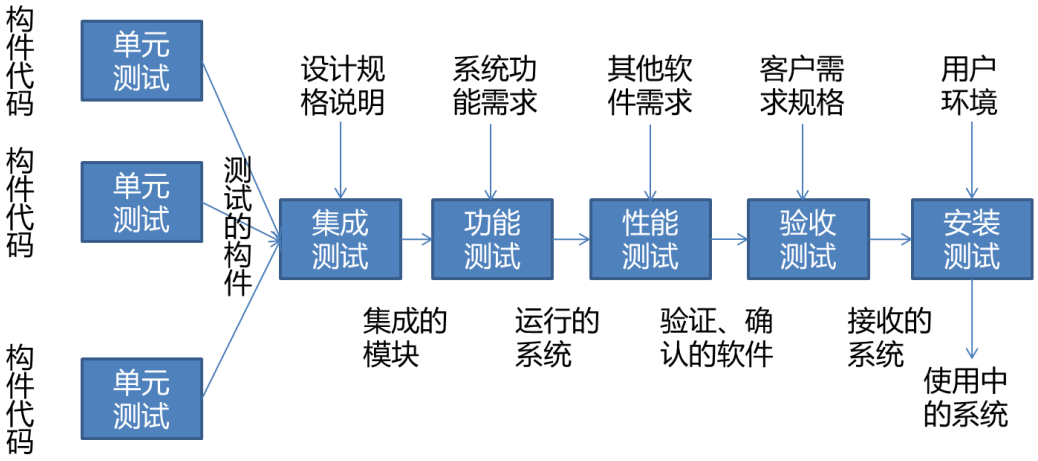
1、 软件测试包含哪些步骤及这些步骤面对哪些对象。

1) **单元测试：**将每个程序构件与系统中其他构件隔离，对其本身进行测试，也称为模块测试或者构件测试，它们验证针对设计预期的输入类型，构件能否适当地运行，其结果是产生测试的构件。面向的对象是**构件代码**。
2) **集成测试：**验证系统构件是否能够按照系统和程序设计规格说明中描述的那样共同工作的过程，其结果是产生集成的模块。面向的对象是**设计规格说明**。
3) **功能测试：**对系统进行评估，以确定集成的系统是否确实执行需求规格说明中描述的功能，其结果是产生运行的系统。面向的对象是**系统功能需求**。
4) **性能测试：**将系统与软件和硬件需求的剩余部分进行比较，当测试在客户的实际工作环境中成功地执行时，其结果是产生一个验证、确认的软件系统。面向的对象是**其他软件需求**。
5) **验收测试：**与客户协商，确保系统是按客户的期望运转，根据客户的需求描述对系统进行检查，其结果是产生接收的系统。面向的对象是**客户需求规格说明**。
6) **安装测试：**确保系统将按照它应该的方式来运行，其结果是产生使用中的系统，面向的对象是**用户环境**。

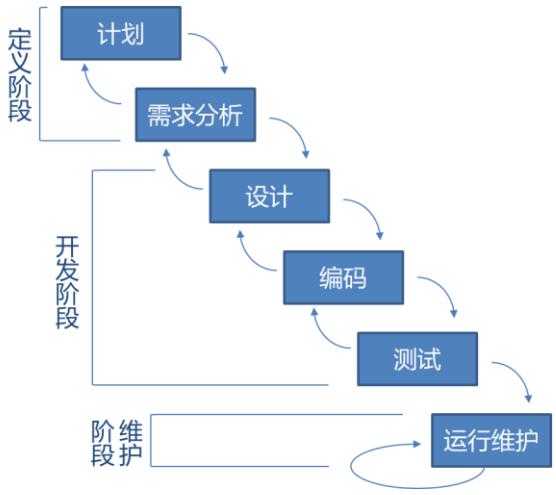


2、 软件测试和调试的目的有什么区别。

1) **软件测试：**由独立的测试组在不了解软件设计的条件下完成，以暴露软件错误为目标，有计划地从已知条件开始，使用预先定义的程序，有预知的结果。
2) **软件调试：**由程序作者进行，以找出软件错误位置并修正为目标，被动地从不可知内部条件开始，结果一般不可预见。

