

# 第6章 其他过程模型



6.1 个体软件过程PSP

6.2 小组软件过程TSP

6.3 集成产品开发模式IPD

6.4 敏捷过程模型

6.5 统一软件过程RUP

CMM/CMMI只关注“应该做什么”，而不关注“应该怎么做”，未提供实现各过程域管理所需要的知识和方法，为了解决上述问题，CMU-SEI在CMM1.1基础上提出了PSP/TSP。

**PSP**是一种可用于控制、管理和改进软件工程师个人工作方式的自我改善过程，是一个包括软件开发表格、指南和规程的结构化框架。

对于一个要改善软件质量和过程质量的组织来说，规范软件工程师的开发过程是解决软件质量问题的必要前提。

## 6.1 个体软件过程PSP

### — 个体软件过程框架

PSP是个个体过程改进框架模型，不依赖于任何技术（语言、工具和设计方法）。

- 说明个体软件过程的原则
- 帮助工程师做准确的计划
- 告诉工程师提高软件质量的步骤
- 建立个人软件过程提升的度量标准
- 确定过程的改进对工程师能力的影响

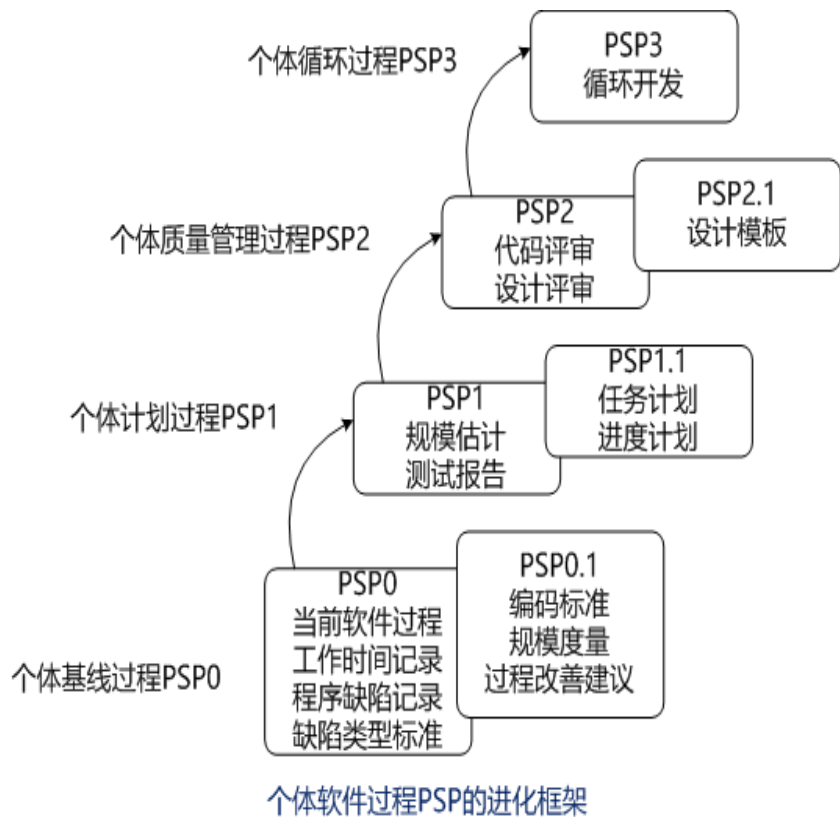
## 6.1 个体软件过程PSP

PSP为个体的能力提供一个阶梯式的进化框架（共4个级别7个台阶），以循序渐进的方法改进个体软件过程。

每个级别都包含了更低级别中的所有元素并增加新的元素。

- 个体度量过程 PSP0和PSP0.1
- 个体计划过程 PSP1和PSP1.1
- 个体质量过程 PSP2和PSP2.1
- 个体循环过程 PSP3

## — 个体软件过程框架



- **PSP0是建立个体过程基线**，目的是为了在个人的工作中引入表格和脚本，以便工程师按照测量和报告格式记录软件过程。

- **PSP0-1 目前过程**：记录软件工程师在工程中使用的具有代表性的软件开发方法。

- **PSP0-2 时间记录**：记录软件工程师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 缺陷记录**：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 缺陷分类标准**：一方面为软件工程师提供在系统中可观察到的典型缺陷种类列表，有助于软件工程师把典型缺陷标准化；另一方面提供一种预定义的步骤和工具方便软件工程师对新的缺陷进行归类 and 记录。

- PSP0可以通过增加下列过程而扩展到 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和编码标准。



## 6.1 个体软件过程PSP

### — 个体基线过程PSP0

— **PSP0.1-2 代码规模度量**：测量代码的长度、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，如代码行度量LOC和功能点度量FPA方法，软件工程师应该了解相关软件规模度量概念。

— **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改进建议，并以结构化的方式表达软件过程、问题、建议教训、改进建议等项目。

PSP1在PSP0的基础上增加了计划步骤。

- **PSP1-1 规模估计：**分为代码规模估算、时间估算、资源估算。

- 代码规模估算：软件工程师可以凭借PSP0级代码规模测量经验预测他们将要写的任务模块或算法的可能规模。

- 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完成每个模块的估算时间（乐观时间、最可能时间、悲观时间）。

## 6.1 个体软件过程PSP

### — 个体计划过程PSP1

— 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，检查规模估计与实际状态之间的差异。

PSP1.1在PSP1的基础上引入了任务计划和安排。

- **PSP1.1-1~2 任务计划及进度安排**：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。

PSP2强调提高质量，引入了缺陷管理。

- **PSP2-1 代码审查。**对代码进行检查和分析，以发现程序缺陷。

- **PSP2-2 设计审查。**设计审查要求提供一些评估设计质量的指标，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在PSP2.0基础上增加了设计模板。

- **PSP2.1-1 设计模板：**提供了设计过程的完全标准化，并且连同缺陷预防、过程分析和过程基准一起形成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将个体开发小程序所等达到的生产效率和生成目标，延伸到大型程序开发；

采用螺旋式上升过程，将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，最后把这些模块逐步集成为完整的软件产品。

# 6.1 个体软件过程PSP

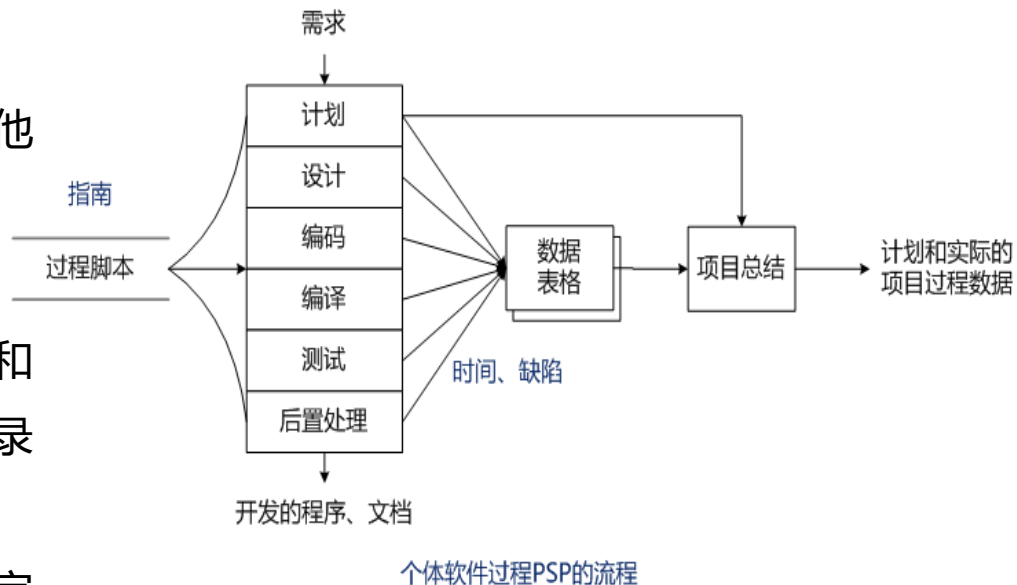
## —— 个体软件过程流程

(1) 从需求状态开始，第一步是计划，有一个计划脚本指导这个工作并且有一个计划汇总记录相关的计划数据；

(2) 各阶段，工程师按照脚本工作，并将他们的时间和缺陷数据记录到时间和缺陷日志

(3) 工作结束即事后分析阶段，汇总时间和缺陷数据，度量程序规模，并将这些数据记录进计划汇总表单；

(4) 当所有这些完成后；将完成的产品和完整的计划汇总表单一起交付。





## 6.1 个体软件过程PSP

## —— 个体软件过程流程

个体软件过程PSP的过程脚本（一系列步骤）

过程目的	指导用户进行小型程序的开发
入口准则	问题描述；项目计划总结表；以前开发程序的实际规模和时间数据；时间记录日志
1、计划	获取对程序功能的描述；估计整个程序的代码行数及其可能的最大和最小值；计算开发效率的历史数据(min/LOC)；估算开发时间及其可能的最大最小值；将计划数据填入项目计划总结表；将计划阶段所花费的时间记入时间记录日志。
2、设计	进行系统设计；按照指定的格式记录设计文档；将设计阶段所花费的时间记入时间记录日志。
3、编码	编码实现系统设计；使用标准的格式来书写程序代码；将编码阶段所花费的时间记入时间记录日志。
4、编译	编译程序；修复所有发现的缺陷；将编译阶段所花费的时间记入时间记录日志。
5、测试	测试整个程序；修复所有发现的缺陷；将测试阶段所花费的时间记入时间记录日志。
6、后置处理	根据实际的规模和时间数据填写项目计划总结表；将后置处理阶段所花费的时间记入时间记录日志。
出口准则	经过详尽测试的程序；设计文档；源程序清单及源程序；项目计划总结表；时间记录日志。

## 6.1 个体软件过程PSP

### — 核心技能

时间管理

计划管理

缺陷管理

规模度量

### ●时间管理的逻辑原理

- 为了管理好时间，首先制定时间分配计划，然后按照计划去做；
- 为了制定切实可行的计划，必须对所用的时间进行跟踪；
- 为了检查时间估计和计划的准确性，必须把它们写成文档并在今后与实际情况进行比较；
- 为了制定出更准确的计划，需要知道以前的计划中存在哪些错误，哪些地方可以进行改进。

