**02326 软件工程**

# 第一章 绪论

## 1.1 软件工程概念的提出与发展

1.1.1软件工程：应用计算机科学理论和技术以及工程原则和方法，按预算和进度实现满足用户要求的软件产品的工程，或以此为研究对象的学科

1.1.2软件工程三要素：工期、质量、成本

1.1.3软件工程作为一门学科，其发展可以划分为两个时期，前期主要研究实现系统技术，后期则开始关注软件质量和软件工程管理。

1.1.4什么是软件危机？软件危机主要表现在哪方面？

1）随着计算机的广泛应用，软件生产率、软件质量远远满足不了社会发展的需求，成为社会、经济发展的制约因素，这一现象称为“软件危机”。

2）主要表现：①超出经费预算；②项目交付时间拖延；③不能满足用户要求；④可维护性差；⑤可靠性差。

## 1.2 软件开发的本质

1.2.1软件=程序+文档

1.2.2软件开发的本质：不同抽象层术语之间的“映射”，以及不同抽象层处理逻辑之间的“映射”。

1.2.3模型：模型是一个抽象

①概念模型：描述软件是什么

②软件模型：实现概念模型的软件解决方案，包括设计模型、实现模型和部署模型

# 需求与获取

## 2.1需求与需求获取

## 2.2需求与需求获取

2.2.1需求的定义（什么是需求）：一个需求是一个“要予构造”的陈述，描述了待开发产品/系统功能能力、性能参数或其它性质。

2.2.2需求的基本性质：必要的、无歧义的、可测的、可跟踪的、可测量的

2.2.3需求的分类：功能需求，规约了系统必须执行的功能，是整个需求的主体。

非功能需求：性能需求、外部接口需求、设计约束和质量属性需求

质量属性需求规约所期望的指标包括：1）可靠性，2）存活性，3）可维护性，4）用户友好性

2.2.4需求发现技术（怎么发现需求）：自悟、交谈、观察、小组会、提炼

（1）自悟：需求人员把自己作为系统的最终用户，审视该系统并提出问题：“如果是我使用该系统，则我需要……”

（2）交谈：为了确定系统应该提供的功能，需求人员通过提出问题、用户回答，直接询问用户想要的是一个什么样的系统。

（3）观察：通过观察用户执行其现行的任务和过程，了解系统运行的环境，了解要建的系统与现有系统之间必须进行的交互，但可能会引起客户抵触。

（4）小组会：举行客户与开发人员的联席会，与客户组织的一些代表共同开发需求。

（5）提炼：复审技术文档，并提取出相关的信息。

## 2.3需求规约（SRS）

2.3.1需求规约的定义（什么是需求规约）：是一个软件/产品/系统所有需求陈述的正式文档，它表达了一个软件/产品/系统的概念模型

2.3.2需求规约的基本性质：重要性和稳定性程度：对需求进行升级

可修改的、完整的：没有被遗漏的需求、一致的：不存在互斥的需求

2.3.3需求规约的格式：引言：目的、范围、定义、缩略语、参考文献、概述

总体描述：产品描述、产品功能、用户特性、约束、假设和依赖

特定需求：是个文档的技术核心、附录、索引

2.3.4需求规约的作用：需求规约是软件开发组织和用户之间一份事实上的技术合同书，是产品功能及其环境的体现。需求规约是一个管理控制点、对于产品/系统的而设计，需求规约是一个正式的、受控的起始点。需求规约是创建产品验收计划和用户指南的基础。

2.3.5基于需求规约一般还会产生初始测试计划和用户系统操作描述两个文档

2.3.6需求规约的表达主要存在3种不同的风格：1）非形式化的需求规约，2）半形式化的需求规约，3）形式化的需求规约

# 结构化方法

## 3.1结构化需求分析

3.1.1结构化方法的定义（什么是结构化方法）：结构化方法作为一种“思想”工具，可以用于定义需求，建立待建系统的功能模型；可用于定义满足需求的结构，给出一种特定的软件解决方案。

3.1.2需求技术的基本特征：提供方便通信的机制、鼓励需求分析人员使用问题空间的术语思考问题，编写文档、提供定义系统边界的方法、提供支持抽象的基本机制、为需求分析人员提供多种可供选择的方案、提供特定的技术，适应需求的变化

## 3.2结构化需求分析

1.需求分析中的基本术语：数据：客观事物的一种表示、信息：具有特定语义的数据、数据是信息的载体、数据流：数据的流动、加工：数据变换单元、数据存储、数据源和数据潭

2.系统功能模型表示方法：数据流程图（DFD图），一种表示数据变换的图形化工具、数据流图的四个组成要素：数据源/数据潭，数据流，数据加工，数据存储

2.1数据流程图的四个图形符号：矩形为数据源/数据潭，椭圆形为数据处理，两条横线为数据存储，箭头为数据流

需求分析面临的挑战：问题空间理解、有效沟通、需求的变化性。

## 3.3结构化需求分析

1.绘制DFD图（数据流图）的八字真言：自顶向下，逐步求精

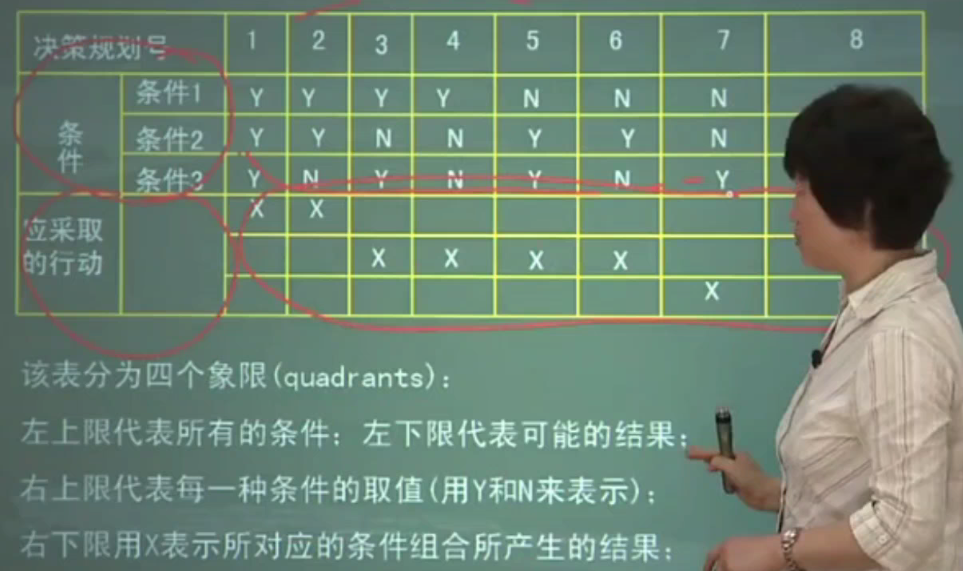
## 3.4结构化需求分析

3.4.1数据字典：定义数据流图中所有数据流和数据存储的数据结构。

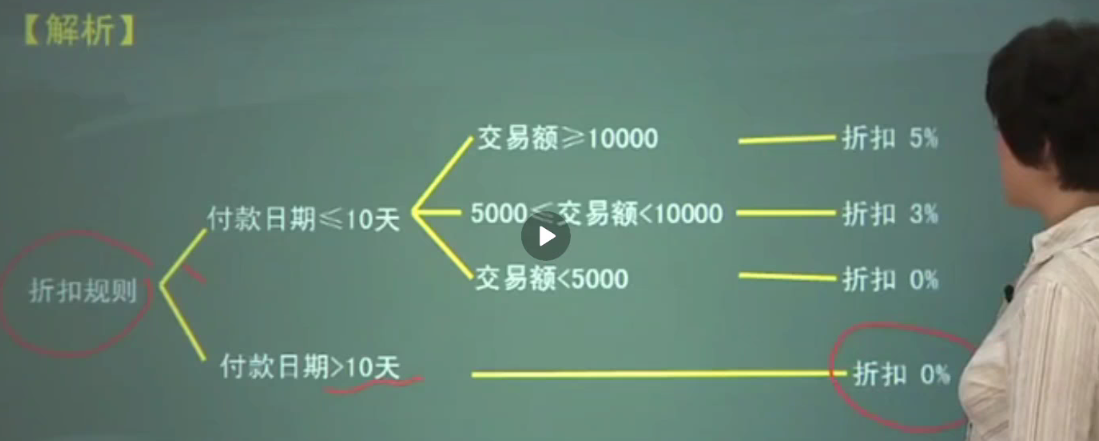
顺序结构：+，选择结构：|，重复结构：{ }，子界：m……n

3.4.2判定表：判断表（Dicision Table）也称为决策表，是一个二维表，它说明了每一种条件组合所产生的结果。

决策表分为4个象限（quadrants）：左上限代表所有的条件；左下限代表可能的结果；右上限代表每一种条件的取值（用Y和N来表示）；右下限用X表示所对应的条件组合所产生的结果；



