



软件项目管理

@author：韦邦浩

17周周五（12.24）3.4节期末考试

一、项目管理概念

1、项目与软件项目

组织的活动逐步分化为两种类型：

1. 连续不断、周而复始的活动，人们称之为“运作”(Operations)，如企业日常的生产产品的活动
2. **临时性、一次性的活动，人们称之为“项目”(Projects)**，如企业的技术改造活动、一项环保工程的实施。

a、项目定义：

由一组有起止日期的、相互协调的受控活动所组成的独特过程，该过程要达到符合包括时间、成本和资源的约束条件在内的规定要求的目标。

项目是为了创造一个唯一的产品或提供一个唯一的服务而进行的临时性的努力。**例如：**

- 建造一座大楼、一座工厂或一座水库
- 举办各种类型的活动，如一次会议、一次旅行、一次晚宴、一次庆典和体育转播等
- 新企业、新产品、新工程的建设和开发
- 城市道路设施建设——厦门翔安隧道、BRT
- 某社区领导选举
- 博导带领研究生解决某个研究课题
- 新建网络系统或开发一套管理软件
- 实施一种全新的经营程序或流程

b、项目与日常运作的区别：

1. 项目是**一次性的**，日常运作是**重复进行**的，
2. 项目是以**目标**为导向的，日常运作是通过效率和有效性体现的，
3. 项目是通过项目经理及其团队工作完成的，而日常运作是职能式的线性管理；
4. 项目存在大量的变更管理，而日常运作则基本保持连贯性的。

c、项目管理三要素：时间、成本、质量

2、项目管理与软件项目管理

a、项目的特征：

1. 有明确的目标
2. 项目之间的活动具有相关性
3. 限定的周期
4. 有独特性
5. 资源成本的约束性
6. 项目的不确定性

b、软件项目的特殊性：

1. 逻辑实体
2. 相互作用的系统
3. 渐近明细
4. 变更

3、项目管理知识体系（PMBOK）

- 1.项目范围管理
- 2.项目时间管理
- 3.项目成本管理
- 4.项目质量管理
- 5.项目风险管理
- 6.项目组织管理
- 7.项目人力资源管理
- 8.项目采购管理
- 9.项目沟通管理

4、过程管理与软件项目管理的关系

a、过程管理：

对过程进行管理，目的是要让**过程能够被共享、复用**，并得到持续的改进。

b、软件过程管理：

注重循序渐进地积累，**积累**项目中的各个环节的**实践经验**和项目管理的**实践经验**，保证我们的生产力持续地发展。

c、过程管理和项目管理关系：

- 项目管理用于**保证项目的成功**
- 过程管理用于管理**最佳实践**
- 这两项管理不是相互孤立的，而是有机地**紧密地结合**的

5、软件项目管理过程

a、项目初始

1. 项目确立

2. 生存期

b、项目计划

1. 范围计划
2. 成本计划
3. 进度计划
4. 质量计划
5. 配置管理计划
6. 人员与沟通计划
7. 风险计划
8. 合同计划
9. 集成计划

c、项目执行控制

1. 集成计划执行控制
2. 核心计划执行控制
3. 辅助计划执行控制

d、项目结束

5、高效的软件项目管理集中于四个P上：人员、产品、过程、项目（人产过项）

1. **People** — the most important element of a successful project
人员—— 一个成功项目最重要的元素
2. **Product** — the software to be built
产品—— 建立的软件
3. **Process** — the set of framework activities and software engineering tasks to get the job done
过程—— 框架活动集和完成工作的软件工程任务
4. **Project** — all work required to make the product a reality
项目—— 所有需要做的工作，以使一个产品变成现实

People 人员

(1) The Stakeholders利益相关者

参与软件过程（和每个软件项目）的利益相关者可以被分为五类：

1. 高级管理者：负责定义业务问题，这些问题往往对项目产生很大影响
2. 项目 (技术) 管理者： 必须计划、激励、组织和控制软件开发人员。
3. 开发人员： 拥有开发产品或应用软件所需技能的人员。
4. 客户： 阐明待开发软件需求的人员以及关心项目成败的其他利益相关者。
5. 最终用户： 一旦软件发布成为产品，最终用户就是直接与软件进行交互的人。

(2) Team leader 团队负责人

(3) Software Teams软件团队

二、项目初始—项目确立

1、项目立项

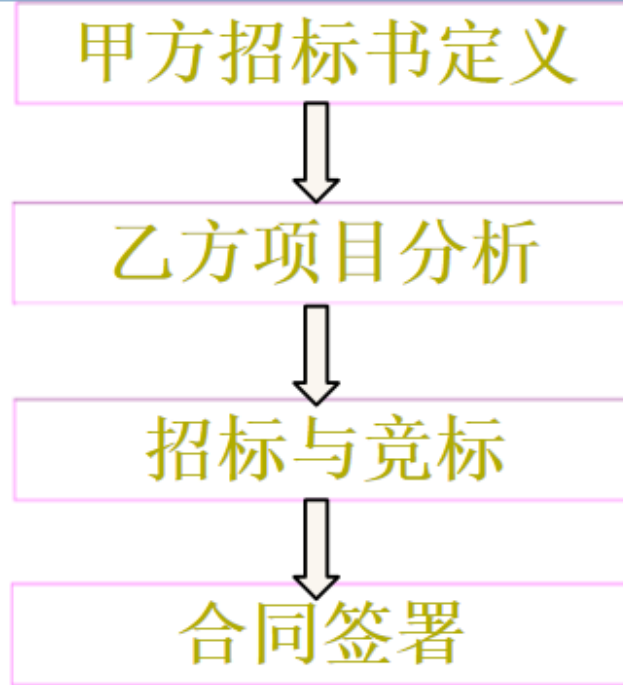
明确项目的**目标、时间表**、项目使用的**资源和经费**，而且得到执行该项目的项目经理和项目发起人的认可。

2、项目招投标

你要造房子叫来一大帮泥水工，你就是甲方（建设单位），泥水工就是乙方（施工单位）。

项目招投标过程

15



chapter_2

甲方招标书：我是领导，找人来干活！

1. 投标邀请
2. 投标人须知
3. 开标与评价
4. 合同专用条款
5. 货物技术规格、参数、要求

乙方项目分析：我是大猛子，我带人来给你们干活！

1. 项目背景
2. 总体需求
3. 实施目标
4. 技术方案
5. 软硬件列表
6. 项目管理



a、可行性分析包括哪些内容？

1、操作可行性

确定：

系统是否能够真正解决问题

是否系统一旦安装后，有足够的人力资源来运行系统

用户对新系统具有抵触情绪可能使操作不可行

2、计划评估

估计项目完成所需的时间

评估项目的时间是否足够

3、技术评估

4、技术方案选择

5、风险分析

6、**社会可行性**

7、**经济分析**

3、项目授权

三、生存期模型选择（重中之重）

1、生存期模型定义

软件生存期模型特征：

1. 描述了开发的主要阶段
2. 定义每一个阶段要完成的主要过程和活动
3. 确定每一个阶段的输入和输出

2、传统生存期模型

a、瀑布模型

适合瀑布模型的项目特征：

- 需求：很明确、固定
- 方案：工作流程 线性
- 类似项目：期项目等

瀑布模型的优缺点

优点：

- 可强迫开发人员采用规范的方法（例如，结构化技术）。
- 要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证。
- 严格地规定了每个阶段必须提交的文档。

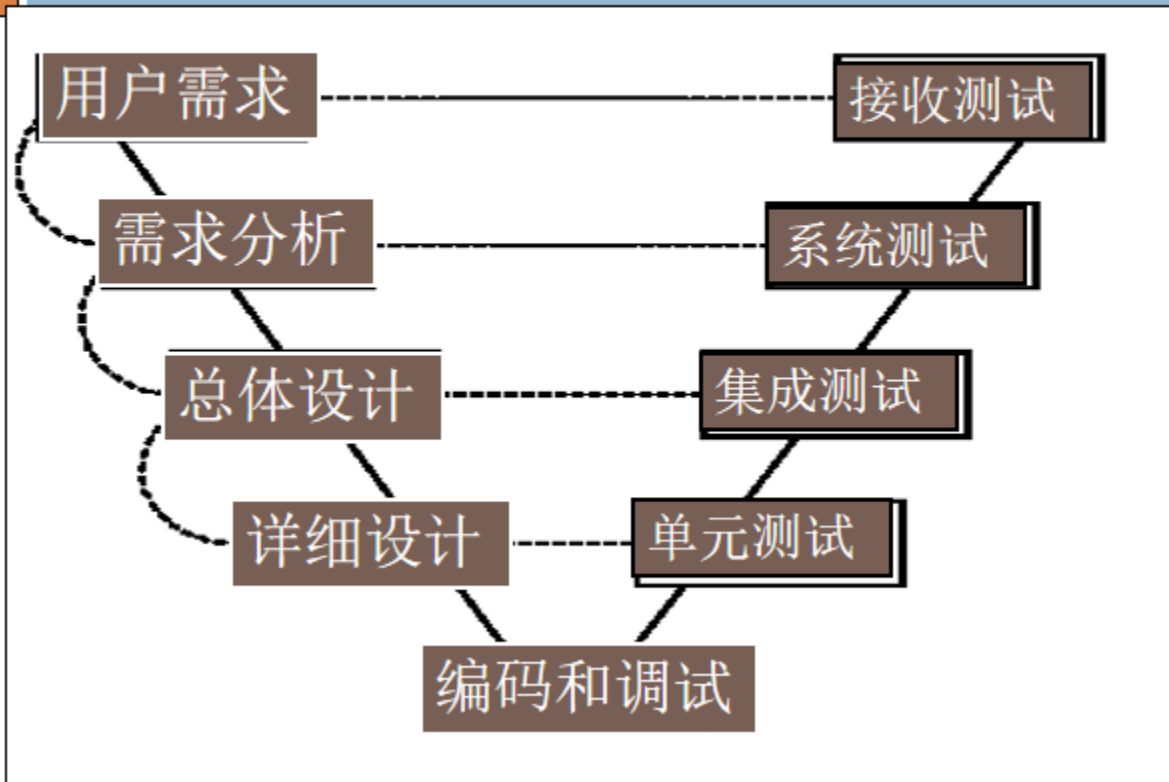
缺点：

- 难以应对需求变化：客户难以准确表达需求，软件团队很难准确理解需求。
- 过于理想：实际项目很少按照该模型给出的顺序进行；
- 风险太大：用户必须有耐心，等到系统开发完成才能见到软件；
- 阻塞状态：开发者常常被不必要地耽搁。

b、V模型

V模型

24



适合V模型的项目特征

- 需求：很明确
- 方案：很明确
- 类似项目：系统性能、安全等有严格要求等

c、原型（重中之重）

- 用户定义了一组一般性目标，但不能标识出详细的输入、处理及输出需求；

- 开发者可能不能确定算法的有效性、操作系统的适应性或人机交互的形式；
- 对需求不是很明朗；
- 用户似乎看到的是软件的工作版本
- 开发者常常需要实现上的折衷，以使原型能够尽快工作。
- 被开发的原型应交付给客户试用，并收集客户的反馈意见，这些反馈意见可在下一轮迭代中对原型进行改进。在前一个原型需要改进，或者需要扩展其范围的时候，进入下一轮原型的迭代开发

原型法

原型是项目系统中的一个方面或者多个方面的工作模型。

- 1、抛弃型原型：用于试验某些概念，试验完系统将无用处
- 2、进化型原型：原型系统不断被开发和被修正，最终它变为一个真正的系统。

原型法的好处：

1. 从实践中学习(Learning by doing)
2. 改善通信
3. 改善用户参与
4. 使部分已知的需求清晰化
5. 验证描述的一致性和完整性
6. 可能可以减少文档
7. 减少了维护成本
8. 特征约束（利用工具构造原型可以将某些特性落到实处，而非在纸上写的那样容易失误）
9. 试验是否能产生期待的结果

原型法的缺点：

1. 用户有时误解了原型的角色，例如他们可能误解原形应该和真实系统一样可靠
2. 缺少项目标准，进化原型法有点像编码修正
3. 缺少控制，由于用户可能不断提出新要求，因而原型迭代的周期很难控制
4. 额外的花费：研究结果表明构造一个原型可能需要10%额外花费
5. 运行效率可能会受影响
6. 原型法要求开发者与用户密切接触，有时这是不可能的。例如外包软件。

