中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

软件工程

主要内容

第一章 软件工程学概述
第二章 可行性研究
第三章 需求分析
第四章 总体设计
第五章 详细设计
第六章 编码与测试
第七章 软件维护
第八章 面向对象方法学
第九章 面向对象分析设计与实现
第十章 软件项目管理

第十章　软件项目管理

第一节　软件项目管理概述

第二节　软件规模估算

第三节　工作量估算

第四节　估算开发时间

第五节　进度计划

第六节　人员组织

第七节　质量保证

第八节　软件配置管理

第十章　软件项目管理

第一节　软件项目管理概述

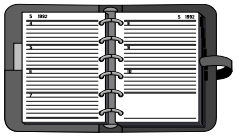
一、为什么要学习软件项目管理

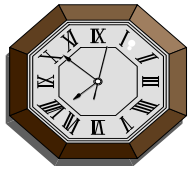
- 残酷的现实
- 成功的渴望
- 未来发展的需要

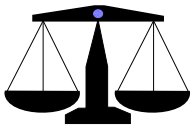
二、经理管什么？

预算


组 织


计 划


进 度


标 准

三、软件项目管理基本概念

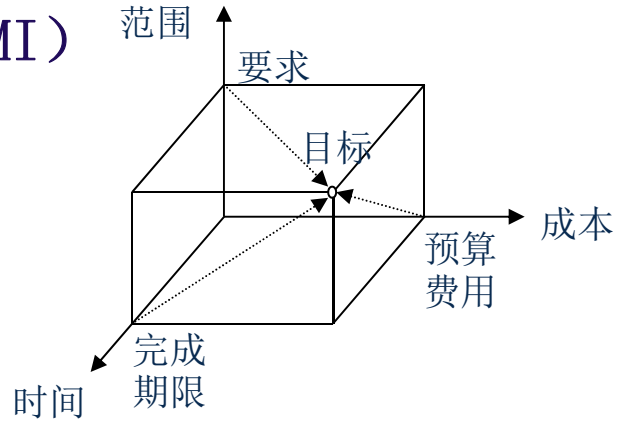
1. 项目

美国项目管理协会(PMI)的定义：项目是为完成某一独特的产品或服务所做的一次性努力。项目具有的基本特性：

- 项目的一次性
- 项目的组织性
- 项目的生命期
- 项目的资源消耗性
- 项目后果的不确定性
- 项目的目标冲突性

2. 项目管理

在项目活动中运用一系列的知识、技能、工具和技术，以满足或超过相关利益者对项目的要求。（PMI）



3. 软件项目管理

软件项目是指对软件系统进行开发、集成和服务为主要目的的项目。软件项目管理和其他项目管理相比，具有很大的独特性：

- 软件项目管理尚不规范，经验成分明显
- 过程没有明显的划分
- 大都是“一次性”的人力消耗型项目

第二节　软件规模估算

一、代码行技术

请多位专家估算程序的最小规模 a ，最可能的规模 m和最大规模 b。以三组平均值估算程序规模：

$$L = \frac{\overline{a} + 4\overline{m} + \overline{b}}{6}$$

单位是代码行数（LOC），或是千行代码数（KLOC）。

二、功能点技术

功能点技术依据对软件信息域特性和软件复杂性的评估结果，估算软件规模。这种方法用功能点（FP）为单位度量软件规模。

① 计算未调整的功能点数UFP：

$$UFP=a_1\times Inp+a_2\times Out+a_3\times Inq+a_4\times Maf+a_5\times Inf$$

其中，$a_i(1\leqslant i\leqslant 5)$是信息域特性系数，其值由相应特性的复杂级别决定；输入项数(Inp)、输出项数(Out)、查询数(Inq)、主文件数(Maf)和外部接口数(Inf)。

