某企业经过多年的信息化建设，存在大量的应用软件系统，为了保证这些系统的运行与维护，专门组建应用系统维护部门。该部门的主要工作是保证系统的正常运行、处理问题以及扩展这些应该系统的功能，以满足企业业务功能的变化与扩展。

目前该部门存在人员流失、变更频繁，文档丢失或长期失于维护，维护成本愈来愈高等问题，具体表现为：

问题（1）：随着时间和人员的变动，程序被多人修改，往往导致程序难以理解，注释混乱，流程复杂；

问题（2）：随着不断修改程序和增加新的功能，模块之间的耦合关系日益复杂，维护成本不断增加。

这些问题导致新来的维护人员需要直接面对大量流程、结构复杂的源程序，维护困难，往往一次改动需要设计大量的软件模块。

为解决应用系统维护部门面对的问题，企业信息部门组织了专门的专家讨论会。各位专家一致认为，逆向工程与重构工程是目前预防性维护采用的主要技术，应该采用逆向工程的技术方法，重构相关应用系统文档，同时采用软件重构来降低软件代码的复杂性，最终降低维护成本。

【问题 1】（8 分）
软件的逆向工程是分析已有程序，寻求比源代码更高级的抽象表现形式。与之相关的概念包括软件重构、设计恢复、重构工程等。请说明设计恢复中常见的恢复信息的 4 种级别。
【问题 2】（11 分）
重构是对软件内部结构的一种调整，目的是不改变软件功能的前提下，提高其可理解性，降低其修改成本。请说明软件重构的三个类别，并简要说明常见的重构方法。针对题干中的问题（1）和问题（2），宜采用何种重构方法？
【问题 3】（6 分）
软件重构做出的修改可能导致程序运行变慢，但也更容易进行软件的性能优化和调整，请分析原因。

参考答案：

【问题 1】

（1）实现级：过程的设计模型。

（2）结构级：程序和数据结构信息。

（3）功能级：对象模型、数据和控制流模型。

（4）领域级：UML 状态图和部署图。

【问题 2】

软件重构的三个类别：

代码重构、设计重构、架构重构。

常见的重构方法：

（1）提取方法（Extract method）

（2）用委托来代替继承（Replace Inheritance witch Delegation）

（3）用子类代替型别码（Replace Type with Subclasses）

（4）用多态来代替条件判断（Replace conditional with polymorphism）

（5）模板函数

（6）提取类

（7）提取接口

问题（1）可采用提取方法的重构方法解决，问题（2）可采用提取接口的方法解决。

【问题 3】

为了使软件更容易理解，同时又需要考虑到各种兼容性，在重构时，可能需要在代码中增加冗余的判断、冗余的代码或结构；也可能需要修改已有的数据库结构和索引等，导致程序运行变慢。

但从长远来看，由于重构以后的软件结构更加清晰，代码复杂性更低，更易于理解，在性能调优时更容易分析瓶颈之所在，然后加以解决，因此，软件重构也更易容进行软件的性能优化和调优。

试题分析：
软件的逆向工程师分析程序，力图在比源代码更高抽象层次上建立程序表示的过程。逆向工程师一个恢复设计的过程，从现有的程序中抽取数据、体系结构和过程的设计信息。
软件重构的目的主要有四个方面，可以分为三类。
模块化是指解决一个复杂问题时自顶向下逐层把软件系统划分为若干模块的过程。

阅读以下关于软件架构风格的说明，在答题纸上回答问题 1 和问题 2。

【说明】

某软件公司为其新推出的字处理软件设计了一种脚本语言，专门用于开发该字处理软件的附加功能插件。为了提高该语言的编程效率，公司组织软件工具开发部门为脚本语言研制一套集成开发环境。软件工具开发部门根据字处理软件的特点，对集成开发环境进行了需求分析，总结出以下 3 项核心需求：

（1）集成开发环境需要提供对脚本语言的编辑、语法检查、解释、执行和调试等功能的支持，并要实现各种功能的灵活组合、配置与替换。

（2）集成开发环境需要提供一组可视化的编程界面，用户通过对界面元素拖拽和代码填充的方式就可以完成功能插件核心业务流程的编写与组织。

（3）在代码调试功能方面，集成开发环境需要实现在脚本语言编辑界面中的代码自动定位功能。具体来说，在调试过程中，编辑界面需要响应调试断点命中事件，并自动跳转到当前断点处所对应的代码。