3、 “瀑布模型”主要阶段性的基本活动？"瀑布模型"的缺陷及其造成缺陷的原因？

瀑布模型的主要阶段性基本活动：根据软件生存周期模型各个阶段的任务，从**可行性研究开始，经过需求分析、概要设计、详细设计、编码和调试、单元测试、组装测试**各个阶段的转变、直到确认测试并得到用户确定的软件产品为止。



1) 瀑布模型的特点：
   a) 阶段间具有顺序性和依赖性。
   b) 推迟实现的观点。
   c) 每个阶段必须完成规定的文档；每个阶段结束前完成文档审查，及早改正错误。
2) 瀑布模型的缺点：
   a) 实际的项目很少按照该模型给出的顺序进行。
   b) 用户常常难以清楚地给出所有需求。
   c) 在软件开发初期指明软件系统的全部需求是困难的，甚至不现实。
   d) 需求确定后，用户和项目负责人要等相当长的时间才能得到项目初期版本。

   造成以上缺点的原因是没有把软件看做一个问题求解的过程。

4、需求工程的基本活动及其主要任务。

1) 需求工程的基本活动。

   需求获取；需求分析；需求规格说明；需求验证；需求变更管理。
2) 需求工程的主要任务。
   a) **需求获取：**通过与用户的交流，对现有系统的观察及对任务进行分析，从而开发、捕获和修订用户的需求。
   b) **需求建模：**为最终用户所看到的系统建立一个概念模型，作为需求的抽象描述，并尽可能多的捕获现实世界的语义。
   c) **形成需求规格：**生成需求模型构建的精确的形式化描述，作为用户和开发者之间的一个协约。
   d) **需求验证：**以需求规格说明为输入，通过符号执行、模拟或快速原型等途径，分析需求规格的正确性和可行性。
   e) **需求管理：**支持系统的需求演进、如需求变化和可跟踪性问题。

5、如何得到目标系统的逻辑模型。

1) 通过对显示环境的调查，获得当前系统的物理模型。
2) 去掉具体模型中的非本质因素，抽象出当前系统的逻辑模型。

3) 分析当前系统与目标系统的差别，建立目标系统的逻辑模型。

6、压力测试有什么用，适合哪些测试。

压力测试检查程序对异常情况的抵抗能力，是检查系统在极限状态下运行的时候性能下降的幅度是否在允许的范围内，测试总是迫使系统在异常的资源配置下运行。

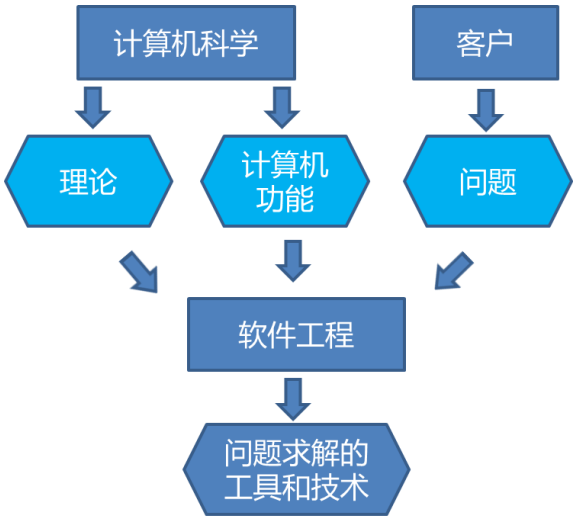
1) 适合于对系统的稳定性进行测试。
2) 适合于对系统的性能及扩展性进行测试。
3) 如果需求规定，系统将处理高达某个指定数目的设备或用户，则压力测试在所有设备中用户同时处于活动状态的时候测试系统的性能。
4) 有的系统通常是在低于其最大能力的情况下动作的，但是，在某个高峰时间，系统会受到严重压力，对于这样的系统进行压力测试则非常必要。

7、净室开发方法的关键特性。

1) 根据规格说明证明软件，而不是等待单元测试发现故障。
2) 产生零故障或接近零故障的软件。
3) 使用盒式结构来说明软件。
4) 将盒式结构转化为功能，并用正确性形式化证明验证正确性条件。
5) 使用概率来随机选择测试用例完成统计使用测试。
6) 净室方法改进了软件质量，但没有单元测试，增加了失误的危险。