② 计算技术复杂性因子TCF：　$TCF=0.65+0.01\times DI$

其中，$DI=\sum_{i=1}^{14}F_i$，F_i为14种技术因素对软件规模的影响。

③ 计算功能点数FP：　$FP=UFP\times TCF$

第三节　工作量估算

软件估算模型使用由经验导出的公式来预测软件开发工作量，工作量是软件规模（KLOC或FP）的函数，工作量的单位通常用人月（pm)表示。

一、静态单变量模型

- 这类模型的总体结构形式：　$E=A+B\times(ev)^C$

其中，E是以人月为单位的工作量，A、B和C是由经验数据导出的常数，ev是估算变量（KLOC或FP）。下面给出几个典型的静态单变量模型：

第十章　软件项目管理

1. 面向KLOC的估算模型

(1) Walston_Felix模型：　$E=5.2\times(KLOC)^{0.91}$

(2) Bailey_Basili模型：　$E=5.5+0.73\times(KLOC)^{1.16}$

(3) Boehm简单模型：　$E=3.2\times(KLOC)^{1.05}$

(4) Doty模型（在KLOC>9时适用）：$E=5.288\times(KLOC)^{1.047}$

2. 面向FP的估算模型

(1)　Albrecht & Gaffney模型：

$E=-13.39+0.0545FP$

(2) Maston,Barnett和Mellichamp模型：

$E=585.7+15.12FP$

二、COCOMO2（Constructive Cost Model）：

$$MM = a \times KLOC^{b} \times \prod_{i=1}^{17} f_i$$

其中：MM是开发工作量（以人月为单位），

a 是模型系数，

KLOC 是估计的源代码行数（以千行为单位），

b 是模型指数，

f_i(i=1~17) 是成本因素，每个成本因素都根据它的重要程度和对工作量影响大小被赋予一定数值（称为工作量系数）。

支持大多数估算模型的经验数据，都是从有限个项目的样本集中总结出来的，因此，没有一个估算模型可以适用于所有类型的软件和开发环境。

第四节　估算开发时间

成本估算模型也同时提供了估算开发时间的方程，但与工作量估算方程不同的是，各种估算开发时间的方程很相似。

(1) Walston_Felix模型：　$T=2.5E^{0.35}$

(2) 原始的COCOMO模型：　$T=2.5E^{0.38}$

(3) COCOMO2模型：　$T=3.0E^{0.33+0.2\times(b-1.01)}$

(4) Putnam模型：　$T=2.4E^{1/3}$

其中，E是开发工作量（以人月为单位），T是开发时间（以月为单位）。

第五节　进度计划

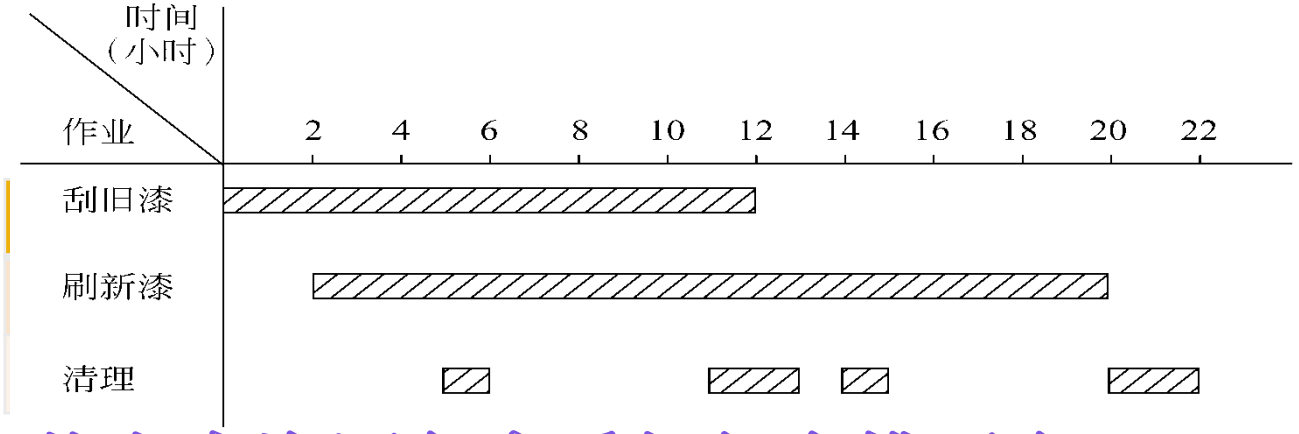
一、甘特图

甘特图也称为条型图，或横道图。它以横坐标表示时间，工程活动在图的左侧纵向排列，以活动所对应的横道位置表示活动的起始时间，横道的长短表示持续时间的长短。它是一种比较简便的工期计划和进度安排工具。

实例分析：

有一座陈旧的矩形木板房（4面墙壁）需要重新刷油漆。这项工作可以分为3步完成：首先刮掉旧漆，然后刷上新漆，最后清除溅在窗户上的油漆。假设一共分配了15名工人去完成这项工作，然而工具却很有限：只有5把刮旧漆用的刮板，5把刷漆用的刷子，5把清除溅在窗户上油漆用的小刮刀。

