第10章 项目管理

教学内容

10.1 估算软件规模

10.2 工作量估算

10.3 进度计划

10.4 人员组织

10.5 质量保证

10.6 软件配置管理

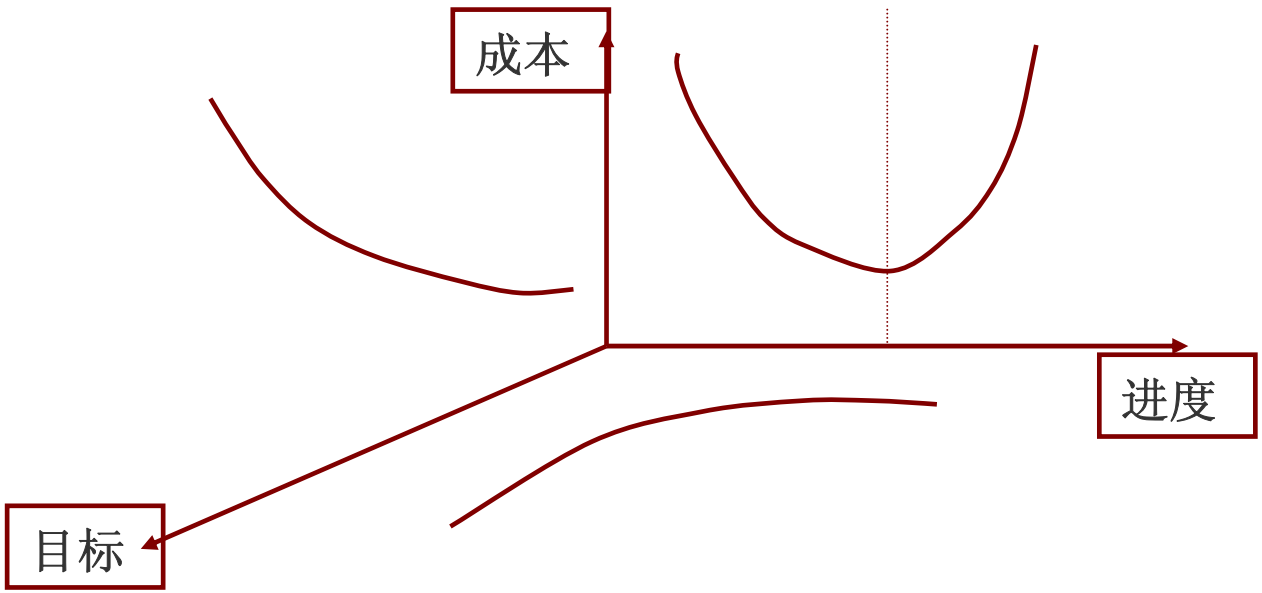
10.7 能力成熟度模型

概述

- 软件项目管理就是对软件工程项目开发过程的管理。具体地说，就是对整个软件生存期的一切活动进行管理，以达到提高生产率、改善产品质量的目的。
- 软件项目管理的职能
  - （1）制定计划：规定待完成的任务、要求、资源、人力和进度等。
  - （2）建立项目组织：为实施计划，保证任务的完成，需要建立分工明确的责任机构。
  - （3）配备人员：任用各种层次的技术人员和管理人员。
  - （4）指导：鼓励和动员软件人员完成所分配的任务。
  - （5）检验：对照计划或标准监督检查实施的情况。

概述

- 项目管理中的三要素：目标、成本、进度
  - 目标、成本、进度三者在项目管理过程中是互相制约的



10.1 估算软件规模

- 为了估算项目的工作量和完成期限，首先需要估算软件的规模。
- 两种方法：
  - 代码行技术
    - 依据以往开发类似产品的经验和历史数据，估计所需要的源程序行数，是比较简单的定量估算的方法。
  - 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。用功能点（FP）为单位度量软件规模。

10.1.1 代码行技术

- 为了使估计值更接近实际值，可以由多名有经验的软件工程师分别做出估计，算出平均值。
- 用代码行技术估算软件规模时，单位是代码行数（**LOC**），千行代码数（**KLOC**）。
- 优缺点
  - 优点是代码是所有软件开发项目都有的，且容易计算行数。
  - 缺点是源程序仅是软件配置的一个成分，用它的规模代表整个软件的规模似乎不太合理；用不同语言实现同一个软件所需要的代码行数并不相同；这种方法不适用于非过程语言。

10.1.2 功能点技术

- 功能点技术定义了信息域的5个特性，分别是输入项数(Inp)、输出项数(Out)、查询数(Inq)、主文件数(Maf)和外部接口数(Inf)。
- 估算功能点的步骤
  - 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
  - 计算技术复杂性因子TCF

    $TCF=0.65+0.01\times DI$　　$DI=(\sum_{i=1}^{14}F_i)$
  - 计算功能点数FP

    $FP=UFP\times TCF$

10.2 工作量估算

- 工作量是软件规模（KLOC或FP）的函数，工作量的单位通常是人月（pm)。
- 几种模型
  - 静态单变量模型
  - 动态多变量模型
  - COCOMO2模型

10.2.1 静态单变量模型

- 总体结构形式如下： $E=A+B\times(ev)^C$
  - A、B和C是由经验数据导出的常数，E是以人月为单位的工作量，ev是估算变量（KLOC或FP）。
- 1. 面向KLOC的估算模型
  - Walston_Felix模型 $E=5.2\times(KLOC)^{0.91}$
  - Bailey_Basili模型 $E=5.5+0.73\times(KLOC)^{1.16}$
  - Boehm简单模型 $E=3.2\times(KLOC)^{1.05}$
  - Doty模型（在KLOC>9时适用）

    $E=5.288\times(KLOC)^{1.047}$

10.2.1 静态单变量模型

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

10.2.2 动态多变量模型

- 也称为软件方程式，是根据从4000多个当代软件项目中收集的生产率数据推导出来的。该模型把工作量看作是软件规模和开发时间这两个变量的函数。
- 动态多变量估算模型的形式如下：

$$E=(LOC\times B^{0.333}/P)^3\times(1/t)^4$$

  - E是以人月或人年为单位的工作量；
  - t是以月或年为单位的项目持续时间；
  - B是特殊技术因子，随着对测试、质量保证、文档及管理技术的需求增加而缓慢增加，对于较小的程序（KLOC=5~15），B=0.16, 对于超过70 KLOC的程序，B=0.39;
  - P是生产率参数。开发实时嵌入式软件时，P的典型值为2000；开发电信系统和系统软件时，P=10000；对于商业应用系统来说，P=28000。

