第一章　绪论

考点一：软件工程概念

1、软件工程：是应用计算机科学理论和技术以及工程管理原则和方法，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。

2、软件危机：软件产率、软件质量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。

3、软件工程概念的提出是倡导以工程的原理、原则和方法进行软件开发，以期解决出现的软件危机。

4、软件工程的发展可分为两个时期：20 世纪 60 年代末到 80 年代初、20 世纪 80 年代以来。前期主要研究系统实现技术；后者主要表现为软件质量和软件工程管理。

考点二：软件开发的本质

1、软件开发的本质概括为: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”。它涉及到两方面的问题：

（1）一是如何实现这样的映射，这是技术层面上的问题；

（2）二是如何管理这样的映射，以保障映射的有效性和正确性，这是管强层面上的问题。

2、计算机软件是计算机系统中的程序及其文档。

3、软件开发的本质可概括为：实现问题空间的概念；和处理逻辑到解空间的概念和处理逻辑之间的映射。实现这一映射的基本途径是系统建模。

4、在软件开发中，软件系统模型大体上分为两类：概念模型和软件模型。概念模型描述了系统是什么；软件模型描述了实现概念模型的软件解决方案。软件模型又分为设计模型、实现模型和部署模型等。

第二章　软件需求与软件需求规约

考点一：需求与需求获取

1、需求的基本性质：

（1）必要的：该需求是用户所要求的。

（2）无歧义的：该需求只能以一种方式解析。

（3）可测的：该需求可进行测试的。

（4）可跟踪的：该需求可从一个开发阶段跟踪到另一个阶段。

（5）可测量的：该需求是可测量的。

2、验证需求是不是歧义的，一般可采用需求复审。验证需求是不是可测的，可在标识任何所需要的数据和设施的基础上，开发一个测试概念。验证需求是不是可测量的，可通过检验一个特征是否存在但需要考虑设计、实现和测试阶段所发生的各种情况。

3、常用的初始需求发现技术包括：

（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。

（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/用户回答的方式，直接询问用户需要的是一个什么样的系统，可能会引起客户抵触。

（3）观察：通过观察用户执行其现行的任务和过程，了解系统运行环境。

（4）小组会：举行客户和开发人员的联系会议，与客户组织的一些代表共同发需求。

（5）提炼：复审技术文档，提取相关的信息。

4、问题空间理解：软件系统产品的需求分析工作中，面临的“三大挑战”的是问题空间理解、人与人之间的通信和需求的变化性。

5、设计约束是一种需求，它限制了软件系统或软件系统构件的设计方案的范围。例如：系统必须用 C++或其他面向对象语言编写，并且系统用户接口需要菜单；任取 1s，一个特定应用所消耗的可用计算能力平均不超过 50%。

考点二：需求规约

1、需求规约是一个软件项/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、需求规约般需要满足以下 4 个基本性质：