适合原型模型的项目特征

需求：不明确

希望：减少项目需求的不确定性

d、螺旋模型

- 风险驱动的软件开发模型
- 采用循环的方式，逐步加深系统定义和实现的深度
- 确定一系列里程碑，确保stakeholders都支持系统解决方案
- 第一圈一般开发出产品的规格说明，接下来开发产品的原型系统，并在每次迭代中逐步完善，开发不同的软件版本
- 结合了原型的迭代性质和瀑布模型的可控性、系统性特点。

螺旋模型优点：

- 结合了**原型的迭代性质**与**瀑布模型的系统性和可控性**，是一种**风险驱动型**的过程模型。
- 采用**循环**的方式逐步加深系统定义和实现的深度，同时更好地理解、应对和降低风险。
- **确定一系列里程碑**，确保各方都得到可行的系统解决方案。
- 始终保持**可操作性**，直到软件生命周期的结束。
- 风险驱动。

螺旋模型存在的问题：

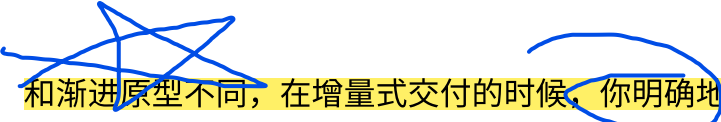
- 螺旋模型依赖大量的**风险评估专家**来保证成功。如果有较大的风险**没有被发现和管理**，肯定会发生问题。
- 软件开发人员应该擅长寻找可能的风险，准确的分析风险，否则将会带来更大的风险。

e、增量模型（重中之重）

- 增量模型以**迭代**的方式运用**瀑布模型**。
- 随着时间的推移，发布一系列称为**增量的版本**，随着每个版本的交付，**逐步为用户提供更多的功能**。
- 先完成一个**系统子集**的开发，再按同样的**开发步骤增加功能 (系统子集)**,如此递增下去直至满足全部系统需求。
- 系统的总体设计在初始子集设计阶段就应作出设想。

增量式交付：

- 增量式交付**持续地在确定的阶段向用户展示软件**。

- 
- 和渐进原型不同，在增量式交付的时候，你明确地知道下一步要完成什么工作
 - 增量式交付的特点是不会在项目结束的时候一下交付全部软件，而是在项目整个开发过程中持续不断地交付阶段性成果。

增量模型优点：

- 提高对用户需求的响应：用户看到可操作的早期版本后会提出一些建议和需求，可以在后续增量中调整。
- 人员分配灵活：如果找不到足够的开发人员，可采用增量模型，早期的增量由少量人员实现，如果客户反响较好，则在下一个增量中投入更多的人力。
- 可规避技术风险：不确定的功能放在后面开发。

增量模型的缺点：

- 每个附加的增量并入现有的软件时，必须不破坏原来已构造好的东西。
- 加入新增量时应简单、方便——软件的体系结构应当是开放的。
- 仍然无法处理需求发生变更的情况。
- 管理人员须有足够的技术能力来协调好各增量之间的关系。
- 难以确定所有版本共需的公用模块。

适合增量模型的项目特征：

- 需求：基本明确，可能发生变化
- 市场、用户：对于市场和用户把握需要逐步了解
- 系统改造：需要一步一步实施

f、渐进式阶段模型（渐进式迭代模型）

特点：

- 渐进式前进
- 阶段式提交

渐进式阶段模型的优点：

3、敏捷生存期模型

敏捷的适用范围

客观因素\最适用方式	敏捷 (Agile)	计划驱动 (Plan-driven)	形式化的开发方法 (Formal Method)
产品可靠性要求	不高, 容忍经常出错	必须有较高可靠性	有极高的可靠性和质量要求
需求变化	经常变化	不经常变化	固定的需求, 需求可以建模
团队人员数量	不多	较多	不多
人员经验	有资深程序员带队	以中层技术人员为主	资深专家
公司文化	鼓励变化, 行业充满变数	崇尚秩序, 按时交付	精益求精
实际的例子	写一个微博网站; 面向消费者的APP	开发下一版本的办公软件; 给商业用户开发软件	开发底层正则表达式解析模块; 科学计算; 复杂系统的核心组件
用错方式的后果	用敏捷的方法开发登月火箭控制程序, 前N批宇航员都挂了。	部分方法还是有效的; 但是, 用全套敏捷方法, 商业用户未必受得了两周一次更新的频率。	敏捷方法的大部分招数都和这类用户无关, 用户关心的是: 把可靠性提高到 99.99%, 不要让微小的错误把系统搞崩溃!

- 敏捷组织提出的一个灵活开发方法
- 应对迅速变化需求的快速软件开发方法
- 是一种迭代、循序渐进的开发方法

敏捷宣言：

- 个人和他们之间的**交流**胜过开发过程和工具
- **可运行的软件**胜过宽泛的文档
- 客户**合作**胜过合同谈判
- 对**变更的良好响应**胜过按部就班地遵循计划

1、Scrum模型

采用短周期迭代交付方式

第一步：收集Product Backlog

找出完成产品需要做的事情 — Product Backlog。

产品负责人领导大家 对于这个 Backlog中的条目进行分析，细化，理清相互关系，估计工作量等工作。每一项工作的时间估计单位为“天”。

第二步：选取Product backlog中的任务加入到sprint中

决定当前的冲刺（Sprint）需要解决的事情 — Sprint Backlog。整个产品的实现被划分为几个互相联系的冲刺（Sprint）。产品订单上的任务被进一步细化了，被分解为以小时为单位。如果一个任务的估计时间太长（如超过 16 个小时），那么它就应该被进一步分解。

第三步：开始执行Sprint

一个Sprint即是一次迭代，往往设置为一周到两周，一个标准的sprint包含以下几个环节
站立会议要求每日都要举行
及时更新燃尽图。

第四步：交付、验收Sprint

得到软件的一个增量版本，产品经理验收。

然后在此基础上又进一步计划增量的新功能和改进。

第五步：总结Sprint

总结本次sprint什么做得好、什么做的不好、后续应该怎么做。

2、XP(eXtreme Programming)极限编程模型

- 极限编程是敏捷软件开发中应用最为广泛和最富有成效的几种方法学之一。
- 极限编程的主要目标在于降低因需求变更而带来的成本。

- 采用迭代的交付方式，每个迭代很短（1-3周时间）。在每个迭代结束的时候，团队交付可运行的，经过测试的功能，这些功能可以马上投入使用。

四、软件需求

1、软件需求定义

需求是指用户对软件的功能和性能的要求。

2、软件需求管理过程

1、需求获取

2、需求分析：需求分析是为最终用户所看到的系统建立一个概念模型，是对需求的抽象描述。

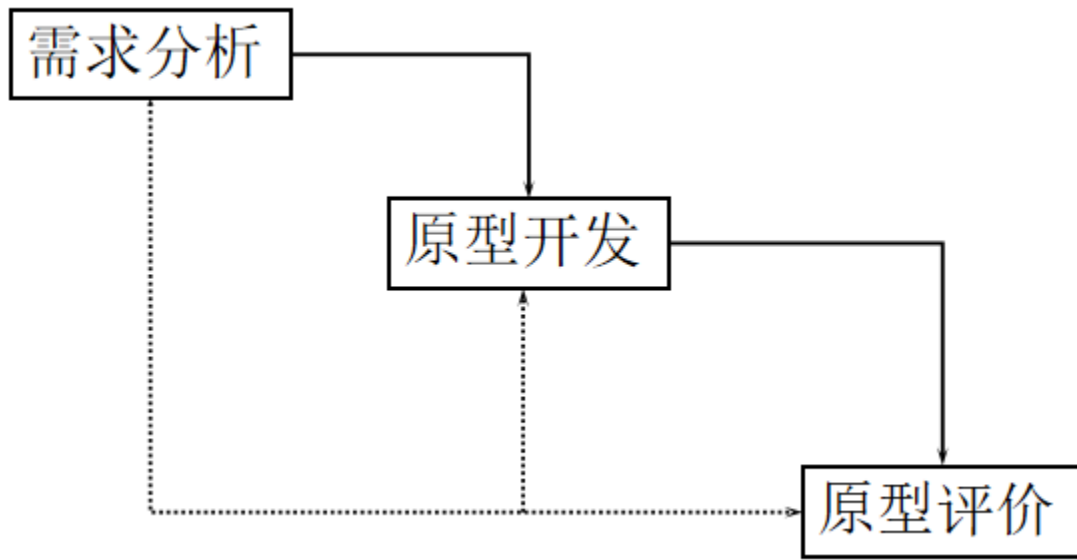
3、需求规格编写：需求分析工作完成的一个基本标志是形成了一份完整的、规范的需求规格说明书

4、需求验证

5、需求总在变化

3、需求建模的基本方法

1、原型方法



2、结构化分析方法

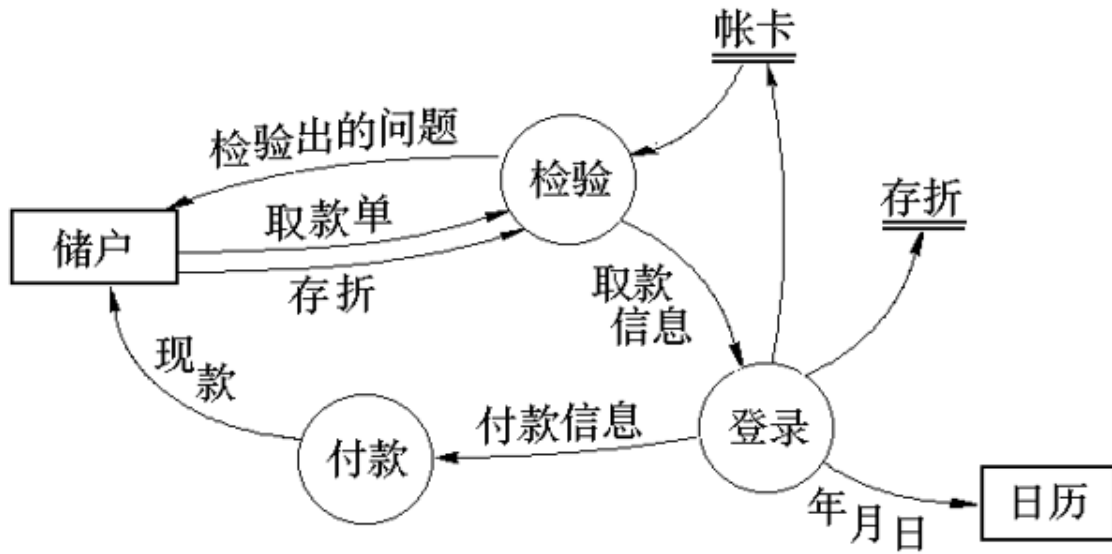
- 20世纪70年代发展起来的面向数据流的方法
- 是一种自顶向下逐步求精的分析方法
- 根据软件内部数据传递、变换的关系进行分析的

结构化分析方法-技术：

a、数据流图(DFD)

描述银行取款过程的数据流图

38



b、数据字典(DD)

学生管理系统-数据字典-数据流

3

学生基本信息：学号+姓名

学生健康信息：学号+健康情况

学生成绩：学号+{课程名+成绩}

查询要求：[健康查询单 | 平均成绩查询单 | 不及格人数查询]

学生健康情况表：优%+良%+一般%+差%

学生成绩单：学号+姓名+{课程名+成绩}+总成绩

不及格人数统计表：学号+成绩+不及格总人数

c、系统流程图

3、面向对象的用例分析

UML需求视图：

1. 用例视图 (Use case Diagram)
2. 顺序图 (Sequence Diagram)
3. 状态图 (State Diagram)
4. 活动图 (Activity Diagram)