针对上述需求，软件工具开发部门对集成开发环境的架构进行分析与设计，王工认为该集成开发环境应该采用管道-过滤器的架构风格实现，李工则认为该集成开发环境应该采用以数据存储为中心的架构风格来实现。公司组织专家对王工和李工的方案进行了评审，最终采用了李工的方案。

【问题 1】（12 分）

请用 200 字以内的文字解释什么是软件架构风格，并从集成开发环境与用户的交互方式、集成开发环境的扩展性、集成开发环境的数据管理三个方面说明为什么最终采用了李工的设计方案。

【问题 2】（13 分）

在对软件系统架构进行设计时，要对架构需求进行分析，针对特定需求选择最为合适的架构风格，因此实际的软件系统通常会混合多种软件架构风格。请对核心需求进行分析，说明为了满足需求（2）和（3），分别应采用何种架构风格，并概要说明采用相应架构风格后的架构设计过程。

【问题 1】

软件架构风格是指描述特定软件系统组织方式的惯用模式。组织方式描述了系统的组成构件和这些构件的组织方式，惯用模式则反映众多系统共有的结构和语义。

从集成开发环境与用户的交互方式看，用户通常采用交互式的方式对脚本语言进行编辑、解释执行与调试。在这种情况下，采用以数据存储为中心的架构风格能够很好地支持交互式数据处理，而管道-过滤器架构风格则对用户的交互式数据处理支持有限。

从集成开发环境的扩展性来看，系统核心需求要求实现各种编辑、语法检查、解释执行等多种功能的灵活组织、配置与替换。在这种情况下，采用以数据存储为中心的架构风格，以数据格式解耦各种功能之间的依赖关系，并可以灵活定义功能之间的逻辑顺序。管道-过滤器架构风格同样以数据格式解耦数据处理过程之间的依赖关系，但其在数据处理逻辑关系的灵活定义方面较差。

从集成开发环境的数据管理来看，集成开发环境需要支持脚本语言、语法树（用

于检查语法错误）、可视化模型、调试信息等多种数据类型，并需要支持数据格式的转换。以数据存储为中心的架构将数据存储在统一的中心存储器中，中心存储器能够表示多种数据格式，并能够为数据格式转换提供各种支持。管道-过滤器架构风格通常只能支持有限度的数据格式，并且在数据格式转换方面的灵活性较差。

【问题 2】

为了满足需求（2），应该采用解释器架构风格。具体来说，需要：① 为可视化编程元素及其拖拽关系定义某种语言，并描述其语法与语义；② 编写解释器对该语言进行解释；③ 生成对应的脚本语言程序。

为了满足需求（3），应该采用隐式调用架构风格。具体来说，首先需要定义“断点在调试过程中命中”这一事件，并实现当断点命中后的屏幕定位函数。集成开发环境维护一个事件注册表结构，将该事件与屏幕定位函数关联起来形成注册表中的一个记录项。在调试过程中，集成开发环境负责监听各种事件，当“断点在调试过程中命中”这一事件发生时，集成开发环境查找事件注册表，找到并调用屏幕定位函数，从而实现脚本语言编辑界面与调试代码的自动定位。

试题分析：

本题主要考查考生对软件架构风格的理解与掌握，特别是针对实际应用问题，如何采用基于软件架构风格的系统软件架构设计。软件架构风格是指描述特定软件系统组织方式的惯用模式。组织方式描述了系统的组成构件和这些构件的组织方式，惯用模式则反映众多系统共有的结构和语义。

【问题 1】

本问题主要考查设计方案的比较与选型。题干明确指出从集成开发环境与用户的交互方式、集成开发环境的扩展性、集成开发环境的数据管理三个方面对两种方案进行比较，并说明选用李工方案的原因。

从集成开发环境与用户的交互方式看，根据题干描述，用户通常采用交互式的方式对脚本语言进行编辑、解释执行与调试。在这种情况下，采用以数据存储为中心的架构风格能够很好地支持交互式数据处理，而管道-过滤器架构风格则对用户的交互式数据处理支持有限。

从集成开发环境的扩展性来看，根据题干描述，要求实现各种编辑、语法检查、解释执行等多种功能的灵活组织、配置与替换。在这种情况下，采用以数据存储为中心的架构风格，以数据格式解耦各种功能之间的依赖关系，并可以灵活定义功能之间的逻辑顺序。管道-过滤器架构风格同样以数据格式解耦数据处理过程之间的依赖关系，但其在数据处理逻辑关系的灵活定义方面较差。

