

NEW PLAN

软考高级架构师

• — 段 新 征 程 •





大纲介绍

项目管理

面向对象开发

软件工程

软件工程概述

软件成熟度模型

软件过程模型

逆向工程

需求工程

系统设计

测试基础知识

系统运行与维护

净室软件工程

基于构件的软件工程

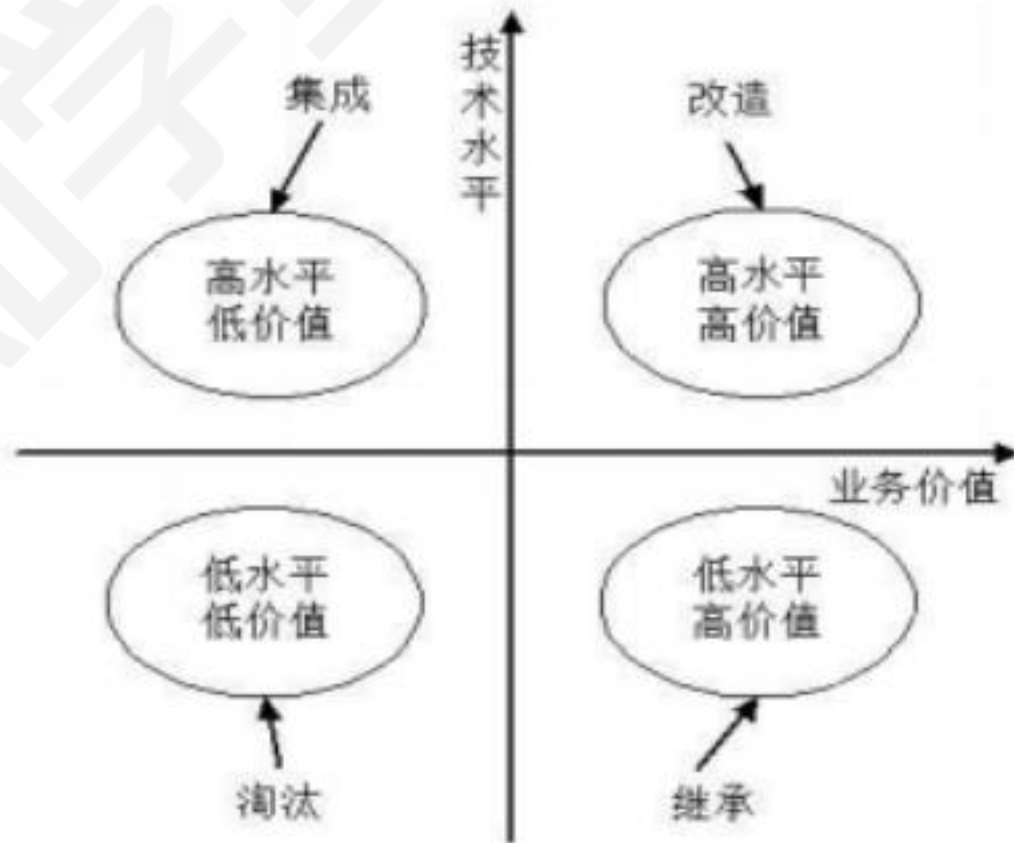


系统维护-遗留系统



遗留系统是指任何基本上不能进行修改和演化以满足新的变化了的业务需求的信息系统，它通常具有以下特点：

- 系统虽然完成企业中许多重要的业务管理工作，但**仍然不能完全满足要求**。一般实现业务处理电子化及部分企业管理功能，很少涉及经营决策。
- 系统在**性能上已经落后，采用的技术已经过时**。例如，多采用主机/终端形式或小型机系统，软件使用汇编语言或第三代程序设计语言的早期版本开发，使用文件系统而不是数据库。
- 通常是大型的软件系统，已经融入企业的业务运作和决策管理机制之中，**维护工作十分困难**。
- 没有使用现代信息系统建设方法进行管理和开发，现在基本上已经**没有文档，很难理解**。





系统维护-系统转换



系统转换是指新系统开发完毕，投入运行，取代现有系统的过程，需要考虑多方面的问题，以实现与老系统的交接，有以下三种转换计划：

- 直接转换：现有系统被新系统直接取代了，风险很大，适用于新系统不复杂，或者现有系统已经不能使用的情况。优点是节省成本，只适合小系统。
- 并行转换：新系统和老系统并行工作一段时间，新系统经过试运行后再取代，若新系统在试运行过程中有问题，也不影响现有系统的运行，风险极小，在试运行过程中还可以比较新老系统的性能，适用于大型系统。缺点是耗费人力和时间资源，难以控制两个系统间的数据转换。
- 分段转换：分期分批逐步转换，是直接和并行转换的集合，将大型系统分为多个子系统，依次试运行每个子系统，成熟一个子系统，就转换一个子系统。同样适用于大型项目，只是更耗时，而且现有系统和新系统间混合使用，需要协调好接口等问题。



