

第三章 软件过程改进的 CMM/PSP/TSP模型

主讲：张雷

zlei@bupt.edu.cn;

北京邮电大学计算机学院

提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- 3.3 CMM组织结构
- 3.4 CMM应用
- 3.5 CMM与ISO9001
- 3.6 Microsoft公司的过程管理
- 3.7 PSP/TSP
- 3.8总结

3.1 CMM产生的背景

- Capability Maturity Model：能力成熟度模型，针对软件的称为SW-CMM。
- CMM是国际公认的对软件公司进行成熟度等级认证的重要标准。
- CMM最早的工作始于1986年11月，当时的情况：
 - ❗ 软件需求越来越大，所解决问题的复杂程度增长速度超过了人们开发和维护的能力。
 - 🔔 产品不能如期交付；质量不能令用户满意；软件开发的开销超过了预算。
 - ❓ 更好的软件开发技术也不能解决问题。
 - ✓ 软件过程的管理

3.1 CMM产生的背景

- 人们逐步认识到：软件开发中的**个人因素**并不是很重要，关键是**软件开发机构**的成熟程度。
- 卡内基 - 梅隆大学的SEI受美国国防部的委托和资助，评估软件供应商能力并帮助其改善软件质量。
- 在Mitre公司的协助下，于1987年9月发布了能力成熟度框架以及一套成熟度问卷。
- **四年后**的1991年推出CMM1.0，1993年SEI又推出了CMM1.1，适用于**500人以上**规模的软件公司。
- 近几年，SEI又推出了CMM2.0，同时进入ISO体系，称为ISO/IEC15504，即SPICE（软件过程改进与能力测定）。

提纲

- 3.1 CMM产生背景
- **3.2 CMM内容及管理透视**
- 3.3 CMM组织结构
- 3.4 CMM应用
- 3.5 CMM与ISO9001
- 3.6 Microsoft公司的过程管理
- 3.7 PSP/TSP
- 3.8总结

3.2 CMM内容及管理透视

3.2.1 CMM简介

3.2.2 CMM内容

- CMM初始级
- CMM可重复级
- CMM确定级
- CMM管理级
- CMM优化级

3.2.3 CMM各级的可视性分析

3.2.1 CMM简介

- 几个概念：
 - **软件过程能力**：描述在遵循一个软件过程后所期待结果的界限范围。
 - **软件过程效果**：描述在遵循一个软件过程后得到的实际结果。
 - **软件过程成熟度**：指一个具体的软件过程被明确地定义、管理、度量、控制和实施的程度。
 - 在软件组织内部，对这一过程章程化，并对组织成员进行培训，软件过程可以被很好地理解，并且可以持续地被它的使用者关注和修改完善。

3.2.1 CMM简介

● 不成熟组织与成熟组织之间的区别：

- 软件过程一般在项目进行中临时确定，有时确定了也不严格执行。
- 软件机构是反应型的，经常需要集中精力处理突发事件。
- 进度和经费预算估计得不切实际，进度延期导致削减软件功能，降低软件质量。
- 产品质量难以预测，质量保证活动不重视。
- 建立了机构级的软件开发和维护过程，软件人员理解后照章完成活动。
- 在成本 - 效益分析的基础上对软件过程进行改进。
- 项目速度和预算是根据以往项目取得的实践经验确定，比较符合实际情况。
- 软件产品质量由质量保证部门负责监控。

3.2.1 CMM简介

- 软件机构成熟的过程也是软件过程不断改进的过程，这个改进是一系列**微小的、不断发展的**，而**不是革命性**的创新实现的。
- 选择好的软件开发模型能够解决机构成熟问题吗？
 - No，因为：软件开发模型只是软件工程过程的组织框架，而软件过程还包括：软件管理过程和组织过程；而且软件开发模型并不强调软件过程执行的程度。
 - **软件工程过程**：常规的软件需求分析、设计、编码、测试等过程。（以软项目组为主）
 - **软件管理过程**：为使软件工程过程顺利进行而组织的管理活动集合。（以软件工程过程组为主）
 - **软件组织过程**：是机构级的软件组织活动。（以企业为主）

3.2.1 CMM简介

- 软件过程改进是**持续的**。
 - 需要设计一个过程改进路线指导软件机构。
 - 但对于软件项目来说，软件过程改进不是一个快速补救措施。
- CMM可指导软件机构在开发和维护软件时如何控制软件过程，如何改进软件工程和管理过程。方法是：
 - 指导软件机构确定现在所处的能力成熟度等级；
 - 确定提高软件质量和过程质量应该注意的关键问题；
 - 选择过程改进策略。
- 目的：CMM指导软件机构关注一套有限的活动并努力实现它们，使机构能持续地改进软件过程，从而从软件过程能力中持久地获益。

3.2.1 CMM简介

- CMM为全面地描述和分析软件过程能力的发展程度而建立了软件机构的过程成熟程度的分级标准，组织成五个成熟度级别，主要特征如下：
 - 初始级：软件过程是杂乱无章甚至混乱的，几乎没有明确定义的步骤，项目的成功依靠个人或核心人物的努力。
 - 可重复级：建立了基本的项目管理过程来跟踪成本、进度和性能。有必要的过程准则来重复以前在同类项目的成功。
 - 确定级：软件工程和管理过程已经文档化、标准化，并综合成整个软件开发组织的标准软件过程。针对具体项目裁减后用于实际的项目开发和维护中。
 - 管理级：制定了软件产品和过程质量的详细度量标准。软件过程和产品的质量都被开发组织的成员所理解和控制。
 - 优化级：根据过程执行的反馈信息以及新技术、新观念的吸纳来持续地改进和优化执行步骤，使软件过程能不断持续地改进。

3.2.1 CMM简介

- CMU-SEI已经或正在开发一系列软件过程产品：
 - 软件能力成熟度模型：SW-CMM
 - 能力成熟度模型关键实践：阐述CMM模型各个等级的关键实践。
 - 成熟度调查表：用于软件过程评价和软件能力评估，以确定一个机构的软件过程实力、弱点和风险。
- CMM虽然是目前最流行适用的软件过程标准，但理解时应该注意：
 - CMM仅指明该做什么，而没有指明如何做。不是方法论，但可以提供分析问题的方法。
 - CMM只描述需改进的关键内容，仅描述软件过程的本质属性，并非面面俱到。
 - CMM从软件过程角度考虑问题，并非关注软件开发工具，与软件开发模型无关，也与软件开发技术无关。
 - CMM为改善整个企业的软件过程提供了指南，而并非针对某个具体项目。也不保证在此框架下开发100%成功的产品。
 - 基于CMM的过程改进投资力度大、周期长。

3.2 CMM内容及管理透视

3.2.1 CMM简介

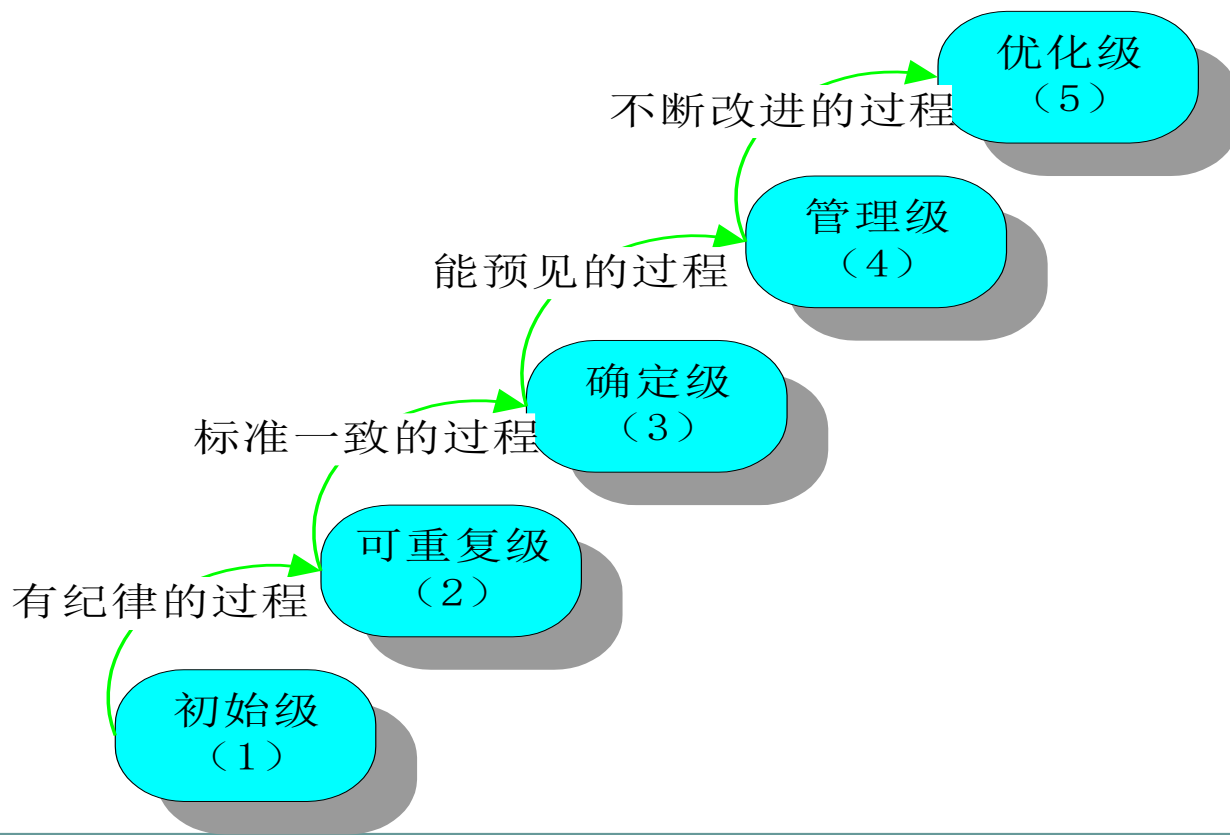
3.2.2 CMM内容

- CMM初始级
- CMM可重复级
- CMM确定级
- CMM管理级
- CMM优化级

3.2.3 CMM各级的可视性分析

4.2.2 CMM内容

- CMM为软件机构从不成熟走向成熟提供了软件过程不断改进的演进步骤的结构性框架，并组织成五个成熟度级别：



CMM初始级

● 特征：

- 软件过程的特点是杂乱无章甚至混乱，几乎没有定义过程的规则和步骤。
- 过分的承诺。实际上出现一系列危机。
- 遇到危机就放弃原计划，反复进行编码和测试。
- 成功完全依赖个人努力和杰出的专业人才，取决于他们的工程和管理经验、职业道德精神。
- 能力只是个人特性，而不是开发组织的特性。（个人一旦离开，组织也就不稳定了。）
- 软件过程是不可确定的和不可预见的。软件过程经常随意变动，软件的计划、预算、功能和质量是不确定的和不可预见的。

● 注意：

- 有些组织制定了一些软件工程规范，但如果这些规范没有覆盖基本的关键过程域（KPA），且执行没有政策、资源方面的保证时，那么该组织仍然被视为处于初始级成熟度。

CMM初始级

● 过程：

- 极少存在或使用稳定的软件过程；所谓的“过程”意味着“就这么干”；（过程无秩序）
- 各种条例、规章制度互不协调，甚至互相矛盾。（开发无规范）

● 人员：

- 依赖个人努力和杰出人物；
- 项目组成员的工作方式就是哪里出现危机就去哪儿解决。

● 技术：

- 引进新技术是极大的风险。

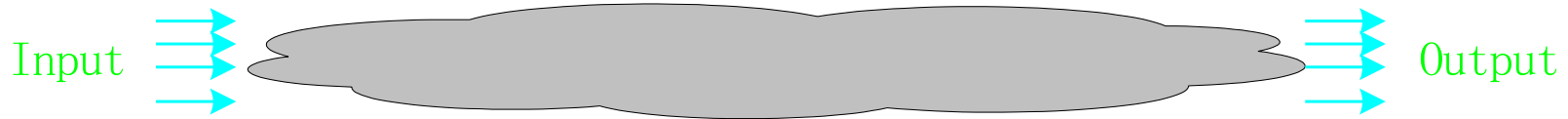
● 度量：

- 不收集和分析数据。

● 改进方向：

- 建立项目管理过程，实施规范化管理，保障项目的承诺。
- 进行需求管理，建立客户与软件项目之间的共同理解，使项目真正反映客户的要求。
- 建立各种软件项目计划。如：软件开发计划、软件质量保证计划、软件配置管理计划、软件测试计划、风险管理计划及过程改进计划。
- 开展软件质量保证活动（SQA）