### ● 了解时间的使用情况

- 将主要活动进行分类
- 记录每项主要活动所花费的时间
- 用标准的方法记录时间
  - 以分钟为测量单位
  - 处理中断时间
  - 将时间数据保存在合适的地方
  - 一周活动总结表
  - 记录时间的提示

### ● 管理好时间

- 复查时间分类
- 制定时间安排
- 找出更多时间
- 制定基本原则
- 设定时间分配的优先级
- 做出全面的时间安排
- 执行时间安排表
- 时间管理目标

### ●制定计划内容

- 阶段计划：关于这段时间内对时间的安排；基于时间段的，时间段即日历上的一段时间，如一天、一周、一个月或一年。
- 产品计划：是关于制作产品活动期间的时间安排。基于活动的计划，开发一个程序或写一份报告

### ●如何制定产品计划

一份合适的产品计划包括三个方面：产品的规模及重要的特性、完成工作所需要时间的估计、项目进度计划。

- 收集历史项目数据：通过收集以前不同任务所用时间的数据，就能够估计将来类似的任务大概所需要的时间；
- 估算程序规模：产品计划的第一步是要估计产品的规模。

- 如何制定阶段计划

- 以周为单位进行活动总结表
- 必须清楚时间的使用情况（记录在周活动表）
- 总结时间日志数据
- 计算阶段时间和工作效率
- 制作并实施阶段计划



### ●什么是缺陷

- 缺陷是指程序中存在的错误；缺陷是任何影响程序完整而有效的满足用户要求的东西；
- 缺陷是可以表示、描述和统计的客观事物；缺陷可能出现在程序或设计中，甚至在需求、规格说明或其他文档中；
- 缺陷是程序员引入的（平均7-10行源程序就会引入一个缺陷）发现和修复缺陷需要大量的时间和费用（一般组织需要50%的精力、时间、经费来查找和修复缺陷）

**减少缺陷对每个工程师都很重要。**

### ●缺陷分类

集中精力预防  
和排除问题  
最大的缺陷  
是缺陷管理  
的关键

类型编号	类型名称	描述
10	文档	注释, 信息
20	语法	拼写, 标点符号, 打字, 指令格式
30	联编打包	变更管理, 库, 版本控制
40	赋值	说明, 重名, 作用域, 限制
50	接口	过程调用和引用, 输入输出, 用户格式
60	检查	出错信息, 不合适的检查
70	数据	结构, 内容
80	函数	逻辑, 指针, 循环, 递归, 计算, 函数缺陷
90	系统	配置, 记时, 内存
100	环境	设计, 编译, 测试, 其他支持系统问题

### ● 尽早发现缺陷

➤ 软件人员通常认为测试能找到产品全部或者大多数缺陷，然而事实证明：即使是运行很好的单元测试，在查找缺陷方面也不超过70%效率；

➤ 集成测试和系统测试为45%；而功能测试只有8%

正确的做法是，在测试前集中精力发现或防止缺陷

### ●发现和修复缺陷的方法

- 编译：能标识出大部分语法缺陷，但它不能辨别你的意图是什么；
- 测试：测试的质量是由这些测试用例覆盖程序功能的程度决定的；
- 用户反馈：用户使用中出现的问题；
- 代码审查：就是逐行研究源代码，从中发现错误。虽然很难彻底清除程序中的缺陷，但事实证明，这是最快且最有效的方法

### ●代码复查

- 复查前，检查相关产品是否已准备好。如：需求规格说明书、程序设计文档、程序的源代码清单、编码标准、代码复查检查表；
- 复查过程。首先完成源程序编码，然后在进行编译和测试之前，打印一份程序清单；进行代码复查时，仔细检查每行源程序，尽可能发现和修复缺陷；
- 修复缺陷：修复所发现的每一个缺陷，确保所作的修复正确无误，将缺陷登记在缺陷记录日志。

### ●代码复查

- 命名、类型和变量检查。包括：验证所有的名字和类型已经正确的声明；检查整型、长整型和浮点型是否正确声明；确保每个变量已初始化化；检查上溢、下溢或越界问题；
- 程序语法检查。验证程序代码符合编程语言的规格说明
- 复查结束条件。完成并修复过的源程序清单；填写完整的时间记录日志；填写完整的缺陷记录日志。

### ●编码标准

- 良好的编码标准将有效地帮助您避免开发有潜在危险的代码，有助于预防缺陷；编码标准还能有效地统一和规范整体开发活动；

### ●同行评查

- 几个工程师彼此复查程序，组织良好的同行评审一般会发现程序中50%~70%的缺陷；

### ●软件项目估算

- 对问题进行分解，把其分解为一组较小的接近于最终解决的可控的子问题，再定义他们的特性；
- 规模测量的方法很多，应根据不同的对象使用不同的估算方法
- 没有任何方法可以保证估计的结果一定准确，做出好的规模估计的关键是要有大量的历史数据和经验，要进行多次规模估计，并且要定期将实际结果与估计值进行比较。



- 估算程序规模方法

- 代码行测量方法;

- 功能点方法

- 用例方法

- 利用历史数据或凭实际经验，对每个功能分别按乐观的、可能的、悲观的三种情况给出功能点（FP）估计值



6.1 个体软件过程PSP

6.2 小组软件过程TSP

6.3 集成产品开发模式IPD

6.4 敏捷过程模型

6.5 统一软件过程RUP

几乎所有的软件都是由项目小组开发的，一个优秀的软件工程师应该学会在小组中工作。

小组工作，意味着小组成员必须目标一致，自由交流，遵循共同的工作过程和规程，对项目进行计划，对进展进行跟踪，对工作进行协调。

为了系统地解决软件项目管理问题，美国国防部于1984年在卡耐基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年在过程工程年会上第一次介绍了TSP草案，于1999年发表了TSP；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。

小组软件过程TSP，致力于开发高质量的产品，通过建立、管理和授权项目小组，指导小组成员在满足计划费用的前提下，在承诺的期限内，不断生产并交付高质量的产品。

- 创建具有自我管理能力的团队；
- 管理人员要善于引导和激励团队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进提供指导、积极培训人才。

在实施团队软件过程TSP的过程中，应该自始至终贯彻集体管理与自我管理相结合的原则：

- 计划工作的原则：

在每一阶段开始时要制定工作计划、规定明确的目标；

- 实事求是的原则：

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做必要的修改；

- 动态监控的原则**：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；
- 自我管理的原则**：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；
- 集体管理的原则**：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、进展追踪和决策制定等工作；

### ●独立负责的原则

- 按TSP原理进行管理，每个成员都要担任一个角色
- 在TSP的实践过程中，TSP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、软件工程、产品质量、工程支持以及产品测试等八类；
- 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。



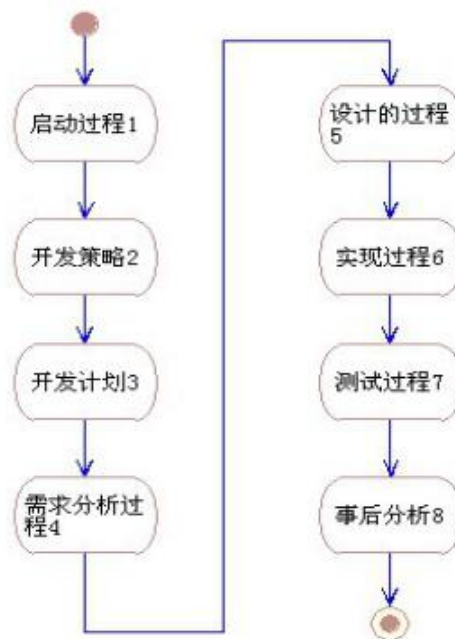
软件开发小组按小组软件过程TSP进行生产、维护软件或提供服务，其质量可用两组元素来表达：

- 一组元素用以度量开发小组的素质，称之为**开发小组素质度量元**；
  - 所编文档的页数；
  - 所编代码的行数；
  - 花费在各个开发阶段或花费在各个开发任务上的时间（以分为单位）；
  - 在各个开发阶段中注入和改正的缺陷数目；
  - 在各个阶段对最终产品增加的价值。

- 另一组用以度量软件过程的质量，称之为**软件过程质量度量元**。
  - 设计工作量应大于编码工作量；
  - 设计评查工作量至少应占一半以上的设计工作量
  - 代码评审工作量应占一半以上的代码编制工作量
  - 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

### TSP小组软件过程

- 1. 启动过程
- 2. 开发的策略
- 3. 开发计划
- 4. 需求分析
- 5. 设计过程
- 6. 实现的过程
- 7. 测试计划
- 8. 事后分析



## 6.2 小组软件过程TSP

### — TSP的过程

1	启动	启动小组开发周期；回顾先前周期的经验教训；形成或重新形成项目小组、分配或重新分配小组各个角色；说明要开发的产品的目标；建立小组会议机制和时间；确定工作进展和过程数据报告方式和时间；明确项目日志的要求。
2	策略	产品元素的概念设计、开发策略、规模和时间估计、风险评估、配置管理；本周期要产生的所有产品元素。
3	计划	任务计划、进度计划、质量计划、个体工程师计划；平衡小组工作量。
4	需求	需求规格说明SRS和测试计划。
5	设计	系统总体结构、设计规格说明SDS。
6	实现	完成经过复审、检查和单元测试的产品元素。
7	测试	集成产品元素到前一版可运行系统中，经过系统测试，形成下一版可运行系统；迭代式开发过程；产生或更新操作手册；缺陷跟踪与管理。
8	后置处理	收集、分析和记录本周期的项目数据；评价小组和每一个角色的表现；识别改进后继周期过程的途径；产生本周期报告。