4、功能列表

基于功能列表的实例

68

	角色	名称	类别	子类别	子角色	角色	描述	实现阶段
1	成员共用的功能	按产品分类浏览产品目录	产品信息	浏览	成员	组织	用户可以按照产品类别（大类、小类）逐级浏览产品目录	I
2		按医院科别浏览产品目录	产品信息	浏览	成员	组织	用户可以按照医院科别（科、次科）逐级浏览产品目录	I
3		按关键字搜索产品	产品信息	查找	成员	组织	在名称和描述字段中搜索含有关键字的记录。	I
4		E-mail	E-mail		成员	组织，个人	注册成Medeal.com成员后享有Medeal.com提供的免费email帐号和相应的email服务。	I
5		E-mail支持POP3/SMTP协议标准	E-mail	使用	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务需支持POP/SMTP协议标准，以满足不同客户的需求。	I
6		E-mail提供WEB形式收发邮件	E-mail	使用	成员	组织，协会/学会，个人	Medeal.com的Free E-mail是嵌入本网站以WEB形式提供各种服务。	I
7		E-mail收件回复与备份	E-mail	使用	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务需有收件回复与备份功能。	I
8		E-mail寄件备份/定时	E-mail	使用	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务对寄出的信件提供备份功能。	I
9		E-mail定时收发	E-mail	使用	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务提供定时收取邮件和定时发送功能，定时的间隔可由用户设定。	I
10		E-mail邮箱管理	E-mail	管理	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务提供邮箱管理功能。包括创建、修改、删除文件夹和删除邮件。	I
11		防邮件炸弹功能	E-mail	管理	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务应具有抗邮件炸弹破坏功能。	I
12		可收取其它POP3服务器信件	E-mail	管理	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务应提供收取其它POP3服务器信件的功能。	I
13		自动回复	E-mail	管理	成员	组织，协会/学会，个人	Medeal.com的Free E-mail服务应提供来信自动回复功能。	I
14								

五、项目范围控制、任务分解

1、项目范围控制、任务分解定义

项目范围控制：确保项目做且仅做成功完成项目所需的全部工作的各过程

任务分解过程：将一个项目分解为更多的工作细目或者子项目，使项目变得更小、更易管理、更易操作

任务分解结构：WBS（Work Breakdown Structure）

在WBS中，下层的工作都是对其上层工作的**细化**，位于最底层的是工作包。

WBS的意义：范围基准的形成

- WBS是对项目**由粗到细**的分解过程。
- 面向交付成果的
- WBS它组织并定义了整个项目范围

工作包（Work packages）：WBS的最低层次的可交付成果，工作包应当由唯一主体负责

2、任务分解方法

- a、**类比**
- b、**模版参照**
- c、**自上而下**
- d、**自下而上**

3、任务分解的基本步骤

1. 确认并分解项目的组成要素(WBS编号)

2. 确定分解**标准**
3. 确定分解是否详细
4. 确定项目交付成果（可以编制WBS字典）
5. 验证分解的正确性

不能同时使用两种标准进行分解，分解标准应统一：学生管理为例

按照生存期阶段分解

1. 规划
2. 需求
3. 设计
4. 编码
5. 测试
6. 提交

按照产品组成分解

- 1.1 招生管理
- 1.2 分班管理
- 1.3 学生档案管理
- 1.4 学生成绩管理

检验分解结果的标准：

1. 最底层的要素是否是实现目标的**充分必要条件**
2. 最底层要素是否有重复的
3. 每个要素是否清晰完整定义
4. 最底层要素是否有定义清晰的责任人
5. 是否可以进行成本估算和进度安排

六、软件工作量估计

1、估算过程概念

- 估算不是很准确，有误差
- 项目经验数据非常重要
- 不要太迷信某些数学模型

软件项目规模：软件项目规模即工作量

例如：软件规划，软件管理，需求，设计，编码，测试，以及后期的维护等任务。

软件规模：

- LOC(Loc of Code)：**源代码长度**的测量
- FP(Function Point)：用**系统的功能数量**来测量

工作量：人月、人天、人年

软件项目成本：

- 完成软件规模相应付出的代价。
- 待开发的软件项目需要的资金。
- 人的劳动的消耗所需要的代价是软件产品的主要成本

软件规模和软件成本的关系：

- 规模是成本的主要因素，是成本估算的基础
- 有了规模就确定了成本

成本估算结果：

- 直接成本：与具体项目相关的成本；例如：参与项目的人员成本
- 间接成本：以分摊到各个具体项目中的成本；例如：房租水电、员工福利

2、估算方法（重点：5个工作量估计的方法）

a、自顶向下方法

自顶向下法是对整个工程项目的总开发时间和总工作量做出估算，然后将它们按阶段、步骤和任务进行分配

自顶向下方法和参数模型：

1. 自顶向下的方法和参数模型相关

2. 一般采用**对比**方法确定总的工作量
3. 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
4. 参数模型的形式：

effort=（系统规模）*（生产率）

- a. 例如系统规模可以用KLOC来计算，生产率以40天/KLOC
- b. 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

b、自底向上方法

利用任务分解图(WBS),对各个具体工作包进行详细的成本估算,然后将结果累加起来得出项目总成本。

- 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迭代的过程，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

自下而上估算特点：

1. 相对比较准确，它的准确度来源于每个任务的估算情况
2. 花费时间

c、专家估算法

由多位专家进行成本估算，一个专家可能会有偏见，最好由多位专家进行估算，取得多个估算值,最后得出综合的估算值。

- 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有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。

d、类比估算法（基于案例的推理 Case-based reasoning）

- 估算人员根据**以往的完成类似项目所消耗的总成本（或工作量）**，来推算将要开发的软件的**总成本（或工作量）**，然后按比例将它分配到各个开发任务单元中
- 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。
- 是一种**自上而下**的估算形式
- 自顶向下法是对整个工程项目的总开发时间和总工作量做出估算，然后将它们按阶段、步骤和任务进行分配
- 如何选择与待预测的项目相近的项目？

欧几里德距离（Euclidean Distance）公式：

distance=（（目标参数1-原参数1）²+（目标参数2-原参数2）²+.....）的平方根

练习

- 假定要匹配的案例基于两个参数, 构造系统的输入和输出参数。新的系统有7个输入, 15个输出, 过去有一个项目A有8个输入, 17个输出, 这两个项目的欧几里的距离是多少?
- 答案: $\sqrt{(7-8)^2 + (15-17)^2} = 2.24$

e、基于分解技术的估算 (三点估算)

- Problem-based Estimation 基于问题的估算(续)
- A three-point or expected value for LOC/FP estimation LOC/FP 估算的三点或期望值计算
 - Estimate an optimistic, most likely, and pessimistic size value for each function or count for each information domain value.
 - 为各个功能或信息域的计数值 分别估算出一个乐观值, 可能值和悲观值
 - The expected value for estimation variable s can be computed as follows:

$$S = \frac{S_{op} + 4S_m + S_{pess}}{6}$$

f、代码行估算法

从软件程序量的角度定义项目规模。

1. 与具体的**编程语言**有关
2. 分解足够详细
3. 有一定的经验数据（类比和经验方法）

代码行技术的主要优点：

代码是所有软件开发项目都有的“产品”，而且很容易计算代码行数。

代码行估算的缺点：

- 对代码行没有公认的可接受的**标准**定义
- 代码行数量依赖于所用的编程语言和个人的编程风格.
- 在项目早期，需求不稳定、设计不成熟、实现不确定的情况下很难准确地估算代码量.
- 代码行强调编码的工作量,只是项目实现阶段的一部分

g、功能点估算

- 与实现的语言和技术没有关系
- 用系统的**功能数量**来测量其规模
- 通过**评估、加权、量化**得出功能点

功能点公式：FP =UFC*TCF

- UFC：未调整功能点计数

- TCF：技术复杂度因子

UFC-未调整功能点计数

功能计数项：(从处理逻辑的角度)

1. **外部输入**：给软件提供面向应用的数据的项（如屏幕、表单、对话框、控件，文件等）；在这个过程中，数据穿越外部边界进入到系统内部。
2. **外部输出**：向用户提供(经过处理的)面向应用的信息，例如，报表和出错信息等。
3. **外部查询**：外部查询是一个输入引出一个即时的简单输出。没有处理过程。
4. **外部接口文件**：外部接口文件是用户可以识别的一组逻辑相关数据，这组数据只能被引用。用这些接口把信息传送给另一个系统。
5. **内部逻辑文件**：用户可以识别的一组逻辑相关的数据，而且完全存在于应用的边界之内，并且通过外部输入维护，是逻辑主文件的数目。

例题：

1. 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。
2. 假定报表是具有高度复杂性的，其它具有一般复杂性

软件项目复杂度权重因素

	复杂度权重因素		
项	简单(低)	一般(中)	复杂(高)
外部输入	3	4	6
外部输出	4	5	7
外部查询	3	4	6
外部接口文件	5	7	10
内部逻辑文件	7	10	15

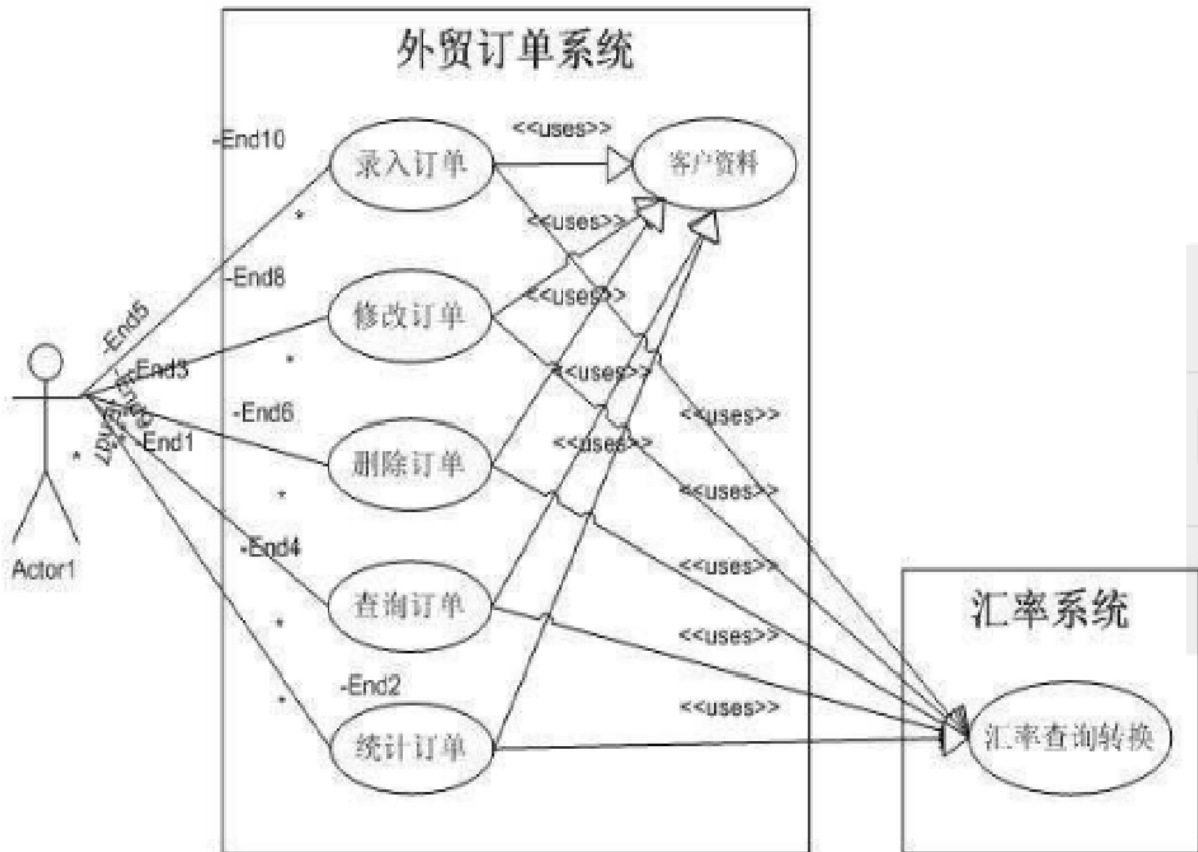
- 外部输入：无
- 外部输出：报告，1
- 内部逻辑文件：财务输入文件，1
- 外部接口文件：工资文件，人员文件，课程文件，财务输入文件，4
- 外部查询：无

考虑加权：

- 外部输入：无；
- 外部输出： $1 \times 7 = 7$ ；
- 内部逻辑文件： $1 \times 10 = 10$ ；
- 外部接口文件： $4 \times 7 = 28$ ；
- 外部查询：无；

- **共：45**

例题：FP估算方法举例



根据上面的外贸订单项目的需求评估:

- 外部输入:3项;
- 外部输出:1项;
- 外部查询:1项;
- 外部接口文件:1项
- 内部逻辑文件:2项

	功能点		
项	简单	一般	复杂
外部输入	2 * 3	1 * 4	0 * 6
外部输出	0 * 4	0 * 5	1 * 7
外部查询	0 * 3	1 * 4	0 * 6
外部接口文件	0 * 5	1 * 7	0 * 10
内部逻辑文件	1 * 7	1 * 10	0 * 15
总计			57
UFC	45		

TCF-技术复杂度因子

技术复杂度因子的取值范围

调整系数	描述
0	不存在或者没有影响
1	不显著的影响
2	相当的影响
3	平均的影响
4	显著的影响
5	强大的影响

chapter__3

61

$$TCF=0.65+0.01(\text{sum}(F_i))$$

- F_i : 0-5
- TCF : 0.65-1.35

技术复杂度因子			
F1	可靠的备份和恢复	F2	数据通信
F3	分布式函数	F4	性能
F5	大量使用的配置	F6	联机数据输入
F7	操作简单性	F8	在线升级
F9	复杂界面	F10	复杂数据处理
F11	重复使用性	F12	安装简易性
F13	多重站点	F14	易于修改

58

外贸订单项目：功能点计算实例

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。

FP原来=UFC*TCF

- $UFC=45$
- $TCF=0.65+0.01(143)=1.07$

FP=45*1.07=48

如果：PE=15工时/功能点

则：Effort=48*15=720工时

h、用例点估算法

- 计算未调整的**角色**的权值UAW;
- 计算未调整的**用例**的权值UUCW ;
- 计算未调整的**用例点**UUCP;
- 计算**技术和环境因子**TEF;
- 计算调整的用例点UCP ;
- 计算工作量(man-hours) 。

i、参数估算法

- 通过项目数据,进行回归分析,得出回归模型
- 通过参数模型估算(规模)成本的方法。

参数估算法使用条件：

- 具有良好的项目数据为基础
- 存在成熟的项目估算模型

参数估算法特点：

- 比较简单,而且也比较准确
- 如果模型选择不当或者数据不准,也会导致偏差

3、成本预算

成本预算：将项目的总成本按照项目的进度分摊到各个工作单元中去

成本预算的目的：产生成本基线

分配项目成本预算包括以下三种情况：

a、给任务分配资源成本

1. 与资源的基本费率紧密相连
2. 设置资源费率

标准费率

加班费率

每次使用费率

b、分配固定资源成本

当一个项目的资源需要固定数量的资金时，可以向任务分配固定资源成本。

例如：项目中的一个兼职人员成本

c、分配固定成本

有些任务是固定成本的类型的任务，也就是说，管理者知道某项任务的成本不变，不管任务的工期有多长，或不管任务使用了那些资源。在这种情况下，管理者向任务直接分配成本。

例如：某外包任务、培训任务

七、进度计划

1、进度管理基本概念

进度的定义：进度是对执行的活动和里程碑制定的工作计划日期表

任务定义（Defining Activities）：确定为完成项目的各个交付成果所必须进行的诸项具体活动

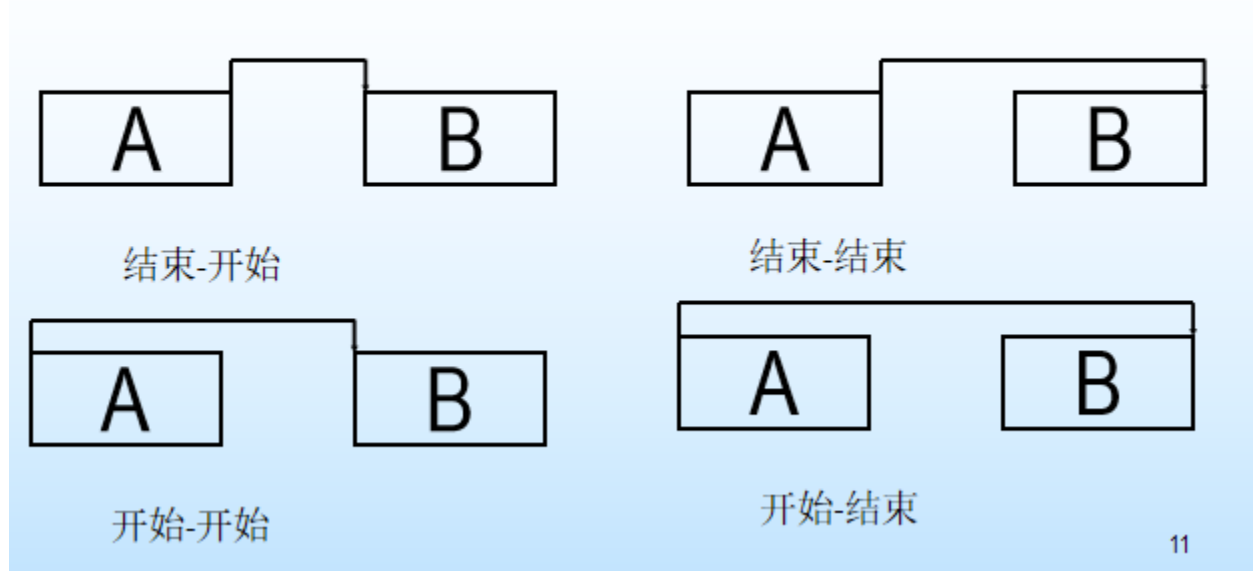
项目任务的关联关系：

项目各项活动之间存在相互联系与相互依赖关系

根据这些关系安排任务之间的顺序

前置活动（任务）---> 后置活动（任务）

任务(活动)之间的关系



任务之间关联关系的依据：

- 强制性依赖关系
- 软逻辑关系
- 外部依赖关系

进度管理图示：

a、网络图：

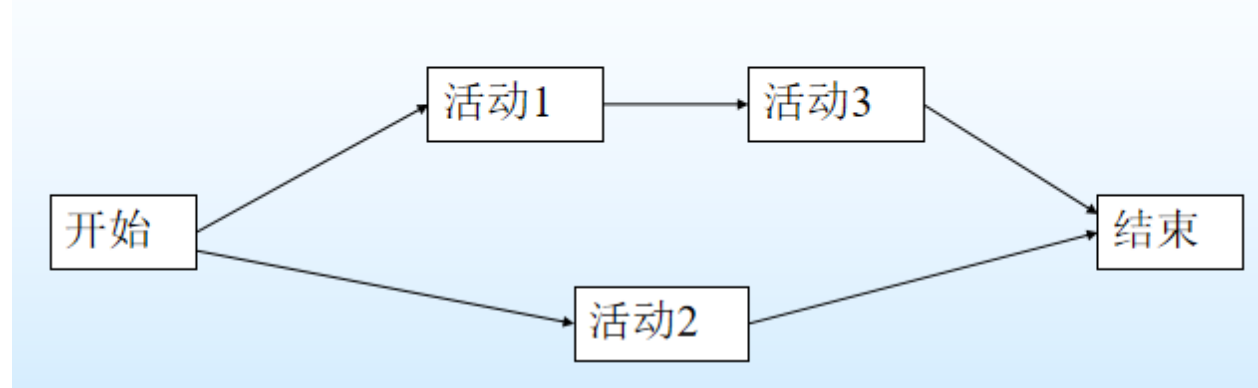
- 网络图是活动排序的一个输出
- 展示项目中各个活动以及活动之间的逻辑关系
- 网络图不仅用在在在项目中，而且可以进行项目跟踪

常用的网络图

一、PDM (Precedence Diagramming Method)

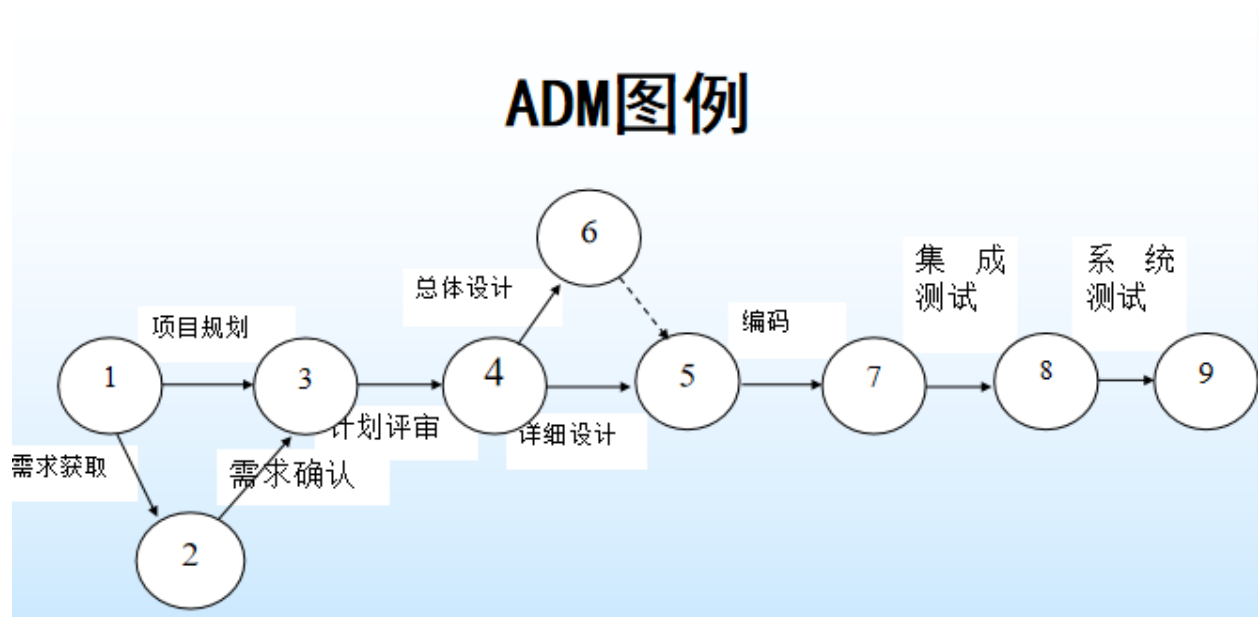
优先图法,节点法(单代号)网络图 (AON)

PDM图例



构成PDM网络图的基本特点是节点(Box)
节点(Box)表示活动（任务）
用箭线表示各活动(任务)之间的逻辑关系.
可以方便的表示活动之间的各种逻辑关系

二、ADM (Arrow Diagramming Method) 箭线法 (双代号)网络图 (AOA)

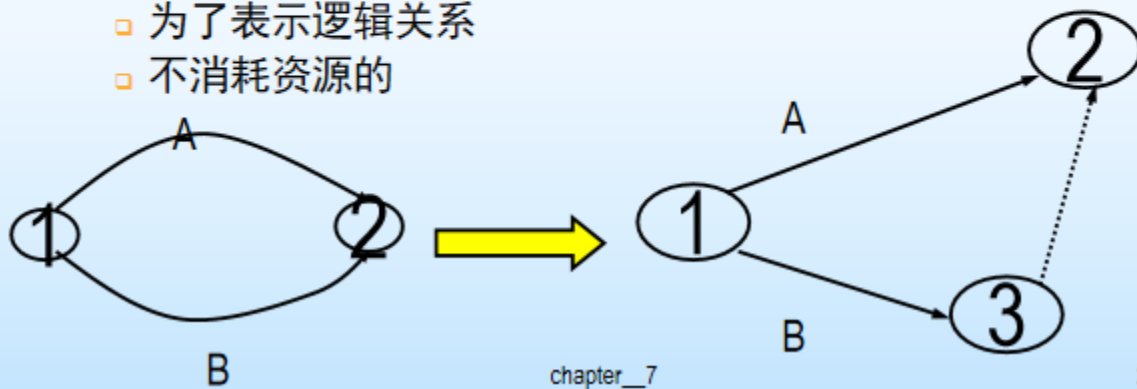


ADM也称为**双代号项目网络图**,
在ADM网络图中,箭线表示活动(任务)
两个代号唯一确定一个任务
代号表示前一任务的结束,同时也表示后一任务的开始.