10.2.3 COCOMO2模型

- 构造性成本模型（constructive cost model)，Boehm提出。
- 给出了3个层次的软件开发工作量估算模型
  - 应用系统组成模型
    - 这个模型主要用于估算构建原型的工作量，模型名字暗示在构建原型时大量使用已有的构件。
  - 早期设计模型
    - 这个模型适用于体系结构设计阶段。
  - 后体系结构模型
    - 这个模型适用于完成体系结构设计之后的软件开发阶段。

10.3 进度计划

- 项目管理者的目标是定义全部项目任务，识别出关键任务，跟踪关键任务的进展状况，以保证能及时发现拖延进度的情况。为达到上述目标，管理者必须制定一个足够详细的进度表，以便监督项目进度并控制整个项目。
- 软件项目的进度安排是这样一种活动，它通过把工作量分配给特定的软件工程任务并规定完成各项任务的起止日期，从而将估算出的项目工作量分布于计划好的项目持续期内。进度计划将随着时间的流逝而不断演化。

10.3.1 估算开发时间

- 几种方法
  - (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

    $T=2.4E^{1/3}$

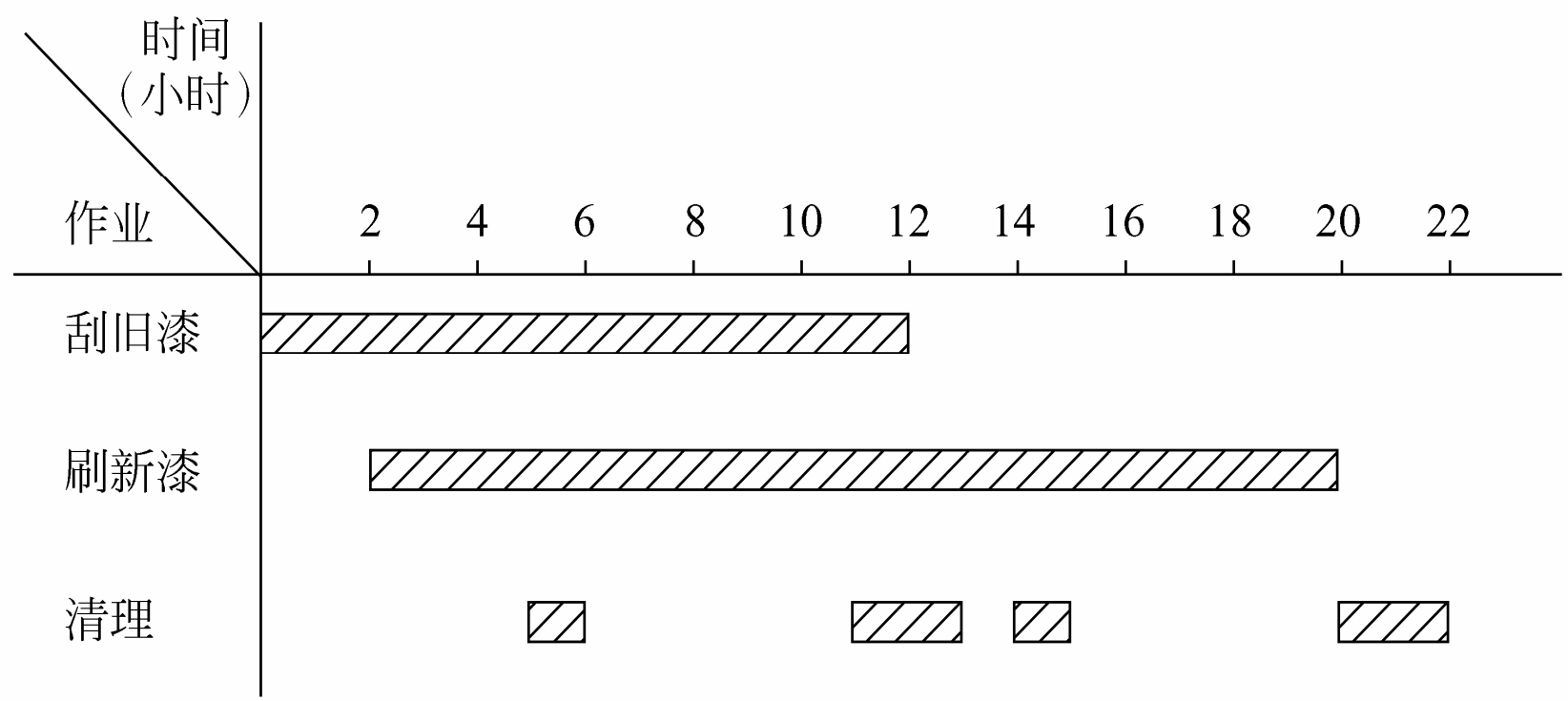
E是开发工作量（以人月为单位），T是开发时间（以月为单位）。

10.3.2 Gantt图

- 甘特图
  - 是历史悠久、应用广泛的制定进度计划的工具，能形象地描绘任务分解情况，以及每个子任务(作业)的开始时间和结束时间，因此是进度计划和进度管理的有力工具。
- 优点：
  - 直观简明和容易掌握、容易绘制
- 缺点：
  - 不能显式地描绘各项作业彼此间的依赖关系；
  - 进度计划的关键部分不明确，难于判定哪些部分应当是主攻和主控的对象；
  - 计划中有潜力的部分及潜力的大小不明确。

10.3.2 Gantt图

- 例：
  - 假设需要油漆一座矩形木板房。这项工作必须分3步完成： 先刮旧漆，然后刷新漆，最后清除溅在窗户上的油漆。假设共15名工人去完成这项工作，然而工具有限： 只有5把刮旧漆用的刮板，5把刷漆用的刷子，5把清除窗户上油漆用的小刮