CMM初始级可视性分析



● 初始级：

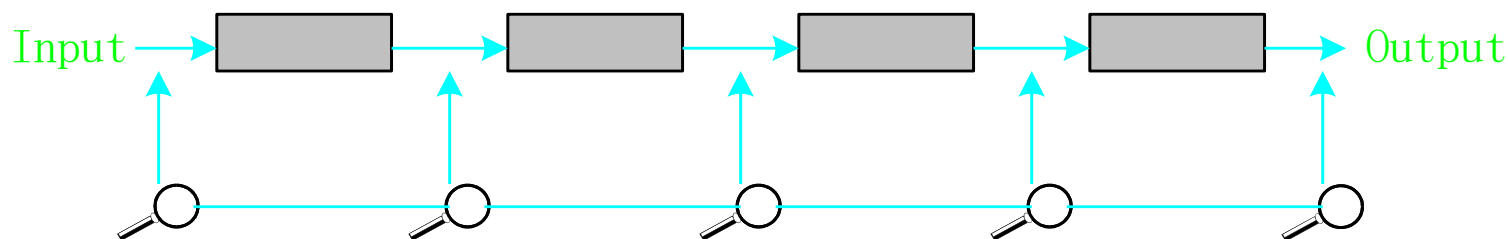
- 对于项目管理人员和用户来说：只能看到项目的要求（input）和结果（output），无法看到整个软件过程，也就无法确定开发活动的阶段，项目开发处于不可控状态。
- 由于管理者无法获得软件过程中的管理信息，因此了解某个具体项目的进度情况，或者制定项目计划都是非常困难的。
- 用户只有在软件交付时才能评估软件产品是否符合需求。

CMM可重复级

● 特征：

- 进行较为现实的承诺，按以前在同类项目上的成功经验建立必要的过程准则以确保再一次成功。
- 逐个项目地建立基本过程管理条例来加强软件过程能力。
- 建立了基本的项目管理过程来跟踪成本、进度和功能。
 - 管理过程包括：需求管理、项目管理（计划过程和跟踪监控过程）、质量管理、配置管理、子合同管理。
 - 通过执行这些过程，从管理角度可以看到一个按计划执行的且阶段可控的软件开发过程。
- 管理工作主要跟踪软件经费支出、进度和功能，识别在承诺方面出现的问题。
- 采用基线（baseline）来标志进展，控制完整性。
- 定义了软件项目的过程标准，并相信它，遵循它。
- 通过子合同建立有效的供求关系。

CMM可重复级可视性分析



● 可重复级：

- 由于采用**基线**来标志进展，控制过程完整性，因此可以按阶段来进行软件开发的控制和管理，这种管理控制发生在定义好的基线上，可使项目活动执行结果情况可视。
- 每一阶段的开发过程仍然是黑盒子，无法透视。因此，项目开发过程就象一系列的黑盒子。
- 用户需求和软件产品只在一定程度上可以控制。基本的项目管理过程一般在检查点上执行，对产品进行检查、考察过程是否正常执行、了解项目的进展等。

CMM可重复级

- **过程：**

- 软件开发和维护过程是相对稳定的，但过程建立在项目级别，而非企业级别。
- 软件工程过程受控于有效的工程管理过程，先前的成功经验可以被重复使用。
- 问题出现时，有能力识别并纠正，承诺可以兑现。

- **人员：**

- 项目成功依赖于个人能力（项目经理）和管理层的支持。
- 理解管理的必要性并对管理有承诺。
- 注意人员的培训。

- **技术：**

- 建立技术支持活动，并有稳定的计划。

- **度量：**

- 每个项目建立资源计划，主要关心成本、进度和功能，有相应的管理数据。

CMM可重复级

● 改进方向：

- 不再按项目制定软件过程，而是总结各种项目的成功经验，使之规则化，把具体经验归纳为全组织机构的标准软件过程。将改进组织机构整体软件过程能力的软件过程活动作为软件组织的责任。
- 确定全组织机构的标准软件过程，把软件工程及管理活动集成到一个稳固而确定的软件过程中。从而可以跨项目改进软件过程效果。
- 建立软件工程过程小组（SEPG）长期承担评估与调整软件过程的任务，以适应未来软件项目的要求。
- 积累数据：建立组织机构的软件过程库及软件过程相关的文档库。
- 加强培训。

CMM确定级

● 特征：

- 软件工程和管理方面的软件过程都已经文档化、标准化，并综合成软件开发组织的标准软件过程。
- 软件过程标准被应用到所有的项目开发和维护当中。有些项目可能要对这些标准软件过程进行裁减。
- 对于任何项目，其生产过程、成本、计划和功能都是可以控制的，从而软件质量也可以控制。
- 软件工程过程组（SEPG）负责软件过程活动。
- 在全组织范围内安排培训计划。

CMM确定级

- **过程：**

- 整个组织采纳标准化的软件工程和管理过程来管理所有项目的开发和维护。软件工程和管理活动是稳定和可重复的，具有连续性。
- 软件过程起了预见及防范问题的作用，能使风险的影响最小化。

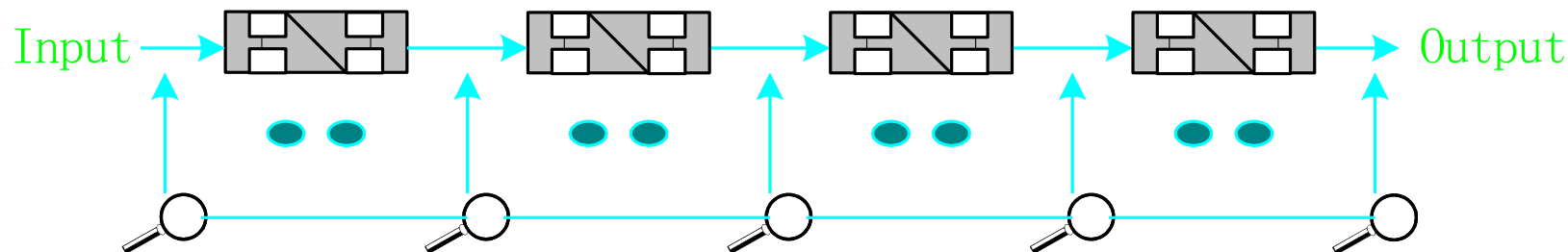
- **人员：**

- 以项目组的方式进行工作。
- 整个组织内部的所有人员对于所定义的软件过程的活动、任务有深入理解，大大增强了软件过程能力。
- 有计划地对人员角色进行培训。

- **技术：**

- 在定性基础上评估新技术。

CMM确定级可视性分析



● 确定级：

- 管理人员可以看到软件过程内部的活动以及任务。了解自己在过程中的责任和任务，在此前提下，一定程度上和各种软件开发活动进行交互。
- 上述内部结构表明软件开发组织将已有的标准软件过程用于特定项目的情况。
- 管理人员能够预见可能发生的风险，为此作一定的准备。
- 用户能够得到较为准确而快速的状况报告。

CMM确定级

- **度量：**
 - 在全过程中收集使用数据。
 - 在全项目中系统性地共享数据。
- **改进方向：**
 - 开始着手软件过程的定量分析，以达到定量控制软件项目过程的效果。
 - 通过软件的质量管理达到软件的质量目标。

CMM管理级

● 特征：

- 制定了软件过程和产品质量的详细而具体的度量标准。软件产品和过程质量都可以被理解和控制。
- 软件组织的能力是可以预见的。因为软件过程是被明确的度量标准所度量和操作的。
- 组织的度量过程保证对所有项目的生产率和质量进行度量，并作为重要的软件过程活动。
- 具有良好定义及一致的度量标准来指导软件过程，并作为评价软件过程及产品的定量基础。
- 在开发组织内已建立软件过程数据库，保存收集到的数据，可用于各项目的软件过程。

CMM管理级

- **过程：**

- 开始定量地认识软件过程。
- 软件过程的变化小，一般在可接受的范围内。
- 可以预见软件过程和产品的质量趋势，一旦度量得到的质量超出标准或有所违反，可以及时采用一些改正方法纠正。

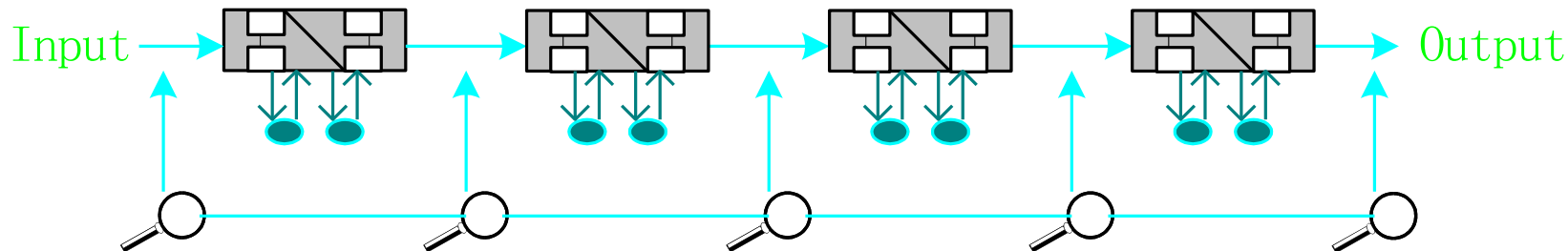
- **人员：**

- 由于每个人都了解个人的作用与组织的关系，因此能够在每个项目中产生强烈的群体意识。

- **技术：**

- 不断地在定量基础上评估新技术。

CMM管理级可视性分析



● 管理级：

- 不仅可以看到软件过程中的活动或任务，而且可以定量地控制和指导所定义的软件过程。
- 管理者可以根据客观的度量，预见过程中的经费支出和其它情况，有目的地做出决定。
- 当软件过程的不稳定性减少时，预见结果的能力增强。
- 用户可以定量地理解过程的能力及存在的风险。

CMM管理级

- **度量：**
 - 在全组织内进行数据收集与确定。
 - 度量标准化。
 - 数据用于定量地理解软件过程并稳定软件过程。
- **改进方向：**
 - 缺陷防范。不仅在发现问题时能及时改进，而且应采取特定行动防止将来出现这类缺陷。
 - 主动进行技术变动管理，标识、选择和评价新技术，使有效的新技术能在开发组织中施行。
 - 进行过程变动管理。定义过程改进的目的，经常不断地进行过程改进。

CMM优化级

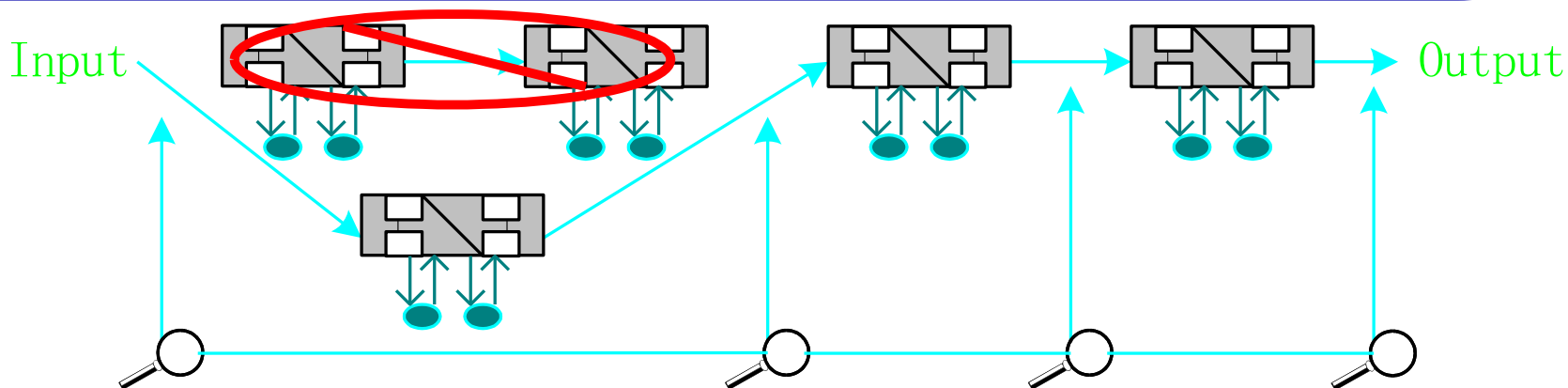
● 特征：

- 整个组织特别关注软件过程改进的持续性、预见性和自身增强性。防止缺陷及问题的发生，不断地提高组织的软件过程能力。
- 加强定量分析，通过软件过程的质量反馈和新观念、新技术的吸纳，使软件过程能力不断地得到改进。
- 根据软件过程效果进行成本/效益分析，总结成功的经验并向全组织推广，对失败的案例，由SEPG进行分析并找出原因。
- 组织能找出过程的不足并预先改进。把失败的教训告知全体组织成员以防止重复以前的错误。
- 在全组织范围内推广软件过程评价和标准软件过程改进活动。

