软件工程

本题库可用来参考复试出题范围，答案可能存在一些问题，感谢群友们随时指出~

1、为什么软件维护的成本越来越高

可从需求分析 系统设计 详细设计开展

软件维护不仅针对程序代码，而且还针对软件定义、开发的各个阶段生成的文知档。而软件在设计阶段很难预料到这个软件交给谁，在什么时候进行什么样的维护工作。软件维护的依据只能靠软件文档和有关的设计信道息。这样，软件维护人员不得不花费大量版的劳动，用语软件系统的再分析和对软件信息的理解。因此软件的维护费用也就一直高居不下。

主要困难在于怎样修改程序代码是软件适应用户需求。

解决这些权困难则需要进行四个阶段：状态捕获、问题定义、技术开发、方案综合。

软件的可维护性、维护过程管理是否规范等对于软件维护成本具有举足轻重的作用。为提高软件的可维护性，在软件开发阶段就必须采取相关措施，如编制相关软件文档、规范软件编码规范、进行充分有效的测试、组织阶段评审等，而在软件提交用户使用后，必须规范维护过程管理。另外，各过程中可使用相关的辅助工具软件。

2、软件测试的结束标准是什么？覆盖率能不能达到100%？

结束标准：

1) 软件系统在进行系统测试过程中，发现一、二级缺陷数目达到项目质量管理目标要求，测试暂停返回开发；

2) 软件项目在其开发生命周期内出现重大估算和进度偏差，需暂停或终止时，测试应随之暂停或终止，并备份暂停或终止点数据；

3) 如有新的需求变更过大，测试活动应暂停，待原测试计划和测试用例修改后，再重新执行测试；

4) 若开发暂停，则相应测试也应暂停，并备份暂停点数据；

5) 所有功能和性能测试用例100%执行完成； 此外，测试是有成本的，当你2周发现2个bug有类似此种情况时，在产品质量要求不是十分严格的情况下，即可以停止测试了。

没有绝百对的答案，只有相对的答案，最重要的是要根据实际情况来。

覆盖率：

不能。

测试覆盖率一般常来衡量测试的完整性和充分性。从广义角度来讲，分为面向项目的需求覆盖率和面向技术的代码覆盖率。

如果我们简单的将测试覆盖率理解为需求覆盖率，代码覆盖率，那么我想这是可以达到的。

但是我们还要考虑另外一个问题，是否由我们未曾列入到需求分析中的需求呢，这种情况是存在的，那么我们实际的“需求覆盖率”就不是100%了。在实际开发过程中，是不可能将需求全部列出来的，所以我们得到的100%的需求覆盖率是存在水分的。

另外，对于一个应用程序(除了一些极其简单的程序)来讲，要覆盖到所有的语句。条件，分支是极其困难的，甚至可以说是不可能的。

3、为什么经过单位测试和集成测试后还要系统测试

单元测试的主要目的是验证软件模块是否按详细设计的规格说明正确运行

集成测试主要目的是检查多个模块间是否按概要设计说明的方式协同工作。

系统测试的主要目的是验证整个系统是否满足需求规格说明。

单元测试是测试各个小zhi的模块，通过对他们的测试，才能找出基本的bug，然后dao为各个模块搭建接口，也就是把模块组装起来，之后进行集成测试，看各个模块的接口是回否正常稳定，打包成软件答后，先做出一个demo版本，由开发和测试一起进行系统测试。

4、基础路径测试的定义和基本步骤？

定义：软件测试的基本路径测试是指根据路径设计测试用例的一种技术，经常用于状态转换测试中。在程序控制流图的抄基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例的方法。设计出的测试用例要保证在测试中程序bai的每个可执行语句至少执行一次。

步骤：属于白盒测试，画出程序控制流图，通过分析控制构造的环路复杂性，从而设计出测试用例的方法。

5、白盒测试的逻辑覆盖测试，有哪几种测试方法？以及他们的优劣？

逻辑覆盖包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖。

语句覆盖法：可以很直观地从源代码得到测试用例，无须细分每条判定表达式；该测试用例虽然覆盖了可执行语句，但并不能检查判断逻辑是否有问题，例如在第一个判断中把&&错误的写成了||，则上面的测试用例仍可以覆盖所有的执行语句。 一般认为“语句覆盖”是很不充分的一种标准，是最弱的逻辑覆盖准则。 判定覆盖法： 能够满足条件覆盖的要求，但是也不能对判断条件进行检查； “判定覆盖”比“语句覆盖”严