ADM图例-虚活动

虚活动

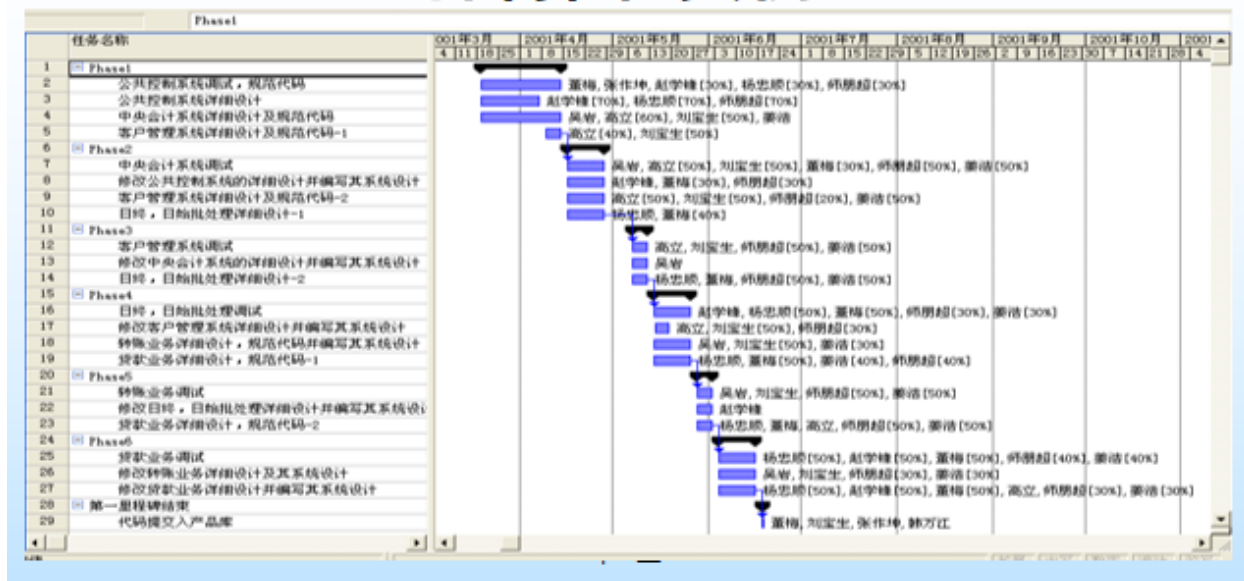
- 为了定义活动
- 为了表示逻辑关系
- 不消耗资源的



21

b、甘特图

甘特图-实例



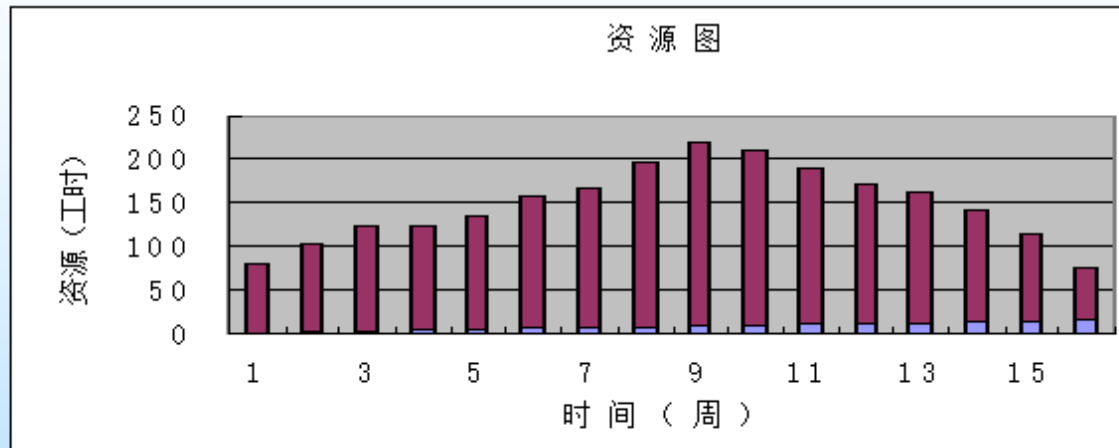
c、里程碑图

里程碑图示



d、资源图

资源图



2、任务历时估算

a、定额估算法

$$T=Q/(R*S)$$

T:活动历时

Q:任务工作量

R:人力数量

S:工作效率 (贡献率)

定额估算法

□ 例如

□ $Q=6$ 人天, $R=2$ 人, $S=1$

□ 则: $T=3$ 天

□ 例如

□ $Q=6$ 人天, $R=2$ 人, $S=1.5$

□ 则: $T=2$ 天

b、经验导出模型

经验导出模型

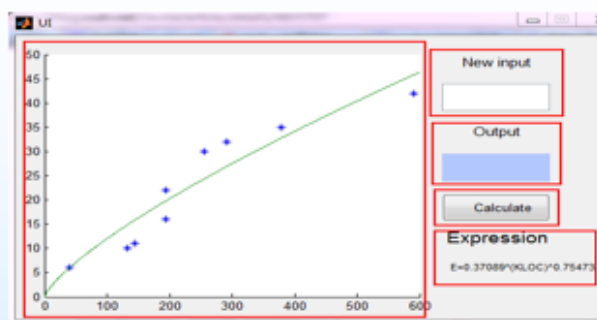
经验导出模型: $D=a \cdot E^b$:

□ D : 进度 (以月单位)

□ E : 工作量 (以人月单位)

□ a : 2—4之间

□ b : 1/3左右: 依赖于项目的自然属性



建议掌握模型

- Walston-Felix模型: $D=2.4 \times E^{0.35}$
- 基本COCOMO: $D=2.5 \times E^b$

方式	b
有机	0.38
半有机	0.35
嵌入式	0.32

基本COCOMO举例

一个33.3 KLOC的软件开发项目，属于中等规模、半有机型的项目，采用基本COCOMO估算进度？

1) 采用基本COCOMO模型估算的规模 $E = 152 \text{ PM}$

2) 采用基本COCOMO模型估算的进度

$$\begin{aligned} D &= 2.5 \times E^{0.35} \\ &= 2.5 \times 152^{0.35} = 14.5 \end{aligned}$$

经验导出模型举例

假设：导出模型 $D=3 \times E^{1/3}$

则：E=65人月 $\implies D=3 \times 65^{1/3} = 12$ 月

c、CPM(关键路径法估计)

1. 确定项目网络图
2. 每个任务有单一的历时估算
3. 确定网络图中任务的逻辑关系
4. 关键路径是网络图中最长的路径。
5. 关键路径可以确定项目完成时间

d、PERT(工程评估评审技术)

(Program Evaluation and Review Technique)利用网络顺序图逻辑关系

项目中某项单独的活动，存在很大的不确定性。

加权算法估算任务历时

利用网络图逻辑关系，确定路径、项目历时

工程评估评审技术 (PERT)-加权算法

- 它是基于对某项任务的乐观，悲观以及最可能的概率时间估计
- 采用加权平均得到期望值 $E = (O + 4m + P) / 6$,
 - O是最小估算值:乐观(Optimistic),
 - P是最大估算值:悲观(Pessimistic),
 - M是最大可能估算(Most Likely)。

(PERT)-加权算法例子

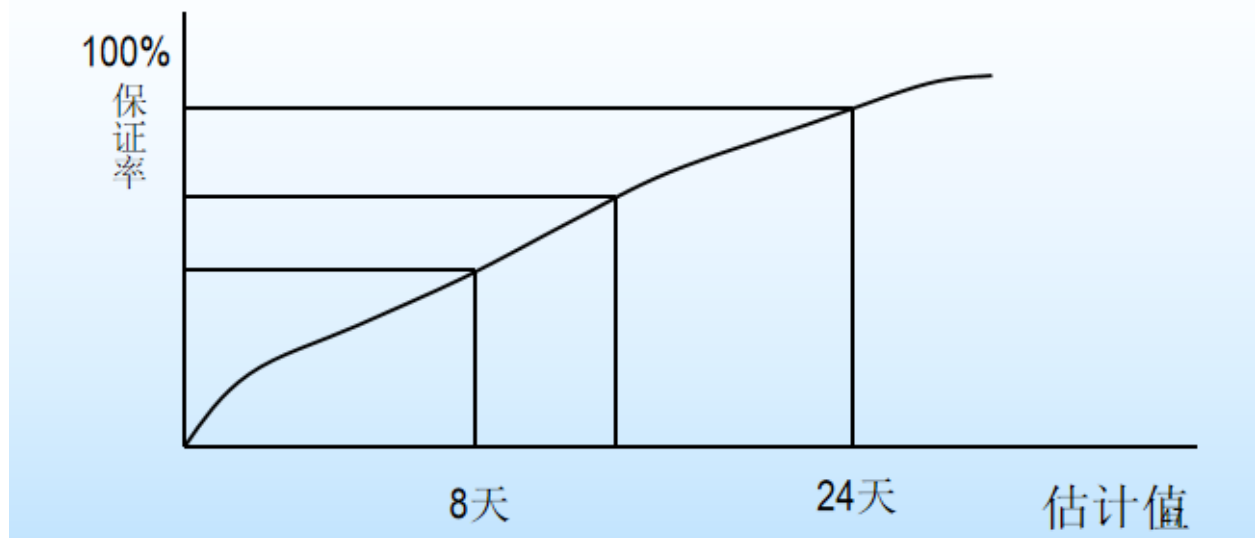
Example:

PERT weighted average =

$$\frac{8 \text{ days} + 4 \times 10 \text{ days} + 24 \text{ days}}{6} = 12 \text{ days}$$

where 8 = optimistic time, 10 = most likely time, and 24 = pessimistic time

PERT的风险性



PERT的风险指标

- 标准差 $\delta = (\text{最大估算值} - \text{最小估算值}) / 6$
- 方差 $\delta^2 = [(\text{最大估算值} - \text{最小估算值}) / 6]^2$

例如上图: $\delta = (24 - 8) / 6 = 2.67$

e、基于承诺的进度估计

- 要求开发人员做出进度承诺

- 不进行中间的工作量（规模）估计

优点

- 有利于开发者对进度的关注
- 有利于开发者在接受承诺之后的士气高昂

缺点

- 易于产生大的估算误差

f、Jones的一阶估算准则

- 幂次表
- 估算项目功能点
- 从幂次表中选择合适的幂次将功能点升幂

3、进度计划编排

a、关键路径法——重点！！！！

见《项目管理》PPT专题篇

- **最早开始时间(Early start)**：指在各紧前工作全部完成后，工作i或i-j有可能完成的最早时刻。
- **最晚开始时间(Late start)**：指在不影响整个项目按期完成的前提下，工作i或i-j必须开始的最迟时刻。

- **最早完成时间(Early finish)：**
- **最晚完成时间(Late finish)：**指在不影响整个项目按期完成的前提下，工作i或i-j必须完成的最迟时刻。
- **浮动时间/(Float)：**浮动时间是一个任务的机动性,它是一个任务在不影响其它任务或者项目完成的情况下可以延迟的时间量。
- **总浮动（总时差）（Total Float）：**指在不影响项目总工期的前提下，工作i或i-j可以利用的机动时间。
- **自由浮动（自由时差）（Free Float）：**指在不影响其紧后工作最早开始时间的前提下，工作i或i-j可以利用的机动时间。
- **超前(Lead)**
- **滞后(Lag)**
- **关键工作：**在网络计划中，工作的总时差反映了该工作对项目工期的敏感程度。总时差越小，表明该工作对项目工期越敏感。因此，总时差最小的工作即关键工作。
- **关键路径（Critical Path）：**首先确定关键工作（总时差最小），自始至终全部由关键工作组成的路径为关键路径，或 路径上总的工作持续时间（所有活动历时累加）最长的路径为关键路径。