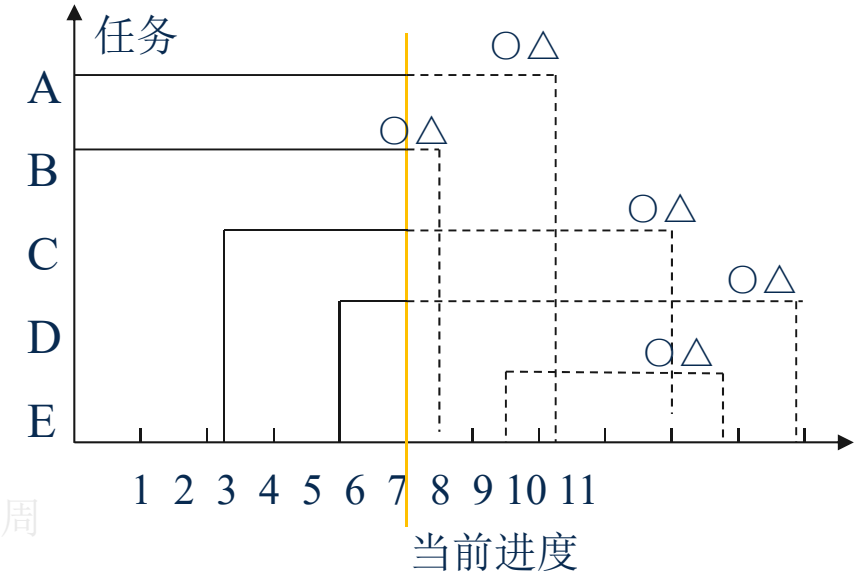
怎样安排才能使工作进行得更有效呢？



思考：从人力资源角度看如何安排更合理？

改进的甘特图

- 甘特图优点：简单，能够动态反映软件开发的进展情况。
- 甘特图缺点：不能够反映多个任务之间的复杂逻辑关系。



二、关键路线法（CPM）

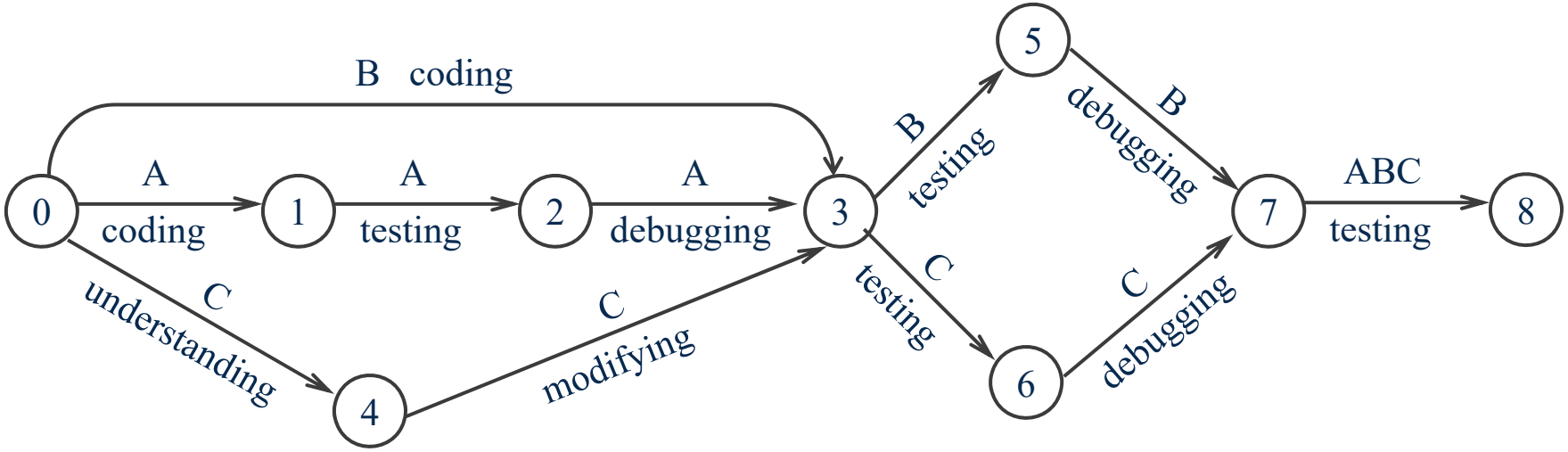
