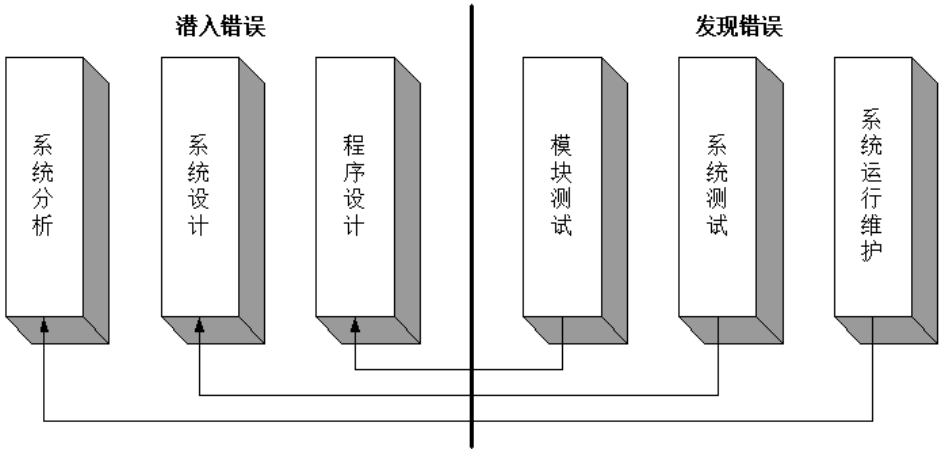
# 软件测试的基本知识

1、软件测试的广义论与狭义论

狭义概念：传统瀑布模型认为测试是指在代码完成后、处于运行维护阶段之前，通过运行程序来发现程序代码或软件系统中错误。

因此，不能在代码完成之前发现软件系统需求及设计上的缺陷，如果需求和设计上的缺陷问题需要留到后期，就会造成大量返工，增加软件开发的成本，延长开发的周期等。

错误的“堆栈”现象：错误纠正的越早，所造成的损失就越少。




广义概念：为了更早地发现问题，将测试延伸到需求评审、设计审查活动中去，将“软件质量保证”的部分活动归为测试活动。软件生命周期每一阶段中都应包含测试，用于检验本阶段的成果是否接近预期的目标，尽可能早的发现错误并加以修正，将软件测试和质量保证合并起来的软件测试，被认为是一种软件测试的广义概念。

2、软件测试的辩证论

验证软件是验证软件是“工作的”，是指软件的功能是按照预先的设计执行的，以正向思维，针对软件系统的所有功能点，逐个验证其正确性。其代表人物是软件测试领域的先驱 Dr.Bill Hetzel（《The Complete Guide to Software Testing》）。

软件是“不工作的”，以反向思维方式，不断思考开发人员理解的误区、不良的习惯、程序代码的边界、无效数据的输入以及系统的弱点，试图破坏系统、摧毁系统，目标就是发现系统中各种各样的问题。其代表人物是 Glenford J. Myers。（《The Art of Software Testing》）强调一个成功的测试必须是发现 Bug 的测试，不然就没有价值，认为软件测试：“就是以发现错误为目的而运行程序的过程。

3、软件测试的风险论

软件测试的风险论认为测试是对软件系统中潜在的各种风险进行评估的活动。对应这种观点，产生基于风险的测试策略。

**Pareto 原则（也叫 80/20 原则）**

第一个含义：80%的软件缺陷常常生存在软件 20%的空间里。如果想使软件测试有效，就要更加关注那些经常或者可能出现错误的程序段，在那里发现软件缺陷的可能性会大的多。这一原则对与软件测试人员提高测试效率及缺陷发现率有着重大的意义。

第二个含义：在系统分析、设计、实现阶段的复审工作中能够发现和避免 80%的软件缺陷，此后的系统测试能够帮助我们找出剩余缺陷中的 80%，最后的 20%的软件缺陷可能只有在系统交付使用后用户经过大范围、长时间使用后才会暴露出来。因为软件测试只能够保证尽可能多地发现软件缺陷，却无法保证能够发现所有的软件缺陷。
第三个含义：实践证明 80%的软件缺陷可以借助人工测试而发现，20%的软件缺陷可以借助自动化测试能够得以发现。由于这两者间具有交叉的部分，因此尚有 5%左右的软件缺陷需要通过其他方式进行发现和修正

4、软件测试的经济学观点
**“一个好的测试用例在于它能发现至今未发现的错误”**，体现了软件测试的经济学观点。
这是由于在需求阶段修正一个错误的代价是 1，而在设计阶段就是它的 3～6 倍，在编程阶段是它的 10 倍，在内部测试阶段是它的 20～40 倍，在外部测试阶段是它的 30～70 倍，而到了产品发布出去时，这个数字就是 40～ 1000 倍。修正错误的代价不是简单地随着时间线性增长，而几乎是呈指数级增长的。因此，应该尽快尽早地发现缺陷。

