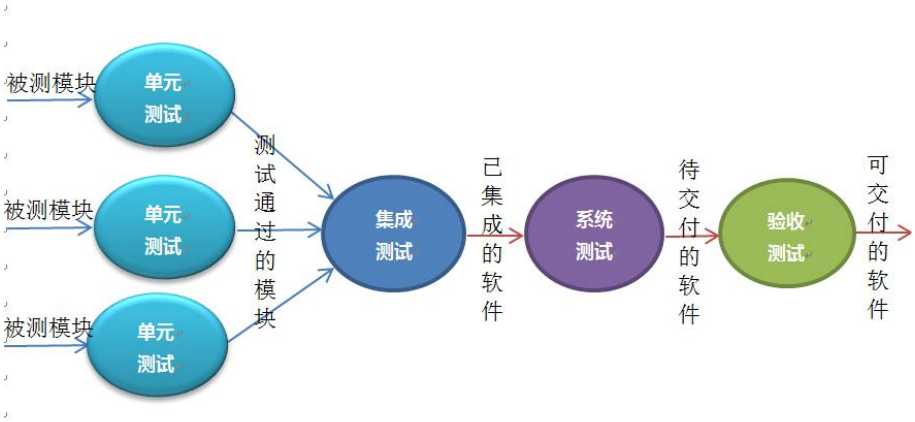
2 软件测试过程与方法

2.1 软件测试过程

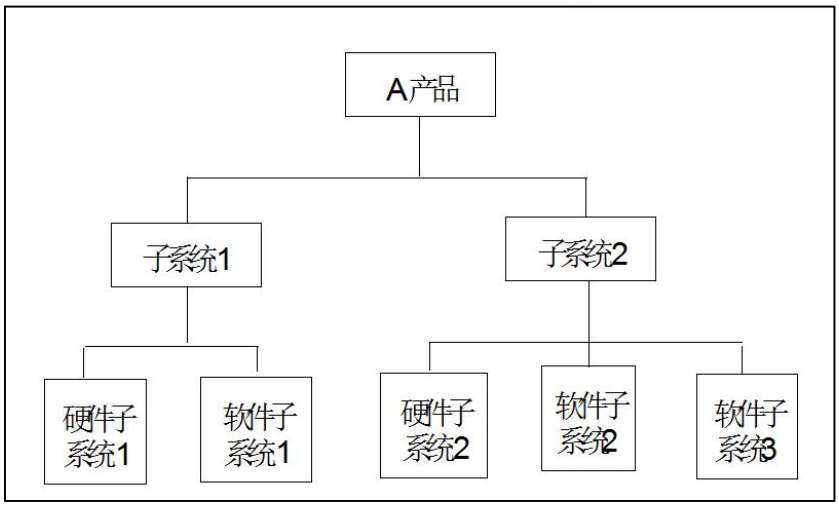
2.1.1 测试过程阶段划分

软件测试过程按阶段划分为：单元测试、集成测试、系统测试、验收测试。



1. **单元测试(Unit Testing)**
   - 单元测试是针对软件基本组成单元来进行正确性检验的测试工作。
   - 单元测试的目的是检测软件模块对《详细设计说明书》的符合程度。
2. **集成测试(Integration Testing)**
   - 集成测试是在单元测试的基础上，将所有模块按照概要设计要求组装成为子系统或系统，验证组装后功能以及模块间接口是否正确的测试工作。
   - 集成测试的目的是检测软件模块对《概要设计说明书》的符合程度。
3. **系统测试(System Testing)**
   - 系统测试是将已经集成好的软件系统，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员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测试活动。
   - 系统测试的目的在于通过与《需求规格说明书》作比较，发现软件与系统需求定义不符合或与之矛盾的地方。
   - 系统测试的对象是软硬件集合在一起的系统，不应是独立的软件与硬件环境。当然具体操作、执行时可根据实际情况来组织。验证时应尽可能模拟实际的运行环境与条件。

系统测试对象对应的是产品级



4. 单元、集成、系统测试的比较

区别方面	区别内容
测试方法不同	单元测试属于白盒测试范畴 集成测试属于灰盒测试范畴 系统测试属于黑盒测试范畴
考察范围不同	单元测试主要测试单元内部的数据结构、逻辑控制、异常处理等 集成测试主要测试模块之间的接口和接口数据传递关系，及模块组合后的整体功能 系统测试主要测试整个系统相对于需求的符合度
评估基准不同	单元测试的评估基准主要是逻辑覆盖率 集成测试的评估基准主要是接口覆盖率 系统测试的评估基准主要是测试用例对需求规格的覆盖率

