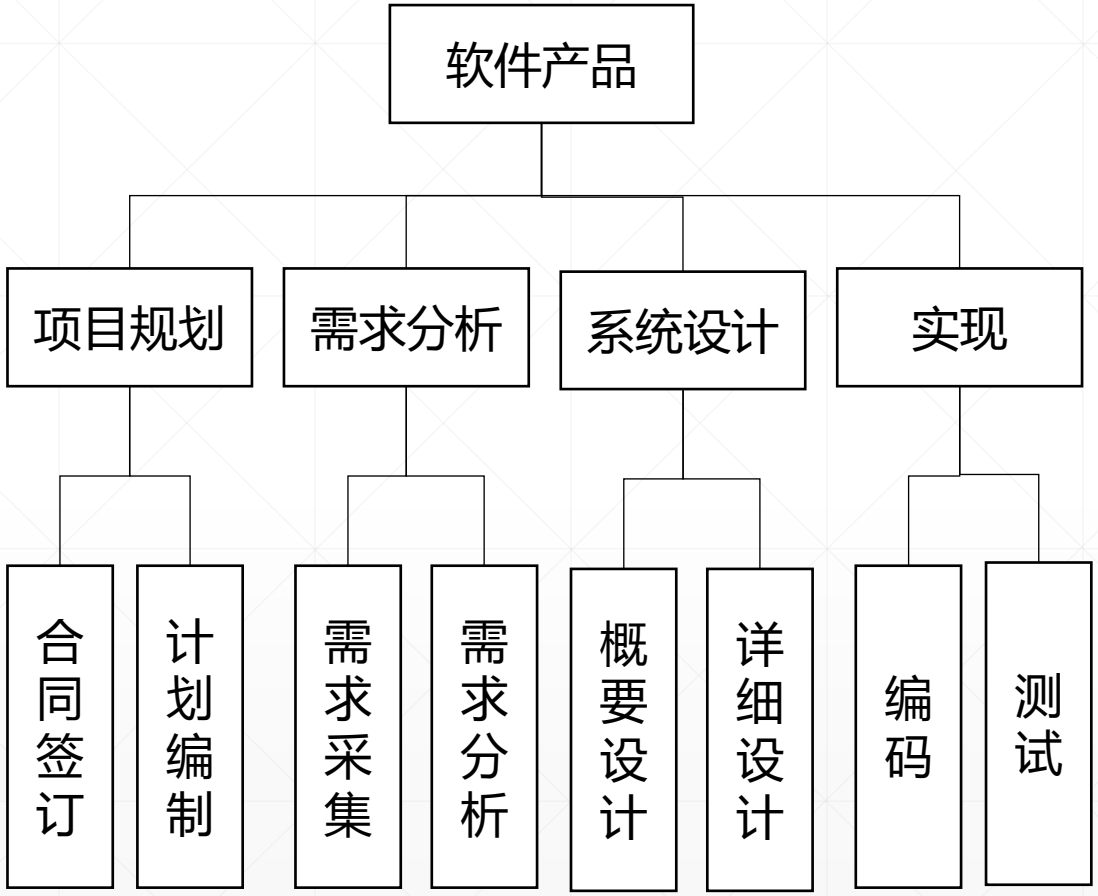
- 相关成员可直观了解软件项目中的各项任务（活动）
- 将项目分解为可管理的任务（活动）
- 作为项目计划与跟踪的基础

8.7.2. 工作分解结构WBS——分解模式

两种分解模式



基于功能的分解



基于过程的分解

8.7.2. 工作分解结构WBS——构建原则

WBS构建应该注意的原则：

- 一个任务只应该在WBS中的一个地方出现
- WBS中某项任务的内容是其下所有WBS项的总和
- 一个WBS项只能由一个人责任，其他人只能是参与者
- WBS必须与实际工作中的执行方式一致
- 应让项目团队成员积极参与创建WBS，以确保WBS的一致性
- 每个WBS项都必须文档化，以确保准确理解已包括和未包括的工作范围
- WBS可以根据需求进行必要变更维护