8、软件工程与计算机科学的联系和区别。

1) 区别：计算机科学更关注计算机本身的理论和结构，集中精力研究计算机和程序设计语言，搜索正确的计算和建模方法，从而改进计算方法本身；而软件工程把计算机看作是用于设计和实现问题解决方案的工具，在时间、资源、人员这三个主要限制条件下构建满足用户需求的软件，而不是研究硬件的设计或者算法的理论证明。
2) 联系：软件工程依赖于计算机科学提供的理论和计算机功能，为客户提出的问题提供求解的工具和技术。



9、 回归测试的概念及其基本步骤。

1) 回归测试概念：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或导致其他代码产生错误。回归测试是指重复以前的部分或者全部测试，因为新加入的模组可能对旧的模组产生副作用,所以需要进行某种程度的回归测试。
2) 回归测试的基本步骤：
   a) 识别出软件中被修改的部分。
   b) 从原基线测试用例库中，排除所有不再使用的测试用例，确定那些对新版本依然有效的测试用例，其结果是建立一个新的基线测试用例库。
   c) 依据一定的策略从新的用例库中选择测试用例测试被修改的软件。
   d) 如果必要,生成新的测试用例集,用于测试原用例库无法充分测试的软件部分。
   e) 用新生成的用例执行修改后的软件

10、 CMMI 的概念及其等级。

CMMI 即能力成熟模型集成，是一种过程改进的方法，为组织机构提供了有效过程的基本元素。它可以用于指导跨项目、部门或者整个主旨的过程改进。等级有：

1) 初始级：软件过程是无序的，对过程几乎没有定义。管理是反应式的。
2) 可重复级：建立了基本的项目管理过程来跟踪费用、进度和功能特性。
3) 已定义级：已将软件管理和工程两方面的过程文档化、标准化，并综合成该组织的标准软件过程。
4) 量化管理级：分析对软件过程和产品质量的详细度量数据，对软件过程和产品都有定量的理解和控制。
5) 优化管理级：使用过程的量化反馈和先进的思想、新技术促使过程持续不断改进。

11、 Lehman 的系统分类观点及其类别特征的特点。

1) Lehman 的系统分类观点：描述了一种根据程序如何变化对它们进行分类的方法。
2) S 系统：有些系统根据形式化的定义进行设计并实现，这种系统是静态的，只要最初的问题不变化，解决方法就不会变化。如果问题变化了，则变成了一个全新的系统，是另外的问题，而不是基于原系统的演化。
3) P 系统：虽然可以利用 S 系统定义并解决某些抽象问题，但很多情况下，一个问题存在理论上的解决方案，但实现该方案几乎不可能。为了开发这种方案，我们必须先以比较抽象的方式对问题进行描述，然后根据我们的抽象描述，编写系统的需求规格说明。
4) E 系统：是融入现实世界中的系统，并随着现实世界的变化而改变。其解决方案是基于对抽象过程监理的模型。

12、 软件再生包含了哪几部分。

1) **文档重构：**是指对源代码进行静态分析以产生系统文档。我们可以检查变量使用、构件调用、控制路径、构件规模、调用参数、测试路径以及其他相关的测量，来帮

助我们理解代码是做什么的以及是如何做的。

2) **重组：**重组目的是使它更容易理解和改变。工具通过解释源代码以及用内部形式表示源代码来帮助我们完成这一任务。接着利用转换规则来简化内部表示。
3) **逆向工程：**从源代码得到软件系统的规格说明和设计信息，它尽量基于软件规格说明和设计方法恢复工程性信息，并随后以某种方式存储这些信息，以使我们能够对其进行处理。
4) **再工程：**是逆向工程的扩展。逆向工程抽取出信息，而再工程在不改变整个系统功能的前提下，生产新的软件源代码。