5. 验收测试（User Acceptance Testing）

验收测试也称交付测试，在通过了内部系统测试及软件配置审查之后，就可以开始验收测试，是针对用户需求、业务流程进行的正式的测试，以确定系统是否满足验收标准，由用户、客户或其他授权机构决定是否接受系统。

验收测试是以用户为主的测试，验收组由项目组成员、用户代表等组成，验收测试根据合同、《需求规格说明书》或《验收测试计划》对成品进行验收测试。验收测试包括α（ALPHA)测试和β（BETA)测试 。

- **Alpha（α）测试**
  1) α 测试是由用户在开发环境下进行的测试，也可以是开发机构内部的用户在模拟实际操作环境下进行的测试。
  2) α 测试时，软件在一个自然设置状态下使用。开发者坐在用户旁，随时记下错误情况和使用中的问题，这是在受控制的环境下进行的测试。
  3) α 测试的目的主要是评价软件产品的 FLURPS（即功能、局域化、可用性、可靠性、性能和技术支持）。
- **Beta（β）测试**
  1) β 测试是由软件的多个用户在一个或多个用户的实际使用环境下进行的测试
  2) 与 α 测试不同的是，β 测试时开发者通常不在测试现场。因而，β 测试是在开

发者无法控制的环境下进行的软件现场应用

- **α 测试与 β 测试区别：**

- 环境方面：α 测试是模拟环境，β 测试是实际环境；
- 用户方面：α 测试是内部用户，β 测试真实用户；
- 可控方面：α 测试开发在现场可指导，可控；β 测试开发不在现场，不可控；
- 数据方面：α 测试数据模拟数据；β 测试真实数据；

- **验收通过的标准：**

- 需求覆盖率达到一定的标准，比如 99%
- 用例覆盖率、用例执行率达到一定的标准，比如：90%
- 缺陷修复率达到一定的标准，比如 98%；
- 不存在 3 级以上的重大 bug；
- 主业务流程通过；
- 主要功能 100%实现；
- 用户体验较好等

6. 回归测试（Regression Testing）

回归测试是在测试过程中对已发现的 bug 经过修复后，再次验证确认是否修改及对其他模块造成影响的过程。其目的是验证缺陷得到了正确的修复，同时对系统的变更没有影响以前的功能。回归测试可以发生在任何一个阶段，包括单元测试、集成测试和系统测试。

- **回归测试流程**

以下流程适合于单元测试、集成测试和系统测试

1) 在测试策略制定阶段，制定回归测试策略
2) 确定需要回归测试的版本
3) 回归测试版本发布，按照回归测试策略执行回归测试
4) 回归测试通过，关闭缺陷跟踪单（问题单）；回归测试不通过，缺陷跟踪单返回开发人员，开发人员重新修改问题，再次提交测试人员回归测试。

- **回归测试策略**

✧ **完全重复测试：**重新执行所有在前期测试阶段建立的测试用例，来确认问题修改的

正确性和修改的扩散局部影响性；

✧ **选择性重复测试**：即有选择地重新执行部分在前期测试阶段建立的测试用例，来测试被修改的程序，常用的用例选择方法可以分为三种：

1) **覆盖修改法**：即针对被修改的部分，选取或重新构造测试用例验证没有错误再次发生的用例选择方法。
2) **周边影响法**：该方法不但要包含覆盖修改法确定的用例，还需要分析修改的扩散影响，对那些受到修改间接影响的部分选择测试用例验证它没有受到不良影响。该方法比覆盖修改法更充分一点。
3) **指标达成方法**： 这是一种类似于单元测试的方法，在重新执行测试前，先确定一个要达成的指标，如修改部分代码 100%的覆盖、与修改有关的接口 60%的覆盖等，基于这种要求选择一个最小的测试用例集合。

● **回归测试自动化**

回归测试是一个重用以前成果的测试，很难预料到要经过多少次回归系统才能达到满意的水平，结果这种回归测试将可能演变成一种重复的、令人心烦意乱的工作，效果与人员的积极性将大打折扣，因此，在回归测试道路上的自动化便是我们工作的追求；

