软件测试是软件质量保证的关键步骤。越早发现软件中存在的问题，修复问题的成本就越低，软件质量也就越高，软件发布后的维护费用越低。

为了能更好的保障软件质量，在软件测试的实践中，慢慢形成了一些流程用来达到这一目标。下面就来介绍一下常见的测试流程。

传统测试流程

在传统的测试流程中包含了如图所示的步骤。

下面分别介绍下每一步流程的含义。

单元测试

单元测试是对软件中的基本组成单位进行的测试。目的是检验软件基本组成单位的正确性。

- 测试阶段：编码后
- 测试对象：最小模块
- 测试人员：开发
- 测试依据：代码、注释、详细设计文档
- 测试方法：白盒测试

集成测试

集成测试是在软件系统集成过程中所进行的测试。目的是检查软件模块之间的接口是否正确。

- 测试阶段：单元测试完成后
- 测试对象：模块间的接口
- 测试人员：开发
- 测试依据：单元测试模块、概要设计文档

- 测试方法：黑盒与白盒结合

冒烟测试

冒烟测试是在软件开发过程中的一种针对软件版本包的快速基本功能验证策略，是对软件基本功能进行确认验证的手段。

- 测试阶段：提测后
- 测试对象：整个系统
- 测试人员：测试
- 测试依据：冒烟测试用例
- 测试方法：黑盒测试（手工或自动化手段）

系统测试

系统测试是对已经集成好的软件系统进行彻底的测试，以验证软件系统的正确性和性能等是否满足其规约所指定的要求。

- 测试阶段：冒烟测试通过后
- 测试对象：整个系统
- 测试人员：测试
- 测试依据：需求文档、测试方案、测试用例
- 测试方法：黑盒测试

一般系统的主要测试工作都集中系统测试阶段。根据不同的系统，所进行的测试种类也很多。
在系统测试中，又包括如下测试种类：

- 功能测试：功能测试是对产品的各功能进行验证，以检查是否满足需求的要求。
- 性能测试：性能测试是通过自动化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。
- 安全测试：安全测试检查系统对非法入侵的防范能力。
- 兼容测试：兼容性测试主要是测试系统在不同的软硬件环境下是否能够正常的运行。

验收测试

验收测试是部署软件之前的最后一个测试操作。验收测试的目的是确保软件准备就绪，向软件购买都展示该软件系统满足其用户的需求。

- 测试阶段：发布前
- 测试对象：整个系统

- 测试人员：用户/需求方
- 测试依据：需求、验收标准
- 测试方法：黑盒测试

软件测试模型

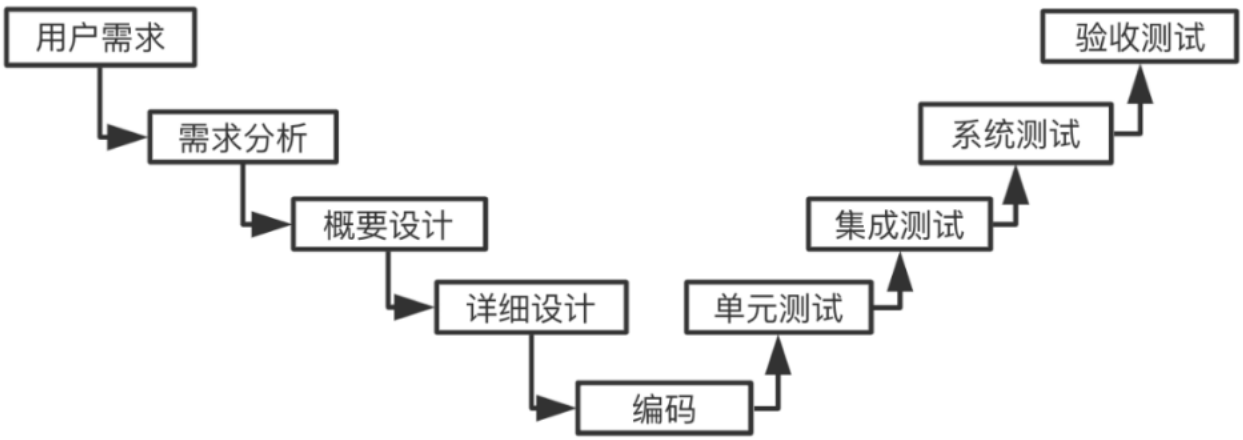
软件测试过程是一种抽象的模型，用于定义软件测试的流程和方法。众所周知，开发过程的质量决定了软件的质量，同样的，测试过程的质量将直接影响测试结果的准确性和有效性。软件测试过程和软件开发过程一样，都遵循软件工程原理，遵循管理学原理。
随着测试过程管理的发展，软件测试专家通过实践总结出了很多很好的测试过程模型。这些模型将测试活动进行了抽象，并与开发活动有机的进行了结合，是测试过程管理的重要参考依据。