## 6.2 小组软件过程TSP

### — TSP的角色

1	小组领导	建造并维护一个有效的小组；激励所有的小组成员在项目上积极工作；解决小组成员带给的所有问题；使指导者充分得到有关小组的进展报告；作为小组会议推动者有效地工作；小组对项目的同行评价是经验有益的。
2	开发管理者	充分利用小组成员的能力和才干，生产高质量的产品。
3	计划管理者	为小组和每个小组成员制定平衡的、完整的、准确的计划；每周准确报告小组状态。
4	质量/过程管理者	所有的小组成员都准确报告并合理使用TSP数据；小组如实遵循TSP，并生产高质量的产品；所有的小组检查均得到合理的协调和报告；所有的小组会议均得到准确报告。
5	支持管理者	小组有合适的方法和工具支持工作；基线产品中没有任何未经批准的变更；所有小组的风险和问题，均在问题跟踪日志中记录，并得到报告；小组满足本开发周期的重用目标。

- **每一个角色都描述四种特征**

- 小组角色的目标（每个目标都可度量）；
- 对小组角色有帮助的技能和能力；
- 小组角色的主要活动；
- 小组角色在工程中的活动；


- PSP、TSP 和CMMI为软件产业提供了一个集成化的、三维的软件过程改革框架。
- PSP注重于个人的技能，能够指导软件工程师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。经过PSP的学习和实践，软件工程师们能够在他们参与的项目中更为高效和高质量地完成工作，从而保证了项目整体的进度和质量。

## 6.2 小组软件过程TSP

### — PSP、TSP和CMM的关系

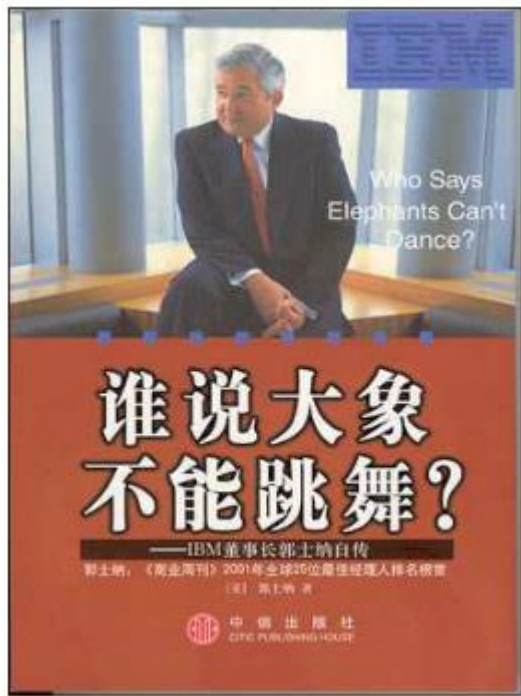
- TSP注重团队的高效工作和产品交付能力，结合PSP的工程技能，使软件工程师将个体过程结合进小组软件过程，并通过指导管理层如何支持和授权项目小组，坚持团队的高质量的工作，并且依据数据进行项目管理，以达到生产高质量产品的目的。
- CMMI注重于组织能力和成熟度的提高，它提供了评价组织的能力、改进组织过程的管理方式，比TSP具有更高的层次。
- CMMI关注“做什么”，PSP和TSP则提供了“怎么做”。



- 
- 6.1 个体软件过程PSP
  - 6.2 小组软件过程TSP
  - 6.3 集成产品开发模式IPD
  - 6.4 敏捷过程模型
  - 6.5 统一软件过程RUP

## 6.3 集成产品开发模型IPD

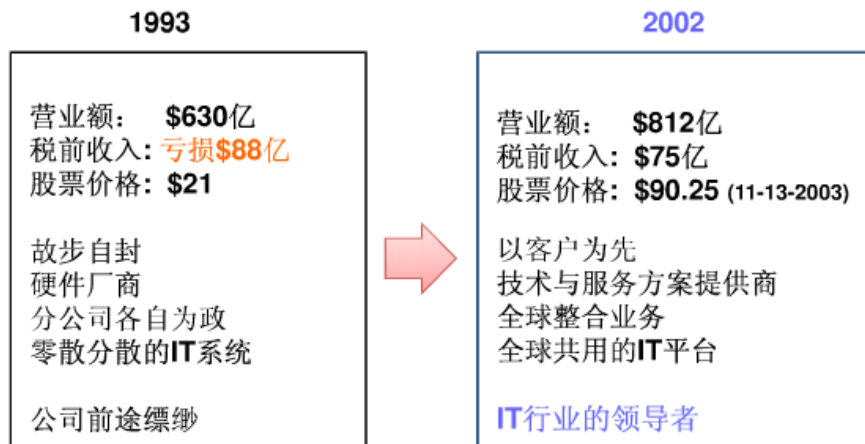
### —— IPD的产生



- 1992年，蓝色巨人IBM曾因企业庞大和机构臃肿而陷入困境，IBM前总裁郭士纳临危受命扭转乾坤；
- 2002年，郭士纳的自传《谁说大象不能跳舞》一书中，编者曾这样评价郭士纳的功绩：一是保持了IBM这头巨象的完整；二是使IBM由生产硬件转型到服务上来；三是把大象跳的舞编成了IPD这首舞曲；
- 书中详细描述了IPD在IBM的诞生过程及带来的变化。

## 6.3 集成产品开发模型IPD

## — IPD的产生



IBM变革转型中策略，财务，产品的成功得到了广泛的关注和赞誉

IBM公司实施IPD的效果，最显著的改进在于：

- 产品研发周期显著缩短；
- 产品成本降低；
- 研发费用占总收入的比率降低，人均产出率大幅提高；
- 产品质量普遍提高；
- 花费在中途废止项目上的费用明显减少；

- IPD: Integrated Product Development（集成产品开发）是一套先进的、成熟的产品开发的管理思想、模式和方法
- IPD是面向客户需求、贯穿产品生命周期的产品开发模式
  - ▣ 关注根据企业战略制定产品战略
  - ▣ 理解市场环境
  - ▣ 设别和选择业务机会并转化为投资组合和业务计划
  - ▣ 通过异步开发模式、跨职能团队和结构化的并行开发流程，将产品成功推向市场
  - ▣ 实施生命周期管理以实现业务目标。所以，IPD本质上是一种产品经营模式

### (1) 产品开发是一项投资决策

IPD强调对产品开发进行有效的投资组合分析和管理，优化投资组合，将资源用于最有前途的市场机会和产品组合上。

在开发过程设置检查点，通过阶段性的投资决策评审来决定项目是继续、暂停、终止、还是改变方向。

### (2) 基于市场的创新和开发

IPD强调产品创新一定是基于市场需求、定位和竞争分析的创新，而不是基于技术或功能上的创新。

IPD强调把**正确定义产品概念**、**市场需求**作为流程的第一步，着眼于一开始就把事情做正确，并将产品开发的流程与市场管理流程有机地集成起来。

### (3) 跨部门、跨系统的协同

产品开发是一个跨部门的流程，必须有一个跨部门的小组对最终结果负责，协同各项活动，确保沟通、协调和决策的高效，达到尽快将产品推向是市场的目的。

### (4) 基于平台的异步开发模式

也称并行工程，是缩短新产品上市周期的重要手段。通过严密的计划、准确的接口设计，将产品开发按照最终产品、平台、子系统和技术分解为不同层次的任务，并行开发所有层次的任务。

通过对每个层次的关注和面向市场的开发，快速、高效、不断地推出具有竞争力的产品。



### (5) 重用

采用公用构建模块，提高产品开发效率。

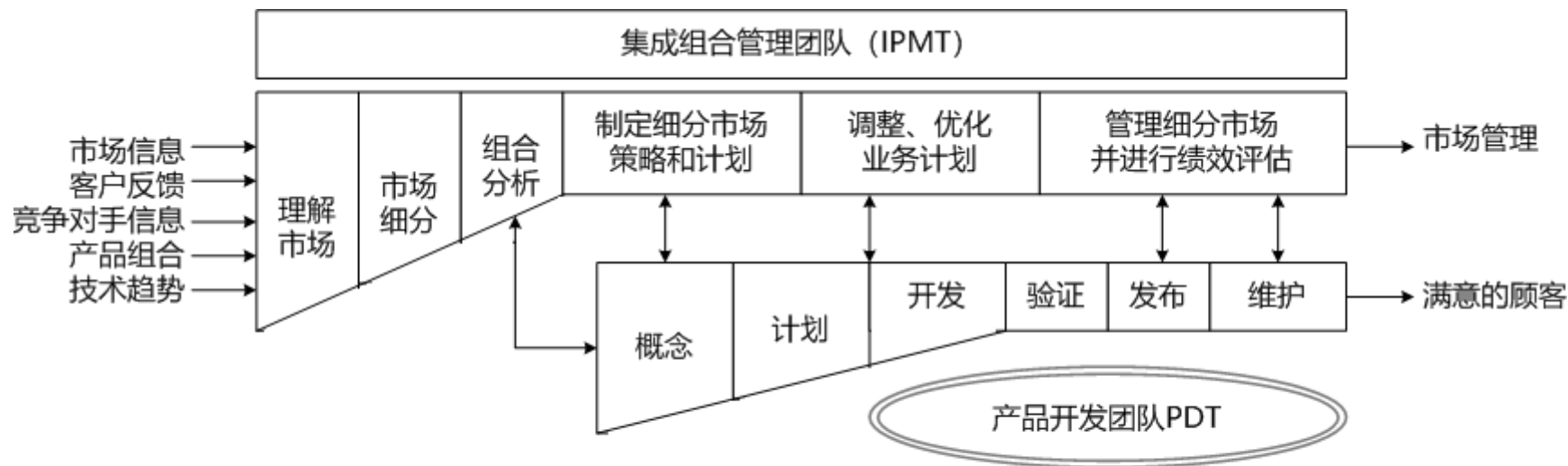
公用构建模块是实现异步开发的基础的手段。当产品的集成是基于许多成熟的、共享的公用构建模块和技术，产品的质量、进度和成本无疑会得到更好的控制和保证。

### (6) 结构化的流程

产品开发活动庞大而复杂，涉及到方方面面，因此产品开发过程应成为结构合理、定义清楚的过程。这是结构化的要求，就像企业生产流水线，但是产品开发流程不同于企业生产流水线，具有相对不确定性。理想的生产是复制，而产品开发是有限度的创新。因此IPD流程是**有限度的结构化**，不能规定得太细太机械。