从集成开发环境的数据管理来看，集成开发环境需要支持脚本语言、语法树（用于检查语法错误）、可视化模型、调试信息等多种数据类型，并需要支持数据格式的转换。以数据存储为中心的架构将数据存储在统一的中心存储器中，中心存储器能够表示多种数据格式，并能够为数据格式转换提供各种支持。管道-过滤器架构风格通常只能支持有限度的数据格式，并且在数据格式转换方面的灵活性较差。

考生在回答上述问题时，不能仅仅列举教科书中对数据存储为中心的架构风格和管道-过滤器架构风格的特点描述，必须紧紧围绕题干对系统要求的描述，将系统要求与架构风格特点结合起来进行回答。

【问题 2】

本问题主要考查考生如何根据应用要求选择合适的架构风格。

需求（2）要求“集成开发环境需要提供一组可视化的编程界面，用户通过对界面元素拖拽和代码填充的方式就可以完成功能插件核心业务流程的编写与组织”，这是一个对可视化开发过程的典型描述，而可视化开发的核心是如何定义并解释可视化编程语言，其核心应该是解释器架构风格。因此针对需求（2），应该采用解释器架构风格。具体来说，需要：① 为可视化编程元素及其拖拽关系定义某种语言，并描述其语法与语义；② 编写解释器对该语言进行解释；③ 生成对应的脚本语言程序。

需求（3）要求“在代码调试功能方面，集成开发环境需要实现在脚本语言编辑界面中的代码自动定位功能。具体来说，在调试过程中，编辑界面需要响应调试断点命中事件，并自动跳转到当前断点处所对应的代码。”从描述中可以看出，这是一个具有“事件触发”能力的功能描述，即由“断点命中”事件触发事先定义的“代码自动定位”功能。可以看出，这样的需求采用隐式调用架构风格最为恰当。具体来说，首先需要定义“断点在调试过程中命中”这一事件，并实现当断点命中后的屏幕定位函数。集成开发环境维护一个事件注册表结构，将该事件与屏幕定位函数关联起来形成注册表中的一个记录项。在调试过程中，集成开发环境负责监听各种事件，当“断点在调试过程中命中”这一事件发生时，集成开发环境查找事件注册表，找到并调用屏幕定位函数，从而实现脚本语言编辑界面与调试代码的自动定位。

试题 2:

阅读以下关于某项目开发计划的说明，在答题纸上回答问题 1 至问题 4。

【说明】

某软件公司拟开发一套电子商务系统，王工作为项目组负责人负责编制项目计划。由于该企业业务发展需要，CEO 急于启动电子商务系统，要求王工尽快准备一份拟开发系统的时间和成本估算报告。

项目组经过讨论后，确定出与项目相关的任务如表 2-1 所示。其中，根据项目组开发经验，分别给出了正常工作及加班赶工两种情况下所需的时间和费用。

表 2-1 项目开发任务进度及费用

任务名称	正常工作	加班工作	前置任务
A.系统调研	4 天/7200 元	3 天/8400 元	-
B.提交项目计划	2 天/1600 元	1 天/1900 元	A
C.需求分析	6 天/9600 元	4 天/14200 元	B
D.系统设计	12 天/22200 元	8 天/27600 元	C
E.数据库开发	3 天/5100 元	2 天/5700 元	D
F.网页开发	6 天/8700 元	5 天/10000 元	D
G.报表开发	4 天/6000 元	任务外包无法赶工	D
H.测试修改	7 天/9800 元	4 天/12800 元	E,F,G
I.安装部署	4 天/4000 元	2 天/5000 元	H

【问题 1】(7 分)

请用 400 字以内文字说明王工拟编制的项目计划中应包括哪些内容。

【问题 2】(8 分)

请根据表 2-1，分别给出正常工作和最短工期两种情况下完成此项目所需的时间和费用。

【问题 3】(4 分)

如果项目在系统调研阶段用了 7 天时间才完成，公司要求尽量控制成本，王工可在后续任务中采取什么措施来保证项目能按照正常工作进度完成？

【问题 4】(6 分)