下面介绍几种常见的测试模型。

V 模型

V 模型是开发模型中瀑布模型的一种改进。瀑布模型将软件生命周期划分为计划、分析、设计、编码、测试和维护六个阶段，由于早期的错误可能要等到开发后期的测试阶段才能发现，所以可能带来严重的后果。

V 模型在这点改进了瀑布模型，在软件开发的生存期，开发活动和测试活动几乎同时开始，在开发活动进行的时候，测试活动开始进行相应的文档准备工作，从而改进软件开发的效率和效果。



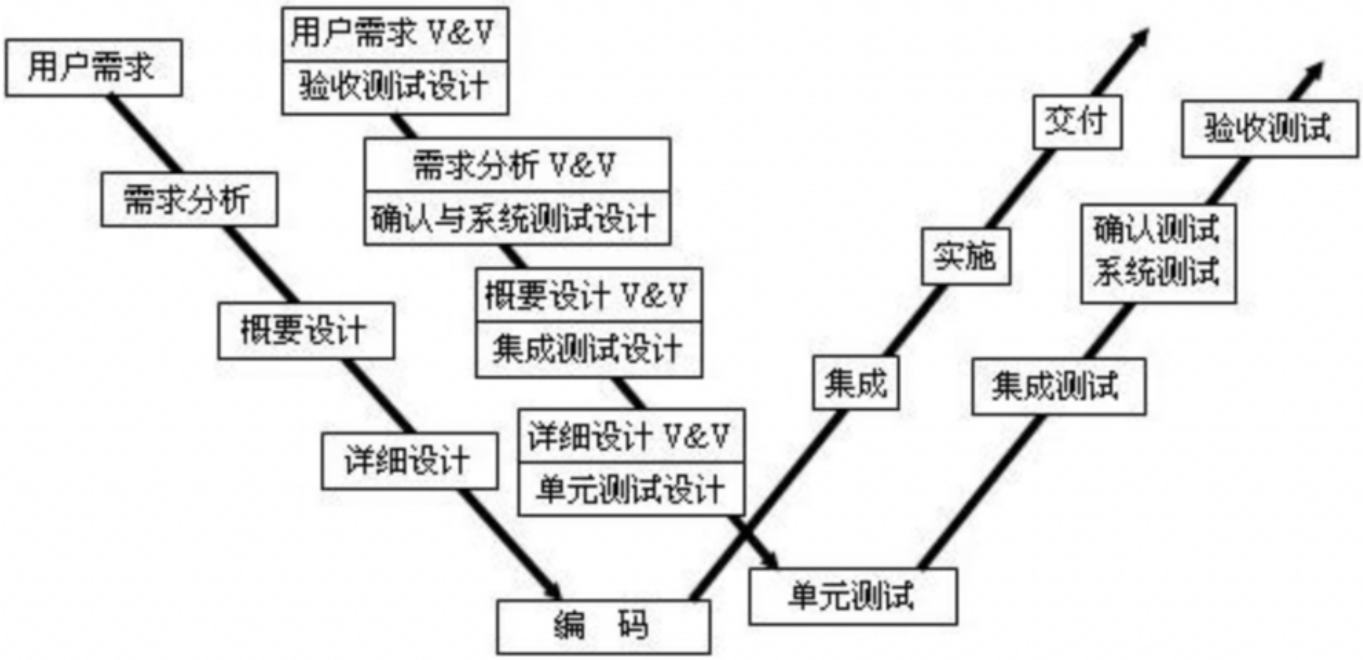
V 模型的优点是明确的标注了测试过程中存在着那些不同的测试类型，并且可以清楚的表达测试阶段和开发过程各阶段的对应关系。