5、软件测试的标准论
**验证**(Verification)是检验软件是否已正确地实现了产品规格书所定义的系统功能和特性
**有效性确认**(Validation)是确认所开发的软件是否满足用户真正需求的活动。
为了深入理解软件测试，可以从以下几个方面考虑：从软件测试的目的来理解、从软件测试的性质来理解、从软件开发角度来理解、从软件工程角度来理解、从软件质量保证角度来理解。

6、软件测试的目的
Grenford J.Myers 曾对软件测试的目的提出过以下观点：
（1）测试是为了证明程序有错，而不是证明程序无错误；
（2）一个好的测试用例是在于它能发现至今未发现的错误；
（3）一个成功的测试是发现了至今未发现的错误的测试。

（1）测试并不仅仅是为了找出错误，通过分析错误产生的原因和错误发生的趋势，可以帮助项目管理者发现当前软件开发过程中的缺陷，以便及时改进。
（2)测试帮助测试人员设计出有针对性的测试方法，改善测试的效率和有效性。
（3）没有发现错误的测试也是有价值的，完整的测试是评定软件质量的一种方法。

7、软件测试的原则
（1）完全测试的不可能性　（2）软件测试是有风险的活动　（3）测试无法显示潜伏的软件缺陷和故障(**任何情况下都不能保证软件缺陷不存在，唯一的方法就是继续测试，找到更多的缺陷**)　（4）充分注意测试中的群集现象　（5）杀虫剂现象(软件测试越多，对测试的免疫力越强，寻找更多软件缺陷就更加困难。克服的办法是：在软件测试中采用单一的方法不能高效和完全的针对所有软件缺

陷，因此软件测试应该尽可能的多采用多种途径进行测试）　（6）并非所有的软件缺陷都能修复(并非意味着软件测试员没有达到目的)　（7）难以描述的软件缺陷　（8）软件测试必须有预期结果(软件缺陷是经过对比而得出来的，没有预期结果的测试是绝不可以的。我们事先不知道或是无法肯定预期的结果，我们必然无法了解测试正确性)　（9）应当把“尽早地和不断地进行软件测试”作为软件测试者的座右铭　（10）程序员应该避免检查自己的程序