1. 时间浮动为0（Float=0）的路径

2. 网络图中最长的路径

3. 关键路径是决定项目完成的最短时间。

4. 关键路径上的任何活动延迟，都会导致整个项目完成时间的延迟

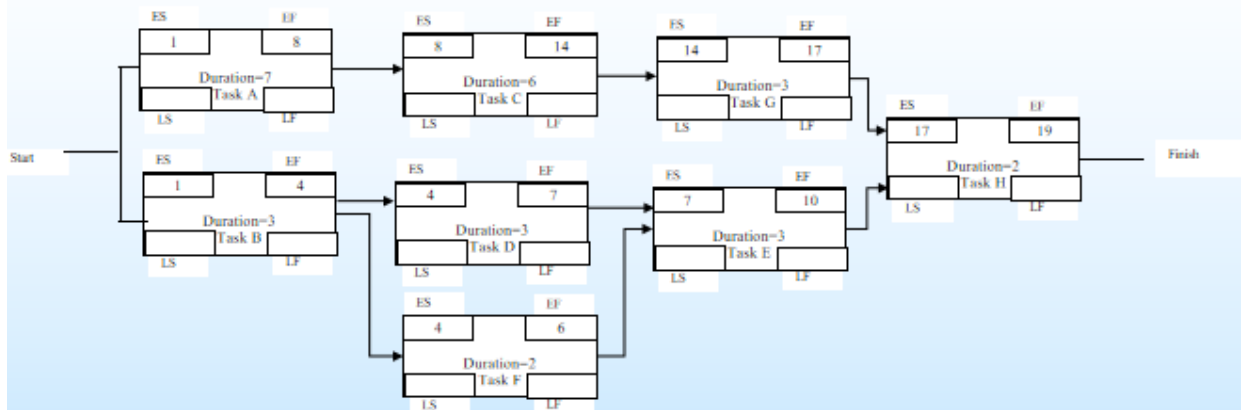
5. 关键路径可能不止一条

正推法(*Forward pass*)

按照时间顺序计算最早开始时间和最早完成时间的方法,称为正推法

- 1) 确定项目的开始时间。
- 2) 从左到右, 从上到下
- 3) 计算每个任务的最早开始时间ES和最早完成时间EF:
 - 网络图中第一个任务的最早开始时间是项目的开始时间;
 - $ES + \text{Duration} = EF$
 - $EF + \text{Lag} = ES$ (s)
 - 当一个任务有多个前置任务时, 选择前置任务中最大的EF加上Lag作为其ES。

正推法实例



当一个任务有多个前置任务时, 选择前置任务中最大的EF加上Lag作为其ES。

79

逆推法(Backward pass)

按照逆时间顺序计算最晚开始时间和最晚结束时间的方法, 称为逆推法

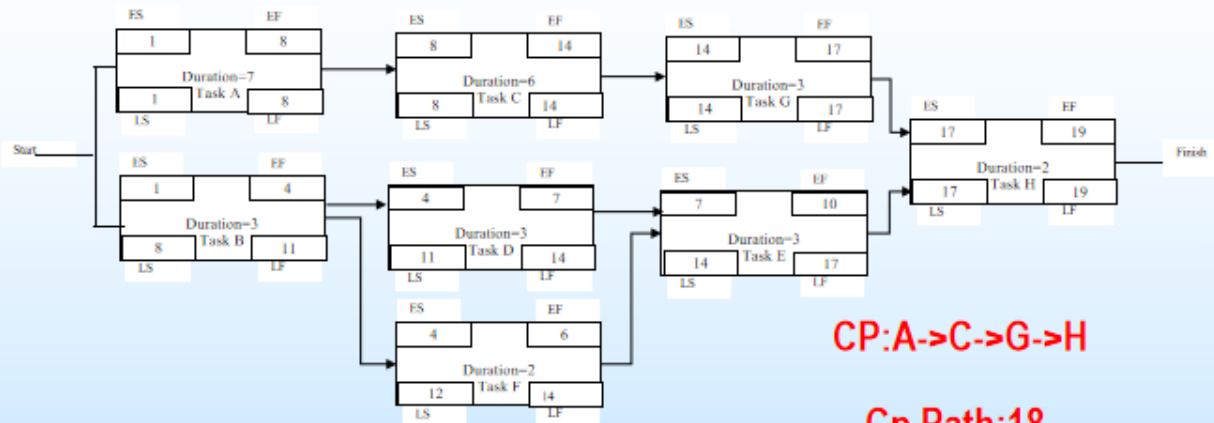
1) 确定项目的结束时间

2) 从右到左, 从上到下

3) 计算每个任务的最晚开始时间LS和最晚完成时间LF

- 网络图中最后一个任务最晚完成时间是项目的结束时间;
- $LF - \text{Duration} = LS$
- $LS - \text{Lag} = LF(p)$
- 当一个任务有多个后置任务时, 选择其后置任务中最小LS减Lag作为其LF

逆推法实例



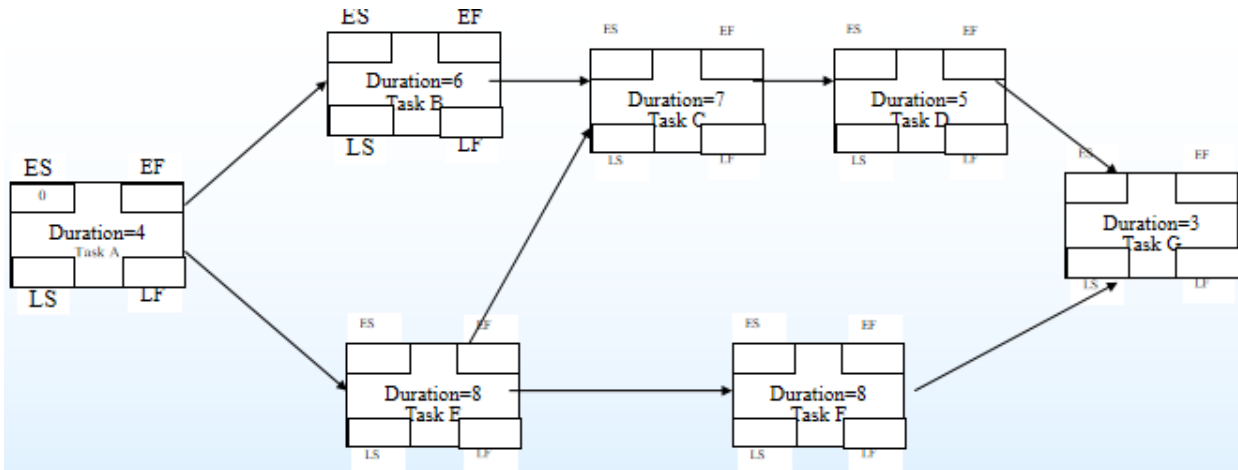
当一个任务有多个后置任务时, 选择其后置任务中最小LS减Lag作为其LF。

81

例题：

作为项目经理, 你需要给一个软件项目做计划安排, 经过任务分解后得到任务A, B, C, D, E, F, G, 假设各个任务之间没有滞后和超前, 下图是这个项目的PDM网络图。通过历时估计已经估算出每个任务的工期, 现已标识在PDM网络图上。假设项目的最早

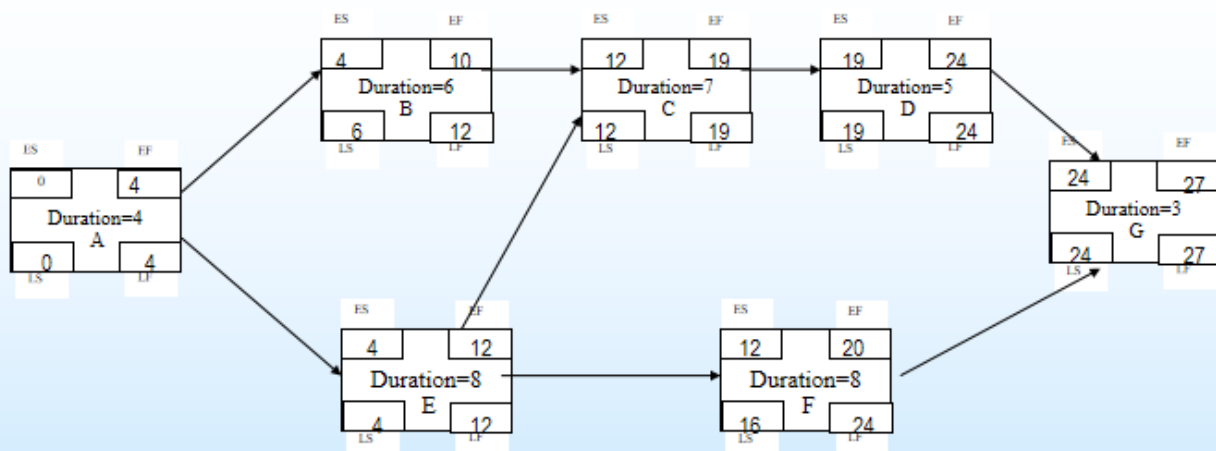
开工日期是第0天，请计算每个任务的最早开始时间，最晚开始时间，最早完成时间，最晚完成时间，同时确定关键路径，并计算关键路径的长度，计算任务F的自由浮动和总浮动



1. 确定所有任务的ES, EF, LS, LF
2. 确定关键路径以及关键路径的长度？
3. 确定F的自由浮动和总浮动？

84

课堂练习-答案



CP:A->E->C->D->G CP Path:27

TF(F)=4
FF(F)=4

85

b、时间压缩法、管理预留、资源平衡、敏捷计划（后面的方法作了解，不作具体讲解）

4、项目进度模型

软件项目进度问题（Software Project Scheduling Problem，SPSP）模型是在给定的项目任务工作量及其关系和资源限制下，对项目确定合适的人员安排，以保证项目的时间最短、成本最小。

计划优化调整

1. 调整资源,解决资源冲突
2. 调整进度,优化项目,缩短工期
3. 调整项目成本预算,以便减少项目费用

七、风险计划

1、风险基本概念

不确定的事件或情况，一旦出现 将会对项目目标产生积极或消极影响

我们常指的风险：一旦出现会对项目目标产生消极影响的不确定事件或情况

a、风险特点（性质）：

1. 风险存在的**客观性和普遍性**
2. 某一具体风险发生的**偶然性**和大量风险发生的**必然性**
3. 风险的**可变性**
4. 风险的**多样性和多层次性**
5. 风险的**不利性**

b、项目风险的三要素：

1. 一个**事件**
2. 事件发生的**概率**
3. 事件的**影响**

c、风险类型：

预测角度：

1. 已知风险—Known known
2. 可预测风险-Known unknown
3. 不可预测风险-unknown unknown

范围角度：

商业风险、管理风险、人员风险、技术风险、开发环境风险、客户风险、过程风险、产品规模风险等。

2、风险管理过程

a、风险识别

风险识别是试图通过系统化地确定对项目计划的威胁，识别已知和可预测的风险。

风险识别方法：

1. 德尔菲方法
2. 头脑风暴法
3. 情景分析法
4. 利用风险条目检查表

b、风险评估

对风险事件发生概率的评估，对项目风险影响的评估，给出项目风险排序。

分析：

- 风险发生的概率（P）
- 风险对项目的危害（I）
- 风险影响(风险暴露度) 风险值， $R=F(P,I)=P*I$

确定优先次序：

- 按风险值排序
- 确定最需要关注的TOP 风险

风险评估-定量确定风险值：

1. 风险的可能性(risk likelihood)：危险发生的概率
2. 对项目的影响程度(risk impact)
3. 风险的重要性：
4. 风险暴露量
 - $\text{Risk exposure} = \text{risk likelihood (概率)} * \text{risk impact (一般以金钱为单位)}$
 - Risk impact一般以金额为单位，而likelihood以概率为单位。

c、风险规划

d、风险控制监督

3、风险管理计划

八、软件项目监督和控制

1、项目监控的内容

- 监控项目的进展
- 比较实际进度与计划的差别
- 修改计划使项目能够返回预定“轨道”

2、项目监控框架framework：过程

关心的四个问题：

1. 工期拖延
2. 质量不过关
3. 功能不合适
4. 成本超出预算

本章主要关心第一、第四个问题

a、项目监控框架：责任

b、项目监控框架：进度评估

- 基础：定期信息收集或者发生的特定事件
- 这些信息必须是客观的和可度量的
- 但是并非每一次都能够得到符合要求的信息，因而通常需要项目成员进行主观判断

c、项目进度监控：检查点设置

检查点(Checkpoints)包括：

- 定期的（如一星期一次，一月一次）
- 与特定的事件绑定的，如生成一份报告或者交付部分产品

d、项目监控框架：监测频率frequency

- 监测的频率依赖于项目的大小和风险情况
团队领导，可能需要每天都了解一下进度
项目经理需要每星期或每月了解情况
- 管理层次越高，频率越低，信息越抽象
- 许多公司利用星期一早晨的短会来激励员工实现短期目标