如果企业 CEO 想在 34 天后系统上线，王工应该采取什么措施来满足这一要求？这种情况下完成项目所需的费用是多少？

参考答案：

【问题 1】

(1) 项目背景

(2) 项目经理、项目经理的主管领导、客户方联系人、客户方的主管领导，项目领导小组(项目管理团队)和项目实施小组人员

(3) 项目的总体技术解决方案

(4) 所选择的项目管理过程及执行水平

(5) 对这些过程的工具、技术和输入输出的描述

(6) 选择的项目的生命周期和相关的项目阶段

(7) 项目最终目标和阶段性目标

(8) 进度计划

(9) 项目预算

(10) 变更流程和变更控制委员会

(11) 对于内容、范围和时间的关键管理评审，以便于确定悬留问题和未决决策

【问题 2】

正常工作成本=74200 元。

正常工作工期=41 天。

最短工期成本=91600 元。

最短工期=27 天。

【问题 3】

在“B 提交项目计划”和“I 安装部署”任务中采用加班工作措施，以使得能够按照正常工作进度完成。

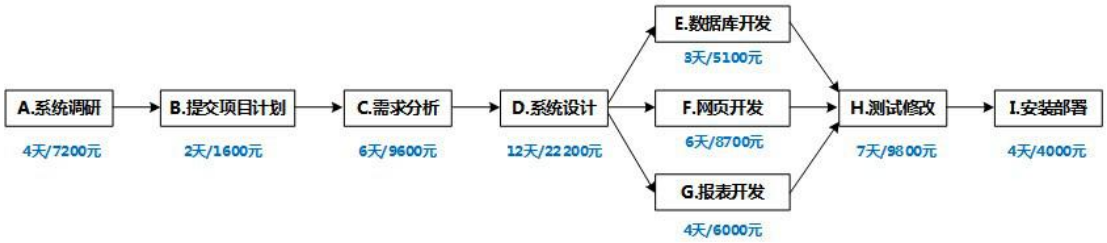
【问题 4】

标准时长 41 天的任务，要 34 天完成，应赶工 7 天。具体赶工的任务包括：将 A、B、H、I 四个任务加班完成，这样正好弥补之前延误的 7 天工期，最终以 79700 元完成项目。

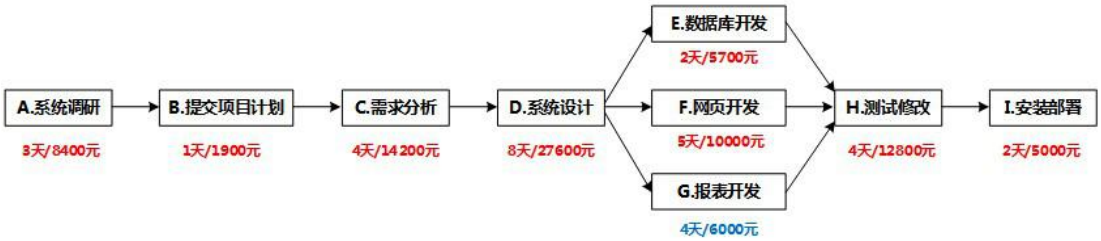
试题分析：

正常工作成本=7200 元+1600 元+9600 元+22200 元+5100 元+8700 元+6000 元+9800 元+4000 元=74200 元。

正常工作工期=4+2+6+12+6+7+4=41 天。



最短工期成本=8400 元+1900 元+14200 元+27600 元+5700 元+10000 元+6000 元+12800 元+5000 元=91600 元。
最短工期=3+1+4+8+5+4+2=27 天。



要缩短项目的工期，主要有两种方法：
赶工：对成本和进度进行权衡，确定如何尽量少增加费用的前提下最大限度地缩短项目所需要的时间，称为赶进度也称赶工。
快速跟进：调整逻辑关系，通过对各种逻辑关系并行确定来缩短项目周期。在进行项目设计中，当风险不大时，通过精心安排而使项目的前后阶段相互搭接以加快项目进展速度的做法叫快速跟进。
其中快速跟进由于只是将部分工作提前开始，所以不会明显增加成本，在当前的环境中，是比较合适的方法。

任务名称	正常工作	加班工作	可压缩天数	压缩 1 天增加费用
A.系统调研	4 天/7200 元	3 天/8400 元	1	1200 元
B.提交项目计划	2 天/1600 元	1 天/1900 元	1	300 元
C.需求分析	6 天/9600 元	4 天/14200 元	2	2300 元
D.系统设计	12 天/22200 元	8 天/27600 元	4	1350 元
E.数据库开发	3 天/5100 元	2 天/5700 元	1	600 元
F.网页开发	6 天/8700 元	5 天/10000 元	1	1300 元
G.报表开发	4 天/6000 元	任务外包无法赶工	0	
H.测试修改	7 天/9800 元	4 天/12800 元	3	1000 元
I.安装部署	4 天/4000 元	2 天/5000 元	2	500 元