3.4.3判断树：判定树（Dicision Tree）也称为决策树，是用来描述在一组不同的条件下，决策的行动是根据不同条件及其取值来选择的处理过程。业务规则的描述通常可以使用判断树这一过程描述工具。



## 3.5结构化需求分析

3.5.1结构化语言（if）：若逻辑关系比较简单，可以用结构化自然语言来描述。

3.5.2需求验证

3.5.2.1验证：必要性、无歧义性、可测性、可跟踪性、可测量性

3.5.2.2需求中发现错误的类型：

不正确的事实：40%

遗漏：31%

不一致：13%

歧义性：5%

错放：2%

其他：9%

3.5.3发现错误的方法：

审查：65%

单元测试：10%

评估：10%

集成：5%

## 3.6结构化设计

3.6.1结构化设计的任务：

定义满足需求所需要的结构

确定“怎么做”的问题

划分为：总体设计：以系统为对象；详细设计：以模块为对象

3.6.2总体设计

总体设计的任务：把系统的功能需求分配到一个特定的软件系统中、引入了两个概念

模块：软件中具有特定标识的独立成分

模块调用：模块之间的一种使用关系

3.6.3如何表达模块和模块调用：

Yourdon提出的模块结构图、层次图、美国IBM提出的HIPO图

总体设计阶段的设计工具很多：模块结构图、层次图、HIPO图

## 3.7结构化设计

3.7.1总体设计的步骤

将DFD图映射成设计层面的模块及模块调用

将DFD图转换为初始的模块结构图

基于“高内聚、低耦合”的软件设计原理，通过模块化，将初始的模块结构图转化为最终的模块结构图

3.7.2两种映射方法（DFD图转换为模块结构图）

变换设计：基于变换的数据流程图，是一个线性的顺序结构，由输入、输出、变换中心三部分组成。

事务设计：基于事务的数据流程图中右一个事务处理中心，它将输入分为许多相互平行的加工路径，然后根据输入的属性，选择某一加工路径。

## 3.8结构化设计

3.8.1模块化：把一个待开发的软件分解成若干个简单的、具有高内聚、低耦合的模块，这一过程称为模块化。

3.8.2模块耦合：耦合（coupling）是对两个模块之间相互依赖程度的一种度量。模块间的依赖程度越大，则其耦合程度也就越大；反之，模块间的依赖程度越小，其耦合程度也就越小。

3.8.3模块间耦合类型：

内容耦合：一个模块直接修改或操作另一模块

公共耦合：两个模块共同引用一个全局数据项

控制耦合：一个模块向另一模块传递控制信号

标记耦合：一个模块向两个模块传递一个公共参数

数据耦合：模块之间通过参数来传递数据（最好的方式）

3.8.4模块内聚：是指一个模块内部各成分之间相互关联程度的度量。也就是说，内聚是对模块内各处理动作组合强度的一种度量。很显然，一个模块的内聚越大越好。

3.8.5内聚的类型：

偶然内聚：模块的各成分没有任何关系

逻辑内聚：逻辑上相关的处理放在一起

时间内聚：模块内的功能在同一时间完成

过程内聚：模块内的处理以特定的次序执行

通信内聚：操作同一数据集

顺序内聚：一个成分的输出作为另一成分的输入

功能内聚：模块的所有成分完成单一的功能（最好的方式）

3.8.6启发式规则

“高内聚，低耦合”

改进软件结构，提高软件独立性，模块分解

模块规模适中

力求深度、宽度、扇出、扇入适中

深度：表示其控制的层数

宽度：同一层次上模块总数的最大值

扇出：一个模块直接控制的下级的模块的数目

扇入：有多少个上级模块直接调用它

尽量使模块的作用域在其控制域内

模块的控制域：这个模块本身以及所有直接或间接从属它的模块的集合

模块的作用域：受该模块内一个判断所影响的所有模块的集合

尽力降低模块接口的复杂度

力求模块功能可以预测

## 3.9结构化设计

3.9.1详细设计任务：具体描述模块结构图中的每一模块，即给出实现模块功能的实施机制，包括一组例程（算法）和数据结构。

详细设计的目标：将总体设计阶段产生的系统高层结构映射为以相关术语表达的低层结构，也是系统的最终结构。

3.9.2结构化（模块化）程序设计方法：是一种基于结构的编程方法，即采用顺序结构、选择结构（分支结构）和重复结构（循环结构）进行编程，其中每一个结构只允许一个入口和一个出口。

结构化程序设计的本质是：使程序的控制流程线性化，实现程序动态执行顺序符合静态书写的结构，提高程序的可读性。

## 3.10结构化设计

3.10.1设计规约：完整准确地描述满足需求规约所要求的所有功能模块，以及伴随功能模块而出现的非功能机制。设计规约包括概要设计规约和详细设计规约。

3.10.2概要设计规约：指明高层软件体系结构

系统环境

软件模块的结构

模块描述

文件结构和全局数据文件的逻辑结构

测试需求

3.10.3详细设计规约：详细设计规约主要作为软件设计人员与程序员之间交流的媒体

各处理过程的算法

算法所涉及的全部数据结构的描述

# 面向对象方法——UML

## 4.1 UML术语表

4.1.1 UML是什么：UML是一种可视化的语言，可用于规约系统制品、构造系统的制品、建立系统制品的文档，可以作为软件需求规约、设计和实现的工具。UML给出了方法学中不同抽象层次术语以及模型表达工具，或者说UML给出规约软件系统/产品的术语和表达格式。

4.1.2 UML术语分类：表达客观事物的术语、表达关系的术语、表达组合关系的术语——包

## 4.2 UML术语表

表达客观事物的术语

4.2.1 对象（object）：对象是系统中用来描述客观事物的一个实体。一个对象由一组属性和对这组属性进行操作的一组方法组成。

4.2.2 类（class）：类是具有相同属性、操作、关系和语义的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述，其内部包括属性和服务两个主要部分。

4.2.3 类的基本属性：1）可以有多个属性，也可以没有属性；2）可见性：public、protected、private；3）多重性；4）初始值；5）性质串；6）作用范围；

4.2.4 类的基本操作：1）操作是对外提供的一种服务；2）可以被其他对象调用；3）一个类可以有多个操作，也可以没有操作；4）操作可以是抽象操作；5）可见性；6）操作的表达语法；7）性质串；

4.2.5 类要满足的基本条件

一个结构良好的类，必须符合下列条件：

1）明确抽象了问题域或解域中某个有形事物或概念

2）包含了一个小的、明确定义的职责集，并能很好实现

3）清晰地分离了抽象和实现

引入”可见性“的目的是为了支持信息隐蔽这一软件设计原则，将可见性分为public、protected、private三种。

## 4.3 UML术语表

4.3.1 接口的含义：接口是操作（方法）的一个集合，其中每个操作描述了类、构件、或子系统的一个服务（方法）。

4.3.2 接口应用中注意的问题：1）接口只能被其它类目（类）使用，其本身不能访问其它类目（类）。

2）接口描述类的外部可见操作，通常是该类的一个特定有限行为。

3）接口不描述其中操作的实现，也没有属性和状态。

4）接口之间没有关联、泛化、实现和依赖。

用况（use case）用例：对一组动作序列的描述，系统执行这些动作应产生对特定参与者有值的、可观察的结果。

## 4.4 UML术语表

4.4.1 主动类：1）至少具有一个进程或线程的类。能够启动系统的控制活动，并且其对象的行为通常与其它元素行为并发的。

2）表示方法：两条竖线

3）用来模型化系统中的并发行为

4.4.2 构件/组件：1）系统设计中的一种模块化部件，通过外部接口隐藏了它的内部实现。

2）具有相同接口的构件可以相互替代

3）构件可以嵌套

4）构件用于表达解空间中可独立标识的成分

4.4.3 制品（Artifact）

1）系统中包含物理信息的、可替代的物理部件

2）部署制品（指上线的程式）：这类制品是构成一个可执行系统必要而充分的制品，例如DLL、EXE文件。

3）工作产品制品：这类制品本质上是开发过程中产物，由源代码文件、数据文件等用来创建部署制品的事物构成

4）执行制品：这类制品是作为一个正在运行的系统的结果而被创建。一般存在于内存之中

4.4.4 节点：节点是在运行时存在的物理元素，通常表示一种具有记忆能力和处理能力的计算机资源

## 4.5UML术语表

表达关系的术语

4.5. 关联（Association）：1）关联反映了类和类之间的静态关系。关联在模型中，特别是永久业务对象模型中是最基本的关系。

2）关联是类目之间的一种结构关系，是对一组具有相同结构、相同链的描述。

3）链：对象之间具有特定语义关系的抽象

4.5.2 关联的语义表达：1）关联名；2）导航（从一个类推断另一个类）；3）角色；4）可见性（public、protected、private）；5）多重性（Multiplicity）：定义了与一个对象/类相联系的对象/类出现一次，该对象/类可能出现的最小和最大数目。