用箭头表示活动（作业），用圆圈表示事件（表示一项活动的开始或结束）。 注意：事件是时间点，不消耗时间和资源；虚箭头仅仅表示活动之间的依赖关系。

- 时差为零的活动是关键活动，其总长度决定了项目总工期；
- 一系列贯穿项目始终的关键活动构成关键路径；
- 关键路径在整个项目执行过程中是可能发生变化的；
- 关键路径决定了项目的最短完成时间，确定方法分2步：

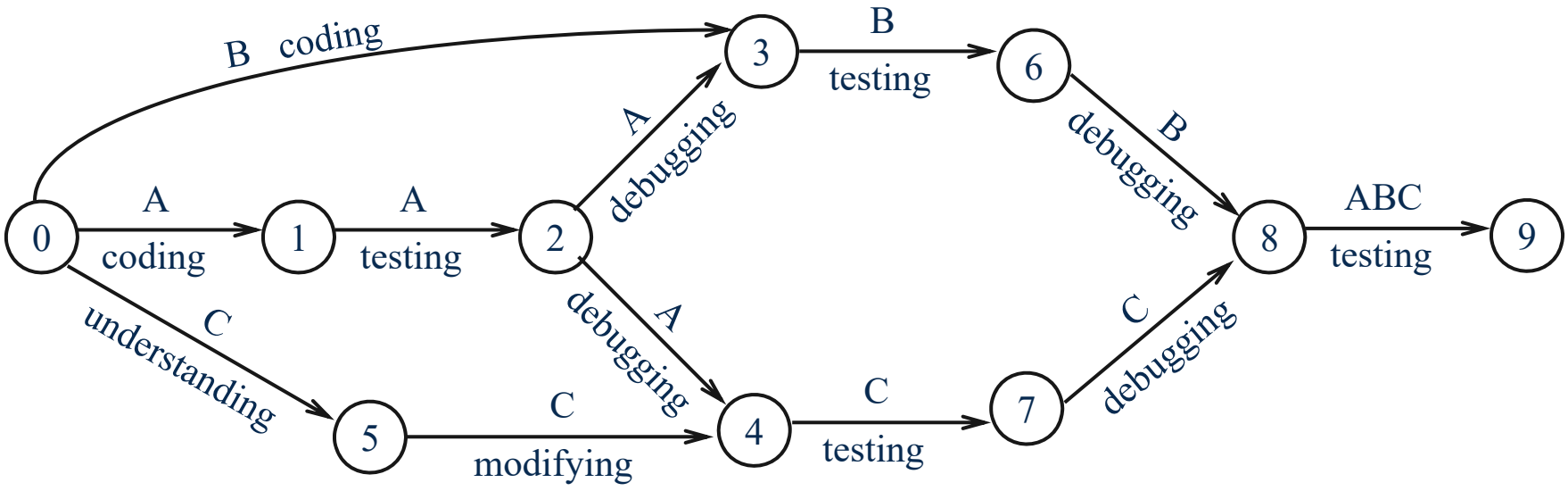
① **正推路径方法：决定了网络中每个活动的最早开始时间**(ES)**和最早完成时间**(EC)；