阅读以下关于软件架构的叙述，回答问题 1、问题 2 和问题 3。
软件架构是指大型、复杂软件的系统结构的设计、规格说明和实施。它以规范的形式装配若干结构元素，从而描述出系统的主要功能和性能要求，同时表述其它非功能性需求（如可靠性、可扩展性、可移植性和可用性等）。软件架构为软件系统提供了一个结构、行为和属性的高级抽象模式，可以使用一个公式来表达；

软件架构={构成系统的元素，指导元素集成的形式，关系和约束}。
“4+1”视图模型用五个视图组成的模型来描述软件架构。该模型包含五个主要的视图：
逻辑视图（Logical View），描述了设计的对象模型，支持系统的功能需求。
进程视图（Process View），描述了设计的并发和同步特征，支持系统的运行特性。
物理视图(Physical view)，描述了软件到硬件的映射，反映了分布式特性，支持系统的拓扑、安装和通信需求。
开发视图（Development view），描述了在开发环境中软件的静态组织结构，支持软件开发的内部需求。
场景(Scenario)，用来说明重要的系统活动，是其它四个视图在用例(Use Case)驱动下的综合。

[问题 1]（7 分）

软件架构在软件需求与设计之间架起一座桥梁，也是风险承担者进行交流的手段，允许不同的风险承担者找出他们所关心的软件架构问题。假设采用面向对象的设计方法，各个视图涉及的组件（元素）包括：任务、类、模块、节点、步骤等，风险承担者包括最终用户、系统设计师、程序员、经理、项目管理师等。请在下表中的（1）到（7）处填入恰当的内容（空白处不用填）。

	逻辑视图	进程视图	物理视图	开发视图	场景
组件	(1)	(2)	(3)	(4)	(5)
风险承担者	(6)		(7)		

[问题 2]（10 分）

对于大型项目，通常采用迭代的方法来进行架构设计。架构先被原型化、测试、评估分析，然后在一系列的迭代过程中被细化。这种方法能够使需求细化、成熟化，并能够被更好地理解。请用 400 字以内文字，简述软件架构基于场景驱动的迭代式设计过程。

[问题 3]（8 分）

开发视图是实现软件详细设计和编码的重要蓝图。请用 300 字以内文字，说明开发视图需要满足软件内部的哪些需求以及开发视图直接影响到项目管理的哪些方面。

【问题 1】

本题相当于选择题，但要获得好的成绩，仍需要仔细构思。

1）逻辑试图表述系统的功能需求。系统分解为一序列的关键抽象，这些抽象（大多数）来自于需求分析中所提出功能要求，对对象或类的形式来表示（采用抽象、封装和继承）。分解并步仅仅是为了功能分析，而且用来识别遍布系统各个部分的通用机制和设计元素。系统的功能需求来自最终用户，最终用户是逻辑视图对应的风险承担者。

2）进程视图表述系统的运行特性。利用进程视图可解决系统的并发性、分布性、系统完整性、容错性等问题。另外。它还可以表达逻辑视图的主要抽象在那个控制线程上被实际执行。风险承担者主要是系统集成人员，组件元素是任务。

3）物理视图表述系统的拓扑、安装和通信需求。用来表达软件系统中的各种元素（元素可以理解为组件或过程）被映射或部署至不同的网络计算机节点上。风险承担者主要是系统实施工程师。

4）开发视图表述软件开发的内部需求。开发视图关注软件开发环境下实际模块的组织（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织成分层结构，每个层为上一层提供良好定义的接口。风险承担者主要师编程人员和软件项目管理人员。

5）场景用来说明重要的系统活动，是其他四个视图在用例（Ues Case）驱动下的综合。在某种意义上场景是最重要的需求抽象。该视图是其他视图的冗余（因此“＋1”），但它起到了两个作用：首先场景可用来发现构架设计过程中的架构元素，其次是场景可作为架构设计结束后的功能验证。它可作为架构原型测试的出发点。风险承担者是最终用户和开发人员，组件元素是步骤。

	逻辑视图	进程视图	物理视图	开发视图	场景
组件（元素）	（1） 对象或类	（2） 任务	（3） 组件或过程	（4） 程序库或子系统	（5） 步骤
风险承担者	（6） 最终用户	系统集成人员	（7） 系统实施工程师	编程人员和软件项目管理人员	最终用户和开发人员

【问题 2】

系统大多数关键的功能以场景（或用例）的形式被捕获。所谓关键是指系统最重要的功能（或系统存在的理由），或使用频率最高的功能，或其应用减轻了一些重要的技术风险。基于场景驱动的迭代式设计过程如下。