8.7.3. 任务网络图

定义：

任务网络图是项目所有任务（活动）及其之间逻辑关系（依赖关系）的一个图解表示，并从左到右来表示项目的时间顺序。

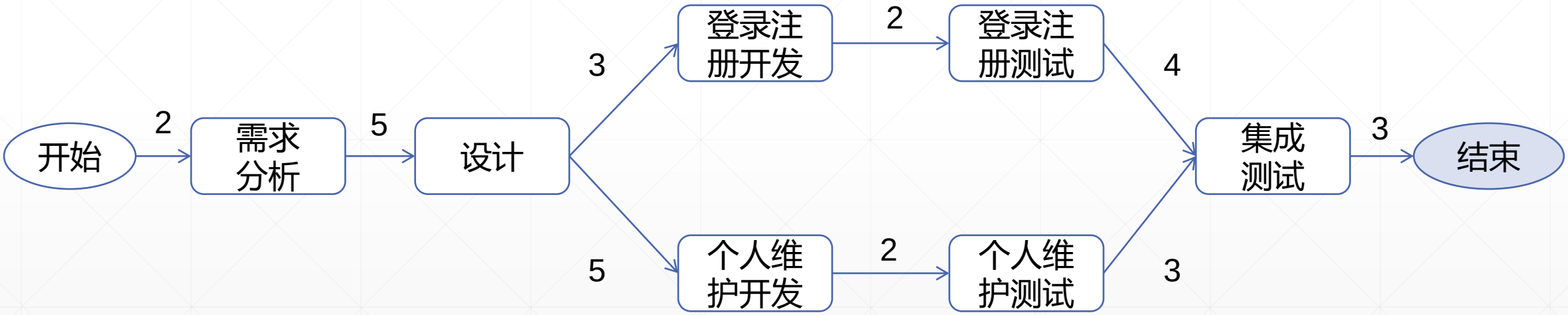
作用：

- 可以分解任务以及各项任务所需要耗费的时间及成本
- 可以显式的描绘各个任务间的时序依赖关系

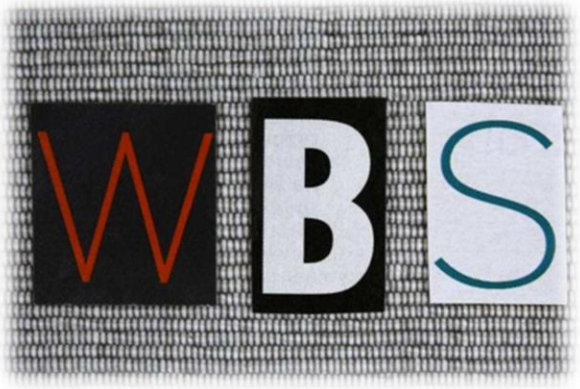
8.7.3. 任务网络图

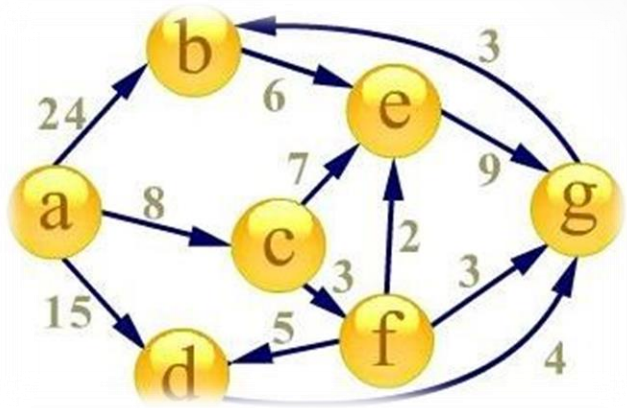
构成：

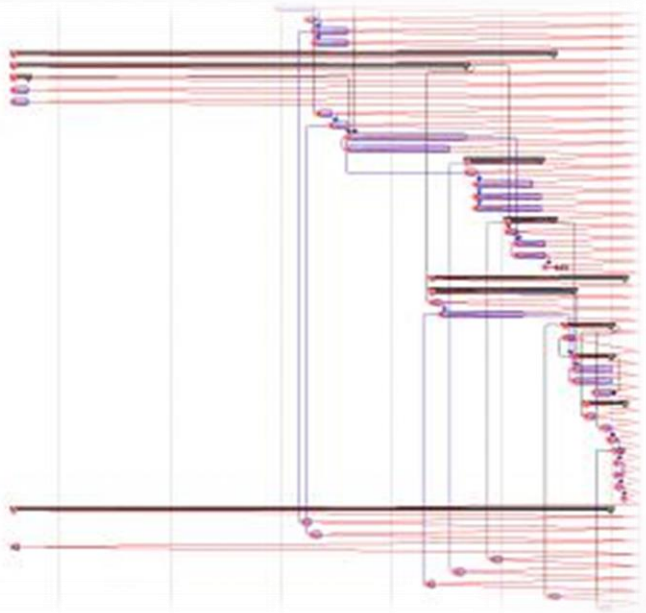
任务网络图是一个有向权重网络图，一般用节点表示事件，弧表示任务（活动），弧上的权值表示任务（活动）耗费的时间