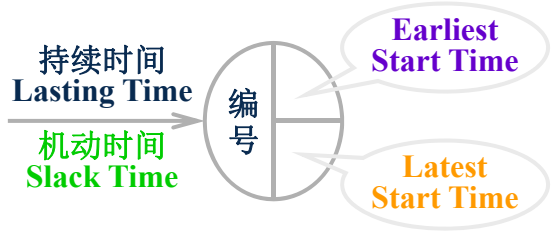
② **逆推路径方法：决定了每个活动的最晚开始时间**(LS)**和最晚完成时间**(LC)；

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。



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。





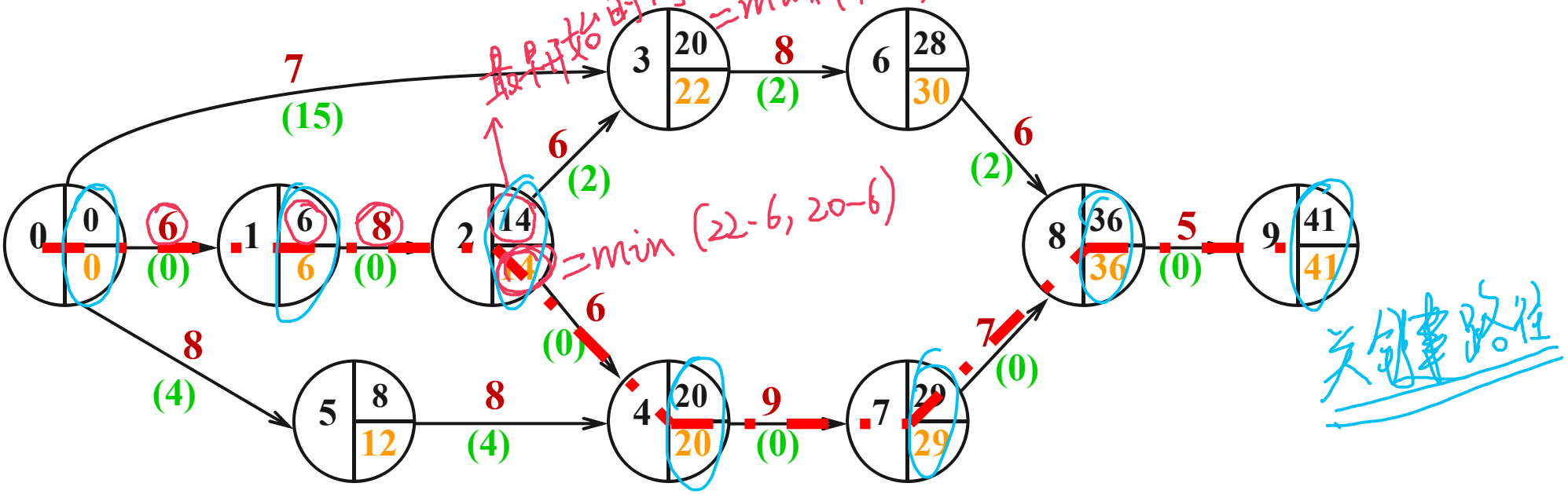
(1) 标出 Lasting Time(LT)

(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



对关键路径的计算与调整优化

- 清醒的认识：关键路径是网络图中最长的线路，它决定了项目的总耗时。
- 必须把注意力集中于那些优先等待完成的任务，确保它们准时完成。关键路径上的推迟即是整个项目推迟。
- 向关键路径要时间，向非关键路径要资源。
- 调整进度，平衡资源。