软件缺陷的定义：

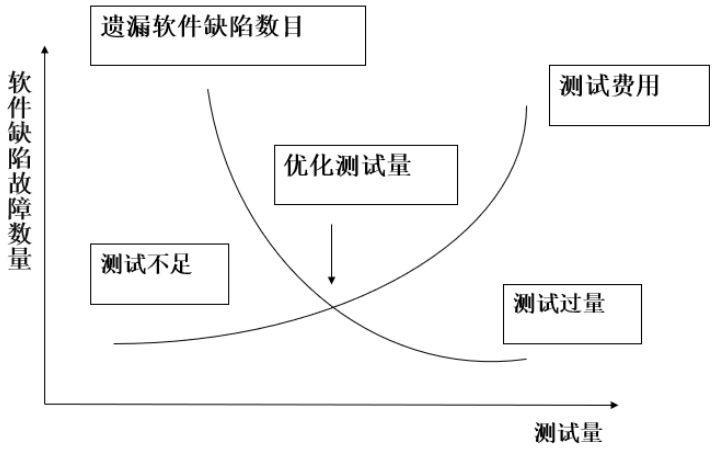
(1)软件未达到产品说明书中已经标明的功能；

(2)软件出现了产品说明书中指明不会出现的错误；

(3)软件未达到产品说明书中虽未指出但应当达到的目标；

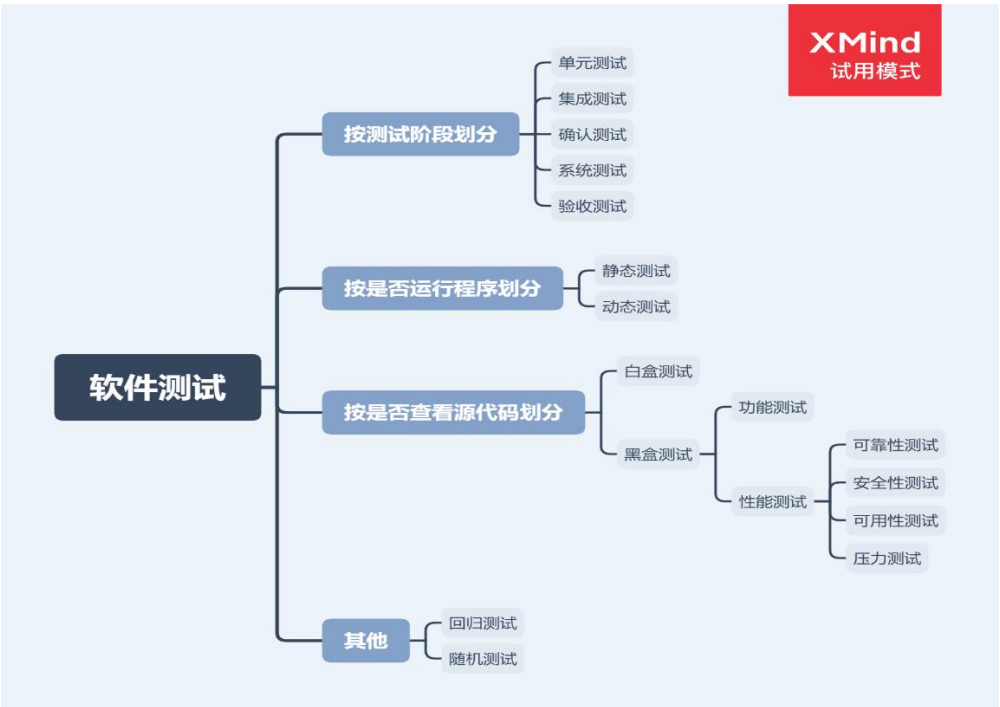
(4)软件功能超出了产品说明书中指明的范围；

(5)软件测试人员认为软件难以理解、不易使用，或者最终用户认为该软件使用效果不良。




•测试工作量与软件缺陷数量之间的关系

8、软件测试的分类



**按照开发阶段划分**

**单元测试：**模块测试，检查每个程序单元是否正确实现详细设计说明中的模块功能

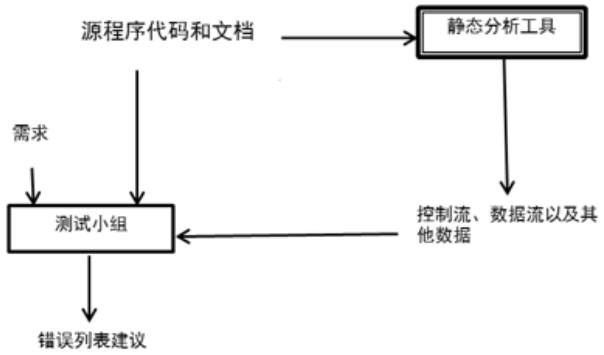
**集成测试：**组装测试，将所有的程序模块进行有序、递增的测试，检查程序单元或部件的接口关系

**系统测试：**检查完整的程序系统能否和系统(包括硬件、外设和网络、系统软件、支持平台等)正确配置、连接，并满足用户需求

**确认测试：**证实软件是否满足特定用途的需求，是否满足软件需求说明书的规定

**验收测试：**按照项目任务或合同，供需双方签订的验收依据文档进行的对整个系统的测试与评审，决定是否接受或拒收系统。

**按照是否运行程序划分**




**静态测试**，是不需要执行被测软件，而是采用分析和查看的方式，来发现软件当中的缺陷，包括需求文档、源代码、设计文档、以及其他与软件相关文档中的二义性和错误。最好由未参加代码编写的个人或小组来完成。测试小组还能够使用一个或多个静态测试工具，以源程序代码作为输入，产生大量的在测试过程有用的数据。

静态测试常用的方法如下：

（1）走查，走查是个非正式的过程，检查所有与源程序代码相关的文档。

（2）审查，审查比走查要求更加正规。

（3）静态代码分析工具，静态结构分析主要是以图形的方式表现程序的内部结构

**动态测试**是指通过运行实际被测试软件，通过观察程序运行时所表现的状态、行为等来发现软件的缺陷。并对被测程序的运行情况进行分析对比，以便发现程序表现的行为与设计规格或客户需求不一致的地方。

动态测试一般包括功能确认与接口测试，覆盖率分析、性能分析、内存分析等。

动态测试是一种经常运行的测试技术。但也有它的**局限性：**必须要借助测试用例完成；需要搭建特定的测试环境；不能发现文档中的问题。

由于动态测试与静态测试之间存在一定的协同性，又具有相对的独立性。所以在程序执行前进行静态测试，尽可能多地发现代码中隐含的缺陷；执行动态测试检查程序实时的行为，发现程序在运行时存在的缺陷。