10.4 人员组织

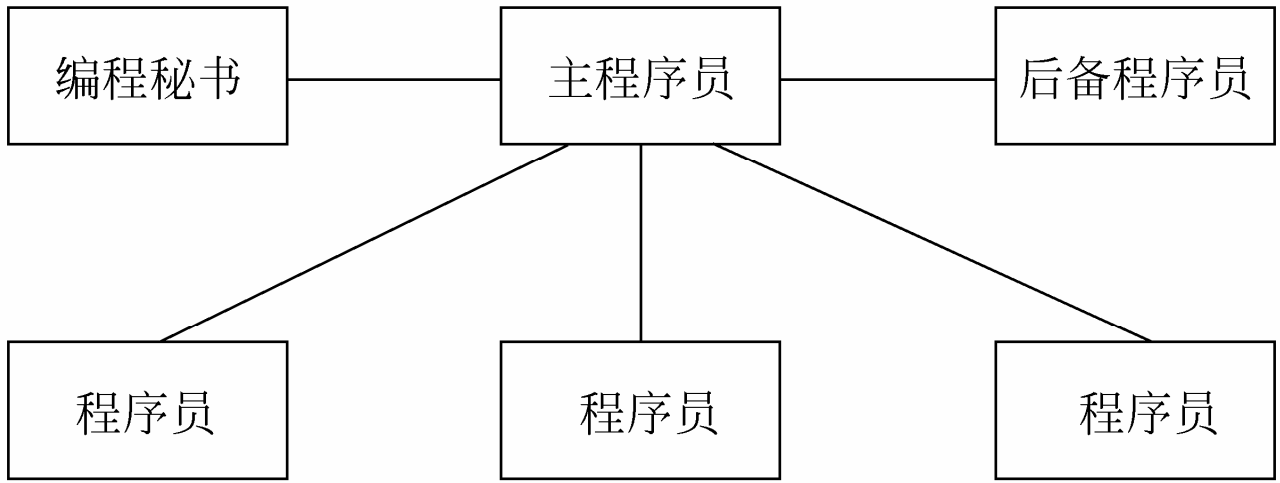
- 软件项目成功的关键是有高素质的软件开发人员。必须把多名软件开发人员合理地组织起来，使他们有效地分工协作共同完成开发工作。
- 经验表明，项目组组织得越好，其生产率越高，而且产品质量也越好。
- 3种典型的组织方式：
  - 民主制程序员组
  - 主程序员组
  - 现代程序员组

10.4.1 民主制程序员组

- 通常采用非正式的组织方式，小组成员完全平等，享有充分民主，通过协商做出技术决策。程序设计小组的规模应该比较小,以2～8名成员为宜。
- 主要优点：
  - 组员们对发现程序错误持积极的态度，这种态度有助于更快速地发现错误，从而导致高质量的代码。
  - 组员们享有充分民主，小组有高度凝聚力，组内学术空气浓厚，有利于攻克技术难关。
- 缺点：
  - 需要经验丰富技术熟练的程序员，否则由于没有明确的权威指导开发工程的进行,组员间将缺乏必要的协调,最终可能导致工程失败。

10.4.2 主程序员组

- 美国IBM公司在20世纪70年代初期开始采用主程序员组的组织方式。采用的考虑：
  - (1) 软件开发人员多数比较缺乏经验；
  - (2) 程序设计过程中有许多事务性的工作，例如，大量信息的存储和更新；
  - (3) 多渠道通信很费时间，将降低程序员的生产率



10.4.2 主程序员组

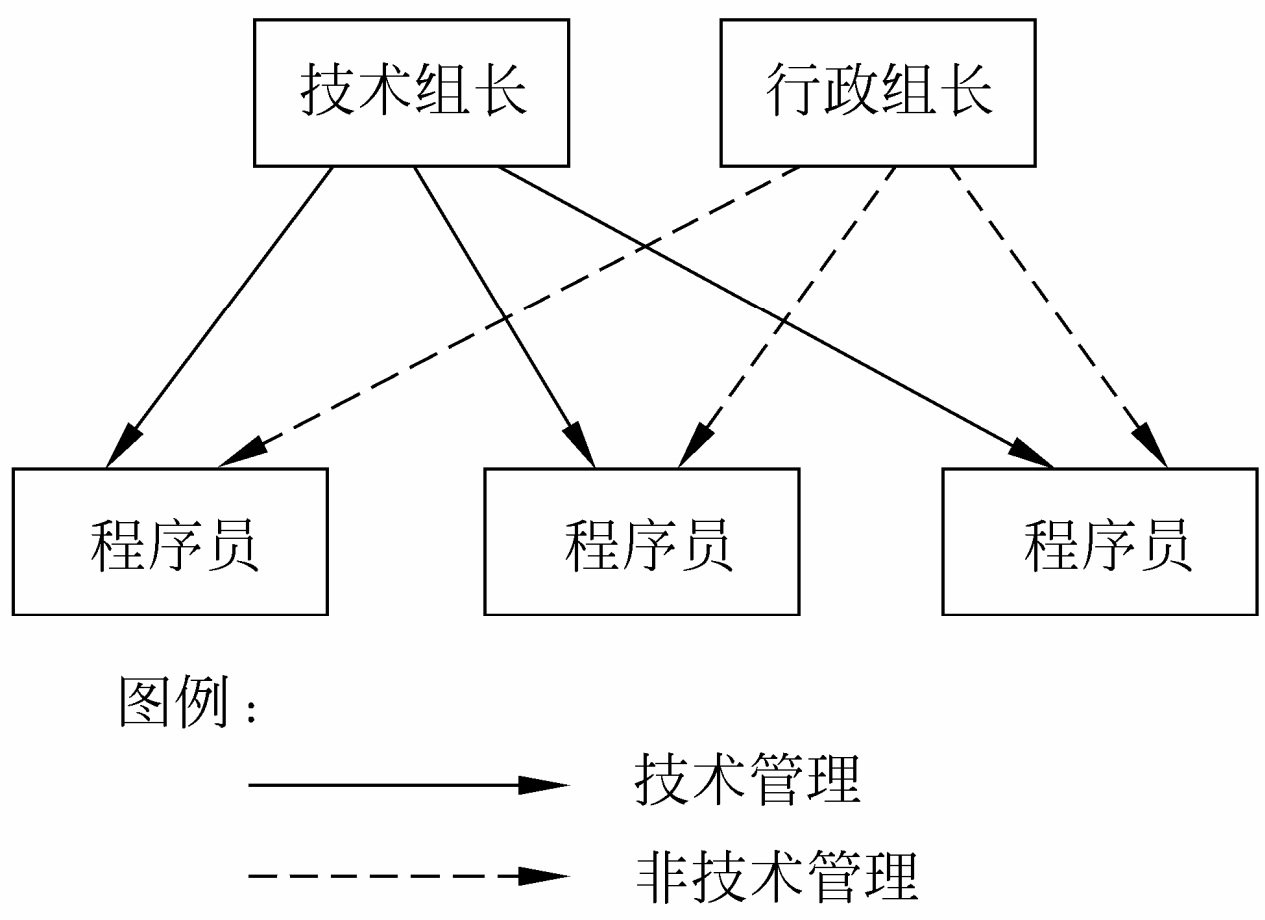
- 主程序员组核心人员的分工如下所述：
  - 主程序员既是成功的管理人员又是经验丰富、技术好、能力强的高级程序员，负责体系结构设计和关键部分（或复杂部分）的详细设计，并且负责指导其他程序员完成详细设计和编码工作。
  - 后备程序员也应该技术熟练而且富于经验，他协助主程序员工作并且在必要时接替主程序员的工作。
  - 编程秘书负责完成与项目有关的全部事务性工作，例如，维护项目资料库和项目文档，编译、链接、执行源程序和测试用例。

