《软件工程》

主讲教师：何彪

手机：15011879768

Email: biao_he@126.com

软 件 工 程

第1章 概论

内容摘要

- 计算机软件
- 软件工程
- 软件过程
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

内容摘要

- 计算机软件
- 软件工程
- 软件过程
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

计算机软件

- 计算机软件指计算机系统中的程序及其文档
  - ◆ 程序是计算任务的处理对象和处理规则的描述
    - 任务：以计算机为处理工具的任务都是计算任务
    - 处理对象：数据（如数据、文字、图形、图像、声音等，它们只是表示，而无含义）或信息（数据及有关的含义）。
    - 处理规则一般指处理的动作和步骤。程序必须装入计算机内才能工作。
  - ◆ 文档是为了便于了解程序所需的阐明性资料，文档一般是给人看的，不一定装入计算机

软件的发展

★ 1946-1956年

从计算机问世到实用的高级程序语言出现前

- Ø存储容量比较小，运算速度比较慢
- Ø采用个体工作方式,用低级语言编写程序
- Ø应用领域主要是以数值数据处理为主的科学计算，其特点是输入、输出量较小
- Ø衡量程序质量的标准主要是功效，即运行时间省、占用内存小
- Ø主要研究内容是科学计算程序、服务性程序和程序库，研究对象是顺序程序

★ 1956-1968年

从实用的高级程序语言出现到软件工程出现前

- Ø存储器容量大，外围设备得到迅速发展，出现了高级程序设计语言
- Ø应用领域包括数据处理（非数值数据），其特点是计算量不大，但输入、输出量却较大
- Ø高速主机与低速外围设备的矛盾突出，出现了操作系统、并发程序、数据库及其管理系统
- Ø20世纪60年代初提出了软件一词，开始认识到文档的重要性
- Ø研究高级程序设计语言、编译程序、操作系统、支持编程的工具及各种应用软件
- Ø工作方式逐步从个体方式转向合作方式
- Ø出现软件危机

★ 1968年-至今

从软件工程出现到现在

- 硬件向巨型机和微型机二个方向发展，出现了计算机网络，软件方面提出了软件工程，出现了“计算机辅助软件工程”（CASE）
- 计算机的应用领域渗透到各个业务领域，出现了嵌入式应用，其特点是受制于它所嵌入的宿主系统
- 开发方式逐步由个体合作方式转向工程方式
- 软件工程方面的研究主要包括软件开发模型、软件开发方法及技术、软件工具与环境、软件过程、软件自动化系统等
- 软件方面研究以智能化、自动化、集成化、并行化、以及自然化为标志的软件开发新技术

★ 计算机软件发展简史小结

在这几十年中，计算机软件经历了三个发展阶段:

- 程序设计阶段，约为46～56年代
- 程序系统阶段，约为56～68年代
- 软件工程阶段，约为68年代以后

请看表1.1软件发展的三个时期的情况.

表1.1 计算机软件发展的三个时期及其特点

特点\时期	程序设计	程序系统	软件工程
软件所指	程序	程序及说明书	程序、文档、数据
主要程序设计	汇编及机器语言	高级语言	软件语言*
软件工作范围	程序编写	包括设计和测试	软件生存期
需求者	程序设计者本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发团队或大中型软件开发机构
软件规模	小型	中小型	大中小型
决定质量的因素	个人程序技术	小组技术水平	管理水平

续：表1.1 计算机软件发展的三个时期及其特点

开发技术和手段	子程序 程序库	结构化 程序设计	数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络及分布式开发、面向对象技术
维护特征	程序设计者	开发小组	专职维护人员
硬件特征	价格高存储容量小工作可靠性差	降价、速度、容量及工作可靠性明显提高	向超高速、大容量、微型化及网络化方向发展
软件特征	完全不受重视	软件技术的发展不能满足需要，出现软件危机	开发技术有进步，但未获突破性进展，价高，未完全摆脱软件危机

表注：这里软件语言包括需求定义语言、软件功能语言、软件设计语言、程序设计语言等。

软件危机

在软件开发中遇到的问题找不到解决的办法，致使问题积累起来，形成了日益尖锐的矛盾，这些主要问题有：

（1）软件开发无计划性

（2）软件需求不充分

（3）软件开发过程无规范

（4）软件产品无评测手段

（5）许多软件项目不能满足客户的要求

（6）许多软件项目超出预算和时间安排

软件危机的表现

- 对软件开发成本和进度的估计常常很不正确
- 用户对“已完成的”软件系统不满意的现象经常发生
- 软件产品的质量往往靠不住
- 软件常常是不可维护的
- 软件通常没有适当的文档资料
- 软件成本在计算机系统总成本中所占的比例逐年上升
- 软件开发生产率提高的速度远远跟不上计算机应用迅速普及深入的趋势

软件危机的原因

- 软件是逻辑产品，开发进度、成本难以估计
- 缺乏或不完整、不一致的文档给维护带来困难
- 用户对软件需求的描述往往不够精确，有遗漏，有二义性、有错误
- 软件开发人员对需求的理解与用户的本来愿望有差异
- 大型软件项目需多人协同完成，缺乏管理经验
- 开发人员不能有效地、独立自主地处理大型软件的全部关系
- 缺乏有力的方法学和工具的支持
- 软件项目的特殊性和人类智力的局限性

克服软件危机的途径

- 消除错误的概念和做法
- 推广使用成功的开发技术和方法
- 使用软件工具和软件工程支持环境
- 加强软件管理

软件的特点

- 软件是一种逻辑实体，而不是有形的系统元件，其开发成本和进度难以准确地估算
- 软件是被开发的或被设计的，它没有明显的制造过程，一旦开发成功，只需复制即可，但其维护的工作量大
- 软件的使用没有硬件那样的机械磨损和老化问题

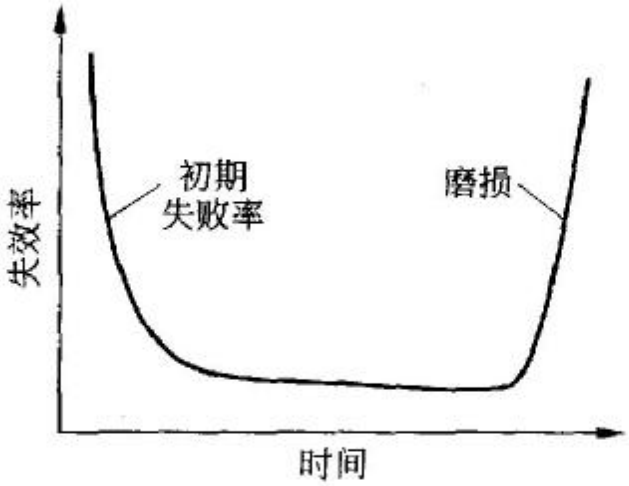


图 1.1　硬件的故障曲线

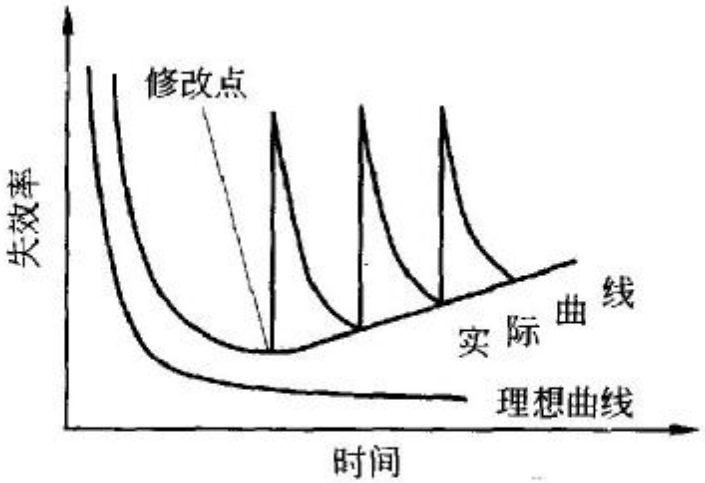
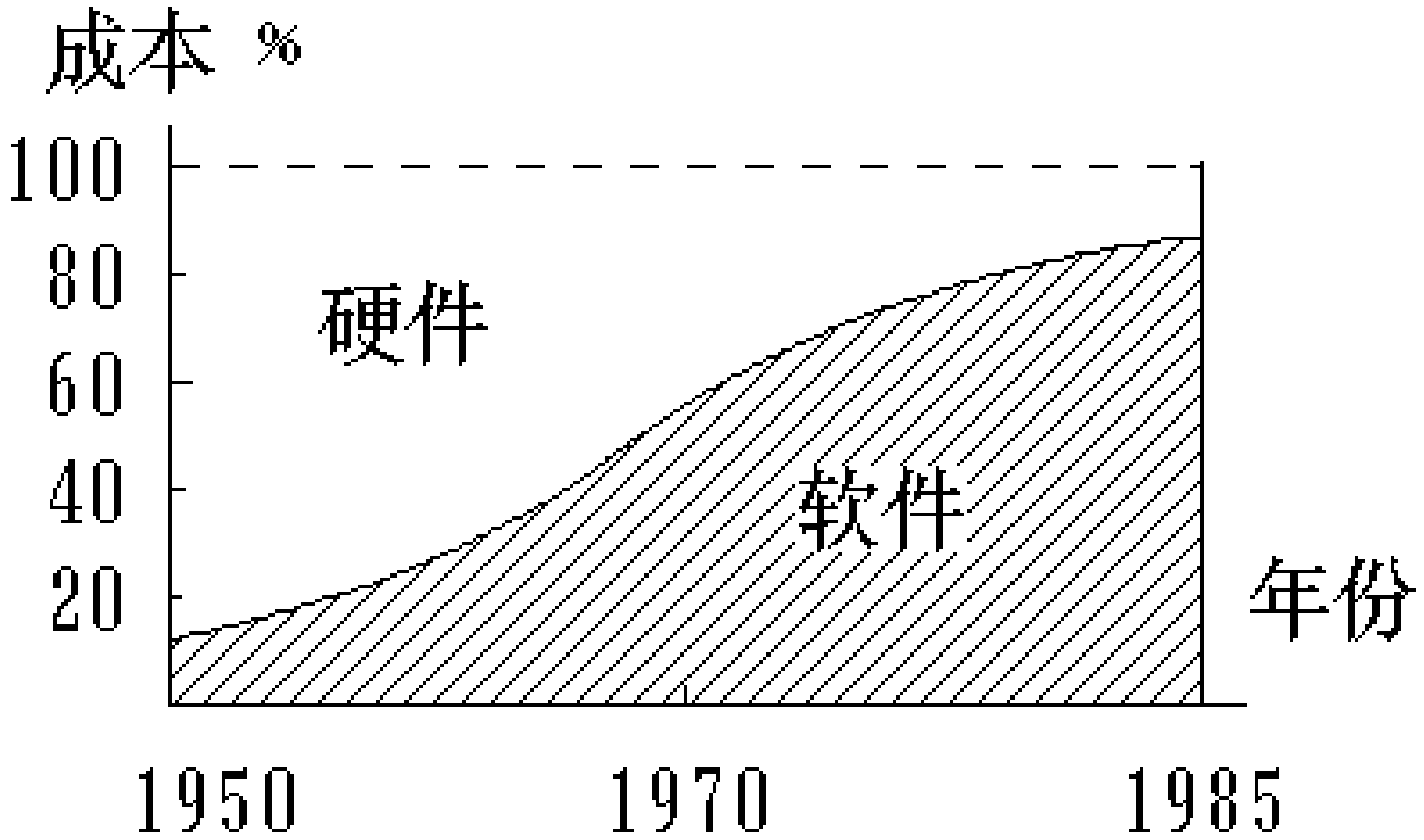


图 1.2　软件的故障曲线

其它特点:

- 软件的开发和运行常受到计算机硬件的限制，对计算机硬件有着不同程度的依赖性
- 软件的开发至今尚未完全实现自动化
- 软件成本相当昂贵
- 相当多的软件工作涉及到社会因素



软件的分类

- 系统软件：属于计算机系统中最靠近硬件的一层，其它软件一般都通过系统软件发挥作用，它与具体的应用领域无关。如操作系统、编译程序等。
- 支持软件：支持软件的开发和维护的软件。如数据库管理系统、网络软件、软件开发环境等。
- 应用软件：特定应用领域专用的软件。如实时软件、嵌入式软件、科学和工程计算软件、事务处理软件、人工智能软件等。

- 按软件工作方式划分:
  - § 实时处理软件
  - § 分时软件
  - § 交互式软件
  - § 批处理软件
- 按软件服务对象的范围划分:
  - § 项目软件
  - § 产品软件

- 按使用的频度进行划分:
  - § 一次使用
  - § 频繁使用
- 按软件失效的影响进行划分:
  - § 高可靠性软件
  - § 一般可靠性软件

软件语言
software language

软件语言是用于书写计算机软件的语言。

它主要包括:

需求定义语言

功能性语言

设计性语言

实现性语言（即程序设计语言）

文档语言

需求定义语言
requirements definition language

需求定义语言用来书写软件需求定义。

软件需求定义是软件功能需求和非功能需求的定义性描述。软件功能需求刻画软件“做什么”，软件非功能需求刻画诸如功能性限制、设计限制、环境描述、数据与通信规程及项目管理等

典型的需求定义语言有PSL语言（Problem Statement Language问题陈述语言）

功能性语言
functional language

功能性语言用来书写软件功能规约（functional specification）

软件功能规约是软件功能的严格而完整的陈述。通常它只刻画软件系统“做什么”的外部功能，而不涉及系统“如何做”的内部算法。

典型的功能性语言有广谱语言、Z语言。

设计性语言
design language

设计性语言用来书写软件设计规约（design specification）

软件设计规约是软件设计的严格而完整的陈述。一方面，它是软件功能归约的算法性细化，刻画软件“如何做”的内部算法，另一方面，它是软件实现的依据。

典型的设计性语言有PDL语言（Program Design Language）

实现性语言

实现性语言用来书写计算机程序

实现性语言也称编程语言或程序设计语言（programming language）

程序设计语言可按语言的级别、对使用者的要求、应用范围、使用方式、成分性质等多种角度进行分类

- 按语言级别分：低级语言和高级语言

低级语言是与特定计算机体系结构密切相关的程序设计语言，如机器语言、汇编语言。其特点是与机器有关，功效高，但使用复杂，开发费时，难维护。

高级语言是不反映特定计算机体系结构的程序设计语言，它的表示方法比低级语言更接近于待解问题的表示方法。其特点是在一定程度上与具体机器无关，易学、易用、易维护。但高级语言程序经编译后产生的目标程序的功效往往较低。

- 按用户要求分：过程式语言和非过程式语言

过程式语言（procedural language）是通过指明一列可执行的运算及运算次序来描述计算过程的程序设计语言。如FORTRAN、COBOL、C等。

非过程式语言（nonprocedural language ）是不显式指明处理过程细节的程序设计语言。在这种语言中尽量引进各种抽象度较高的非过程性描述手段，以期做到在程序中增加“做什么”的描述成分，减少“如何做”的细节描述。如第四代语言（4GL）、函数式语言、逻辑式语言。