## 6.3 集成产品开发模型IPD

### — IPD的开发框架



IPD开发框架

### (1) 二大流程

**市场管理流程：**从理解市场、细分市场到管理、评估市场的全过程。

**IPD流程：**从产品概念形成、产品开发计划、开发、验证、发布和维护等软件开发生命周期过程。

### (2) 二个跨部门团队

**集成组合管理团队IPMT：**是决策团队，负责整个产品的运作，包括市场分析、定义、管理和软件产品开发生的管理。

**产品开发团队PDT：**是执行团队，集中在软件产品的开发计划、实施和评估。

### (3) IPD过程要素

集成组合管理团队IPMT的决策机制、产品开发团队PDT运作机制，定义各层次的业务流程，并细化为操作指南和模板。

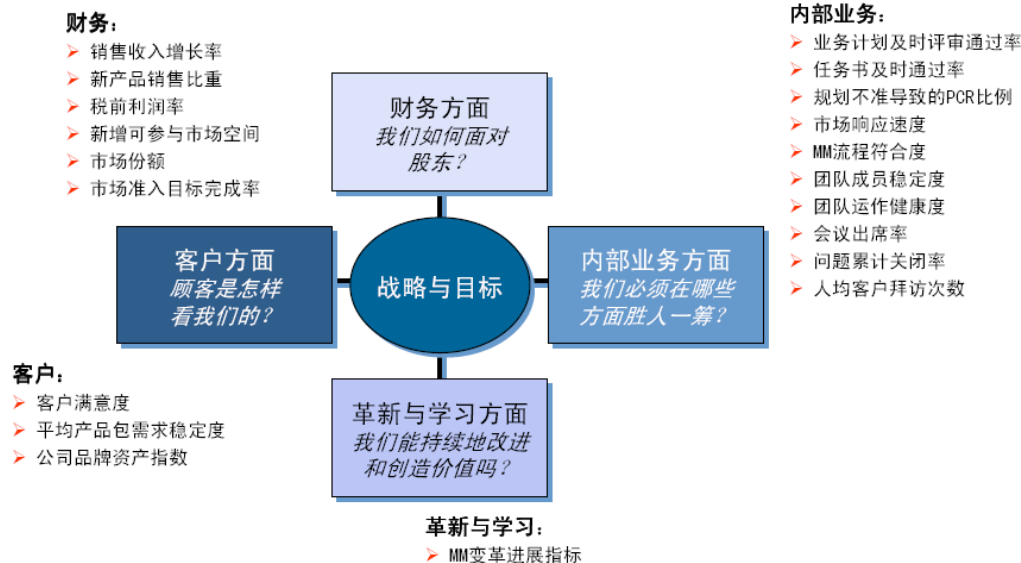
项目管理、阶段决策、以用户为中心的设计、技术评审、系统工程、重用、文档管理、质量控制、采购管理、技术与工具管理等过程要素。

## 6.3 集成产品开发模型IPD

## —— IPD的开发框架

### (4) 过程性能评估

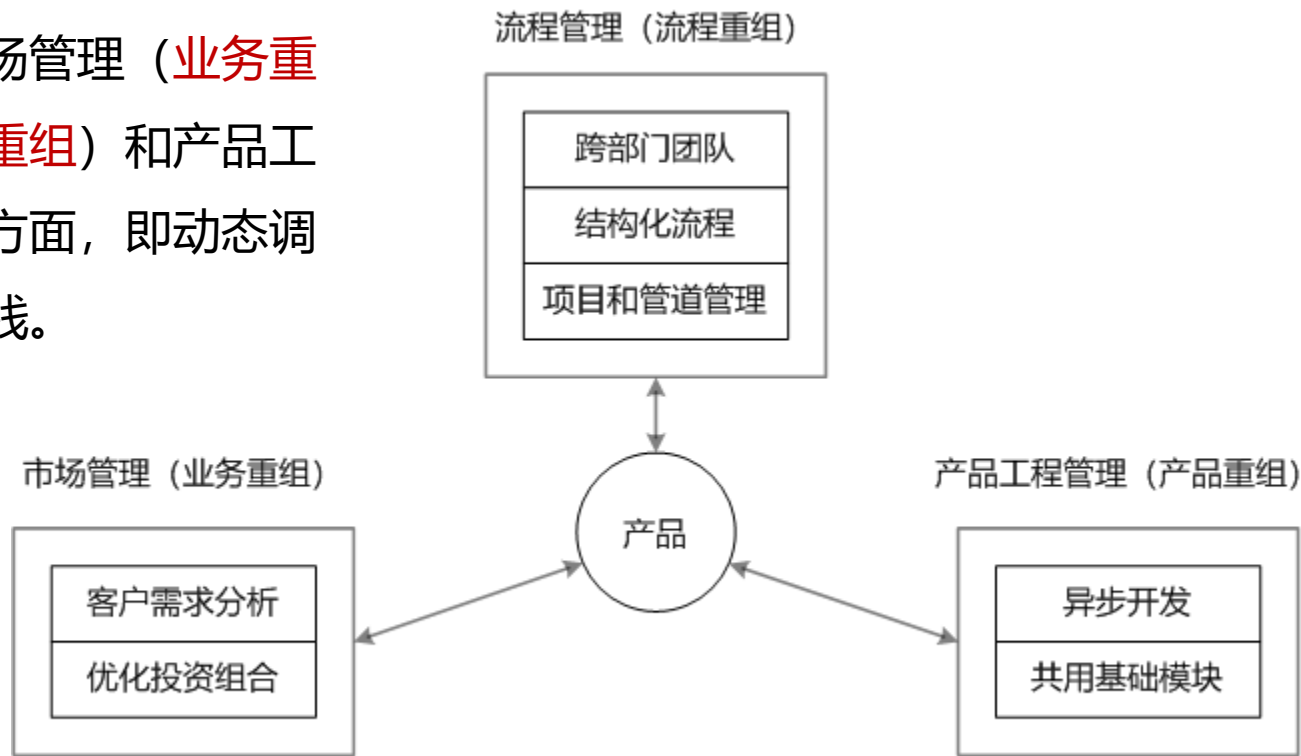
采用**平衡计分卡BSC**，从财务、客户、内部运营、学习与成长四个维度，将组织的战略落实为可操作的衡量指标和目标值的一种新型绩效管理体系。



## 6.3 集成产品开发模型IPD

### —— IPD的关键实践

IPD的应用，不仅取决于自身开发框架的建立，还取决于市场管理（**业务重组**）、流程管理（**流程重组**）和产品工程管理（**产品重组**）等方面，即动态调优的方法体系和关键实践。