7. 冒烟测试（Smoke Test）

- 冒烟测试（Smoke Test），也是版本验证测试，是自由测试的一种，由开发人员与测试人员共同进行。主要应用于系统、集成测试的新版本发布，确认新的版本是否存在致命性 bug，功能可以正常运行，不会影响下一轮测试的进行。如果上述都符合那么这个版本就可以进行下一轮测试。
- 冒烟测试目的主要在于验证产品的主要功能、流程，用于确认新版本可测，决定是否开启一轮全新的系统测试。
- 冒烟测试的最大优点在于耗时较短，节约测试的时间成本，减少测试轮数。

● **冒烟测试通过的标准：（提前与开发达成一致）**

- 主要业务流程能够进行；
- 主要重要功能模块要实现；
- 无重大或致命 bug，影响项目流程；
- 功能完成率达到 80%以上；
- 冒烟测试用例通过率到达 90%以上；

主要的测试文档

- 测试计划：指明测试范围、方法、资源，以及相应测试活动的时间进度安排表的文档。
- 测试方案：指明为完成软件或软件集成特性的测试而进行的设计测试方法的细节文档。
- 测试用例：指明为完成一个测试项的测试输入、预期结果、测试执行条件等因素的文档。
- 测试规程：指明执行测试时测试活动序列的文档。

- 测试报告：指明执行测试结果的文档。
- 测试日报：每天测试执行情况的记录和总结。

- **系统测试过程**

系统测试过程	输入	输出
计划阶段	《软件开发计划》《软件测试计划》《需求规格说明书》	《系统测试计划》
设计阶段	《需求规格说明书》《系统测试计划》	《系统测试方案》
实现阶段	《需求规格说明书》《系统测试计划》《系统测试方案》	《系统测试预测试用例》 《系统测试用例》 《系统测试规程》
执行阶段	《系统测试计划》《系统测试方案》、《系统测试预测试用例》、《系统测试用例》、《系统测试规程》	《系统测试预测试报告》 《系统测试报告》 《缺陷报告单》

- **集成测试过程**

集成测试过程	输入	输出
计划阶段	《软件开发计划》《软件测试计划》《概要设计说明书》	《集成测试计划》
设计阶段	《概要设计说明书》《集成测试计划》	《集成测试方案》
实现阶段	《概要设计说明书》《集成测试计划》《集成测试方案》	《集成测试用例》 《集成测试规程》
执行阶段	《集成测试计划》《集成测试方案》《集成测试用例》《集成测试规程》	《集成测试报告》 《缺陷报告单》

- **单元测试过程**

单元测试过程	输入	输出
计划阶段	《软件开发计划》《软件测试计划》《详细设计说明书》	《单元测试计划》
设计阶段	《详细设计说明书》《单元测试计划》	《单元测试方案》
实现阶段	《详细设计说明书》《单元测试计划》 《单元测试方案》	《单元测试用例》 《单元测试规程》
执行阶段	《单元测试计划》《单元测试方案》 《单元测试用例》《单元测试规程》	《单元测试报告》 《缺陷报告单》

1. 软件研发各阶段的主要任务

软件研发阶段	主要任务
需求分析阶段	需求分析，完成 SRS 软件需求规格说明书的评审 进行需求跟踪 系统测试计划、设计及评审
概要设计阶段	进行概要设计及评审，完成 HLD 文档 系统测试方案、用例的设计及评审 需求跟踪更新 集成测试计划、设计及评审
	进行软件详细设计及评审，完成 LLD 文档

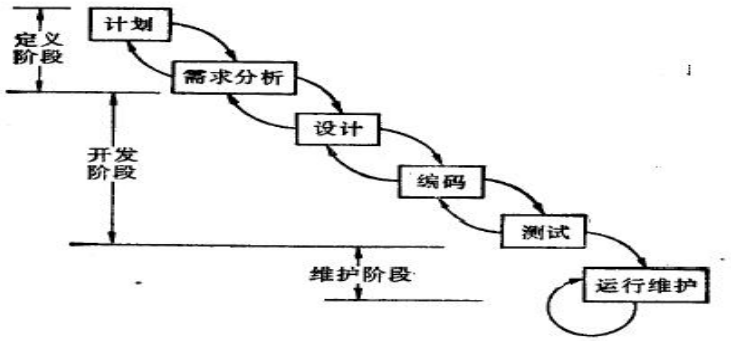
详细设计阶段	集成测试方案和用例的设计及评审 需求跟踪更新 单元测试计划、设计及评审
软件编码阶段	软件编码 代码静态质量检查 代码评审 单元测试方案和用例的设计及评审
单元测试执行阶段	单元测试用例执行 单元测试缺陷记录及修复 单元测试日报和报告写作 单元测试缺陷的回归测试
集成测试执行阶段	集成测试用例执行 集成测试缺陷记录及修复 集成测试日报和报告写作 集成测试缺陷的回归测试
系统测试执行阶段	系统测试预测试项执行 系统测试预测试报告写作 系统测试用例执行 系统测试缺陷记录及修复 系统测试日报和报告写作 系统测试缺陷的回归测试