（1）重要性和稳定性程度：按需求的重要性和稳定性，对需求进行分级。

（2）可修改的：在不过多地影响其他需求的前提下，可以容易修改一个单一需求。

（3）完整的：没有被遗漏的需求。

（4）一致的：不存在互斥的需求。

3、需求规约的表达：

（1）非形式化的需求规约：以一种自然语言来表达需求规约。

（2）半形式化的需求规约：以半形式化符号体系来表达需求规约。

（3）形式化的需求规约：以一种基于良构数学概念的符号体系来编制需求规约。

4、需求规约的作用，其包含以下四点：

（1）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。

（2）对于产品/系统的设计，需求规约是一个正式的、受控的起始点。

（3）对于项目的其余大多数工作，需求规约是一个管理控制点。

（4）需求规约是创建产品验收计划和用户指南的基础。

第三章　结构化方法

考点一：结构化需求分析

1、数据流是数据的流动，表示形式例如：学生成绩 ⟶

2、加工是数据的变换单元，即它接受输入的数据，对其进行处理并产生输出，用 ◯ 表示。

3、数据存储是数据的静态结构，用于表达在分析中使用的、表达“结构化客体”的信息，用 ═══ 表示。

4、数据源是数据流的起点；数据潭是数据流的归宿地，用 □ 表示。

5、结构化分析方法给出了一种能表达功能模型的图形化工具是数据流图(DFD 图)。

6、结构化分析建模的基本步骤：

（1）建立系统环境图，确定系统语境；

（2）自顶向下，逐步求精，建立系统的层次数据流图；

（3）定义数据字典；

（4）描述加工。

7、定义数据结构的符号：

符　号	描　述	符　号	描　述
=	定义为	{ }	重复
+	顺序	m..n	边界
[|]	选择		

8、描述加工的工具有：结构化自然语言、判定表、判定树。

考点二：结构化设计——总体设计

1、在结构化方法中，HIPO 图应用在总体设计阶段，由 IPO 图和 H 图两部分组成的。

2、事务设计的基本步骤：

（1）设计准备，复审并精化系统模型；

（2）确定事务处理中心；

（3）设计系统模块结构图的顶层和第一层；

（4）自顶向下，逐步求精。

3、耦合是指不同模块之间相互依赖程度的度量。耦合程度从强到弱依次为：内容耦合、公共耦合、控制耦合、标记耦合、数据耦合。

（1）内容耦合：当一个模块直接修改或操作另一个模块的数据，或一个模块不通过正常入口而转入到另一

个模块时，这样的耦合称为内容耦合。

（2）公共耦合：两个或连个以上的模块共同引用一个全局数据项，这种耦合被称为公共耦合。

（3）控制耦合：一个模块通过接口向另一个模块传递一个控制信号，接收信号的模块根绝信号值进行适当的动作，这种耦合称为控制耦合。

（4）标记耦合：若一个模块 A 通过接口向两个模块 B 和 C 传递一个公共参数，那么成模块 B 和 C 之间存在一个标记耦合。

（5）数据耦合：模块之间通过参数来传递数据。

4、内聚是指内部各成分之间相互关联程度的度量。高内聚是指一个模块中各部分之间存在着很强的依赖；低内聚是指一个模块中各部分之间存在较少的依赖。

5、常见的内聚类型从低到高分别为：偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚、功能内聚。

（1）偶然内聚：一个模块的各成分之间基本不存在任何关系。

（2）逻辑内聚：几个逻辑上相关的功能被放在同一个模块中。

（3）时间内聚：一个模块完成的功能必须在同一时间内执行，但这些功能只是因为时间因素关联在一起。

（4）过程内聚：一个模块内部的处理成分是相关的，而且这些处理必须以特定的次序执行。

（5）通信内聚：一个模块的所有成分都操作同一数据集或生成同一数据集。

（6）顺序内聚：一个模块的各个成分和同一个功能密切相关。

（7）功能内聚：最理想的内聚，模块的所有成分对于完成单一的功能都是本的。

6、在一个软件结果中，深度表示其控制的层数；宽度是指同一个层次上模块总数的最大值。模块扇出是指一个模块直接控制（调用）的下级模块数目；模块扇入表示有多少个上级模块直接调用它。

7、模块的作用域是指受该模块内一个判定所影响的所有模块的集合。

考点三：结构化设计——详细设计

1、详细设计的任务是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，包括一组例程和数据结构，从而精确地定义了满足需求所规约的结构。

2、详细设计的目标是将总体设计阶段所产生的系统高层结构映射为以这些术语所表达的低层结构，也是系统的最终结构。

3、结构化程序设计的基本控制结构包括顺序、选择和循环。

4、典型的详细设计工具有：程序流程图、盒图（N-S 图）、PAD 图。

第四章　面向对象方法——UML

考点一：表示客观事物的术语——类与对象

1、类是一组具有相同属性、操作、关系和语义的对象的描述。对象是类的一个实例。

2、通常把类表示为具有 3 个栏目的矩形，每个栏目分别代表类名、属性、操作，其中类名使用黑体字，以大写字母开始，位于第一栏的中央，第二栏是类所具有的属性，第三栏为属性具有的操作。

3、UML 通过在角色名前添加符号+、-、#和~来描述该关联的可见性。

4、+：公共可见的

5、-：对该关联之外的任何对象而言，该端的对象是不可见的

6、#：该端的对象只有另一端的“子孙”是可以访问的

7、~：在同一包中声明的类是可访问的

考点二：表示客观事物的术语——接口

1、接口是操作的一个集合，其中每个操作描述了类、构件或子系统的一个服务。

2、接口的表示形式：①采用具有分栏和关键字《interface》的矩形符号表示；②采用小圆圈和半圆圈来表示，左边的圈表示由类提供的接口，右边的半圆表示类需要的接口。

