软件测试基础理论

2023年4月6日 9:46

经典定义：

a.软件测试是为了发现错误而执行程序的过程。

b.软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例(即输入数据及其预期的输出结果)，并利用这些测试用例去运行程序，以发现程序错误的过程。

黑盒测试：

把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。因此黑盒测试又叫功能测试或数据驱动测试。

黑盒测试方法包括：等价类划分、边界值分析、因果图分析、错误推测法、功能图分析等。

白盒测试：

是对软件的过程性细节做细致的检查。是把测试对象看做一个打开的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序状态，确定实际状态是否与预期的状态一致。因此白盒测试又称为结构测试或逻辑驱动测试。

白盒测试方法包括：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖等。

单元测试：

是对软件中的基本组成单位进行的测试，如一个模块、一个过程等等。它是软件动态测试的最基本的部分，也是最重要的部分之一，其目的是检验软件基本组成单位的正确性。一个软件单元的正确性是相对于该单元的规约(详细设计)而言的。因此，单元测试以被测试单位的规约为基准。

单元测试方法包括：控制流测试、数据流测试、排错测试、分域测试等。

集成测试：

是在软件系统集成过程中所进行的测试，其主要目的是检查软件单位之间的接口是否正确。它根据集成测试计划，一边将模块或其他软件单位组合成越来越大的系统，一边运行该系统，以分析所组成的系统是否正确，各组成部分是否合拍。集成测试的策略主要有自顶向下和自底向上两种。

功能测试：

对产品的各功能进行验证，根据功能测试用例，逐项测试，检查产品是否达到用户要求的功能。

性能测试：

是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。通过负载测试，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

负载测试：

定义:数据在超负荷环境下运行，测试软件系统是否能够承担。这种超负荷主要指多并发用户。

方法:人为生成大数据量，并利用工具模拟频繁并发访问

工具:一般需要使用自动化工具

考察指标:响应时间、交易容量、资源使用率等

压力测试：

定义:指系统不断施加越来越大的负载（并发，循环操作，多用户，网络流量）的测试。

目标:通过确定一个系统的瓶颈或者不能接收的性能点，来确定系统能提供的最大服务级别的测试。

恢复测试：

恢复测试主要检查系统的容错能力。当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。恢复测试首先要采用各种办法强迫系统失败，然后验证系统是否能尽快恢复。如果系统恢复是自动的（即恢复由系统自身完成），则应该检验以下内容：重新初始化、检验点设置机构、数据恢复以及重新启动是否正确。

兼容性测试：

定义:测试软件在一个特定的硬件、软件、操作系统、网络等环境下系统能否正常运行。

目的:检验被测软件对其他应用软件或者其他系统的兼容性。

安全性测试：

定义:安全测试检测系统对非法入侵的防范能力。

1.应用程序级别的安全性测试

2.数据库安全性测试

3.系统级别的安全性测试

UI测试：

指测试用户界面的风格是否满足客户要求，文字是否正确，页面美工是否好看，文字，图片组合是否完美，背景是否美观，操作是否友好等；用户界面(UI)测试用于核实用户与软件之间的交互。UI测试的目标是确保用户界面会通过测试对象的功能来为用户提供相应的访问或浏览功能。另外，UI测试还可确保UI中的对象按照预期的方式运行，并符合公司或行业的标准。包括用户友好性，人性化，易操作性测试。UI测试比较主观，与测试人员的喜好有关。

自动化测试：

利用软件测试工具自动实现全部或部分测试，它是软件测试的一个重要组成部分，能完成许多手工测试无法实现或难以实现的测试；正确、合理的实施自动测试，能够快速、全面的对软件进行测试，从而提高软件质量，节省经费，缩短软件发布周期。

先进行UI测试、功能测试、正常功能测试，异常功能测试、安全性测试、性能测试、负载测试、压力测试、算法测试
用例名称、所属模块、用例等级、测试步骤、预期结果、标签、关联的需求、前置条件、后置条件

测试方法分类：

黑盒测试：等价类划分 边界值分析 因果图分析 错误测试
白盒测试：语句覆盖 判定覆盖 条件覆盖 判定/条件覆盖 多重条件覆盖

等价类划分方法

等价类划分：指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的，并合理地假定；测试某等价类的代表值就等于对这一类其它值的测试。因此，可以把全部输入数据合理划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件，就可以用少量代表性的测试数据，取得较好的测试结果。
等价类划分可有两种不同的情况：有效等价类和无效等价类。

边界值分析方法

边界值：是对等价类划分方法的补充，测试工作经验告诉我,大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

错误推测方法

错误推测：基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性的设计测试用例的方法。
错误推测方法的基本思想：列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据他们选择测试用例。例如：在单元测试时曾列出的许多在模块中常见的错误。以前产品测试中

曾经发现的错误等，这些就是经验的总结。还有，输入数据和输出数据为0的情况。输入表格为空格或输入表格只有一行。这些都是容易发生错误的情况。可选择这些情况下的例子作为测试用例。

因果图方法

前面介绍的等价类划分方法和边界值分析方法，都是着重考虑输入条件,但未考虑输入条件之间的联系，相互组合等。考虑输入条件之间的相互组合，可能会产生一些新的情况。但要检查输入条件的组合不是一件容易的事情,即使把所有输入条件划分成等价类，他们之间的组合情况也相当多。因此必须考虑采用一种适合于描述对于多种条件的组合，相应产生多个动作的形式来考虑设计测试用例。这就需要利用因果图（逻辑模型）。因果图方法最终生成的就是判定表。它适合于检查程序输入条件的各种组合情况。

判定表驱动分析方法

判定表：是分析和表达多逻辑条件下执行不同操作的情况的工具。

正交表设计分析方法

有时候，可能因为大量的参数的组合而引起测试用例数量上的激增，同时，这些测试用例并没有明显的优先级上的差距，而测试人员又无法完成这么多数量的测试，就可以通过正交表来进行缩减一些用例，从而达到尽量少的用例覆盖尽量大的范围的可能性。

功能图分析方法

功能图：由状态迁移图和布尔函数组成，状态迁移图用状态和迁移来描述。一个状态指出数据输入的位置（或时间），而迁移则指明状态的改变。同时要依靠判定表或因果图表示的逻辑功能。

场景模拟分析方法

指根据用户场景来模拟用户的操作步骤,这个比较类似因果图,但是可能执行的深度和可行性更好。

正常操作场景：

测试电梯能否正常运行，并按需求停靠在指定楼层。
测试电梯能否正确响应用户按下楼层按钮的请求，并按照请求顺序运行。
测试电梯运行过程中是否能按需求关闭或打开门。
测试电梯运行过程中是否能正确显示当前楼层。
异常场景：

测试电梯在运行时，突然断电后再次启动是否能够继续运行。
测试电梯在运行中，遇到故障时是否能够正确处理故障情况。
测试电梯超载时是否能正确报警并停止运行。
测试电梯在运行中是否能正确处理误操作，如按下错误楼层按钮或重复按下同一楼层按钮等情况。
性能和稳定性：

测试电梯在高负载情况下是否仍然能正常运行和响应用户请求。
测试电梯在连续长时间运行过程中是否会出现性能问题，如卡顿、延迟等情况。

其他功能：

测试电梯在非工作时间段是否能正确进入节能模式。
测试电梯在紧急情况下是否能正确响应紧急按钮，如火警或停电。