6）限定符；7）聚合：一个类是另一个类的一部分；8）组合：是聚合的一种特殊形式；9）关联类：具有关联和类特性的模型元素；10）约束：有序（orderod）、无重复的（set）、有重复的（bag）、有序集合（order set）、列表（list）、只读（read only）

## 4.6 UML术语表

4.6.1 泛化/继承：特殊类（子类）的对象拥有其一般类（超类）的全部属性与服务，称做特殊类对一般类的继承（Inheritance）。利用继承（inheritance），子类可以继承父类的属性和方法。子类/父类也可分别叫特殊类/一般类、子类/超类、派生类/基类。

4.6.2 细化/实现：1）细化是类目之间的语义关系，其中一个类目规约了保证另一个类目执行契约。2）用空心三角形的虚线表示

3）在以下2个地方会使用实现关系：接口与实现它们的类和构件之间用况与实现它们的协作之间。

4.6.3 依赖：1）依赖是一种使用关系，用于描述一个类目使用另一类目的信息和服务

2）用有向虚线段表示

3）依赖的分类：绑定（Bind）导出（Derive）允许（Perimt）实例（Instantiate）

4.6.4 关系术语的使用：1）结构关系；2）继承关系；3）精化关系；4）依赖关系；

4.6.5 包：是模型元素的一个分组，一个包本身可以嵌套在其它包中，并且可以含有子包和其它类型的模型元素。

包的可见性；为了模型化包之间的关系，UML给出了两种依赖：访问（use/access）引入（import）

为了控制信息组织的复杂性，UML提供了组织信息的一种通用机制--包

## 4.7 UML的模型表达格式

### 4.7.1 类图

1）类图（class diagram）表达系统的静态结构信息，即系统是由哪些类组成的，这些类之间的关系是什么。

类图显示系统各个部分以及怎样将它们组装起来。

2）构造类图的3个关键问题点是：

系统中有哪些需要关心的类？

这些类是如何描述的？

这些类之间的联系是什么？

### 4.7.2 用况图（use case图）

1）用况是对一个参与者（actor）使用系统的一项功能时所进行的交互过程的一个文字描述序列。

2）用况图是一种表达系统功能模型的图形化工具

3）用况图的6个模型元素：

主题；用况；参与者：系统用户、另一个系统、时间；关联、泛化、依赖；用况是系统开发的支点（使用用况图获取系统需求）；大多数的系统功能都可以表示成用况

## 4.8 UML的模型表达格式

### 4.8.1 状态图

状态图（state chart diagram）使用状态、事件和转换来记录对象在其生命周期中所历经的状态序列。

对象的初始状态是图中任何事件都未对该对象起作用时的状态。

状态代表对象生命周期的某一瞬间。

转换表明作为对事件响应结果，对象将从一种状态转换到另一种状态并执行某个动作

触发状态转换的事件在状态转换字符串中命名。双击一个状态转换，除事件签名以外，还可用字符串为其加注临界条件，动作表达式等标签。

状态图中的3个术语：

1）状态：一个实例所处的特定阶段、所具有的对外呈现以及所能提供的服务。

2）事件：信号事件；调用事件；时间事件；变化事件

3）状态转移

## 4.9 UML的模型表达格式

### 4.9.1 顺序图

4.9.1.1 顺序图（sequence diagram）表示了对象之间传送消息的时间顺序，也就是对象之间的交互顺序。

这些交互顺序是指在场景或用况的事件交流中发生的。

每一个对象（类）用一条生命线来表示——即用垂直线代表整个交互过程中对象的生命周期。

生命线之间的箭头连线代表消息。

4.9.1.2 顺序图中的基本元素包括：

活动者，指用况中的活动者。

对象，指在用况中的内部对象。

生命线：在顺序图中的一个对象下面的竖线，用以显示这个对象的生命期。

消息，指场景内由事件流定义的内部事件成为在对象和活动者或其他对象之间的消息。

4.9.1.3 消息的类型：

1）同步消息——返回消息，同步消息假定有一个返回消息，同步消息用有实心的箭头表示；返回消息用虚线、箭头也不是实心来表示；

2）反身消息——消息的发送方和接收方是同一个对象；

3）异步消息——没有返回值的消息，用非实心箭头表示；

4）定时消息——对消息附加时间约束条件，包括：发送时间、接受时间、已用时间等；

4.9.1.4 控制操作子：为了控制交互行为描述的复杂性，以便更清晰地表达顺序图中的复杂控制，UML给出了4种最常用的控制操作子。

1）选择执行操作子（Operator for Optional Execution），该控制操作子记为“Opt”，由两部分组成，一是监护条件，二是控制体。

2）条件执行操作子（Operator for Condtional Execution），该控制操作子记为“alt”，控制体通过水平线将其分成一部分些，每一部分表示一个条件分支，每个分支有一个监护条件。

3）并发执行操作子（Operator for Parallel Execution）,该控制操作子记为“par”，该控制操作子的体通过水平线将其分为多个部分。每一部分表示一个并行计算，在大多数情况下，每一部分涉及不同的生命线。该控制操作子表明，当进入该控制操作子时，所有部分并发执行。

4）迭代操作子（Operator for Iterative Execution）,该控制操作子记为“loop”

# 第五章 面向对象方法——RUP

## 5.1 RUP的特点

5.1.1 统一软件开发过程（Rational Unified Process：RUP）是对象管理组织（OMG）所推荐的一个有关过程的标准。

2）RUP是基于UML的一种过程框架。比较完整地定义了将用户需求转换成产品所需要的活动集，并提供了活动指南以及对产生相关文档的需求。

3）RUP适应于大多数软件系统的开发，基于构件。

5.1.2 过程模型：

1）需求获取模型

2）需求分析模型

3）软件设计模型

4）软件实现模型

5）软件测试模型

5.1.3 RUP的特点：以用况驱动的、以体系结构为中心的迭代、增量式开发。

1）用况驱动，在系统的生存周期中，以用况作为基础，驱动系统有关人员对所要建立系统的功能需求进行交流，驱动系统分析、设计、实现和测试等活动，包括制定计划、分配任务、监控执行和进行测试等，并将它们有机地组织在一起，使各个阶段中都可以回溯到用户的实际需求。

2）以体系结构为中心

系统体系结构：是对系统语义的概括描述，对所有项目有关人员都是可以理解的。

关注子系统、构件、接口、协作、关系和节点等重要模型元素，而忽略其他细节。

3）迭代与增量

迭代是重复的部分

增量是增加的部分

5.1.4 初始阶段的基本目标

获得与特定用况和平台无关的系统体系结构轮廓；为系统建立商业案例；确定项目的边界；从业务角度指出该项目的价值；第一个重要的里程碑：生命周期目标（Lifecycle Objective）里程碑

5.1.5 细化阶段

细化阶段的目标是分析问题领域，建立健全的体系结构基础，编制项目计划，淘汰项目中最高风险的元素；第二个重要的里程碑：生命周期结构（Lifecycle Architecture）里程碑。

5.1.6 构造阶段

形成最终的体系结构基线；开发完整的系统；确保产品可以开始向客户交付；第三个重要的里程碑：初始功能（Initial Operational）里程碑

5.1.7 交付阶段

确保软件对最终用户是可用；基于用户反馈的少量调整；第4个里程碑：产品发布（Product Release）里程碑

## 5.2 核心工作流

### 5.2.1 需求获取

5.2.1.1 核心工作流

RUP中有9个核心工作流，分为6个核心过程工作流（Core Process Workflows）和3个核心支持工作流（Core Supporting Workflows）。