也可称：命令式语言和申述式语言

命令式语言（imperative language）即过程式语言。

申述式语言（declarative language）是着重描述要处理什么，而非描述如何处理的语言。申述式语言程序是关于问题解的约束陈述，这些约束迫使含于实现中的算法处理机制生成一个解或一组解。如函数式语言、逻辑式语言。

函数式语言(functional programming language)中函数是构造程序的基本成分，它提供一些设施用于构造更为复杂的函数。程序人员根据提出的问题去定义求解函数（即主程序），其中可能包含一些辅助函数。如Lisp语言。

逻辑式语言(logic programming language)的基本运算单位是谓词。谓词定义了变元间的逻辑关系。例如，Prolog语言的基本形式是Horn子句，其程序围绕着某一主题的事实、规则和询问三类语句组成。这三类语句分别用来陈述事实、定义规则和提出问题。

·按应用范围分：通用语言和专用语言

通用语言指目标非单一的语言，如FORTRAN、COBOL、C等。

专用语言指目标单一的语言，如自动数控程序APT。

·按使用方式分：交互式语言和非交互式语言

交互式语言指具有反映人机交互作用的语言，如BASIC。

非交互式语言指不反映人机交互作用的语言，如FORTRAN、COBOL。

- 按成分性质分：顺序语言、并发语言、分布语言

顺序语言指只含顺序成分的语言，如FORTRAN、C。

并发语言指含有并发成分的语言，如Modula、Ada、并发Pascal。

分布语言指考虑到分布计算要求的语言，如Modula。

文档语言
documentation language

- 文档语言用来书写软件文档。

计算机软件文档是计算机开发、维护和使用过程的档案资料和对软件本身的阐述性资料。

通常用自然语言或半形式化语言书写。

内容摘要

- 计算机软件
- 软件工程
- 软件过程
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

软件工程定义

1968年NATO(北大西洋公约组织)会议上首次提出

- Fritz Bauer: 软件工程是为了经济地获得可靠的和能在实际机器上高效运行的软件而建立和使用的好的工程原则

[鲍威尔（Fritz Bauer）：“建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法”。]

- IEEE: 软件工程是（1）将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中；（2）（1）中所述方法的研究。

  [1983年IEEE给出的定义为：“软件工程是开发、运行、维护和修复软件的系统方法”。其中“软件”的定义为：计算机程序、方法、规则、相关的文档资料以及在计算机上运行时所需要的数据。]

- 计算机科学技术百科全书：软件工程是应用计算机科学、数学及管理科学等原理，以工程化的原则和方法制作软件的工程。

鲍姆(B.W.Boehm)曾为软件工程定义:

“运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料”。

这里对“设计”一词应有广义的理解，它应包括软件的需求分析和对软件进行修改时所进行的再设计活动。

- 费莱(Fairley)认为：“软件工程学是为了在成本限额以内按时完成开发任务和修改软件产品所需的系统生产和维护的技术和管理的学科”。

软件工程包括3个要素：

方法、工具和过程。

软件工程方法为软件工程开发提供了“如何做”的技术。它包括了多方面的任务、如项目计划与估算、软件系统需求分析、数据结构、系统总体结构的设计、算法的设计、编码、测试以及维护等。软件工程方法常采用某种特殊的语言或图形的表达方式及一套质量保证标准。

- 软件工程工具为软件工程方法提供自动的或半自动的软件支撑环境。目前，已经开发出了许多软件工具，能够支持上述的软件工程方法。有人已经把诸多软件工具集成起来，使得一种工具产生的信息可以为其它的工具所使用,这样建立起一种称之为计算机辅助软件工程（CASE）的软件开发支撑系统。

CASE将各种软件工具、开发机器和一个存放开发过程信息的工程数据库组合起来形成一个软件工程环境。

软件工程的过程则是将软件工程方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。

过程定义了方法使用的顺序、要求交付的文档资料、为保证质量和协调变更所需要的管理、及软件开发各个阶段完成的里程碑。

软件工程就是包含上述方法、工具和过程在内的一些步骤。

软件工程的框架

- 目标:

生产具有正确性、可用性以及价格合宜的产品

正确性反映软件产品实现相应功能规约的程度;

可用性反映软件的基本结构、实现及其文档为用户可用的程度;

价格合宜反映软件开发与运行的总代价满足用户要求的程度。

过程（Process）：

生产一个最终满足需求且达到工程目标的软件产品所需要的步骤

软件工程过程包括：开发过程、运作过程、维护过程、管理过程、支持过程、获取过程、供应过程、剪裁过程等

- 原则：

选取适宜的开发模型

采用合适的设计方法

提供高质量的工程支持

重视软件工程的管理

软件生存周期
（ software life cycle）

- 软件有一个孕育、诞生、成长、成熟、衰亡的生存过程。这个过程即为计算机软件的生存周期
- 软件生存周期大体可分为如下几个活动：计算机系统工程、需求分析、设计、编码、测试、运行和维护

- 计算机系统工程

Ø 计算机系统包括计算机硬件、软件、使用计算机系统的人、数据库、文档、规程等系统元素。

Ø 计算机系统工程的任务:

v 确定待开发软件的总体要求和范围，以及它与其它计算机系统元素之间的关系

v 进行成本估算，做出进度安排

v 进行可行性分析，即从经济、技术、法律等方面分析待开发的软件是否有可行的解决方案，并在若干个可行的解决方案中作出选择。

- 软件需求分析

Ø主要解决待开发软件要“做什么”的问题

Ø确定软件的功能、性能、数据、界面等要求，生成软件需求规约。

- 软件设计

Ø主要解决待开发软件“怎么做”的问题。

Ø软件设计通常可分为系统设计（也称概要设计或总体设计）和详细设计。

Ø系统设计的任务是设计软件系统的体系结构，包括软件系统的组成成分、各成分的功能和接口、成分间的连接和通信，同时设计全局数据结构;

Ø详细设计的任务是设计各个组成成分的实现细节，包括局部数据结构和算法等。

- 编码

  用某种程序设计语言，将设计的结果转换为可执行的程序代码。

- 测试

  发现并纠正软件中的错误和缺陷。测试主要包括单元测试、集成测试、确认测试和系统测试。

- 运行和维护

  在软件运行期间，当发现了软件中潜藏的错误或需要增加新的功能或使软件适应外界环境的变化等情况出现时对软件进行修改。

内容摘要

- 计算机软件
- 软件工程
- 软件过程—P54-113
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

软件过程

- 软件过程是软件生存周期中的一系列相关的过程。过程是活动的集合，活动是任务的集合。
- 软件过程有三层含义:
  - Ø个体含义，即指软件产品或系统在生存周期中的某一类活动的集合，如软件开发过程，软件管理过程等;
  - Ø整体含义，即指软件产品或系统在所有上述含义下的软件过程的总体;
  - Ø工程含义，即指解决软件过程的工程，它应用软件工程的原则、方法来构造软件过程模型，并结合软件产品的具体要求进行实例化，以及在用户环境下的运作，以此进一步提高软件生产率，降低成本。

ISO12207软件生存周期过程

----P9

- ISO/IEC 12207标准把软件生存周期中可以开展的活动分为5个基本过程，8个支持过程和4个组织过程。（见P9-表1.1）。每一个过程划分为一组活动，每项活动进一步划分为一组任务。

一、基本（primary）过程（5个）供各当事方在软件生存周期期间使用。包括:

（1）获取（acquisition）过程：确定需方和组织向供方获取系统、软件或软件服务的活动。

（2）供应（supply）过程：确定供方和组织向需方提供系统、软件或软件服务的活动。

（3）开发（development）过程：确定开发者和组织定义并开发软件的活动。

（4）运作（operation）过程：确定操作者和组织在规定的环境中为其用户提供运行计算机系统服务的活动。

（5）维护（maintenance）过程：确定维护者和组织提供维护软件服务的活动。

二、支持（supporting）过程（8个）用于支持其他过程，它有助于软件项目的成功和质量提高。包括:

（1）文档编制（documentation）过程: 确定记录生存周期过程产生的信息所需的活动。

（2）配置管理（configuration management）过程: 确定配置管理活动。

（3）质量保证（quality assurance）过程: 确定客观地保证软件和过程符合规定的要求以及已建立的计划所需的活动。

（4）验证（verification）过程：根据软件项目要求，按不同深度确定验证软件所需的活动。

（5）确认（validation）过程：确定确认软件所需的活动。

（6）联合评审（joint review）过程：确定评价一项活动的状态和产品所需的活动。

（7）审计（audit）过程：确定为判断符合要求、计划和合同所需的活动。

（8）问题解决（problem resolution）过程：确定一个用于分析和解决问题的过程。

三、组织（organizational）过程（4个）用于软件组织建立和实现构成相关生存周期的基础结构和人事制度，并不断改进这种结构和过程。包括:

（1）管理（management）过程: 确定生存周期过程中的基本管理活动。

（2）基础设施（infrastructure）过程: 确定建立生存周期过程基础结构的基本活动。

（3）改进（improvement）过程: 确定一个组织为建立、测量、控制和改进其生存周期过程所需开展的基本活动。

（4）培训（training）过程: 确定提供经适当培训的人员所需的活动。

ISO/IEC 12207为软件生存周期过程建立了一个公共框架，它提供了一组标准的过程、活动和任务。对于一个软件项目，可根据其具体情况对标准的过程、活动和任务进行剪裁，即删除不适用的过程、活动和任务。

ISO/IEC 12207标准的附录A中的剪裁（tailoring）过程规定了在针对该标准进行剪裁时所需要的基本活动（包括：明确项目环境；请求输入；选择过程、活动和任务；把剪裁决定和理由写成文档），剪裁过程的输出是：剪裁决定和理由记录。

附录B就剪裁要点提供简要说明，并列出一些关键要素，可以根据这些要素作出剪裁决定。

能力成熟度模型—P12

CMM和CMMI的简介

1、CMM模型简单描述:

CMM（Capability Maturity Model---即能力成熟度模型），是美国卡耐基梅隆大学软件工程研究所（SEI -Software Engineering Institute ）在美国国防部资助下于二十世纪八十年代末建立的，用于评价软件机构的软件过程能力成熟度的模型。此模型在建立和发展之初，主要目的在于提供一种评价软件承接方能力的方法，为大型软件项目的招投标活动提供一种全面而客观的评审依据。而发展到后来，又同时被软件组织用于改进其软件过程。

在CMM认证方面，中国有19家通过CMM5/CMMI5级的软件企业，名单如下（2011年3月3日资料）：

[印度通过CMM5级认证的企业高达40家（全世界共有52家）。--（ 2008-05-的资料）]

中国通过CMM5级的企业，共9家：

1、摩托罗拉中国软件中心，2、沈阳东软股份有限公司，3、大连海辉科技股份有限公司，4、华为印度研究所，5、大连华信计算机技术有限公司，6、惠普中国软件研发中心，

7、毕博全球开发中心，8、北京用友软件工程有限公司，9、埃森哲全球信息技术中心。

中国通过CMMI5级的企业，共10家：

1、新宇科技(上海)集团，2、 塔塔信息技术(上海)有限公司杭州分公司，3、恩益禧-中科院软件研究所有限公司，4、北京软通动力信息技术有限公司，5、南京富士通南大软件技术有限公司，6、新电信息科技（苏州）有限公司，7、华微软件有限公司，8、普天信息技术研究院，9、上海宝信软件股份有限公司和亚信科技（中国）有限公司，10、亚信科技（中国）有限公司。

软件过程成熟度等级

CMM提供了一个成熟度等级框架： 1级-初始级、 2级-可重复级、 3级-已定义级、 4级-已管理级和5级-优化级。

1.初始（initial）级:

软件过程的特点是无秩序的，甚至是混乱的。几乎没有什么过程是经过妥善定义的，成功往往依赖于个人或小组的努力。

2.可重复（repeatable）级:

建立了基本的项目管理过程来跟踪成本、进度和功能特性。制定了必要的过程纪律，能重复早先类似应用项目取得的成功。

3.已定义（defined）级:

已将管理和工程活动两方面的软件过程文档化、标准化，并综合成该机构的标准软件过程。所有项目均使用经批准、剪裁的标准软件过程来开发和维护软件。

4.已管理（managed）级:

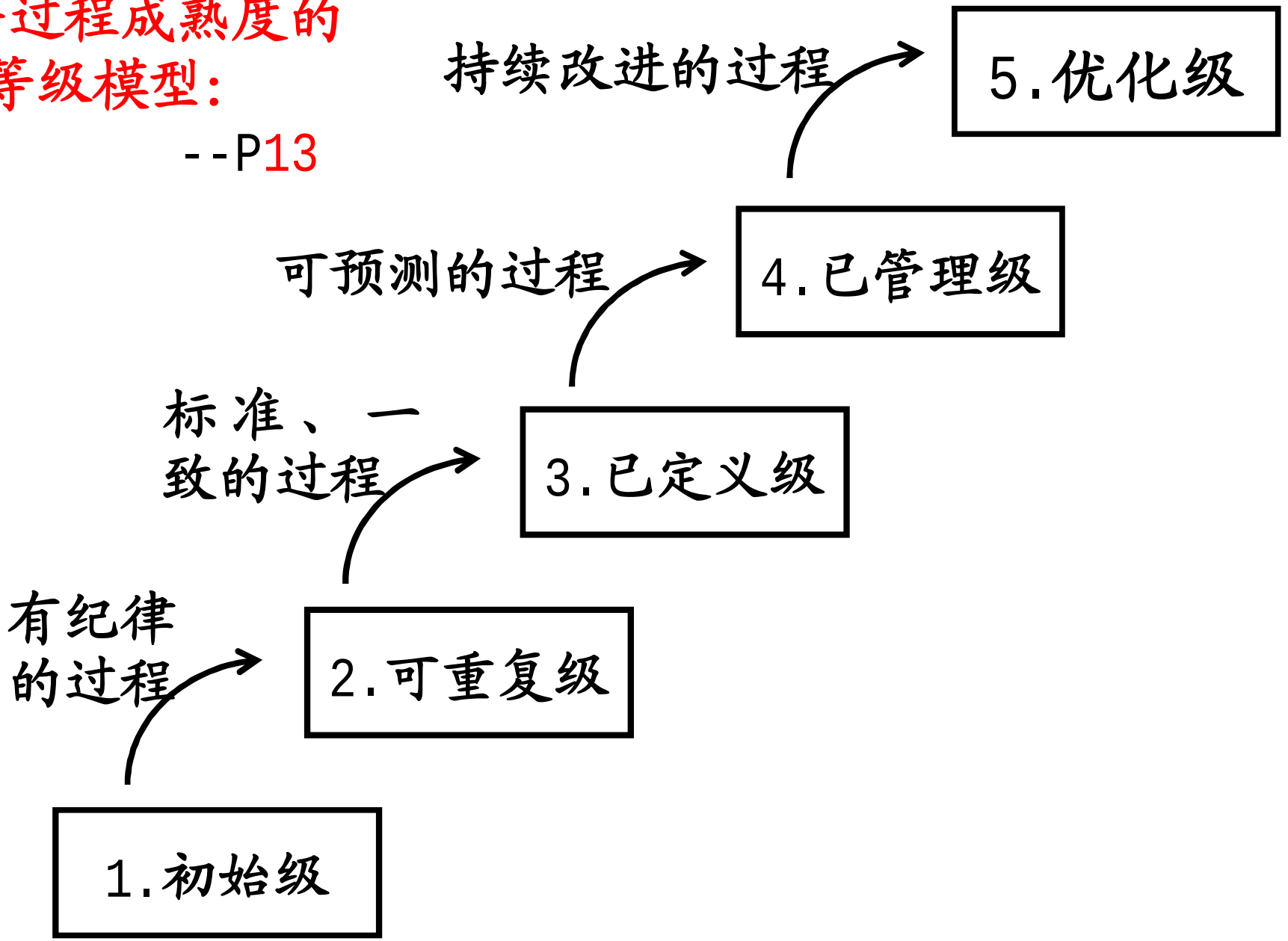
收集对软件过程和产品质量的详细度量值，对软件过程和产品都有定量的理解和控制。

5.优化（optimizing）级:

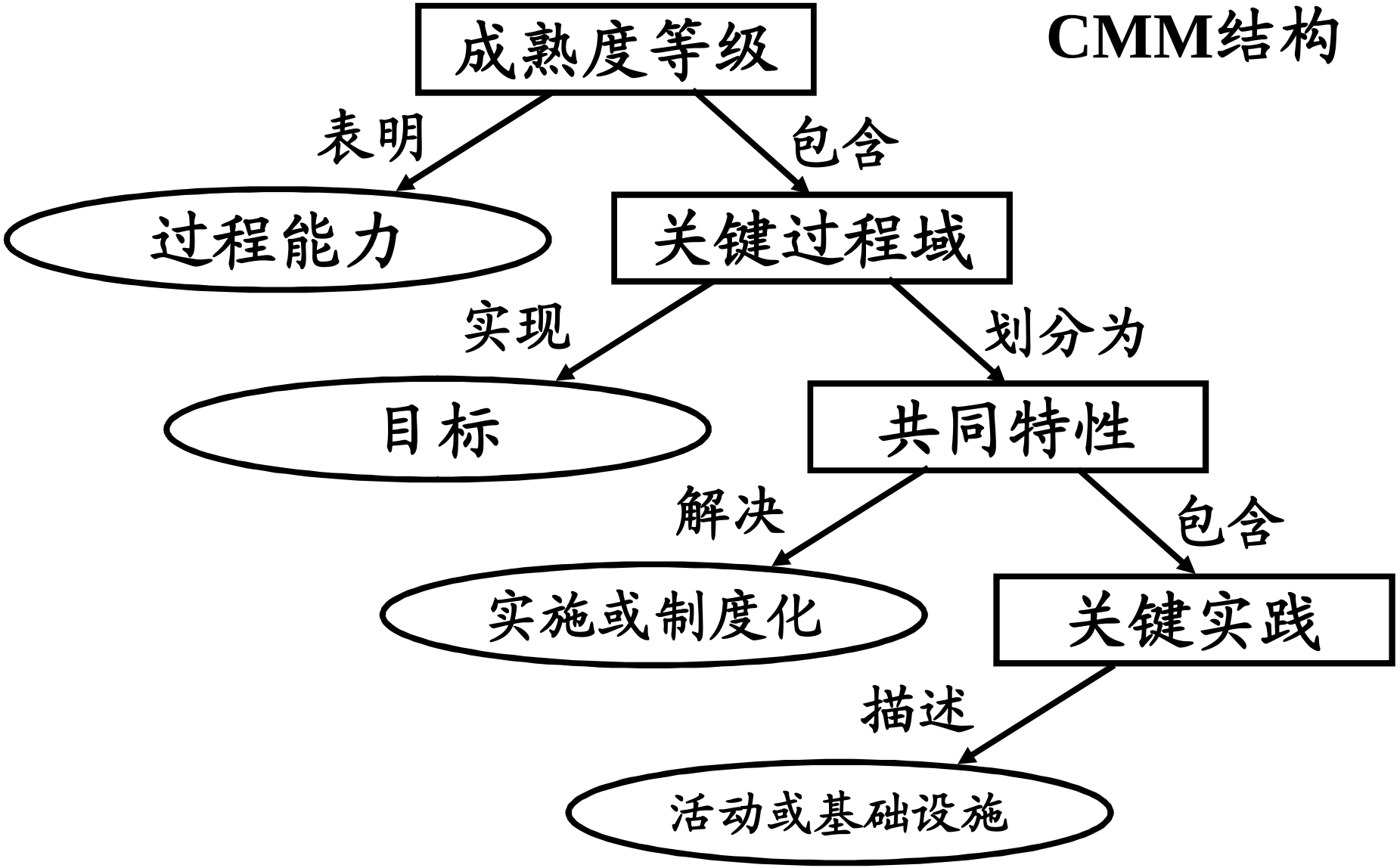
整个组织关注软件过程改进的持续性、预见及增强自身，防止缺陷及问题的发生。过程的量化反馈和先进的新思想、新技术促使过程不断改进。

软件过程成熟度的5个等级模型：

--P13



能力成熟度模型的结构—P14



2、CMMI模型简单描述:

CMMI(Capability Maturity Model Integration--能力成熟模型集成)是由美国软件工程研究所（SEI）正在开发的一种过程改进模型，由SEI发布了CMMI最新标准文件V1.2---即CMMI V1.2）。该模型的出现是过程改进领域的一次重要变化，会对今后软件组织的过程改进工作产生巨大影响。CMMI产品组决定尽量将CMMI与新的国际标准(包括:15015504、15012207、15015288等)兼容，为使CMMI尽早成为一个国际标准而努力。

CMMI与以前的过程改进模型CMM相比，是一个质的飞跃。基于SEI在过程改进方面的主导地位、CMM模型的广泛应用已成为事实上的国际标准，CMMI有可能成为新一代的过程改进标准的代表，并在软件开发工业化中发挥重要作用。

CMMI的出现，不仅仅是将多种模型结合起来，更重要的是，CMMI是能力成熟模型发展史上的一次重要变革，CMMI提供了更广泛的过程改进范围环境。CMMI之前的CMM能力成熟模型，都是偏向于对某一单独规范(如:软件工程或系统工程)过程的改进，而CMMI模型并不考虑规范本身的局限性，将重点放于产品或服务的开发和维护过程的改进之上。因此，有可能会对过程改进领域中的所有相关规范产生巨大影响。

软件组织的成熟与不成熟

1. 不成熟的软件组织

- 软件过程一般并不预先计划，而是在项目进行中由实际工作人员及管理员临时计划
- 有时，即使软件过程已计划好，仍不按计划执行
- 没有一个客观的基准来判断产品质量，或解决产品和过程中的问题
- 对软件过程步骤如何影响软件质量，一无所知，产品质量得不到保证。而且，一些提高质量的环节，如检查、测试等经常由于要赶进度而减少或取消
- 产品在交付前，对客户来说，一切都是不可见的

- 没有长远目标，管理员通常只关注解决任何当前的危机
- 由于没有实事求是地估计进度、预算，因此他们经常超支、超时。当最后期限临近，他们往往在功能性和质量上妥协，或以加班加点方式赶进度

2. 成熟的软件组织

- 具有全面而充分的组织和管理软件开发和维护过程的能力
- 管理员监视软件产品的质量以及生产这些产品的过程
- 制定了一系列客观基准来判别产品质量，并分析产品和过程中的问题
- 进度和预算可以按照以前积累的经验来制定，结果可行。预期的成本、进度、功能与性能和质量都能实现，并达到目的
- 能准确及时地向工作人员通报实际软件过程，并按照计划有规则地(前后一致，不互相矛盾)工作
- 凡规定的过程都编成文档

- 软件过程和实际工作方法相吻合。必要时，过程定义会及时更新，通过测试，或者通过成本-效益分析来改进过程。
- 全体人员普遍地、积极地参与改进软件过程的活动。在组织内部的各项目中，每人在软件过程中的职责都十分清晰而明确，每人各守其责，协同工作，有条不紊，甚至能预见和防范问题的发生。

能力成熟度模型的结构

- 成熟度等级表明了一个软件组织的过程能力的水平。除初始级外，每个成熟度等级都包含若干个关键过程域（Key Process Area，简称KPA）
- 达到某个成熟度级别，该级别（以及较低级别）的所有关键过程域都必须得到满足，并且过程必须实现制度化。

- CMM提供了18个关键过程域（P14-表1.2），每个关键过程域都有一组对改进过程能力非常重要的目标，并确定了一组相应的关键实践
- 目标说明了每一个关键过程域的范围、界限和意义。
- 关键实践描述了建立一个过程能力必须完成的活动和必须具备的基础设施，完成了这些关键实践就达到了相应关键过程域的目标，该关键过程域也就得到了满足。
- 每个关键过程域的关键实践都是按照五个共同特性（执行约定，执行能力，执行活动，测量和分析，验证实现）进行组织的，主要解决关键实践的实施或制度化问题。

• 共同特性将描述关键过程域的关键实践组织起来。共同特性是一些属性，指明一个关键过程域的执行和规范化是否有效、可重复和可持续。共有5个共同特性：执行约定，执行能力，执行活动，测量和分析，验证实现。

Ø**执行约定**：

执行约定描述机构为确保过程的建立和持续而必须采取的一些措施。典型内容包括建立机构策略和领导关系。

Ø**执行能力**:

执行能力描述了项目或机构完整地实现软件过程所必须有的先决条件。典型内容包括资源、机构结构和培训。

Ø**执行活动**:

执行活动描述了执行一个关键过程域所必需的活动、任务和规程。典型内容包括制定计划和规程、执行和跟踪以及必要时采取纠正措施。

Ø**测量和分析**:

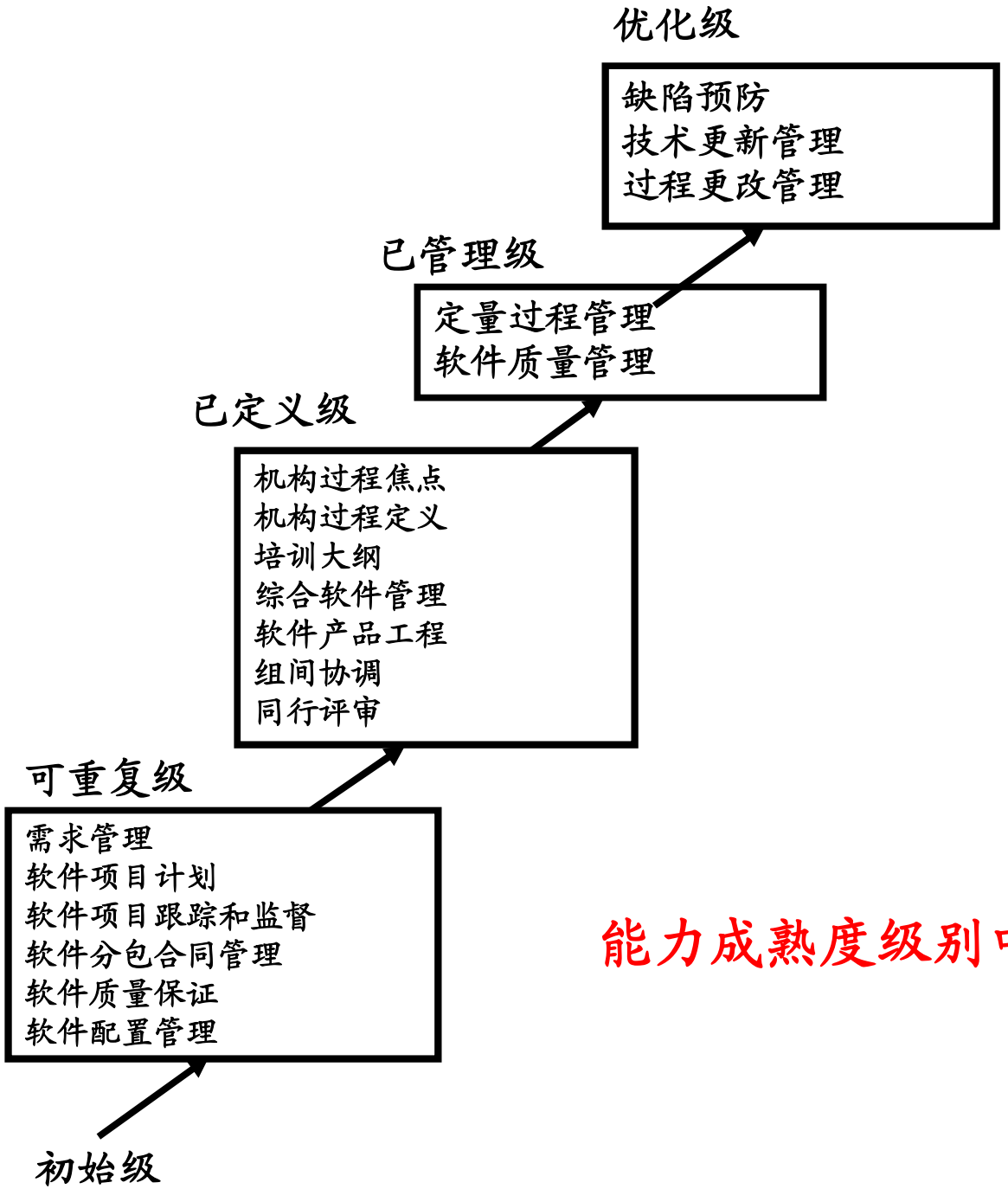
测量和分析描述了为确定与过程有关的状态所需的基本测量实践。这些测量可用来控制和改进过程。典型内容包括可能采用的测量实例。

Ø**验证实现**:

验证实现描述了为确保执行的活动与已建立的过程一致所采取的步骤。典型内容包括管理部门和软件质量保证组实施的评审和审核。

在执行活动这个共同特性中的实践描述了建立一个过程能力所必须完成的活动。所有其他实践共同形成了一个使机构能将执行活动中描述的实践进行规范化的基础。

各关键过程域的详细描述，参见《能力成熟度模型（CMM）：软件过程改进指南》，卡耐基梅隆大学软件工程研究所编著，刘孟仁等译，电子工业出版社出版。



能力成熟度级别中的关键过程域

关键过程域实例

机构过程焦点

第3级的关键过程域：已定义级

机构过程焦点的目的是，为能改进机构整体软件过程能力的软件过程活动建立机构的职责。

机构过程焦点包括，建立和维护机构软件过程和项目软件过程的默契关系，并协调有关评估、开发、维护和改进这些过程的活动。

机构提供长期的约定和资源，以协调现在和将来的软件项目的软件过程的开发和维护。该项工作由某个小组实施，例如软件工程过程组。它负责机构的软件过程活动，特别是负责开发和维护机构标准软件过程和相关过程资源（如在机构过程定义关键过程域中说明的），并协调软件项目的过程活动。