数据转换与迁移：将数据从旧数据库迁移到新数据库中。有三种方法：系统切换前通过工具迁移、系统切换前采用手工录入、系统切换后通过新系统生成。



系统维护



系统维护是整个系统开发过程中耗时最长的，系统的**可维护性**可以定义为**维护人员理解、改正、改动和改进这个软件的难易程度**，其评价指标如下：

- **易分析性**。软件产品诊断软件中的缺陷或失效原因或识别待修改部分的能力。
- **易改变性**。软件产品使指定的修改可以被实现的能力，实现包括编码、设计和文档的更改。
- **稳定性**。软件产品避免由于软件修改而造成意外结果的能力。
- **易测试性**。软件产品使已修改软件能被确认的能力。



系统维护包括硬件维护、软件维护和数据维护，其中软件维护类型如下：

- **正确性维护**：发现了bug而进行的修改。
- **适应性维护**：由于外部环境发生了改变，被动进行的对软件的修改和升级。
- **完善性维护**：基于用户主动对软件提出更多的需求，修改软件，增加更多的功能，使其比之前的软件功能、性能更高，更加完善。
- **预防性维护**：对未来可能发生的问题进行预防性的修改。



考试真题



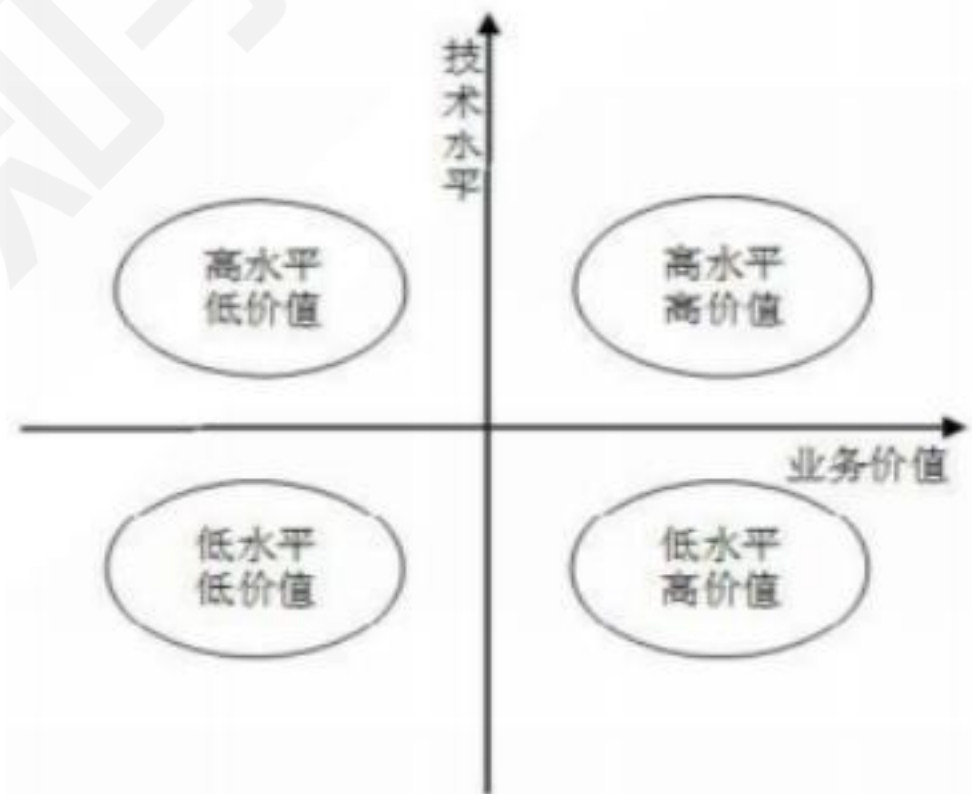
对于遗留系统的评价框架如下图所示，那么处于“高水平、低价值”区的遗留系统适合于采用的演化策略为()。

A.淘汰

B.继承

C.改造

D.集成





考试真题



以下关于软件维护和可维护性的叙述中，不正确的是()。

- A. 软件维护要解决软件产品交付用户之后运行中发生的各种问题
- B. 软件的维护期通常比开发期长得多，其投入也大得多
- C. 进行质量保证审查可以提高软件产品的可维护性
- D. 提高可维护性是在软件维护阶段考虑的问题



某企业由于外部市场环境和管理需求的变化对现有软件系统提出新的需求，则对该软件系统进行的维护属于_____维护。

- A. 正确性
- B. 完善性
- C. 适应性
- D. 预防性



净室软件工程



净室软件工程是一种应用数学与统计学理论以经济的方式生产高质量软件的工程技术，力图通过严格的工程化的软件过程达到开发中的零缺陷或接近零缺陷，强调的是预防大于检查。净室方法不是先制作一个产品，再去消除缺陷，而是要求在规约和设计中消除错误，然后以“净”的方式制作，可以降低软件开发中的风险，以合理的成本开发出高质量的软件。