10.4.2 主程序员组

- 缺点：
  - 主程序员应该是高级程序员和优秀管理者的结合体，在现实社会中这样的人才并不多见。
  - 后备程序员更难找。人们期望后备程序员像主程序员一样优秀，但是，他们必须坐在“替补席”上，拿着较低的工资等待随时接替主程序员的工作。几乎没有一个高级程序员或高级管理人员愿意接受这样的工作。
  - 编程秘书也很难找到。专业的软件技术人员一般都厌烦日常的事务性工作，但是，人们却期望编程秘书整天只干这类工作。

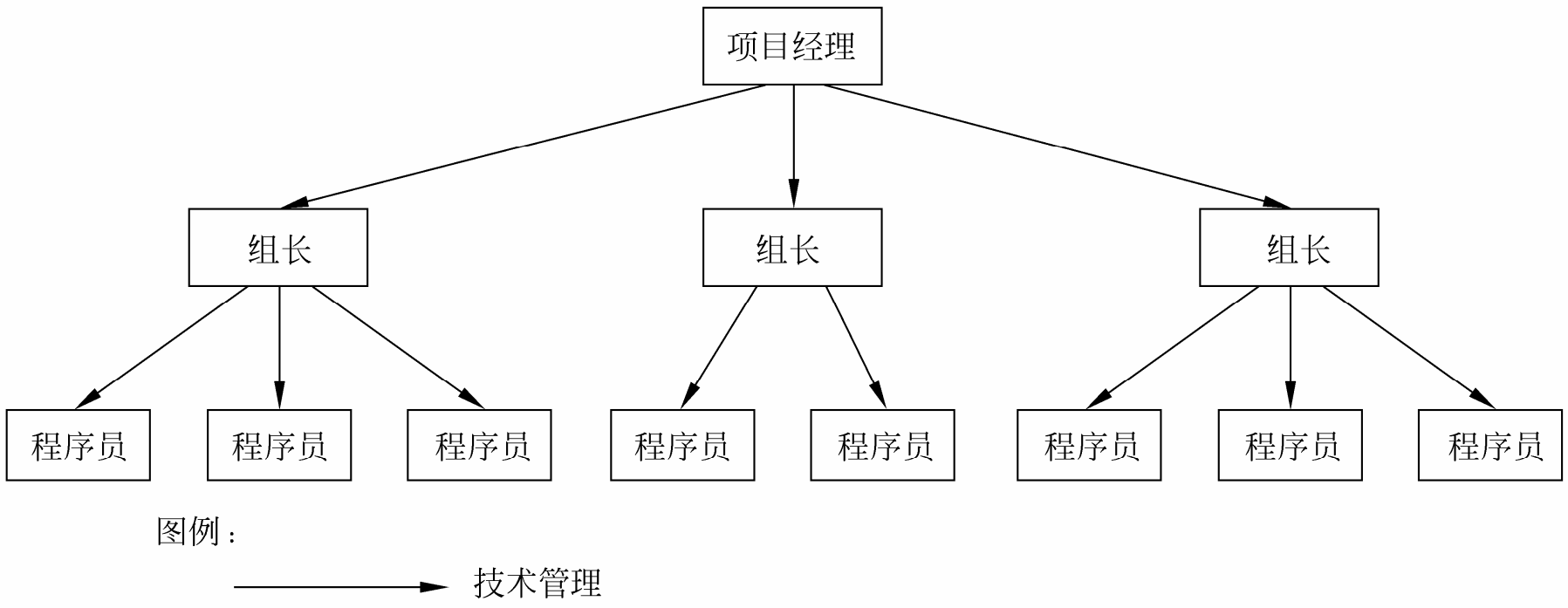
10.4.3 现代程序员组

- 现代程序员组的结构



10.4.3 现代程序员组

- 大型项目的技术管理组织结构

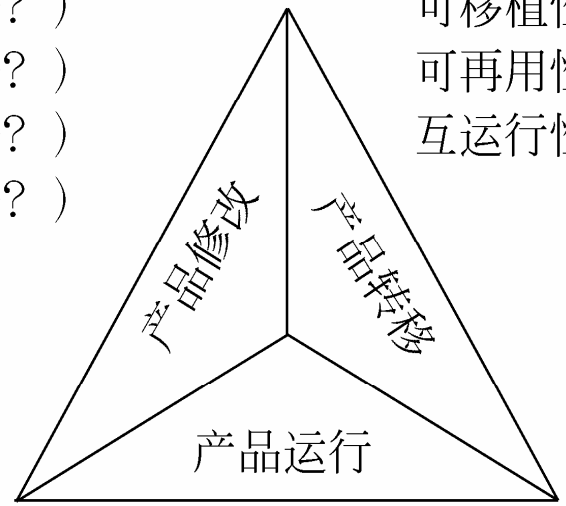


10.5 质量保证

- 软件质量定义：
  - 明确声明的功能和性能需求、明确文档化过的开发标准、以及专业人员开发的软件所应具有的所有隐含特征都得到满足
- 质量保证的目标是为管理层提供为获知产品质量信息所需的数据，从而获得产品质量是否符合预定目标的认识和信心。

10.5.1 质量因素与产品活动

可理解性（我能理解它吗？）
可维修性（我能修复它吗？）
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？）
可再用性（我能再用它的某些部分吗？）
互运行性（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（我能使用它吗？）
风险（能按预定计划完成它吗？）

10.5.2 质量保证的措施

- SQA包含
  - 一种质量管理方法
  - 有效的软件工程技术(方法和工具)
  - 在整个软件过程中采用的正式技术复审
  - 一种多层次的测试策略
  - 对软件文档及其修改的控制
  - 保证软件遵从软件开发标准的规程(在适用时)
  - 度量和报告机制。

10.5.3 软件工程标准化

- 软件工程标准的级别分类
  1. 国际标准
  2. 国家标准
  3. 行业标准
  4. 企业规范
  5. 项目规范

10.5.3 软件工程标准化

- 国际标准
  - **IS09001**：设计、开发、生产、安装和服务的质量保证模式
  - **IS09002**：生产、安装和服务的质量保证模式
  - **IS09003**：最终检验和试验的质量保证模式。

10.5.3 软件工程标准化

- 国家标准—基础标准
  - 软件工程术语 **GB/T 11457-89**
  - 信息处理-数据流程图、程序流程图……的文 件符号编制及约定**GB 1526-891(ISO 5807-85)**
  - 软件工程标准分类法**GB/T 15538-95**
  - 信息处理-程序构造及其表示法的约定 **GB 13502-92 (ISO 8631)**
  - 信息处理-单命中判定表规范 **GB/T 15535-95 (ISO 5806)**
  - 信息处理一计算机系统配置图符号及其约定 **GB/T 14085-93 (ISO 8790)**