**按照是否查看源代码划分**

**黑盒测试**又称功能测试或数据驱动测试，是将被测试软件看做一个黑盒子，**完全不考虑程序的内部结构和处理过程，只考虑系统的输入和输出**，在程序的接口进行测试，检查系统功能是否符合需求规格说明书的要求。

黑盒测试常用的测试方法有：等价类划分、边界值法、决策表法、因果图法、错误推测试法等。

黑盒测试的**优点**：黑盒测试用例与程序如何实现无关；测试用例的设计与程

序开发可并行设计；没有编程经验的人也可以设计测试用例。黑盒测试的**局限性**：不可能做到穷举测试；可能存在漏洞。

**白盒测试**又称结构测试或逻辑驱动测试；是根据被测试程序源代码的内部结构来设计测试用例的方法。常见的测试方法有：逻辑覆盖、基本路径和数据流测试等。

白盒测试的**优点**：可以利用不同的覆盖准则测试程序代码的各个分支，发现程序内部的编码错误；可以直接发现内存泄露问题；可以充当黑盒测试的检查手段等。白盒测试的**局限性**：因程序路径组合太多，同样不能做到穷举测试；由于设计测试用例不是根据客户需求说明进行的测试，可能存在需求方面的漏洞。

**灰盒测试**结合了白盒测试和黑盒测试的要素，关注输入的正确性，同时关注内部的表现；考虑了用户端、特定的系统知识和操作环境。他在系统组件的协同环境中评价应用软件的设计。

**按照是否需要人工干预划分**

**人工测试**是人为测试和手工测试的统称。人为测试的主要方法有桌前检查、代码审查和走查。用于软件开发各阶段的审查或评审都是人为测试。手工测试主要指在测试过程中，按照测试计划一步一步执行程序，得出测试结果并进行分析的测试行为。

**自动(化)测试**指的是利用测试工具对各种测试活动的管理与执行，并对测试结果自动进行分析。在测试的执行过程中，一般不需要人工干预。常用在功能测试、回归测试和性能测试等。

自动化测试的优点：提高测试效率；降低测试成本；具有一致性和可重复性；降低风险，增加软件的质量等。

自动化测试的局限性：自动化测试软件本身的问题；测试人员期望过高；有些人工测试是不能用自动化测试替代等。

**按照测试组织划分**

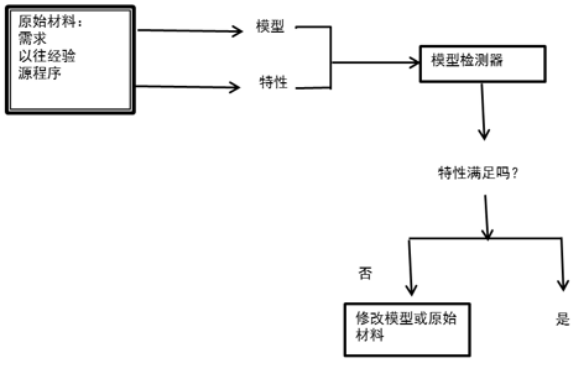
**开发方测试**：开发方通过检测和提供客观证据，证实软件的实现是否满足规定的需求，在开发环境下，开发方对提交的软件进行全面的自我检查

**用户测试**：在用户的应用环境中，用户通过运行软件，检测软件实现是否符合自己预期的要求，这里指用户的使用性测试。

**第三方测试**：介于软件开发方和用户方之间的测试组织的测试

**其他测试分类**

**（1）基于模型的测试与模型检测**，基于模型的测试是指对软件行为进行建模以及根据软件的形式化模型设计测试的活动；模型检测是指用来验证软件特定模型中的一个或多个特性的一类技术。




模型检测的要素

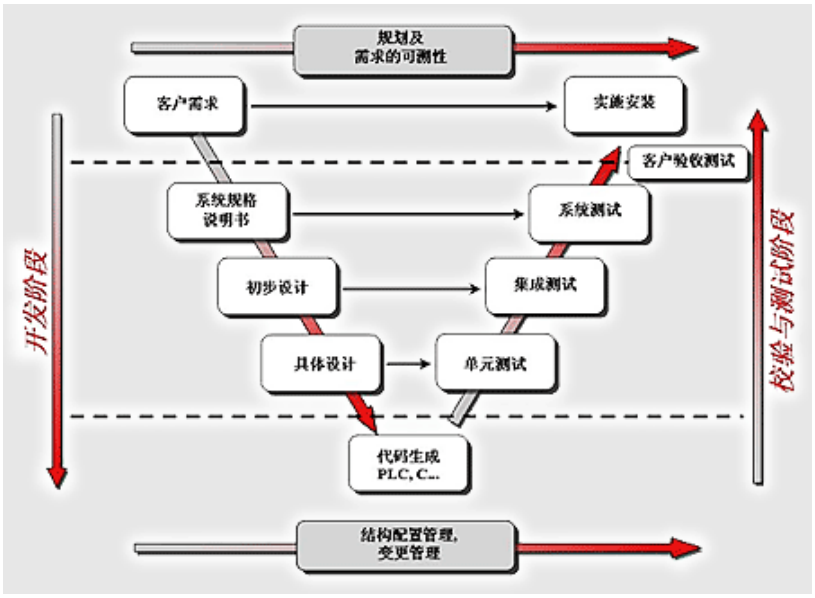
模型通常是有限状态的，是从一些原始材料中提取出来的，这些原始材料可能是需求文档，也可能是系统源代码本身。有穷状态模型中的每一个状态前都有一个或多个前置条件，当软件处于该状态时，这些特性必须满足。如左图所示说明模型检测过程。

