**软件工程：**

IBM公司的OS/360，共约100万条指令，花费了5000个人力资源;经费达数亿美元，而结果却令人沮丧，错误多达2000个以上，系统根本无法正常运行。OS/360系统的负责人Brooks这样描述开发过程的困难和混乱:“…像巨兽在泥潭中作垂死挣扎，挣扎得越猛，泥浆就沾得越多，最后没有一个野兽能够逃脱淹没在泥潭中的命运。…”

试回答以下问题:

1)导致上述资料问题的原因是什么?

2)如何克服?

1）软件危机或需求问题、管理问题、方法问题、工具问题、软件本身的特点

2）克服途径:采用工程化的方法来开发和管理软件。包括:

[1].对计算机软件有一个正确的认识(软件≠程序)

[2].必须充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、

管理严密、各类人员协同配合、共同完成的工程项目过程。

[3].开发和使用更好的软件工具。

[4].推广使用在实践中总结出来的开发软件的成功技术和方法

为什么说复杂性、不可见性、易变性是软件固有的三大特点?并请利用所学的软件工程

知识分析在软件开发的过程中如何避免这三个特点带来的负面影响?

**复杂性:**软件所解决的问题开发不仅涉及到很多领域或行业的专门知识，而且还与社会、人的组织和管理因素相关。导致软件开发过程中的很多不确定的或不可预见性。利用抽象、信息隐藏、模块等概念降低复杂性、提高可控性

**不可见性:**软件是逻辑产品,集体智慧的结晶。采用生命周期的方法分阶段开发和评审。

**易变性:**软件是用来解决现实问题的，现实世界在不断地变化，所以软件也需要随之改变。在开发过程中提高软件的可维护性。

有人认为，程序是给机器执行的，所以只要正确，能被计算机所理解就可以了。你同意他的观点吗﹖请说明理由。

答:不同意。

因为在软件的生命周期中需要经常阅读程序。编写程序的人、参与测试的人以及参与维护的人都要阅读程序。所以，在编写程序时，应当意识到今后会有人反复地阅读这个程序并理解程序，所以应当在编程时讲究程序的风格。力图在保证正确性的基础上尽量提高程序的可读性和可理解性。

试述软件工程的**目标**是什么﹖并谈谈在软件开发过程中应该如何做才能达到这个目标?

答:在给定成本和进度的前提下，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性并满足用户需求的软件产品。

从重视分析和设计;好的编码习惯;加强软件测试和复审等方面回答。

在进行硬件产品的生产时，增派人手是加快进度的最直接和有效的手段。但有人说:“在

已拖延的软件项目上追加更多人手，往往使项目更难如期完成”，这种说法对吗，为什么?

答:这种说法是对的。

对于一项复杂的软件开发任务，通常难以通过增加人力来缩短开发时间。这样做会带来包括培训、沟通交流、任务的重新分配、系统测试等额外工作量。

由于软件的特殊性，它本质上是一项系统工作。有些任务由于次序等的限制不能做进一步分解，对于这种任务，人力的增加是没有帮助的。即便对于可继续分解的任务，其各子任务之间往往也有着这样那样的联系，开发时需要协调、沟通和交流,这无疑是需要耗时耗力的。增加了人力，任务必须被重新分解分配，可能又会损失一部分原有已完成的工作，也导致系统集成测试工作量的增加。另外，增加的人手需要原有成员停下手里的工作去帮助他们了解熟悉项目情况和当前进展，这又是额外的培训工作量。

因此，由于软件系统的结构复杂，各部分联系紧密，开发时沟通交流的工作量非常大，过度分解实际上增加了大量的额外工作量。复杂系统靠增加更多人力所带来的效益，还不足以抵消增加的额外负担，因而往往不是缩短时间，而是延缓进度。而这对于一般硬件产品来说是难以想象的。

所以，在进度允许的情况下，参与软件开发的人员应该少而精，并且在各阶段合理分布人力，使之与进度相协调。

开发高质量的软件是软件工程追求的重要目标，学习了《软件工程》课程后，简要谈谈

你认为应在软件生产和管理时采取哪些有效措施来保证软件质量。

答:实施有效的软件质量管理应做好以下几个方面的工作:

(1)采用先进的开发方法、工具和环境

(2）分阶段生产

(3）严格评审和质量控制 (4）加强测试

(5）制定规范(标准）并贯彻执行 (6）合理分工，职责分明

(7）实施软件变更控制 (8）进行软件质量度量

**软件过程：**

某大型企业计划开发一个“综合信息管理系统”，该系统涉及销售、供应、财务、生产、人力资源等多个部门的信息管理。该企业的想法是按照部门优先级别逐个实现，边应用。边开发。如果你作为乙方代表，请为该系统选择一个符合应用需求的软件开发过程模型，并详细说明选择的理由。

可采用**增量模型**。

它是瀑布模型与原型进化模型的综合，它对软件过程的考虑是:在整体上按照瀑布模型的流程实施项目开发，以方便对项目的管理;但在软件的实际创建中，则把软件系统功能分解为许多增量构件，并以构件为单位逐个创建与交付，直到全部增量构件创建完毕，并都被集成到系统之中交付用户使用。

**瀑布模型**是采用生命周期方法学的软件过程模型，它对于实施有效的软件管理有着重要

意义，试分析瀑布模型是如何辅助软件管理的。

瀑布模型采用“分而治之”的思想,从时间角度对软件开发和维护的复杂问题进行分解，将软件漫长的生命周期划分为若干阶段，逐步完成各个阶段的任务。每个阶段的任务相对独立，较为简单，便于不同人员的分工协作，从而降低了求解问题的复杂程度。开发过程从抽象到具体地逐步求精，各阶段有条不紊地开展工作和规范管理，有利于控制管理的难度，便于实现规范有效的软件管理。

各阶段结束的位置设置里程碑，从技术和管理两个角度进行严格的审查，合格之后才能开始下一阶段工作。利于尽早地发现错误，解除错误积累带来的风险，保证了软件质量。

各个阶段必须提交相应的文档，提高了软件管理的可见性，使得在编码之前实施有效的软件质量评价成为可能。为将来的维护工作也奠定了良好的基础。

可见，瀑布模型对于规范软件管理，保证软件质量和开发效率，合理进行任务的分解和人员组织，保证科学的管理技术和先进开发方法、工具的使用，以及软件配置的管理，都有着重要的意义和积极的作用。

某公司要开发一个应用系统，该系统的特点是:规模不大;工期紧;某些系统模块有类似软件可供参考;开发人员对应用领域知识知之甚少，但有条件和用户充分沟通;使用环境将来可能需要变更。在选择开发过程时项目组内部产生了分歧，有人支持瀑布模型，也有人建议采用原型方法。你会选择哪种开发模型呢?为什么?

答:**原型法**更适合这个项目。

原型法的主要思想是通过快速建立原型，启发、揭示、完善用户需求，统一用户和开发方对需求的理解。因而适用于对应用领域不熟悉的系统开发，通过和用户的反复沟通修改、反馈，再修改、再反馈，直至准确理解需求为止，从而大大降低了由于需求问题带来的软件开发风险。而瀑布模型开发时，要求需求一步到位，开发初期就确定下来，在进入设计之后尽量避免修改。因为后期需求变动带来的修改，必为之付出巨大代价。所以瀑布模型主要适用于需求很少变化、开发人员对领域很熟悉、用户使用环境稳定的系统开发而这个项目涉及不熟悉的领域知识，因此早期就确定并冻结全部需求是不现实的。

其次，该项目的某些模块有类似软件可供参考，实际上就是现成的原型可供使用。此外，原型法具有快速开发的优势，而瀑布模型开发周期相对长，在“工期紧”的情况下不适用。

**软件开发：**

为什么要引入面向对象方法学?