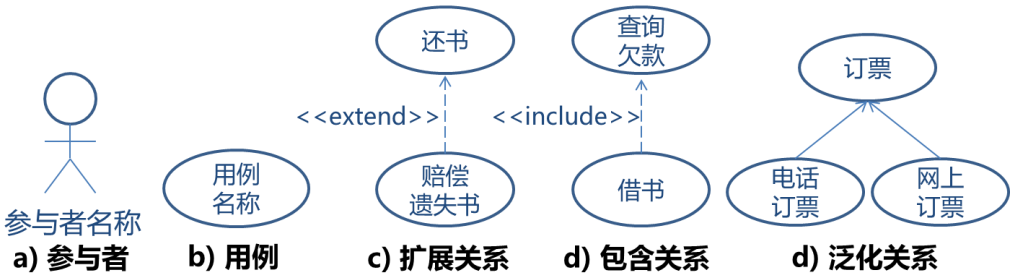
13、 集成测试。

1) 自底向上集成：每一个处于系统层次中最底层的构件首先被单独测试，接着测试那些调用了前面已测试后街的构件。反复采用此种方法，直到所有构件被测试完毕。
2) 自顶向下集成：顶层构件通常是一个控制构件，独立进行测试。然后将被测构件调用的所有构件组合起来，作为一个更大的单元进行测试。反复执行这种方法，直到所有构件都被测试。
3) 一次性集成 :将所有构件分别经过测试 ,再讲它们合在一起作为最终吸引进行测试。
4) 三明治集成 :将自顶向下和自底向上结合起来。将系统看成三层 ,目标层处于中间、目标上面的一层和目标下面的一层。在顶层使用自顶向下方法，在较低层次寻找自底向上方法，测试集中于目标层。目标层根据系统特性和构件层次结构来选择。

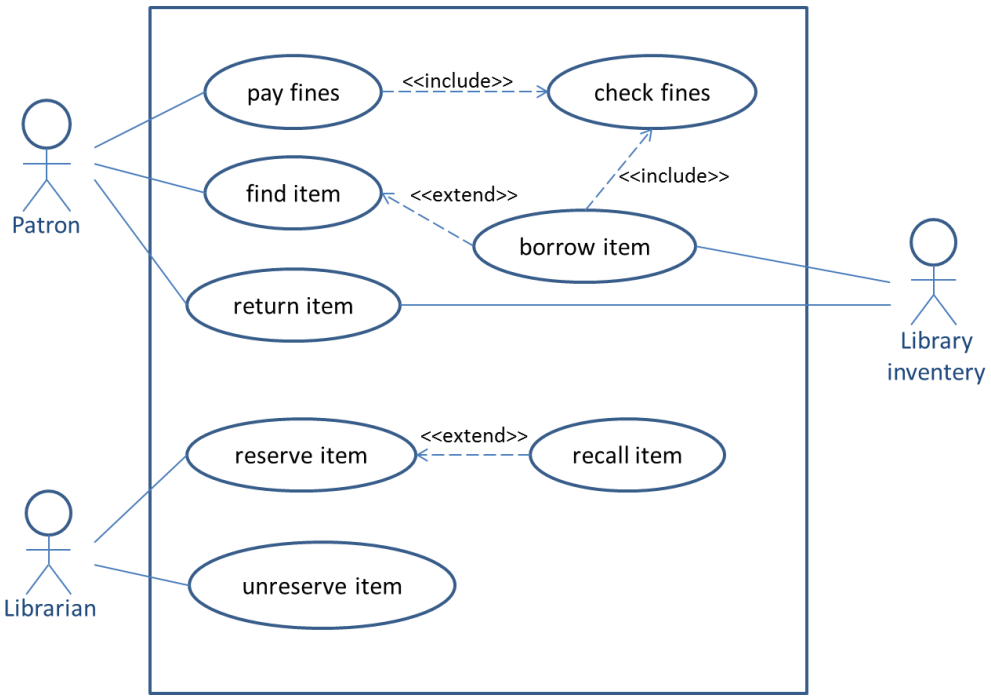
14、 UML、UML 用例图、UML 类图、UML 类关系图。

UML 是统一建模语言，用于文档化软件规格说明和设计的一组表示法。

1) **用例图：**根据系统和系统的环境之间的交互，描述可观察到的、用户发起的功能。
   a) 参与者：用小人表示，包括人或系统。
   b) 用例：用椭圆表示，表示必需的主要功能及其变种。
   c) 扩展关系：
   d) 包含关系：
   e) 泛化关系：



给个图书馆的例子：



2) **UML 类图**：在所有 UML 规格说明中，类图是“旗舰性”的模型，它是与规格说明中的类相关的高级 ER 图。每个类用一个 3 行 1 列的方框表示，其中第一行为类名，第二行为成员，第三行为方法。类与类之间参考 UML 类关系图，其中要注意的是“类范围属性”，即被类的所有实例共享的数值（类似 static），该数值要画下划线。

publication
call_number title value depreciation loan_period fine_rate reserve_loan_period reserve_fine_rate recall_period
find(title):Publication buy() lose() borrow() return() reserve() unreserve() decreasevalue()

