2021软件工程背诵

软件工程概述

==什么是软件开发过程？==

> 与软件过程的概念不同，软件开发过程（Software Development Process）是把用户要求转换为软件需求，把软件需求转化为设计，用代码来实现设计，对代码进行测试，完成文档编制并确认软件可以投入运行使用的过程。

什么是软件?

> 软件=程序+数据+文档
>
> 程序：按事先设计的功能和性能需求执行的指令序列
>
> 数据：是程序能正常操作信息的数据结构
>
> 文档：与程序开发、维护和使用有关的图文材料

什么是软件工程

> 软件工程是指导计算机软件开发和维护的工程学科，它采用工程的概念、原理、技术和方法来开发与维护软件。软件工程的目标是实现软件的优质高产。软件工程的目的是在规定的时间和开发费用内，开发出满足用户需求的、高质量的软件产品。

什么是软件危机

> 软件危机是指在计算机软件开发和维护过程中碰到问题的集合。

软件危机产生的原因

> 客观：
>
> 规模庞大
>
> 主观：
>
> - 忽视软件需求分析的重要性，对软件可维护性不重视
> - 软件一般要使用5～10年，在这段时间里，很可能出现开发时没有预料到的问题
> - 生产方式和开发工具落后。

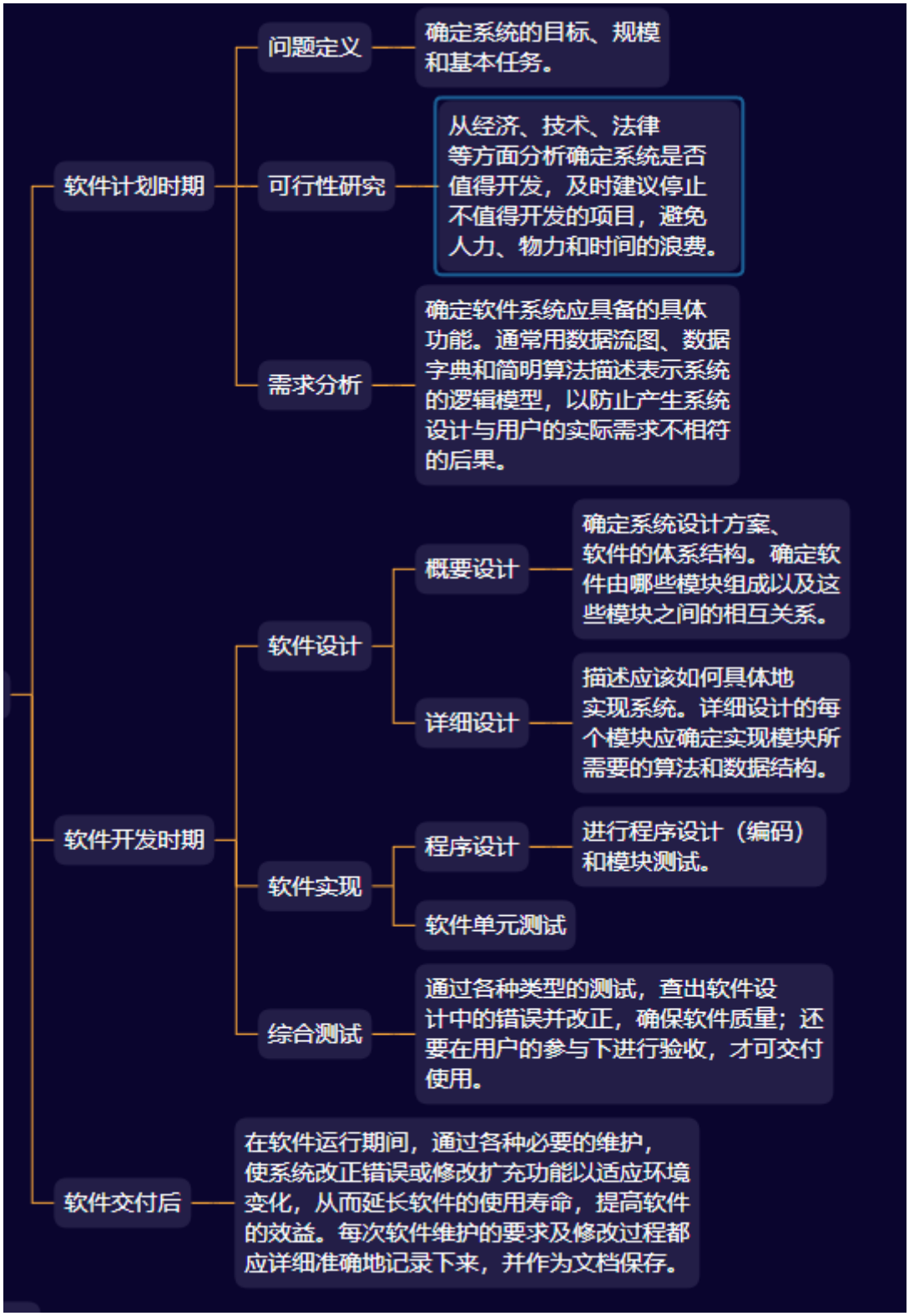
软件危机的表现形式

> - 软件的发展速度跟不上硬件的发展和用户的需求。
> - 软件的成本和开发进度不能预先估计，用户不满意。
> - 软件产品质量差，可靠性不能保证。
> - 软件产品可维护性差。
> - 软件没有合适的文档资料。

解决软件危机的途径

> - 使用好的软件开发技术和方法。
> - 开发软件时有良好的组织、严密的管理，各方面人员相互配合共同完成任务。
> - 使用好的软件开发工具，提高软件生产率。

==软件生存周期的内容==

> 
>
>
> 需求分析 概要设计 详细设计
>
> 软件开发时期包含哪几个主要阶段

软件过程

什么是软件生命周期?

> 软件产品或软件系统从设计、投入使用到被淘汰的全过程

什么是软件过程?

> **软件过程**是在工作产品构建过程中，所需完成的工作活动、动作和任务的集合。

什么是软件过程模型?

> 是软件开发全部过程、活动和任务的结构框架。它能直观表达软件开发全过程，明确规定要完成的主要活动、任务和开发策略。

