软考系统架构设计师教程考点精讲（四）

软考系统架构设计师 属于软考中的一项高级资格考试， 考试分综合知识、 案例分析和论文 3 个科目。系统架构设计师考试作为一项高级资格考试， 有一定的考试难度，那么该如何备考才能顺利通过考试呢？面对系统架构设计师教程无从下手的同学， 希赛为您准备了几个重要的教程章节考点精讲， 希望对您的学习有所帮助。

第四章

4.1 软件开发方法

4.1.1 软件开发生命周期

传统的软件生命期是指软件产品从形成概念 (构思)开始，经过定义、开发、使用、维护、废弃，的全过程。

可以把软件生命期划分为软件定义、 软件开发、 软件运行与维护， 三个阶段。

1、软件定义时期

1.问题定义，目标系统“ 是什么 ”，系统的定位以及范围。

2.可行性研究，技术可行性、经济可行性、操作可行性、社会可行性。

3.需求分析，确定软件系统的功能需求、性能需求、运行环境的约束，写出需求规格说明书、软件系统测试大纲、用户手册概要。

充分理解用户的需求， 并以书面形式写出规格说明书， 这是以后软件设计和验收的依据 ;用户也许很难一次性说清楚系统应该做什么。

系统分析员、软件开发人员、用户，共同完成，逐步细化、一致化、完全化等。

软件需求规格说明 SRS，内容可以有系统 (或子系统)名称、功能描述、 接口、

基本数据结构、性能、设计需求、开发标准、验收原则等。

2、软件开发时期

软件开发时期就是软件的设计与实现，　概要设计、详细设计、编码、测试等。

概要设计是在软件需求规格说明的基础上，建立系统的总体结构　(含子系统的划分)和模块间的关系，定义功能模块及各功能模块之间的关系。

详细设计对概要设计产生的功能模块逐步细化，　包括算法与结构、数据分布、数据组织、模块间接口信息、用户界面等，写出详细设计报告。

测试可分成单元测试、集成测试、确认测试、系统测试等。通常把编码和测试称为系统的实现。

3、软件运行和维护

软件维护就是尽可能地延长软件的寿命，　没有维护的价值时，　宣告退役，　软件的生命结束。

4.1.2　软件开发模型

软件生存周期模型又称软件开发模型或软件过程模型，模型的特点是简单化，是软件开发实际过程的抽象与概括。

为软件工程管理提供里程碑和进度表，　为软件开发过程提供原则和方法。　软件过程有各种各样的模型。

1、瀑布型

瀑布型的特点是因果关系紧密相连，　前一个阶段工作的结果是后一个阶段工作的输入，前一个阶段的错漏会隐蔽地带到后一个阶段，　每一个阶段工作完成后，都要进行审查和确认，

它的出现有利于人员的组织管理，有利于软件开发方法和工具的研究。

2、原型模型

根据用户提出的软件系统的定义， 快速地开发一个原型， 包含目标系统的关键问题和反映目标系统的大致面貌。

三种途径：

利用模拟软件系统的人机界面和人机交互方式。

真正开发一个原型。

找来一个或几个正在运行的类似软件进行比较。

实际工作中， 由于各种原因， 大多数原型都废弃不用， 仅仅把建立原型的过程当作帮助定义软件需要的一种手段。

用户对系统模糊不清，无法准确回答目标系统的需求。

经过对原型若干次修改，应该收敛到目标范围内，否则可能会失败。

对大型软件来说，如果没有现成的，就不应该考虑用原型法。

3、螺旋模型

是生命周期模型与原型模型的一个结合，分成多个阶段，每一个阶段都由 4 部分组成：

1.目标设定，指定对过程和产品的约束，并且制订详细的管理计划。

2.风险分析，制订解决办法。

3.开发和有效性验证，即开发软件产品。

4.评审，确定是否需要进入螺线的下一次回路。

增加一周， 软件系统就生成一个新版本， 系统应该尽快地收敛到用户允许或可以接受的目标范围内。

该模型支持大型软件开发， 适用于面向规格说明、 面向过程、 面向对象的软

件开发方法，也适用于几种开发方法的组合。

4、基于可重用构件的模型

把软件工程项目所创建的构件不断地积累和存储在一个构件库中，系统将依赖构件的健壮性。

5、基于面向对象的模型

构件重用是非常重要的技术之一。一方面进行构件开发，另一方面进行需求开发，快速建立 OOA、OOD 原型，由重用构件组装而成，甚至通过组装可重用的子系统而创建更大的系统。