（2）**冒烟测试**是指在测试中发现问题，也就是说找到了一个缺陷，由开发人员来修复这个缺陷，当修复完成后，是否真的解决了这个缺陷，或对其他模块是否存在影响，因此要针对这个问题进行专门的测试，这个测试过程称为冒烟测试。在许多情况下，经过测试后，发现修复某个问题会引起其他功能模块一系列的反应，导致产生新的缺陷。冒烟测试的优点是节省测试时间，防止创建失败。缺点是覆盖率较低。

（3）**随机测试**是根据测试说明书执行样例测试的一种重要补充手段，是保证测试覆盖完整性的有效方式和过程。随机测试主要针对系统的一些重要功能进行复测。还对软件更新和新增功能要进行重点测试。常与回归测试一起进行

9、软件测试模型

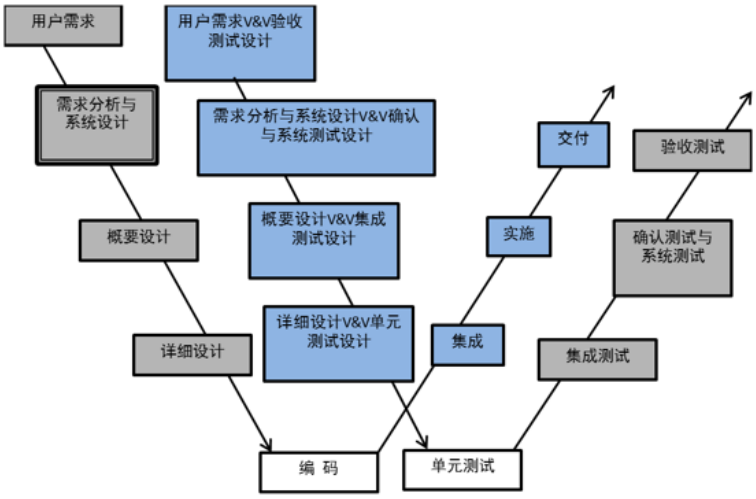
**(1)v-model 模型**是最早的软件生存期模型，在 20 世纪 80 年代由 Paul Rook 提出的。v-model 包含了三个等级，分别是生存期模型，分配模型，功能性工具需求模型，阐述了应当实施哪些活动，应当产生哪些结果，诸如此类。




v-model 指出，单元测试所检测代码的开发是否符合详细设计的要求。集成测试所检测此前测试过的各组成部分是否能完好地结合到一起。系统测试所检测已集成在一起的产品是否符合系统规格说明书的要求。而验收测试则检测产品是否符合最终用户的需求。所以 V-model 模型软件测试的策略既包括低层测试又包括高层测试，底层测试是为了验证系统源代码的正确性，高层是为了测试整个系统是否满足用户的需求。

V-model 的缺陷:仅仅把测试过程作为在需求分析、系统设计及编码之后的一个阶段,忽视了测试对需求分析,系统设计的验证,一直到后期的验收测试才被发现。

**(2)W 模型**由 Evolutif 公司提出,相对于 V-model,W-model 更科学,W-model 是 V-model 的发展。由于 V-model 的局限性，没有明确地说明早期的测试,无法体现“尽早地和不断地进行软件测试”的原则。在 V-model 中增加软件各开发阶段应同步进行的测试，演化为 W-model。如下图




相对于 V 模型而言，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（V&V）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，测试伴随着整个软件开发周期，测试的对象不仅仅是程序，需求、设计等同样要测试，也就是说，**测试与开发同步进行**