3) **UML 时序图**：是强调消息时间顺序的交换图，描述了对象之间传送消息的时间顺序，用来表示用例中的行为顺序。【P112-P113】

a) 实体：使用方框表示。

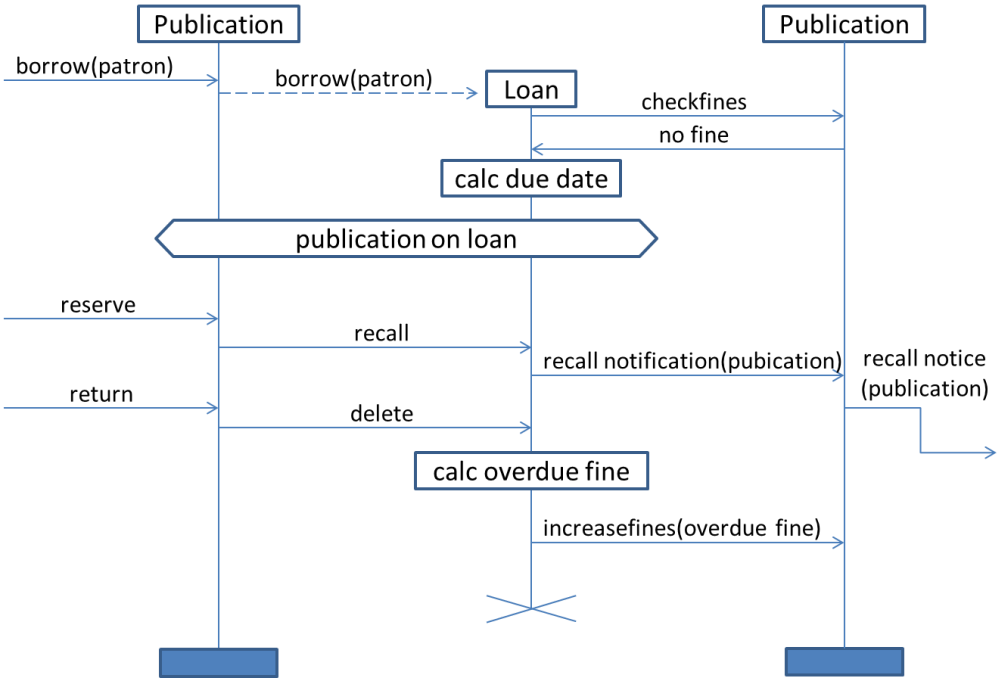
b) 时间线：在实体下方的竖线，表示实体的生存时间。

c) 消息的发送：使用箭头表示，箭头上的标记指定该消息的名称和数据参数。若箭头有折线，则体现了消息发送时间和消息接收时间之间的时间段。虚线箭头表示创建事件，它产生新的实体。

d) 实体的消亡：实体线底部的交叉符号，表示实体执行的终结。实体线底部的实心矩形表示实体的规格说明的结束，而并不意味着执行的终结。

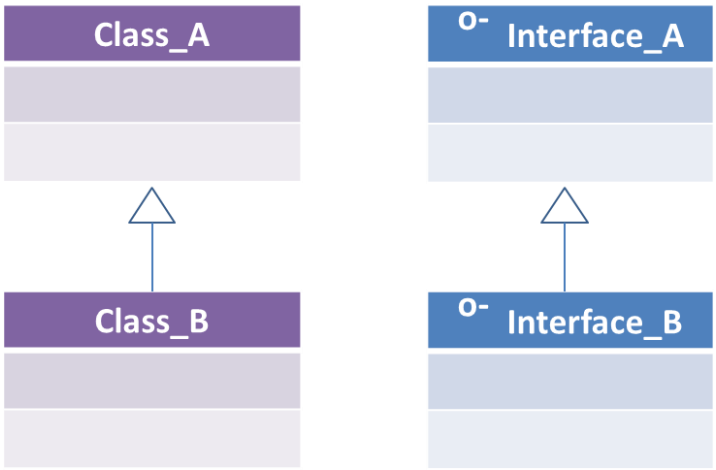
e) 动作：例如被调用的操作或对变量值的改变等，表示为位于实体执行线上的带标记的矩形，处于足迹中动作发生的那一刻。

f) 条件：可以将一个实体演化的重要状态指定为条件，用有标记的六边形表示，然后可以指定条件之间的一组子踪迹，并通过在实体状态相同的时刻组合消息时序图来得出各种踪迹。