目标

目标1：机构内部软件过程的制定和改进活动协调一致。

目标2：相对于过程标准，所使用的软件过程的优势和薄弱环节标识清楚。

目标3：机构级的过程开发和改进活动有计划。

执行约定

约定1：机构遵循书面的管理策略，协调整个机构范围内的软件过程开发和改进活动。

该策略一般规定：

1. 建立一个小组，负责机构级的软件过程活动，使这些活动与各项目协调一致。

2. 定期评估项目所使用的软件过程，以确定其优势和薄弱环节。

3. 对机构标准软件过程进行合理地剪裁，以得到项目使用的软件过程。

关于机构标准软件过程，参见综合软件管理关键过程域的活动1。

4. 每个项目的软件过程、工具和方法的改进和其他有用信息，可用于其他项目。

约定2：上级管理部门倡导和支持机构的软件过程开发和改进活动。

上级管理部门：

1. 向机构成员和负责人说明有关软件过程活动的约定。

2. 制定资金、人员配备和其他资源的长期计划和约定。

3. 制定管理和执行有关软件过程开发和改进活动的策略。

约定3：上级管理部门监督机构的软件过程开发和改进活动。

l. 确保机构标准软件过程满足企业目标和策略。

2. 提出关于软件过程开发和改进活动优先次序的建议。

3. 参与制定软件过程开发和改进计划。

a. 上级管理部门与更高层人员和负责人共同协调软件过程需求及问题

b. 上级管理部门与该机构负责人进行协调，以获得负责人和机构成员的支持和参与

执行能力

能力1：有一个负责机构的软件过程活动的小组。

一个小组是一些部门、负责人和人员的组合，负责一组任务和活动。小组的规模可以不同，既可以是单个兼职的人，也可以是多个来自不同部门的兼职人员，也可以由几个专职人员组成。组成小组时考虑的因素包括：分派的任务和活动、项目规模、机构结构和机构文化。某些小组，如软件质量保证组，集中关注项目活动;而其他一些小组，例如软件工程过程组，集中关注机构范围内的活动。

1. 条件可能时，小组成员以专职工作的软件专业人员为核心，并尽可能有其他的兼职人员支持。

该小组最一般的例子是软件工程过程组（SEPG）。

2. 小组成员中有软件工程及软件相关科目的代表。

软件工程及软件相关科目的实例有:

- 软件需求分析
- 软件设计
- 程序编码
- 软件测试
- 软件配置管理
- 软件质量保证

能力2：为实施机构的软件过程活动提供了充足的资源和资金。

1. 委派在特定领域具有特长的人员支持该小组。

   特定领域的实例有：

   - 软件重用
   - 计算机辅助软件工程技术(CASE)
   - 测量
   - 培训课程开设

2. 有支持该机构软件过程活动的工具。

   支持工具的实例有：

   - 统计分析工具
   - 桌面出版工具
   - 数据库管理系统
   - 过程建模工具

能力3：负责机构软件过程活动的小组成员接受过实施这些活动所需的培训。

培训的实例有：

- **软件工程实践**
- **过程控制技术**
- **机构过程变动管理**
- **软件过程计划、管理和监督**
- **技术转变**

参见培训大纲关键过程域

能力4：软件工程组和其他软件相关小组的成员接受过机构软件过程活动及其在这些活动中的任务方面的定向培训。

参见培训大纲关键过程域

执行活动

活动1：定期评估软件过程，并根据评估结果制定行动计划。

评估一般每隔一年、一年半至三年进行一次。评估可针对机构中所使用的所有软件过程，也可通过对过程和项目进行抽样评估。评估机构软件过程能力的方法实例之一是SEI软件过程评估方法。

行动计划标识：

- 涉及哪些评估结果
- 针对评估结果实施更改软件过程的准则
- 负责实施更改的小组或个人

活动2：机构制定和维护它的软件过程开发和改进活动的计划。

该计划：

1. 以软件过程评估后的行动计划和其他的机构过程改进倡议为基础。
2. 确定要实施的活动及实施这些活动的进度。
3. 确定负责这些活动的小组和个人。
4. 确定所需的资源，包括人员配备和工具。
5. 初始发布和有大改动时通过同行评审。

   参见同行评审关键过程域。
6. 机构的软件负责人和上级负责人评审认可。

活动3：在机构级协调关于机构和项目的软件过程的开发和改进活动。

涉及的软件过程有：

1. 机构标准软件过程。

关于机构标准过程，参见机构过程定义关键过程域的活动1和活动2。

2. 项目定义的软件过程。

关于项目定义的软件过程。参见综合软件管理关键过程域的活动1和活动2。

活动4：在机构级协调有关软件过程数据库的使用。

机构的软件过程数据库用来收集机构和项目的软件过程以及生成的软件产品的信息。

关于机构的软件过程数据库，参见机构过程定义关键过程域的活动5。

活动5：监控和评价机构中限制使用的新过程、方法和工具。合适时，推广到机构的其他部分。

活动6：在机构内协调机构和项目的软件过程的培训。

1. 制定有关机构和项目软件过程的专题培训计划。

2. 合适时，培训由负责机构软件过程活动的小组（如软件工程过程组）或培训小组准备和实施。

参见培训大纲关键过程域。

活动7：向与实施软件过程有关的小组通报机构和项目中软件过程开发和改进活动的情况。

通报方式的实例有：

- 过程电子公告板
- 过程咨询委员会
- 工作小组
- 信息交流会
- 调查
- 过程改进组
- 日常讨论

测量和分析

测量：测量机构的软件过程开发和改进活动的状态

测量的实例有：

- 机构在过程评估、开发和改进活动中已完成的工作、工作量和耗费的资金，与计划相比较
- 每次软件过程的评估结果，与以前的评估结果和建议相比较

验证实现

验证：上级管理部门定期评审软件过程开发和改进活动。

上级管理部门实施定期评审的主要目的是适当地、及时地掌握软件过程活动。在满足机构需求的前提下，只要有适当的机制来报告异常情况，评审的时间间隔就尽可能长些。

1. 对照计划，评审有关开发和改进软件过程活动的进展和状态。

2. 讨论低层不能解决的冲突和问题。

3. 指定和评审行动措施，并跟踪到关闭。

4. 编写评审的总结报告，并分发给相关的小组和个人。

能力成熟度模型集成CMMI
Capability Maturity Model Integration

- CMM的成功导致了各种模型的衍生，每一种模型都探讨了某一特定领域中的过程改进问题
  - Ø SW-CMM：适用于软件开发
  - Ø SE-CMM：系统工程能力成熟度模型
  - Ø SA-CMM：适用于软件获取
  - Ø SECAM：系统工程能力评估模型
  - Ø People CMM：讨论软件组织吸引、开发、激励、组织和留住人才的能力
  - Ø EIA/IS 731：替代SW-CMM和SECAM
  - Ø IPD-CMM：适用于集成化产品开发
  - Ø FAA-iCMM：集成了SE-CMM、 SA-CMM、 SW-CMM

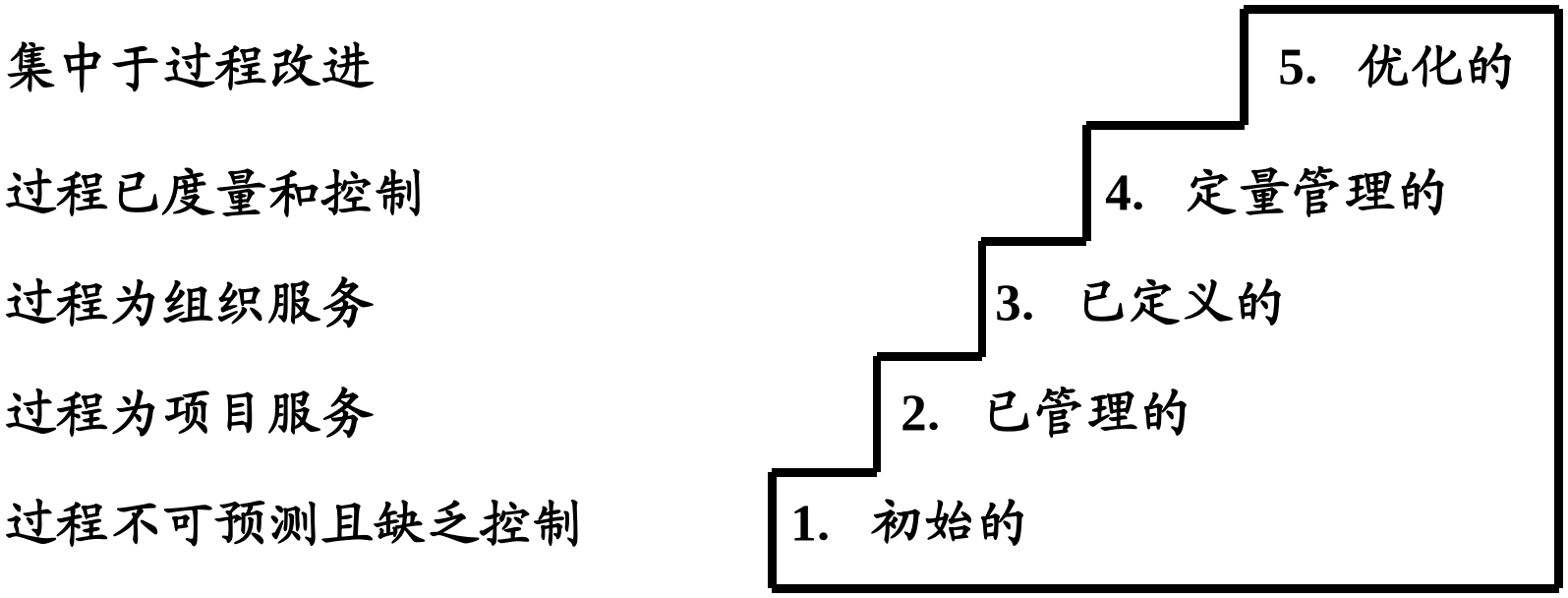
- Ø 相应的国际标准： ISO/IEC 12207（软件生存周期过程）、ISO/IEC 15288（系统生存周期过程）、ISO/IEC 15504（软件过程评估）

- 模型的繁衍导致模型框架、术语等方面的矛盾和不一致
- 包含在当代工程中各种各样的学科和工程是密切交叉在一起的，应用不同模型时效率低下且容易混淆，常常要付出极其昂贵的代价
- 美国国防部、美国国防工业委员会和SEI/CMU于1998年启动CMMI项目，希望CMMI是若干过程模型的综合和改进，是支持多个工程学科和领域的系统的、一致的过程改进框架，能适应现代工程的特点和需要，能提高过程的质量和工作效率

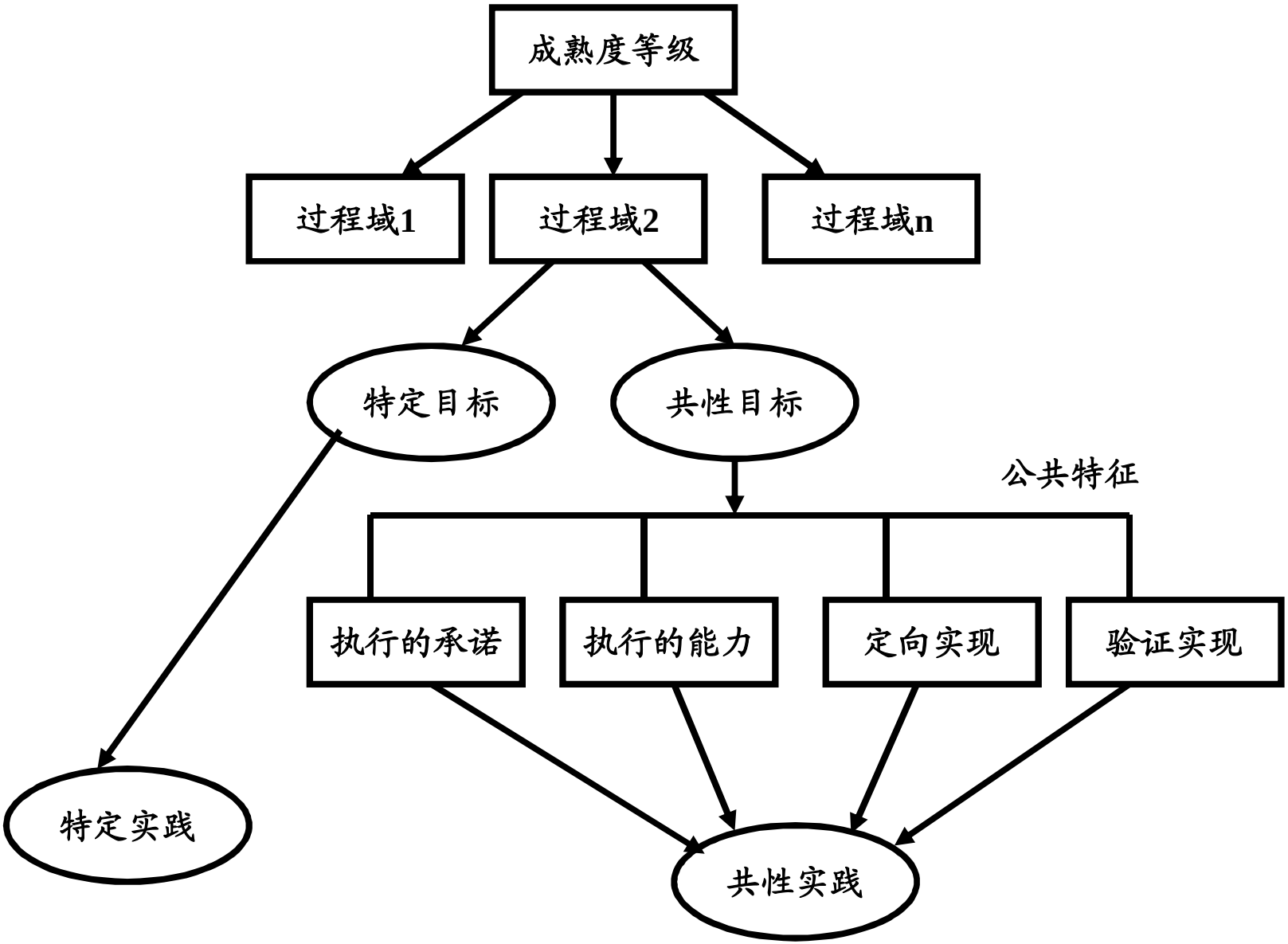
- 2000年发布第一个CMMI模型CMMI-SE/SW/IPPD V1.0：集成了SW-CMM、EIA/IS 731、IPD CMM V0.98
- 2002年1月发布CMMI-SE/SW/IPPD V1.1，美国国防工业委员会在第一届CMMI国际研讨会上宣布，CMMI V1.1将至少稳定五年不变
- CMMI模型为每个学科的组合都提供两种表示法：阶段式模型和连续式模型