CMM优化级

- **过程：**
 - 不断系统地改进软件过程。
 - 理解并消除产生问题的公共根源，防止浪费的发生。
- **人员：**
 - 整个组织存在自觉的强烈的团队意识。
 - 每个人都致力于过程改进。人们不再以达到里程碑的成就而满足，而要力求减少错误率。
- **技术：**
 - 基于定量的控制和管理，事先主动考虑新技术、利用新技术。可以实现软件开发中的方法和新技术的革新，以防止出现错误，不断提高产品质量和生产率。
- **度量：**
 - 利用数据来评估，选择过程改进。
- **改进方向：**
 - 保持持续不断地软件过程改进。

CMM优化级可视性分析



● 优化级：

- 可以清楚地看到软件过程。
- 从组织上有控制地、不断地、系统性地尝试新的改进方法。
- 对现存过程的认识超越了仅仅考虑过程变化的可能影响，而是自觉地识别不够有效和可能出错的活动，并加以改正和替换。工作的目的不再是达到里程碑的要求，而是减少错误率。
- 管理人员有能力估计并定量跟踪变化的影响和效果。
- 用户和开发组织合作关系良好。

3.2 CMM内容及管理透视

3.2.1 CMM简介

3.2.2 CMM内容

- CMM初始级
- CMM可重复级
- CMM确定级
- CMM管理级
- CMM优化级

3.2.3 CMM各级的可视性分析

3.2.3 CMM各级的可视性分析

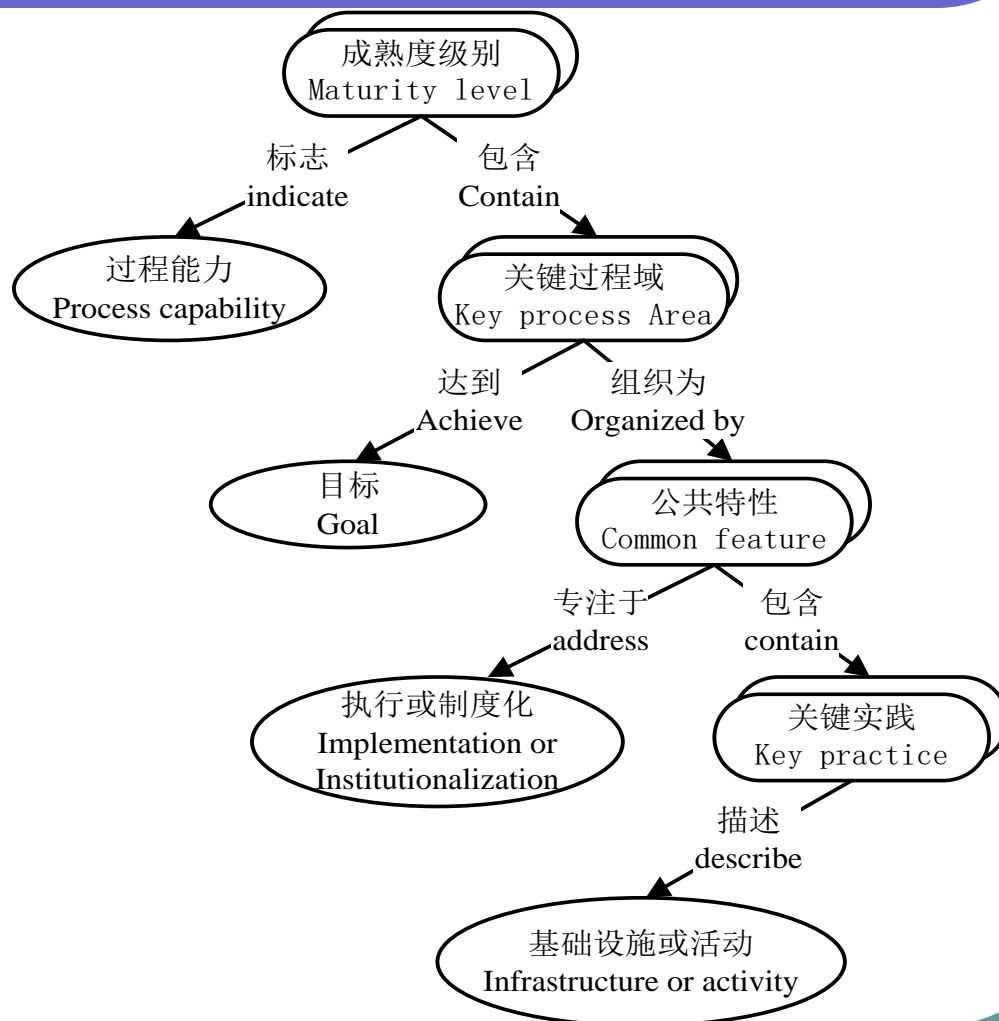
- 比较CMM中的5个等级可以看出：
 - 各成熟度等级之间的差别主要反映在**软件过程管理水平**的高低上。
 - 进一步反映在软件过程提供管理信息的能力上。
- 管理信息是管理人员和开发人员之间交流的主要手段。管理信息获取受控于软件过程的可视性，即软件过程对于管理人员的透视性。
- CMM中不同成熟度等级采纳了不同的管理方式，软件过程提供管理信息的能力也不同。
 - 每一个后继的成熟度等级的透视性都比低一些的成熟度等级的透视性更强。

提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- **3.3 CMM组织结构**
- 3.4 CMM应用
- 3.5 CMM与ISO9001
- 3.6 Microsoft公司的过程管理
- 3.7 PSP/TSP
- 3.8总结

3.3 CMM组织结构

- 为了在软件过程改进实践中体现CMM模型的可操作性，CMM给出了每一个成熟度级的详细结构规定。



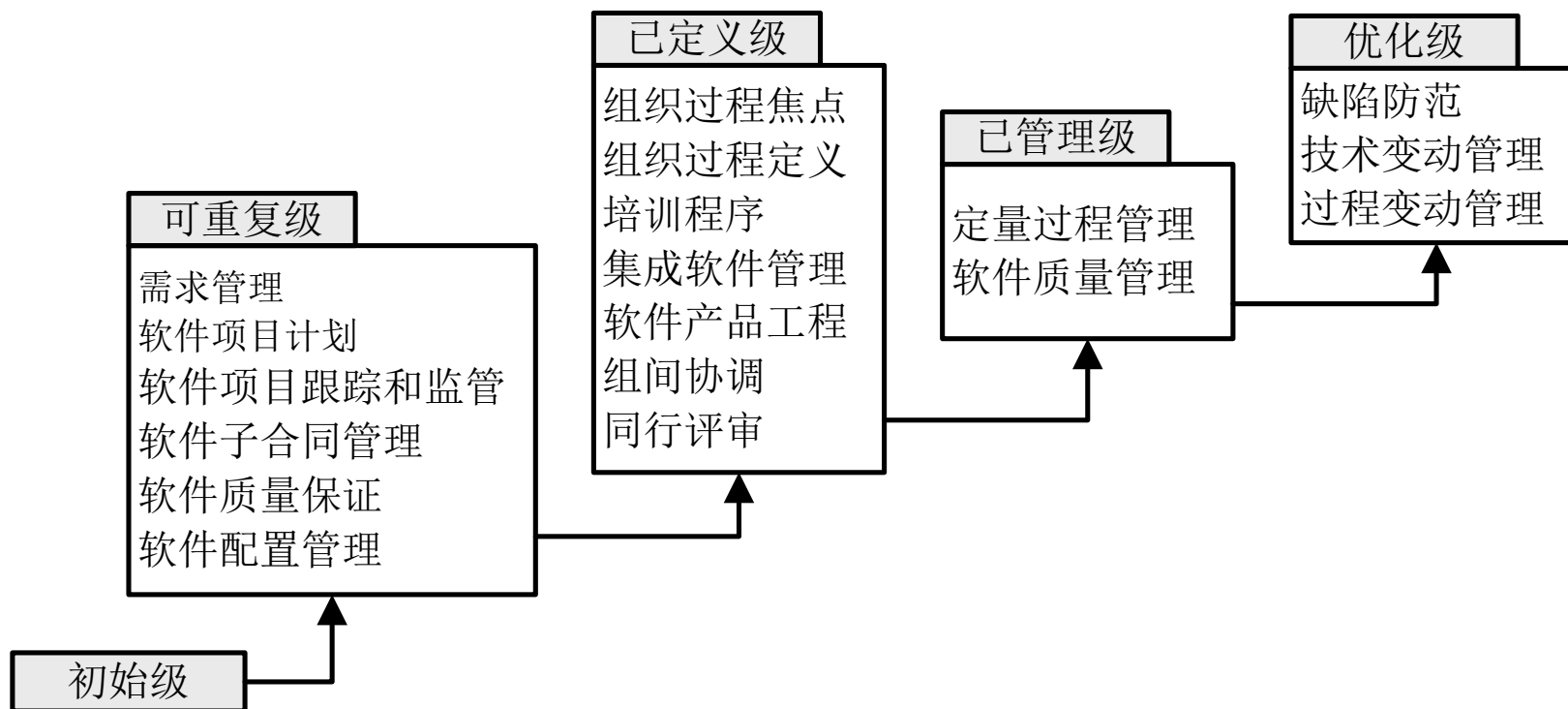
3.3.1 成熟度等级

- 五个成熟度等级构成了CMM的顶层结构：
 - 一个成熟度等级是一个妥善定义的、朝着实现成熟软件过程目标进化途中的平台。
- 每个成熟度等级显示了一定水平的过程能力。
 - 软件过程能力描述了经历某种软件过程之后，预期的结果范围。
- 每个成熟度等级由几个关键过程域（ Key Process Area , KPA ）组成。

3.3.2 关键过程域

- KPA :
 - 表示当软件组织改进软件过程时必须集中精力改进的几个方面。也就是说，KPA标识了达到某个成熟度级别时必须满足的条件。
 - 每一个KPA定义了一系列互相关联的操作活动，当这些活动得以实现后，意味着对提高软件过程能力起关键作用的目标就达到了。
- CMM并没有详尽地描述所有的过程域，只是描述一些特定的决定软件过程能力的主要方面。
- 一个组织要想达到某个成熟度级别，必须满足该等级包含的KPA的所有要求，满足每个KPA的每一个目标。
 - 目标确定了每个KPA的界限、范围和内容。可以用来判断一个组织或项目是否有效地实现了某个特定的KPA。

3.3.2 关键过程域



可重复级KPA

- 主要是为软件项目建立项目管理控制的内容。
 - 需求管理 (requirement management)
 - 目的是在顾客和软件项目组之间建立对顾客需求的共同理解，顾客需求将由软件项目组处理。
 - 与顾客的协议是策划和管理软件项目的基础。
 - 与顾客关系的控制遵循有效的更改控制过程。
 - 软件项目计划 (software project planning)
 - 为软件工程的运作、软件项目的管理提供一个合理的基础和可行的计划。
 - 软件项目跟踪和监管 (software project tracking and oversight)
 - 为软件实际运行工程建立一种透明机制，以便当软件项目进展偏离计划时能够有效地采取措施。