5.2.1.2 需求获取：1）RUP运用用况（Use case）技术来获取需求

2）需求获取的基本步骤：

列出候选的需求：特征列表

理解系统语境：领域模型或业务模型

捕获功能需求：用况模型

捕获非功能需求：补充需求或针对特定的一些用况

5.2.1.3 列出候选的需求：搜取特征：是一个新的项（Item）及其简要描述

5.2.1.4 理解系统语境：

1）创建领域模型或业务模型

2）业务用况模型：业务参与者和业务用况

3）业务对象模型：三个术语：工作人员、业务实体、工作单元；用交互图和活动图来表达

5.2.1.5 捕获功能需求：建立系统的用况模型（用况图）

1）发现和描述参与者

2）发现并描述用况（用户所需要的功能）

3）确定用况的优先级

4）精化用况

5）构造用户界面模型

6）用况模型的结构化（拆分成各种图）

## 5.3 核心工作流

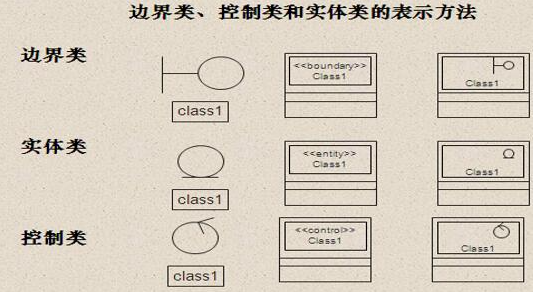
### 5.3.1 需求分析

5.3.1.1 需求分析的目标：在系统用况模型的基础上，创建系统分析模型以及在该分析模型视角下的体系结构描述

5.3.1.2 术语

1）分析类：是类的一种衍型，很少有操作和特征标记，而用责任来定义其行为，并且其属性和关系也是概念性的。

存在3种不同的类：实体类、边界类、和控制类



2）用况细化：用况细化是一个协作；针对一个用况，其行为可以用多个分析类之间的相互作用细化，并记为用况细化。

协作是一个交互，涉及交互的三要素：交互各方、交互方式、以及交互内容

3）分析包

分析包体现了“局部化”、“问题分离”等软件设计原理；分析包把一些变化限制到一个业务过程、一个参与者的行为或一组紧密相关的用况，形成一些不同的分析包；体现“高内聚，低耦合”

5.3.1.3 分析模型的表达

1）分析模型是由“分析系统”来定义的

2）分析系统包含一组具有层次结构的包

3）一个包可以包含一些分析类和用况细化

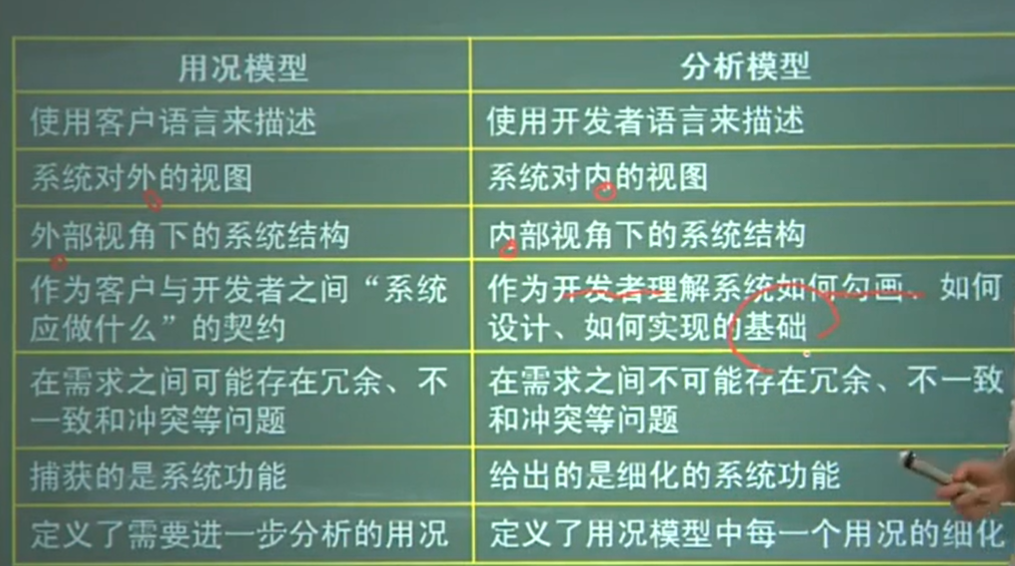
RUP需求分析中，所创建系统分析模型也是系统的一种概念模型，解决系统用况模型中存在的二义性和不一致性等问题。

5.3.1.4 需求分析总结

1）三个术语：分析包、分析类、用况细化

2）四个步骤：体系结构分析、细化用况、对类分析、对包进行分析

3）一个成果：分析模型



## 5.4 核心工作流

### 5.4.1 软件设计

5.4.1.1 软件设计：定义满足需求规约所需要的软件结构。

RUP的设计目标：定义满足系统/产品分析模型所规约需求的软件结构。

5.4.1.2 相关术语：1）设计类；2）用况细化；3）设计子系统；4）接口；

5.4.1.3 设计模型、部署模型及相关视角下的体系结构描述

1）设计模型：设计子系统；设计类；用况细化；接口；

2）部署模型：是一个对象模型，描述了系统的物理分布，即如何将功能分布于各个节点上。

5.4.1.4 设计的主要活动：1）体系结构设计；2）用况的设计；3）类的设计；4）子系统设计；

5.4.1.5 RUP设计小结

1）RUP设计的特点：使用了一种公共的思想来思考设计，并使设计可视化；给出了有关子系统、设计类和接口的需求；支持对底线工作的分解，时之称为一些可以由不同开发组尽可能同时处理的、可管理的部分。

2）RUP的设计方法

给出表达设计模型的基本成分的术语；规约了设计模型的语法，指导模型的表达；给出了创建设计模型的过程以及相应的指导

3）RUP的设计模型

设计子系统和服务子系统，以及它们的依赖、接口和内容；设计类，以及它们具有的操作、属性、关系及其实现需求；用况细化；设计模型视角下的体系结构描述；

4）RUP的部署模型

节点，它们的特征以及连接；主动类到节点的初始映射；

5.4.1.6 RUP设计对实现的影响

1）设计子系统和服务子系统由实现子系统予以实现；2）设计类由文件化构件予以实现；3）在规划实现工作时，将要使用用况细化以产生一些“构造”；4）在节点上部署构件、形成分布系统时，将使用部署模型和网络配置

## 5.5 核心工作流

### 5.5.1 RUP的实现与测试

5.5.1.1 RUP的实现目标：1）基于设计类和子系统生成构件；2）对构成进行单元测试；3）进行集成和连接；4）把可执行的构件映射到部署模型；

5.5.1.2 RUP实现的主要活动：1）实现体系结构；2）集成系统；3）实现子系统；4）实现类；5）完成单元测试；

5.5.1.3 RUP的测试：包括内部测试、中间测试、和最终测试

5.5.1.4 RUP测试包括的主要活动：1）计划测试；2）设计测试；3）实现测试；4）执行集成测试；5）执行系统测试；6）评价测试；

# 第六章 软件测试

## 6.1 软件测试目标与软件测试过程模型

6.1.1 软件测试的对象

软件=程序+文档

测试对象：各个阶段产生的源程序和文档。

6.1.2 软件测试的目的

测试的目的应该是通过软件测试尽可能多地发现并改正软件中存在的错误。

6.1.3 软件测试的定义

1）软件测试（Software Testing）是按照特定规程发现软件错误的过程。

2）使用人工或自动手段，运行或测定某个系统的过程，其目的是检验它是否满足规定的需求，或清楚了解预期结果与实际结果之间的差异。

6.1.4 “测试”和“调试”的区别

1）测试证明“失败”，调试证明“正确”

2）测试以已知条件开始

3）测试时有计划的

4）测试是一个发现错误、改正错误、重新测试的过程

5）测试的执行是有规程的

6）测试由独立的测试小组完成

7）测试的执行和设计可由工具支持

6.1.5 测试过程模型

1）测试设计：环境模型；对象模型；错误模型

2）测试执行

3）测试结果比较

## 6.2 软件测试技术

### 6.2.1 路径测试技术

1）是一种百盒测试技术；2）依据的是程序的逻辑结构；3）采用控制流程图来表达被测程序模型；4）通过合理地选择一组穿过程序的路径，以达到某种测量度量。

6.2.1.1 控制流程图：是一种表示程序控制结构的图形化工具，其基本元素是过程块、节点、判定。

过程块：是没有被判定和/或被节点分开的一组程序语句。基本特性：若过程块中的某个语句被执行，则块中所有语句都被执行。

判定：是一个程序点，此处控制流出现分叉。

节点：也是程序点

链：是过程块、判定、节点之间一种具有特定语义的关系。

路径：是由链组成的，包含一串指令或语句，其长度由链的数目决定。

对软件测试而言，限定路径为：从程序的入口开始，在出口结束。

人们关于软件测试的认识，大体经历了5个阶段：1.第一阶段认为软件测试和调试没有任何区别；2.第二阶段认为测试是为了表明软件能正常工作；3.第三阶段认为软件测试是为了表明软件不能正常工作；4.第四阶段认为测试仅是为了将已察觉的错误风险降低到一个可接收的程度；5.第五阶段认为测试不仅是一种行为而且是一种理念，即测试是产生低风险软件的一种训练