1）开始阶段。基于风险和重要性为某次迭代选择一些场景。场景可以被归纳为对若干用户需求的抽象；场景进行“描述”，以识别主要的抽象（类、机制、过程、子系统）；将所有发现的构架元素分布到 4 个视图中；然后实施、测试、评估该架构，这个过程中可能检测到一些缺点或潜在的增强要求；捕获经验教训。

2）循环阶段。重信评估风险，选择能减轻风险或提高结构覆盖的额外的少量场景，然后试着在原先的架构中描述这些场景，发现额外的架构元素，或找出适应这些场景所需要的架构变更，更新 4 个主要视图;根据变更修订现有的场景；升级实现工具（架构原型）以支持新的、以支持新的、扩展了的场景集合。

3）测试。如果有可能（比如，在已有可重用的组件下快速实现系统），在实际的目标环境和负载下进行测试。

4）评审这 5 个视图，检测架构在简洁性、可重用性和通用性方面可能存在的潜在的问题。

5）更新设计准则和基本原理。

6）捕获经验教训。

对于实际的系统，初始的架构原型需要不断进行演化。一般的情况是在经过两次或第三次迭代，当找到了主要的抽象，子系统和过程都已经完成并且已经实现所有的接口，系统架构可认为趋向于稳定。

【问题 3】

软件内部需求是指任何一个软件都要满足的一些非功能方面的要求，大部分情况下，开发视图架构考虑的内部需求与以下几项因素有关：开发难度、软件管理、重用性和通用性及由工具集、编程语言所带来的限制与约束。开发视图是项目管理的基础，通过开发视图对系统功能、模块的层次性分解，能够预估开发工作量、安排开发任务，编制开发计划，进而控制进度，即开发视图是需求分解、团队工作的分配（或任务管理）、成本评估和计划（或成本管理）、项目进度的监控（或进度管理）等方面的基础。

本题主要考查软件架构“4+1”视图的有关知识和实施方法，熟悉以下关于软件架构的知识是回答本题的前提。

首先要准确把握软件架构的定义。架构（Architecture）愿意为建筑学设计和建筑物建造的艺术与科学。软件构架(Software Architecture)，或称为软件构架）是软件系统的高层描述，它给出了关于软件系统组织结构的一系列高级的、重要的抽象，包括：系统组成的机构性构件了；组成构件之间的接口；构件相对系统其他部分的可视行为；构件之间所采取的交互和协作的关系。软件构架在 RUP 中的定义是指系统核心构件的组织或结构，这些核心构件通过接口与不断减小的构件与接口所组成的构件进行交互。

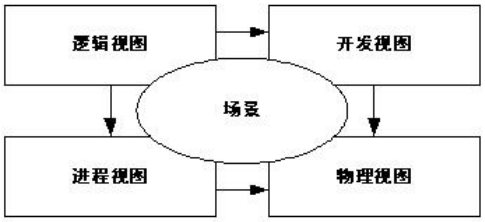
人们在软件开发过程中积累了丰富的构架知识，形成了特定的构架风格，这些架构风格为高层次的软件复用技术建立了坚实的基础：例如，C/S 架构、管道/过滤器架构、分层架构、解释器架构、黑板架构等等，而各种分布式组件技术如 DCOM,EJB，Web-Services 也都和软件架构密切相关。

长期以来，人们一直在努力软件架构更加精确的形式化描述，力图用一种类似于某种编程语言的形式来描述软件架构，如 Rapide,Wright,Aesop,UniCon,ACME 等。XML 描述于软件建模 UML 技术的发展为软件架构描述语言注入了新的发展思路，新一代的架构描述语言（如 xArch,xADL 等）充分应用了这些新的描述手段的特点。同时，伴随着架构描述技术的进步，架构评估等研究在不断的深入。

其次，要正确理解软件架构的重要作用。

- 软件架构能够知道整个系统的设计和演进，它是软件需求分析的结果，同时是下一步进行软件设计的规格和蓝图。对于复杂软件系统而言，在架构阶段，系统的结构和规格说明非常重要，而在软件设计阶段，算法和数据结构更重要。
- 软件架构对系统的描述，借鉴了建筑工程设计的思想，通过各种视图从不同角度以规范、一致、易理解的“语言”来表达系统的各种规格和行为。以某一特定角度看到的系统架构之规格、行为，主要是结构、核心构件和主要控制流等。
- 软件架构是风险承担者进行交流的手段。所谓奉献承担者是指对软件系统某个方面（或层次）负责或（关注）的人员。也可以这样来理解风险承担者：软件系统的的某个方面（或层次）如果存在缺陷或问题，对此负责任或受影响的人员。风险承担者包括最终用户、系统设计师、程序员、经理、项目管理等。
- 软件架构是可传递、可重用的模型。
- 软件架构是软件工程早期设计决策的体现，而且在整个开发周期中不断演进，软件架构对于软件质量（功能属性、非功能属性）都有重要影响。