6、基于四代技术的原型

四代语言完全不用变成方式来构造应用系统，而是利用一些生成器。

与通常的软件工程环境或计算机辅助软件工程不同，只侧重于支持应用软件开发过程中的设计阶段和实现阶段，特别是支持界面以及与界面有关的处理过程。

4.1.3 敏捷方法

1、敏捷方法的特点

敏捷方法是“适应性”而非“预设性”的，重型方法在计划制定完成后拒绝变化，而敏捷方法则欢迎变化。

“面向人的”而非“面向过程的”

传统的软件开发方法的基本思路一般是只要图纸设计得合理并考虑充分，施工队伍可以完全遵照图纸顺利构造。

但是，一些设计错误只能在编码和测试时才能发现。

传统正规开发方法是个体不重要，角色才是重要的，尽量减少人的因素对开

发过程的影响，但是敏捷方法正好相反。

管理人员已经脱离实际开发活动相当长的时间了，　如此设计出来的开发过程是难以为开发人员所接受的。

只有在第一线的开发人员才能真正掌握和理解开发过程中的技术细节，　所以技术方面的决定必须由他们来做出。

敏捷方法特别强调相关人员之间的信息交流。　因为项目失败的原因最终都可以追溯到信息没有及时准确地传递到应该接受它的人。

特别提倡直接的面对面交流，交流成本远远低于文档的交流。

按照高内聚、松散耦合的原则将项目划分为若干个小组，以增加沟通。

2、敏捷方法的核心思想

1.适应性型，利用变化来发展。

2.以人为本，在无过程控制和过于严格繁琐的过程控制中取得一种平衡，以保证软件的质量。

3.迭代增量式的开发过程，发行版本小型化，根据客户需求的优先级和开发风险，制订版本发行计划。

3、敏捷方法的含义及其特征

重型方法注重开发文档的完备和充分性　;而敏捷方法认为最根本的文档应该是源码。

4、敏捷方法的适用范围

实际上，满足工程设计标准的唯一文档是源代码清单。

敏捷方法比较适合需求变化比较大或者开发前期对需求不是很清晰的项目。

敏捷方法对设计者、开发者、客户之间的有效沟通和及时反馈要求比较高，

不易在开发团队比较庞大的项目中实施。

5、敏捷方法的主要内容

四个核心价值观：沟通、简单、反馈、勇气。

简单：只要满足当前功能需求，不做假象设计。

勇气：用于抉择，用于实践，用于重构。

12 条实践规则：简单设计、测试驱动、代码重构、结对编程、继续集成、现场客户、 开发版本小型化、 系统隐喻、 代码集体所有制、 规划策略、 规范代码、40 小时工作机制。

6、主要敏捷方法简介

极限编程

水晶系列方法

开放式源码，任何人发现 Bug 都可以将补丁发给维护者。

SCRUM

Coad 的功用驱动开发方法：短时迭代阶段和可见可用的功能，一个迭代周期一般为两周，编程人员分为类程序员、首席程序员。

ASD 方法，猜测、合作、学习。

4.1.4 RUP

RUP 把软件开发生命周期划分为多个循环 (cycle)，每个 cycle 生成产品的一个新版本，每个 cycle 依次由 4 个连续阶段 (phase) 组成：