举例：个人中心模块开发任务网络图，节点表示任务的开始，弧权重表示时间







8.8. 关键路径

- 概念
- 意义
- 可用资源对关键路径的影响

8.8.1 关键路径——概念

关键路径（critical path）：

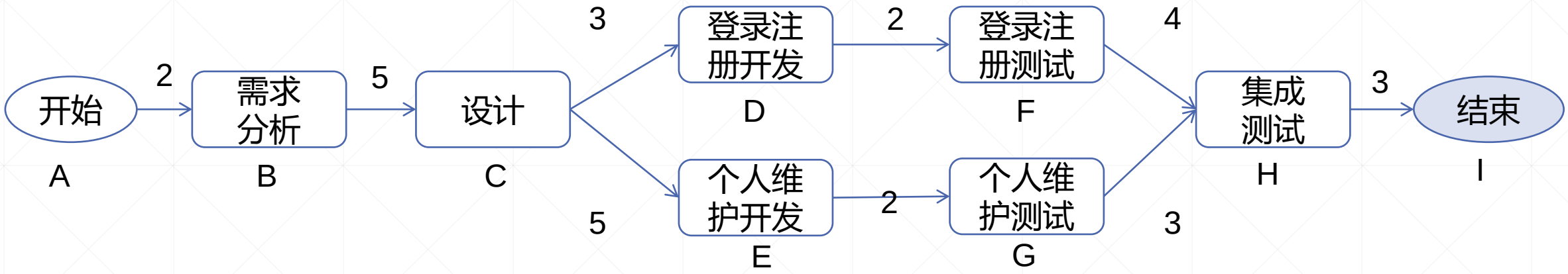
在任务网络图中，从项目开始到项目完成有许多条路径，路径上所有弧权重之和最大的路径（路径最长）叫关键路径。

非关键路径（noncritical path）：

在整个任务网络图中非最长的路径都叫非关键路径。

8.8.1. 关键路径——概念

任务网络图的关键路径例子（假定时间为天）



(a) 此网络中有多少路径? 两条：ABCDFHI, ABCEGHI

(b) 每条路径的长度是多少? ABCDFHI=19天，ABCEGHI=20天

(c) 哪一条是关键路径? ABCEGHI

(d) 完成项目的最短时间是多少? 20天

8.8.2. 关键路径的意义

- 关键路径上任何任务（活动）的延长都会导致整个项目周期的延长
- 如果想缩短项目周期，就必须缩短关键路径的长度
- 项目经理应该随时关注关键路径上任务（活动）的完成情况以及关键路径是否发生了变化
- 对WBS中任务的串行与并行安排方式有指导意义

8.8.3 可用资源对项目计划与关键路径的影响

例子：

有一个停车管理软件需要开发，包含三个功能：停车位管理、停车收费管理、人员管理。

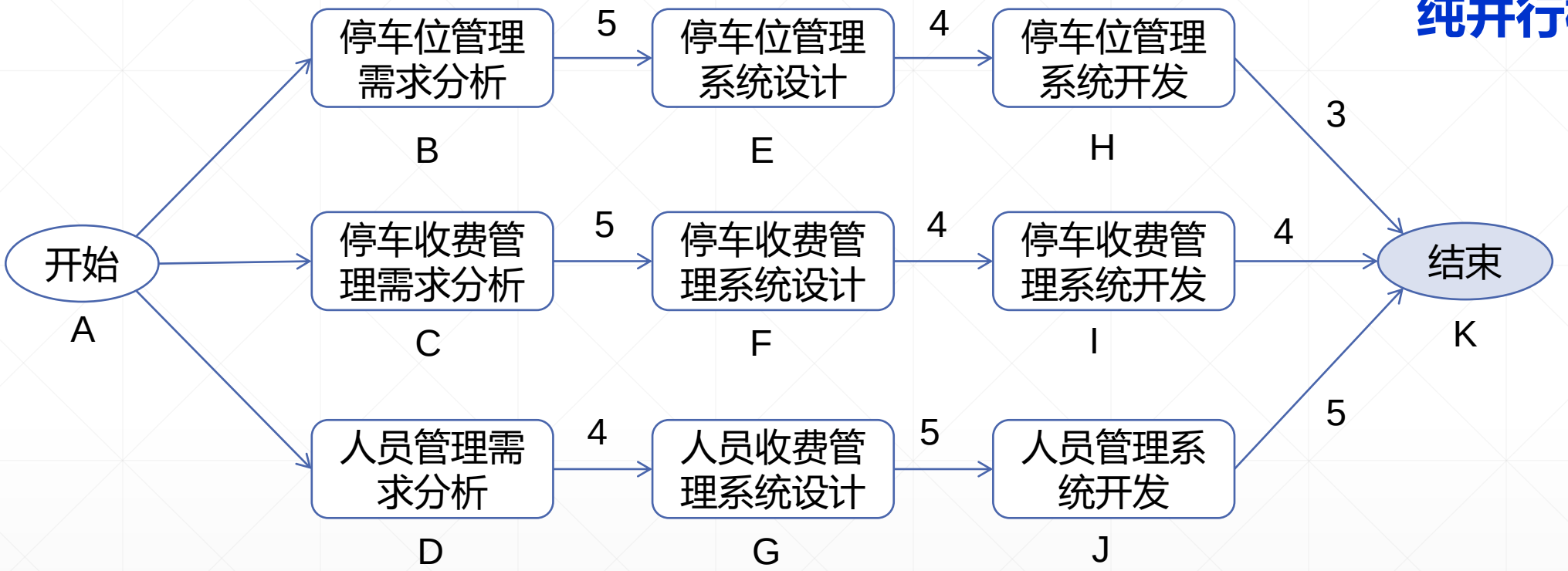
每个功能都需要经过三个活动：需求分析、系统设计、系统开发，假定这三个功能在这三个活动上花费的时间分别为（5天、4天、3天），（5天、4天、4天），（4天、5天、5天）