答:

瀑布模型的缺点:僵化

SA -SD - SP(结构程序设计）技术的缺点:本质上是基于过程的设计不易

被理解;且功能变化往往引起结构变化较大,稳定性不好。系统有明确的边界定义，且系统结构依赖于系统边界的定义，这样的系统不易扩充和修改。数据与操作分开处理，可能造成软构件对具体应用环境的依赖，可重用性(reusability)较差。

为什么说面向对象的方法为软件复用提供了**良好的环境**?。

在oo方法中重用层是类。类封装了数据和操作，提供了比较理想的模块化机制和比较理想的可重用的软件成分。另外，继承性机制使得子类不仅可以重用其父类的数据结构和程序代码，而且可以在父类代码的基础上方便地修改和扩充，而不影响对原有类的使用。但是，继承性并不是oo方法实现重用的唯一方式。如可以通过使用某个已有类的实例来重用其方法，而并非通过继承实现。

为什么说面向对象方法学开发出的系统比传统方法学开发出的系统**可重用性**好?

答题要点:传统的软件重用技术是利用标准函数库来构造新的软件系统，但是标准函数少必要的“柔性”，不能适应不同应用场合的不同需求，并不是理想的可重用的软件成分。在面向对象方法所使用的对象中，数据和操作正是作为平等伙伴出现的，因此，对象具有很强的自含性。此外，对象所固有的封装性和信息，隐藏机理，使得对象内部实现与外界隔离，具有较强的独立性。由此可见，对象是比较理想的模块和可重用的软件成分。

为什么说面向对象方法学开发出的系统比传统方法学开发出的系统**可维护性**好?

答题要点:传统方法以算法为核心，开发过程基于功能分析和分解。也就是用传统方法所建立起来的软件系统的结构紧密依赖于系统所要完成的功能,当功能需求发生变化时将引起软件结构的整体修改，不利于系统的维护。

面向对象方法以对象为核心，对象独立性强，需求变化只会引起对象中的属性与操作以及对象之间的消息通信方式的变化，不会引起结构的整体变化。另外，对象具有继承、多态等特点，使功能扩充或修改易于实现。

传统结构化方法缓解了软件危机，但不能很好地解决大型复杂软件难以维护的问题，面向对象方法则弥补了这一不足。为什么面向对象的开发方法可以提高大型软件的**可维护性**?

采用面向对象方法开发的软件可维护性好。

原因:

1、面向对象的软件稳定性比较好

传统的软件开发方法以算法为核心，开发过程基于功能分解。也就是用传统方法所建立起来的软件系统的结构紧密依赖于系统所要完成的功能,当功能需求发生变化时将引起软件结构的整体修改。事实上，用户需求变化大部分是针对功能的，所以，传统的方法建立起来的软件系统不稳定。

面向对象软件系统的结构是根据问题领域结构的模型建立起来，而不是基于功能的分解。所以，当系统的功能需求变化时，并不会引起结构的整体变化，只需局部性修改。例如，可以从已有类派生出新的子类来实现功能扩充或修改，等。所以，以对象为中心构造的软件系统是比较稳定的。

2、面向对象的软件比较容易修改

面向对象方法所使用的对象中，操作和数据是相辅相成的，对象具有很强的自含性。另外，对象所固有的封装性和信息隐藏机制，使得对象的内部实现与外界隔离，具有较强的独立性。另外，继承性机制使得子类不仅可以重用其父类的数据结构和程序代码，而且还可以在父类代码的基础上方便地修改和扩充，而且这种修改不会影响对原有类的使用。因而在面向对象方法下，软件的修改比较容易。

3、面向对象的软件比较容易理解

面向对象的设计方法的基本原理是:是按照人们习惯的思维方式建立问题域的模型,开发出尽可能直观、自然地表现求解方法的软件系统。面向对象系统中广泛使用的对象，实际上是一种抽象数据类型的实例（数据抽象和过程抽象)。另外，OO方法中通过建立类等级来获得继承特性，支持从一般到特殊的演绎思维过程。这些符合人们认识客观世界解决复杂问题时逐步深化的渐进过程。因而面向对象方法建立的软件系统的结构和问题空间的结构基本一致，易于理解。

4、面向对象的软件易于测试和调试

面向对象的软件的测试和调试工作主要围绕新派生出来的类进行的，类是独立性很强的模块，对类的测试比较容易实现，如果发现错误也往往集中在类的内部，比较容易调试

《软件工程》课程详细介绍了**传统的系统开发方法和面向对象（OO）方法**，对比二者的主要特点，回答下面的问题:

1.“分而治之”的是人类认识和求解复杂问题最有效的手段，SA/SD和OO方法中都用到了这个思想来组织和构造软件系统,但采用的系统分解方式有本质差异，请对比分析之。

两种方法都用到了“分而治之”的方法来控制求解软件的复杂性，但通过分解建立软件结构的方式则有本质区别。

传统的SA/SD方法强调功能分解，通过模块化分解系统，由模块之间的调用关系构成软件的层次结构。每个模块能够完成相对独立的功能，整个软件系统就是模块(函数或过程)的集合。功能分解方式使得建立起来的软件系统结构紧密依赖于系统功能，功能的变化易造成软件结构的整体变化，因此软件系统是不稳定的，维护工作量大。

面向对象方法强调对象分解，由客观世界实体抽象出问题域对象，并建立与问题域结构一致的求解域。求解域中的解空间对象实现对客观世界实体的直接模拟，将对象的属性和行为封装成一体。整个软件系统就是相互协作而又彼此独立的对象的集合。这种分解方式更利于人们采用习惯的思维方式思考和求解问题，实施有效的软件管理。由对象构成的软件系统稳定性好，功能需求的变化不会引起软件结构的整体变化，一般只需作局部修改，系统可维护性好。

2．采用传统方法开发的复杂软件难以维护，是长期困扰人们的一个严重问题，也是软件危

机的突出表现。为什么OO方法所开发的软件**可维护性**好?试说明原因。

面向对象的软件稳定性比较好，软件功能或性能变化，通常只需作局部修改，不变动软件结构，因而容易实现。

任何维护工作归根到底都要对软件进行修改，在修改之前首先要深入理解软件，才能正确地完成维护工作。传统软件之所以难以维护，很大程度上是由于修改所涉及的部分分散在各个模块，理解的工作量大，困难程度高。而0O软件由于采用对象分解技术建立的软件结构与问题域结构基本一致，易于理解。由于类和对象是独立性好的特殊模块，封装性使得修改局部化，较少考虑和影响软件的其他部分。另外，继承机制和多态性使得面向对象软件易于实现功能的修改和扩充，理解的工作量小，修改的工作量小。

为保证质量，对软件进行维护修改之后必须进行必要的测试。对类的测试通过向其实例发送消息即可运行它，测试容易实现。如果发现错误也往往集中在类的内部，容易调试。

综上所述，由于OO方法开发的软件具有稳定性好、容易理解、可修改性好、易于测试和调试的特点，使得软件的可维护性好。

**软件需求：**

根据你的开发经验，简述需求分析的困难在哪里?

答:一方面是由于交流障碍，另一方面是由于用户通常对需求的陈述不完备、不准确和不全面，并且还可能在不断地变化

**软件设计：**

什么是模块化?软件设计为什么要模块化?

答:模块化就是把程序划分成可独立命名且独立访问的模块，每个模块完成一个子功能，把这些模块集成起来构成一个整体，可以完成指定的功能满足用户的需求。采用模块化原理可以使软件结构清晰，不仅容易设计也容易阅读和理解。

在软件总体设计中，如何分解模块是首先要考虑的问题。模块分解有两种常用的方式:水平分解和垂直分解。请分别说明采用这两种分解方式的特点，并分析其优缺点。

**水平分解**:按功能模块划分。顶层模块是控制模块，用来协调程序各个功能之间的通信和运行，下级模块划分为输入、处理和输出。即，先不研究整个问题，而是研究问题的一部分，分割我要，各个击破。

优点:主要的功能相互分离，易于修改、易于扩充、易于实现。程序员没必要和过多的人沟通、写作，自己一个人完成某模块由上至下的所有代码。从团队协作、代码安全等角度考虑优于垂直划分。

缺点:需要通过模块接口传递更多的数据，使程序流的整体控制复杂化。

**垂直分解**:也叫因子划分。工作自顶向下逐层分布。顶层模块执行控制功能，少做实际处理工作;而底层模块是实际输入、计算和输出的具体执行者。即，先不把问题研究得那么深、那么细，浅尝辄止，见好就收。

优点:对底层模块的修改不太可能引起副作用的传播，而恰恰对计算机程序的修改常常发生在底层的输入、计算或输出模块中，因此，程序的整体控制结构不太可能被修改，便于将来的维护。

缺点:不利于团队协作。如果一个项目需要3个人完成，一人负责一层。即使很小的功能前前后后需要3个人协作完成。

什么是**特征耦合**?举例说明特征耦合对数据安全性的影响?