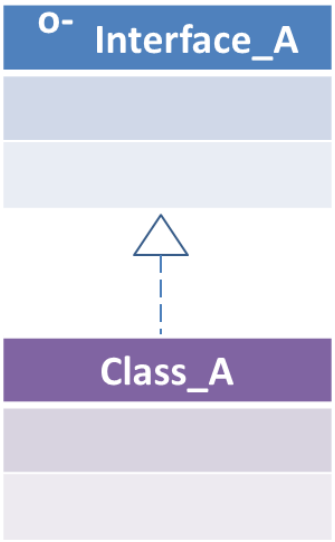


4) **UML 类关系图**：

a) 继承：一个类继承另一个类的功能。

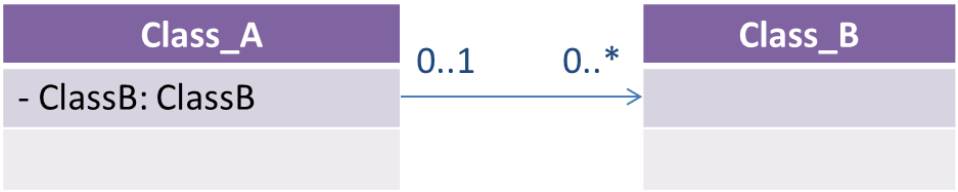


b) 实现：一个类实现接口的功能。

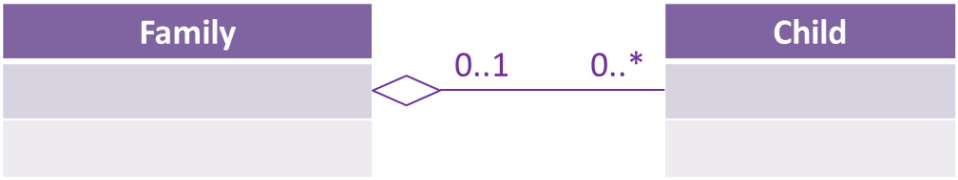


c) 依赖：类 A 使用到了另一个类 B。

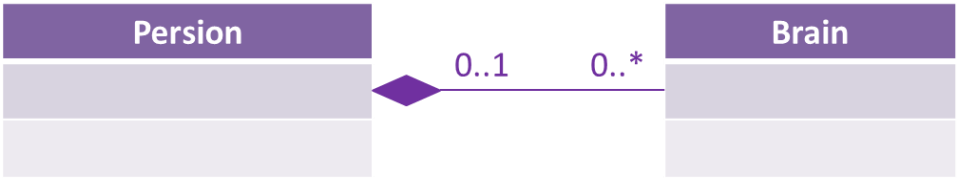
d) 关联：体现两个类或两个接口之间的一种强依赖关系。数字表示多重关联，如下面的表示每个类 B 只能和一个类 A 关联，但类 A 可以和无限个类 B 关联。



