# 软件测试入门指南:周期、模型和文档化

　　世界上没有任何软件能够保证是完美无缺的。但是这不应当成为软件缺陷的托词。为了提高产品的质量，确保软件应用的有效性、以及应用的平稳运行，我们需要进行各种有计划有步骤的[**软件测试**](javascript:;)。在本文中，我们将通过向您介绍有关软件测试的基本方面，以帮助您把控软件质量，并能交付出满意的产品。

**软件测试简介**

　　从概念上说，软件测试是一个评估已开发软件的功能，审查应用是否满足既定的要求，识别程序中任何潜在的缺陷，进而提高产品质量的过程。它的输出是发现目标系统与实际需求之间的差距，以及自身存在的错误与缺失。

[](http://www.51testing.com/batch.download.php?aid=99716)

　　在业界，我们通常将软件测试称为验证和确证软件产品的过程。因此，它关注软件产品的如下三个方面：

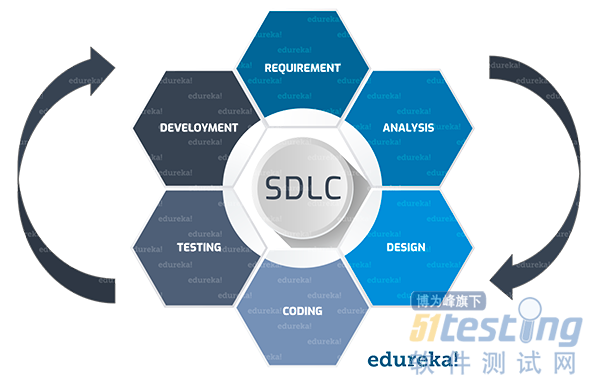
　　是否满足那些指导其设计和开发的业务和[**技术**](javascript:;)需求

　　是否能够按需运行并提供稳定的服务

　　既定的服务功能是否能够重复实现

[**软件开发**](javascript:;)**生命周期**

　　首先，我们来了解一下软件开发生命周期(Software Development Life Cycle，SDLC)的概念。它是整个软件行业用于设计、开发和测试高质量软件的过程。SDLC旨在规定的时间和成本预算之内，生产出那些满足甚至超出客户期望的高质量软件。下图概括性地描绘了SDLC所涉及的各个阶段。

[](http://www.51testing.com/batch.download.php?aid=99717)

**要求阶段**

　　需求的收集和分析是软件开发生命周期中最重要的阶段。业务分析师们会从客户和用户处收集他们的业务需求，并在既有的业务需求规范(具体文档名称会因组织的不同而有所差异)中[**记录**](javascript:;)这些需求要点。

**分析阶段**

　　在完成对需求的收集和分析后，下一步就是要定义并记录产品的规范需求，进而获得客户的确认。我们一般需要通过软件需求规范(Software Requirement Specification，SRS)文档来实现。SRS一般包含了在项目生命周期中，那些与设计和开发相关的所有产品需求。

**设计阶段**

　　此阶段分为两个步骤：

　　高级设计(High-Level Design，HLD)：有时也称概要设计，它交付的是待开发软件产品的体系结构，一般由架构师和高级资深开发人员来完成。

　　低级设计(Low-Level Design，LLD)：有时也称详细设计，它描述的是产品的每一项功能、以及每一个组件应该如何运作。一般由高级资深开发人员来完成。

　　可见，此阶段的输出：高级文档和低级文档，将作为下一阶段的输入。

**开发阶段**

　　所有级别的开发人员(从资深到初级)都参与到该阶段当中，着手编写并构建软件的相关代码。

**测试阶段**

　　软件在完成编码之后，会被发送到测试部门，对可能出现的缺陷进行全面的测试。他们既可以手动测试软件，又可以通过[**自动化测试**](javascript:;)工具，以确保软件的每个组件都能够正常工作。软件只有通过了测试，才能得到质量保证(QA)，也才能进入下面的实施阶段。