软件测试技术通常分为两种，即基于程序路径的“白盒”测试技术和基于规约的“黑盒”测试技术。黑盒法又称功能测试法，它是根据程序功能的分析，推演出由函数定义域中有代表性的元素组成测试集。功能测试法又具体分为等价类法、边值分析法、因果图法和错误猜测法等。白盒法也称之为结构测试或逻辑覆盖法，它是根据对软件内部逻辑结构的分析，选取测试数据集（即测试用例主要主要技术是路径测试技术）

6.2.1.2 测试策略

1）路径覆盖（PX）：执行所有可能穿过程序控制流程的路径。最强的测试度量。

2）语句覆盖（C1）：至少执行程序中所有语句一次。最低的测试度量。

3）分支覆盖（P2）：至少将程序的每个分支执行一次，条件覆盖与条件组合覆盖。

4）几种测试覆盖存在以下基本关系：语句覆盖<=分支覆盖<=条件组合覆盖<=路径覆盖

6.2.1.3 路径选取与用例设计

最小的强制性测试需求语句覆盖率。

路径选取的一般原则：1）选择最简单的、具有一定功能含义的入口/出口路径。

2）在已选取的基础上，选择无循环的路径，选取短路径、简单路径。

3）选取没有明显功能含义的路径，要研究该路径为什么存在

### 6.2.2 基于事务流的测试技术

是一种功能测试技术，属于黑盒测试技术

1）事务：是指从系统用户的角度出发所见到的一个工作单元，有其“生”，有其“忘”。

2）事务是由一系列操作组成，用“事务流”表达。

事务流：是系统行为的一种表示方法，为功能测试建立了程序的动作模式

事务流程图：表达系统的行为，多个事务流的执行。

3）事务流程图中的相关概念

并生：事务处理产生一个新事务，由此这两个事务继续执行。丝分裂：事务处理产生两个新事务。汇集：事务的不同活动可以汇集一处。吸收：一个事务可以被另一个事务吸食。结合：两个事务结合后产生一个新事务。

### 6.2.3 其他功能测试技术简述

#### 6.2.3.1 等价类法

1.等价类法（属于黑盒测试）：1）是根据程序的I/O特性，将程序的输入划分为有限个等价区域，使得从每个区域内抽取的代表性数据进行的测试等价于该区段内任何数据的测试。

2）对于每个输入条件存在着程序有效输入的有效等价类和对程序错误输入的无效等价类。

2.划分等价类的一些原则

1）如果某个输入条件规定了输入数据的取值范围，则可以确立一个有效等价类和两个无效等价类。

2）如果某个输入条件规定了输入数据的个数，则可以确立一个有效等价类和两个无效等价类。

3）如果某个输入条件规定了输入数据的一组可能取的值，每一个输入值就是一个有效等价类，一个无效等价类。

4）如果某个输入条件是一个布尔值，则可以划分一个有效等价类和一个无效等价类。

5）如果某个输入条件规定了必须符合的条件，则可以划分一个有效的等价类和一个无效的等价类。

6）若已划分的某个等价类中各元素在程序中的处理方式不同，则应将此等价类进一步划分为更小的等价类。

#### 6.2.3.2 边界分析法

边界分析法：是一种根据I/O边界等价类上或仅靠边界的条件，选择测试用例的更有效的方法。

#### 6.2.3.3 因果图法

1.因果图法：因一输入条件和果一输出结果，通过因果图将功能说明转换成一张判定表，然后为每种输入条件组合设计测试用例，着重检查输入条件的组合。

2.原因和原因之间的约束关系

1）E（互斥）：只能有一个成立

2）I（包含）：至少有一个条件成立

3）O（唯一）：有且仅有一个成立

4）R（要求）：当a出现时，b必须出现

5）M（屏蔽）：当a=1时，b必须是0，当a=0时，b的值不确定。

3.用因果图生成测试用例的步骤

步骤1：找出模块的原因（输入条件或输入条件的等价类）

步骤2：分析原因与结果之间的对应关系，画出因果图

步骤3：在因果图上标识出一些特定的约束和限制条件

步骤4：把判定表的每一列作为依据，设计测试用例。

## 6.7 软件测试步骤

1.软件测试步骤：软件测试是按照与系统开发相反的方向来进行的，依次为：单元测试、集成测试、有效性测试、系统测试

2.单元测试（Unit Testing）又称模块测试（Module Testing），或模块分调用于测试单个程序模块，确定模块的逻辑和功能是否正确。

单元测试采用白盒测试技术

1）单元测试要考虑模块的4个特征：

模块接口；局部数据结构；重要的执行路径；错误执行路径

2）单元测试还需要开发驱动模块和承接模块：

驱动模块：调用被测试模块的模块；承接模块：被测试模块调用的模块

3.集成测试：集成测试（Integration Testing）用来测试模块之间接口的正确性，也即模块之间的数据和控制传递，集成测试与单元测试平行进行。

1）两种策略：自顶向下的集成测试：需要设计承接模块；自底向上的集成测试，需要设计驱动模块；每加入新的模块，还要进行回归测试（重新测试）。

1. 有效性测试：目标是发现软件实现的功能与需求规格说明书不一致的错误，采用黑盒测试技术。

# 软件生存周期过程与管理

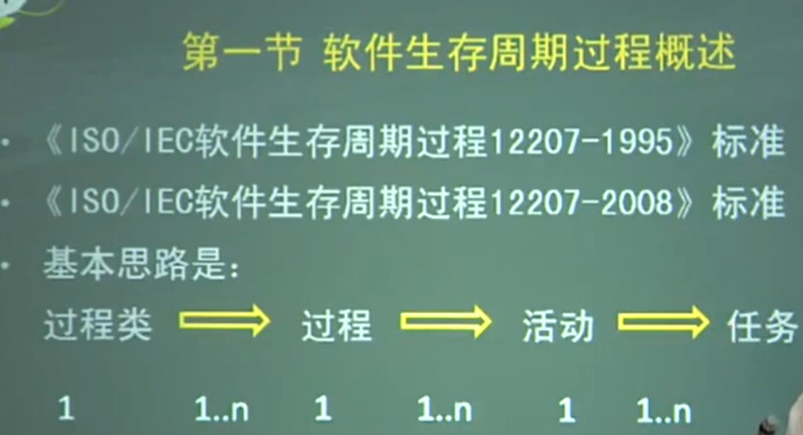
## 7.1 软件生存周期过程概述（1）

1.《ISO/IEC软件生存周期过程12207-1995》标准

2.《ISO/IEC软件生存周期过程12207-2008》标准

3.基本思路：

过程类 过程 活动 任务（完成目标的最小单元）



## 7.2 软件生存周期过程概述（2）

一、《ISO/IEC软件生存周期过程12207-1995》标准

软件生存周期过程的分类（属于过程类）：

1）基本过程；2）支持过程；3）组织过程

1.基本过程（属于过程）

1）指那些与软件生产直接相关的活动集

2）包括5个过程：

获取过程（需求方）；供应过程（软件供应方）；开发过程；运行过程；维护过程；

3）开发过程包含的活动（属于活动）

（1）过程实现；（2）系统需求分析；（3）系统体系结构设计；（4）软件需求分析；（5）软件按体系结构设计；（6）软件详细设计；（7）软件编码和测试；（8）软件集成；（9）软件合格性测试；（10）系统集成；（11）系统合格性测试；（12）软件安装；（13）软件验收支持；

过程实现包含的任务（属于任务）

选择合适的生存周期模型；选择相应的标准、方法、工具和程序设计语言；制定实施开发计划；可以使用一些非交付的软件项。

2.支持过程（属于过程类）：是指有关各方按他们的目标所从事的一系列支持活动集。支持活动有助于提高系统或软件产品的质量。

1）文档过程；2）配置管理过程；3）质量保证过程；4）验证过程；5）确认过程；6）联合评审过程；7）审计过程；8）问题解决过程；

3.组织过程（属于过程类）：与软件生产组织有关的活动集。

1）包括4个过程：管理过程；基础设施过程；培训过程；改进过程

二、《ISO/IEC软件生存周期过程12207-2008》标准

1）“2个过程类、7个过程组、43个过程”。

2）2个过程类：“系统语境的过程”“软件开发的过程”