可重复级KPA

- 软件子合同管理 (software subcontract management)
 - 选择较好的软件分包人以及有效地管理他们。
- 软件质量保证 (software quality assurance)
 - 是提供在软件过程中对于软件项目和软件产品的恰当监督和控制。是大多数软件工程和管理过程的核心组成部分。
 - 质量保证组对软件产品、活动的评审和审计，以验证它们是否符合合适的规程和标准，并向项目负责人提供结果。
- 软件配置管理 (software configuration management)
 - 在整个软件生命周期中建立和标识软件配置项（软件工作产品及其描述等），并对其进行控制和管理，维护其完整性、一致性和可跟踪性。
 - 任务包括：
 - 制定软件配置管理计划，建立软件配置管理机构；
 - 在给定的时间点上对软件配置管理项进行标识；
 - 系统地控制软件配置的变更、存储、发行管理和交付；
 - 配置状态报告、审计；

确定级KPA

- 主要过程是关于软件项目和组织的策略，帮助软件组织确定软件项目中有效的计划和管理过程的内部细节。
 - 组织过程焦点（organization process focus）
 - 规定组织在改进其整体软件过程能力的软件过程活动中的职责，以增强软件组织的整体软件过程改进能力。
 - 组织将提供长期的承诺和资源，通过诸如SEPG来协调当前正进行和未来可能进行的软件项目的软件过程的制定和维护工作。
 - 组织过程定义（organization process definition）
 - 开发和维护一组有用的软件过程资产，这些资产能改进各项目的过程性能，它以积累的方式使组织长期受益。
 - 包括制定和维护组织的标准软件过程，以及相关的**过程资产**。
 - 软件生存周期描述；
 - 过程裁减指南和准则；
 - 组织的软件过程数据库；
 - 与软件过程有关的文档库。

确定级KPA

- 培训大纲 (training program)
 - 提高软件开发者的经验和知识，使他们更加高效率和高质量地完成自己的任务。
 - 培训组首先需要明确组织、项目、个人需要的培训，然后部署或设法获得培训以满足需求。
- 集成软件管理 (integrated software management)
 - 把软件工程和管理活动集成为一个项目应该遵循的、协调的、妥善定义的软件过程，项目开发计划根据该项目定义软件过程进行制定。
 - 该项目定义软件过程是从组织的标准软件过程和有关的过程资产经剪裁而得到的。
- 软件产品工程 (software production engineering)
 - 采用项目定义软件过程和适当的方法和工具来实施一系列软件工程任务，以便构造与维护软件产品。
 - 描述了项目中具体的技术活动，如：系统需求和软件需求分析、设计、编码和测试，以及编写和评审必需的文档。

确定级KPA

- 组间协调 (intergroup coordination)
 - 目的是为了建立一种工作方式，使软件工程组能积极参与和其他工程组间的协调，以使项目能够更有效地满足顾客的需要。
 - 包括：软件工程组和项目其他工程组一起参与阐述系统层的需求、目标和问题，必要时和用户一起参与。
 - 对组间的技术工作界面和交互活动加以计划和管理，以保证整个系统的质量和完整性。
- 同行评审 (peer reviews)
 - 目的是为了能够较早和有效地发现软件产品中存在的错误并改正它们。
 - 必然结果是对软件工作产品及可预防的缺陷得到更好的了解。
 - 包括开发同行对软件工作产品进行系统的考察，以便识别出缺陷和需作更动的地方。
 - 同行评审活动应该置于项目计划中。

管理级KPA

- 为软件过程和软件产品建立一种可以理解的定量的方式
 - 定量过程管理 (quantitative process management)
 - 定量地控制软件项目的过程性能。
 - 把可理解的测量标准加入到组织过程定义、集成软件管理、组间协调和同行评审中，并进行性能数据的采集。
 - 根据性能数据表征过程能力，软件项目根据过程能力数据建立或者修订过程性能目标。
 - 软件质量管理 (software quality management)
 - 建立一个对软件项目的产品质量的定量理解的目标及如何实现这些目标的方法。
 - 包括：确定软件产品的质量目标，制定实现这些目标的计划，并监控及调整软件计划、软件工作产品、活动和质量目标，以满足顾客和最终用户对高质量产品的需求及愿望。
 - 根据组织、客户和最终用户的需要建立软件产品的质量目标。为了实现目标，组织要制定方针和计划，项目则要适时地调整其已定义的软件过程。

优化级KPA

- 覆盖了软件组织和项目中如何实现持续不断的过程改进。
 - 缺陷防范 (defect prevention)
 - 定义错误发生的原因，并且防止它们再次发生。
 - 是一种在项目间传播和共享经验教训的良好机制。
 - 技术变动管理 (technology change management)
 - 识别、理解适用的新技术 (工具、方法和过程)，并且以有序的方式提交软件组织。
 - 组织应建立类似的技术支持组，识别、评价和选择新技术。
 - 过程变动管理 (process change management)
 - 持续不断地改进组织中应用的软件过程，以便改进软件质量，提高效率，缩短产品开发周期。

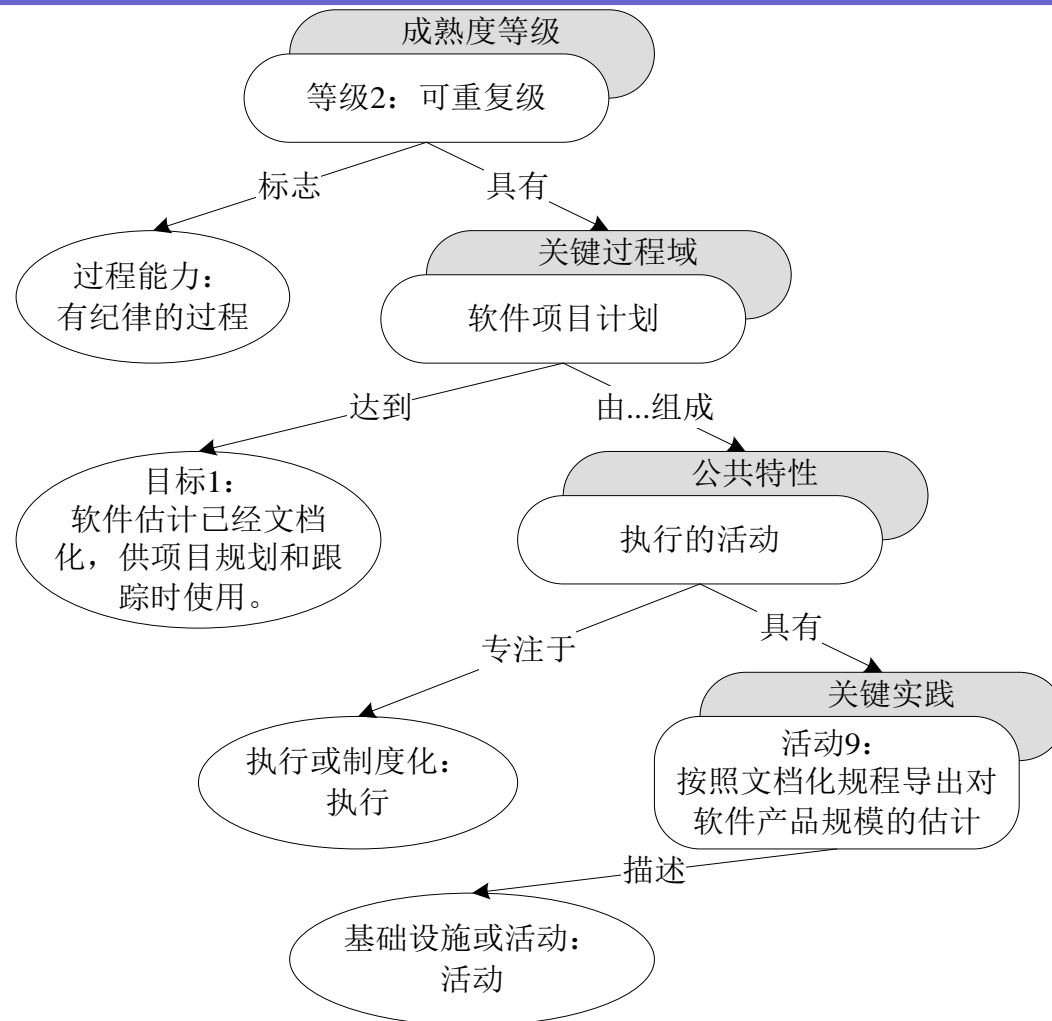
3.3.3 公共特性

- KPA经常被软件组织分为几个公共特性(common facility)，这些公共特性有效地指出了在一个KPA的实现和组织结构是否有效、可重复和持久。
 - 执行的责任 (commitment to perform)：描述组织为了建立和实施相应的KPA所必须采取的行动。这些活动主要包括制定企业范围的政策和高层管理者的责任。（**实施保证**）
 - 执行的能力 (ability to perform)：描述了在软件过程中每个项目或整个组织实施KPA“进行活动”必须达到的先决条件。一般包括资源保证、组织结构、人员培训等内容。（**实施能力**）
 - 执行的活动 (activity performed)：描述了实现一个KPA所必须的角色和执行任务的步骤（流程）。包括：制定计划、制定步骤（流程）、进行工作、跟踪工作、必要时进行的改进。
 - 度量与分析 (measurement and analysis)：描述了过程度量的基本规则，分析度量结果后确定、改进过程的状态和有效性。
 - 验证实施 (verifying implementation)：为了保证建立的软件过程活动是否遵循已制定的步骤？通过管理者和SQA部门的评审和审计进行核查。

3.3.4 关键实践

- 每个KPA最终由一些关键实践（ the key practices ）组成，通过实现这些活动来满足KPA要达到的目标。
 - 关键实践描述对KPA的**有效实施**和**规范化**贡献最大的基础设施和活动。
 - 每一个关键实践活动由一条规则说明，而且通常还有一些更详细的描述，在这些描述里可能包含例子。
 - 关键实践活动描述了应该“做什么”，并没有说明如何去达到这些目标，必须由项目小组自己解决。

3.3.4 关键实践



提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- 3.3 CMM组织结构
- **3.4 CMM应用**
- 3.5 CMM与ISO9001
- 3.6 Microsoft公司的过程管理
- 3.7 PSP/TSP
- 3.8总结

3.4 CMM应用

- CMM是**标准**：CMM建立了一个可用的标准描述，项目招标方与中标方签订合同时可以利用这些标准对风险进行评估。
- CMM是**框架**：软件组织可以利用这些标准改进组织内部的软件开发和维护过程，也就是说代表了软件改进的道路。
- CMM是**参考模型**：CMM描述了成熟软件组织的特征，标志了可期望的开发效果。

3.4.1 软件过程评估

- 目的：关注于软件组织内部的软件过程，发现缺陷，提出改进的方向。
- 方法：
 - 判断一个组织当前的软件过程的能力状态。在CMM关键实践活动的指导下发现过程的缺陷。
 - 判断并确定一个组织面对的更高等级上的与软件过程相关的改进策略。
 - 利用组织的鼎力支持来对该组织的软件过程进行有效的改进。

3.4.2 软件能力评价

- 目的：确定特定项目中的风险，包括合作者是否有能力按计划开发软件产品，以及是否能按预算完成等。
- 方法：
 - 利用CMM判断有意承担某个软件项目的软件组织（投标者）的软件过程能力。
 - 利用评价结果确定选择某一承包者的风险。
 - 判断已进行的软件过程所处的状态是否正确或是否正常。
 - 推动承包者在工作过程中改进他们的软件过程。

3.4.3 过程评估和能力评价的步骤

- **挑选队伍**：队员必须具有专业的软件工程和管理方面的知识，并接受过基本CMM概念和特定评估及评价方法的训练。
- **提出问题**：从评价和评估的角度设计成熟度提问单。
- **反应分析**：记录软件组织对成熟度提问单上的问题的反应，确定下一步更深入工作的方向。
- **现场记录**：从反应分析的结果出发，小组在KPA和关键实践活动的指导下进行提问、倾听、检查、协商等，以获取专业性的结论，说明软件过程的KPA是否达到了应有的目标。
- **CMM对照**：单方面的活动结束后，小组提供一个定义软件过程优缺点的结果清单。对于软件过程评估来说，这些结果将成为过程改进的基础和参考；对于软件能力评价来说，这些结果为决策者提供风险分析的技术基础。
- **评估大纲**：小组完成KPA基本概况的描述文件，给出组织已经满足的KPA目标和尚未满足的KPA目标。