3、数据收集

尽管整个过程被分成了容易管理的活动，但是项目执行中仍然需要在活动中对任务完成的比例进行评估，这种评估通常是困难的。

a、部分完成报告

1. 许多组织采用财务系统中的每周时刻表来记录每个职员在每项工作中花费的时间，但是该表无法告诉项目经理目前产出了什么，进度是否满足要求。
2. 因而可以对每周时刻表进行扩展，以包含完成的工作内容

b、风险报告

- 询问小组成员完成计划的可能性
- 交通灯方法：
- 识别评价某项工作中的关键元素
- 将这些关键元素分解为组成元素
- 对于每一元素：

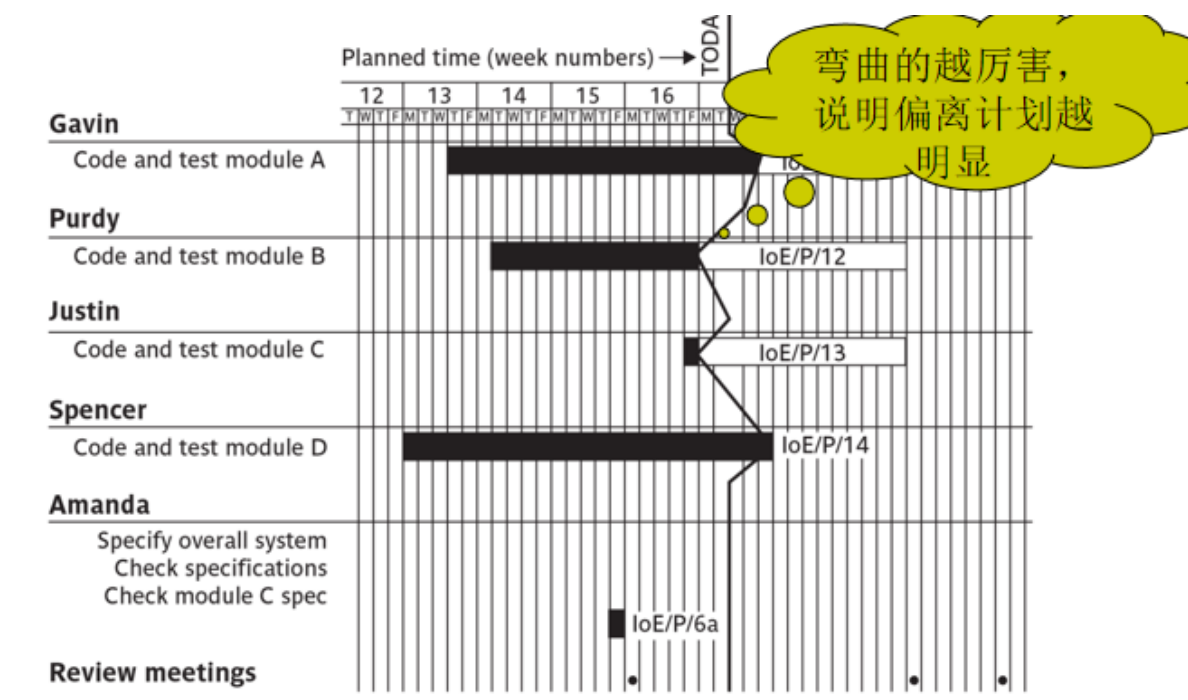
如果符合计划要求：绿灯

目前已经拖后，但是可以恢复，黄灯

已经拖后，恢复很困难，红灯

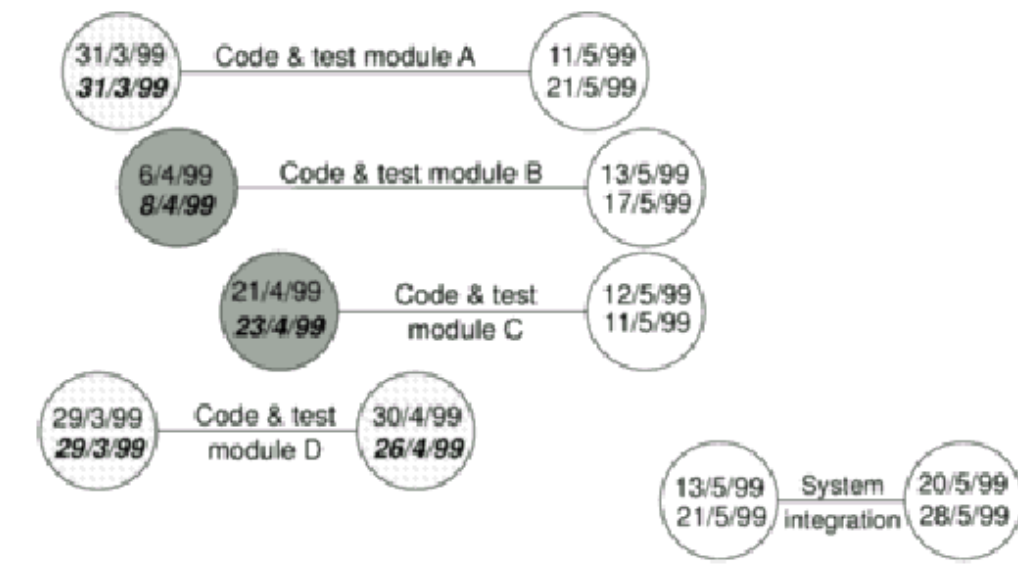
4、进度可视化

a、延迟图(slip chart)



b、球图

1. 计划开始，计划结束作为两个球，每次计划改变后，日期添加到球中，如果时间是按计划的，球被填为绿色，否则被填为红色。
2. 每次更新后，图不需重画

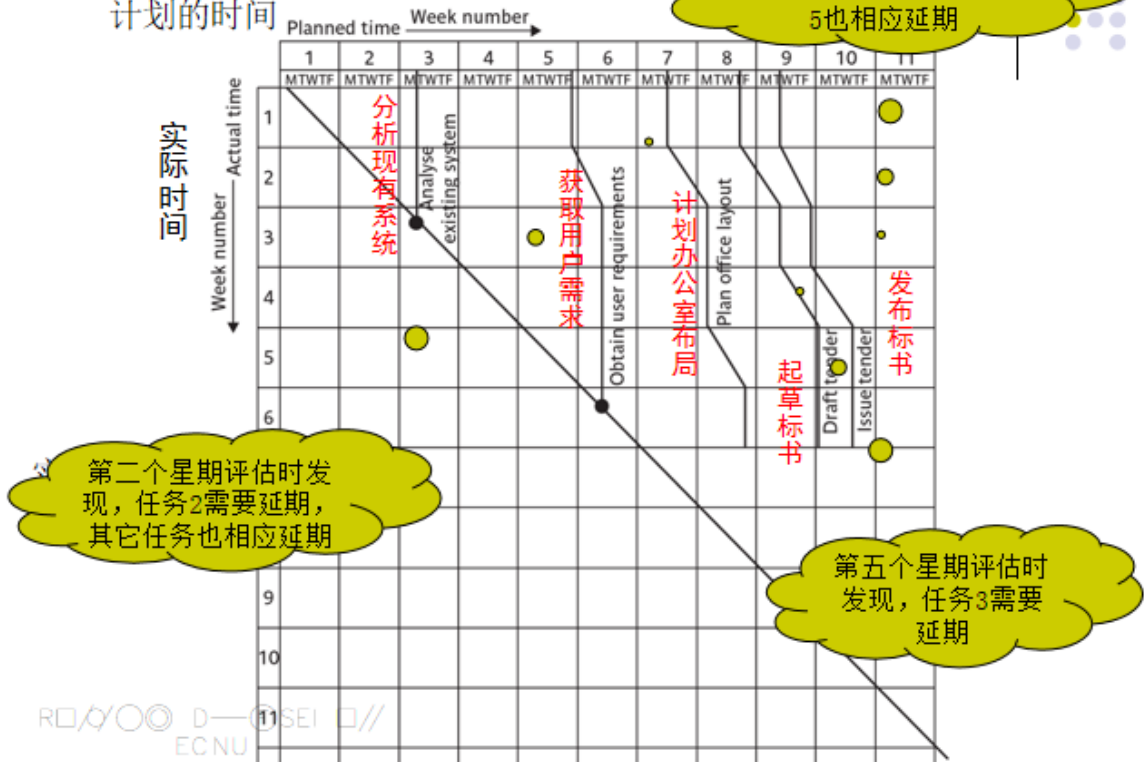


c、时间线图 (timeline)

- 前面的方法不能表示出项目生命周期中偏离计划的情况。
- 对计划偏离的趋势分析能够避免将来的项目偏离。

进度可视化

计划的时间



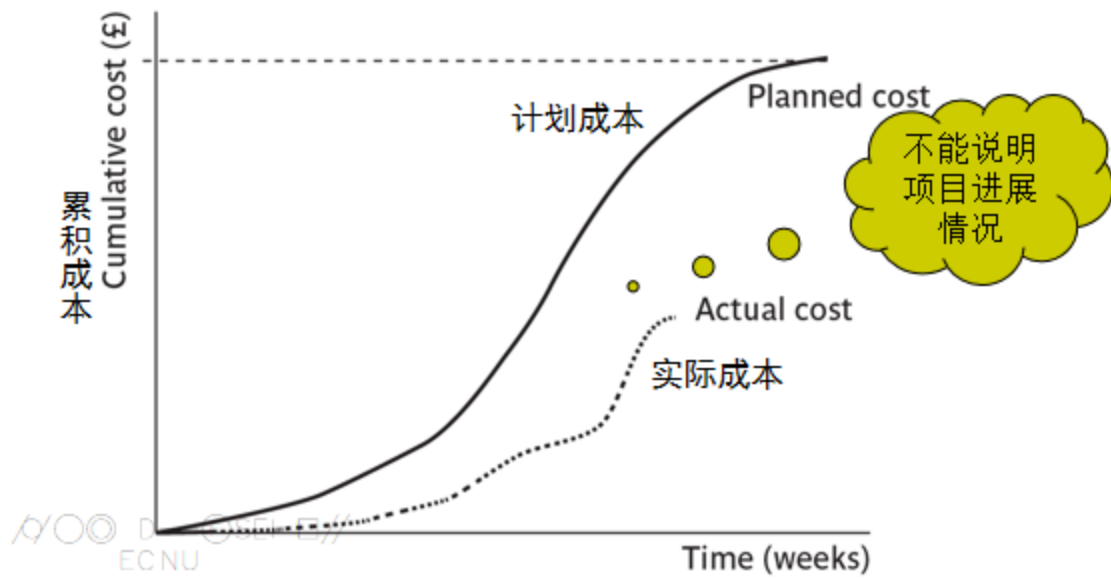
5、成本监控

。 监控的意义

成本本身是项目中的重要元素

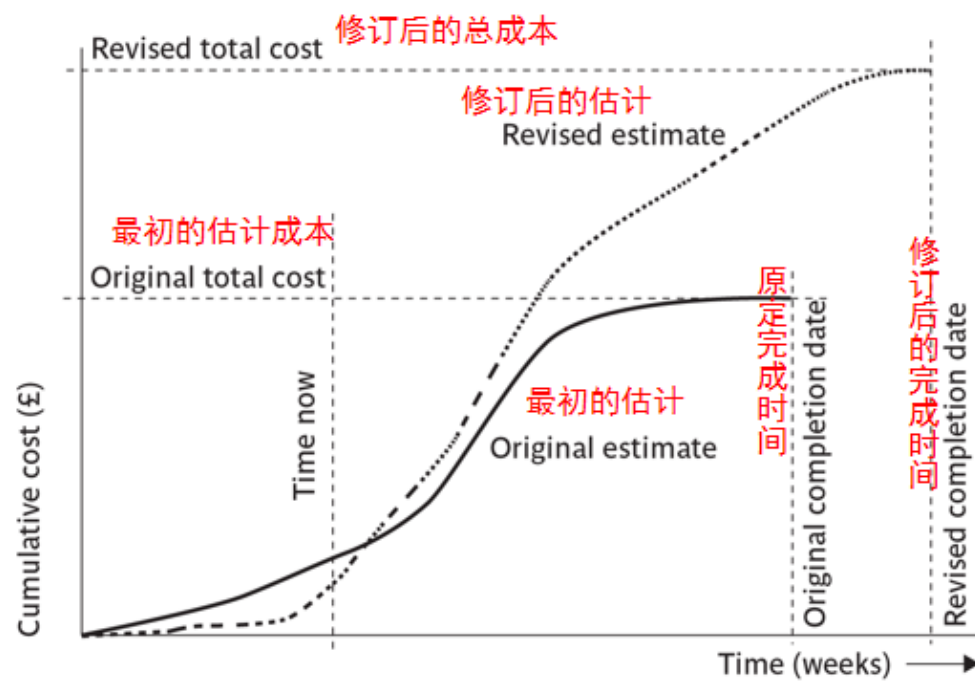
成本监控也能展示已经花费了多少劳力

。 简单的监控方法：累积消耗图



累积消耗图

- 对普通的累积消耗图上加上项目时间信息



- 为了实现 动态监督 动态改变 预算成本和周期
- 进行挣值分析

6、赢得值(挣值,盈余值) Earned Value

a、赢得值(EV, BCWP (budgeted cost of work performed, BCWP))