3）7个过程组：协议过程组；项目过程组；技术过程组；组织上项目使能过程组；以上属于系统语境的过程类；软件实现过程组；软件支持过程组；软件复用过程组；以上属于软件开发的过程类。

## 7.3 过程描述

一、《ISO/IEC软件生存周期过程12207-2008》标准

1.供应过程

1）意图：为获取方提供满足所协商需求的产品或服务

2）活动和任务

活动1：机遇标识

活动2：供应方投标

活动3：合同协商

活动4：合同执行

3）结果

1）标识了产品或服务的获取方

2）对获取方的要求作了必要的响应

3）建立了获取方和供应方之间的协议

4）供应方开发了满足所协商需求的产品/服务

5）按所协商的需求向获取方交付了相应的产品/服务

6）按所协商的需求安装了产品

2.软件实现过程

1）意图：把已规约的行为、接口和实现约束转换为一些动作，创建称为“软件项”的软件产品和服务作为系统元素。

2）任务和活动

活动：软件实现策略

3）结果：定义了实现策略；标识了有关设计方面的实现技术约束；实现了一个软件；按提供协议，把该软件打包成一个软件并存储。

3.软件需求分析过程

1）意图：建立系统软件部分的需求

2）活动和任务：

任务1：建立软件需求和文档；

任务2：评估软件需求，并建立相应的评估结果文档

任务3：按软件复审过程进行软件需求复审

3）结果：需求已分配给系统的软件元素；已分析软件需求的正确性和可测性；已了解软件需求对运行环境的影响；在软件需求和系统需求之间建立了一致性和可跟踪性；已定义了实现软件需求的优先级别；软件已得到批准并按需求进行了调整；针对软件需求的更改对成本、进度和技术影响，已进行了相应的评估。建立了软件需求的基线，并与有关部门进行了沟通。

4.软件结构体系设计

1）意图：为软件的实现和按需求进行验证提供设计方案。

2）活动和任务：软件体系结构设计

3）结果：开发一种软件体系结构设计，描述实现该软件需求的软件项；设计每一软件项的内部接口和外部接口；

5.软件验证过程

1）意图：证明软件产品是否满足了所规约的需求。

2）活动和任务：过程和实现；验证；

3）结果：开发并实现了验证策略；标识了验证准则；执行了所需要的验证活动；标识并记录了缺点；给出了可用于客户和其它参与人员的验证结果

6.软件确认过程

1）意图：证实所期望使用的软件产品是否满足需求；

2）活动和任务：过程实现；确认；

3）结果：开发并实现了确认策略；标识了所有需要的软件工作产品的确认准则；执行了所需要确认活动；标识并记录了发现的问题；提供了证据证明：软件产品能够按照用户所期望的方式来使用；给出了可用于客户和其他参与人员的验证活动结果；

## 7.4 应用说明

是对标准“ISO/IEC系统与软件工程-软件生存周期过程12207-2008”的应用说明。

1.系统和软件

软件是整个系统的组成部分；区分系统需求分析和软件需求分析；

2.与《ISO/IEC系统生存周期15288》的关系，当系统中包括非常重要的非软件因素时，要应用《ISO/IEC系统生存周期15288》

3.组织层和项目层，项目可能由组织执行

4.过程之间的时序关系

1）没有明确过程、活动、任务之间的时间依赖的序列。

2）支持活动之间的迭代和再现

5.过程分解：把过程划分为一些小的“片段”；

6.生存周期模型和阶段

用生存周期模型对系统或软件产品的生存模型化

模型由阶段组成

7.剪裁：针对特定的情况，修改生存周期过程

## 7.5 软件生存周期模型（1）

软件生存周期模型的含义：是一个包含软件产品开发、运行和维护中有关过程、活动和任务的框架，覆盖了从该系统的需求定义到系统的使用终止。

作用：不但为软件开发确定了一些抽象层，还确定了每一抽象层之间的基本关系。

一、瀑布模型流程顺序：系统需求 软件需求 需求分析 设计 编码

测试 运行

1.瀑布模型的原理：1）自上而下具有相互衔接的固定顺序；2）每一阶段的输入，即工作对象以及本阶段的工作成果，作为输出传送到下一阶段。

2.瀑布模型的贡献：1）在决定系统怎么做之前存在一个需求阶段，它鼓励对系统做什么有一个规约。2）在系统构造之前有一个设计阶段，它鼓励规划系统结构。3）每一阶段都有评审，允许获取方和用户的参与。4）前一步作为下一步被认可的、文档化的基线。

3.瀑布模型的问题：1）要求客户能够完整、正确和清晰地表达他们的需求，并要求开发人员一开始就理解这一应用。

2）由于需求的不确定性，使设计、编码和测试阶段都可能发生延期，并且当项目接近结束时，出现了大量的集成和测试工作。

3）在开始的阶段中，很难评估真正的进度状态，并且直到项目结束之前都不能演示系统的功能。

4）在一个项目的早期阶段，过分地强调了基线和里程碑处的文档，并可能需要花费更多的时间用于建立一些用处不大的文档。

1. 增量模型

1）增量模型融合了瀑布模型的基本成分（重复应用）和原型实现的迭代特征，该模型采用随着日程时间的进展而交错的线性序列，每一个线性序列产生软件的一个可发布的“增量”。

2）前提：需求可结构化

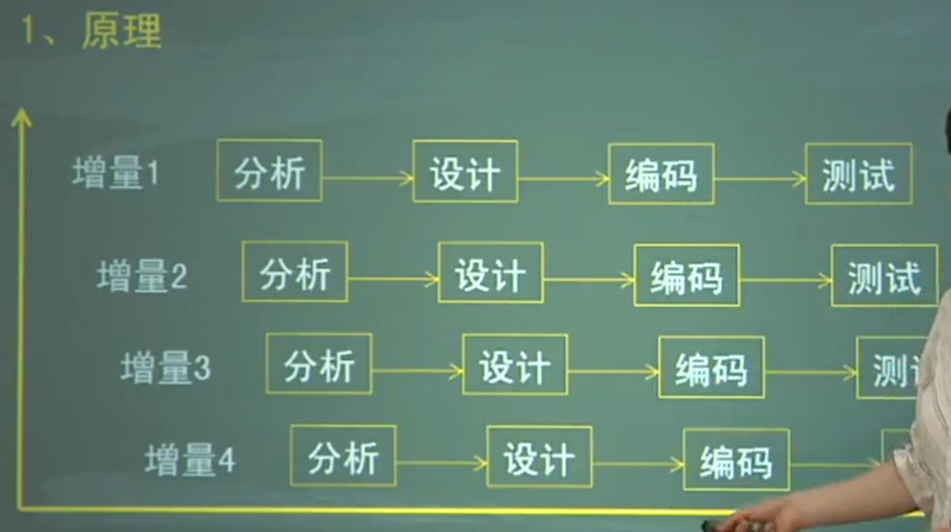
3）适用于“技术驱动”的软件产品开发。

2.增量模型的优点：1）第一个交付的版本所需的成本和时间较少；2）由于很快发布第一个版本，可以减少用户需求的变更；3）允许增量投资，即开始时只对一个或两个增量投资。

3.增量模型的缺点：1）如果没有对用户的变更要求进行规划，那么产生的初始增量可能会造成后来增量的不稳定。

2）如果需求不像早期思考的那样稳定和完整，那么一些增量就可能需要重新开发，重新发布。

3）由于进度和配置的复杂性，可能会增大管理成本，超出组织的能力



## 7.6 软件生存周期模型（2）

三、演化模型：1）演化模型是一种全局的软件（或产品）生存周期模型，属于迭代开发方法。

2）演化模型主要针对事先不能完整定义需求的软件开发。

3）该模型可以表示为：第一次迭代（需求->设计->实现->测试->集成）->反馈->第二次迭代（需求->设计->实现->测试->集成）->反馈->……

2.演化模型的优点：1）任何功能一经开发就能进入测试以便验证是否符合产品需求。2）帮助导引出高质量的产品要求。3）减少软件开发活动的盲目性。

4.演化模型的不足：很容易弱化需求分析阶段的工作

四、螺旋模型：1）螺旋模型是在“瀑布模型”和演化模型的基础上，加入两者都忽略的风险分析所建立的一种软件开发模型。2）螺旋模型强调风险分析，使得开发人员和用户对每个演化层出现的风险有所了解，继而做出应用的反应。3）因此特别适用于庞大、复杂并具有高风险的系统。