但是，它也有一些缺点。比如容易让人误解为测试是在开发完成之后的一个阶段。而且由于它的顺序性，当编码完成之后，正式进入测试时，这时发现的一些 Bug 可能不容易找到其根源，并且代码修改起来很困难。在实际工作中，因为需求变更较大，使用 V 模型可能导致要重复变更需求、设计、编码、测试，返工量会比较大。

W 模型

W 模型从 V 模型演化过来。相对于 V 模型，W 模型增加了软件各开发阶段中应同步进行的验证和确认活动。

W 模型由两个 V 字型模型组成，分别代表测试与开发过程，图中明确表示出了测试与开发的并行关系。测试与开发是同步进行的，有利于尽早的全面发现问题。



W 模型认为测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、设计等同样要测试。

W 模型有利于尽早地全面的发现问题。例如，需求分析完成后，测试人员就应该参与到对需求的验证和确认活动中，以尽早地找出缺陷所在。

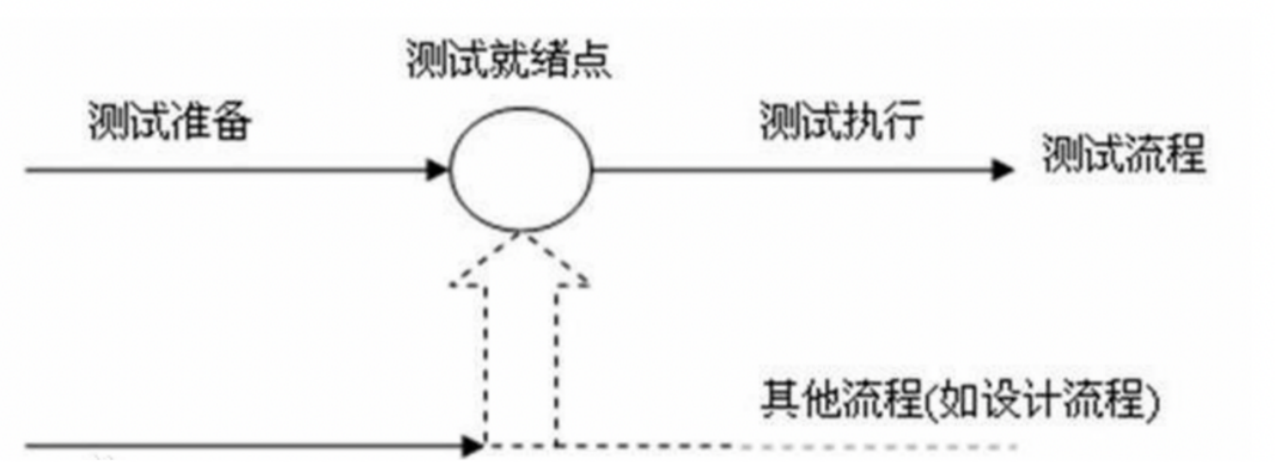
对需求的测试也有利于及时了解项目难度和测试风险，及早制定应对措施，这将显著减少总体测试时间，加快项目进度。