e) 聚合：是关联关系的一种特例，体现的是附属、拥有的关系，即 has-a。其中空心菱形一端为包含者（整体），另一端为被包含者（部分）。

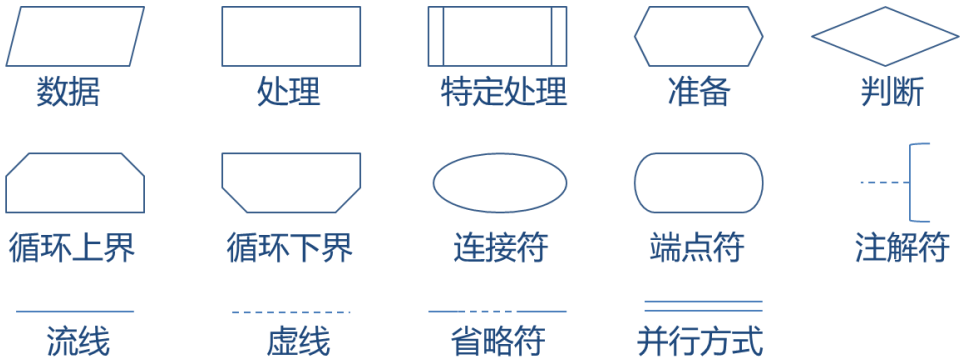


f) 组合：也是关联关系的一种特例，体现的是一种 contains-a 的关系，复合类是物理上由成分类的实例组成的，这种关系更强，也称强聚合。

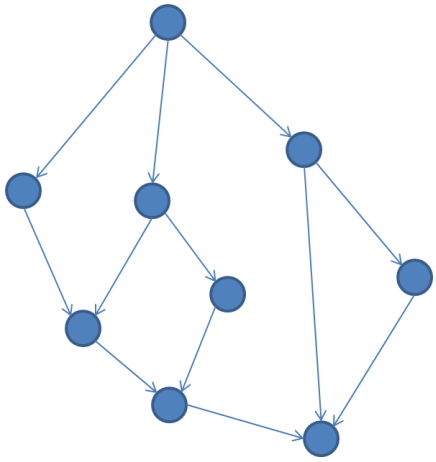


15、 程序流图，环路复杂度计算。

1) 程序流图的符号：



2) 环路复杂度计算：V(G) = E - N + 2
E 为流程图中的边数，N 为流程图中的结点数。
如下图中，边数为 12，结点为 9，则 V(G) = 12-9+2 = 5。



16、 软件工程的定义与提出的时间。

1) 目前比较认可的一种定义：软件工程是研究和应用如何以系统性的、规范化的、可定量的过程化方法去开发和维护软件，以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来。
2) 软件工程的提出时间：1968 年这一术语在 NATO 会议上首次提出使用。

17、 测试之间的不同，开盒和闭盒，白盒黑盒。

1) **闭盒：**如果从外部观察测试对象，将其看做是一个不了解其内容的闭盒，那么，在测试的时候向闭盒提供输入数据，并记录产生的输出。在这种情况下，测试的目标是确保每一种输入都被提交。缺点：不能总是用这种方法进行完备的测试，测试小组不可能生成一组证明所有情况下功能正确的、有代表性的测试用例。
2) **开盒**:将测试对象看作一个开盒,可以根据测试对象的结构用不同的方式进行测试。例如，可以设计执行构件内所有语句或所有控制路径的测试用例。缺点：这种方法可能不切实际，工作量太大，且难以进行完全的测试。
3) **黑盒测试：**指把测试对象看成一个黑盒子，测试人员不考虑程序的内部结构和处理过程，只在软件的接口处进行测试，依据需求规格说明书，检查程序是否满足功能

要求，又称为功能测试或数据驱动测试。

4) **白盒测试：**把测试对象看成一个打开的盒子，测试人员需了解程序的内部结构和处理过程，以检查处理过程的细节为基础，对程序中尽可能多的逻辑路径进行测试，检验内部控制结构和数据结构是否有错，实际的运行状态与预期的状态是否一致。
5) 测试方法的选择依赖因素有：
   a) 可能的逻辑路径数目。
   b) 输入数据的性质。
   c) 涉及的计算量。
   d) 算法的复杂性。

18、 软件设计过程包含哪些阶段，哪些活动。

软件设计是把软件需求变换成软件表示的过程，主要包含了：

1) **数据/类设计：**将分析-类模型变换成类的实现和软件实现所需要的数据结构。
2) **软件体系结构设计：**体系结构设计定义了软件的整体结构。
3) **接口设计 :**接口设计描述了软件内部、软件和协作系统之间及软件同人之间的通信。
4) **部件级设计 :**部件级设计将软件体系结构的结构性元素变换为软件部件过程性描述。

19、 回归测试的概念及其基本步骤。

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或导致其他代码产生错误。回归测试是指重复以前的部分或者全部测试，因为新加入的模组可能对旧的模组产生副作用，所以需要进行某种程序的回归测试。