初始：定义最终产品视图和业务模型，并确定系统范围。

细化：制定工作计划及资源要求。

构造。

移交。

迭代并不是重复地做相同的事， 而是针对不同用例细化和实现， 每一个迭代都是一个完整的开发过程。

每个阶段结束前有一个里程碑 (milestone) 评估该阶段的工作。如果未能通过该里程碑的评估， 则决策者应该做出决定， 是取消该项目还是继续做该阶段的工作。

RUP 中的核心概念

角色 (Role)， who 的问题，某个人或一个小组的行为与职责。

活动 (Activity) ， how 的问题，是一个有明确目的的独立工作单元。

制品 (Artifact) ， what 的问题，是活动生成、创建、修改第一段信息。

工作流 (Workflow) ，when 的问题， 每个工作流产生一些有价值的产品， 并显示了角色之间的关系。

RUP 的特点

RUP 是用例驱动的、以体系结构为中心的、迭代和增量的软件开发过程。

用例驱动：需求分析、设计、实现、测试，都是用例驱动的。

以体系结构为中心： 刻画了系统的整体设计， 去掉了细节部分， 突出了系统的重要特征。

不依赖于具体语言，是软件设计过程的一个层次。

体系结构层次的设计问题包括：总体组织和全局控制、通讯协议、同步、数据存取、给设计元素分配特定功能、 设计元素的组织、 物理分布、 系统的伸缩性、性能等。

一个系统不可能在所有特性上都达到最优， 对于一个系统， 不同人员所关心

的内容也是不一样的， 对于不同类型的人员， 只需提供这类人员关心的视图即可。

分析和测试人员关心用例图， 最终用户关心逻辑视图， 程序员关心实现视图，系统工程师关心部署视图。

RUB 强调采用迭代和增量的方法来开发软件，每次迭代中，之考虑系统的一部分需求，每次增加一些新的功能实现。

好处：

早期就可以对关键的、影响大的风险进行处理。

可以提出一个软件体系结构来指导开发。

处理不可避免的需求变更。

可以较早地得到一个可运行的系统， 鼓舞开发团队的士气， 增强项目成功的信心。

更有效工作的开发过程。

没有一个项目会使用 RUP 中所有的东西，用用 RUP 时要裁剪，裁剪步骤：

1.确定本项目需要哪些工作流。

2.确定每个工作流要产出哪些制品。

3.确定四个阶段之间 (初始阶段、 细化阶段、 构造阶段、 移交阶段)如何演进。

4.确定每个阶段内迭代计划。

5.规划工作流内部结构。

4.1.5 软件系统工具

按软件过程活动将软件工具分为软件开发工具、 软件维护工具、 软件管理和软件支持工具。

软件开发工具有： 需求分析工具、 设计工具、编码与排错工具、 测试工具等。

需求分析工具，生成完整的、清晰的、一致的功能规范。功能规范是软件开发者和用户间的契约，　也是软件设计者的和实现者的依据。　正确、完整表达清晰的、无歧义的。

需求分析工具分为基于自然语言或图形描述的工具，　基于形式化需求定义语言的工具。

项目管理工具：项目的计划、调度、通信、成本估算、资源分配、质量控制等。

4.2 需求管理

需求最终文档经过评审批准后，则定义了需求基线　Baseline; 构筑了功能需求和非功能需求的一个约定　Agreement　。约定是需求开发和需求管理之间的桥梁。

需求管理是一个对系统需求变更、　了解和控制的过程，　初始需求导出的同时就启动了需求管理规划。

4.2.1 需求管理原则

过程能力成熟度模型　CMM ，指导软件过程改进，　5 个成熟级别，　6 个关键过程域 KPA。

一旦需求文档化了，　开发组和有关团队需要评审文档。　发现问题应与客户或者其他需求源协商解决。软件开发计划是基于已确认的需求。

绝不要承诺任何无法实现的事。

关键处理领域通过版本控制和变更控制来管理需求文档。　确保与新的需求保持一致。

4.2.2 需求规格说明的版本控制

版本控制是管理需求的一个必要方面，必须统一确定需求文档的每一个版本，当需求发生变更时，及时通知所有涉及人员。

为了尽量减少困惑、冲突、误传，应该仅允许指定的人员来更新需求。

清楚地区分草稿和文档定稿版本。

4.2.3 需求变更

迟到的需求变更会对已进行的工作产生非常大的影响。

如果每一个建议的需求变更都采用，该项目将可能永远无法完成。

需求文档应该精确描述要交付的产品。

项目负责人在信息充分的条件下做出决策。

变更成本计算应该包括需求文档的修改、系统修改的设计、实现的成本。

变更控制过程并不是给变更设置障碍，　相反，它是一个渠道和过滤器，　确保采纳最合适的变更，使变更产生的负面影响降到最低，变更过程应该做成文档。

绝不能删除或者修改变更请求的原始文档。

变更控制委员会只要能决定合适的人做正确的事就足够了，　在保证权威性的前提下应尽可能精简人员。

对每个变更权衡利弊做出决定。

“利”包括节省资金或额外收入、客户满意度、竞争优势、减少上市时间　;