阶段式模型

- 阶段式模型的结构类同于软件CMM，它关注组织的成熟度，其成熟度等级如下图所示



阶段式成熟度等级



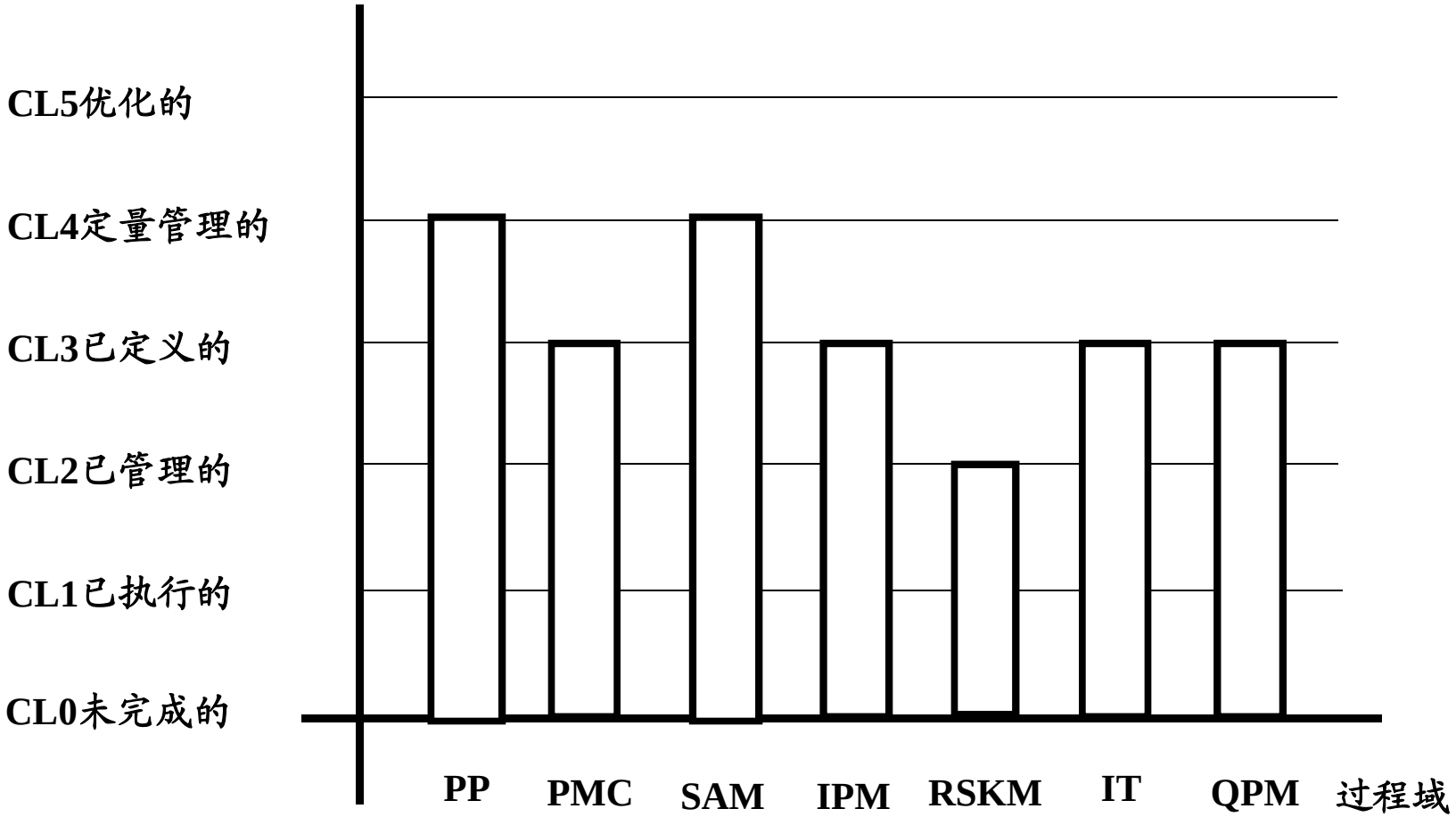
阶段式模型的成熟度等级结构—P16图1.6

CMMI V1.1的24个过程域的分组如下： P16-表1.3

成熟度等级	过程域
已管理的	需求管理REQM，项目计划PP，项目监督和控制PMC，供应商合同管理SAM，度量和分析MA，过程和产品质量保证PPQA，配置管理CM
已定义的	需求开发RD，技术解决方案TS，产品集成PI，验证VER，确认VAL，组织级过程焦点OPF，组织级过程定义OPD，组织级培训OT，集成化项目管理IPM，风险管理RSKM，集成化建组IT，决策分析和解决方案DAR，组织级集成环境OEI
定量管理的	组织级过程性能OPP，项目定量管理QPM
优化的	组织级改革和实施OID，因果分析和解决方案CAR

连续式模型

- 连续式模型关注每个过程域的能力，一个组织对不同的过程域可以达到不同的过程域能力等级（Capability level，CL）。
- CMMI中包括六个过程域能力等级，等级号为0～5。能力等级表明了单个过程域中组织执行的好坏程度。
- 允许组织对连续式模型的过程域进行剪裁，也允许对不同的过程域采用不同的能力等级
- 下图给出了某组织的过程域能力等级



能力等级特征示意图

- 能力等级包括共性目标及相关的共性实践，这些实践在过程域内被添加到特定目标和实践中。当组织满足过程域的特定目标和共性目标时，就说该组织达到了那个过程域的能力等级。
- 能力等级2～5的名字与成熟度等级2～5同名，但含义不同。能力等级可以独立地应用于任何单独的过程域，任何一个能力等级都必须满足比它等级低的能力等级的所有准则，各能力等级的含义简述如下：（P17 ）

- CL0 未完成的：过程域未执行或未达到CL1中定义的所有目标。
- CL1 已执行的：其共性目标是过程将可标识的输入工作产品转换成可标识的输出工作产品，以实现支持过程域的特定目标。
- CL2 已管理的：其共性目标集中于已管理的过程的制度化。根据组织级政策规定过程的运作将使用哪个过程，项目遵循已文档化的计划和过程描述，所有正在工作的人都有权使用足够的资源，所有工作任务和工作产品都被监控、控制和评审。

- CL3　已定义的：其共性目标集中于已定义的过程的制度化。 过程是按照组织的剪裁指南从组织的标准过程集中剪裁得到的，还必须收集过程资产和过程的度量，并用于将来对该过程的改进上。
- CL4　定量管理的：其共性目标集中于可定量管理的过程的制度化。使用测量和质量保证来控制和改进过程域，建立和使用关于质量和过程执行的定量目标作为管理准则。
- CL5　优化的：使用量化（统计学）手段改编和优化过程域，以对付客户要求的改变和持续改进计划中的过程域的功效。

- 连续式模型将24个过程域划分为过程管理、项目管理、工程和支持四个过程组： P17-表1.4

连续式分组	过程域
过程管理	组织级过程焦点OPF，组织级过程定义OPD，组织级培训OT，组织级过程性能OPP，组织级改革和实施OID
项目管理	项目计划PP，项目监督和控制PMC，供应商合同管理SAM，集成化项目管理IPM，风险管理RSKM，集成化建组IT，项目定量管理QPM
工　　程	需求管理REQM，需求开发RD，技术解决方案TS，产品集成PI，验证VER，确认VAL
支　　持	配置管理CM，过程和产品质量保证PPQA，度量和分析MA，决策分析和解决方案DAR，组织级集成环境OEI，因果分析和解决方案CAR

内容摘要

- 计算机软件
- 软件工程
- 软件过程
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

软件过程模型

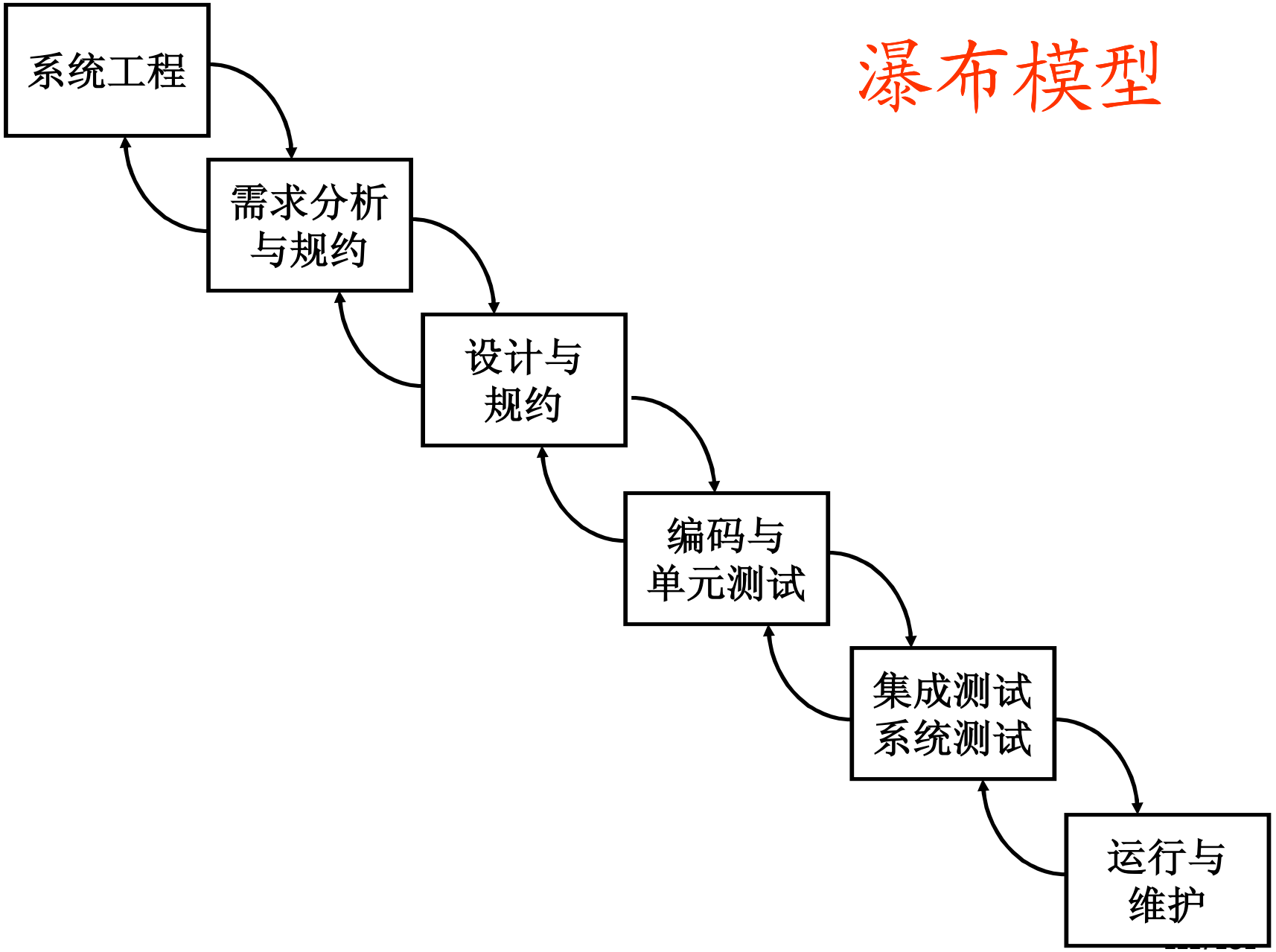
- 软件过程模型是软件开发全部
- 过程、活动和任务的结构框架
- 也称软件开发模型或软件生存
- 周期模型

软件过程模型

典型的软件过程模型有:

- **瀑布模型**（waterfall model）
- **演化模型**（evolutionary model）
- 增量模型（incremental model）
- 原型模型（prototyping model)
- 螺旋模型（spiral model）
- **喷泉模型**（water fountain model）
- **基于构件的开发模型**（component-based development model）
- **形式方法模型**（formal methods model）

瀑布模型



- 1970年W.Royce提出瀑布模型
- 特征
  - 接受上一阶段的结果作为本阶段的输入
  - 利用这一输入实施本阶段应完成的活动
  - 对本阶段的工作进行评审
  - 将本阶段的结果作为输出，传递给下一阶段
- 缺点
  - 缺乏灵活性，难以适应需求不明确或需求经常变化的软件开发
  - 开发早期存在的问题往往要到交付使用时才发现，维护代价大

演化模型

许多软件项目在开发早期对软件需求的认识是模糊的、不确定的，因此软件很难一次开发成功。

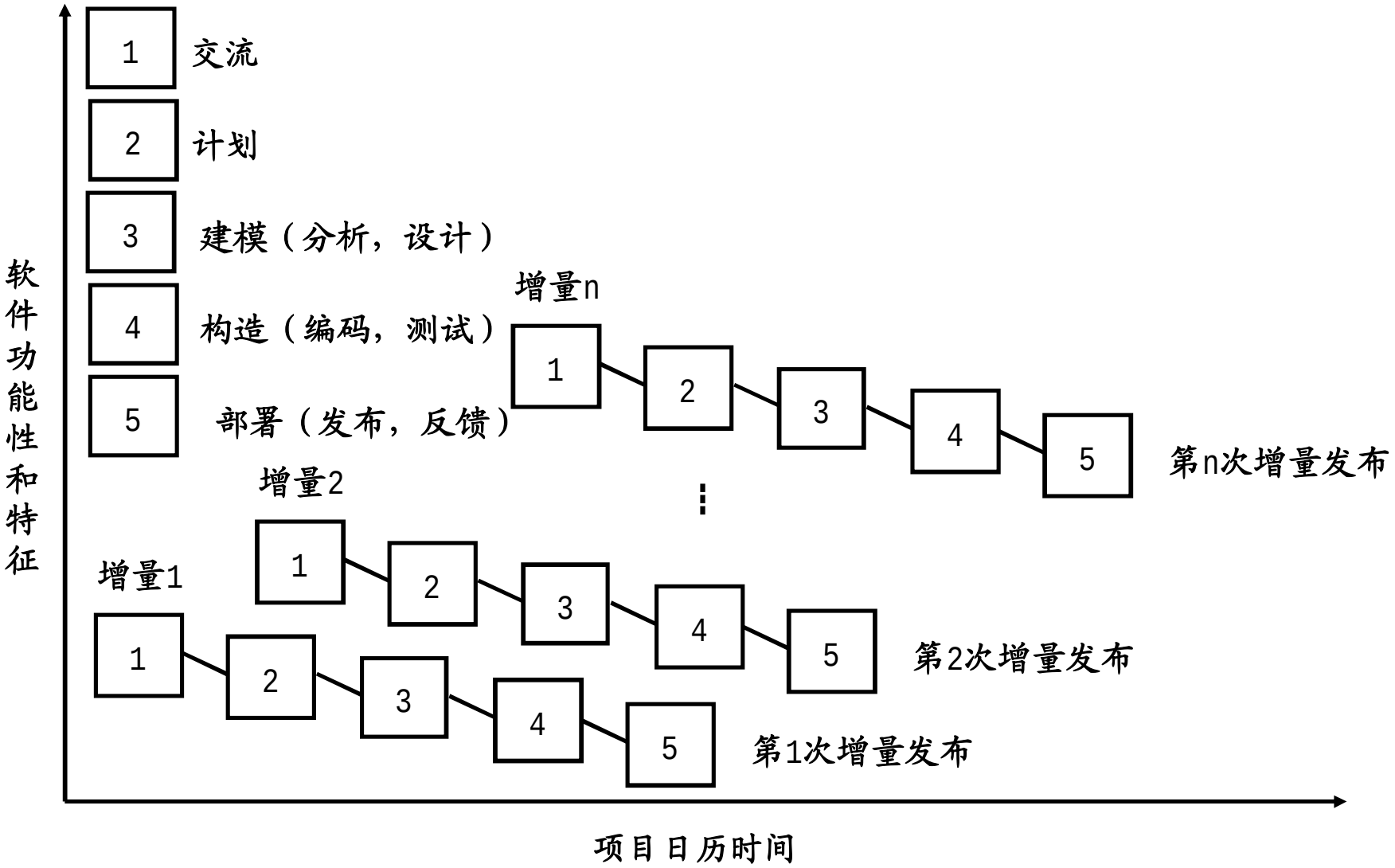
可以在获取了一组基本的需求后，通过快速分析构造出该软件的一个初始可运行版本，称之谓原型（prototype），然后根据用户在试用原型的过程中提出的意见和建议、或者增加新的需求，对原型进行改造，获得原型的新版本，重复这一过程，最终得到令客户满意的软件产品。