3.1.2 测试过程模型

1. 常见的测试过程模型

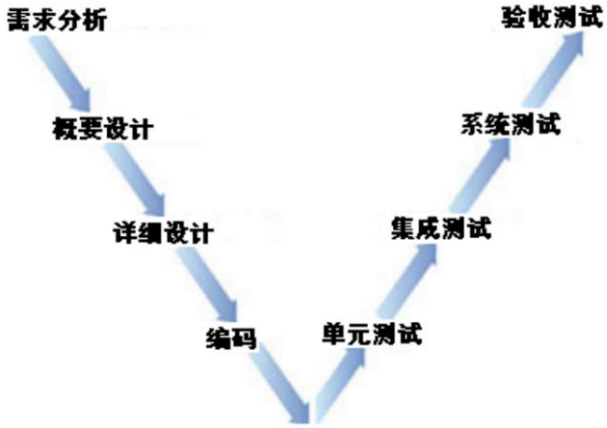
测试过程中常见的模型有：瀑布模型、 V 模型、 W/V&V 模型、 H 模型、 X 模型和敏捷模型等。各模型的具体介绍如下：

- **瀑布模型**



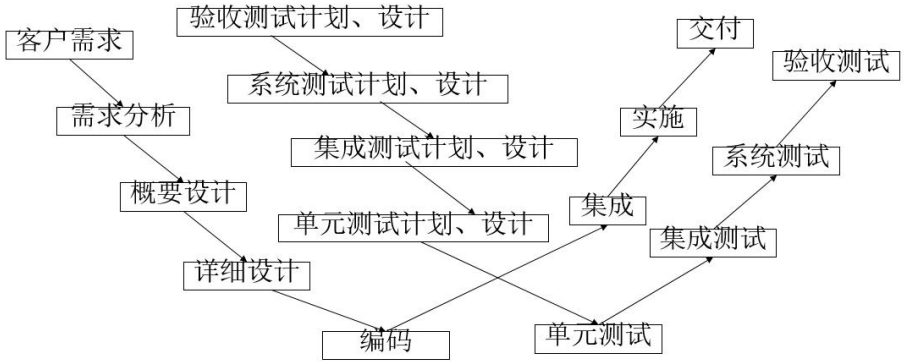
- 瀑布模型是应用的最为广泛的一种模型，也是最容易理解和掌握的模型，然而它的缺陷也是显而易见的。遗漏的需求或者不断变更的需求会使得该模型无所适从。然而，对于那些容易理解、需求稳定的项目，采用瀑布模型会是比较适合的，因为你可以按部就班的去处理复杂的问题。在质量要求高于成本和进度的时候，该模型表现的尤其突出。
- 在瀑布模型中，测试是在编码结束后才介入，对软件开发流程前期质量没有保证。

- V 模型



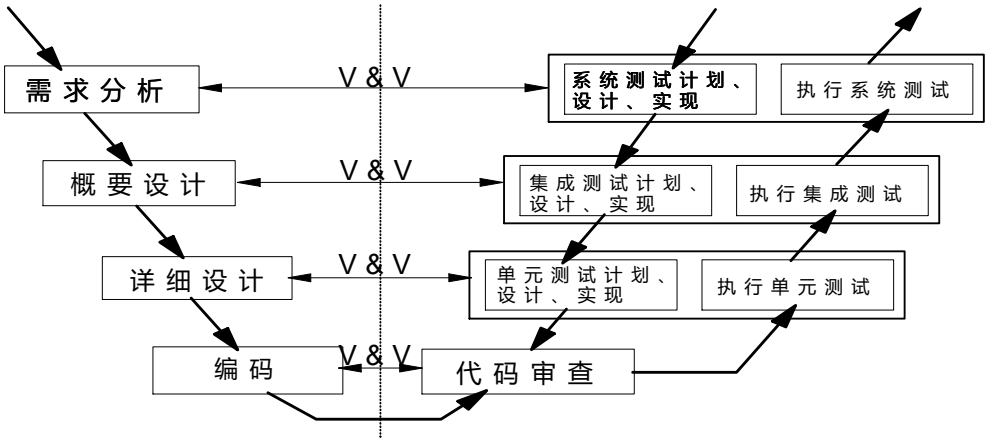
- V 模型：是软件开发瀑布模型的变种，主要反映测试活动与分析和设计的关系；
- 局限性：把测试作为编码之后的最后一个活动，需求分析等前期产生的错误直到后期的验收测试才能发现。