**部署和维护阶段**

　　最终，软件产品需要交付和部署，以供客户使用。此阶段通常是由部署与实施工程师来完成。而在客户使用系统与服务的过程中，他们所碰到的任何实际问题，都需要在维护阶段，得到持续并及时地解决。

　　如果您想深入了解上述软件开发生命周期中有关测试环境的具体内容，请参阅软件测试生命周期一文(https://www.edureka.co/blog/software-testing-life-cycle/)。下面我们来讨论一下所谓的V模型。

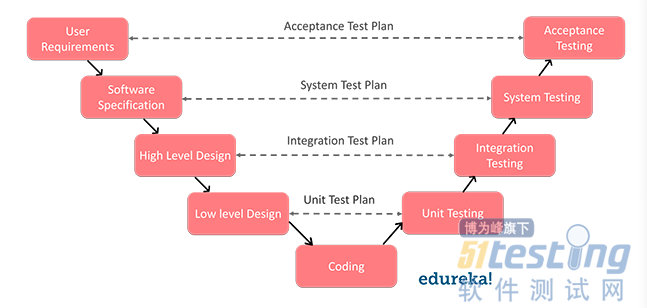
**验证和确证模型**

　　V模型是目前使用最为广泛的软件开发过程之一。实际上，V模型将测试环节贯穿到了从需求阶段到开始实施的整个过程之中。在业界，V模型也称为验证和确证模型(verification and validation model)。那么到底什么是软件测试中的验证和确证呢?

　　验证：是一种静态分析的技术。它是在不执行代码的情况下，执行诸如：回顾、检查、以及逐行审验之类的测试。

　　确证：是一种动态分析的技术，它是对代码执行功能性和非功能性的测试。

　　在V模型中，测试并不是作为一个单独离散的阶段，而是跟随着需求阶段一起开始，与软件开发的整个进程同步推进，分不同的步骤对产品开展验证和确证活动。该模型的阶段名称如下图所示：

[](http://www.51testing.com/batch.download.php?aid=99718)

　　由上图可见，在V模型的左侧是一系列开发活动，其中包括：用户需求、软件定义、概要设计、详细设计、以及软件编码;而右侧则是一系列对应的测试活动，其中包括：[**单元测试**](javascript:;)、集成测试、系统测试、以及验收测试。因此，左右不同阶段的联系可以简述为：单元测试着眼于开发出的代码是否符合详细设计的要求;集成测试检查的是各组成组件能否协同工作;系统测试关注的是集成到一起的产品是否符合规格说明的要求;而验收测试则注重的是产品能否让最终用户满意。

[**软件测试方法**](javascript:;)

　　通常而言，我们会采取如下三种方法开展软件的相关测试，它们是：

[**黑盒测试**](javascript:;)：测试人员并不知晓被测项目的内部结构、设计、以及实现原理。

[**白盒测试**](javascript:;)：测试人员完全知晓被测项目的内部结构、设计、以及实现原理。

　　灰盒测试：测试人员仅仅知晓被测项目有限的内部功能信息。

**软件测试阶段**

　　我们运用软件测试的不同阶段，来对目标软件系统的每个单元或组件进行审验。由于系统测试的主要目标就是评估其是否符合既定的开发需求。因此，这些不同的测试阶段不但有助于检查软件的行为和性能，也能够识别出其中的缺陷(bug)，以及在整个开发的生命周期中起到一定的协调作用。具体说来，软件测试分为如下四个阶段：

　　单元测试：通过设定目标软件的最小测试单位，尽快地找出各个模块或组件中的明显错误，以提高单块程序代码的质量、并减少后期返工的成本。

　　集成测试：通过测试整个系统能否编译和构建成功，以发现系统架构和模块之间、模块与模块之间是否存在接口问题，并记录下测试结果。

　　系统测试：通过运行整个系统，来根据系统[**测试用例**](javascript:;)执行全面测试，验证并确证系统的功能与性能是否符合需求规格说明中的要求。

　　验收测试：在系统安装部署完成之后，通过邀请客户参与进来，进而确认软件系统能否按照既定的要求平稳运行。

**软件测试文档化**

　　将各项测试用例进行文档化，将有助于我们评估测试的工作量，以及跟踪测试的覆盖率。那么，常用的软件测试相关文档包括如下四种：

　　1.测试计划：为目标应用程序提供测试大纲与策略。其中包括：测试的范围、方法、资源和进度等。具体还会涉及到各阶段的测试任务、时间进度安排、测试执行团队、以及风险揭示等。