10.5.3 软件工程标准化

- 国家标准—开发标准
  - 软件开发规范 **GB 8566-88**
  - 计算机软件单元测试 **GB/T 15532-95**
  - 软件支撑环境信息处理-按记录组处理顺序文卷的程序流程 （**ISO 6593-85**）
  - 软件维护指南 **GB/T 14079-93**

10.5.3 软件工程标准化

- 软件文档管理指南
  - 计算机软件产品开发文件编制指南

    GB 8567-88
  - 计算机软件需求说明编制指南

    GB 9585-88 （ANSI/IEEE 829）
  - 计算机软件测试文件编制指南

    GB 9386-88 （ANSI/IEEE 830）

10.6 软件配置管理

- 软件配置（Software Config）
  - 软件过程的输出信息三个主要类别：
    - 计算机程序（源代码和可执行程序）
    - 描述计算机程序的文档（针对技术开发者和用户）
    - 数据（包含在程序内部或在程序外部）
  - 这些项包含了所有在软件过程中产生的信息，总称为软件配置。
- Software Config Management
  - 其主要责任是控制变化，也负责个体SCI和软件的各种版本的标识、软件配置的审计、以及配置中所有变更的报告。

10.6 软件配置管理

- 软件配置管理任务
  - 1、配置标识
  - 2、建立系统受控配置库
  - 3、版本管理
  - 4、变更控制
  - 5、配置审核
  - 6、配置状态报告
  - 7、发布管理

10.6.1 建立系统受控配置库

- 根据配置管理的不同阶段可将受控配置库划分11个受控配置目录：
  - 初始配置
  - 启动
  - 需求分析
  - 设计
  - 编码
  - 测试
  - 安装
  - 总结
  - 变更
  - 项目管理
  - 环境配置

10.6.2 版本控制

- 版本控制结合了规程和工具以管理在软件工程过程中所创建的配置对象的不同版本。任务如下：
  - 建立项目
  - 重构任何修订版的某一项或某一文件
  - 利用加锁技术防止覆盖
  - 当增加一个修订版时要求输入变更描述
  - 提供比较任意两个修订版的使用工具
  - 采用增量存储方式
  - 提供对修订版历史和锁定状态的报告功能
  - 提供归并功能
  - 允许在任何时候重构任何版本
  - 权限的设置
  - 提供各种报告

10.6.3 变更控制

- 对于大型的软件开发项目，无控制的变化将迅速导致混乱，变更控制结合人的规程和自动化工具以提供一个变化控制的机制。
- 典型的变化控制过程如下：
  - 接到变化请求之后，首先评估该变化在技术方面的得失、可能产生的副作用、对其他配置对象和系统功能的整体影响以及估算出的修改成本。评估的结果形成“变化报告”，该报告供“变化控制审批者”审阅。所谓变化控制审批者既可以是一个人也可以由一组人组成，其对变化的状态和优先级做最终决策。

10.6.4 配置审核

- 配置审核包括两方面的内容：
  - 配置管理活动审核
    - 用于确保项目组成员的所有配置管理活动，遵循已批准的软件配置管理方针和规程，如检入(Check in)/检出(Check Out)的频度、工作产品成熟度提升原则等。
  - 基线审核
    - 实施"基线审核"，要保证作为基线的软件工作产品的完整性和一致性，并且满足其功能要求。

10.7 能力成熟度模型

Capability Maturity Model （CMM）

- 美国卡内基梅隆大学软件工程研究所在美国国防部资助下于20世纪80年代末建立，是用于评价软件机构的软件过程能力成熟度的模型。
- 它的基本思想是，因为问题是管理软件过程的方法不恰当造成的，所以采用新技术并不会自动提高软件生产率和软件质量，应该下大力气改进对软件过程的管理。事实上对软件过程的改进不可能一蹴而就，因此，CMM以增量方式逐步引入变化，它明确地定义了5个成熟度等级，一个软件开发组织可以用一系列小的改良性步骤迈入更高的成熟度等级。

10.7.1 软件组织的比较

比较项目	不成熟的软件组织	成熟的软件组织
软件过程	临时拼凑、不能贯彻	有统一标准，且切实可行，并不断改进；通过培训，全员理解，各司其职，纪律严明。
管理方式	反应式（消防式）	主动式，监控产品质量和顾客满意程度。
进度、经费估计	无实际根据，硬件限定时，常在质量上作让步。	有历史数据和客观依据，比较准确。
质量管理	问题判断无基础，难预；进度滞后时，常减少或取消评审、测试等保证质量的活动。	产品质量有保证，软件过程有纪律，有必要的支持性基础设施。

10.7.2 CMM的一些基本概念

- 软件过程：
  - 人们用于开发和维护软件及其相关过程的一系列活动，包括软件工程活动和软件管理活动
- 软件过程能力：
  - 描述(开发组织或项目组)遵循其软件过程能够实现预期结果的程度，它既可对整个软件开发组织而言，也可对一个软件项目而言。
- 软件过程性能：
  - 表示(开发组织或项目组)遵循其软件过程所得到的实际结果，软件过程性能描述的是已得到的实际结果，而软件过程能力则描述的是最可能的预期结果，它既可对整个软件开发组织而言，也可对一个特定项目而言。

10.7.2 CMM的一些基本概念

- 软件过程成熟度：
  - 一个特定软件过程被明确和有效地定义，管理测量和控制的程度。
- 软件能力成熟度等级：
  - 软件开发组织在走向成熟的途中几个具有明确定义的表示软件过程能力成熟度的平台。
- 关键过程域：
  - 每个软件能力成熟度等级包含若干个对该成熟度等级至关重要的过程域，它们的实施对达到该成熟度等级的目标起到保证作用。这些过程域就称为该成熟度等级的关键过程域，

10.7.2 CMM的一些基本概念

- 关键实践：
  - 对关键过程域的实践起关键作用的方针、规程、措施、活动以及相关基础设施的建立。
- 软件能力成熟度模型：
  - 随着软件组织定义、实施、测量、控制和改进其软件过程，软件组织的能力也伴随着这些阶段逐步前进，完成对软件组织进化阶段的描述模型。

10.7.3 能力成熟度等级

- 五级
  1. 初始级
  2. 可重复级
  3. 已定义级
  4. 已管理级
  5. 优化级

10.7.3 能力成熟度等级

- 初始级
  - 软件过程的特征是无序的，有时甚至是混乱的。几乎没有什么过程是经过定义的，项目能否成功完全取决于个人能力。管理是反应式(消防式)。
  - 关键过程域：无