演化模型的开发过程就是从构造初始的原型出发，逐步将其演化成最终软件产品的过程。

演化模型适用于对软件需求缺乏准确认识的情况。

典型的演化模型有：增量模型、原型模型、螺旋模型。

增量模型



- 增量模型将软件的开发过程分成若干个日程时间交错的线性序列，每个线性序列产生软件的一个可发布的“增量”版本，后一个版本是对前一版本的修改和补充，重复增量发布的过程，直至产生最终的完善产品。
- 增量模型融合了瀑布模型的基本成分（重复地应用）和演化模型的迭代特征
- 增量模型强调每一个增量都发布一个可运行的产品

增量模型特别适用于:

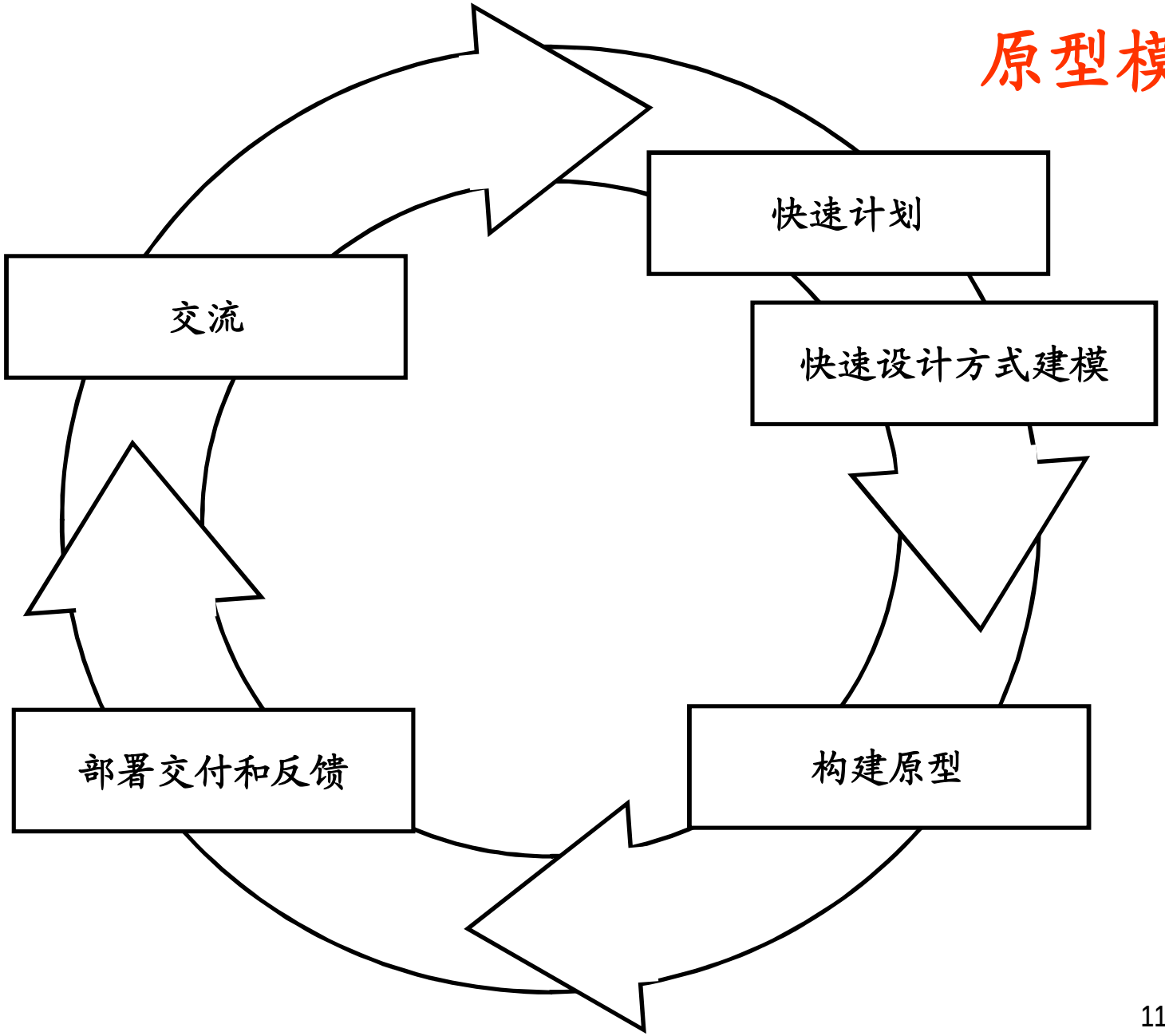
- 需求经常变化的软件开发
- 市场急需而开发人员和资金不能在设定的市场期限之前实现一个完善的产品的软件开发

增量模型能有计划地管理技术风险,如早期增量版本中避免采用尚未成熟的技术

原型模型

- 原型（prototype）是预期系统的一个可执行版本，它反映了系统性质（如功能、计算结果等）的一个选定的子集。一个原型不必满足目标软件的所有约束，其目的是能快速、低成本地构建原型。
- 原型方法从软件工程师与客户的交流开始，其目的是定义软件的总体目标，标识需求。然后快速制订原型开发的计划，确定原型的目标和范围，采用快速设计的方式对其建模，并构建原型。
- 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。

原型模型



原型的类型:

- Ø 探索型（exploratory prototyping）

  其目的是要弄清目标系统的要求，确定所希望的特性，并探讨多种方案的可行性

- Ø 实验型（experimental prototyping）

  其目的是验证方案或算法的合理性，它是在大规模开发和实现前，用于考核方案是否合适，规格说明是否可靠。

- Ø 演化型（evolutionary prototyping）

  其目的是将原型作为目标系统的一部分，通过对原型的多次改进，逐步将原型演化成最终的目标系统。

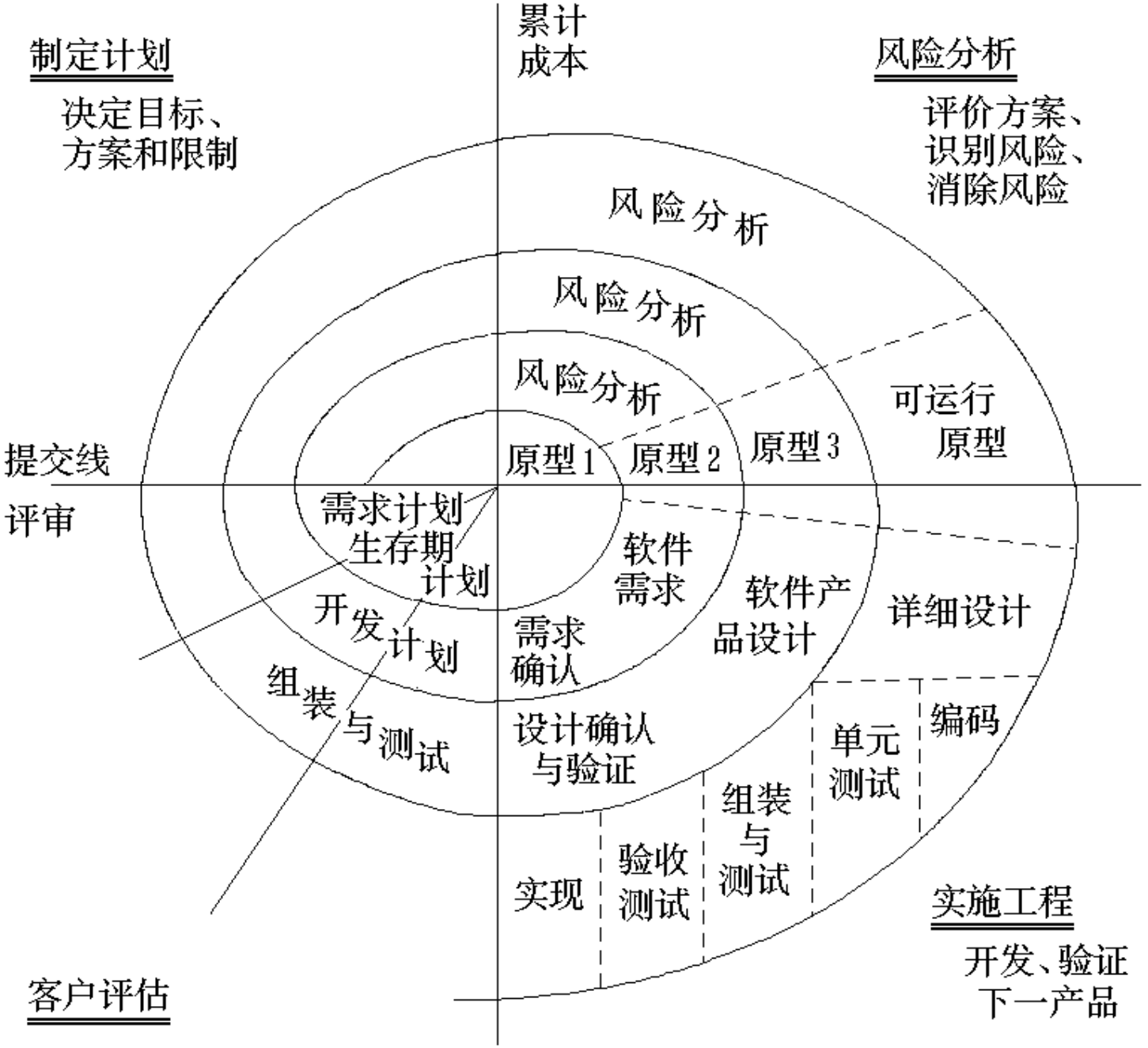
- 原型的使用策略:
  - Ø 废弃（throw away）策略

    主要用于探索型和实验型原型的开发。这些原型关注于目标系统的某些特性，而不是全部特性，开发这些原型时通常不考虑与探索或实验目的无关的功能、质量、结构等因素，这种原型通常被废丢，然后根据探索或实验的结果用良好的结构和设计思想重新设计目标系统。
  - Ø 追加（add on）策略

    主要用于演化型原型的开发。这种原型通常是实现了目标系统中已明确定义的特性的一个子集，通过对它的不断修改和扩充，逐步追加新的要求，最后使其演化成最终的目标系统。
- 原型可作为单独的过程模型使用，它也常被作为一种方法或实现技术应用于其它的过程模型中。

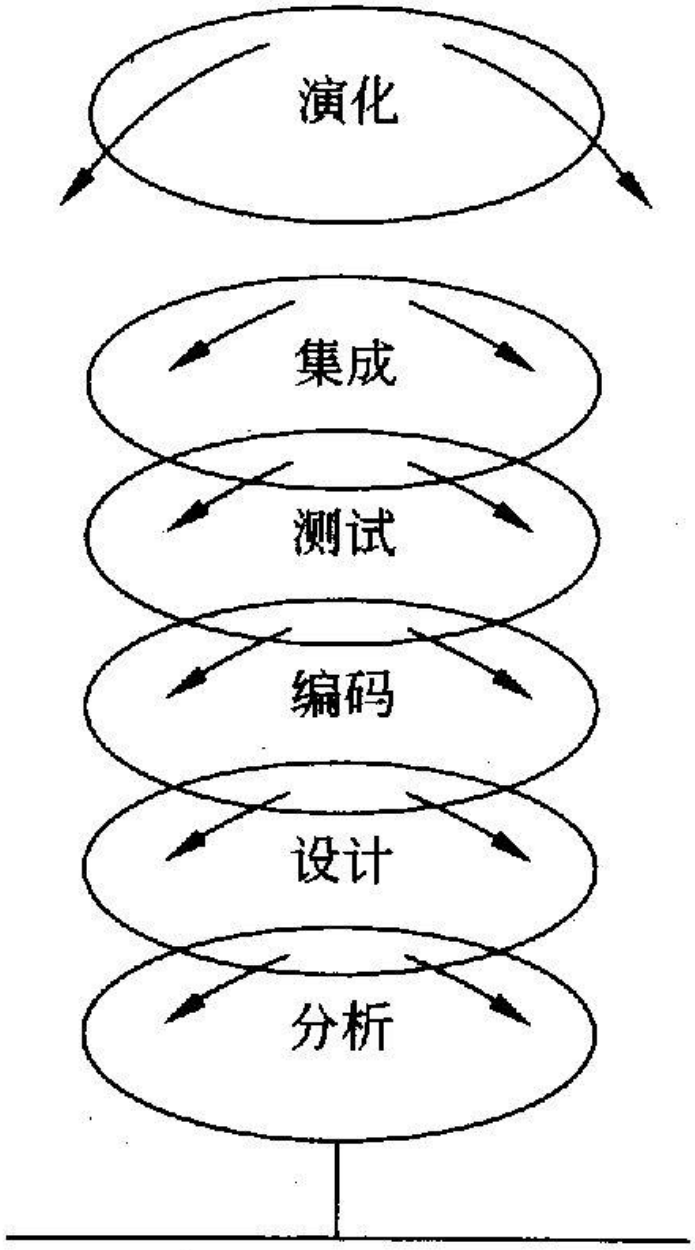
螺旋模型

- B.Boehm于1988年提出
- 是瀑布模型和演化模型的结合，并增加了风险分析
- 螺旋模型沿着螺线旋转，在四个象限上分别表达四个方面的活动，即：
  - Ø制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件
  - Ø风险分析：评价所选的方案，识别风险，消除风险
  - Ø工程实施：实施软件开发，验证工作产品
  - Ø客户评估：评价开发工作，提出修正建议



- 螺旋模型出现了一些变种，它可以有3到6个任务区域。
- 螺旋模型指引的软件项目开发沿着螺线自内向外旋转，每旋转一圈，表示开发出一个更为完善的新软件版本。
- 如果发现风险太大，开发者和客户无法承受，则项目就可能因此而终止。
- 多数情况下沿着螺线的活动会继续下去，自内向外，逐步延伸，最终得到所期望的系统。