　　2.测试场景：描述目标的特性、测试的方法、环境的要求、工具的选择、以及测试的范围。具体还会涉及到各阶段的启动、停止、完成标准等条件。

　　3.测试用例：它是由一组条件或变量所组成，测试人员籍此确定被测系统是否满足要求，以及能否正常工作。开发测试用例的过程，也有助于发现应用程序在代码与设计中的问题。因此，在具体测试期间，我们可以设计出许多类型的测试用例。例如：

　　功能性测试用例

　　负面错误测试用例

　　逻辑测试用例

　　物理测试用例

　　用户[**接口测试**](javascript:;)用例

　　4. 追溯矩阵：也称为需求追踪矩阵(Requirement Traceability Matrix，RTM)，它是一张被用于在创建产品的SDLC模型时，草拟各种需求的表格。在实际应用中，我们既可以采取从设计到编码的前向跟踪方式，也可以采用相反的向后跟踪方式。

[**缺陷管理**](javascript:;)**流程**

　　众所周知，软件开发是一个非常复杂的过程。它往往要求团队成员在严格的时间期限内，每天都必须完成大量的代码编写工作，因此他们通常没有太多的时间去考虑如何避免出现错误。因此，对于每个软件开发项目来说，我们都需要有一个能够检测错误与缺陷，进而及时修复的缺陷管理过程。

　　缺陷管理，或称错误跟踪，通常是在产品的测试阶段进行的。我们可以通过两种不同的方式实现缺陷管理，即：开发人员自行测试他们的产品;或邀请用户参与测试。虽然最终用户能够提供不同的视角来识别缺陷，但是他们的识别途径往往不够全面。

　　因此，缺陷管理通常会涉及到如下四个步骤：

　　检测与识别缺陷

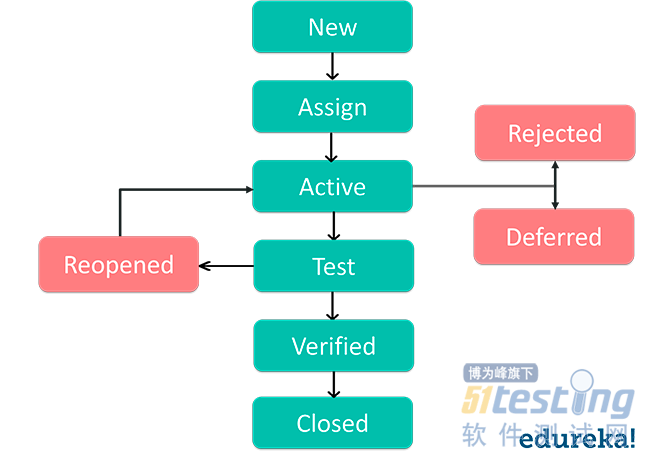
　　分析与定位缺陷，进而填写并提交缺陷报告

　　提请整改并修复缺陷

　　验证修改并记录缺陷列表

**缺陷生命周期**

　　一般而言，缺陷的生命周期是从bug被发现开始，直至它被最终修复，并能确保不再复发为止。不过有时候，根据组织的策略，使用到的软件开发模型(如敏捷或迭代)，项目的时间表，以及团队的结构等因素的不同，缺陷的生命周期也可能有所差异。总的说来，它所要经历的步骤与过程，如下图所示：

[](http://www.51testing.com/batch.download.php?aid=99719)