3.4.4 方法的特点

- 考察中运用成熟度问题集作为出发点。
- CMM为考察指引方向。
- 对照KPA发现差异，定义软件过程中的优缺点。
- 从要求满足KPA的目标出发，分析满意程度，并提出书面报告。

3.4.5 软件过程评估和软件能力评价之间的不同

- 软件过程评估和软件能力评价的结果可能不同。（主要是因为评估和评价的侧重点不一样，而且被评估和被评价的组织、项目、软件产品会发生变化，因此，应该考虑评估和评价的Context。）
- 软件过程评估和软件能力评价在出发点、目标、结果和结果的作用等方面都是不同的。（导致成熟度提问单的内容组织不一样，收集的信息不一样，结论的评价不一样。）
- **软件过程评估是在一个开放的、互相协作的环境下进行的。而软件能力评价往往是在有较大阻力的环境中进行的。**
（过程评估是为了提高管理者和工程师的工作水平，而能力评价是为了表明一个软件组织的实际软件过程能力，为选择承包者和减少费用服务。）

3.4.6 CMM在SPI其它方面的应用

- 过程改进活动计划的定义。针对目标成熟度等级的过程要求制定过程改进计划。
- 过程改进计划的执行。执行过程中始终判断当前软件过程与目标成熟度等级要求的KPA之间的差距，根据组织的管理方式、实际操作水平、投入的资金、企业本身实现软件过程活动的的能力等确定这种差距能否改善，并确定进一步改进的方向。
- 规范化过程改进过程。根据过程改进方向帮助软件组织将“改进要求”变为“可操作的步骤”。

3.4.7应用CMM时注意的问题

- CMM模型的问题：
 - CMM指明该做什么，但没有指明如何做，它不是方法论，没有给出特定应用领域内的专门技术。
 - CMM是一个用于改进软件产品和管理过程的结构化模型，但是仅描述软件过程的本质属性，并非涉及软件工程的所有问题。
 - CMM是从软件过程角度定义了成熟的软件过程的实践活动，但是对于成熟的软件组织而言，人的因素和技术的因素也同样重要。

问题I：过程成熟需要多长时间？多少费用？ 对企业有何好处？

- 一般需要2年才能把成熟度提升一级（建议安排1.5年到2年）。
- 每年每个软件工程师所需费用为490美元到2004美元，平均1375美元（包括评估费、CMM培训费、SEPG的工资及经费等）。
- 基于CMM的过程改进是有一定成效的，如：
 - 生产率按照35%的平均年递增率提高；
 - 上市时间提早；
 - 软件发布后的缺陷每年减少39%；
 - 收益比平均在5.0:1。

问题2：影响软件过程改进的成败因素是什么？

- 软件过程改进必须有高级主管的支持与委托（**授权**），并积极地管理过程改进的进展。
- 获取中层管理的支持，以方便地获取过程改进的资源（人员、时间、经费和设备）。
- 基层技术人员的参与和支持极端重要。
- 利用定量的可观察数据尽快使过程改进的成果可见，从而激励参与者的兴趣。
- 按照软件过程改进对企业文化的要求进行变革，要求软件过程改进为商业利益服务，并与企业其他部分协调。

问题3：CMM是否会导致官僚主义？

是否会使组织变得更保守，不愿意冒风险？

- 84%~96%的被调查者不认为CMM会使组织变成僵化的官僚主义机构，难于创新。
 - 一级组织仅有42%愿意承担中等风险；
 - 二级组织有74%愿意承担中等风险；
 - 三级组织有79%愿意承担中等风险。

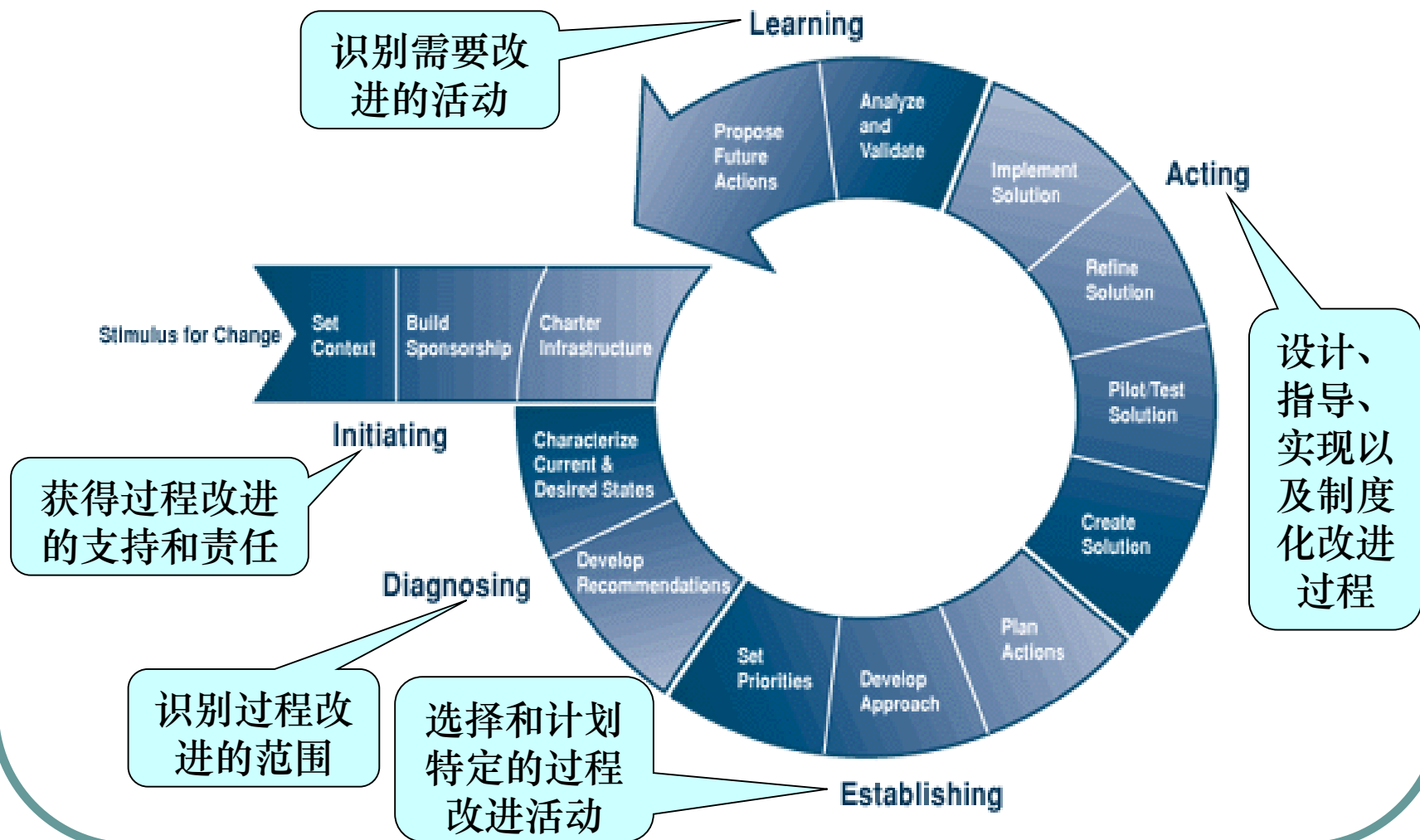
问题4：有无合适的、易理解的框架（不仅仅是做什么，而且告诉如何做）可指导所有软件组织进行CMM改进？

- 不断总结经验，提出办法，没有万能灵药。
- 最大危险不是缺乏技术，而是错误地使用技术。

3.4.8 IDEAL过程改进生命周期

- CMU-SEI与HP公司合作，提出了实施SW-CMM的IDEAL模型。
- IDEAL模型是一个单位用于启动、规划和实现过程改善措施蓝图的模式，概括了建立一个成功的过程改善项目的必要步骤。
 - 其中I:Initiating（启动）、D:Diagnosing（诊断）、E:Establishing（构建）、A:Acting（执行）、L:Learning（学习）。

3.4.8 IDEAL过程改进生命周期



提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- 3.3 CMM组织结构
- 3.4 CMM应用
- **3.5 CMM与ISO9001**
- 3.6 Microsoft公司的过程管理
- 3.7 PSP/TSP
- 3.8总结

3.5 CMM与ISO9000

- ISO9000是一族国际标准，其中ISO9001是针对服务行业关于产品质量保证的标准，ISO9000-3是关于软件产品质量管理的指南。
- ISO9001的20条款内容：管理职责；质量体系；合同评审；设计控制；文件和资料控制；采购；顾客提供产品的控制；产品标识和可追溯性；过程控制；检验、测量和试验设备的控制；检验和试验状态；不合格品的控制；纠正和预防措施；搬运、储存、包装、防护和交付；质量记录的控制；内部质量审核；培训；服务和统计技术。
- ISO9001是面向所有工业的，主要针对传统制造业，标准中缺乏对软件行业的足够指导。
- ISO9001与CMM是强相关的，ISO9001不覆盖CMM，CMM也不完全覆盖ISO9001。一般而言，通过ISO 9001认证的企业可达到CMM 2级或略高的程度，通过CMM 3级的企业只要稍做补充，就可较容易地通过ISO 9001认证。粗略地说，ISO 9001近似于CMM "2.5级"。ISO 9001约有80%的文件可以用于CMM 2级评估。

3.5 CMM与ISO9000

- **共同点**：强调了软件产品的质量。
- **不同点**：
 - CMM是专门针对软件工业的，而ISO9001则面向所有工业。因此，相对而言，CMM更具体些，ISO9001更抽象些。
 - CMM是面向内部的软件过程改善框架，而ISO9001是供需合同关系下基于过程的质量需求，强调可接受的质量体系的最低标准，没有告诉软件开发人员如何达到好的目标，如何避免差错。
 - CMM通过KPA中的关键实践活动的执行程度判断软件过程的能力成熟性；ISO9001针对合同环境下设计、开发、生产、服务等环节给出了所需要的最基本的质量要素，通过这些要素实施的有效程度判断企业是否符合要求。
 - CMM的结构是层次化的结构，由等级、KPA、公共属性、关键实践活动组成；ISO9001是简单的线性结构，包含20个质量要素。
 - 在应用概念上，CMM强调企业内部素质，而ISO9001重在整体。实施CMM的最大益处是可以较大程度避免形式主义。

提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- 3.3 CMM组织结构
- 3.4 CMM应用
- 3.5 CMM与ISO9001
- **3.6 Microsoft公司的过程管理**
- 3.7 PSP/TSP
- 3.8总结

提纲

- 4.1 CMM产生背景
- 4.2 CMM内容及管理透视
- 4.3 CMM组织结构
- 4.4 CMM应用
- 4.5 CMM与ISO9001
- **4.6 Microsoft公司的过程管理**
- 4.7 PSP/TSP
- 4.8总结

3.6 Microsoft公司的过程管理

- **微软的哲理：追求高度灵活。** 企图将松散形式的小组风格提升为正规的产品开发。
- **微软的目标：既有自由，又有严格性。**
 - 使许多小的、平行的小组或单个程序员能一起合作工作，成为一个单一的、相当大的组织，并以相对快的速度开发大型产品。
 - 小型平行小组自主地发展产品的功能特性，可以自由地进行创新。
 - 采用“同步稳定”方法保持各小组的同步，使产品部件可以共同工作。

3.6 Microsoft公司的过程管理

- 微软的策略：在产品定义与开发过程中，靠改进特性和固定资源来激发创造力。其原则是：
 - 将大项目分成多个里程碑式的重要阶段（计划阶段、开发阶段和稳定阶段），各阶段之间有缓冲时间（20%~50%的项目时间，处理可能发生的变动、尚未考虑到的困难及延期问题），但没有单独的维护阶段。
 - 利用“预想陈述”和“特性规格概述”来指导项目。而不采用在编码之前进行详细设计，不强调对产品进行全面规格说明。（这些工作事先极难确定，交给开发组成员去进行创新，去适应变化、尚未预见的竞争机会或危险。）
 - 根据用户的活动情况及获得的数据确定产品的特性及其优先顺序。（要求产品反映用户的需求，要求开发人员观察用户的反应以及活动情况。）
 - 建立模块化和水平式的设计结构，并使项目人事组织结构反映产品结构的特点。
 - 靠个人负责和固定项目资源实施控制。（管理人员让开发人员详细地分析任务并自己制定进度。管理人员负责分配资源。）