喷泉模型



喷泉模型是一种支持面向对象开发的模型

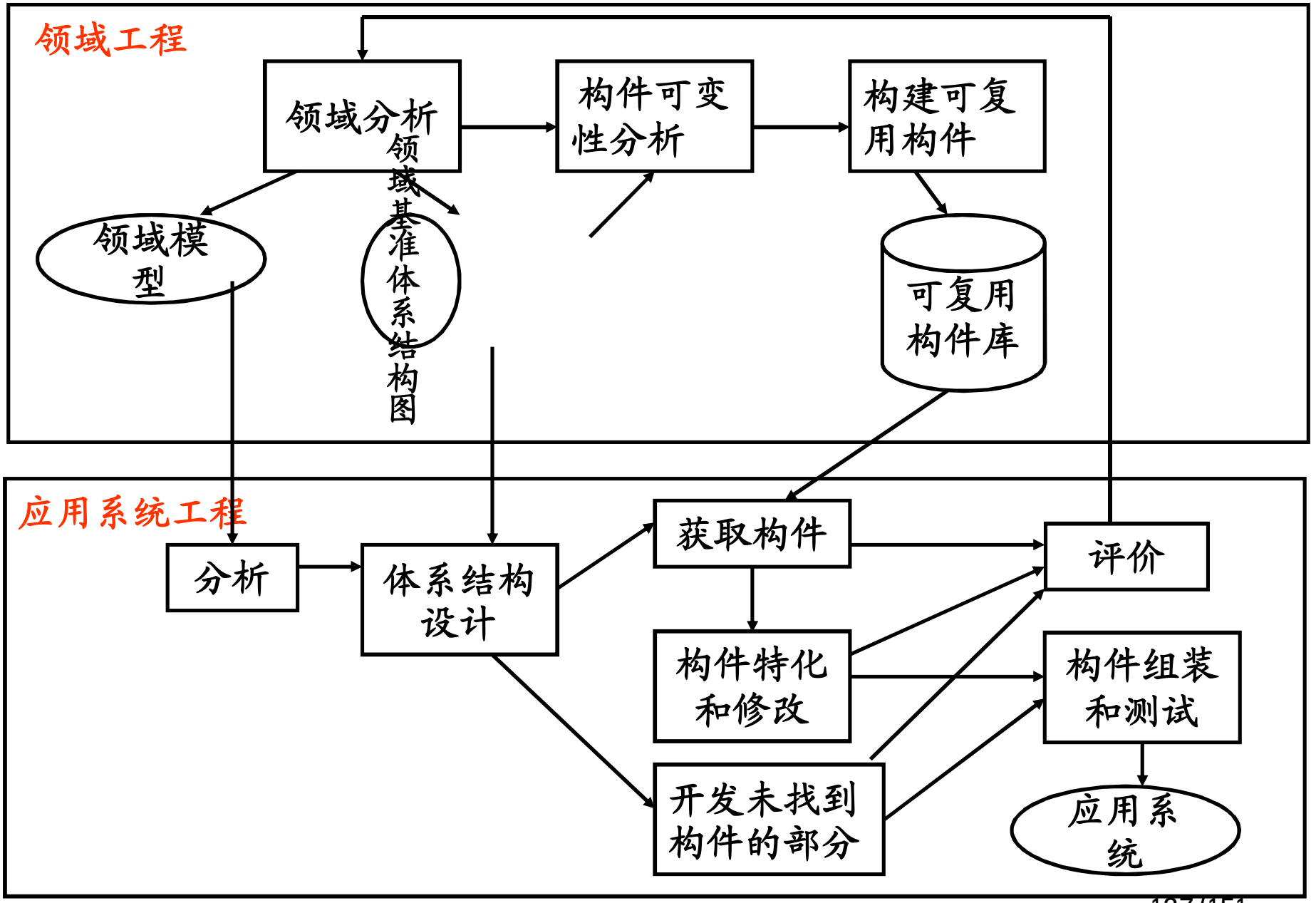
体现迭代和无间隙特征

- 迭代：各开发活动常常重复工作多次，相关的功能在每次迭代中随之加入演进的系统
- 无间隙：开发活动之间不存在明显的边界

基于构件的开发模型

支持软件复用（reuse）

利用预先包装好的软件构件（包括组织内部开发的构件和现存商品化构件COTS）来构造应用系统



- 领域工程的目的是构建领域模型、领域基准体系结构和可复用构件库。
  - Ø领域分析分析该领域中各种应用系统的
  - Ø公共部分或相似部分，构建领域模型和
  - Ø领域基准体系结构（reference
  - Øarchitecture），标识领域的候选构件。
  - Ø对候选构件进行可变性分析，以适应多
  - Ø个应用系统的需要。
  - Ø构建可复用构件，经严格测试和包装后
  - Ø存入可复用构件库（称为构件工程）。

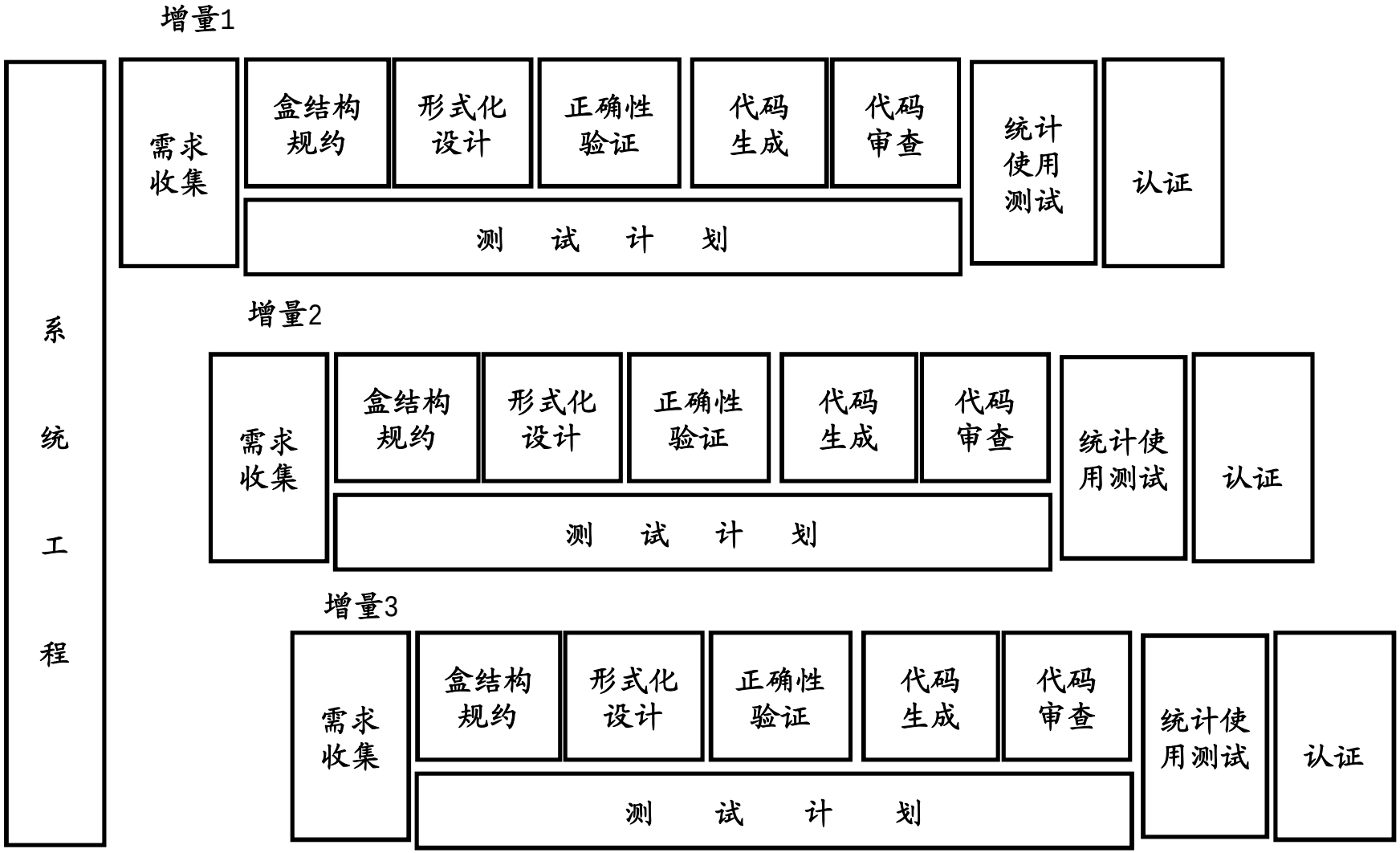
- 应用系统工程的目的是使用可复用构件组装应用系统。
  - Ø分析待开发的应用系统，设计应用系统的体系结构，标识应用系统所需的构件。
  - Ø在可复用构件库中查找合适的构件（也可购买第三方的构件）。
  - Ø特化选中的构件，必要时作适当的修改，以适应该应用系统的需要。
  - Ø开发那些未找到合适构件的应用部分。
  - Ø组装应用系统。
  - Ø评价构件的复用情况，以改进可复用构件，同时对新开发的部分进行评价，并向构件工程推荐候选构件。

根据AT&T、Ericsson、HP公司的经验，有的软件复用率高达90%以上，产品上市时间可缩短2～5倍，错误率减少5～10倍，开发成本减少15%～75%。仅管这些结论出自一些较好使用基于构件开发的实例，但毫无疑问，基于构件的开发模型对提高软件生产率、提高软件质量、降低成本、提早上市时间起到很大的作用。

形式方法模型

- 形式化方法（formal methods）是建立在严格数学基础上的一种软件开发方法。软件开发的全过程中，从需求分析、规约、设计、编程、系统集成、测试、文档生成、直至维护各个阶段，凡是采用严格的数学语言，具有精确的数学语义的方法，都称为形式化方法。
- 形式化方法用严格的数学语言和语义描述功能规约和设计规约，通过数学的分析和推导，易于发现需求的岐义性、不完整性和不一致性，易于对分析模型、设计模型和程序进行验证。通过数学的演算，使得从形式化功能规约到形式化设计规约，以及从形式化设计规约到程序代码的转换成为可能。

净室过程模型



内容摘要

- 计算机软件
- 软件工程
- 软件过程
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

敏捷软件开发

- 软件开发的新挑战
  - Ø快速的市场进入时间，要求高生产率
  - Ø快速变化的需求
  - Ø快速发展的技术
- 传统的软件开发方法
  - Ø强调过程
  - Ø强调文档
  - Ø开发人员负担过重

称为重载(Heavyweight)方法

- 针对上述问题，产生了一系列轻载(Lightweight)方法，如XP、SCRUM等。
- 2001年2月，新方法的一些创始人在美国犹他州成立了敏捷软件开发联盟，简称Agile联盟。
- Agile 联盟起草了一个敏捷软件开发宣言，该宣言由四个价值观声明组成，并提炼出敏捷软件开发方法必须遵循的12条原则。
- Agile方法是在保证软件开发有成功产出的前提下，尽量减少开发过程中的活动和制品的方法。笼统的讲就是，“刚刚好”（Just enough），即开发中的活动及制品既不要太多也不要太少。

Agile方法的价值观

- 个人和交互高于过程和工具

  不是否定过程和工具的重要性，而是更强调软件开发中人的作用和交流的作用。

  软件是由人组成的团队来开发的，与软件项目相关的各类人员通过充分的交流和有效的合作，才能成功地开发出得到用户满意的软件。

  如果光有定义良好的过程和先进的工具，而人员的技能很差，又不能很好地交流和协作，软件是很难成功地开发的。

- 可运行软件高于详尽的文档

  通过执行一个可运行的软件来了解软件做了什么，远比阅读厚厚的文档要容易得多。

  敏捷软件开发强调不断地快速地向用户提交可运行的软件（不一定是完整的软件），以得到用户的认可。

  好的必要的文档仍是需要的，它能帮助我们理解软件做什么，怎么做以及如何使用，但软件开发的主要目标是创建可运行的软件。

- 与客户协作高于合同（契约）谈判

  只有客户才能明确说明需要什么样的软件，然而，大量的实践表明，在开发的早期客户常常不能完整地表达他们的全部需求，有些早期确定的需求，以后也可能会改变。

  要想通过合同谈判的方式，将需求固定下来常常是困难的。

  敏捷软件开发强调与客户的协作，通过与客户的交流和紧密合作来发现用户的需求。

- 对变更及时做出反应高于遵循计划

  任何软件项目的开发都应该制订一个项目计划，以确定各开发任务的优先顺序和起止日期。然而，随着项目的进展，需求、业务环境、技术等都可能变化，任务的优先顺序和起止日期也可能因种种原因会改变。

  因此，项目计划应具有可塑性，有变动的余地。当出现变化时及时做出反应，修订计划以适应变化。

Agile方法的指导原则

（1）最优先的是通过尽早地和不断地提交有价值的软件使客户满意

（2）欢迎变化的需求，即使该变化出现在开发的后期，为了提升对客户的竞争优势，Agile过程利用变化作为动力

（3）以几周到几个月为周期，尽快、不断地发布可运行软件

（4）在整个项目过程中，业务人员和开发人员必须天天一起工作

（5）以积极向上的员工为中心建立项目组，给予他们所需的环境和支持，对他们的工作予以充分的信任

（6）项目组内效率最高、最有效的信息传递方式是面对面的交流

（7）测量项目进展的首要依据是可运行的软件

（8）敏捷过程提倡可持续的开发，项目发起者、开发者和用户应能长期保持恒定的速度

（9） 应时刻关注技术上的精益求精和好的设计，以增强敏捷性

（10）简单化是必不可少的，这是尽可能减少不必要工作的艺术

（11）最好的构架、需求和设计出自于自我组织的团队

（12）团队要定期反思怎样才能更有效，并据此调整自己的行为

Agile方法的适用范围

Martin Fowler认为：新方法不是到处可适用的

适合采用Agile方法的情况:

- 需求不确定、易挥发（Volatile,意指今天的要求明天就不需要了）
- 有责任感和积极向上的开发人员
- 用户容易沟通并能参与

Agile的典型方法

- Extreme Programming (简称XP)
- SCRUM
- Crystal Methodologies (简称Crystal)
- Feature Driven Development(简称FDD)
- Dynamic Systems Development Methodology(简称DSDM)
- Adaptive Software Development(简称ASD)
- Pragmatic Programming等

XP方法

- 由Kent Beck提出，是Agile方法中最引人注目的一个
- XP最初实践于1997年Crysler公司的C3项目（Smalltalk开发）
- 适用于10人以下项目组、开发地点集中的场合
- 广泛用于需求模糊和挥发性强的场合
- IONA公司的Obix技术支持小组在采用了XP方法后，软件生产率提高了67%

XP方法的4个价值观

1.交流（Communication）

- 实践表明，项目失败的重要原因之一是交流不畅，使得客户的需求不能准确地传递给开发人员，造成开发人员不能充分理解需求；模型或设计的变动未能及时告知相关人员，造成系统的不一致和集成的困难
- 所有项目相关人员之间充分的有效的交流是软件开发成功所必不可少的
- XP方法提倡面对面的交流，这是一种有效的也是效率最高的交流方式

2.简单（Simplicity）

- 指在确保得到客户满意的软件的前提下，做最简洁的工作（简单的过程、模型、文档、设计和实现）
- 在开发中不断优化设计，时刻保持代码简洁、无冗余
- 体现了敏捷开发的“刚刚好(Just enough)”思想，即开发中的活动及制品既不要太多也不要太少，刚好即可