- W 模型



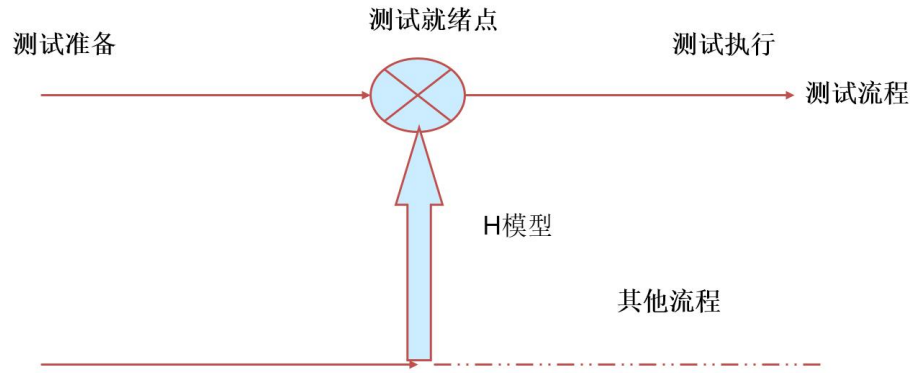
- 特点：W 模型是 V 模型的延伸，加强了 V 模型中各个阶段的测试，并将开发与测试同步。
- 作用：针对各个阶段进行监控，能够较早的发现软件中的缺陷。
- 局限：同 V 模型一样，仍将需求至编码阶段作为串行实施，前后依赖较强，不利于变更。

- V&V 模型



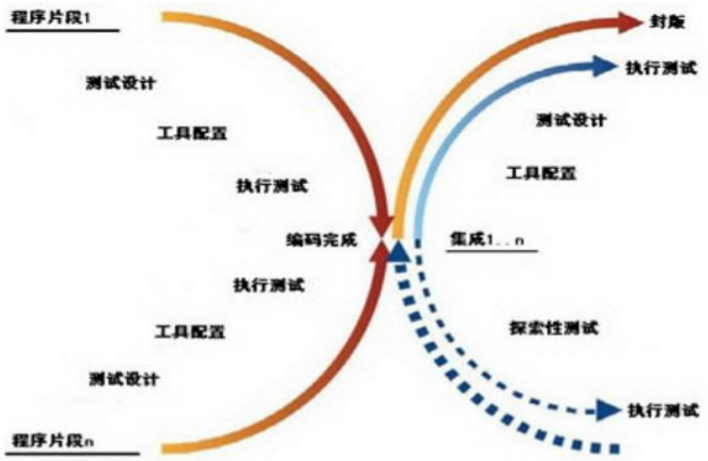
- V&V（验证与确认 V&V）模型实现了测试设计和测试执行相分离；揭示了软件测试活动分层和分阶段的本质特性，测试执行的顺序与开发活动相反；
- 验证(Verification)：是保证软件正确地实现特定功能的一系列活动，是检测每一阶段形成的工作产品是否与前一阶段定义的规格相一致。Are we building the product right?
- 确认(Validation)：是指保证所生产的软件可追溯到用户需求的一系列活动，是检测每一阶段的工作产品是否与最初定义的软件需求规格相一致，Are we building the right product?