3、接口应该注意的问题：

（1）接口之间没有关联、泛化、实现和依赖，但可以参与泛化、实现和依赖。

（2）接口只可以被其他类目使用，而其本身不可以访问其他类目。

（3）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。

（4）接口不描述其中操作的实现，也没有属性和状态。

考点三：其他表示客观事物的术语

1、协作：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，交互各方之间的相互作用，提供了某种协作性行为（UML 中，协作用虚线椭圆表示）。在系统/产品建模中，可以通过协作来刻画一种由一组特定元素参与的、具有特定行为的结构。

2、用况：用况是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（UML 中，用况用实线椭圆表示）。在系统/产品建模中，用况一般用于模型化系统中的功能行为。

3、主动类：主动类是一种至少具有一个进程或线程的类，主动类能够启动系统的控制活动。在系统/产品建模中，主动类一般用于模型化系统中的并发行为。

4、构件：构件是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。在系统/产品建模中，构件一般用于表达解空间中可独立标识的成分（构件不能用于问题定义）

5、制品：制品是系统中包含物理信息的、可替代的物理部件。在软件开发中，制品通常用于代表有关源代码信息或运行时信息的一个物理打包。

6、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。

考点四：表达关系的术语

1、关联：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（links）的描述。链是对象之间具有特定语义关系的抽象，实现之后的链通常称为对象之间的连接。聚合是关联的一种特殊形式，表达的是一种“整体/部分”关系。

2、泛化：泛化是一般性类目和它的较为特殊性类目之间的一种关系，称为“is-a-kind-of”。泛化的四个约束：完整、不完整、互斥、重叠

3、细化：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的契约（细化表示为带空心三角形的虚线段）

4、依赖：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（使用有向虚线段表示依赖，箭头端为目标，另一端为源）

考点五：UML 模型表达格式

1、UML 图形化工具分为两类：结构图和行为图，前者用于表达系统或系 统成分的静态结构模型。

2、类图：类图是可视化地表达系统静态结构模型的工具，通常包含类、接口、关联、泛化、和依赖关系等。

3、用况图：用况图是一种表达系统功能模型的图形化工具。一个用况图通常包含六个模型元素：主题（Subject）、用况（Use cases）、参与者（Actor）、关联、泛化、依赖。

4、使用用况图可以为系统建模，描述软件系统行为的功能结构；也可以对业务建模，描述企业或组织的业务过程结构。业务模型和系统模型之间具有“整体/部分”关系，即业务模型是整体，而系统模型是业务模型的一个组成部分。

5、状态图：状态图是显示一个状态机的图，其中强调了从一个状态到另一个状态的控制流。

6、顺序图：顺序图是一种交互图，即由一组对象以及按时序组织的对象之间的关系组成，其中还包含这些对象之间所发送的消息。顺序图包含参与交互的对象、基本的交互方式（同步和异步）以及消息等。

第五章　面向对象方法——RUP

考点一：RUP 的特点

1、RUP 是一种软件开发的过程框架，它的突出特点是以用况(或 Use Case)为驱动、以体系结构为中心的迭代、增量式开发。

2、迭代、增量式开发是指通过开发活动的迭代，不断地产生相应的增量。在 RUP 中，规定了 4 个开发阶段：初始阶段、精化阶段、构造接管、移交阶段；每一阶段都有同样的工作流，即需求、分析、设计、实现和测试。