10.7.3 能力成熟度等级

可重复级

- 建立了基本的项目管理过程来跟踪成本、进度、功能和质量。已经建立起必要的过程规范，对新项目的策划和管理过程是基于以前类似项目的实践经验，使得有类似应用经验的软件项目能够再次取得成功。
- 关键过程域：
  - 软件配置管理、软件质量保证、软件子合同管理、软件项目跟踪和监督、软件项目计划、需求管理

10.7.3 能力成熟度等级

已定义级

- 软件机构已经定义了完整的软件过程（过程模型），软件过程已经文档化和标准化。所有项目组都使用文档化的、经过批准的过程来开发和维护软件。
- 关键过程域：
  - 同事复审、组间协作、软件产品工程、集成的软件管理、培训计划、组织过程定义、组织过程焦点

10.7.3 能力成熟度等级

已管理级

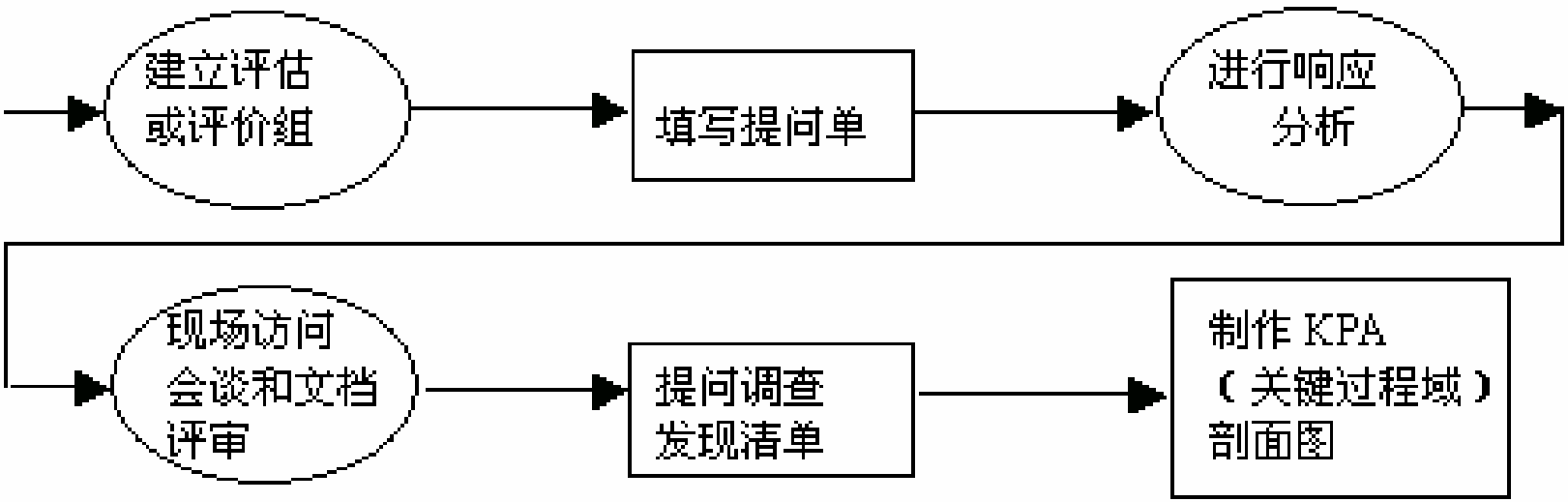
- 软件机构对软件过程（过程模型和过程实例）和软件产品都建立了定量的质量目标，所有项目的重要的过程活动都是可度量的。
- 关键过程域：
  - 软件质量管理
  - 定量的过程管理

10.7.3 能力成熟度等级

- 优化级
  - 过程的量化反馈和先进的新思想、新技术促进过程不断改进。
  - 关键过程域：
    - 过程变化管理
    - 技术变化管理
    - 错误预防

10.7.3 能力成熟度等级

- CMM的应用



10.7.3 能力成熟度等级

- CMM一共有5级，18个关键过程域（KPA），52个目标，300多个关键实践。据美国卡内基.梅隆大学SEI统计，至2001年6月底，CMM的认证情况如下表所示。

级别	通过数量	份额
5级	41家	3%
4级	54家	4%
3级	232家	17%
2级	438家	32%
1级	600家	44%

10.8 PSP简介

PSP

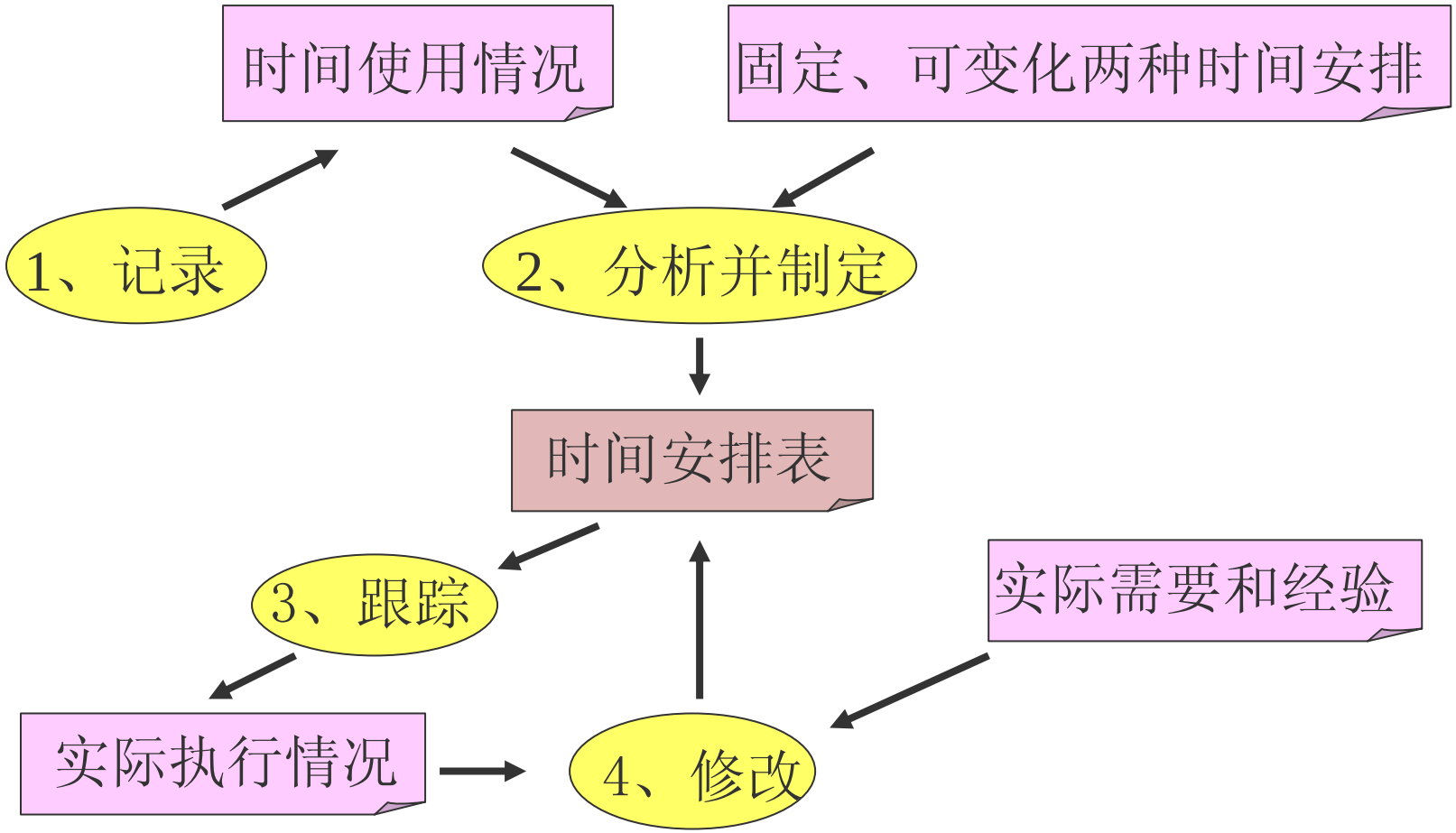
- 个体软件过程(Personal Software Process)能够指导软件工程师如何保证自己的工作质量，估计和规划自身的工作，度量和追踪个人的表现，管理自身的软件过程和产品质量。