IPD方法体系结构



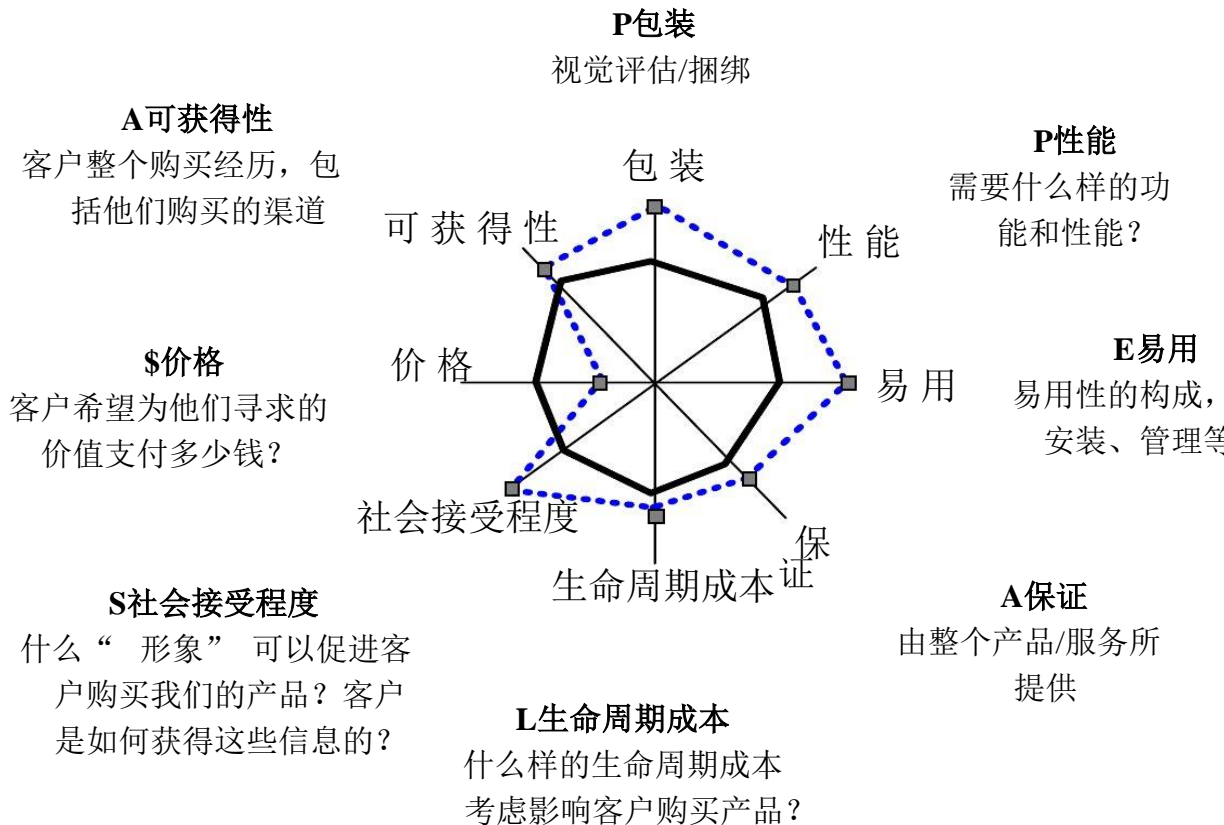
## 6.3 集成产品开发模型IPD

### —— IPD的关键实践

#### 1、客户需求分析

没有需求就没有产品，及时地、准确地把握市场需求是产品开发成功的关键。

IPD客户需求分析是指八个方面（即\$APPEALS）来了解客户对产品的需求，并依此与业界主要对手进行对比分析，确定细分市场的产品需求定位和竞争策略



\$APPEALS是客户的购买行为标准，以此了解我们所提供的服务/解决竞争力

## 6.3 集成产品开发模型IPD

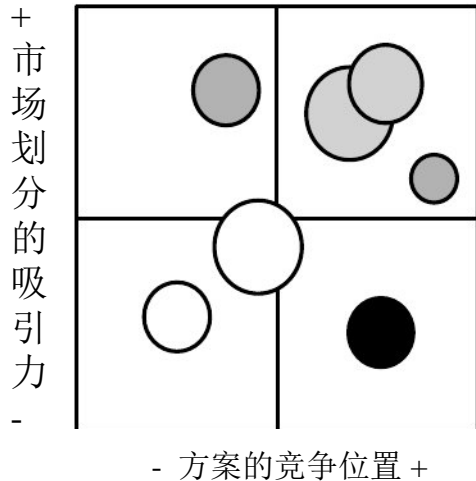
## — IPD的关键实践

### 2、投资组合分析

投资组合分析的主要任务是正确的制定投资对策、提高运营效率、取得满意的产品收益。

投资组合分析需要考虑服务方向、竞争对手、市场需求、企业优势、资源条件、收益目标等因素，还必须研究产品结构、投资利润率与市场占有率、市场成长率的关系等方面。

投资组合分析要贯穿在整个产品生命周期，通过阶段性的决策评审来决定项目继续、暂停、终止等。



### 3、跨部门团队

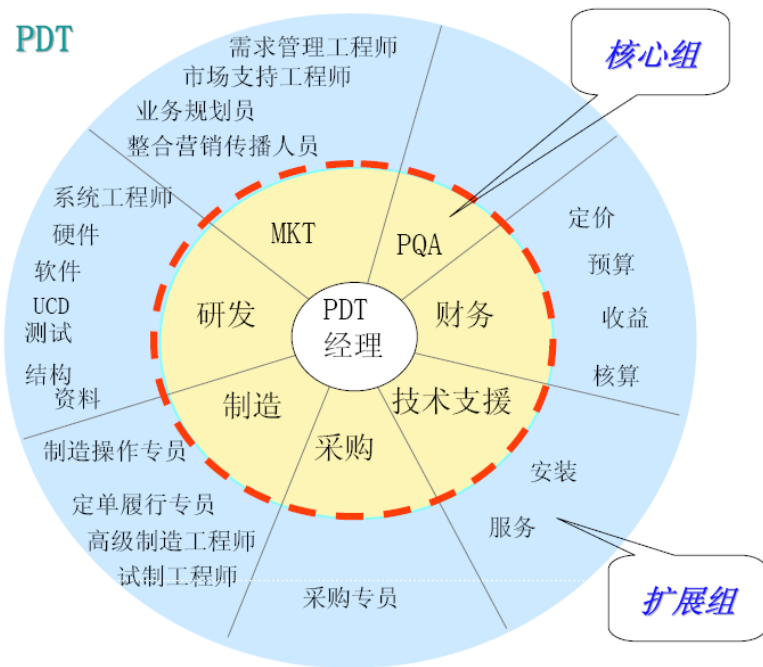
组织结构是流程运作的基本保证。集成组合管理团队IPMT和产品开发团队PDT都是由跨部门的人员组成。

IPMT由组织决策人员组成，其工作是确保在市场上有正确的产品定位，保证项目投资、控制投资。一个IPMT团队可以管理多个PDT团队。

## 6.3 集成产品开发模型IPD

## —— IPD的关键实践

PDT是具体的产品开发团队，其工作是制定具体产品策略和开发计划，并保证按照计划及时将产品投放到市场。PDT是一个动态组织单元，其成员在产品开发期间一起工作，产品开发结束后解散。



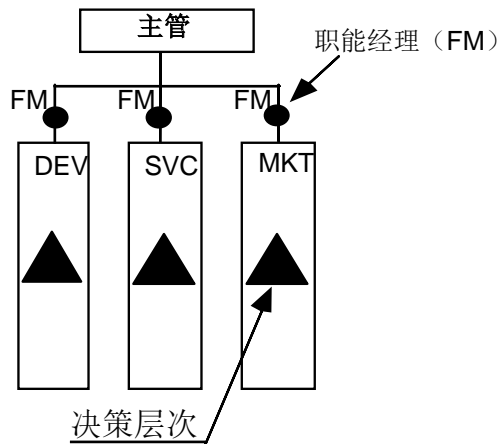
**PDT采用“重度矩阵结构”模式，保证沟通、协调和决策的高效。**

- **PDT经理**在不同功能中发挥直接的、综合性的影响
- 组员完全代表相应的职能部门
- **PDT经理和成员**有项目权力和责任
- 职能部门经理（资源经理）关注于建立优秀的部门，而不是日常的决策
- 是复杂项目和组织的最佳结构

## 6.3 集成产品开发模型IPD

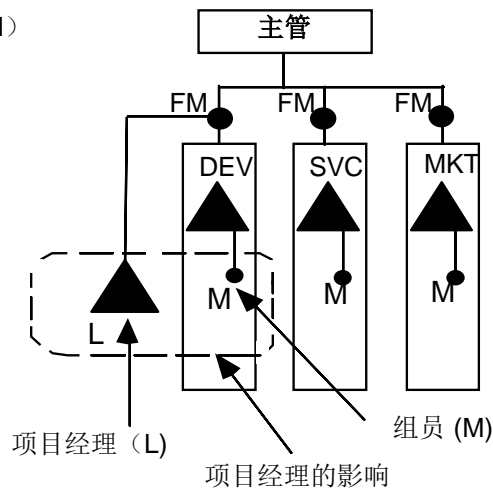
## — IPD的关键实践

职能结构



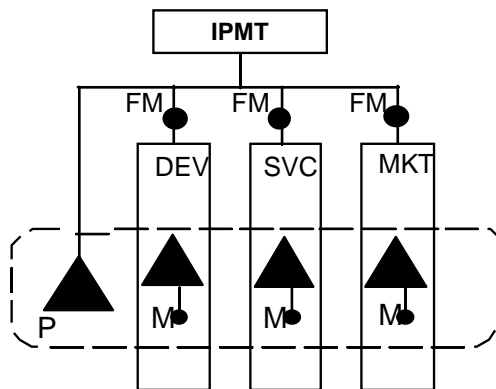
- 职能部门经理处理本部门的所有决策
- 当项目或组织变得很大或需要广泛的跨部门运作时，难于协调

“轻度矩阵”团队结构



- 项目经理是协调人
- 项目组成员是职能部门的联络员（没有权力）
- 职能部门经理仍然做出本部门的关键决策
- 当规模和复杂度增大时，这种结构也难于支持

“重度矩阵”团队结构



- 项目经理在不同功能中发挥直接的、综合性的影响
- 组员完全代表相应的职能部门
- 项目经理和成员有项目权力和责任
- 职能部门经理关注于建立优秀的部门，而不是日常的决策
- 是复杂项目和组织的最好的组织结构