有三个工程师：一个需求分析员、一个软件设计师、一个程序员

如何安排此项目活动比较好?

8.8.3 可用资源对项目计划与关键路径的影响

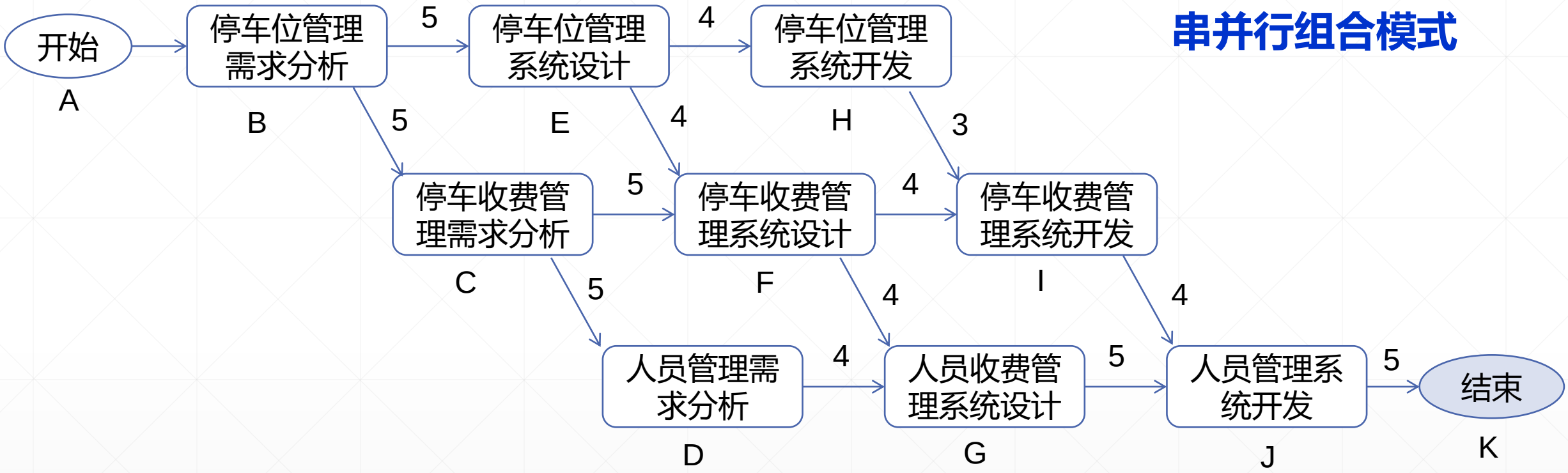
纯并行模式



此网络图的关键路径为ADGJK=14天

但不合理，因为需要三个需求分析员、三个软件设计师、三个程序员

8.8.3 可用资源对项目计划与关键路径的影响



串并行组合模式

共有6条路径：ABEHIJK=21天，ABEFIJK=22天，ABEFGJK=23，

ABCFIJK=23，ABCFGJK=24，ABCDGJK=24 **合理！**

此网络图的关键路径为ABCFGJK和ABCDGJK，均为24天

感谢观看！

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com