- H 模型



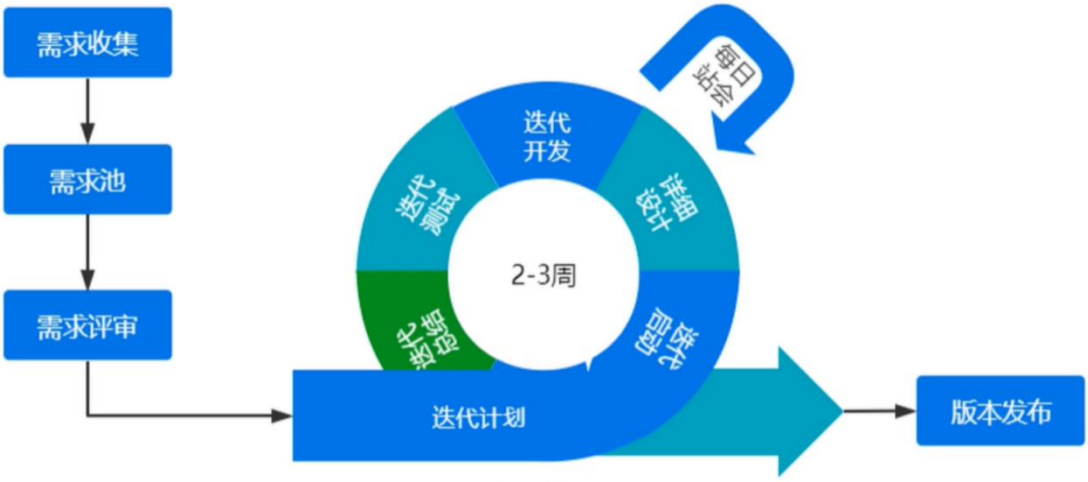
- H 模型：测试分为两类活动，测试准备活动和测试执行活动；测试准备活动是指后期的执行准备的前期的计划，策略，桩模块；测试流程是独立的，与软件研发的其它流程是并行的；
- 融合点：依照其他的流程的实现情况,触发测试的就绪点,从测试准备进入到测试执行。
- 特点：较早发现软件中的缺陷，能够适应软件项目的动态变更。
- 作用：将软件测试过程贯穿于整个项目周期，各阶段测试按层次进行，并且测试过程可循环实施。
- 局限：前期投入较大，各阶段交互时间较长，变更频繁也对软件产品质量构成威胁。

- X 模型



- 特点：该模型左半部分是针对程序片段进行相互分离后进行频繁的交接，最后合成可执行程序；右半部分即对生成的可执行程序进行测试，同时有针对性进行各种测试。
- 作用：能够节约时间，较早发现软件产品的缺陷。
- 局限：前期的拆分过多，对测试人员要求较高，实施较难。
- X 模型的左边描述的是针对单独程序片段所进行的相互分离的编码和测试，此后将进行频繁的交接，通过集成最终成为可执行的程序，然后再对这些可执 行程序进行测试。已通过集成测试的成品可以进行封装并提交给用户，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 模型还定位了探索性测试，不进行事先计划的特殊类型的测试，这一方式往往能帮助有经验的测试人员在测试计划之外发现更多的软件错误。

- 敏捷模型-迭代测试



2.2 测试常见方法

测试活动从不同的角度出发，可以有不同的分类。常见的测试分类包括：

- 黑盒测试和白盒测试

- 静态测试和动态测试
- 人工测试和自动化测试

还有其它一些分类方式，要把握的是搞清每一种分类的具体含义。任何软件产品都可以使用以下的两种方法之一进行测试：

1) 已知产品的需求规格，但不知道其内部实现，可以进行测试证明每个需求是否实现。
2) 已知产品的内部实现过程，可以通过测试证明每种内部操作是否符合设计规格的要求，所有内部成分是否已经过检查。

2.2.1 白盒测试和黑盒测试

1. 什么是白盒测试

白盒测试是依据被测软件分析程序内部构造，并根据内部构造设计用例，来对内部控制流程进行测试，可完全不顾程序的整体功能实现情况。白盒测试是基于程序结构的逻辑驱动测试。白盒测试又可以被称为玻璃盒测试、透明盒测试、开放盒测试、结构化测试、逻辑驱动测试.

- **为什么进行白盒测试**

- 我们只要保证需求得到实现就行了，为何要花费时间和精力来测试内部的逻辑细节呢？
- 白盒测试一般在测试前期进行，通过达到一定的逻辑覆盖率指标，使得软件内部逻辑辑控制结构上的问题能基本得到消除
- 白盒测试能保证内部逻辑结构达到一定的覆盖程度，能给予软件代码质量更大的保证
- 白盒测试发现问题后解决问题的成本较低

- **白盒测试的常用技术**

白盒测试一般会用到静态分析和动态分析两类技术。常用的有：

- 静态分析：控制流分析、数据流分析、信息流分析等
- 动态分析：逻辑覆盖测试（分支测试、路径测试等）、程序插装等

- **白盒测试的特点**

- 测试人员需要了解软件的实现；
- 可以检测代码中的每条分支和路径；
- 揭示隐藏在代码中的错误；
- 对代码的测试比较彻底；
- 实现代码结构上的优化；
- 白盒测试投入较大，成本高；
- 白盒测试不验证规格的正确性。