使用 W 模型的优点很明显。首先测试的活动与软件开发同步进行，而且测试的对象不仅仅是程序，还包括需求和设计。这样可以尽早发现软件缺陷可降低软件开发的成本。

但是 W 模型还是有一些缺点存在。比如开发和测试依然是线性的关系，需求的变更和调整，依然不方便。而且如果没有文档，根本无法执行 W 模型。使用 W 模型对于项目组成员的技术要求也更高。

H 模型

相对于 V 模型和 W 模型，H 模型将测试活动完全独立出来，形成了一个完全独立的流程，将测试准备活动和测试执行活动清晰地体现出来。



这个示意图仅仅演示了在整个生产周期中某个层次上的一次测试“微循环”。图中标注的其他流程可以是任意的开发流程，例如，设计流程或编码流程。也就是说，只要测试条件成熟了，测试准备活动完成了，测试执行活动就可以（或者说需要）进行了。

H 模型中包含了如下概念：

- 测试准备：所有测试活动的准备判断是否到测试就绪点。
- 测试就绪点：测试准入准则，即是否可以开始执行测试的条件。
- 测试执行：具体的执行测试的程序。
- 其它流程：设计流程或编码流程。

H 模型揭示了软件测试除测试执行外，还有很多工作。它让测试活动完全独立贯穿整个生命周期与其它流程并发进行。在 H 模型中，软件测试活动可以尽早准备尽早执行，具有很强的灵活性。而且软件测试可以根据被测对象的不同而分层次、分阶段、分次序的执行，同时也是可以被迭代的。

但是 H 模型对于管理要求很高，因为需要定义清晰的规则和管理制度，否则测试过程将很难管理和控制。而且对于技能要求也很高，因为 H 模型要求能够很好的定义每个迭代的规模，不能太大也不能太小。在 H 模型中，测试就绪点的分析也比较困难。因为测试过程中，并不知道测试准备到什么时候是合适的，就绪点在哪，就绪点标准是什么，这就对后续的测试执行启动带来很大的困难。

三种模型对比

上面介绍的三种测试模型是现在比较常见的，但是它们的使用场景会有一些不同。

- V 模型适用于中小企业。
- W 模型适用于中大型企业。
- H 模型人员要求非常高，现阶段使用比较少。

系统测试工作流程

对于系统测试来说，工作流程如图所示：

下面分别解释一下每一步的具体含义。

项目计划

描述测试目的、范围、方法和软件测试的重点等等内容的文档。

需求分析

测试工程师参与需求分析，可以增加对需求的了解，减少后期与产品和开发人员的沟通，节省时间。早期确定测试用例的编写思路，可以为测试打好基础。在需求分析的过程中可以获取一些测试数据，为测试用例设计提供帮助。而且在分析过程中可以发现需求不合理的地方，降低测试成本。

测试设计

测试设计是指把概括的测试目标转化为具体的测试用例的一系列活动。设计时要同时评审测试依据，也就是需求、系统架构、设计和接口说明等文档。通过对测试项、规格说明、测试对象行为和结构的分析，识别测试条件并确定优先级。根据分析的内容设计测试用例，并确定优先级，同时确定测试条件和测试用例所需的必要的测试数据。

用例评审

测试用例评审一般进行两轮。

一轮是组内评审，组内人员会评审测试用例是否完全覆盖了需求，提出一些修改意见。

二轮评审需要和产品、研发一起进行，产品和研发会从不同角度对用例进行一些补充。经过用例评审并且把评审中的建议补充完毕之后，测试用例才最终设计完毕，进入等待执行的状态。

测试执行

开发人员完成需求的开发之后会提测，也就是把可以测试产品交付给测试人员进行测试。提测后需要先执行冒烟测试，通过之后正式进入测试执行阶段。

开始执行之前要确认已经正确搭建了测试环境。环境没有问题后，就根据计划的执行顺序，通过手工或使用测试工具来执行测试用例。执行过程中需要记录测试执行的结果，以及被测软件、测试工具的标识和版本。将实际结果和预期结果进行比较。对实际结果和预期结果之间的差异，作为 Bug 上报，并且进行分析以确定引起差异的原因。缺陷修正后，重新进行测试活动。