在净室软件工程背后的哲学是：通过在**第1次**正确地书写代码增量，并在**测试前验证它们的正确性**，来**避免对成本很高的错误消除过程的依赖**。它的过程模型是在代码增量积聚到系统的过程的同时，进行代码增量的统计质量验证。它甚至**提倡开发者不需要进行单元测试，而是进行正确性验证和统计质量控制**。



净室软件工程(CSE)的理论基础主要是**函数理论和抽样理论**。



净室软件工程**应用技术手段**:

- 统计过程控制下的增量式开发。
- 基于函数的规范与设计。
- 正确性验证 (CSE的核心)
- 统计测试和软件认证。



净室软件工程在使用过程的一些**缺点**:

- CSE太理论化，需要更多的数学知识。其正确性验证的步骤比较困难且比较耗时。
- CSE开发小组不进行传统的模块测试，这是不现实的。
- CSE也会带有传统软件工程的一些弊端。



基于构件的软件工程

基于构件的软件工程(CBSE)是一种基于**分布对象技术**、强调通过**可复用构件设计与构造软件系统的软件复用途径**。CBSE体现了“**购买而不是重新构造**”的哲学，**将软件开发的重点从程序编写转移到了基于已有构件的组装**。用于CBSE的构件应该具备以下特征。

- **可组装型**：对于可组装的构件，所有外部交互必须通过公开定义的接口进行。同时它还必须对自身信息的外部访问。
- **可部署性**：软件必须是自包含的，必须能作为一个独立实体在提供其构件模型实现的构件平台上运行。构件总是二进制形式，无须在部署前编译。
- **文档化**：构件必须是完全文档化的，用户根据文档来判断构件是否满足需求。
- **独立性**：构件应该是独立的，应该可以在无其他特殊构件的情况下进行组装和部署，如确实需要其他构件提供服务，则应显示声明。
- **标准化**：构件标准化意味着在CBSE过程中使用的构件必须符合某种标准化的构件模型。构件模型定义了构件实现、文档化以及开发的标准，其包含的模型要素为：
 - 接口。构件通过构件接口来定义，构件模型规定应如何定义构件接口以及在接口定义中应该包含的要素，如操作名、参数以及异常等。
 - 使用信息。为使构件远程分布和访问，必须给构件一个特定的、全局唯一的名字或句柄。构件元数据是构件本身相关的数据，比如构件的接口和属性信息。
 - 部署。构件模型包括一个规格说明，指出应该如何打包构件使其部署成为一个独立的可执行实体。部署信息中包含有关包中内容的信息和它的二进制构成的信息。



基于构件的软件工程



构件模型提供了一组**被构件使用的通用服务**，这种服务包括以下两种。

- 平台服务，允许构件在分布式环境下通信和互操作。
- 支持服务，这是很多构件需要的共性服务。例如，构件都需要的身份认证服务、需要GPS服务之类。
- 中间件实现共性的构件服务，并提供这些服务的接口。



CBSE过程是支持基于构件组装的软件开发过程，过程中的6个主要活动：**系统需求概览、识别候选构件、根据发现的构件修改需求、体系结构设计、构件定制与适配、组装构件创建系统。**



CBSE过程与传统软件开发过程不同点：

- CBSE早期需要**完整的需求(即需求明确)**，以便尽可能多地识别出可复用的构件。
- 在过程早期阶段根据**可利用的构件来细化和修改需求**。如果可利用的构件不能满足用户需求，就应该考虑由复用构件支持的相关需求。
- 在系统体系结构设计完成后，会有一个**进一步的对构件搜索及设计精化的活动**。可能需要为某些构件寻找备用构件，或者修改构件以适合功能和架构的要求。
- 开发就是将**已经找到的构件集成在一起的组装过程**



基于构件的软件工程



构件组装是指构件相互直接集成或是用专门编写的‘**胶水代码**’将它们整合在一起创造出一个系统或另一个构件的过程。常见的组装构件有以下3种组装方式。

- **顺序组装**。通过按顺序调用已经存在的构件，可以用两个已经存在的构件来创造一个新的构件。如上一个构件输出作为下一个构件的输入。
- **层次组装**。这种情况发生在一个构件直接调用自另一个构件所提供的服务时。被调用的构件为调用的构件提所需的服务。二者之间接口匹配兼容。
- **叠加组装**。这种情况发生在两个或两个以上构件放在一起创建一个新构件的时候。这个新构件合并了原构件的功能，从而对外提供了新的接口。外部应用可以通过新接口来调用原有构件的接口，而原有构件不互相依赖，也不互相调用。这种组装类型适合于构件是程序单元或者构件是服务的情况。



构件组装的三种不兼容问题(通过编写适配器解决):

- **参数不兼容**。接口每一侧的操作有相同的名字，但参数类型或参数个数不相同。
- **操作不兼容**。提供接口和请求接口的操作名不同。
- **操作不完备**。一个构件的提供接口是另一个构件请求接口的一个子集，或者相反。

T H E E N D

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