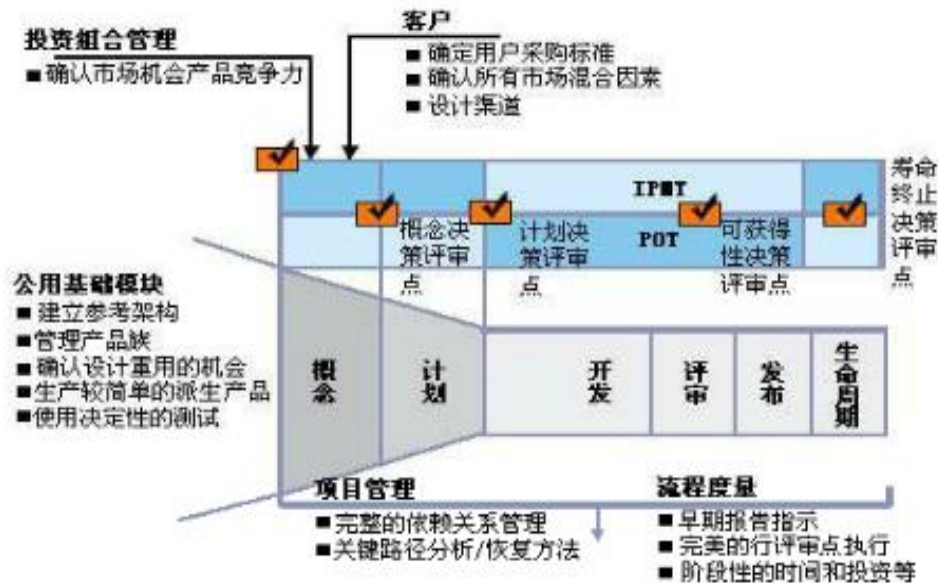
## 6.3 集成产品开发模型IPD

## —— IPD的关键实践

### 4、结构化流程

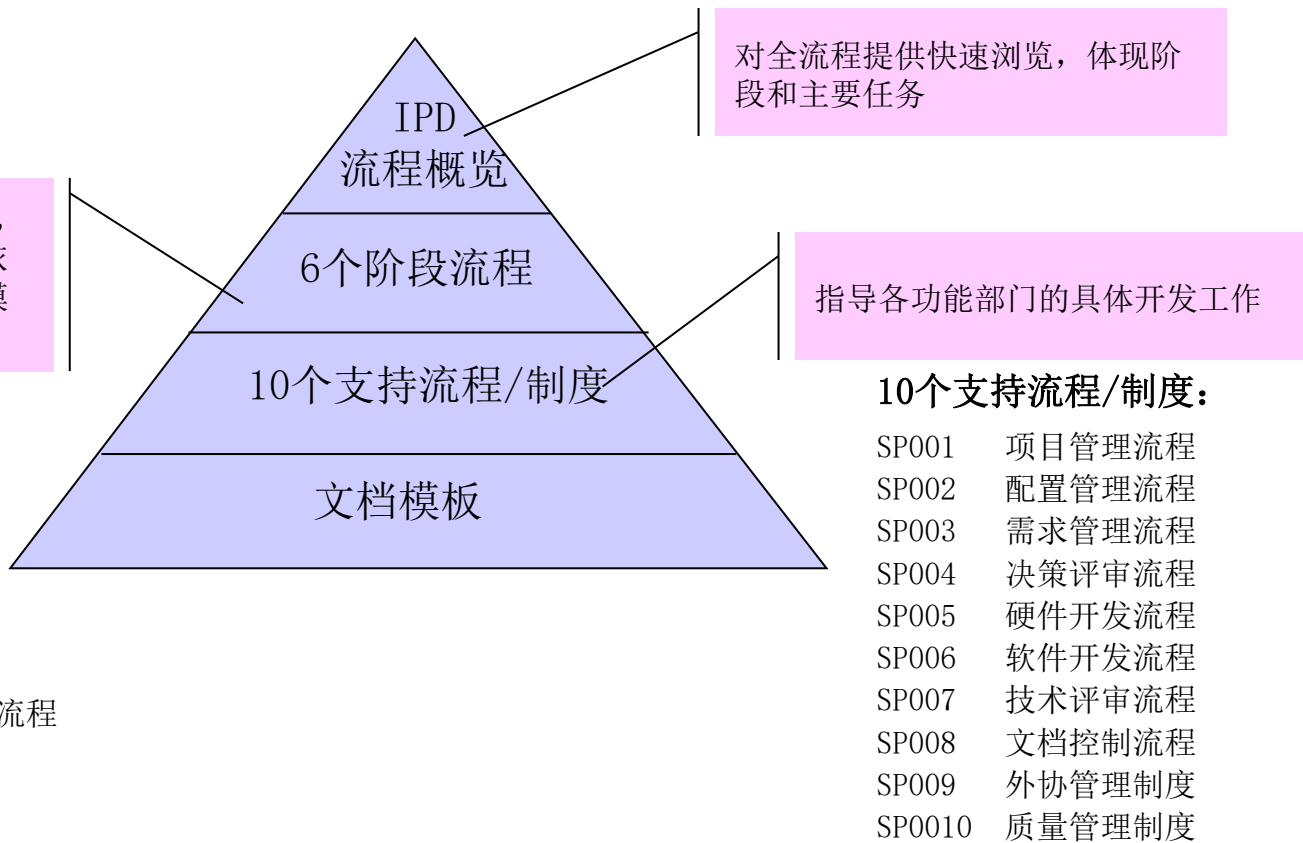
IPD产品开发流程被明确划分为概念、计划、开发、验证、发布和维护六个阶段，并且在流程中有清晰的决策评审点。决策评审点有一致的衡量标准，只有通过评审才能进入下一个阶段。

在每个阶段，可根据产品特点制定详细的子流程、任务、活动、技术、工具等内容。



## 6.3 集成产品开发模型IPD

## — IPD的关键实践



### 6个阶段流程:

PP001	概念阶段流程
PP002	计划阶段流程
PP003	开发阶段流程
PP004	验证阶段流程
PP005	发布阶段流程
PP006	产品生命周期管理流程

### 10个支持流程/制度:

SP001	项目管理流程
SP002	配置管理流程
SP003	需求管理流程
SP004	决策评审流程
SP005	硬件开发流程
SP006	软件开发流程
SP007	技术评审流程
SP008	文档控制流程
SP009	外协管理制度
SP0010	质量管理制度

### 5、项目和管道管理

项目管理包括确定任务及任务关系，安排各项活动的预算、进度和资源，按计划监督与控制，跨部门团队有效配合等内容。

管道管理类似于多任务处理系统中的资源调度和管理，指根据业务策略对开发项目及其所需资源进行优先排序及动态平衡的过程。



### 6、异步开发

异步开发模式的基本思想是在纵向将产品开发划分为不同层次，如业务层、逻辑层、数据层和平台层等。不同层次工作由不同小组异步开发完成。

通过定义各层次的标准接口和规范，尽量减少各层次之间的相互依赖关系。

### 7、共用基础模块

共用基础模块是指那些可以在不同产品、系统之间共用的组件、构件、数据块、技术及其相关的设计成果，实现重用和共享。

产品开发中尽可能地使用成熟的公用基础模块和技术，则产品的质量、进度和成本会得到很好的控制和保证，产品开发中的技术风险也将极大地降低。

CMM是SEI针对软件质量保证制定的能力成熟度模型，与ISO9000系列标准和MIL标准一样，均属于过程质量模型。CMMI是在CMM的基础上发展起来的，与CMM一样，本质上关注的是过程质量。

IPD关注的是将影响产品成功的关键要素（如结构化流程、产品决策评审、产品开发团队等）有机地整合起来，形成集成的产品开发模式，推动产品的成功。

具体而言，IPD与CMMI在以下方面具有明显的区别：

### (1) 两者的层面不一样

IPD是企业层面的一套产品开发管理的思想、模式和方法，本质上是一种产品经营管理的模式。CMMI是面向研发的，而且更多是面向软件开发的。

### (2) 思想高度不一样

CMMI主要倡导通过过程和活动来保证质量。IPD是从更高和更加全面的角度来看待产品开发的。

### (3) 对流程的结构化不一样

IPD把产品开发看作一个流程，并建立一个涵盖了流程概览、阶段流程、子流程和模板的分层结构框架，对涉及到的产品开发活动进行合理的结构化。CMMI把流程分解为一个个关键过程域，是相对离散地来定义流程的，这决定了在CMMI体系下，产品开发流程的结构化不够。

### (4) 管理的范围不一样

IPD需要对所有的产品开发活动进行管理，横向上涉及市场、设计、测试、试制、制造、采购、服务、销售、财务各职能部门在产品开发中的活动，纵向上涉及决策、管理、执行三个层面。**CMMI**主要是面向研发部门的活动，如软件开发、系统集成、项目管理等。

### (5) 关注重点不一样

IPD不仅关注把事情做正确，同时也关注做正确的事情，所以IPD既强调执行的重要，也强调决策的重要。**CMMI**主要关注执行，即把事情做正确，而**CMMI**对如何执行好开发活动要求更规范、更细。


### (6) 人员管理不一样

IPD包括了对团队和个人的考评。**CMMI**则不包括人员管理的内容。

### IPD与CMMI的相同之处

IPD与CMMI有这么多的不同，但就对具体流程和活动进行管理而言，两者所依据的原则、方法和实践是相通的和一致的

**结论：企业在优化产品开发体系时，完全可以将两者融合，实施IPD+CMMI的解决方案。**

- 
- 6.1 个体软件过程PSP
  - 6.2 小组软件过程TSP
  - 6.3 集成产品开发模式IPD
  - 6.4 敏捷过程模型
  - 6.5 统一软件过程RUP

- 敏捷软件开发方法的核心是**以人为本，迭代、循序渐进的开发方法**。
- 敏捷软件开发方法认为，对项目最重要的影响因素是人，而不是过程和技术。不能把人当做由过程驱动的“可插拔替换的编程单元”，而要发挥人的能动性，建立紧密协作的、自组织的团队。



通过亲身实践以及帮助他人实践，揭示更好的软件开发方法。

通过这项工作，我们认为：

人和交互                      重于    过程和工具

可以工作的软件    重于    面面俱到的文档

客户合作                      重于    合同谈判

随时应对变化        重于    遵循计划

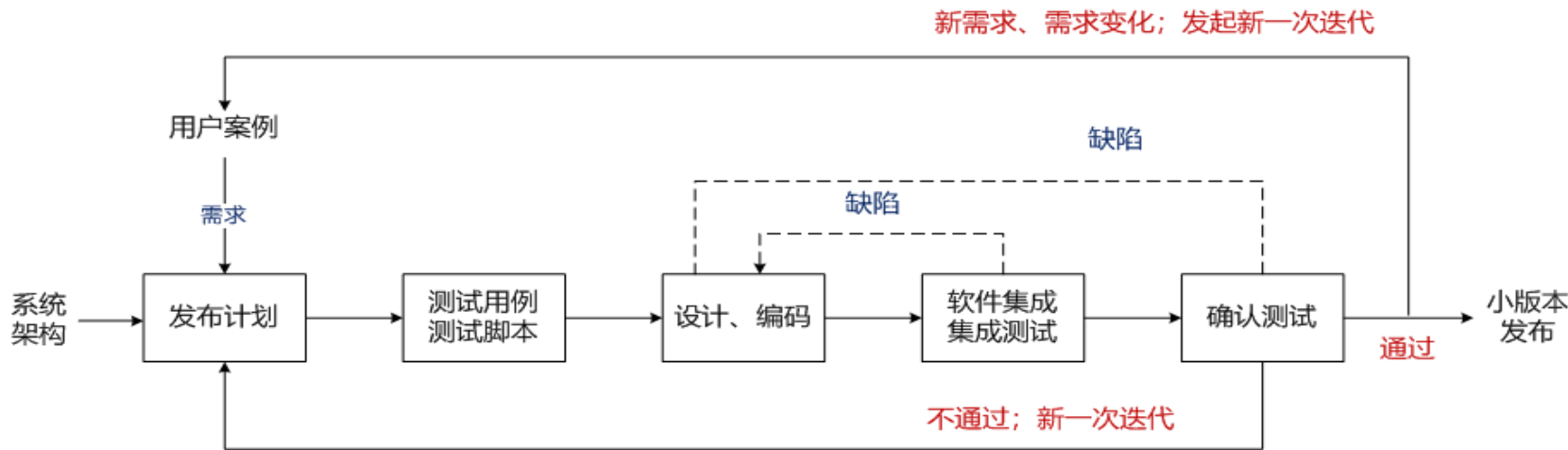
虽然右项也有其价值，但我们认为左项更加重要。

## 6.4 敏捷过程模型

## —— 极限编程

**极限编程XP** (Extreme Programming) 是最著名的敏捷开发方法，它由一系列简单的、互相依赖的最佳实践组成。

XP团队使用现场客户、特殊计划方法和持续测试来提供快速的反馈和全面的交流。这可以帮助团队最大化地发挥他们的价值。



极限编程XP的生命周期


- **沟通：**问题往往是开发人员与设计人员，设计人员和客户之间沟通不畅导致的。团队成员之间通过日常沟通，简单设计，测试，系统隐喻以及代码本身来沟通产品需求和系统设计。团队成员不是通过文档来交流，文档不是必须的。
- **反馈：**尽快获得用户的反馈，并且越详细越好，使得开发人员能够保证自己的成果符合用户的需要。