10.8 PSP简介

- 从以下几个方面采取措施提高个体软件过程成熟度：
  - 时间管理
  - 产品计划
  - 契约
  - 进度管理
  - 质量管理

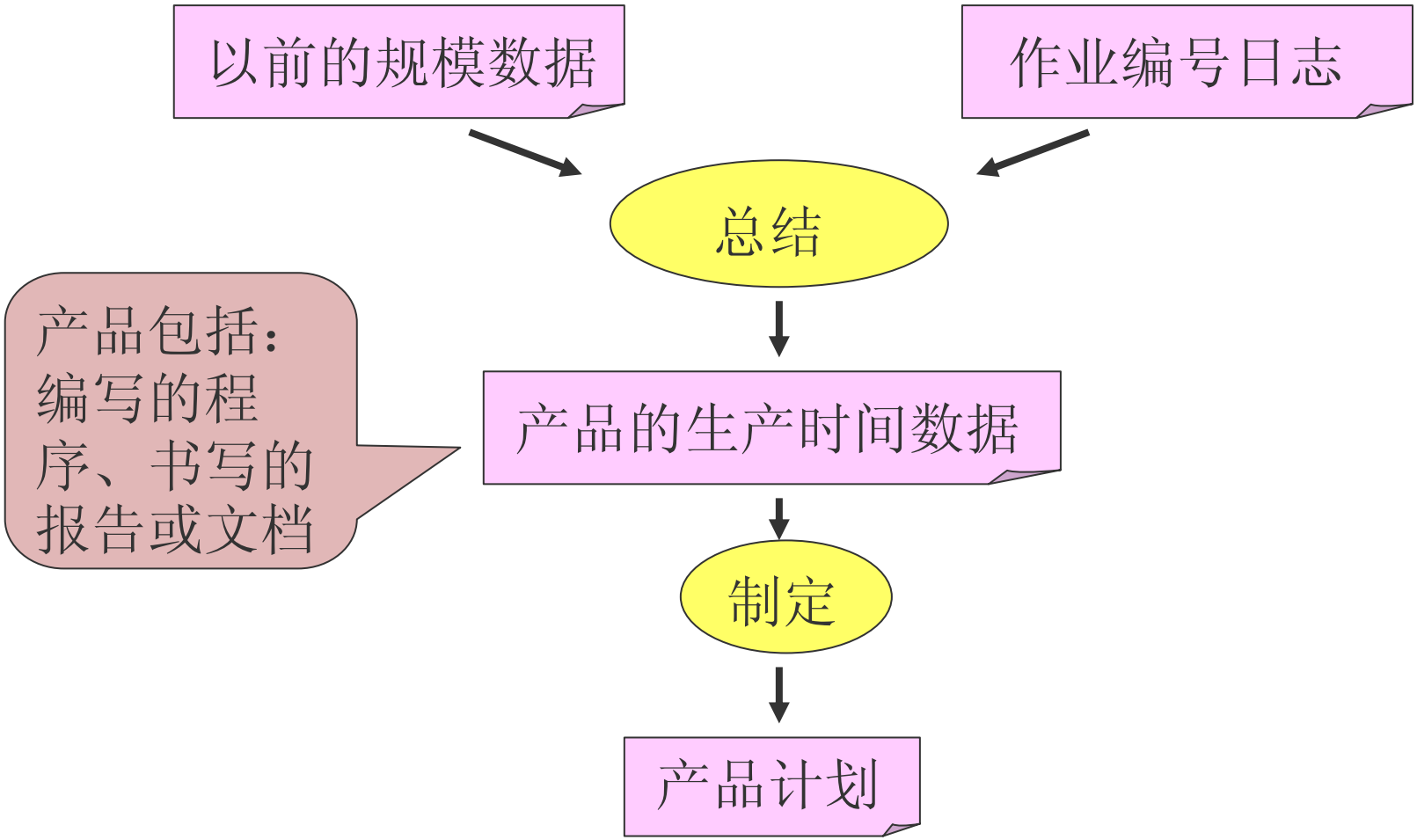
10.8 PSP简介

- 时间管理