3、RUP 利用 UML 提供的术语和工具定义了需求获取层、系统分析层、设计层和实现层，并给出了实现各层模型之间映射的基本活动以及相关指导。

4、RUP 和 UML 之间的关系：

（1）RUP 和 UML 构成了一种特定的软件开发方法学；

（2）UML 作为一种可视化建模语言，给出了表达事物和事物之间关系的基本术语，给出了多种模型的表达工具；

（3）RUP 利用这些术语定义了需求获取层、系统分析层、设计层和实现层，并给出了实现各层模型之间映射的基本活动以及相关的指导。

考点二：需求获取

1、RUP 采用用况技术来获取需求。

2、业务对象模型：使用工作人员、业务实体、工作单元三个术语。

3、用况模型是软件和客户就需求而达成的一个共识，是系统分析、设计、实现以及测试的基本输入。

考点三：需求分析

1、分析类

（1）边界类：边界类用于规约系统与其参与者之间的交互，该交互一般涉及向用户/外部系统发出请求和从他们那里接受信息。

（2）实体类：实体类用于规约那些需要长期驻留在系统中的模型化对象以及与行为相关的某些现象。

（3）控制类：控制类用于规约基本动作和控制流的处理和协调，涉及向其他对象委派工作。

2、在 RUP 中，一个系统的分析模型是由一个分析系统定义的，该分析系统包含一组具有层次结构的包，每一个包中可包含一些分析类和用况细化[分析]；并且一些分析类和用况细化[分析]还可单独地出现在分析模型中，以凸显它们在系统体系结构方面的作用。

3、用况模型和分析模型：

用况模型	分析模型
使用客户语言来描述	使用开发者语言来描述
给出的是系统对外的视图	给出的是系统对内的视图
使用用况予以结构化，但给出的是外部视角下的系统结构	使用衍型类予以结构化，但给出的是内部视角下的系统结构
可以作为客户和开发者之间关于“系统应做什么，不应做什么”的制约	可以作为开发者理解系统如何勾画、如何设计和如何实现的基础
在需求之间可能存在一些冗余、不一致和冲突等问题	在需求之间不应存在冗余、不一致和冲突等问题
捕获的是系统功能，包括在体系结构方面具有意义的功能	给出的是细化的系统功能，包括在体系结构方面具有意义的功能
定义了一些进一步需要在分析模型中予以分析的用况	定义了用况模型中每一个用况的细化

考点四：RUP 的实现和测试

1、RUP 的部署模型包含节点和主动类到节点的初始映射。

2、RUP 测试活动包括计划测试、设计测试、实现测试、执行集成测试、执行系统测试和评价测试

3、根据 RUP 测试活动，输入为测试用况、测试过程、实现模型，活动为实现测试，执行者为构件工程师，则输出为测试构件。

第六章　软件测试

考点一：软件测试目标与软件测试过程模型

1、软件评估可分为静态评估和动态评估。

2、软件测试的首要目标是预防错误。

3、软件测试的第二目标是发现错误。

4、软件测试和软件调试相比，在目的、技术和方法等方面都有着很大区别：

（1）测试从一个侧面证明程序员的“失败”。调试是为证明程序员的正确。

（2）测试从已知条件开始，使用预先定义的程序且有预知的结果，不可预见的仅是程序是否通过测试。调试是以不可知的内部条件开始，结果很难预见。

（3）测试是有计划的，并要进行测试设计。调试是不受时间约束的。

（4）测试是一个发现错误、改正错误、重新测试的过程。调试是一个推理过程。

（5）测试的执行是有规程的。调试的执行往往要求程序员进行必要推理。

（6）测试经常由独立的测试组在不了解软件设计的条件下完成的。调试必须了解详细设计的程序员完成。

（7）大多数测试的执行和设计可由工具支持。调试时，程序员能利用的工具主要是调试器。

5、软件测试是一个有程序的过程，包括测试设计、测试执行以及测试结果比较等。

考点二：软件测试技术

1、软件测试技术分为两大类：一类是白盒技术，又称为结构测试技术，典型的是路径测试技术；另一类是黑盒技术，又称为功能测试技术，包括事务处理流程技术、状态测试技术、定义域测试技术等。白盒测试技术依据的是程序的逻辑结构；而黑盒测试技术依据的是软件行为的描述。

2、测试度量从低到强排列为：语句覆盖<分支覆盖<条件覆盖<条件组合覆盖<路径覆盖。

（1）路径覆盖：执行所有可能穿过程序控制流程的路径。

（2）语句覆盖：至少执行程序中所有语句一次。

（3）分支覆盖：至少将程序中的每一个分支执行一次。

（4）条件覆盖：每个判定中的所有可能的条件取值至少执行一次。

（5）条件组合覆盖：是指设计足够的测试用例，使每个判定中的所有可能的条件取值组合至少执行一次（只要满足了条件组合覆盖，就一定满足分支覆盖）

3、划分等价类的规则：

（1）如果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）如果输入条件规定了输入值数据的个数，则可划分一个有效等价类和多个无效等价类；