3.6.1 同步—稳定开发方法

● 计划阶段

- 提出产品的预想陈述（来自于产品经理和程序经理）
 - 新产品的市场营销分析：竞争对手的产品分析和未来版本的规划；
 - 对前一版本的改进。
- 确定产品的功能规格（定义产品的特性及其优先级），即描述产品做什么和怎么做。
- 确定进度及特性组（由程序经理协调进度和安排特性组）。

● 开发阶段

- 分为三个里程碑：
 - 头1/3关键特性及共享部件的开发；
 - 第二个1/3特性；
 - 最后不关键的1/3特性的开发。
- 规格的演进是由程序经理控制的，项目40%点需锁定需求，最后一个里程碑后锁定界面。

● 稳定阶段

- 综合性地进行内部测试和外部测试，稳定并发布最终产品。
- 本阶段不再增加功能，除非竞争对手产品发生变化。

3.6.1 同步—稳定开发方法

- 同步-稳定开发方法的目标：
 - 致力于高度竞争，研制快速变化的企业环境所追求的、要求创新精神的、高度复杂的产品。
 - 用户要求难于确定、软件技术变化快、产品部件难于精确定义。→企图事先设计出完整的软件系统是不明智的。
 - 给予开发人员足够的灵活性来创造及发展产品的具体细节；
 - 让开发人员和客户一起测试产品，并在开发时就可以改进设计。

3.6.1 同步—稳定开发方法

- 同步-稳定开发方法的原则：
 - 频繁地同步，周期性地稳定。
 - 不断同步特性小组成员正在开展的工作；
 - 周期性地稳定不断扩大中的产品。
如：里程碑、日建立、零缺陷等。
 - “日建立”步骤：
 - 为了开发某特性从源码中心主版本中得到源码文件的私有版本，进行修改或增加特性后checkin到源码中心主版本中，如有问题必须立即修正。
 - 指定开发人员每天用源码主版本文件产生产品。用build脚本来构造、产生各平台内部版本，并自动把源码变成多个可执行文件、各种库文件，使用户可以裁减产品。

3.6.2 微软开发和交付产品的策略

- 平行开发，并且经常同步所做的每一件事情。
- 永远保持至少存在一个可交付的产品。
- 在单个开发站点上只用一个公共的开发语言进行交流。
- 在构造产品时不断地测试。
- 利用定量的度量数据，指导下一步工作。

提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- 3.3 CMM组织结构
- 3.4 CMM应用
- 3.5 CMM与ISO9001
- 3.6 Microsoft公司的过程管理
- **3.7 PSP/TSP**
- 3.8总结

3.7 PSP/TSP

- **PSP/TSP的引入**
- PSP
- TSP

3.7.1 PSP/TSP的引入

- CMM的成功与否与组织内部的相关人员的积极参与和创造性活动密不可分，但CMM并未提供KPA关键实践活动实施者所需要具备的具体知识和技能。怎样将CMM应用到具体的工作环境中去呢？
- 这时候，PSP应运而生。
 - PSP (Personal Software Process) 为基于个体和小型群组软件过程的改进与优化提供了具体而有效的途径，例如：如何制订计划、如何控制质量、如何与其他人相互协作等。
 - 在设计阶段，PSP的着眼点在于**软件缺陷的预防**，**具体办法是强化设计结束准则**，而不是设计方法的选择。
 - PSP的研究结果表明：绝大多数软件缺陷是由于对问题的错误理解或简单的失误造成的，只有很少一部分是由于技术问题而产生的。如果在设计阶段引入一个错误，则这个错误在编码阶段会引发3-5个错误，修复费用要高一个数量级。因此，PSP保障软件产品质量的一个重要途径是提高设计质量。

3.7.1 PSP/TSP的引入

- 实践证明，仅有PSP是不够的。CMU-SEI CMM在PSP基础上发展出了TSP（Team Software Process）的方法。
 - TSP指导项目组中的成员如何有效规划和管理所面临的项目开发任务，并且告诉管理人员如何指导软件开发队伍始终以最佳状态来完成工作。
 - TSP实施集体管理与自己管理自己相结合的原则，最终目的在于指导开发人员如何在最少的时间内，以预定的费用生产出高质量的软件产品，所采用的方法是对小组开发过程的定义、度量和改进。
- 小组远不只是一群有才能的个人的集合，为了建立并保持高效率的工作关系，小组需要共同的目标，大家一致同意的行动计划和适当的领导，还需要在需要的时候乐于寻求帮助。

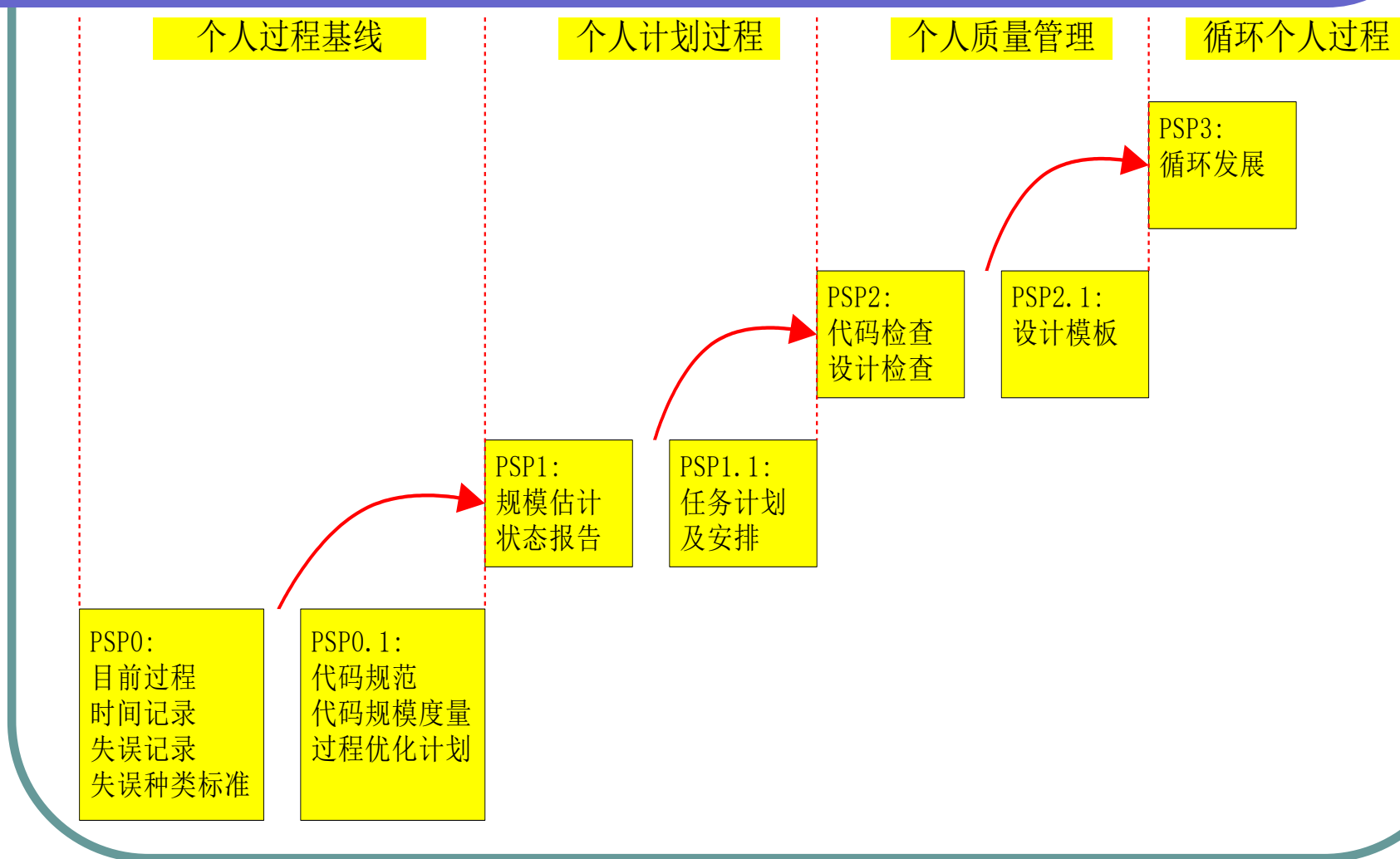
3.7 PSP/TSP

- PSP/TSP的引入
- **PSP**
- TSP

3.7.2 PSP

- PSP是一个软件过程的描述、测量和改进方法的结构化集合，它可以为软件工程师带来更少的错误代码、更好的预算和计划以及更高的生产率，从而能够帮助软件工程师改善其个人性能。
- PSP提供了帮助软件工程师开发软件的表格、脚本和标准，以估算和计划软件工程师的工作，以便软件工程师可以更加清楚自己的个人技术并且提升个人表现。PSP显示了如何定义过程及如何测量其质量和生产率。
- PSP不依赖于任何技术（语言、工具和设计方法），它：
 - 示范了软件过程原则；
 - 帮助工程师做正确的计划；
 - 告诉工程师怎样提高软件质量；
 - 建立个人软件过程提升的度量标准；
 - 决定了过程改进在工程师表现中的影响。
- PSP用一系列的步骤解释个人软件过程的改进，每一步包含前一步所有元素并且有所增加。

3.7.2 PSP



PSP0（个人过程基线）

- **PSP0是过程基线，目的是为了在个人的工作中引入表格和脚本，以便记录软件过程。**
 - **PSP0-1.目前过程**：记录在工程中使用的具有代表性的软件开发方法和在当前工程中使用的方法。
 - **PSP0-2.时间记录**：记录在不同的软件开发阶段（计划；设计、编码、编译和测试；维护）所花费的时间。
 - **PSP0-3.失误记录**：按照一致的格式记录软件工程师引入软件中的缺陷，并记录软件工程师尝试解决问题的方法和步骤。
 - **PSP0-4.失误种类标准**：
 - 一方面为软件工程师提供在系统中可观察到的典型缺陷标准列表；
 - 另一方面提供一种预定义步骤工具方便软件工程师对新的缺陷进行归类和记录，这种工具应该采纳不同的描述风格，如：声明式、程序式、非正式的等，并有合适的GUI。

时间记录日志及统计

日期	开始时间	结束时间	中断时间	净时间	活动	用例	状态	备注
2005.3. 9	8:30	10:30	20	100	分析	出库	完成	
	10:30	12:00	30	60	评审	出库	完成	
	13:00	17:30	40	230	分析	入库	完成	
2005.3. 10								