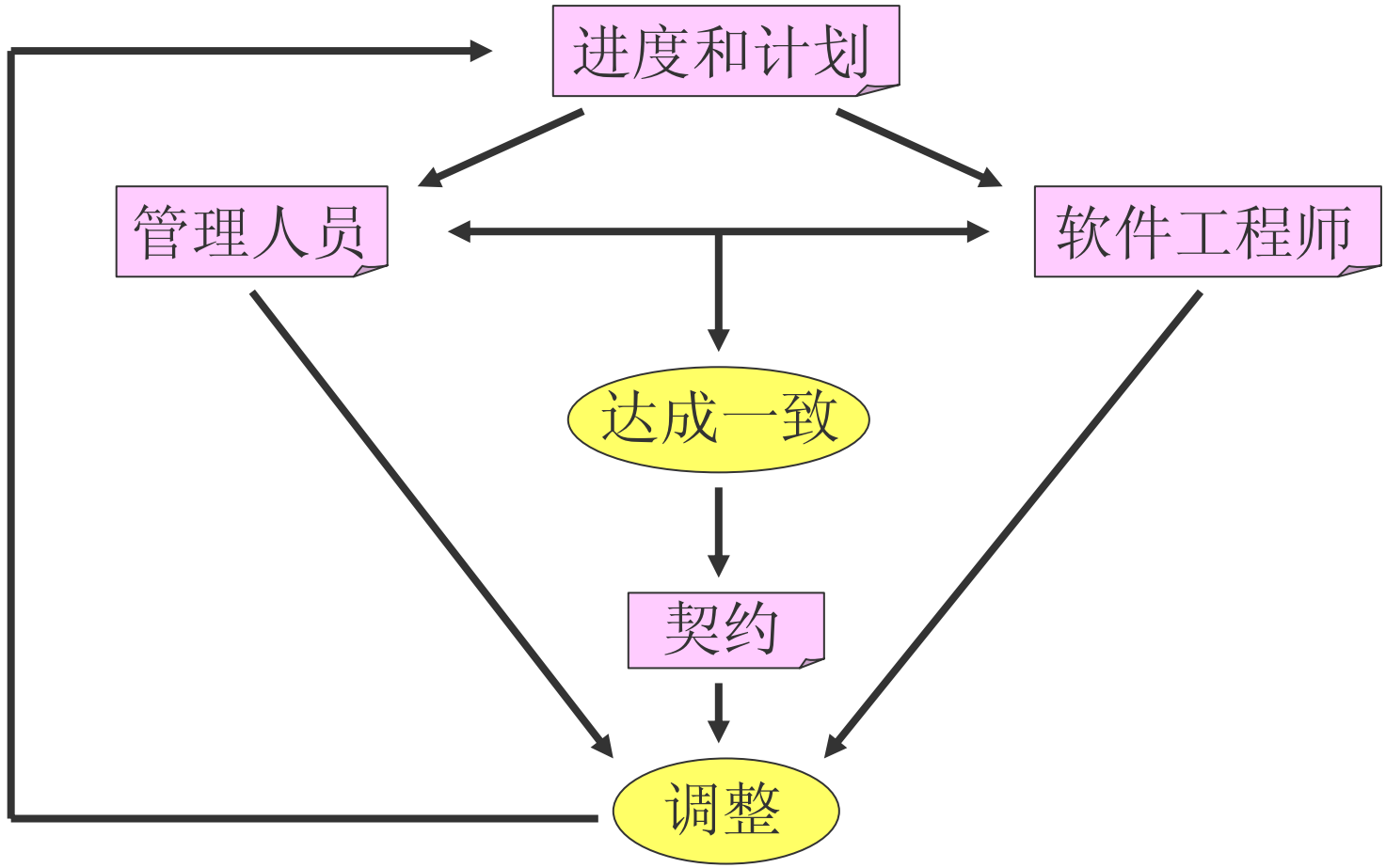
10.8 PSP简介

- 产品计划



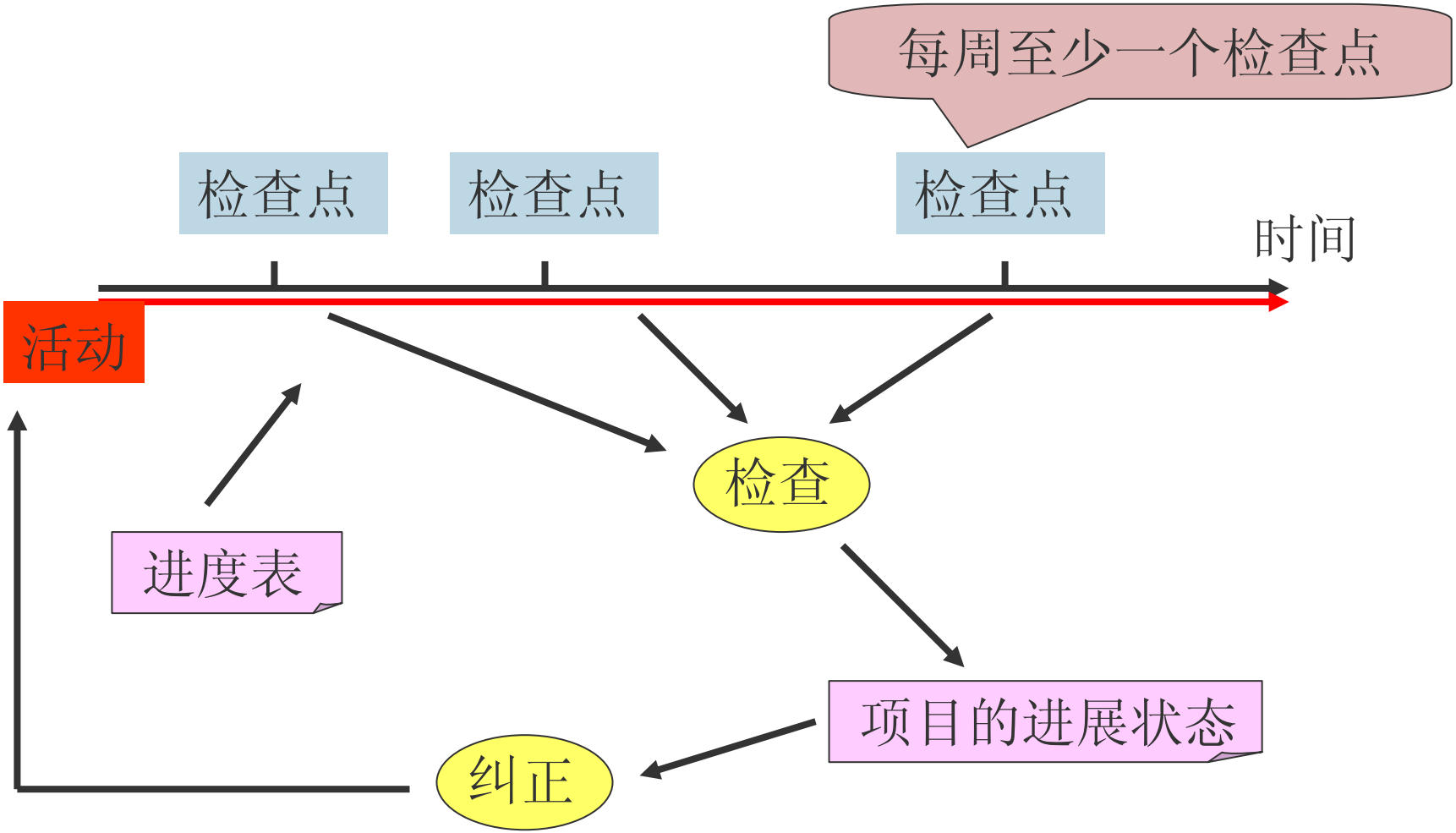
10.8 PSP简介

- 契约



10.8 PSP简介

- 进度管理



10.8 PSP简介

- 质量管理