==瀑布模型==

> 定义：
>
> - 按照软件生命周期的各个阶段，依次向下，逐步求精的方式完成软件项目
> - 每一阶段都有每一阶段明确的任务，每一阶段的完成都有相应的文档生成
>
> 优点：
>
> - 能够逐步稳定的使项目向前发展。
>
> 缺点：
>
> - 增加工作量
> - 开发风险大
> - 早期错误发现晚
> - 不适应需求变化
>
> 适用场合：
>
> - 瀑布模型适用于系统需求明确且稳定、技术成熟、工程管理较严格的场合，如军工、航天、医疗。

==原型模型==

> 原型化的目的：
>
> - 明确并完善需求，如演示原型
> - 研究技术选择方案，如技术验证原型
>
> 优点：
>
> - 减少需求不明带来的风险
>
> 缺点：
>
> - 客户意识不到一些质量问题
> - 设计者在质量和原型中进行折中
> - 快速建立起来的系统加上连续的修改可能导致原型质量低下
> - 构造原型采用的技术和工具不一定主流
>
> 适用场合：
>
> - 客户定义一个总体目标集，但是不知道系统的具体输入输出
> - 开发者不确定算法的效率、软件与操作系统是否兼容以及客户与计算机交互的方式

==增量模型==

> 优点：
>
> - 在项目的初始阶段不需要投入太多的人力资源
> - 开放式体系结构，便于维护
> - 增量概念的引入，不需要提供完整的需求，只要有一个增量，开发就可以进行
> - 软件能够更早的投入市场
> - 产品逐步交付，能够较好的适应需求的变化
> - 能看到软件中间产品，提出改进意见，减少返工，降低开发风险
>
> 缺点：
>
> - 软件必须具备开放式体系结构
> - 已退化成边做边改的方式，使软件过程控制失去整体性
> - 每个增量必须提供一些系统功能，使开发者很难根据客户需求给出大小合适的增量
>
> 适用场合：
>
> - 适用于软件开发中需求可能发生变化，具有较大风险，或者希望尽早进入市场的项目

==螺旋模型==

> 优点：
>
> - 利于把软件质量作为软件开发目标
> - 减少测试
> - 维护和开发不分开
>
> 缺点：
>
> - 风险估计困难
>
> 适用场合：
>
> - 适用于需求不明确或者需求可能发生变化的大型复杂的软件系统
> - 支持面向过程、面向对象等多种软件开发方法，是一种具有关阔前景的模型