**W 模型具有如下特点：**

（1）在 V 模型中增加软件各开发阶段应同步进行的测试，被演化为一种 W 模型。

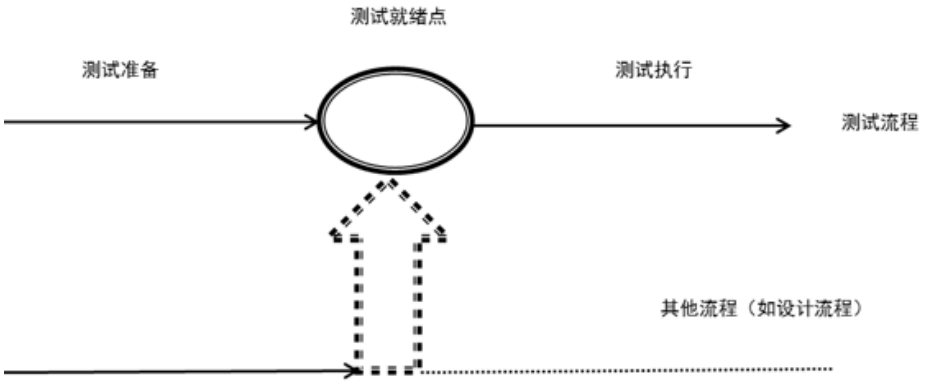
（2）开发是“V”，测试也是与此相重叠的“V”。

（3）W 模型体现了“尽早地和不断地进行软件测试”的原则。

（4）W 模型也存在局限性：在 W 模型中，需求、设计、编码等活动被视为串行，测试和开发活动保持着一种线性的前后关系，上一阶段结束，才开始下一个阶段工作，因此，W 模型无法支持迭代开发模型。

对于当前很多文档需要事后补充，或者根本没有文档的做法下，开发人员和测试人员都面临同样的困惑。

**(3)H 模型**，它将测试活动完全独立出来，形成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，如下图




V 模型和 W 模型都认为软件开发是需求、设计、编码等一系列**串行**的活动，而事实上，这些活动在大部分时间内可以**交叉**，因此，相应的测试也不存在严格的次序关系，单元测试、集成测试、系统测试之间具有反复迭代。

正因为 V 模型和 W 模型存在这样的问题，H 模型将测试活动完全独立出来，使得测试准备活动和测试执行活动清晰地体现出来。

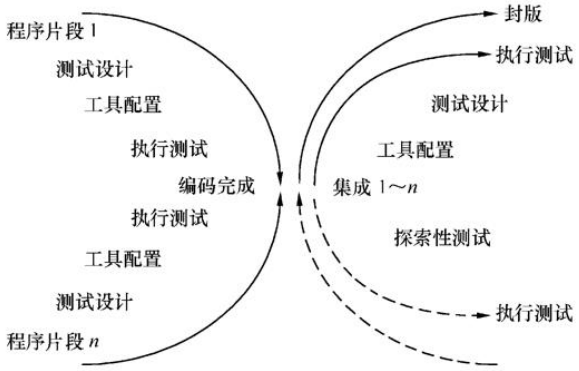
H 模型揭示了软件测试作为一个独立的流程贯穿于软件整个生命周期，与其他流程并发地进行，并指出软件测试要**尽早准备，尽早执行**。不同的测试活动可以按照某个次序先后进行，也可能是反复的，只要某个测试达到准备就绪点，测试执行活动就可以开展。

因此，H 模型具有如下意义：

（1）测试准备与测试执行分离，有利于资源调配，降低成本，提高效率。

（2）充分体现测试过程（不是技术）的复杂性。

**(4)X 模型**的基本思想是由 Marick 提出的，他认为一个模型必须能处理开发的所有方面，包括交接，频繁重复的集成，以及需求文档的缺乏等等。 而 X-model 填补了 V-model 的缺陷，并可为测试人员和开发人员带来明显的帮助。如图所示。

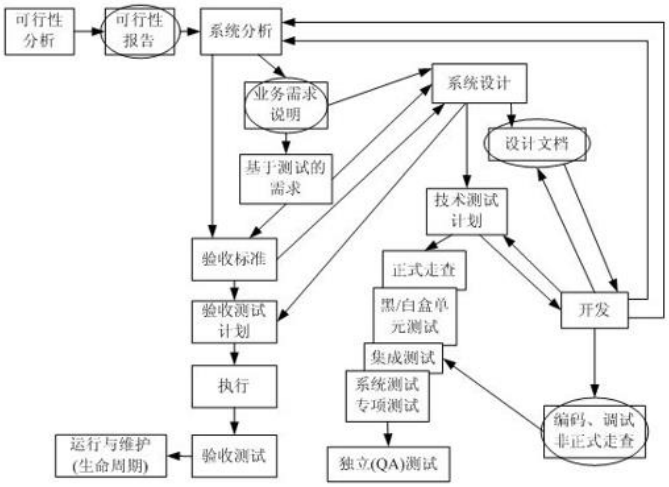



X 模型的左边描述的是针对单独程序片段所进行的编码和测试，通过频繁的交接，集成最终合成为可执行的程序。

X 模型右上方定位了已通过集成测试的成品进行封版并提交给用户，也可以作为更大规模和范围内集成的一部分。多根并行的曲线表示变更可以在各个部分发生。

X 模型右下方定位了探索性测试。这是不进行事先计划的特殊类型的测试，往往是在测试计划之外发现软件错误。

**(5)前置模型**，它是将测试和开发紧密结合的模型，该模型提供了轻松的方式，可以使你的项目加快速度，如图所示




Pretest-model 体现了以下的要点:

开发和测试相结合

对每一个交付内容进行测试

让验收测试和技术测试保持相互独立

反复交替的开发和测试

引入新的测试理念

10、测试模型的使用

V-model 强调了在整个软件项目开发中需要经历的若干个测试级别，而且每一个级别都与一个开发级别相对应，但它忽略了测试的对象不应该仅仅包括程序，或者说它没有明确地之处应该对软件的需求、设计进行测试。

W-model 强调了测试计划等工作的先行核对系统需求和系统设计的测试，但 W-model 和 V-model 一样也没有专门对软件测试流程予以说明，因为事实上，随着软件质量要求越来越为大家所重视，软件测试也逐步发展成为一个独立于软件开发部的组织，就每一个软件测试的细节而言，它都有一个独立的操作流程。比

如，现在的第三方测试，就包含了从测试计划和测试案例编写，到测试实施以及测试报告编写的全过程。

H-model 强调测试是独立的，只要测试准备完成，就可以执行测试了。

X-model 和 Pretest-model 又在此基础上增加了许多不确定因素的处理情况，因为在真实项目中，经常会有变更的发生，例如需要重新访问前一阶段的内容，或者跟踪并纠正以前提交的内容，修复错误，排除多余的成分，以及增加新发现的功能等。

11、测试用例(Test Case)是指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。其内容包括测试目标、测试环境、输入数据、测试步骤、预期结果、测试脚本等，最终形成文档。

IEEE 标准 610(1990)给出的定义:测试用例是一组测试输入、执行条件和预期结果的集合，目的是要满足一个特定的目标，比如执行一条特定的程序路径或检验是否符合一个特定的需求。

测试用例的作用：指导测试的实施、规划测试数据的准备、评估测试结果的度量基准、保证软件可维护性和可复用性、分析缺陷的标准

**软件测试设计的关键问题可以概括为 5W1H:**

Why:为什么测试？对功能、性能、可用性、容错性、安全性等测试，检验是否符合相关要求。

What:测什么？测试的对象可以是文档，代码，图表等。

Where:在哪里测？测试用例的环境，包括系统的硬件、软件和网络环境等。

When:什么时候测？测试用例所需的前提条件，尽早开始。

Which:什么数据？测试用例设计的各种数据。

How:如何执行？结果怎样？要据测试用例设计的步骤来执行，最后进行结果比较，确定是否一致。若一致才能通过测试。

**从两个层次考虑测试用例：**

(1)低层次——从单个测试用例看，衡量其描述的规范性、可理解性及可维护性等

(2)高层次——以满足某一个测试目标或测试任务来衡量一组测试用例的结构、设计思路和覆盖率等。

**测试用例设计的基本原则**

（1） 代表性：测试用例能代表并覆盖各种合法的或不合法、边界内的或越界的以及极限的输入数据、操作和环境的设置

（2） 可判定性：测试执行的结果的正确性是可以判定的。每一个测试用例都应该有相应的预期结果

（3） 可在现性：对于同样的测试用例，系统执行的结果应该是相同的，并且相同的测试的执行过程可以反复操作

**测试用例的设计步骤**

（1） 制定测试用例的策略和思想，在测试计划中描述出来

（2） 设计测试用例的框架，也就是测试用例的结构

（3） 细化结构，逐步设计出具体的测试用例
（4） 通过测试用例的评审，不断优化测试用例

测试用例的维护
（1） 产品特性没变，只是根据漏掉的缺陷来完善测试用例。这时候，增加和修改测试用例均可，因为当前被修改的测试用例对相应的版本都有效，不会影响某个特定版本所拥有的测试用例。
（2） 原有产品特性发生变化，不是新功能特性，而是功能增强，这时候原有的测试用例只对先前版本有效，对当前新的版本无效。此时，绝不能修改测试用例，只能增加新的测试用例，不能影响源于的测试用例。
（3） 原有功能取消了，这是只要将与该功能对应的测试用例在新版本上置为空标志或“无效”状态，但不能删除这些测试用例，因为它们对先前某个版本还是有效的。
（4） 完全新增加的特性，需要增加新的测试用例