备注：当活动变换时需另起一行，以便对单个活动进行时间统计。

统计	规定工作时间	实际工作时间	中断累计时间	有效时间利用率
日统计	480	510	90	82.4%
周统计				
月统计				

活动总结表

活动名称	用例	计划时间	实际时间	活动状态	误差百分比	原因分析
分析	出库	80	100	完成	-25%	
评审	出库	60	60	完成	0	

备注：可以周或阶段为单位

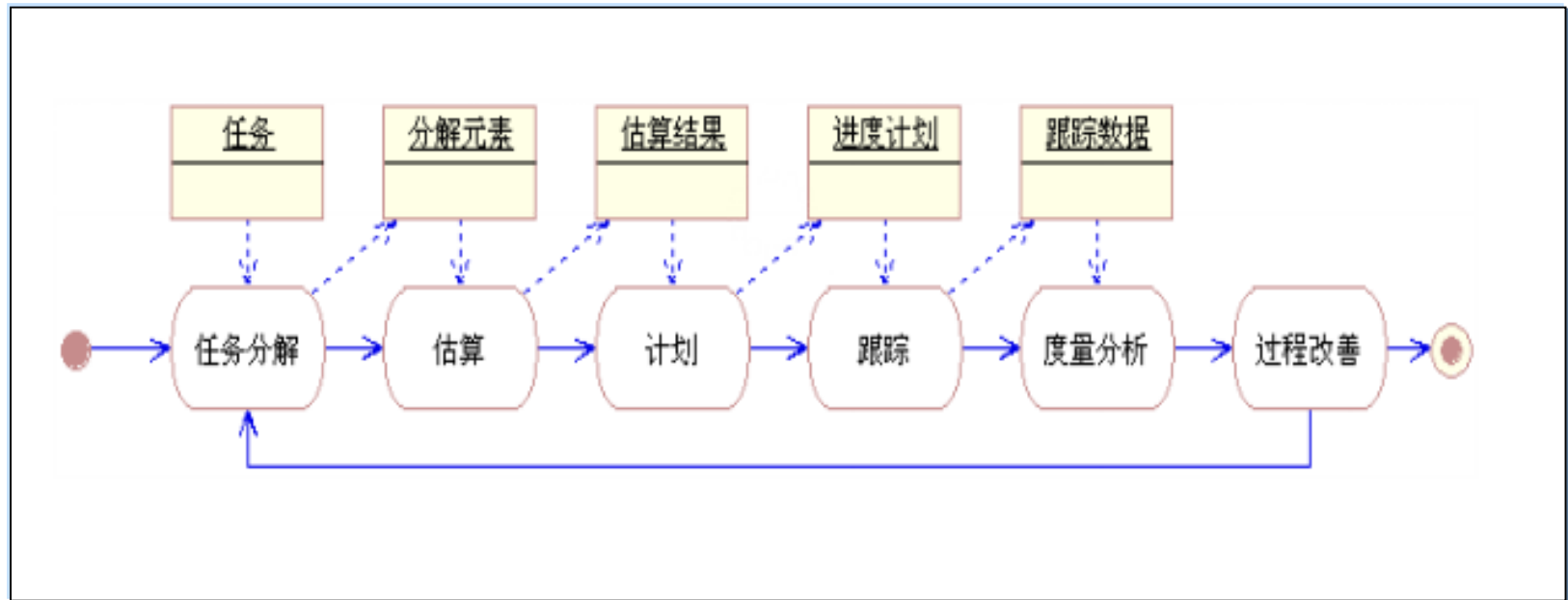
PSP0.1（个人过程基线）

- PSP0可以通过增加下列过程方便地扩展到PSP0.1。
 - PSP0.1-1.**代码规范**：通过对设计过程、开发过程和设计语言结构进行规范，约束具有不同技术背景和软件开发风格的软件工程师。**由组织统一制订设计方法学和编码标准。**
 - PSP0.1-2.**代码规模度量**：测量代码的尺寸，如：长度、功能、复杂度、再利用数、冗余数等。一般基于某种测量标准进行，如LOC，软件工程师应该了解LOC及相关测量概念。
 - PSP0.1-3.**过程优化计划**：针对已经记录的软件过程中的问题和经验教训，帮助软件工程师给出软件过程能力的提高建议，并以结构化的方式表达软件过程、问题、建议教训、提高建议等项目。

PSP1（个人计划过程）

- PSP1就是个人计划过程，它在PSP0的基础上增加了计划步骤。
 - PSP1-1.**规模估计**：分为代码尺寸估算、时间估算、资源估算。
 - 代码尺寸估算：软件工程师可以凭借PSP0级代码尺寸测量经验预测他们将要写的任务模块或算法的可能尺寸。
 - 时间估算：PSP0级时间测量过程可以总结出不同复杂度模块的编写时间，凭借这些经验，软件工程师可以针对当前系统的模块结构层次给出每个模块的估算时间（乐观时间、可能时间、悲观时间）。
 - 资源估算：对于软件开发的一段生存期，软件工程师预测所需要的软件、硬件和人力资源，其中人力资源预测包括人力需求、人力成本估算和项目管理标准。
 - PSP1-2.**状态报告**：对软件工程师的工作进行跟踪，测试规模估计与实际状态之间的差异。

PSP1的基本流程



PSP1.1（个人计划过程）




- PSP1.1在PSP1的基础上引入了任务计划和安排过程。
 - PSP1.1-1.**任务计划及安排**：基于PSP1中的规模估计数据制订软件项目的需要完成的任务计划，并将任务按时间段分配给不同的人力资源。
 - 一般采纳网络安排技术，如PERT(Program Evaluation and Review Techniques)和CPM(Critical Path Method)，软件工程师应该理解网络安排技术和计划策略。

注：关于计划执行的跟踪应该在PSP1的状态报告过程中完成。

任务分解与估算

用例	任务	活动	操作	复杂度	时间估算	备份时间25%	小计	实际数据
用例	任务1	活动1	操作1	中	150	50	200	
		活动2	操作2	高	300	100	400	
			操作3	低	120	40	160	
			操作4	中	240	80	320	
		活动3	操作5	中	150	50	200	
	小计	3	5		960	320	1280	
	任务2							

注：以编码阶段为例
数字单位：分钟

任务	活动	3. 14 (480)	3. 15 (960)	3. 16 (1440)	3. 17 (1920)
任务1	活动1				
	活动1				
	活动1				

PSP2（个人质量管理）

- PSP2是个人质量管理过程，介绍了失误管理。PSP2强调设计的结束准则。
 - PSP2-1.**代码检查**：Wirth在软件过程评估调查表的代码评估章节中给出了：代码检查标准、应用、覆盖、数据搜集、表示法、分析和效率。
 - PSP2-2.**设计检查**：目的是确定设计完成标准，包括检查设计和设计的一致性。
 - 检查设计要求提供一些评估设计质量的一些指标，如：
 - 代码重用率、代码冗余、代码完整性和协作性。
 - 设计的一致性检查主要涉及：结构化（控制和数据）一致性、耦合一致性、可移植和互用一致性。

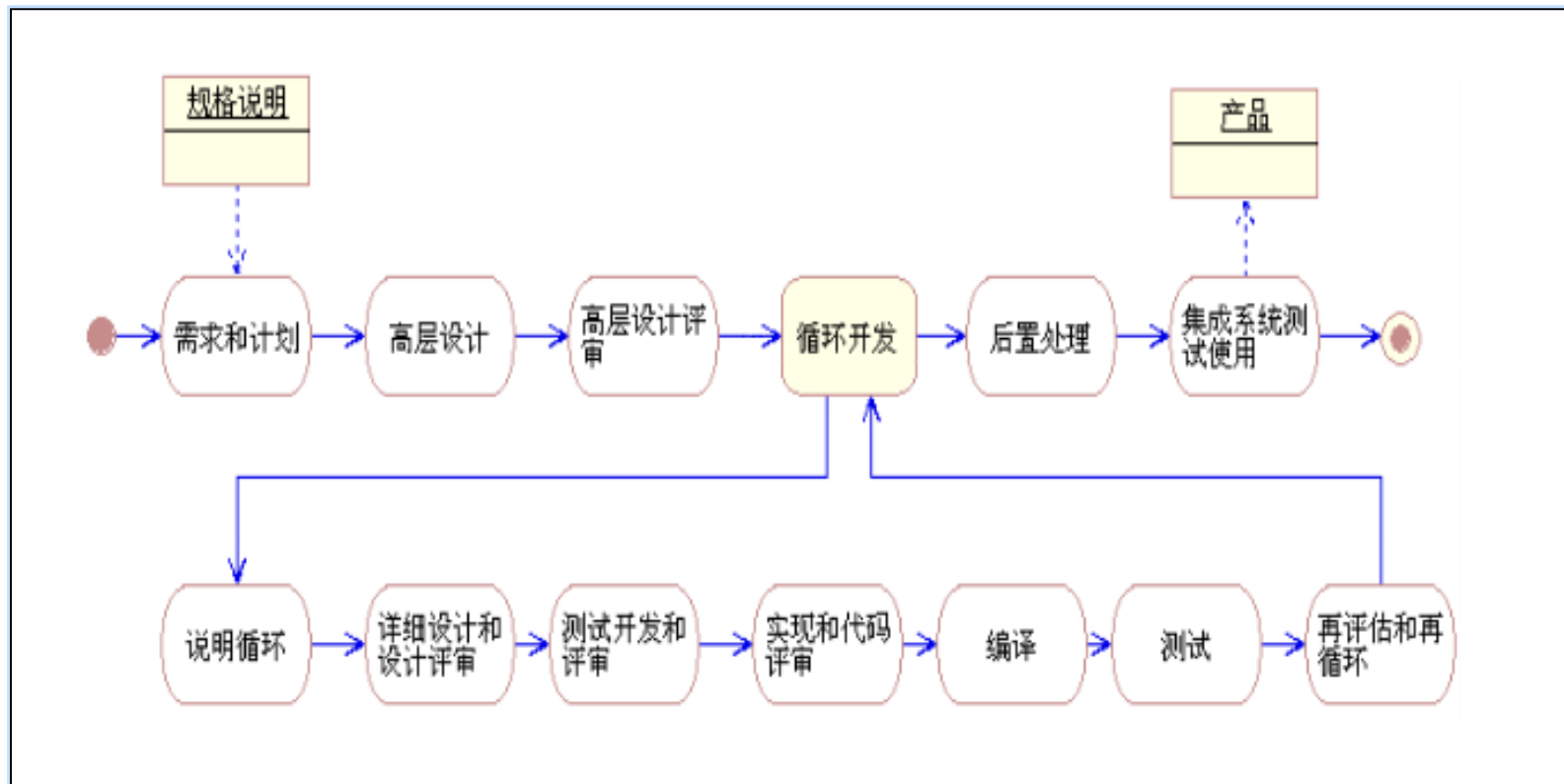
PSP2.1（个人质量过程）

- PSP2.1.**设计模板**：提供了设计过程的完全标准化，并且连同失误预防、过程分析和过程基准一起检查了各种设计检验技术。
- 可类似地将此方法应用到许多过程阶段之中去，包括：需求说明、文档和测试等。

PSP3（循环个人过程）

- PSP3就是循环个人过程。
- 以前的PSP过程专注于编写小程序的线性过程，可以适用于小规模的程序开发，PSP3则将个人软件过程的应用拓展到大规模程序应用当中。
 - PSP4.**循环过程**：将开发大型程序的个体过程细分为可以应用PSP2的片段，遵照PSP2过程循环增量地开发大型程序。在任何时间点，只有一个PSP2级过程是活动的。

PSP3 的个人循环过程



3.7 PSP/TSP

- PSP/TSP的引入
- PSP
- **TSP**

3.7.3 TSP

- PSP3可以设计过万行的程序，但是仍然存在由程序导致的两个问题：
 - 程序越大，个人花费的时间和精力越大；
 - 软件工程师很难面面俱到地关心整个程序的所有方面，以致忽略某些“视觉盲点”的错误。
- TSP通过大家共同分担来解决上述难题。也可以包括一些特殊的优化手段：定期找一个局外人来协助设计审查。

3.7.3 TSP

- 在机构中，软件开发的主体是开发小组，因此提高机构的软件开发能力实际上就是提高小组的软件开发能力。TSP提供了在**开发过程、产品和小组协同工作之间平衡**的重点，并且在规划和管理软件工程中利用了广泛的工业经验基础。
- 为了更好地实现CMM中的级别跃迁，TSP实际上是实现CMM框架的**活动指南**，它提供了一系列为特定目标而设计的活动和步骤。换句话说，CMM是战略目标，关注组织级，而TSP是战术策略，关注项目级。

3.7.3 TSP

- TSP对群组软件过程的定义、度量和改进提出了一整套原则、策略和方法，把CMM要求实施的管理与PSP要求开发人员具有的技巧结合起来，以按时交付高质量的软件，并把成本控制在预算的范围之内。
- 在TSP中，讲述了下列软件工程管理问题：
 - 如何创建高效且具有自我管理能力的工程小组；
 - 工程人员如何才能成为合格的项目组成员；
 - 管理人员如何对群组提供指导和支持；
 - 如何保持良好的工程环境使项目组能充分发挥自己的水平。
- **TSP的目标**为创建具有自我管理能力的群组，管理人员要善于引导和激励群组的全体成员，使他们能发挥自己的最高水平，采用CMM来进行软件过程的改革，为处于高成熟度的软件组织的过程改革提供指导，