第六节　人员组织

1. 民主分权式开发组织

没有领导者、提倡无私精神的团体组织，民主氛围浓郁，组员们工作积极性高，这使得整个团队能多出、快出更高质量的产品。

- 民主分权组织方式比较强调个人的作用，所以希望小组成员都是经验丰富、技术和技能熟练的人员。
- 民主分权式开发组织方式特别适用于较小规模或研究型产品的开发。

2. 控制集权式组织

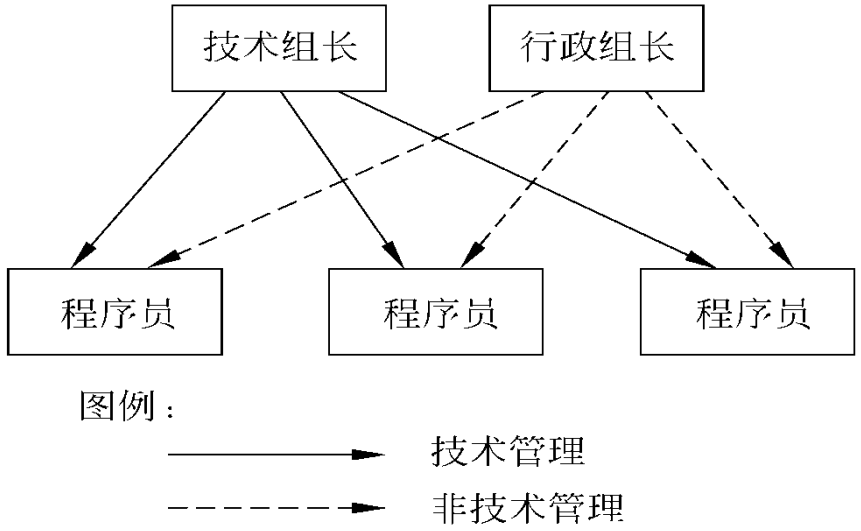
控制集权式组织由一名高级工程师（主程序员）、一名后备工程师、资料管理员，以及2-5个技术人员组成。

- 特点：一是专业化，每个成员分工明确，执行各自的专业任务；二是层次性，每个成员在组织中处于一定的领导或被领导地位。
- 小组负责人由高级工程师（主程序员）担任，他既是管理者，又是高级专业人员，负责计划、协调和复审小组的所有技术活动；后备工程师是协助负责人工作的专业人员；资料管理员是专职的，职责是控制和维护所有的软件配置，协助小组进行研究、评估和文档准备。

3. 控制分权式组织

软件的开发通常采用一种更合理的、责任范围更清楚的人员组织方式---控制分权式组织。

控制分权开发组由一个技术负责人（负责小组的技术活动）和一个行政负责人（负责所有非技术的管理决策）两个人承担。

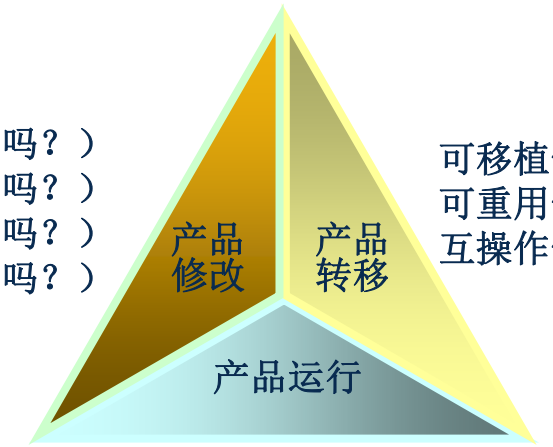


第七节　质量保证

1. 软件质量的定义

软件质量就是“软件与明确地和隐含地定义的需求相一致的程度”。

可理解性（我能理解它吗？）
可维护性（我能修改它吗？）
灵活性　（我能改变它吗？）
可测试性（我能测试它吗？）



可移植性（我能在另一台机器上使用它吗？）
可重用性（我能重用它的某些部分吗？）
互操作性（我能把它和另一个系统结合吗？）

正确性（它按我的需要工作吗？）
健壮性（对意外环境它能适当地响应吗？）
效率（完成预定功能时它需要的计算机资源多吗？）
完整性（它是安全的吗？）
可用性（我能使用他吗？）
风险（能按预定计划完成它吗？）