12、软件测试的关键问题
**（1）测试由谁来执行？**
通常软件产品的开发设计包括开发者和测试者两种角色。开发者：通过开发形成产品，如分析、设计、编码调试或文档编制等。测试者通过测试来检测产品中是否存在缺陷，包括根据特定目的而设计测试用例、构造测试、执行测试和评价测试结果等。通常由开发者负责完成第一阶段的代码单元测试，而系统测试则由独立的测试人员或专门的测试机构进行。
按照测试实施组织划分，软件测试可分为开发方测试(α测试)、用户测试(β测试)、第三方测试。
**开发方测试**通常也叫“验证测试”或“α测试”。开发方通过检测和提供客观证据，证实软件的实现是否满足规定的需求，验证测试(开发方测试)是在**软件开发环境下**，由开发者检测与证实软件的实现是否满足软件设计说明或软件需求说明的要求。主要是指在软件开发完成以后，开发方对要提交的软件进行全面的**自我检查与验证**，可以和软件的“系统测试”一并进行。
**用户测试：**在用户的应用环境下，用户通过运行和使用软件，检测与核实软件实现是否符合自己预期的要求。通常情况用户测试不是指用户的“验收测试”，而是指用户的使用性测试，由用户找出软件的应用过程中发现的软件缺陷与问题，并对使用质量进行评价。β测试通常被看成是一种“用户测试”。β测试主要是把软件产品有计划地免费分发到目标市场，让用户大量使用，并评价、检查软件。通过用户各种方式的大量使用，来发现软件存在的问题与错误，把信息反馈给开发者修改。
**第三方测试：**介于软件开发和用户方之间的测试组织的测试。也称为独立测试。软件质量工程强调开展独立验证和确认（IV&V）活动。由在技术、管理和财务上与开发方和用户方相对独立的组织进行的软件测试。一般情况下是在模拟用户真实应用环境下，进行软件确认测试。

**（2）测试什么**
① 软件产品的组成：软件产品并不仅仅是从软盘或者光盘安装到计算机上的程序，还包括许多隐含的内容，容易被忽视，但这些也往往是包含软件缺陷的测试

对象，需要软件测试员铭记在心！
② 软件测试对象：软件测试不仅仅是对程序的测试，而是贯穿于软件定义和开发的整个过程。因此，软件开发过程中产生的需求分析、概要设计、详细设计以及编码等各个阶段所得到的文档，包括需求规格说明、概要设计说明、详细设计规格说明以及源程序，都是软件测试的对象。

软件测试在软件生命周期，也就是从开发设计、运行、直到结束使用的全过程中，主要横跨以下两个测试阶段：
第一阶段：**单元测试阶段**，即在每个模块编写出以后所做的必要测试。
第二阶段：**综合测试阶段**，即在完成单元测试后进行的测试，如集成测试、系统测试、验收测试等。
研究表明，通常表现在程序中的故障，并不一定是由编码所引起的，可能是需求分析，概要设计、详细设计等阶段的问题所致，要排除故障就必须追溯到前期的工作（这就涉及到下一个问题——什么时候进行测试）

**（3）什么时候进行测试？**
测试可以是一个与开发并行的过程，也可以是开发完成某个阶段任务后的活动，即模块开发结束之后，还可以在各模块装配成为一个完整的程序之后再进行测试。

**（4）怎样进行测试**
对软件进行测试就是根据软件的功能规范说明和程序实现，利用各种测试方法，生成有效的测试用例，对软件进行测试

**（5）测试停止的标准**
从现实和经济的角度来看，对软件进行完全测试是不可能的，那么，什么时候停止测试呢？
① 测试超过了预定时间，则终止测试
② 执行了所有测试用例，但并没有发现故障，则终止测试
③ 使用特定的测试用例设计方法作为判定测试停止的基础
④ 给出测试停止的要求，例如发现并修改 100 个软件故障
⑤ 根据单位时内查出故障的数量决定是否停止测试
⑥ 软件系统经过单元、集成、系统测试，分别达到各自停止标准。软件系统通过验收测试，并已得出验收测试结论
⑦ 软件项目需暂停以进行调整时，测试应随之暂停，并备份暂停点数据。或者软件项目在开发生命周期内出现重大估算、进度偏差，需暂停或终止时，测试应随之暂停或终止，并备份暂停或终止点数据。

13、软件测试中的误区
误区 1：调试和测试是一样的
误区 2：软件测试对象就是程序
误区 3：软件测试是测试人员的事情
误区 4：好的软件质量是通过测试得到的
误区 5：把不合格的开发人员安排做测试
误区 6：关注于测试的执行而忽视测试的设计

误区 7：测试自动化是万能的
误区 8：测试是为了证明软件的正确性