Bug 管理

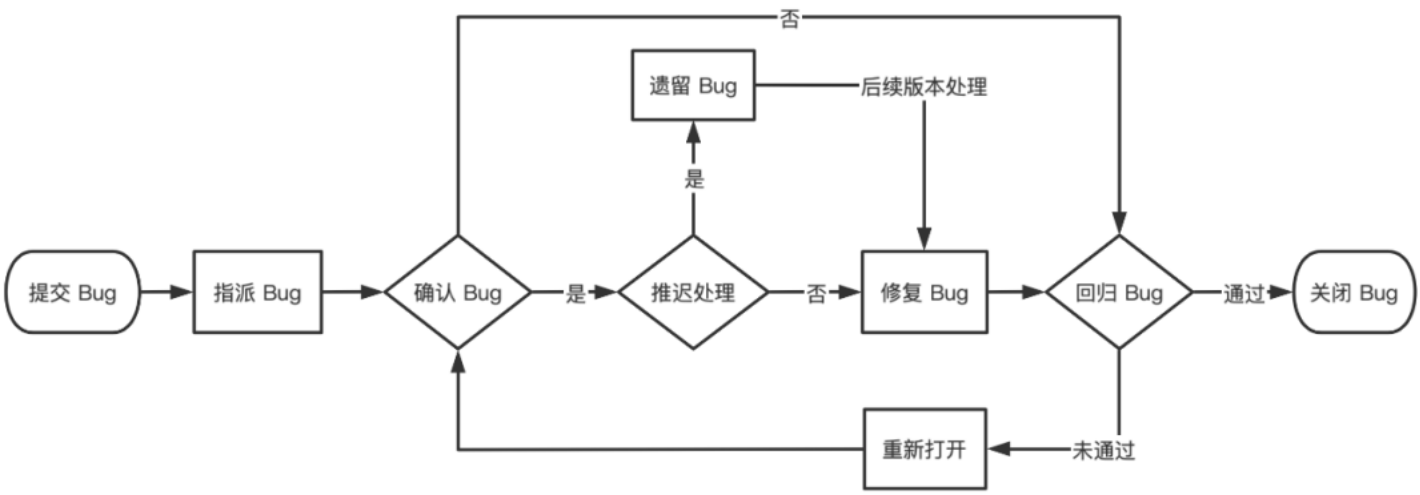
软件缺陷（Bug）是一种泛称，它可以指功能的错误，也可以指性能低下，易用性差等。

发布维护

监控上线后的产品，发现问题及时修复。

Bug管理流程

Bug 的管理也需要遵循一定的流程，基本流程如图所示：



提交 Bug

在提交一个 Bug 的时候，首先尽量描述这个 Bug 的属性。重现环境，类型，等级，优先级以及详细的重现步骤，结果与期望等。当然，在提交一个问题之前首先应该保证，这个 Bug 是没有被提过的，以免造成重复提交。

指派 Bug

有些公司测试部门与开发部门独立，那么测试人员就不确定自己测试的模块是由哪位开发人员负责的，在这种情况下，测试人员统一把问题指派给项目组长或经理，由项目组长（或经理）对问题进行确认后再次分配给相应的开发人员。

有些测试人员是和研发团队在一起工作的，这时，测试人员会对开人发员负责的模块非常清楚，就可以将问题直接指派给相应的开发人员。

确认 Bug

当开发人员接到一个 Bug 时，首先是对其进行分析与重现，如果对其进行分析发现不是 Bug（可能由于测试人员不了解需求）或无法对此问题进行重现，那么就需要将此问题返回给测试人员再次进行回归，并注明原因。如果确认为是 Bug 则需要对其进行处理。