（3）如果输入条件规定了输入数据的一组可能取的值，而且程序可以对每个输入值分别进行处理，则可为每一个输入值确立一个有效等价类和一个无效等价类；

（4）如果输入条件是一个布尔量，则可以确立一个有效等价类和一个无效等价类；

（5）如果输入条件规定了必须符合的条件，则可划分一个有效等价类和一个无效等价类；

（6）若在已知划分的等价类中各元素在程序处理中的方式不同，则应将该等价类进一步的划分为更小的等价类。

4、边界值分析与等价类划分技术的区别：

（1）边界值分析与等价类划分技术的区别在于：边界值分析着重边界的测试，应选取等于、刚刚大于或刚刚小于边界的值作为测试数据；

（2）而等价类划分是选取等价类中的典型值或任意值作为测试数据。

5、因果图方法生成测试用例的基本步骤：

（1）通过对软件规格说明书的分析，找出一个模块的原因和结果，并给每个原因和结果赋予一个标识符。

（2）分析原因与结果之间以及原因与原因之间对应的关系，并画出因果图。

（3）在因果图上标识出一些特定的约束或限制条件。

（4）把因果图转换成判定表。

（5）为判定表的每一列设计测试用例。

考点三：软件测试步骤

1、在软件工程测试中应综合运用测试技术，并且应实施合理的测试序列：单元测试，集成测试，有效性测试和系统测试。

2、单元测试主要检验软件设计的最小单元——模块，该测试以详细设计文档为指导，测试模块内的重要控制路径。一般来说，单元测试往往采用白盒测试技术。提高模块的内聚程度，可简化单元测试。

