从是否关心软件内部结构和具体实现的角度划分

（以实现算法细节和系统内部结构相关的情况为根据）

白盒测试、黑盒测试、灰盒测试

**(1)白盒测试**：又称为结构测试或逻辑驱动测试，是一种按照程序内部逻辑结构和编码结构，设计测试数据并完成测试的一种测试方法。

其与黑盒测试不同，它主要是借助程序内部的逻辑和相关信息，通过检测内部动作是否按照设计规格说明书的设定进行，检查每一条通路能否正常工作。白盒测试是从程序结构方面出发对测试用例进行设计。其主要用于检查各个逻辑结构是否合理，对应的模块独立路径是否正常以及内部结构是否有效。常用的白盒测试法有控制流分析、数据流分析、路径分析、程序变异等，其中逻辑覆盖法是主要的测试方法。

逻辑覆盖法：

1.语句覆盖。 这个是起码要做到的覆盖了，程序里的每条可执行的语句都要至少执行一次。这个设计起来比较简单，用例数据很直观的就能看出来。但是语句里的判定，分支等就没什么意义了。可以说这样的测试是最低的要求了。   
2.判定覆盖。 每个判断的真假分支至少执行一次，就是真要至少取一次，假要至少取一次。这个设计起来也不难，覆盖率要比语句覆盖高近乎一倍，但是也在判定语句中也会遗漏许多路径，因为每个条件的取值是不在考虑范围内的。   
3.条件覆盖。 和判定覆盖思路一样，只是把重点从判定移动到条件上来了，每个判定中的每个条件可能至少满足一次，也就是每个条件至少要取一次真的，再取一次假的。同样它也会遗漏许多路径，条件取真假并不能满足判定也取到真假两次。   
4.判定条件覆盖。 既然上面的判定和条件多是片面的，那么这个两个覆盖相结合是呼之欲出判定条件覆盖。它要求判断中的每个条件所有可能至少出现一次,并且每个判定本身的判定结果也要出现一次。不要以为这样就行了，要看看条件，条件和判定不一样，判定取真假就覆盖了判定，可是条件取真假两次完全不能满足条件的各种组合。所以才有了5~。   
5.条件组合覆盖。 每个判定中条件的各种可能组合至少满足一次。条件各种可能都出现了，必然把判定给覆盖了，它覆盖了上面的4个哦，可是用例数量大大增加了！看项目情况定吧。   
6.路径覆盖。 概念比较好理解，把所有可能路径至少都走一遍。

**(2)黑盒测试：**又称为数据驱动测试，把测试对象当做看不见的黑盒，在完全不考虑程序内部结构和处理过程的情况下，测试者仅依据程序功能的需求规范考虑，确定测试用例和推断测试结果的正确性，它是站在使用软件或程序的角度，从输入数据与输出数据的对应关系出发进行的测试。

之所以被称为黑盒测试是因为可以将被测程序看成是一个无法打开的黑盒，而工作人员在不考虑任何程序内部结构和特性的条件下，根据需求规格说明书设计测试实例，并检查程序的功能是否能够按照规范说明准确无误的运行。其主要是对软件界面和软件功能进行测试。对于黑盒测试行为必须加以量化才能够有效的保证软件的质量

**(3)灰盒测试：**是一种综合测试法，它将“黑盒”测试与“白盒”测试结合在一起，是基于程序运行时的外部表现又结合内部逻辑结构来设计用例，执行程序并采集路径执行信息和外部用户接口结果的测试技术。

灰盒测试则介于黑盒测试和白盒测试之间。灰盒测试除了重视输出相对于出入的正确性，也看重其内部表现。但是它不可能像白盒测试那样详细和完整。它只是简单的靠一些象征性的现象或标志来判断其内部的运行情况，因此在内部结果出现错误，但输出结果正确的情况下可以采取灰盒测试方法。因为在此情况下灰盒比白盒高效，比黑盒适用性广的优势就凸显出来了。