3.反馈（Feedback）

及时有效的反馈能确定开发工作是否正确，及时发现开发工作的偏差并加以纠正。

强调各种形式的反馈，如非正式的评审（走查，Walkthrough）、小发布等。

4.勇气（Courage）

采用敏捷软件开发需要勇气:

- Ø信任合作的同事，也相信自己
- Ø做能做到的最简单的事
- Ø只有在绝对需要的时候才创建文档
- Ø让业务人员制定业务决策，技术人员制定技术决策
- Ø用可能的最简单的工具，例如白板和纸，只有在复杂建模工具能提供可能的最好价值时才去使用它们
- Ø相信程序员能制定设计决策，不需要给他们提供过多的细节
- Ø需要勇气来承认自己是会犯错误的，需要勇气来相信自己明天能克服明天出现的问题。

XP方法的12个核心实践

1.完整的团队（Whole Team）

- Ø 所有的小组成员应在同一个工作地点工作
- Ø 成员中必须有一个现场用户（On-site User）由他提出需求，确定开发优先级
- Ø 通常还设一个“教练”（Coach）角色

  教练指导XP方法的实施，以及与外部的沟通和协调

2.计划对策（Planning Game）

包括两类：发布计划和迭代（Iteration）计划

3. 系统比喻（Metaphor）

系统比喻是对待开发的软件的一种形象化比喻，相当于一个粗略的软件体系结构

4. 小发布（Small release）

经常、不断地发布可运行的、具有商业价值的小软件版本，供现场用户评估或最终使用

5. 测试（testing）

XP方法提倡测试优先，即先写测试后编代码（testing then coding）

6. 简单设计（Simple Design）

- Ø 设计只考虑当前定义的功能而不考虑以后需求的变化
- Ø 该设计是完成目前功能所需的最简洁的设计

7. 结对编程（Pair Programming）

一个程序员编程的同时，另一个程序员负责检查程序的正确性和可读性

结对的伙伴可以动态调整

8. 设计改进（Design Improvement）

在不影响程序的外部可见行为的情况下，按高内聚低耦合的原则对程序结构进行改进，保持代码简洁、无冗余

9. 持续集成（Continuous Integration）

每完成一个模块的开发（包括该模块的单元测试）后，立即将其组装到系统中，并进行集成测试，完成该集成测试后才能进行下一次集成

10. 代码全体共有（Collective code Ownership）

团队中的任何人可以在任何时候修改系统任何位置上的任何代码

团队的成员都可以参加模型的开发，又有系统比喻、结对编程、编码标准、持续集成等实践，这些都为代码全体共有提供了支持

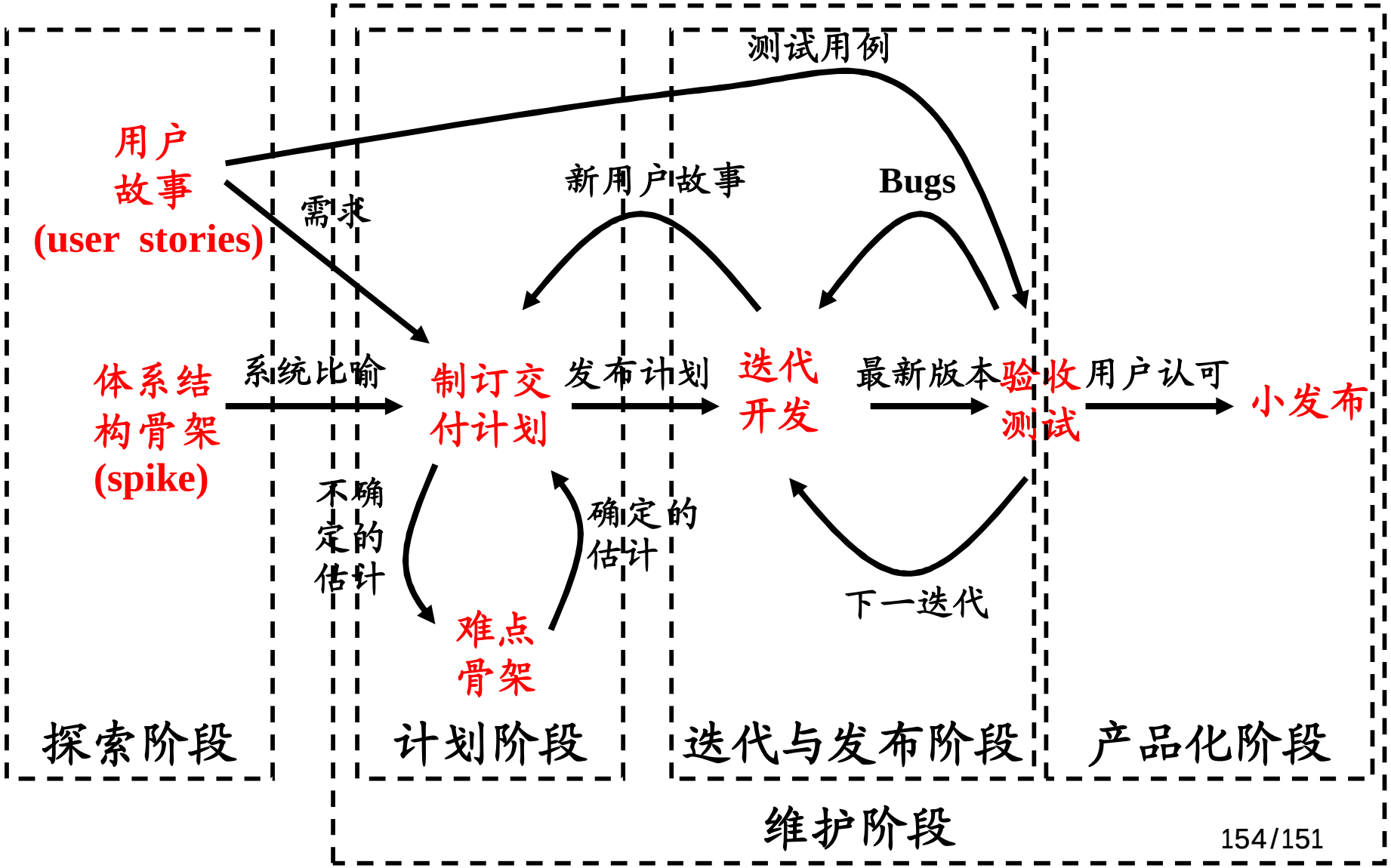
11. 编码标准（Coding Standard）

XP方法强调制订一个统一的编码标准，包括命名、注释、格式等编程风格

12. 可持续步调（Sustainable Pace）

每周40小时工作制

XP方法的开发过程—P31图1.15



探索阶段

- 探索阶段的主要工作是开发初始的用户故事（User Stories）和体系结构骨架（architecture spike）。
- 用户故事描述了系统高层的需求，它是制订发布计划的输入。
- 在探索阶段，试探找到系统中固定不变的部分（体系结构骨架），并找出一种形象的比喻，这种比喻描述了你打算如何构建系统，起到概念框架的作用。
- 探索阶段还应根据用户故事编制相应的测试用例，供以后验收测试时使用。

计划阶段

- 计划阶段的任务是根据用户故事描述的需求、系统体系结构骨架和系统比喻来制订迭代计划和发布计划。
- 使用你最熟悉的形式为用户故事建模，这个模型描述了用户故事的任务以及这些任务之间的关系。
- 通常图形方式（可以是草图）比文字描述更直观。
- 尽可能精确地估算工作量，这是制订计划的重要依据。对于那些不能确切估算其工作量的难点部分，要进一步作分析，直至能确定其工作量估算。

迭代到发布阶段

- 迭代到发布阶段根据迭代和发布计划，开发满足指定用户故事需求的软件，并与前面已完成的软件版本集成，得到软件的一个新版本。
- 根据在探索阶段编写的测试用例，进行验收测试。一旦发现错误或者通过验收测试想进入下一轮迭代时，就重复迭代开发的工作。
- 在这一阶段当客户提出新的用户故事，或者根据项目的进展情况认为有必要时，可以回到计划阶段，对迭代和发布计划做出修改或调整。

产品化阶段

- 产品化阶段的工作主要是确认迭代开发的软件已经做好进入产品化的准备。
- 在此阶段可进行更多的测试，如系统测试、负载测试、安装测试等。
- 另一个工作就是整理文档。虽然敏捷软件开发的价值观中强调“可运行软件高于详尽的文档”，但是，必要的文档仍是需要的。

可能要写的文档:

- 系统文档

系统文档的目的在于为系统提供一个总览，来帮助人们理解它。主要包括：系统技术体系结构和业务体系结构的总览、高层次的系统需求、关键设计决策的总结、体系结构图以及重要的设计模型（如果有的话）等。

- 操作文档

操作文档的内容包括：系统涉及的依赖关系，与其他系统、数据库以及文件文互的特性，对备份流程的参考引用，系统的联系人列表以及联系方法，系统的适用性及可靠性需求的总结，系统预期负载情况概况，以及排错指导原则。

- 支持文档

支持文档的内容包括：支持人员专用的培训教材，解决问题时作为参考的用户文档，排错指导原则，解决疑难问题时的上报流程，以及维护团队的联系列表。

- 用户文档

参考手册用于快速查询；用户指南用于指明系统的工作方式；支持指南用于指导如何获取其他的帮助；培训资料则主要用于培训。

维护阶段

- 维护阶段涵盖了计划阶段、迭代到发布阶段和产品化阶段
- 通常这个阶段主要包括面向产品的活动，如系统的运行和支持。

内容摘要

- 计算机软件
- 软件工程
- 软件过程
- 软件过程模型
- 敏捷软件开发
- CASE工具与环境

计算机辅助软件工程（CASE）
Computer Aided Software Engineering

在软件工程活动中，软件工程师和管理人员按照软件工程的方法和原则，借助于计算机及其软件工具的帮助，开发、维护、管理软件产品的过程称为计算机辅助软件工程

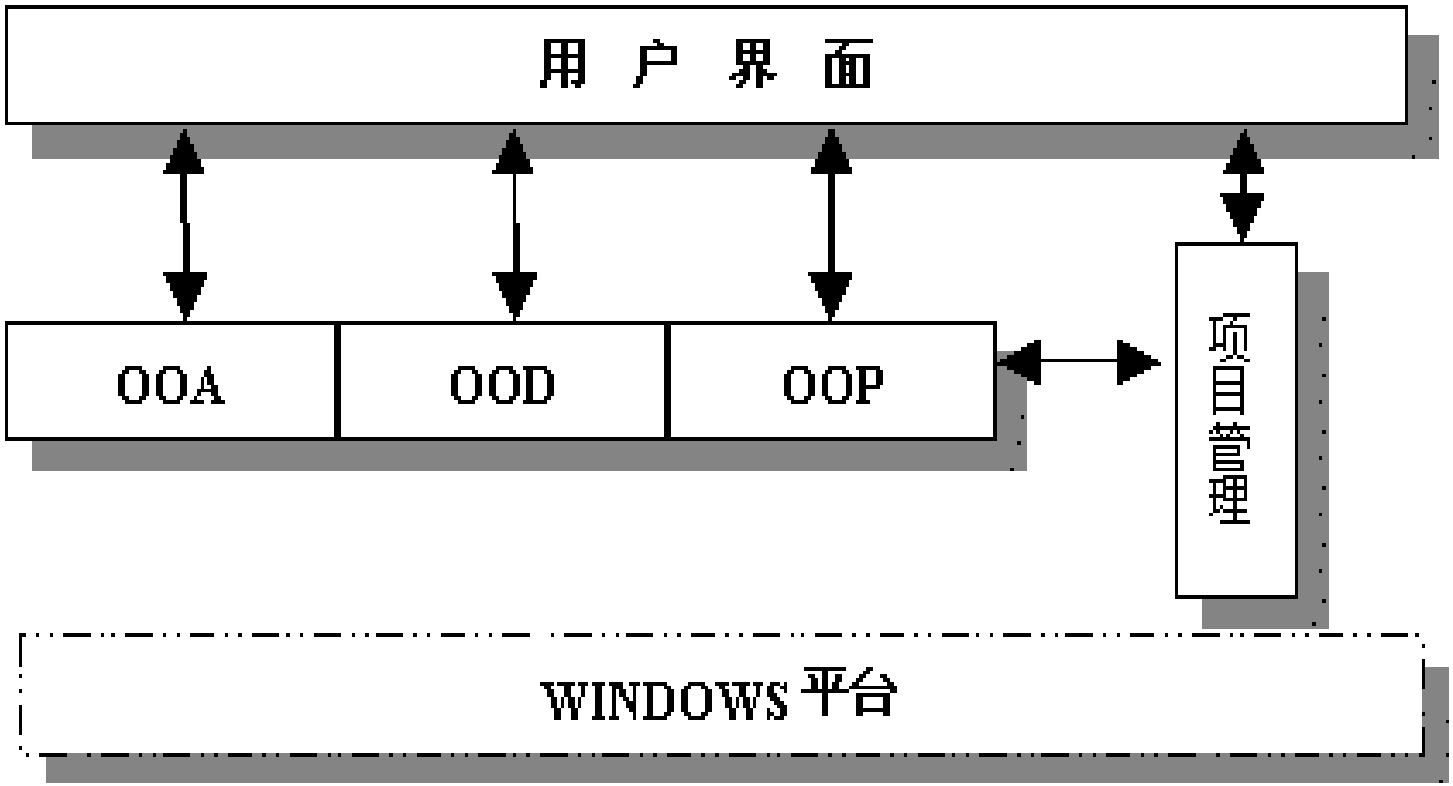
CASE工具

- 软件工具是用来辅助计算机软件的开发、运行、维护、管理、支持过程中的活动或任务的软件
- 按支持的软件过程活动分类：
  - 开发过程：需求分析工具，设计工具，编码工具，测试工具

    它们还可按支持的开发方法分为：结构化XX工具，面向对象XX工具

- 结构化工具:
- 结构化分析的常用工具有数据流图、数据字典、判定树和判定表。而PAD图是常见的过程设计工具中的图形设计。
- 面向对象工具:
- 面向对象分析与设计-OOA与OOD，建模工具UML。另外，由北京大学软件工程研究所开发的青鸟面向对象软件开发工具JBOO。

- JBOO由四部分组成：项目管理器、面向对象分析工具、面向对象设计工具和面向对象编程工具。如下图所示：



JBOO 1.52系统结构图

- 维护过程：版本控制工具，文档分析工具，逆向工程（reverse engineering）工具，再工程（reengineering）工具
- 管理过程：项目管理工具，配置管理工具，软件评价工具
- 应用类工具

集成型软件开发环境

- 集成型开发环境是一种把支持多种软件开发方法和过程模型的软件工具集成到一起的软件开发环境
- 集成型开发环境由环境集成机制和工具集组成

- 环境集成机制包括:
  - ø数据集成机制：为各种相互协作的工具提供统一的数据接口规范
  - ø控制集成机制：支持各个工具或开发活动之间的通信、切换、调度和协同工作，并支持软件开发过程的描述、执行与转接
  - ø界面集成机制：支持工具界面的集成和应用系统的界面开发，统一界面风格

- ------END------