2. 什么是黑盒测试

黑盒测试把被测对象看成一个黑盒，只考虑其整体特性，不考虑其内部具体实现。黑盒测试针对的被测对象可以是一个系统、一个子系统、一个模块、一个子模块、一个函数等。黑盒测试又可以被称为基于规格的测试。

- **常见的黑盒测试类型**

- 功能性测试：一种是顺序测试每个程序特性或功能，另一种途径是一个模块一个模块的测试，即每个功能在其最先调用的地方被测试；
- 容量测试：检测软件在处理海量数据时的局限性，能发现系统效率方面的问题；
- 负载测试：检测系统在一个很短时间内处理一个巨大的数据量或执行许多功能调用上的能力；
- 恢复性测试：主要保证系统在崩溃后能够恢复外部数据的能力；

- **黑盒测试类型和质量模型**
  - 黑盒测试类型都来源于质量模型
  - 将软件的特性和质量特性结合起来就得到了测试类型
  - 一个软件特性可以和一个质量特性结合得到一个测试类型
  - 一个软件特性可以和多个不同的质量特性结合得到多个不同的测试类型
- **常用的黑盒测试方法**

  常用的黑盒测试方法：等价类划分法、边界值分析法、因果图分析法、判定表法、状态迁移法、流程分析法、正交试验法等。

  不管是什么测试方法，都是为了减少测试时的测试用例数，都是为了用尽量少的测试用例去完成测试，去发现更多的问题。
- **黑盒测试的特点**
  - 对于更大的代码单元来说（子系统甚至系统级）比白盒测试效率要高；
  - 测试人员不需要了解实现的细节，包括特定的编程语言；
  - 从用户的视角进行测试，很容易被大家理解和接受；
  - 有助于暴露任何规格不一致或有歧义的问题；
  - 没有清晰的和简明的规格，测试用例是很难设计的；
  - 不能控制内部执行路径，会有很多内部程序路径没有被测试到；
  - 不能直接针对特定的程序段，这些程序可能非常复杂（因此可能隐藏更多的问题）。

3. **灰盒测试**
   - 根据利用的被测对象信息的不同，会采用不同的方法进行测试。
   - 利用被测对象的整体特性信息，采用黑盒测试方法
   - 利用被测对象的内部具体实现信息，采用白盒测试方法
   - 如果既利用被测对象的整体特性信息，又利用被测对象的内部具体实现信息，采用的就是灰盒测试方法。两种信息占的比例不同，相应的灰度就不同。完全是整体特性信息，就是黑盒测试，完全是内部具体实现信息，就是白盒测试
   - 典型的灰盒测试比如集成测试和系统测试时借助 log 信息

2.2.2 静态测试和动态测试

静态测试：不运行被测试的软件系统，而是采用其他手段和技术对被测试软件进行检测的一种测试技术。例如：代码走读、文档评审、程序分析等都是静态测试的范畴。

动态测试： 按照预先设计的数据和步骤去运行被测软件系统，从而对被测软件系统进行检测的一种测试技术。

2.2.3 人工测试和自动化测试

人工测试：测试活动（如评审、测试设计、测试执行等）由人来完成，狭义上是指测试执行由人工完成，这是最基本的测试形式。

自动化测试：一般是指通过计算机模拟人的测试行为，替代人的测试活动，狭义上是指测试执行由计算机来完成。

1. **适合自动化测试的活动**
   - 软件需求变动不频繁（局部模块也可以）
   - 项目周期较长
   - 自动化测试脚本可重复使用
2. **自动化测试的意义**
   - 对程序新版本运行前一版本执行的测试，提高回归测试效率
   - 可以运行更多更频繁的测试，比如冒烟测试
   - 可以执行手工测试困难或不可能做的测试，比如大量的重复操作或者集成测试
3. **自动化测试的误区**
   - 不现实的期望，希望自动化能取代手工测试
   - 缺乏测试实践经验，手工测试都做不好，或者经验积累不够，就尝试自动化，很难成功
   - 期望自动化测试发现大量新缺陷，自动化只能保证测试执行效率，确保已有的问题不会再发生，发现新缺陷不是其目的
   - 安全性错觉，认为进行了自动化测试的软件就是安全的、质量有保证的
   - 只有手工测试做好了，明确了测试观察点，才能把自动化测试做好，所以手工测试是自动化测试的一个基础