“4+1”视图模型是最重要软件架构模式，由 Philippe Kruchten 在 1995 年提出。如下图所示。



需要指出的是，并不是所有的软件架构都需要“4+1”视图。无用的视图可以从架构描述中省略，例如，单击软件，可以省略物理视图;而如果仅有一个进程或程序，则可以省略过程视图。对于非常小型的系统，甚至可能逻辑视图与开发视图非常相似，而不需要分开的描述。

第一步：总结出问题的要点。

【问题 1】

考查采用面向对象的架构设计方法，“4+1”视图各个视图涉及的组件要素与风险承担者。

【问题 2】

考查“4+1”视图架构的场景驱动实现方法。

【问题 3】

开查发现视图的标书内容及其与实施项目管理的关系。

第二步：根据问题要点，仔细阅读正文，找出相应段落。

（1）在题目的说明中给出了软件构架是“指大型、复杂软件的系统结构的设计、规格说明和实施”和“4+1”视图模型的基本概念。

（2）题目的说明中进一步指出，软件架构“描述出系统的主要功能和性能需求，同时表述其他非功能性要求（如可靠性、可扩展性、可移植性和可用性等）”，认真体会，结合项目管理方面的基础知识，不难回答问题 3

（3）在问题 1 的说明中限定了视图涉及的组件包括：任务、类、模块、节点、步骤等，风险承担者包括最终用户、系统设计师、程序员、经理、项目管理师等，问题的答案在这个范围内选择。正确地理解视图组建和给定地风险承担者角色概念，很容易回答问题 1。

第三步：分析试题地内容，构思答案地要点。

【问题 1】

本题相当于选择题，但要获得好的成绩，仍需要仔细构思。

1）逻辑试图表述系统的功能需求。系统分解为一序列的关键抽象，这些抽象（大多数）来自于需求分析中所提出功能要求，对对象或类的形式来表示（采用抽象、封装和继承）。分解并步仅仅是为了功能分析，而且用来识别遍布系统各个部分的通用机制和设计元素。系统的功能需求来自最终用户，最终用户是逻辑视图对应的风险承担者。

2）进程视图表述系统的运行特性。利用进程视图可解决系统的并发性、分布性、系统完整性、容错性等问题。另外。它还可以表达逻辑视图的主要抽象在那个控制线程上被实际执行。风险承担者主要是系统集成人员，组件元素是任务。

3）物理视图表述系统的拓扑、安装和通信需求。用来表达软件系统中的各种元素（元素可以理解为组件或过程）被映射或部署至不同的网络计算机节点上。风险承担者主要是系统实施工程师。

4）开发视图表述软件开发的内部需求。开发视图关注软件开发环境下实际模块的组织（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织成分层结构，每个层为上一层提供良好定义的接口。风险承担者主要师编程人员和软件项目管理人员。

5）场景用来说明重要的系统活动，是其他四个视图在用例（Ues Case）驱动下的综合。在某种意义上场景是最重要的需求抽象。该视图是其他视图的冗余（因此“＋1”），但它起到了两个作用：首先场景可用来发现构架设计过程中的架构元素，其次是场景可作为架构设计结束后的功能验证。它可作为架构原型测试的出发点。风险承担者是最终用户和开发人员，组件元素是步骤。

	逻辑视图	进程视图	物理视图	开发视图	场景
组件（元素）	（1） 对象或类	（2） 任务	（3） 组件或过程	（4） 程序库或子系统	（5） 步骤
风险承担者	（6） 最终用户	系统集成人员	（7） 系统实施工程师	编程人员和软件项目管理人员	最终用户和开发人员

【问题 2】

系统大多数关键的功能以场景（或用例）的形式被捕获。所谓关键是指系统最重要的功能（或系统存在的理由），或使用频率最高的功能，或其应用减轻了一些重要的技术风险。基于场景驱动的迭代式设计过程如下。

1）开始阶段。基于风险和重要性为某次迭代选择一些场景。场景可以被归纳为对若干用户需求的抽象；场景进行“描述”，以识别主要的抽象（类、机制、过程、子系统）；将所有发现的构架元素分布到 4 个视图中；然后实施、测试、评估该架构，这个过程中可能检测到一些缺点或潜在的增强要求；捕获经验教训。