3.7.4.1 TSP的四条基本原理

- 1、应该遵循一个确定的、可重复的过程并迅速获得反馈，这样才能使学习和改革最有成效；
- 2、一个群组是否高效，是由明确的目标、有效的工作环境、有能力的教练和积极的领导等四方面因素的综合作用所确定的，因此应在这四个方面同时努力，而不能偏废其中任何一个方面；
- 3、应注意及时总结经验教训，在项目中面临各种各样的实际问题并寻求有效的解决问题方案时，就会更深刻地体会到TSP的威力；
- 4、应注意借鉴前人和他人的经验，在已经可资利用的工程、科学和教学法经验的基础上来规定过程改进的指令。

3.7.4.2 TSP过程框架设计的原则

- **循序前进**，首先在PSP的基础上提出一个简单的过程框架，然后逐步完善；
- **迭代开发**，选用增量式迭代开发方法，通过几个循环开发一个产品；
- **质量优先**，对按TSP开发的软件产品，建立质量和性能的度量标准；
- **目标明确**，对实施TSP的群组及其成员的工作效果提供准确的度量；
- **定期评审**，在TSP的实施过程中，对角色和群组进行定期的评价；
- **过程规范**，对每一个项目的TSP规定明确的过程规范；
- **指令明确**，对实施TSP中可能遇到的协作问题提供解决问题的指南。

3.7.4.3 实施TSP应具备的条件

- 首先需要有**高层主管和各级管理人员**的支持，以取得必要的资源，这是实施TSP必须具备的物质基础；
- 软件过程的改善需要**全体有关人员**的积极参与，不仅需要有改革的热情和明确的目标，而且需要对当前过程有很好的了解；
- 任何过程改进都有一定的风险，都有一个实践、改进、评审直至完善的**循环往复、持续改善**的过程，不可能一蹴而就；
- 项目组的开发人员需要经过**PSP的培训**，使之具备自我改善的能力；
- 整个开发单位的能力成熟度在总体上应处于**CMM2**以上。

3.7.4.4 TSP六项管理原则

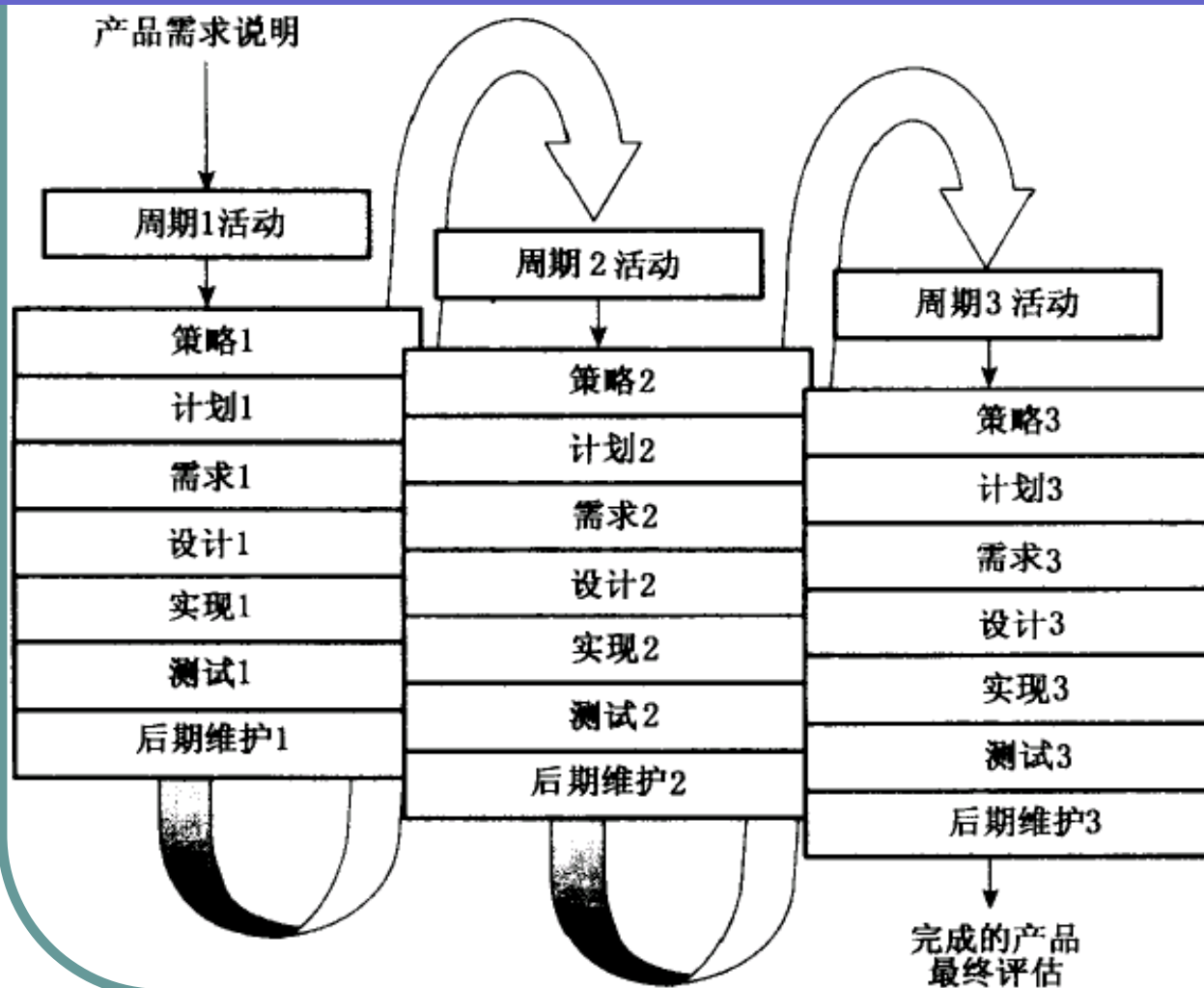
- TSP实施集体管理和自己管理自己相结合的原则：
 - **计划工作**，在每一阶段开始时制订工作计划，规定明确的目标；
 - **实事求是**，目标不应过高也不应过低而应实事求是，在检查计划时如果发现未能完成或者已经超越规定的目标，应分析原因，并根据实际情况对原有计划作必要的修改；
 - **动态监控**，一方面应定期追踪项目进展状态并向有关人员汇报，另一方面应经常评审自己是否按PSP原理进行工作；
 - **自我管理**，开发小组成员如发现过程不合适，应主动、及时地进行改进，以保证始终用高质量的过程来生产高质量的软件，任何消极埋怨或坐视等待的态度都是不对的；
 - **集体管理**，项目开发小组的全体成员都要积极参加和关心小组的工作规划、进展追踪和决策制订等工作；
 - **独立负责**，按TSP原理进行管理，每个成员都要担任一个角色。

3.7.4.4 TSP六项管理原则

- TSP的创始人Humphrey 建议在一个软件开发小组内把管理的角色分成：
 - 客户界面
 - 设计方案
 - 实现技术
 - 工作规划
 - 软件过程
 - 产品质量
 - 工程支持
 - 产品测试

如果小组成员的数目较少，则可将其中的某些角色合并，如果小组成员的数目较多，则可将其中的某些角色拆分。总之，每个成员都要独立担当一个角色。

3.7.4.5 TSP结构和流程



如何决定每个周期的内容？

(1) 每一个周期都必须完成一个最终产品的前期可测试产品。

(2) 每一个周期必须小到可以保证在一定时间内完成开发和测试任务。

(3) 各个周期产品可以组合成所要的最终产品。

3.7.4.5 TSP结构和流程

主要进行以下几项活动：

- 把产品开发划分为数个周期；
- 建立标准的质量和效益评估机制；
- 为小组和成员提供明确的评估标准；
- 进行角色和小组评估；
- 建立必要的开发纪律；
- 提供协同工作的指导。

3.7.4.5 TSP结构和流程

- 在项目开始之前，项目组应该执行启动过程，对整个任务进行全面地规划和组织。
- 在每个周期之前，项目组应该执行重启过程，对下一个周期的任务进行规划。
 - 一般来说，如果项目组的成员经过了PSP的培训，项目组的启动过程约需3天时间，重启过程约需2天时间。此时，项目组同管理人员一起评审项目计划和分析关键风险。
 - 经过3天的项目启动过程之后，项目组应该产生以下结果：**项目的目标；项目组各成员的明确角色；过程开发计划；项目组的质量计划；全面的开发计划和进度计划；下一周期每个成员的详细工作计划；项目的风险分析结果以及项目的状态报告。**
 - 在项目已经启动之后，项目组应每周进行一次项目进展讨论会，另外还应及时向有关主管和客户报告项目的进展情况。

3.7.4.5 TSP结构和流程

- TSP使用23个过程指南、14个数据表格和3个标准。
- 在这些过程指南中定义了173个启动和开发步骤。每一个步骤都不复杂，但它们的描述都非常详细，以便开发人员能够清楚地知道下一步应该做什么，应该怎样去做。这些过程指南可用来指导项目组来完成启动过程和一步步地完成整个项目。

表格缩写	表格及说明名称
DEFECT	缺陷报告表
GOAL	小组目标
INS	检查表
INV	过程目录
ITL	问题/风险跟踪日志
LOGD	缺陷记录日志
LOGT	时间记录日志
MTG	会议报告表
PIP	过程改进建议
ROLE	小组角色
ROLEMX	角色责任矩阵
SCHED	进度计划模板
STRAT	策略计划表
SUMDI	引入缺陷总结
SUMDR	排除缺陷总结
SUMP	计划总结表
SUMQ	质量总结表
SUMS	程序大小总结
SUMT	开发时间总结表
SUMTASK	任务计划总结表
TASK	任务计划模板
TESTLOG	测试日志
WEEK	周状态报告

3.7.4.6 TSP过程质量度量要素

- 对软件开发小组素质进行度量的基本要素：
 - 所编文档页数；
 - 所编代码行数；
 - 花费在各个开发阶段或各个开发任务上的时间；
 - 在各个开发阶段中注入和改正的差错数目；
 - 在各个阶段对最终产品增加的价值。
- 对软件过程质量的度量要素：
 - 软件设计时间应大于软件实现时间；
 - 设计评审时间至少应占一半以上的设计时间；
 - 代码评审时间应大于编制代码的时间；
 - 每千行源程序在编译阶段发现的差错不应超过10个；
 - 每千行源程序在测试阶段发现的差错不应超过5个。

3.7.4.7 TSP应用

- TSP的早期应用结果
 - 1996年，美国CMU/SEI建立了两个TSP项目，其中1个成功，另1个失败。
 - 1997年，CMU/SEI建立了9个TSP项目，并先举办了TSP技术培训班，其中2个很成功，2个相当成功，1个失败，时至1998年9月，另外4个尚在进之中。
 - 1998年，CMU/SEI建立了6个TSP项目，并先对TSP项目组组长进行了培训，时至1998年9月，其中1个很成功；另外5个尚未完成，但根据进展情况来看，4个进展情况良好，1个看来存在问题。
- 从公布的TSP实验数据来看，结果是令人鼓舞的。但由于尚未在巨型项目中进行TSP试验，因而尚难断定在巨型项目中实施TSP会出现什么问题，目前认为TSP比较适合规模为3~20人的开发小组。

提纲

- 3.1 CMM产生背景
- 3.2 CMM内容及管理透视
- 3.3 CMM组织结构
- 3.4 CMM应用
- 3.5 CMM与ISO9001
- 3.6 Microsoft公司的过程管理
- 3.7 PSP/TSP
- **3.8 总结**

3.8总结

- 为了系统地解决软件项目管理问题，美国国防部于1984年在Carnegie-Mellon大学建立了软件工程研究所：
 - 1986年开始研究并于1991年提出能力成熟度模型CMM；
 - 1989年开始研究并于1994年提出个体软件过程PSP；
 - 1994年开始研究并于1998年由CMU/SEI召开的过程工程年会上第一次介绍了TSP草案，于1999年发表了有关TSP的一本书，使软件过程框架形成一个包含CMM、PSP和TSP三者的严密的整体。

3.8总结

- CMM给出了一个软件组织的能力成熟度框架，并以成熟度提问单为工具进行过程评估和能力评价。
- 由于CMM中并未提供有关实现KPA所需要的具体知识和技能，因此进行PSP的研究与实践，为基于个体和小型群组软件过程的优化提供了具体、有效的途径。
- TSP结合了CMM的管理方法和PSP的工程技能，建立、管理、授权并且指导项目小组如何在满足计划费用的前提下，在承诺的期限范围内，不断生产并交付高质量的产品。