“弊”是指增加开发费用、推迟交付日期、产品质量下降、减少功能、用户不满意。

变更总是有代价的，即使拒绝的变更也因为决策行为而耗费资源。

接受了重要的需求变更时，　为了适应变更情况要与管理部门和客户重新协商约定。推迟交货时间、增加人手、推迟实现尚未实现的较低优先级的需求，或质

量上进行折中。

要是不能获得一些约定的调整，应该把面临的风险写进风险计划中。

4.2.4 需求跟踪

需求、体系结构、其他设计部件、源代码模块、测试、帮助文件、文档等。

跟踪能力 (联系)链(traceability link)　是优秀需求规格说明书的一个特征，确保软件需求规格说明包括所有客户需求。

跟踪能力联系链记录了单个需求之间的父层、互连、依赖的关系。

不必拥有所有种类的跟踪能力联系链，要根据具体情况调整。

4.2.5 需求变更的代价和风险

只有在知道变更成本后才能做出理智的选择，　一个表面上很简单的变更也可能转变成很复杂的局面。

影响分析确定对现有系统做出是修改或者抛弃的决定，　创建新系统以及评估每个任务的工作量，进行影响分析的能力依赖于跟踪能力、　数据的质量、完整性。

4.3 开发管理

1、范围

可交付物、架设、约束条件的基础上准备详细的项目范围说明书，　是项目成功的关键。

2、时间

进度安排的准确程度可能比成本估计的准确程度更重要。　对于成本估计的偏差，可以靠重新定价或大量的销售来弥补成本的增加，

如果进度计划不能得到实施，　则会导致市场机会的丧失或用户不满意，　而且会使成本增加。

工作分解结构 Work Breakdown Structure WBS

4.3.1 配置管理文档管理

1、配置管理

配置项 Configuration Item CI ，

属于产品组成部分的工作成果，如需求文档、设计文档、源代码、测试用例等。

属于项目管理和机构支撑过程域产生的文档，如工作计划、项目质量报告、项目跟踪报告等。

每个配置项的主要属性有名称、标识符、文件状态、版本、作者、日期等。

2、文档管理

文档是影响软件可维护性的决定因素， 使用过程中必然会经受多次修改， 所以文档比程序代码更重要。

用户文档：主要描述系统功能和使用方法。

系统文档：描述系统设计、实现、测试等各方面内容。

软件文档应该满足下述要求：

1.如何使用

2.怎样安装和管理

3.需求和设计

4.实现和测试

说明用户操作错误时应该怎样恢复和重新启动。

4.3.2 软件开发的质量与风险

1、软件质量

IOS9000 对项目质量的定义：一组固有特性满足需求的程度。

质量与范围、 成本和时间， 是项目成功的关键因素， 通过范围管理转换隐含需求为项目需求。

质量低说明产品或服务存在问题，而低等级的产品或服务不一定存在问题，二者概念不同。

2、软件开发风险

认识不足或者没有足够的力量加以控制。

了解、掌握风险的来源、性质、发生规律，进而施行有效的管理。

或然性、不确定性、涉及到某种选择时，才成为有风险，以上三个是风险定义的必要条件，不是充分条件，具有不确定性的事件不一定是风险。

4.3.3 结构化分析与设计

结构程序设计较流行的定义为： 采用自顶向下逐步求精的设计方法和单入口单出口的控制构件。

自顶向下逐步求精的方法是： 先整体后局部， 先抽象后具体， 一般具有较清晰的层次。

仅使用单入口单出口的控制构件，具有良好的结构特征。

采用结构程序设计， 可能会多占用一些时间和空间资源， 这也是那些反对从高级语言中排除 GOTO 语句者的主要依据。实际上，硬件飞速发展，这点耗费，不再是重要的因素。

4.3.4 面向对象的分析设计

面向对象的分析模型主要由顶层架构图、 用例与用例图、 领域概念模型构成 ；

设计模型包含：

以包图表示的软件体系结构图、

以交互图表示的用例实现图、

完整精确的类图、

针对复杂对象的状态图、

描述流程化处理过程的活动图等。

4.4 软件的重用

重复使用相同或相似软件元素。

软件元素：需求分析文档、设计过程、设计文档、程序代码、测试用例、领域知识等，通产这些软件元素称为软部件。

不断地进行软部件的积累，并将它们组织成软部件库。

横向重用 (horizontal reuse) ：重用不同应用领域中的软件元素。

标准函数库是一种典型的、原始的横向重用机制。

纵向重用广受瞩目，并称为软件重用技术的真正希望所在，关键点是域分析，根据应用领域的特征以及相似性预测软部件的可重用性。

库的组织结构直接影响软部件的检索效率。

由于软部件大都经过严格的质量认证，并在实际运行环境中得到检验，因此重用软部件有助于改善软件质量。

4.5 逆向工程与重构工程

逆向工程就是分析已有的程序，寻找比源代码更高级的抽象表现形式。

相关概念：

重构 Restructuring ，在同一抽象级别上转换系统描述形式 ；

设计恢复 design recovery ，

重构工程 re-engineering ，也称修复和改造工程。

1、恢复信息的级别

逆向工程导出的信息， 4 个抽象层次

1.实现级

2.结构级

3.功能级

4.领域级

2、恢复信息的方法， 4 类：

1.用户指导下搜索与变换

2.变换式方法

3.基于领域知识的

4.铅板恢复法