2）循环阶段。重信评估风险，选择能减轻风险或提高结构覆盖的额外的少量场景，然后试着在原先的架构中描述这些场景，发现额外的架构元素，或找出适应这些场景所需要的架构变更，更新 4 个主要视图;根据变更修订现有的场景；升级实现工具（架构原型）以支持新的、以支持新的、扩展了的场景集合。

3）测试。如果有可能（比如，在已有可重用的组件下快速实现系统），在实际的目标环境和负载下进行测试。

4）评审这 5 个视图，检测架构在简洁性、可重用性和通用性方面可能存在的潜在的问题。

5）更新设计准则和基本原理。

6）捕获经验教训。

对于实际的系统，初始的架构原型需要不断进行演化。一般的情况是在经过两次或第三次迭代，当找到了主要的抽象，子系统和过程都已经完成并且已经实现所有的接口，系统架构可认为趋向于稳定。

【问题 3】

软件内部需求是指任何一个软件都要满足的一些非功能方面的要求，大部分情况下，开发视图架构考虑的内部需求与以下几项因素有关：开发难度、软件管理、重用性和通用性及由工具集、编程语言所带来的限制与约束。开发视图是项目管理的基础，通过开发视图对系统功能、模块的层次性分解，能够预估开发工作量、安排开发任务，编制开发计划，进而控制进度，即开发视图是需求分解、团队工作的分配（或任务管理）、成本评估和计划（或成本管理）、项目进度的监控（或进度管理）等方面的基础。

试题 5

某企业委托软件公司开发一套运动器材综合销售平台，以改进已有的销售管理系统，拓展现有的实体店销售模式，综合管理线上线下的器材销售业务。该软件公司组建项目组开发该系统，现正处于需求获取阶段。经过项目组讨论，由于目标系统业务功能比较复杂，所以在需求获取中针对不同类型的业务需求，采用不同的需求获取方法。项目组列出可选的需求获取方法包括：用户访谈、联合需求计划（JRP）、问卷调查、文档分析和实地观察等。

需求获取的要求如下：

（1）获取已有销售管理系统中所实现的实体店销售模式和过程；

（2）获取系统的改进需求和期望增加的业务功能；

（3）获取当前业务过程中的详细数据并深入了解这些数据产生的原因；

（4）从企业管理人员、销售人员、各种文档资源等尽可能多的来源获取需求；

（5）消除需求中出现的冲突，尽可能获取全面、一致的需求；

（6）尽可能多地让用户参与需求获取过程。

【问题 1】（10 分）
联合需求计划（JRP）是一种流行的需求获取方法。请说明什么是 JRP，JRP 与其它需求获取方法相比有什么优势？
【问题 2】（12 分）
针对题目中所描述的需求获取要求（1）~（6），选择最适合的需求获取方法填入表 1-1 中的（a）~（f）处。

表 1-1 需求获取方法选择

需求获取要求	需求获取方法
（1）	（a）
（2）	（b）
（3）	（c）
（4）	（d）
（5）	（e）
（6）	（f）

【问题 3】（3 分）
由于该企业销售规模较大，所积累的企业业务文档数量庞大，所以只能通过抽样实现不同类型的文档分析。如果对于每种类型的文档要求 90%的可信度（可信度因子为 1.645），那么不同类型的文档分别需要抽样多少份就能达到该要求？

参考答案：

【问题 1】

联合需求计划是一个通过高度组织的群体会议来分析企业内的问题并获取需求的过程，它是联合应用开发的一部分。JRP 是一种相对来说成本较高的需求获取方法，但也是十分有效的一种。它通过联合各个关键用户代表、系统分析师、开发团队代表一起，通过有组织的会议来讨论需求。JRP 将会起到群策群力的效果，对于一些问题最有歧义的时候、对需求最不清晰的领域都是十分有用的一种方法。优势：1、发挥用户和管理人员参与系统开发过程的积极性，提高系统开发效率；2、降低系统需求获取的时间成本，加速系统开发周期；3、采用原型确认系统需求并获取设计审批，具有原型化开发方法的优点。

【问题 2】

（a）实地考察或文档分析

（b）用户访谈或联合需求计划

（c）用户访谈或联合需求计划

（d）问卷调查或文档分析

（e）联合需求计划

（f）联合需求计划

【问题 3】

样本大小 = $0.25 \times (1.645/(1-0.90))^2$ = 67.65063

因此，需要抽取 68 份文档。

试题分析：

【问题 3】

样本数量=0.25*（可信度因子/错误率）^{2}