从是否执行代码角度（程序执行状态）

静态测试、动态测试

**(1)静态测试：**指不运行被测程序本身，仅通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性。

**(2)动态测试：**是指通过运行被测程序，检查运行结果与预期结果的差异，并分析运行效率、正确性和健壮性等性能指标。

从软件开发的过程按阶段划分有

单元测试、集成测试、确认测试、系统测试、验收测试、回归测试

**(1)单元测试：**又称模块测试，是针对软件设计的最小单位----程序模块或功能模块，进行正确性检验的测试工作。其目的在于检验程序各模块是否存在各种差错，是否能正确地实现了其功能，满足其性能和接口要求。

单元测试主要是对该软件的模块进行测试，通过测试以发现该模块的实际功能出现不符合的情况和编码错误。由于该模块的规模不大，功能单一，结构较简单，且测试人员可通过阅读源程序清楚知道其逻辑结构，首先应通过静态测试方法，比如静态分析、代码审查等，对该模块的源程序进行分析，按照模块的程序设计的控制流程图，以满足软件覆盖率要求的逻辑测试要求。另外，也可采用黑盒测试方法提出一组基本的测试用例，再用白盒测试方法进行验证。若用黑盒测试方法所产生的测试用例满足不了软件的覆盖要求，可采用白盒法增补出新的测试用例，以满足所需的覆盖标准。其所需的覆盖标准应视模块的实际具体情况而定。对一些质量要求和可靠性要求较高的模块，一般要满足所需条件的组合覆盖或者路径覆盖标准。

**(2)集成测试：**又叫组装测试或联合，是单元测试的多级扩展，是在单元测试的基础上进行的一种有序测试。旨在检验软件单元之间的接口关系，以期望通过测试发现各软件单元接口之间存在的问题，最终把经过测试的单元组成符合设计要求的软件。

集成测试是软件测试的第二阶段，在这个阶段，通常要对已经严格按照程序设计要求和标准组装起来的模块同时进行测试，明确该程序结构组装的正确性，发现和接口有关的问题，比如模块接口的数据是否会在穿越接口时发生丢失；各个模块之间因某种疏忽而产生不利的影响；将模块各个子功能组合起来后产生的功能要求达不到预期的功能要求；一些在误差范围内且可接受的误差由于长时间的积累进而到达了不能接受的程度；数据库因单个模块发生错误造成自身出现错误等等。同时因集成测试是界于单元测试和系统测试之间的，所以，集成测试具有承上启下的作用。因此有关测试人员必须做好集成测试工作。在这一阶段，一般采用的是白盒和黑盒结合的方法进行测试，验证这一阶段设计的合理性以及需求功能的实现性。

**(3)确认测试：**又称有效性测试。任务是验证软件的功能和性能及其它特性是否与用户的要求一致。对软件的功能和性能要求在软件需求规格说明书中已经明确规定。它包含的信息就是软件确认测试的基础。

**(4)系统测试：**是为判断系统是否符合要求而对集成的软、硬件系统进行的测试活动、它是将已经集成好的软件系统，作为基于整个计算机系统的一个元素，与计算机硬件、外设、某些支持软件、人员、数据等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的组装测试和确认测试。  
一般情况下，系统测试采用黑盒法来进行测试的，以此来检查该系统是否符合软件需求。本阶段的主要测试内容包括[健壮性测试](/item/%E5%81%A5%E5%A3%AE%E6%80%A7%E6%B5%8B%E8%AF%95/4435189)、性能测试、功能测试、安装或反安装测试、用户界面测试、压力测试、可靠性及安全性测试等。为了有效保证这一阶段测试的客观性，必须由独立的测试小组来进行相关的系统测试。另外，系统测试过程较为复杂，由于在系统测试阶段不断变更需求造成功能的删除或增加，从而使程序不断出现相应的更改，而程序在更改后可能会出现新的问题，或者原本没有问题的功能由于更改导致出现问题。所以，测试人员必须进行回归测试。

**(5)验收测试：**以用户为主的测试，软件开发人员和质量保证人员参加，由用户设计测试用例。不是对系统进行全覆盖测试，而是对核心业务流程进行测试。

验收测试是最后一个阶段的测试操作，在软件产品投入正式运行前的所要进行的测试工作。和系统测试相比而言，验收测试与之的区别就只是测试人员不同，验收测试则是由用户来执行这一操作的。验收测试的主要目标是为向用户展示所开发出来的软件符合预定的要求和有关标准，并验证软件实际工作的有效性和可靠性，确保用户能用该软件顺利完成既定的任务和功能。通过了验收测试，该产品就可进行发布。但是，在实际交付给用户之后，开发人员是无法预测该软件用户在实际运用过程中是如何使用该程序的，所以从用户的角度出发，测试人员还应进行[Alpha测试](/item/Alpha%E6%B5%8B%E8%AF%95/7555188)或[Beta测试](/item/Beta%E6%B5%8B%E8%AF%95/7555227)这两种情形的测试。

Alpha测试是在软件开发环境下由用户进行的测试，或者模拟实际操作环境进而进行的测试。Alpha测试主要是对软件产品的功能、局域化、界面、可使用性以及性能等等方面进行评价。而Beta测试是在实际环境中由多个用户对其进行测试，并将在测试过程中发现的错误有效反馈给软件开发者。所以在测试过程中用户必须定期将所遇到的问题反馈给开发者。

**(6)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。