2.工作过程

1）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；

2）风险分析：分析评估所选方案，考虑如何识别和消除风险。

3）实施过程：实施软件开发和验证

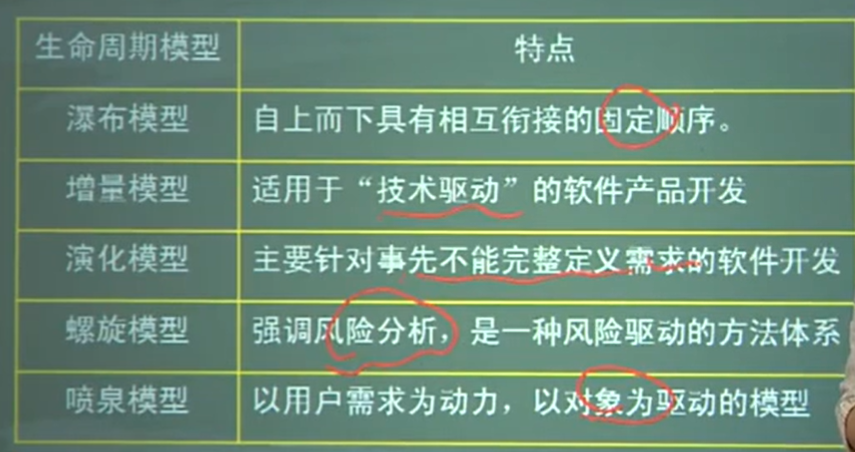
4）客户评估：评估开发工作，提出修正建议，制定下一步计划。

3.螺旋模型的特点：1）螺旋模型很大程度上是一种“风险驱动”的方法体系。

2）螺旋模型关注解决问题的基本步骤

五、喷泉模型：1）是一种以用户需求为动力，以对象为驱动的模型，主要用于采用对象技术的软件开发项目。2）喷泉模型认为，软件开发过程自下而上周期的各阶段是相互迭代和无间隙的特性。

1. 喷泉模型原理：喷泉模型主要用于支持面向对象技术的软件开发。



## 7.7 过程规划与管理

1）属于“组织上项目使能过程组”

2）包括四个环节：过程规划（P）过程检测（C）过程执行（D）过程调整（A）

1.过程建立

1）选择软件生存周期模型

2）细化所选择的生存周期模型

3）为每一个活动或任务标识合适的实例数目

4）确定活动的时序关系，并检查信息流

5）建立过程计划的文档

成果：项目的过程计划

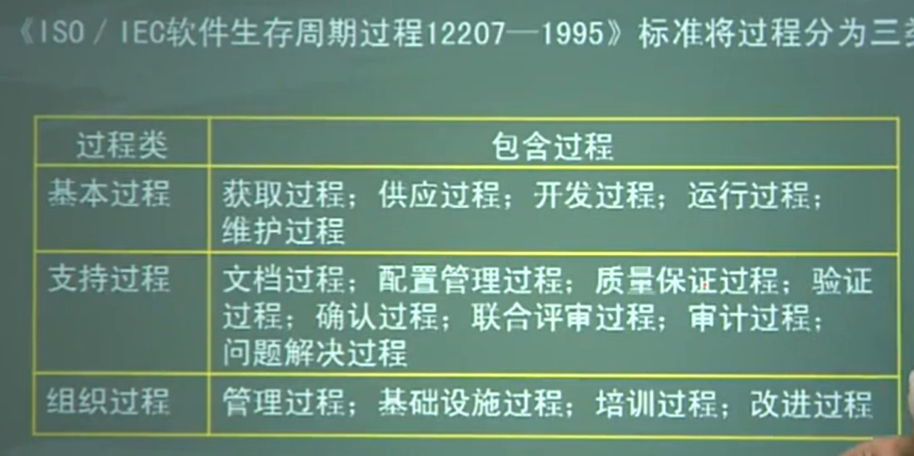
2.过程监控

1）软件生存周期过程的监控

2）软件生存周期过程改变所产生的影响的评估

3）改变的实施

4）实现改变



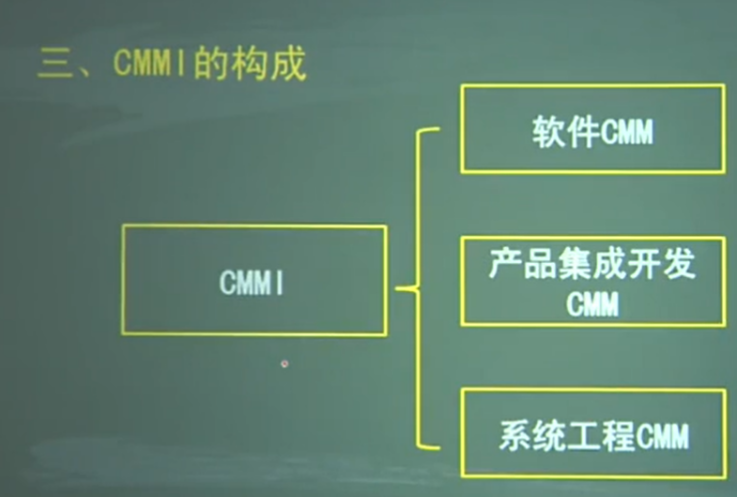
# 集成化能力成熟度模型CMMI

## 8.1 背景和愿景

1.CMMI的含义：Capability Maturity Model Integration for Development，集成化能力成熟度模型是由美国国防部与卡内基-梅隆大学和美国国防工业协会共同开发和研制的