- **简单**：XP提倡简单的设计，简单的解决方案。应该尽量保持代码的简单，与其实现一个复杂的系统，不如设计一个能够满足目前需要的、简单的系统，因为你所考虑的情况可能**永远都不会发生**。
- **勇气**：XP鼓励一些有较高风险的良好做法。例如，它要求程序员尽可能频繁地重构代码，必须删除过时的代码，不解决技术难题就不罢休，等等。
- **团队**：XP 提倡团队合作，相互尊重。

- 1.现场客户 (On-site Customer)
- 2.计划游戏 (Planning Game)
- 3.系统隐喻 (System Metaphor)
- 4.简单设计 (Simple Design)
- 5.代码集体所有 (Collective Code Ownership)
- 6.结对编程 (Pair Programming)
- 7.测试驱动 (Test-driven)
- 8.小型发布 (Small Releases)
- 9.重构 (Refactoring)
- 10.持续集成 (Continuous integration)
- 11.每周40小时工作制 (40-hour Weeks)
- 12.代码规范 (Coding Standards)

XP适合规模小、进度紧、需求变化大、质量要求严的项目。

它希望以最高的效率和质量来解决用户目前的问题，以最大的灵活性和最小的代价来满足用户未来的需求，XP在平衡短期和长期利益之间做了巧妙的选择。

- 
- 6.1 个体软件过程PSP
  - 6.2 小组软件过程TSP
  - 6.3 集成产品开发模式IPD
  - 6.4 敏捷过程模型
  - 6.5 统一软件过程RUP

## 6.5 统一软件过程

统一软件过程RUP是IBM-Rational建立的、以用例驱动、以可执行的体系结构为中心、以迭代和增量方式推进、通过执行工作流程递增地产生结果的软件开发过程。

统一软件过程把UML统一建模语言有机的结合到一起，非常适合采用面向对象软件技术进行软件项目开发。



### ● 用例驱动

所有的软件开发都是**用户需求驱动**的。RUP采用用例来描述用户需求，同时提供一套方法把用例转化为设计的类图，进一步变成最终的程序代码。用例(Use Case)贯彻到整个软件生命周期。

- 需求分析中，客户或用户对Use Case进行描述；
- 系统分析和系统设计过程中，设计师对Use Case进行分析；
- 实现过程中，开发编程人员对Use Case进行实现；
- 测试过程中，测试人员对Use Case进行检验。

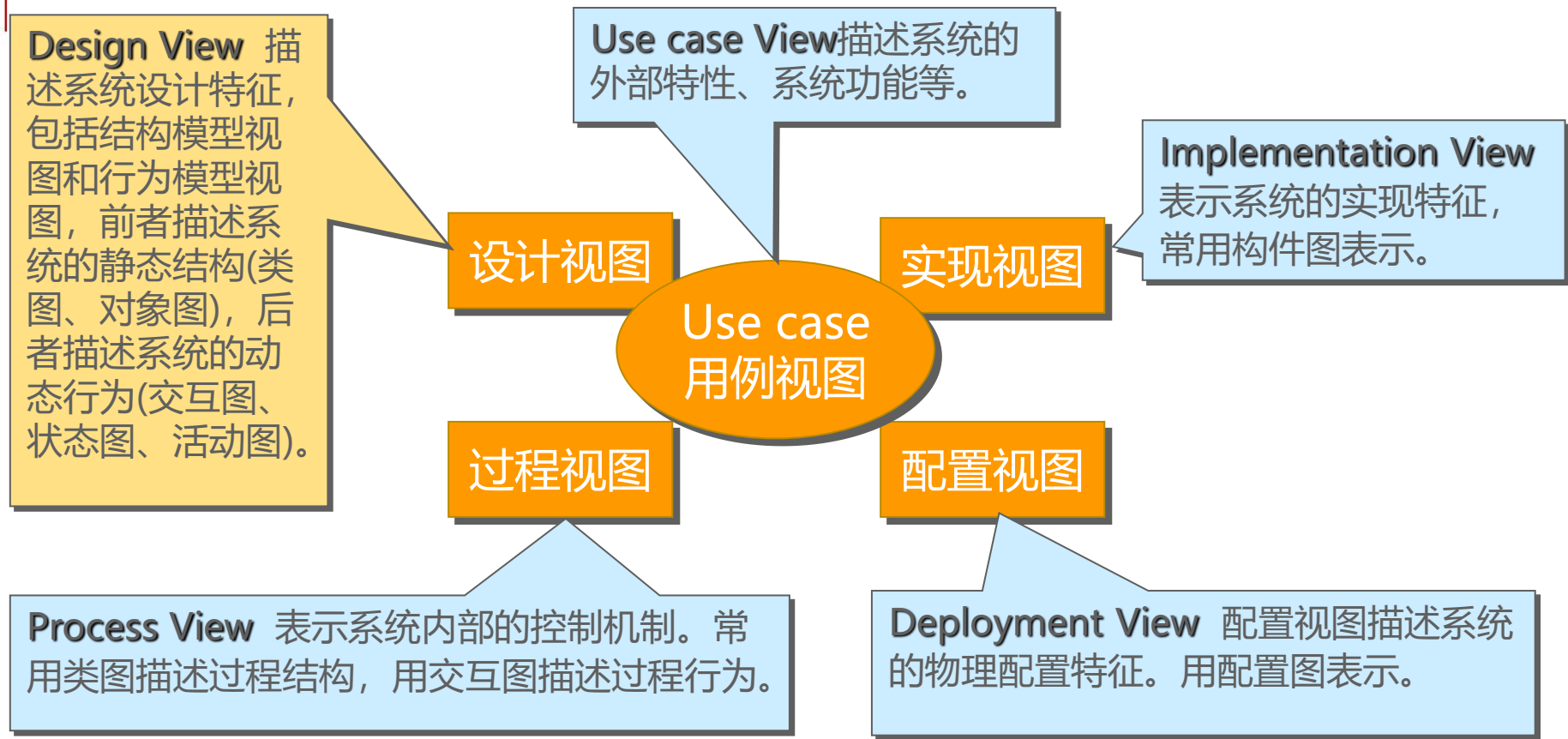
- 以架构为中心

软件架构是关于构成系统的元素、元素之间的交互、元素和元素之间地组成模式以及作用在这些组成模式上的约束等方面的描述。

由于在项目的开发过程中不同的开发人员所关心的角度是不一样的，因此软件的架构应该是一个多维的结构，RUP采用如下图所示的4+1视图模型，利用UML语言来描述软件的体系结构。

## 6.5 统一软件过程

### — 特点

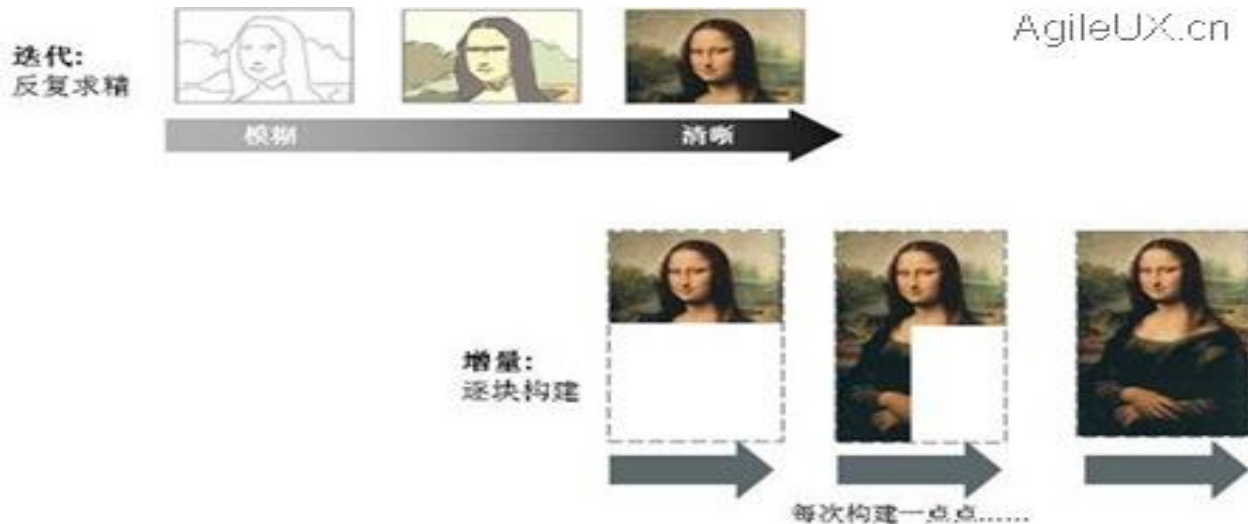


## 6.5 统一软件过程

### — 特点

- 采用迭代和增量模型

RUP采用**迭代和增量**的开发方式，在每次迭代中，只考虑系统的一部分需求，进行分析、设计、实现、测试、部署等过程，每次迭代是在已完成部分的基础上进行的，每次增加一些新的功能实现，以此进行下去，直至最后项目的完成。