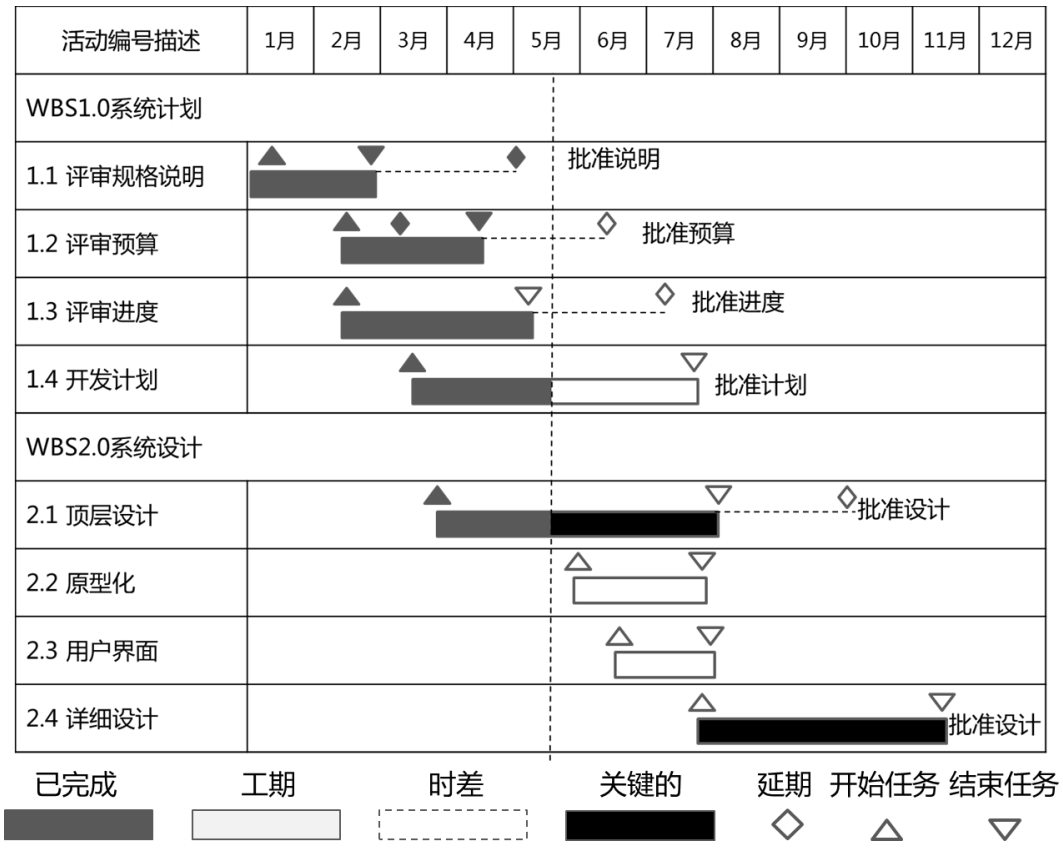
1) 识别出软件中被修改的部分。
2) 从原基线测试用例库中，排除所有不再使用的测试用例，确定那些对新版本依然有效的测试用例，其结果是建立一个新的基线测试用例库。
3) 依据一定的策略从新的用例库中选择测试用例测试被修改的软件。
4) 如果必要，生成新的测试用例集，用于测试原用例库无法充分测试的软件部分。
5) 用新生成的用例执行修改后的软件。

20、 需求过程包含哪些阶段。

1) **引发需求：**分析员通过提问、检查当前行为、示范类似系统与客户一起引发需求。
2) **分析需求：**用模型或原型来获取需求，有助于我们更好地理解需要的行为，并且通常会引发关于客户在一定情况下想要什么的额外问题。
3) **规格说明：**决定哪些必需的行为将在软件中出现。
4) **需求确认：**检查我们的规格说明是否与客户期望在最终产品中看到的相匹配。
5) **形成软件需求规格说明书：**最终产生的结果，用于与其他软件开发人员交流，探讨最终产品的行为。

21、 甘特图的意义和绘画。

甘特图能够绘制工作分解结构，是对项目的描述，显示在什么地方活动是并行进行的，并用颜色或图标来指明完成的程度。



22、 面向对象编程的特征。

1）**标识：**指将数据组织为离散的、可区别的实体，称为对象。

2）**抽象：**抽象有助于表示正在开发的系统中的不同观点。全起来，这些抽象开成一个层次，以说明不同的系统观点是如何彼此相关的。

3）**分类：**将具在共同属性的行为的对象分组。

4）**封装：**类封装对象的行为和属性，隐藏实现细节。

5）**继承：**根据类之间的相同性或差异性来组织类层次。

6）**多态：**同样行为在不同的类或子类中会有不同的表现的性质。

7）**持久性：**即对象的名称、状态和行为超越时间或空间的能力。