2.CMMI目的：其目的是帮助软件企业对软件工程进行管理和改进，增强开发与改进能力，从而能按时地、不超预算地开发出高质量的软件。

3.CMMI的构成



4.CMMI的应用：过程途径的基本假设：系统或产品的质量高度受开发和维护中所使用的过程质量的影响。

质量支撑点：人员；规程和方法；工具和设备；

## 8.2 CMMI的模型部件

1.CMMI是一种过程改善框架

2.过程改善：是指认为设计的一个活动程序，其目的是改进组织的过程性能和成熟度，并改进这一程序的结果。

CMMI模型部件：

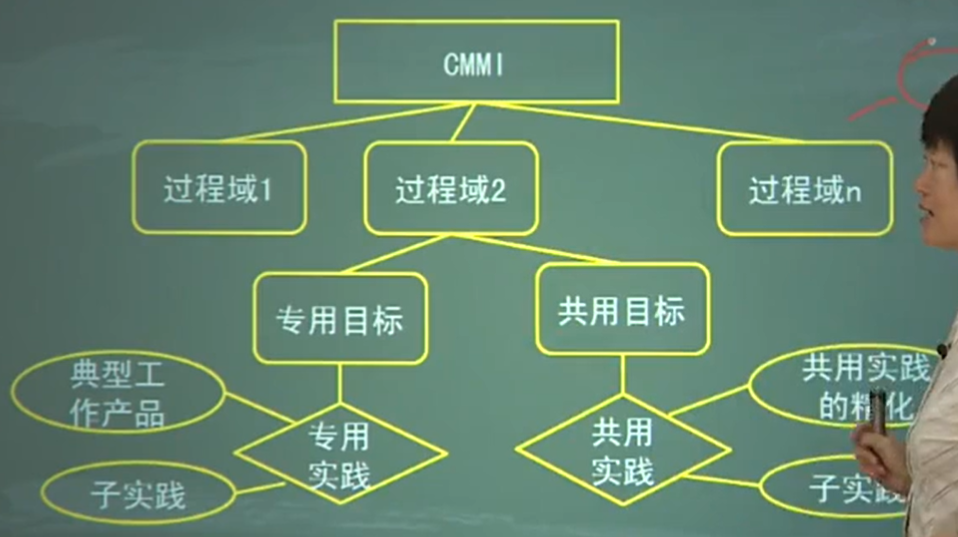
3.由一些过程域组成，过程域有自己确定的专用目标和公共目标。用矩形表示

4.每个专用目标和公共目标的实现，分别依赖一些实践。用菱形表示

5.每个专用实践有自己的子实践和确定的典型工作产品，符号： ，资料性部件。

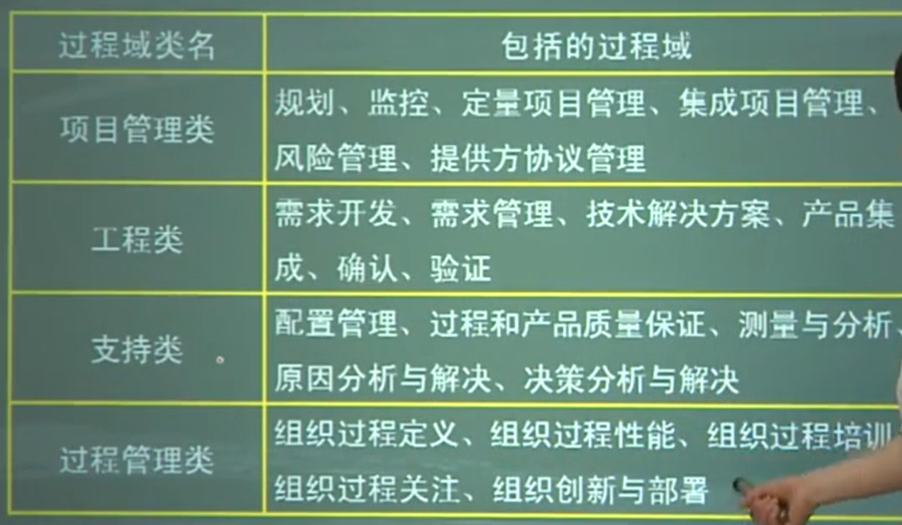
6.每个过程域还有意图陈述、简介性注释以及相关过程域。

7.CMMI层次结构



一、过程域

1）过程域：一个业务域中一束相关的实践，当它们一起得以实现时，就满足被认为对该过程域的改善具有重要作用的一组条件。



2）CMMI有22个过程域，分为四类，见左表所表：

二、专用目标

1）一个过程域中都有一个或多个专用目标。

2）描述该过程域必须呈现的一些独有特征

3）专用目标可用于帮助确定一个过程域是否得以满足

三、专用实践

1）对于达到专用目标是重要的活动。

2）期望以专用实践所描述的活动，会导致达到一个过程域的专用目标。

四、共用目标和共用实践

1）可用于多个过程域

五、典型工作产品

1）专用实践所产生的输出样品

【案例】专用实践“依据项目，监视项目规划参数的实际值”

2）典型工作产品是：重大偏差的记录

六、子实践

1）子实践是对专用实践、共用实践的详细描述。

【案例】

专用实践：针对已标识的问题，采取纠正措施。

子实践：针对已标识的问题，确定所需要的适当措施，并建立相应的文档。

七、共用实践的精华

1）为一个共用实践唯一地应用于一个过程域，提供了相关的指导。

八、意图描述

1）用来描述过程域的意图

【意图】

“组织过程定义”过程域的意图：建立并维护一组可用的组织过程资产和工作环境标准。

九、简介性解释

1.用来描述该过程域中所涉及的主要概念。

【案例】

在项目规划过程域中，规划从需求开始，定义产品和项目。

十、相关过程域

1）用来描述该过程域所引用的相关过程域

2）反映了过程域之间的关系

## 8.3 CMMI的等级

1.两种类型的等级

1）能力等级：是一种过程改善路径，该路径可使组织针对单一过程域不断改善该过程域。

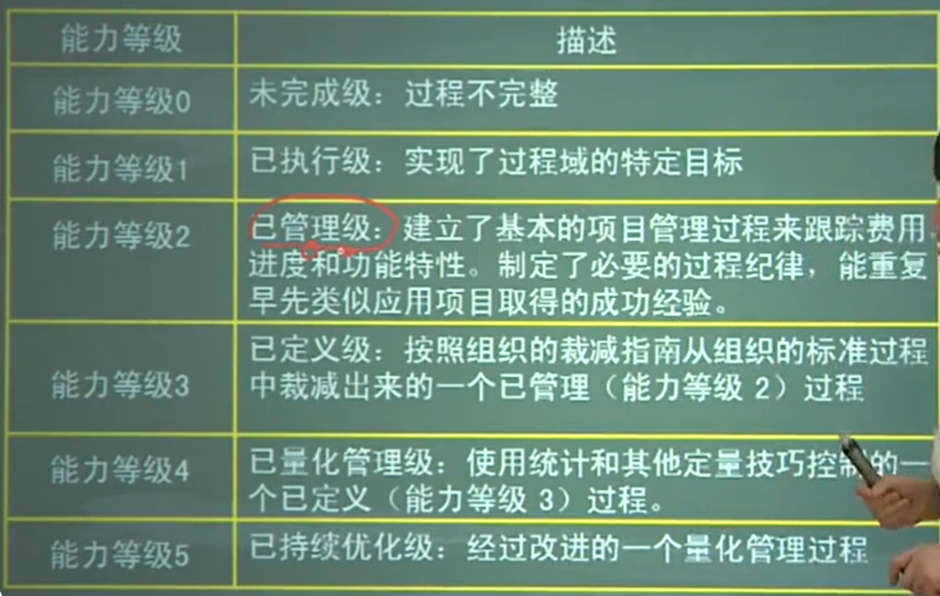
2）成熟度等级：是一种过程改善路径，该路径可使组织针对一组过程域不断改善一组相关的过程域。

一、能力等级

1.过程能力：1）遵循一个过程可达到的预期结果的程度。

2）表征组织对一个过程域的改善，是不断改善一个给定的过程域的一种手段。

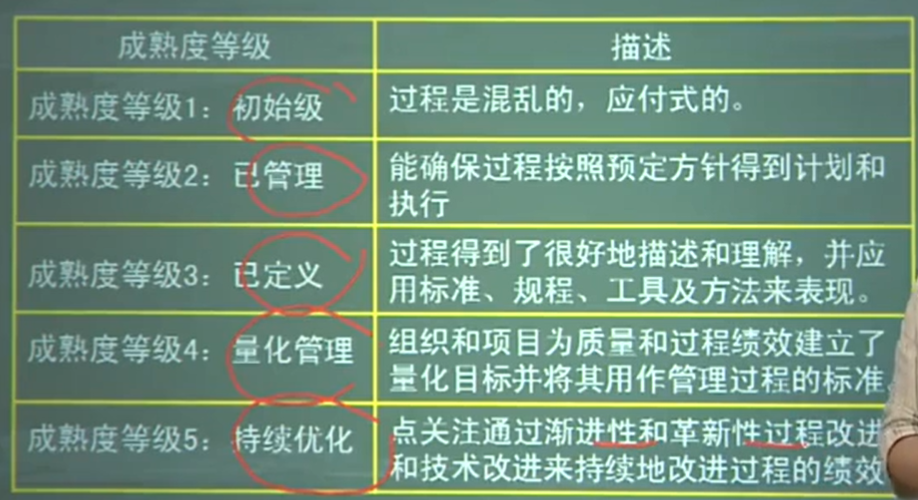
2.能力等级：包含一个共性目标及其相关的共性实践，它们与一个过程域相关联，能够改进组织同那个过程域相关联的过程；能力等级划分如下图，下一个等级是在上一个等级的基础上改善并优化



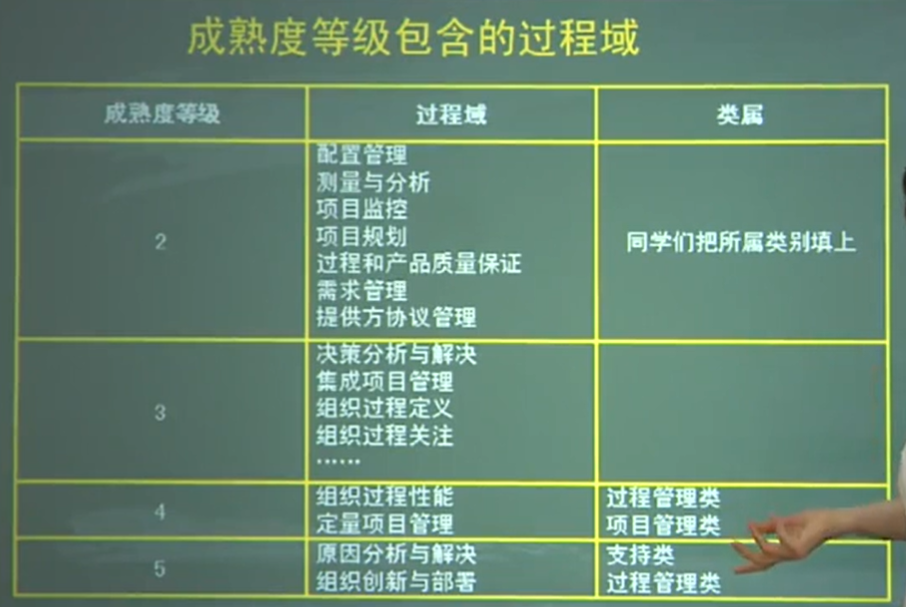
二、成熟度等级

1.成熟度等级是指达到预先定义的一组过程域所有目标的一种过程改善等级。一个成熟度等级是由预先定义的一个过程域集及其相关的一些专用实践和共用实践组成。

2.CMMI的阶段式表示模型定义了5个成熟度等级，在持续的过程改进上，每一等级都是构成下一阶段基础的一个层次，这些等级用1到5的数字表示



1. 成熟度等级包含的过程域



## 8.3 过程域举例