3、在单元测试期间，通常考虑模块的一下 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。

4、在单元测试中，由于模块不是一个独立的程序，必须为每个模块单元测试开发驱动模块和承接模块。

5、集成测试的目标是为了发现与接口有关的错误。

6、有效性测试的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。有效性测试通常采用黑盒测试技术。

7、单元测试关注的是每个独立的模块；集成测试关注的是模块的组装；有效性测试关注的是检验是否符合用户所见的文档。

第七章　软件生存周期过程与管理

考点一：软件生存周期过程

1、在标准《ISO/IEC 软件生存周期过程 12207—1995》中，按过程主体把软件生存周期过程分为基本过程、支持过程和组织过程。

2、基本过程是指那些与软件生产直接相关的活动集。主要分为 5 个过程：获取过程、供应过程、开发过程、运行过程和维护过程。

3、组织过程是指那些与软件生产组织有关的活动集。主要分为以下过程：管理过程、基础设施过程、培训过程和改进过程。

考点二：过程描述

1、供应过程：为获取方提供满足所协商需求的产品或服务。

2、软件实现过程：为了生产一个已规约系统的元素，作为一个软件产品或服务而实现的。

3、软件需求分析过程：建立系统软件部分的需求。

4、软件体系结构设计：为软件的实现及其可以按需求进行验证，提供一种设计。

5、软件验证过程：证实一个过程或项目的每一软件工作产品/服务是否正确地反映了所规约的需求。

6、软件确认过程：证实所期望使用的软件工作产品是否满足需求。

7、验证就是证实一个过程或项目的每一软件工作产品/服务是否正确地反映了所规约的需求；

8、验证和确认以及它们的区别：

（1）验证就是证实一个过程或项目的每一软件工作产品/服务是否正确地反映了所规约的需求；

（2）确认就是证实所期望使用的软件工作产品是否满足其需求；

（3）区别：验证是通过提供的客观证据，证实规约的需求是否得以满足；确认是通过提供的客观证据，证实有特定期望的使用或应用的需求是否得以满足。

考点三：软件生存周期模型

1、软件生存周期过程、软件生存周期模型、软件项目过程管理之间的关系：

（1）软件生存周期过程回答软件开发需要做哪些工作；

（2）软件生存周期模型回答软件开发活动或任务如何组织；

（3）软件项目过程管理回答软件过程如何管理；

（4）软件生存周期过程是软件生存周期模型和软件项目过程管理的基础；

（5）软件生存周期模型为软件项目过程管理提供支持。

考点四：瀑布模型

1、瀑布模型将软件生存周期的各项活动规定为按固定顺序而连接的若干阶段工作，形如瀑布流水，最终得到软件产品。瀑布模型适用于需求被清晰定义的项目。

2、瀑布模型规定了各开发阶段的活动，并且自上而下具有相互衔接的固定顺序，还规定了每一阶段的输入以及本阶段的工作成果作为输出传到下一阶段。

3、瀑布模型各开发阶段的活动：系统需求、软件需求、需求分析、设计、编码、测试和运行。

4、瀑布模型存在的问题主要是：

（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要理解这一-应用。

（2）由于需求的不稳定性，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能力。

（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。

考点五：增量模型

1、增量模型意指需求可以结构化分组，形成一个个增量，并形成一个结构，可见该模型有一个前提，即需求可结构化。增量模型第一个可交付版本所需要的时间和成本较少，可减少用户需求的变更，减少由增量引入带来的风险。

2、优点：增量模型第一同交付版本所需要的时间和成本较少。可以减少用户需求的变更。允许增量投资，即在项目开始时可以仅对一个或两个增量投资。

3、缺点：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。如果需求不像早期思者的那样稳定和完整，那么一些增量就可能需要重新开发.重新发布。用的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。

考点六：演化模型

1、演化模型本质上是迭代的；可以很容易适应需求的变化；通常不会抛弃所产生的系统。

2、演化模型是一种迭代、增量式开发模型。在用户提出待开发系统的核心需求的基础上，软件开发人员按照这一需求，首先开发一个核心系统并投入运行，以便用户能够有效提出反馈，接着软件开发人员根据用户反馈，实施开发的迭代过程，每次迭代均由需求、设计、编码、测试、集成等阶段组成，通过增加或修正，产生软件产品的增量，最终完成软件产品的开发。

3、演化模型显式地把需求获取扩展到需求阶段，在一定程度上可减少软件开发活动的盲目性。

4、该模型主要针对事先不能完整定义需求的软件开发的，通过不断的选代、增量开发，最终得到软件产品。

5、同螺旋模型相比，演化模型主要缺少风险分析。

考点七：螺旋模型

1、螺旋模型是在瀑布模型和演化模型的基础上，加入两者所忽略的风险分析所建立的一种软件开发模型。

2、螺旋模型沿着螺旋线，经历制定计划，风险分析，实施工程，客户评估等 4 个方面的活动，自内向外每旋转一圈便产生一个更为完善的新版本。

3、该模型适应的情况：项目的开发风险很大或客户不能确定系统需求。

考点八：喷泉模型

1、喷泉模型体现了软件创建所固有的迭代和无间隙的特征。

2、喷泉模型表明了软件活动需要多次重复。

3、喷泉模型主要用于面向对象技术的软件开发方法。

第八章　集成化能力成熟模型（CMMI）

考点一：背景与原理

1、集成化能力成熟度模型（CMMI）集成了软件 CMM、产品集成开发 CMM 和系统工程 CMM 等 3 个模型。

2、CMMI 模型基于过程途径思想，通过过程把软件质量的 3 个支撑点：受训的人员、规程和方法、工具和设备进行集成，以开发所期望的系列产品。

3、CMMI 紧紧围绕开发、维护和运行，把经过证明的最佳实践放在一个结构中。

考点二：CMMI 的模型部件

1、专用目标可用于帮助确定一个过程域是否得以满足；意图陈述用来描述改过程域的意图。

2、过程域有自己的确实专用目标、共用目标是 CMMI 必要的模型部件，用符号  表示

3、专用实践和共用实践作为期望的 CMMI 部件，用符号  表示

4、实践、典型工作产品和有关该共用实践的精化等 CMMI 资料性部件，用符号  表示

5、每个过程域还有意图陈述、简介性注释以及相关过程域，也用符号  表示

考点三：CMMI 等级

1、CMMI 模型支持的两种过程改善路径：

（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）成熟度等级也是一种过程改善路径，该路径可使组织通过关注一组过程域不断改善一组相关的过程域；

（3）这两种等级还可用于评定活动、估算，作为过程评估的结果。

2、CMMI 中，针对每个过程域共 6 个能力等级：

（1）0 级：未完成级

（2）1 级：已执行级

（3）2 级：已管理级

（4）3 级：已定义级

（5）4 级：已定量管理级

（6）5 级：持续优化级

3、CMMI 的能力等级和成熟度等级在概念上是互补的，区别在于能力等级是用来表征组织对单个过程域的改善。