喷泉模型 优点 缺点 适用场合

> 定义：
>
> 喷泉模型是一种以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程
>
> 优点：
>
> - 各个阶段没有明显的界限，开发人员可以同步进行开发
> - 可以提高软件项目的开发效率，节省开发时间
> - 适应于面向对象的软件开发过程
>
> 缺点：
>
> - 由于喷泉模型在各个开发阶段是重叠的，在开发过程中需要大量的开发人员，因此不利于项目的管理
> - 要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入的各种信息、需求和资料的情况
>
> 适用场合：
>
> 面向对象开发

基于构建的开发模型

> 优点：
>
> - 软件复用思想
> - 降低开发成本和风险，加快开发进度，提高软件质量
>
> 缺点：
>
> - 模型复杂
> - 商业构件不能修改，会导致修改需求，进而导致系统不能完全符合客户需求无法完全控制所开发系统的演化
> - 项目划分的好坏直接影响项目结果的好坏
>
> 适用场合：
>
> - 适用于系统之间有共性的情况。

敏捷模型

> 优点：
>
> - 快速响应变化和不确定性
> - 可持续开发速度
> - 适应商业竞争环境下的有限资源和有限时间
>
> 缺点：
>
> - 测试驱动开发可能导致通过测试但非用户期望
> - 重构而不降低质量困难
>
> 适用场合：
>
> - 适用于需求模糊且经常改变的场合
> - 适合商业竞争环境下的项目

==如何选择过程模型==

> - 前期需求明确的情况下，尽量采用瀑布模型
> - 用户无系统使用经验，需求分析人员技能不足的情况下，尽量采用原型模型
> - 不确定因素很多，很多东西无法提前计划的情况下，采用增量模型或者螺旋模型
> - 需求不稳定的情况下，尽量采用增量模型
> - 资金和成本无法一次到位的情况下，采用增量模型
> - 对于完成多个独立功能开发的情况，可在需求分析阶段进行功能并行，每个功能内部都尽量遵循瀑布模型
> - 全新系统的开发必须在总体设计完成后再开始增量或者并行
> - 编码人员经验较少的情况下，尽量不要采用敏捷或者迭代模型
> - 增量、迭代和原型可以综合使用，但是每一次增量或迭代都必须有明确的交付和出口原则

软件项目管理

A 软件项目管理

软件项目管理定义

> 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4p要素

> 人员 产品 项目 过程

B 软件项目度量

==软件项目度量的目的==

> 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。

面向规模的度量

> 定义：
>
> 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。
>
> KLOC：千行代码
>
> PM：生产率
>
> CKL：每千行代码的平均成本
>
> EQRI：代码出错率
>
> DI：文档与代码比
>
> 优点：
>
> - 简单易行，自然美观
>
> 缺点：
>
> - 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
> - 依赖于程序设计语言的表达能力和功能
>
> 例题：