2. 软件质量保证措施

软件质量保证（Software Quality Assurance,SQA）的措施主要有：基于非执行的测试（也称为复审或评审，包括走查、审查等技术方法），基于执行的测试（即以前讲过的软件测试）和程序正确性证明。

- 复审主要用来保证在编码之前各阶段产生的文档的质量；
- 基于执行的测试需要在程序编写出来之后进行，是保证软件质量的最后一道防线；
- 程序正确性证明使用数学方法严格验证程序是否与对它的说明完全一致。

第八节　软件配置管理

- 软件配置管理是在软件的整个生命期内管理变化的一组活动：
  ① 标识变化；
  ② 控制变化；
  ③ 确保适当地实现了变化；
  ④ 向需要知道这类信息的人报告变化。
- 配置管理是在软件项目启动时就开始，并且一直持续到软件退役后才终止的一组跟踪和控制活动。
- 软件配置管理的目标是，使变化更正确且更容易被适应，在必须变化时减少所需花费的工作量。

1. 软件配置项

软件过程的输出信息可以分为3类：

① 计算机程序（源代码和可执行程序）；

② 描述计算机程序的文档（供技术人员或用户使用）；

③ 数据（程序内包含的或在程序外的）。

上述这些项组成了在软件过程中产生的全部信息，我们把它们统称为软件配置，而这些项就是软件配置项。

2. 基线

基线是一个软件配置管理概念。

IEEE把基线定义为：已经通过了正式复审的规格说明或中间产品，它可以作为进一步开发的基础，并且只有通过正式的变化控制过程才能改变它。

简而言之，基线就是通过了正式复审的软件配置项。

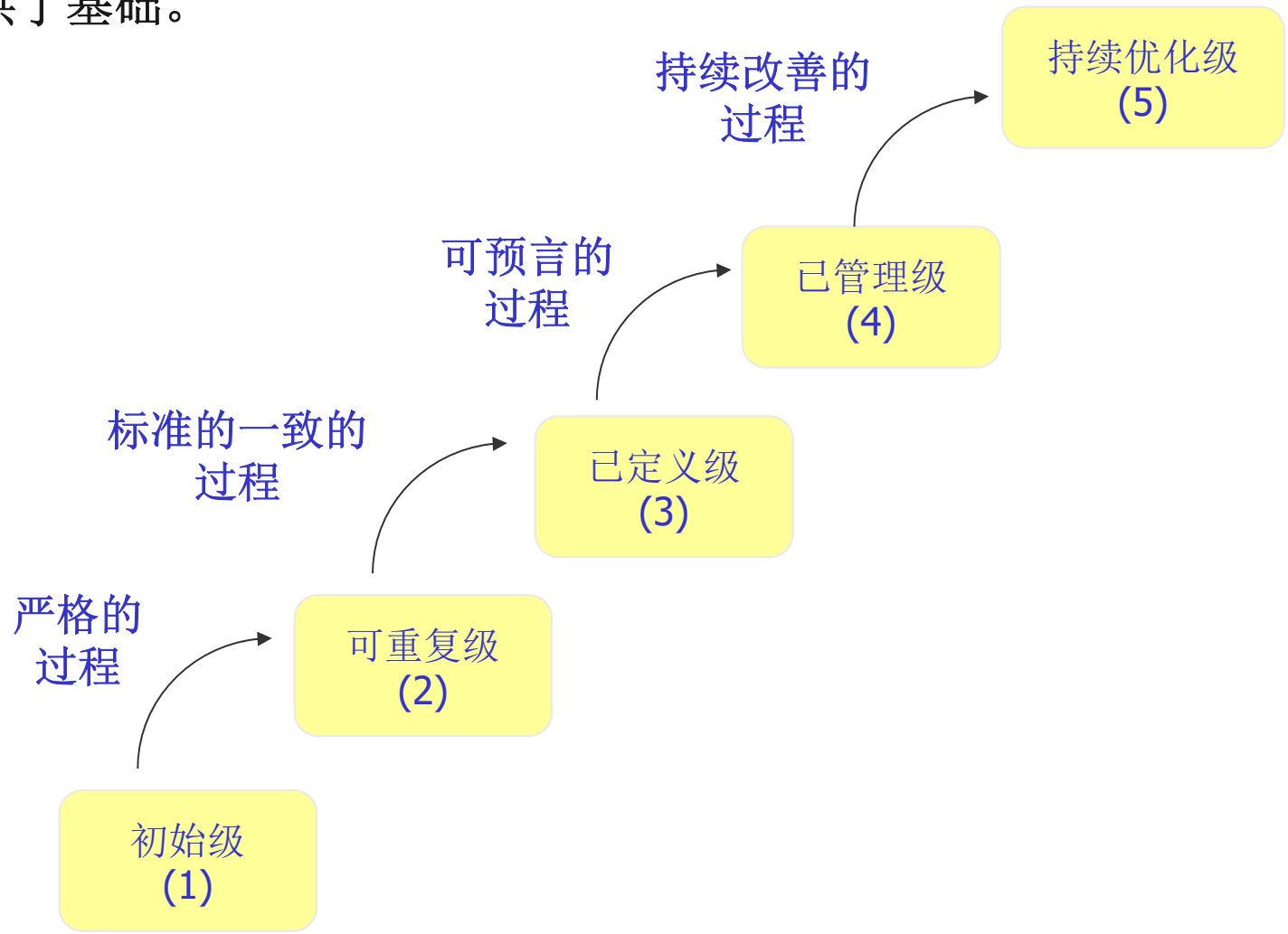
第九节　能力成熟度模型

- CMM是什么