判断是否推迟处理

在处理问题之后，还需要进行一次判断，是否需要推迟处理。有些 Bug 已经确认了是问题，由于其可能在极端情况下才会出现，或需要对系统架构进行改动，或其优先级非常低，所以暂时不需要对此问题进行处理，或到下个版本进再进行修复。

遗留 Bug

对于推迟处理的问题可以暂时进行遗留。一般遗留的问题需要经过项目经理与测试经理协商后才可以。

处理 Bug

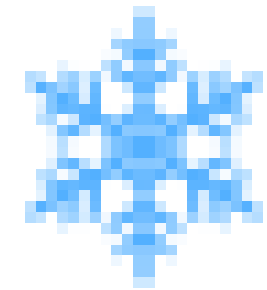
开发人员在确认完一个问题需要处理时，那么就对其进行处理工作。

回归 Bug

确认非 Bug 问题：对于提交的一个 Bug，开人员处理为非 Bug 或无法重现，然后直接转交给测试人员回归。测试人员再次确认，如果真如开发人员所说，则将问题关闭。如果非开发人员所说，是由于问题描述模糊或其它原因未重现问题，则再次注明原因转给开发人员。

确认修复问题：对开发人员修复的问题再次进行确认，确认能过，则关闭问题。确认不通过，将问题再次打开并转给开发人员。

确认遗留问题：有计划的对遗留的问题进行确认，有些遗留问题随着时间的推移，版本的更新或已经不存在了，对这类问题应该及时关闭。有些遗留问题依然存在且变得紧急，对于这类问题应该及时打开交给开发人员处理。



关闭 Bug

对于已经修复的 Bug 进行关闭，这也是一个 Bug 的最后一个状态。

测试左移和测试右移

传统的流程就是接到项目后参与需求评审，然后根据需求文写用例和准备脚本，等开发提测之后正式开始测试、提 Bug、回归，测试通过后就结束了，项目交给运维上线，之后投入下一个项目继续重复这样的流程。

这样的流程看似没什么问题，但缺点是测试过程是在一定时间间隔内发生的，测试人员必须等待产品完全构建才能找到错误和故障。有时候等待产品花费的时间超过了可以商定的时间，等待代码成为测试人员的瓶颈。

而测试左移以及测试右移，能够让测试拥有更多的主动权，有更充足的时间进行测试，同时不会像之前因为质量差风险高每次都延期上线，并且产品的线上质量也能有保证。

不管是测试左移还是测试右移，都是为产品质量服务。不要把提测认为是测试活动的开始，上线是测试活动的结束，更不要认为质量只是测试同学需要关注的。

测试左移

测试左移是向测试之前的开发阶段移动。

测试左移的原则支持测试团队在软件开发周期早期和所有干系人合作。因此他们能清晰地理解需求以及设计测试用例去帮助软件“快速失败”，促使团队更早的修改所有的 Bug。

参与和理解会使测试人员获取产品完整的知识，彻底想清楚各种场景，根据软件行为设计实时的场景，这些都会帮助团队在编码完成之前识别出一些缺陷。

左移聚焦在使测试人员在全部和最重要的项目阶段参与进来。这就是测试人员把焦点从发现 Bug 转移到 Bug 的预防上，同时也驱动项目的商业目标。

随着测试团队的责任的提高，团队不在仅仅聚焦在“测试软件去发现 Bug”，而是积极团队合作，参与项目初始阶段的计划和建立强壮有效的测试策略，而测试策略又为团队提供好的测试领导力和指导，使团队聚焦在产品的长远的视角，而不仅仅是测试工作。

左移首先为测试人员提供了设计测试的机会，无论这些测试是被聚焦在客户的体验还是期望，也促使开发人员根据这些测试去开发软件以满足客户需求。

测试右移

测试右移是测试活动向产品发布之后的步骤移动。

是产品上线了之后也可以进行一些测试活动。可以在生产环境做监控，监控线上性能和可用率，一旦线上发生任何问题，尽快反映，提前反映，给用户良好的体验。