统一软件过程开发模型按软件开发生命周期划分为四个阶段：初始阶段、细化阶段、构造阶段、产品化阶段（交付阶段）。每个阶段都可通过多次迭代来完成。

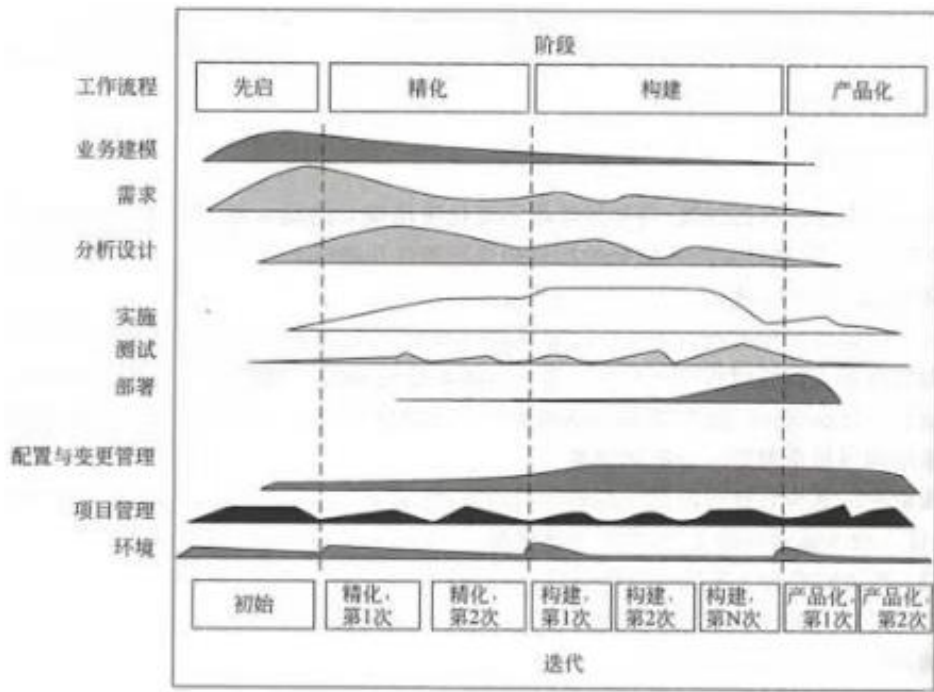
## 6.5 统一软件过程

### — 框架

RUP开发模型由软件生命周期（四个阶段）和RUP的核心工作流（活动）构成一个二维空间。

横轴表示项目的时间维，是对过程的动态描述，通过迭代式软件开发的周期、阶段、迭代和里程碑等动态信息表示；

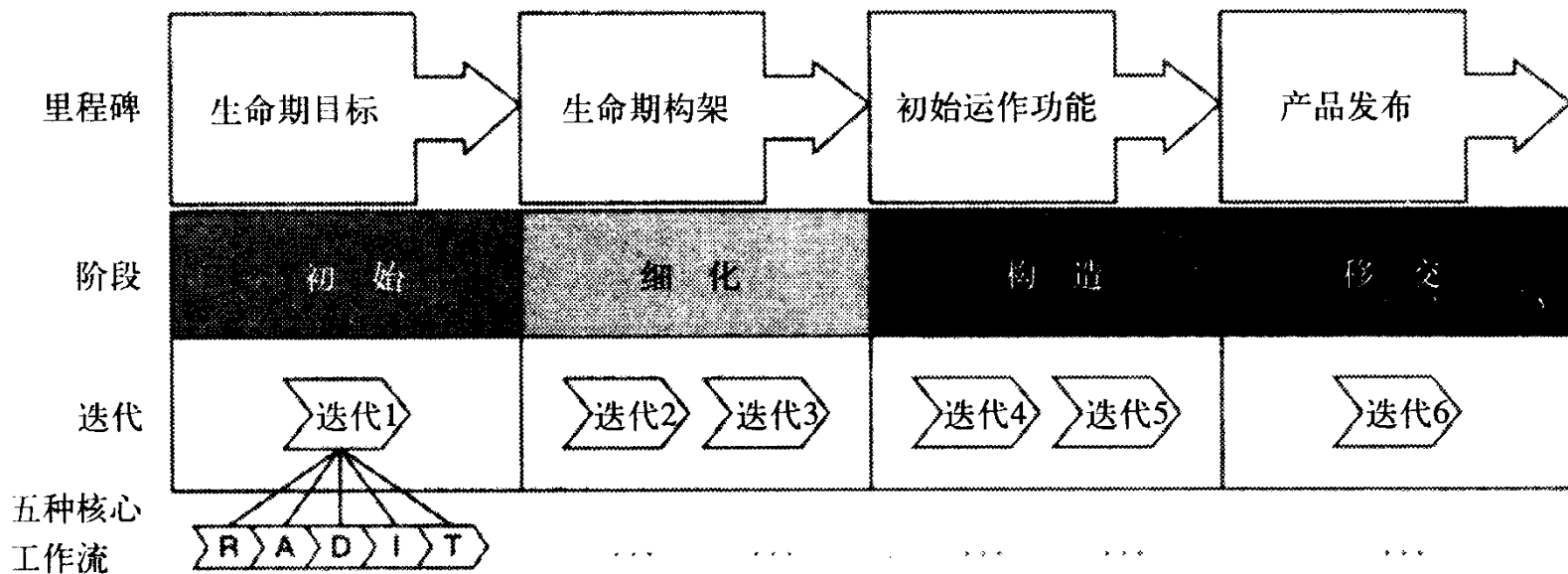
纵轴以内容来组织为自然的逻辑活动，是对过程的静态描述，通过过程的构件、活动、工作流、产物和角色等静态概念来描述系统；



## 6.5 统一软件过程

### — 工作阶段

RUP周期分为起始、细化、构建、移交阶段。每个阶段开始时都有特定的目标，结束时有里程碑。在每个阶段中存在一个或多个迭代。在每个迭代中，可以有多个工作流。



### 1、初始阶段

- (1) 阐述项目范围、核心要求、关键性质、主要限制等内容；
- (2) 初始的项目术语表；
- (3) 主要的用例模型（完成10-20%）和商业用例，将主要需求无二义性地表达出来；
- (4) 可执行的体系结构原型的主要要求；
- (5) 规模/工作量/资源/成本/进度估计、优先级、开发过程可信度等项目规划，明确各个阶段及其迭代；
- (6) 初始的风险评估。



### 2、细化阶段

- (1) 从低风险的运作到高成本、高风险运作的过渡，最关键；
- (2) 识别出所有的用例、角色及其主要描述，建立用例模型；
- (3) 识别出所有的需求，包括非功能性需求，确保需求稳定；
- (4) 可执行的体系结构原型及其描述，确保体系结构稳定；
- (5) 完整的项目计划，包括策略、规模、工作量、资源、成本、进度、迭代过程以及相应的评估准则，确保项目计划置信度；
- (6) 制订风险表以及防控措施，确保主要风险受控；
- (7) 相关共同利益者对项目用例、需求定义、项目计划、可执行的体系结构原型等取得一致理解。

### 3、构造阶段

构造阶段是执行过程，将初始阶段和细化阶段所建立的用例模型、体系结构转化为实实在在的产品，其成果是可以提交给最终用户使用的产品及其相关文档。

### 4、产品化阶段

也称交付阶段，把软件产品交付给用户。

一旦产品交付给最终用户，通常会产生新的要求，如客户报告的问题需要修正、功能增强而要开发新的版本等等。

RUP中有9个工作流，分为6个核心过程工作流（商业建模、需求、分析与设计、实现、测试、部署）；3个核心辅助工作流（设置与变更管理、项目管理、环境）。这些工作流在整个生命周期中一次又一次被访问。9个工作流在项目中轮流被使用，在每一次迭代中以不同的重点和强度重复。

### 1、商业建模 (Business Modeling)

商业建模 workflow 描述了如何为新的目标组织开发一个构想，并基于这个构想在商业用例模型和商业对象模型中定义组织的过程、角色和责任。

### 2、需求 (Requirements)

需求 workflow 的目标是描述系统应该做什么，并使开发人员和用户就这一描述达成共识。为了达到该目标，要对需要的功能和约束进行提取、组织、文档化，最重要的是理解系统所解决问题的定义和范围。

### 3、分析（Analysis）与设计（Design）

初始阶段的尾期，主要的分析任务开始了，分析工作主要集中在细化阶段，细化阶段的大部分活动是捕获需求，并进行需求分析，分析工作与需求捕获在很大程度上重叠。

设计活动以架构设计为中心，开展构架设计、设计用例、设计类和设计子系统设计

### 4、实现 (Implementation)

实现是把设计模型映射成可执行代码的过程，目的包括以层次化的子系统形式定义代码的组织结构，以组件的形式（源文件、二进制文件、可执行文件）实现类和对象，将开发出的组件作为单元进行测试以及集成由单个开发者（或小组）所产生的结果，使其成为可执行的系统。

### 5、测试 (Test)

测试 workflow 要验证对象间的交互作用，验证软件中所有组件的正确集成，检验所有的需求已被正确的实现，识别并确认缺陷在软件部署之前被提出并处理。

### 6、部署 (Deployment)

部署工作流的目的是成功的生成版本并将软件分发给最终用户。部署工作流描述了那些与确保软件产品对最终用户具有可用性相关的活动，包括软件打包、生成软件本身以外的产品、安装软件、为用户提供帮助。

### 7、配置和变更管理

配置和变更管理工作流提供了准则来管理演化系统中的多个变体，跟踪软件创建过程中的版本。工作流描述了如何管理并行开发、分布式开发、如何自动化创建工程。同时也阐述了对产品修改原因、时间、人员控制等。



### 8、项目管理

软件项目管理平衡各种可能产生冲突的目标，管理风险，克服各种约束并成功交付使用户满意的产品。

其目标包括：为项目的管理提供框架，为计划、人员配备、执行和监控项目提供实用的准则，为管理风险提供框架等。

### 9、环境 (Environment)

环境工作流的目的是向软件开发组织提供软件开发环境，包括过程和工具。环境工作流集中于配置项目过程中所需要的活动，同样也支持开发项目规范的活动，提供了逐步的指导手册并介绍了如何在组织中实现过程。

# 课后作业4

## 1、根据你的理解，如何做好项目计划？

(1) 估算项目范围、项目阶段、产品规模、工作量、成本等，为制定《项目计划》提供依据；

(2) 制定项目计划，以此作为管理项目的基础。包括：进度计划、项目资源计划、人员培训计划、干系人参与计划、风险管理计划及文档等产出物管理计划等；

(3) 对项目计划进行评审，以满足可用的资源并获得相关干系人的确认承诺。

# 课后作业4

## 2、根据你的理解，如何做好项目风险管理？

(1) 组织层面（EPG）收集日常项目和行业内常见的风险，确定来源并进行分类，然后定义风险的属性（如：发生概率、严重程度等）、最后制定风险的管理策略；

(2) 针对具体项目识别项目风险，定性分析风险的发生概念及对项目影响程序并行风险排序，针对每个已经识别和分析认为应该受控的风险制定风险管理计划和策略；

(3) 开展常态化的风险监控，一是按照风险管理计划跟踪已识别风险、当风险发生时触发风险应对方案；二是收集风险管理数据、评估风险管理活动、报告风险管理绩效。