项目挣值是一个表示项目已完成作业量的计划价值的中间变量。

项目挣值 (EV) = 项目实际已完成作业量 (WP) * 已完成作业量的预算成本 (BC)

随着项目的进行，可以不断进行赢得值监控，判断项目的进度。

b、PV (planned value) 称为基线预算(baseline budget)或计划工作的预算成本 (budgeted cost of work scheduled, BCWS)

按计划的完成某任务的预算成本



第六节 项目挣值管理方法



三、项目挣值管理方法的三个关键变量

1. **项目计划成本** (Budgeted Cost of Work Schedule, **BCWS**) : 按照项目计划成本 (预算单价) 乘上项目计划工作量而得到的项目计划价值 (Plan Value, PV) 。
✓ $PV = \text{计划成本} * \text{计划工作量}$
2. **项目实际成本** (Actual Cost of Work Performed, **ACWP**) : 按照项目实际发生的成本 (实际单价) 乘上项目实际已完成工作量而得到的项目实际成本 (Actual Cost, AC) 。
✓ $AC = \text{实际成本} * \text{实际工作量}$
3. **项目挣值** (Budgeted Cost of Work Performed, **BCWP**) : 按照项目计划成本 (预算单价) 乘上项目实际工作量而得到的一个项目成本的中间变量 (Earned Value, EV) 。
✓ $EV = \text{计划成本} * \text{实际工作量}$

... ZL-2021

四、项目挣值管理方法的绩效分析变量

1. 项目成本/进度的绝对差异 (Cost Schedule Variance, **CSV**)

$$CSV = PV - AC = BCWS - ACWP \quad (>0, \text{好})$$

2. 项目成本绝对差异 (Cost Variance, **CV**)

$$CV = EV - AC = BCWP - ACWP \quad (>0, \text{好, 即没有超预算})$$

3. 项目进度绝对差异 (Schedule Variance, **SV**)

$$SV = EV - PV = BCWP - BCWS \quad (>0, \text{好, 即没有拖期/滞后})$$

4. 项目成本绩效指数 (Cost Performance Index, **CPI**)

$$CPI = EV / AC = BCWP / ACWP = \text{项目挣值} / \text{项目实际成本} \quad (>1, \text{好})$$

5. 项目进度绩效指数 (Schedule Completion Index, **SCI**)

$$SCI = EV / PV = BCWP / BCWS = \text{项目挣值} / \text{项目计划价值} \quad (>1, \text{好})$$

挣值分析—PV和EV例子

- 如下任务earned value – an example

- 详细设计计划 5工作量
- 编码 计划 8工作量
- 测试 计划 6工作量

到目前为止, 已经完成了编码, 但计划应该完成测试工作, 计算 PV, EV

$$PV = 5 + 8 + 6 = 19 \text{ 工作量}$$

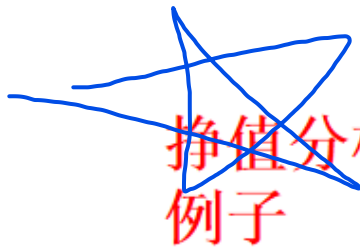
$$EV = 5 + 8 = 13 \text{ 工作量}$$

进度偏差 (Schedule variance) $SV = EV - PV = 13 - 19 = -6$

进度性能指标SPI (Schedule performance indicator)
 $= EV / PV = 13 / 19 = 0.68$

- SV 为负数或 SPI < 1.00, 项目说明在该时间点延期





挣值分析-实际成本AC(actual cost) 例子



- 如下任务earned value – an example
 - 详细设计计划 5工作量，实际花了3个工作量
 - 编码 计划 8工作量，实际花了4工作量
 - 测试 计划 6工作量
- 到目前为止，已经完成了编码，按计划应完成测试工作，计算PV, EV, AC
- $AC=3+4=7$ 工作量
- 成本偏差CV (Cost variance) $=EV-AC=13-7=6$ 工作量
- 成本性能指标CPI (Cost performance index) $=EV/AC=13/7=1.86$
- $CV>0$ or $CPI > 1.00$ 意思是 项目在预算范围内执行

EV 永远在前

赢得值监控



- 预算偏差(Budget variance): $AC - PV$
- 进度偏差(Schedule variance) $EV - PV$
- 成本偏差(Cost variance): $EV - AC$
- 性能比例(Performance ratios):
 - 成本性能指数: $CPI = EV$ (赢得值) / AC (实际的成本消耗)
 - 进度性能指数: $SPI = EV / PV$ (预算成本)
 - 值越大, 工作完成得越好

挣值分析-修订成本和周期



- CPI可以用来修正项目未来的完成成本预算 (EAC, Estimate at completion)
- $EAC =$ 项目当前计划的预算BAC (Budget at completion) / CPI
- Estimate at completion (EAC) = updated estimate = BAC / CPI

例如: 当期项目 预算费用£19,000 , $CPI = 1.86$

则 估计新预算 $EAC = BAC / CPI = £10,215$

SPI 可用来修正估计项目的周期

新的 完成 时间估计值 (TEAC) = 项目计划周期 / SPI

例如 项目计划23个月, $SPI = 0.88$

新的 完成 时间估计值 (TEAC) = 项目计划周期 / $SPI = 23 / 0.88 = 26.14$

假设你是一个软件项目管理者，受命为一个小型软件项目进行获得值统计。这个项目共计划了56个工作任务，估计需要582人.日，目前有12个工作任务已经完成。但是，按照项目进度，现在应该完成15个任务，下面给出相关进度安排数据（单位：人.日），请做出挣值分析

计算 项目的进度表执行指标SPI、进度偏差SV、预定百分比、完成百分比、成本执行指标CPI、成本偏差CV。

任务	计划工作量	实际工作量
1	12.0	12.5
2	15.0	11.0
3	13.0	17.0
4	8.0	9.5
5	9.5	9.0
6	18.0	19.0
7	10.0	10.0
8	4.0	4.5
9	12.0	10.0
10	6.0	6.5
11	5.0	4.0
12	14.0	14.5
13	16.0	—
14	6.0	—
15	8.0	—

- 进度表执行指标 $SPI = EV/PV$
- 进度表偏差 $SV = EV - PV$
- 预定完成百分比 = PV/BAC
- 完成百分比 = EV/BAC
- 已完成工作的实际成本 **ACWP**，是在项目进度表中某时间点已经完成的工作任务的实际工作量之和。然后，再计算：

- 成本执行指标 $CPI = EV/AC$
- 成本偏差 $CV = EV - AC$

$PV=15$ 个任务计划工作量的和 156.5

$EV=12$ 个任务计划工作量的和 126.5

$BAC=$ 所有任务计划工作量的和 582

$AC=12$ 个任务实际工作量的和 127.5

$SPI= 126.5/156.5 = 0.8$

$SV=126.5-156.5=-30$

预定完成百分比 = $PV/BAC=156.5/582=26\%$

完成百分比 = $EV/BAC=126.5/582=21\%$

成本执行指标 $CPI = EV/AC=126.5/127.5=0.99$

成本偏差 $CV = EV - AC=126.5-127.5=-1$

School of Computer

Estimation for Software

c、基线预算

- 建立赢得值分析的第一步是为项目建立一个基线预算 (baseline budget)
- 预算基线是建立在项目计划的基础上的，它是随时间显示赢得值的预测。

- 赢得值可以用货币单位来衡量，在人员密集的案例里,如软件开发,通常可以用人员工作量(人-时或工作日)来衡量或PV的百分比表示。

7、监控的优先级Prioritizing monitoring

1. 列出优先级以决定监控的层次
2. 关键路径活动
3. 没有自由浮动的活动
4. 小自由浮动时间的活动
5. 高风险的活动
6. 使用关键资源的活动

8、使项目回到目标Getting the project back to target

几乎所有的项目都会遇到延误和意外事件。项目经理的一项任务就是识别这些事件发生的时间，在最小延迟时间和对项目团队有最小的影响的情况下，消除问题的影响。

a、缩短关键路径

- 要求项目组人员“Work harder”有一些效果，但是不能轻易使用。
- 分配额外的资源可以加快进度，但是并不总是奏效，例如分配给某一人员的小模块，再增加一个人员并不一定能够缩短时间。
- 将非关键路径上的资源调整到关键路径上
- 注意：缩短关键路径可能使其它路径成为关键路径。

1. 为关键任务重新分配人力资源
2. 减少交付范围（功能）
3. 降低质量要求

b、重新考虑任务优先关系

- 网络计划考虑的是理想情况和普通工作情况，因而在无法缩短关键路径时，可以重新考虑任务优先关系。
- 另一种方法是将活动再进行划分，从而一部分可以与其它活动并行。
- 重新考虑任务优先关系可能带来风险或者质量上的影响。

9、变更控制（Change Control）

- 用户的需求可能变化，项目内部可能变化.....
- 变更需要仔细考虑，因为一个部分的变化可能会对另外部分的造成影响
- 问题：对程序描述的改变将引起软件的设计和代码的改变，还有什么其它产品可能需要修改？
- 答案：测试数据，期待结果和用户手册等

a、变更管理员角色

- **Configuration Librarian**
- **责任：**
 - 识别所有需要变更控制的内容
 - 建立一个保存所有项目文档和代码的中央库

建立一套管理变更的正式过程
维护读取库中内容的记录和库中每一项的状态

b、变更控制过程

- 用户意识到需要对系统进行修改，考虑将修改请求提交给开发人员
- 用户端的管理者考虑是否将该修改请求提交给项目承担者
- 项目承担端的管理者将该任务指派给一个成员，该成员将判断该修改的成本以及修改的影响，并提交一个报告
- 该报告被提交给用户，用户将考虑是否能够承受额外的成本
- 用户同意后，一个或多个开发者被授权从主产品上取出要修改的部分的拷贝
- 拷贝被修改。
- 新版本开发出来后，将通知用户，用户进行接受测试
- 当用户满意后，产品的配置项被新版本所代替。

10、项目修复

a、修复方案

问题：如何挽救项目

1. 缩减项目规模，以便在计划的时间与工作量内完成项目
2. 把注意力放在短期的改善上，以提高过程的生产率
3. 面对现实，放弃计划

b、修复计划

- 评估你的处境
- 应用W-理论分析
- 作好修复项目的思想准备
- 向开发组成员探问拯救项目的方法
- 变得现实一些

项目修复：人员

1. 消除重大领导问题
2. 更换子项目经理
3. 为经理配备助手
4. 增加新手一定要慎重
充分利用开发人员的时间
减轻他们其它的负担
为他们处理一些日常工作
5. 允许开发组人员各有不同
绝不允许破坏士气
6. 观察开发人员的节奏
不断根据修复的情况来调整安排

项目修复：过程

1. 修正软件开发过程中出问题的环节
出问题的环节必然是项目有意或无意忽略了软件开发的基本原则

2. 创建详细的小型里程碑
小型的（一、两天规模的）
二元性（要么完成，要么没有完成）
彻底性（所有里程碑完成，项目就完成了）
3. 依据里程碑的完成来安排进度
为每个里程碑设置完成的时间，里程碑的设置必然是考虑了人员正常的工作时间

项目修复：过程

1. 细致地最终进度进展情况
每天检查小型里程碑的完成情况
误了某个里程碑，就要求他加班加点完成
2. 记录里程碑未完成的原因
3. 短期后再调整——一周或两周
4. 慎重提交进度计划
经过一两周的实际检验
5. 严格的风险管理

项目修复：产品

- 稳定需求
- 修正特性集
- 删除优先级低的需求
- 评估你的政治地位
是否本身产品就是不重要的？
目前比产品更重要的是什么？
- 去除产品中没用的“垃圾”
- 降低缺陷数目，并要持续降低
公开缺陷图