CMM（Capability Maturity Model）是用于衡量软件过程能力的事实上的标准，同时也是目前软件过程改进最好的参考标准。

美国卡内基-梅隆大学软件工程研究所（SEI）研制

CMM提供了将这些演化步骤组织为5个成熟度级别的框架，这为持续的过程改进提供了基础。



- 初始级
  - ✓ 组织：组织通常没有提供开发和维护软件的稳定的环境。
  - ✓ 项目：当发生危机时，项目通常放弃计划的过程，回复到编码和测试。
  - ✓ 过程能力：不可预测。**(unpredictable)**
- 可重复级
  - ✓ 组织：将软件项目的有效管理过程制度化，这使得组织能够重复以前项目中的成功实践。
  - ✓ 项目：配备了基本的软件管理控制。
  - ✓ 过程能力：严格的。**(disciplined)**
- 已定义级
  - ✓ 组织：在组织范围内开发和维护软件的标准过程被文档化，其中包括软件工程过程和管理过程，它们集成为一个一致的整体。
  - ✓ 项目：对组织的标准软件过程进行裁剪，来开发它们自己的定义软件过程。
  - ✓ 过程能力：标准的和一致的。**(standard and consistent)**

- 已管理级
  - ✓ 组织：为软件产品和过程都设定了量化的质量目标。
  - ✓ 项目：项目减小过程性能的变化性，使其进入可接收的量化边界，从而达到对产品和过程的控制。
  - ✓ 过程能力：可预言的。**(predictable)**
- 持续优化级
  - ✓ 组织：关注于持续的过程改进。
  - ✓ 项目：软件过程被评价，以防止过失重复发生，从中获得的教训散布给其它项目。
  - ✓ 过程能力：持续的改善。**(continuously improving)**


Thank
You