16.2.4. 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

面向功能的度量

定义：

用软件的功能表示软件的规模

优点：

- 与程序设计语言无关
- 项目开发初期就可估算出

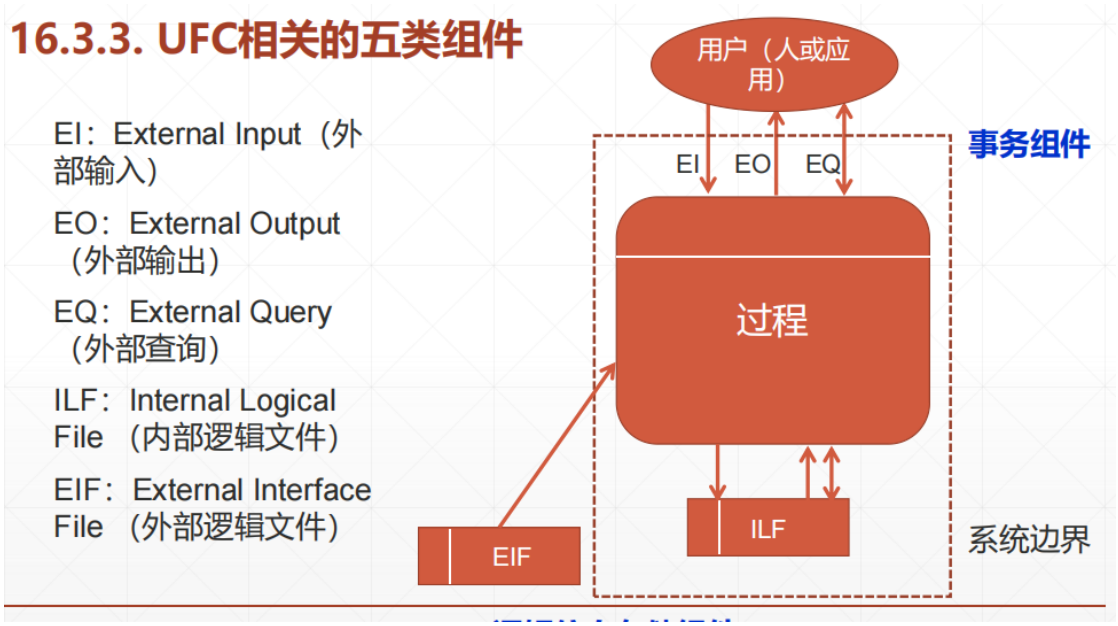
缺点：

- 功能点计算目前主要基于经验公式，主观因素比较多
- 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

功能点计算方法：

- FP = UFC×TCF = UFC × (0.65 + 0.01×ΣFi)
  - 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。

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

C 软件项目估算

> 定义：
>
> 项目启动之前，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，也就是对这些内容进行预测
>
> 策略：
>
> - 根据已经完成的项目进行估算
> - 尽量把估算延迟到项目的后期进行
> - 项目度量方法为项目估算提供了依据和有效输入

基于分解

基于问题分解

> 三点值期望法：
>
> 估计期望值=（最大值+最小值+4*最可能值）/6

基于过程分解

16.4.5. 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	16.4
2DGA				0.75	4.0	0.6	2.0	n/a	7.35
3DGA				0.5	4.0	1.0	3.0	n/a	16.5
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%		

基于经验

基于回归分析

> 定义：
>
> 通过对以往软件项目中搜集的数据进行回归分析而导出

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

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

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

==COCOMO模型(计算)==

> 定义：
>
> 构造性成本模型，用于对软件开发项目的规模、成本、进度等方面进行估算
>
> 基本COCOMO：
>
> 定义：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

- 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

中间COCOMO：

定义：估算各个子系统的工作量和开发时间

- 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

详细COCOMO：

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

例题：

16.5.2. 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)

D 项目进度计划

概念：

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

价值：

- 有序的、可控制的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

可视化：

- 甘特图

- Project软件

里程碑：

E WBS分解与任务网络图

编制项目进度计划的步骤：

- 定义项目任务
- 估算任务所需要的时间与成本
- 定义任务间时序关系，形成任务网络
- 为项目任务分配资源
- 关键路径评估与监控

WBS

定义：

工作分解结构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

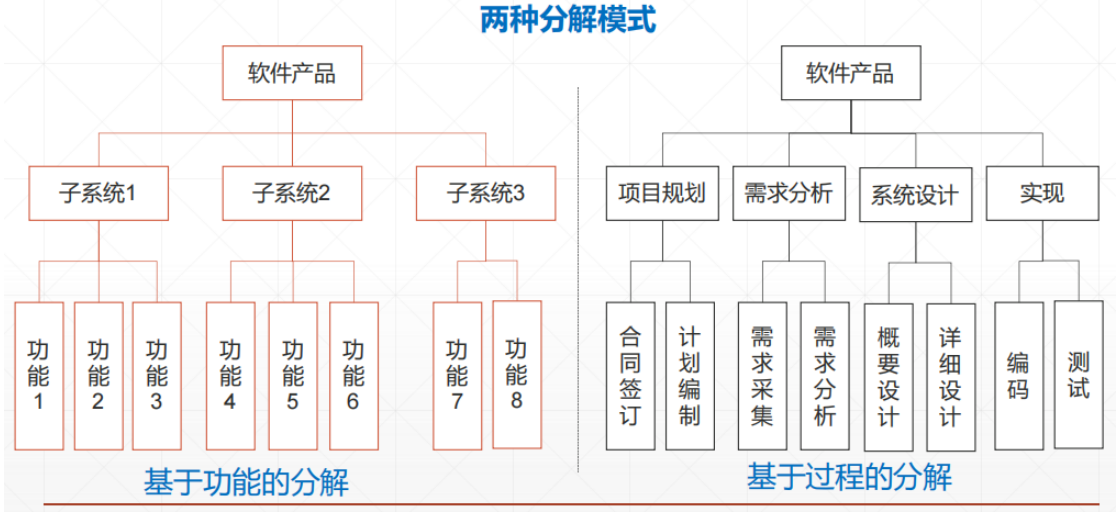
作用：

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

两种分解模式：

- 基于功能
- 基于过程

16.7.2. 工作分解结构WBS——分解模式



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

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

F关键路径

概念：

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。

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