答题要点:特征耦合:一个模块访问另一个模块时，传递的是整个数据结构（地址)。调用模块只需要使用整个数据结构中的一部分数据元素。被调用模块可使用的数据多余它确实需要的数据，将导致对数据的访问失去控制，从而给计算机犯罪提供了机会。

**软件测试：**

规定文件名是“以字母开头，由字母、数字组成的字符串”，采用等价类划分方法设计测试方案。某同学设计了如下测试用例，你认为下列测试用例哪些无效?为什么?

1. file1 B. 2file C. file\_3 D. 4file\_1

答:D不合理;

因为采用等价划分法设计测试用例时，为了避免多个不合理的等价类交叉在一起影响测试效果，所以要求设计出的每个测试用例只包含一个尚未被包含的不合理的等价类。题中选用“4file\_1”作为测试用例，程序运行出错时，难以判断是由于文件名中的第一个字符不合法，还是由于其余的字符中有不合理的字符引起的。所以“4file\_1”作为测试用例不合理。

**软件维护：**

软件维护是软件生命周期的重要组成，而影响维护的因素较多，比如: a)软件系统的规模;b)软件系统的年龄;c)软件系统的结构。

试回答以下问题:

1)给出上述三项因素的解释，阐述“规模”、“年龄”和“结构”对维护工作量影响的具体

含义。

2)除了上述三个因素外,你认为还有什么因素会对软件维护工作量的因素产生影响?怎么

影响?

答题要点:

系统的规模。指软件系统规模的大小，系统规模越大,维护困难越大。

系统的年龄。系统运行时间越长，可能经过多次修改，从而造成维护的困难。系统的结构。系统结构合理与否，会给维护带来困难。

增加维护工作量的因素还有:程序复杂性、用户的数量、应用的变化和文档的质量。

减少维护工作量的因素有:结构化技术的使用、自动化工具的使用、数据库技术应用、高性能软件使用、新的软件技术的应用。

为什么说维护的代价很高?应该怎样做才能降低维护的代价?

一般维护的工作量占生存周期70%以上,维护成本约为开发成本的4倍(满足8-2规则)。维护代价分为:

1．有形代价:费用已上升至总预算的80%;

2．无形代价:占用资源以致延误开发;修改不及时引起用户不满;维护引入新错误，降低了软

件质量;等等。

1. 维护工作量的经验模型:M = P ＋K\*ec-d降低维护代价的通常认识是:提高程序的可维护性，这也是软件工程学的主要目的

某些软件工程师不同意“目前国外许多软件开发组织把60%以上的人力用于维护已有软件”的说法，他们说:“我并没有花费我60%的时间去改正我的程序中的错误”。你如何看?

首先，软件维护并非仅仅是改正程序中的错误，它还包括为了使软件适应变化了的环境而修改软件的活动，以及为了满足用户在使用软件的过程中提出的新需求而修改软件的活动，甚至包括为了提高软件未来的可维护性或可靠性而主动地修改软件的活动。实际上，为了消除程序中潜藏的错误而进行的改正性维护，仅占全部维护活动的20%左右。

其次，“目前国外许多软件开发组织把60%以上的人力用于维护已有软件”，指的是软件开发组织内人力分配的整体状况，至于具体到软件组织内的每位软件人员，则分工不同。有些人专职负责软件维护工作，他们的全部工作都花费在维护已有的软件产品的工作上，另一些人专职负责软件开发工作，他们可能并不花费时间去维护已有的软件产品，还有一些人既要从事软件开发工作又要兼管软件维护工作。

第三，软件维护人员并非只负责维护自己开发的程序，通常，一名维护人员参与多个软件产品的维护工作。