格，因为如果每个分支都执行过了，则每个语句也就执行过了。但是，“判定覆盖”还是很不够的。 条件覆盖法：“条件覆盖”通常比“判定覆盖”强，因为它使一个判定中的每一个条件都取到了两个不同的结果，而判定覆盖则不保证这一点。 要达到条件覆盖，需要足够多的测试用例，但条件覆盖并不能保证判定覆盖。条件覆盖只能保证每个条件至少有一次为真，而不考虑所有的判定结果。 判定/条件覆盖法：判定/条件覆盖从表面来看，它测试了所有条件的取值，但是实际上某些条件掩盖了另一些条件。条件组合覆盖法：上面的测试用例覆盖了所有条件的可能取值的组合，覆盖了所有判断的可取分支，但是却丢失了一条路径abe 路径覆盖法：这种测试方法可以对程序进行彻底的测试，比前面五种的覆盖面都广；由于路径覆盖需要对所有可能的路径进行测试（包括循环、条件组合、分支选择等），那么需要设计大量、复杂的测试用例，使得工作量呈指数级增长(而指数级增长通常是设计算法时要尽力避免的)

6、回归测试和为什么回归测试要自动完成

概念：回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误2113或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护5261升级等阶段的成本。 即项目测试完为了保证质量再做一次全面的测试。 回归测试包括两部分：函数本身的测试、其他代码的测试。

原因： 自动化测试比人工更快;自动化测试可以在任何时间进行;自动化测试非常精确;自动化可以用在几乎所有的测试过程中；可以自动创建报告。

7、如何设计测试用例。

测试用例：测试用例（Test Case）是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。

测试用例编写流程：

需求分析--》提取测试点--》测试用例编写--》测试用例评审

（可具体展开）

8、软件工程中有哪两种测试方法，各是什么含义，分别用在哪种情况

（两种划分应该是从结构上分为黑盒和白盒测试）

黑盒测试（black-box testing）：忽略程序内部机制，测试程序的每项功能是否符合要求，也称为能性测试。**应用于**只关心输入和输出的结果情景。

白盒测试（white-box testing）：利用程序内部逻辑结构及有关信息，设计测试用例测试，也称结构性测试。**应用于**去研究里面的源代码和程序结构的情景。

9、什么是交付测试，解释一下 alpha 测试、 beta 测试

交付测试：交付过程中保证用户对所交付的系统的满意的测试。交付测试主要的参与者应该是目标客户。客户参与越多越好。交付测试的内容一般包括前面提到的安装测试、可用性测试、 alpha 测试、 beta 测试等。

Alpha测试：是一种非正式验收测试，是由一个用户在开发环境下进行抄的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。

Beta测试：是一种验收测试，bai是软件产品完成了功能测试和系统测试之后，在产品发布之前所进行的软件测试活动。

10、需求分析和程序设计有什么区别?

需求分析是开发人员经过深入细致的调研和分析，准确理解用户和项目的功能、性能、可靠性等具体要求，将用户非形式的需求表述转化为完整的需求定义，从而确定系统必须做什么的过程。

设计的起点是需求。在产品生命周期中，需求是一个动态变化的过程，产bai品可分为：导入期、成长期、成熟期和衰退期，产品在不同阶段有着不同的需求，而且需求的种类也不同。

程序设计就是根据前期的调查，分析专，设计文档来进属行程序设计（详细代码编写）。

11、什么是结构分析法，它的工具有哪些?

结构化分析方法(Structured Method)是强调开发方法的结构合理性以及所开发软件的结构合理性的软件开发方法。

结构化分析方法给出一组帮助系统分析人员产生功能规约的原理与技术。它一般利用图形表达用户需求，使用的手段主要有数据流图、数据字典、结构化语言、判定表以及判定树等。

12、什么是等价类?

等价类划分，指的是一种典型的、重要的黑盒测试方法。其就是解决如何选择适当的数据子集来代表整个数据集的问题，通过降低测试的数目去实现“合理的”覆盖，以此发现更多的软件缺陷，统计好数据后由此对软件进行改进升级。

等价类划分法将程序所有可能的输入数据（有效的和无效的）划分成若干个等价类。然后从每个部分中选取具有代表性的数据当做测试用例进行合理的分类，测试用例由有效等价类和无效等价类的代表组成，从而保证测试用例具有完整性和代表性。