　　新建：当缺陷被首次递交时，它处于“新建”状态。不过，该缺陷可能需要得到进一步的确认。

　　分配：经由测试人员创建之后，其主管负责人会审核该缺陷的真实性，一旦确认，则会分配给相应的开发人员或团队接手处理。

　　打开：开发人员受理该缺陷，着手分析并研发缺陷的修复程序。

　　修复：当开发人员完成了必要的代码更改，并验证了其修复效果之后，他会将缺陷状态设置为“已修复”，并将缺陷的相关信息回传给测试团队。

　　测试：在此阶段，测试人员对开发人员传回的已更改代码进行测试，以检验缺陷是否已被修复。

　　验证：测试人员针对该缺陷展开新一轮的测试。如测试通过，他会将该缺陷的状态变更为“已验证”。

　　重新打开：相反，如果测试未能通过，则测试人员将状态更改为“重新打开”。该缺陷将再次经历上述周期。

　　关闭：如果测试人员最终认为目标软件中不再存在此类缺陷，他会将错误状态更改为“已关闭”。

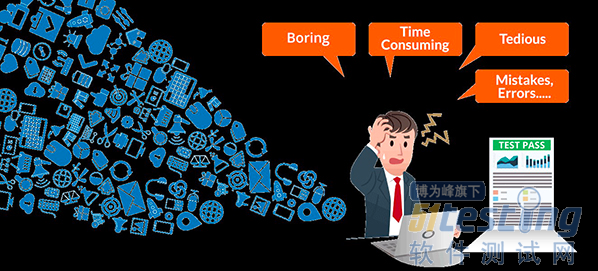
　　重复：如果同一个缺陷被重复提交、或者两个缺陷属于同一种错误，那么这个缺陷状态会被设置为“重复提交”。

　　拒绝：如果开发人员不认为这是一个缺陷，他会拒绝，然后将错误的状态更改为“已拒绝”。

　　延期：如果缺陷的状态被置为“延期”，则意味着该缺陷将会在下一个版本中被修复。当然，将缺陷置为“延期”原因有许多，例如：该缺陷的优先级不高，或是时间紧迫，又或是对于软件不会造成太大的影响。

**手动测试与自动化测试**

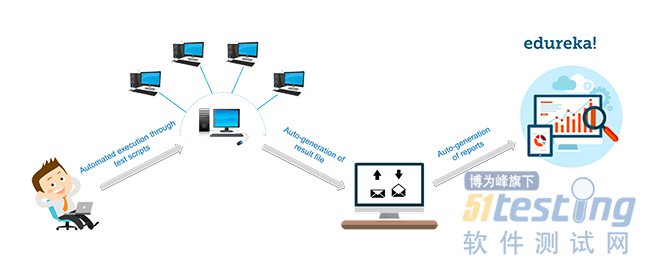
　　如果选择手动测试，则意味着我们需要QA测试人员，对目标软件的每一个测试用例，都采取手动执行的方式，人工提供不同的数据集，并且仔细地记录下每一个环节的成功与失败率。

[](http://www.51testing.com/batch.download.php?aid=99720)

　　由于需要人工进行测试结果的验证和输出的判断，因此不但费时费力，而且难免会产生人为的疏忽。

　　不言而喻，自动化测试意味着更高的效率，更少的人力，更低的出错几率，以及适用于回归和功能性测试环节的、更频繁的执行次数。

　　例如：对于某个登录页面而言，如果我们需要验证所有登录尝试的可能性及其结果，那么我们只需要通过编写一段测试代码，就能自行输入所有可能的数据，执行各种类型的登录方法，进而判断并记录所有登录尝试的成败结果。

[](http://www.51testing.com/batch.download.php?aid=99715)

　　此外，您也可以在不同系统环境，以及不同的[**浏览器**](javascript:;)中进行模拟测试。而且，您还可以指定在一天中的某个特定时间段，自动生成结果文件及报告。如今，市面上已有许多自动化的测试工具，例如：[**Selenium**](javascript:;)等。

　　以上就是我想向您介绍的有关软件测试的一些入门知识，希望能够帮助您建立起相关的知识概念，并能指导您后继的学习以及测试工作。