2.2.4 软件产品的概念

- 软件研发可以看成一个生产过程，在这个过程中会有产品输出，或者叫做工件输出
- 输出的产品分成两类：

  最终产品，如编译后的软件、用户手册等，

  中间产品，如 SRS、HLD、LLD、代码等
- 无论是最终产品还是中间产品，都可以分成代码和文档
- 文档进一步细分还可以分成：

  开发文档，如 SRS、HLD、LLD 等

  测试文档，如测试计划、测试方案、测试用例等
- 只要是软件产品，都是测试的对象

2.3 系统测试常用类型

功能测试	**文档测试**
GUI 测试	**在线帮助测试**

性能测试	**配置测试**
兼容性测试	**可靠性测试**
可用性测试	**备份测试**
安装卸载测试	**网络测试**
安全性测试	**健壮性测试**
异常测试	**稳定性测试**

2.3.1 功能测试

概念：功能测试是根据产品的需求规格说明书和测试需求列表，验证产品的功能实现是否符合产品的需求规格。

目标： 功能测试主要是为了发现以下几类错误：

1) 是否有不正确或遗漏了的功能？
2) 功能实现是否满足用户需求和系统设计的隐藏需求？（要善于利用和分析顾客数据，隐含的需求往往就隐藏在顾客相关的历史数据和新调查的数据中。）
3) 输入能否正确接受（日期、格式、大小写、字符类型等）？能否正确输出结果（是否符合输出格式）？
4) 验证各业务流程正确性（所有可能业务路径）

2.3.2 GUI 测试

GUI（Graphical User Interface）测试是针对软件系统图形用户界面进行的测试，通常也叫用户界面测试。GUI 测试主要验证两方面的内容：1）界面实现与界面设计的吻合情况；2）确认界面处理的正确性。

GUI 测试对象：界面整体和界面中的元素。

1. 界面整体测试

界面整体是由一系列界面元素通过适当的形式组合而成的界面形式，最为常见的为各种窗口。包括各种对话框、单文档窗口、多文档父窗口，多文档子窗口等。界面整体测试主要是对界面的规范性、一致性、合理性以及定制性进行测试。

1) 规范性测试：软件的界面要尽量符合现行的标准和规范，并在应用软件中保持一致。
2) 合理性测试：测试软件界面是否与软件功能相融合，界面的颜色和布局是否协调。软件界面测试一般通过观察进行。比如：界面元素大小和布局是否协调，窗口比例是否合适。
3) 一致性测试：主要测试软件使用标准的控件、相同的信息表现方法等方面确保一致。比如：窗口布局是否一致，界面外观是否一致，颜色的使用是否一致等。
4) 界面定制性测试。主要针对适用于多层次用户的软件，测试由于用户熟练程度不同、使用频度不同、角色不同，需要不同的操作方式或用户界面。比如：界面工具栏、状态栏等要素的可定制性。

   页面风格、排版布局、字体颜色、工具按钮等是否协调统一、符合标准规范;

2. 界面元素测试

界面元素包括简单界面元素和组合类界面元素。简单界面元素是指功能和属性相对

比较单一的界面区域，即通常所指的各种控件。组合类界面元素主要指一些复杂的界面元素，比如工具栏，组合框，表格，菜单栏等。通常，我们从以下方面对界面元素进行测试。

1) 窗口测试：窗口控件的大小、对齐方向、颜色、背景等是否和程序设计规约一致、窗口是否支持最大化或放大。
2) 菜单测试：菜单的设计是否符合要求、菜单项的顺序是否合理、图形的布局是否一致。
3) 图标测试：图标是否符合常规的表达习惯、图标的外形与实际功能是否相似。
4) 鼠标测试：在整个交互式语境中，是否可以识别鼠标操作、支持鼠标滑轮上下翻动操作。
5) 文字测试：文字是否零乱、是否拼写正确，是否易懂，不存在歧义性、是否表达了设计主题和构想意念。

2.3.3 性能测试

性能测试(Performance Testing)就是用来测试软件在集成系统中的运行性能的。性能测试的目标是度量系统相对于预定义目标的差距。性能测试必须要有工具支持，市面上有一些专门用于 GUI 或 Web 的性能测试工具，如 Loadrunner, SilkPerformer, WebLoad。

性能测试应该在功能测试之后,等功能稳定后再进行,否则功能方面缺陷会导致性能测试无法进行,另外功能缺陷的修复也可能使得之前的性能测试结果无意义。

1. 性能测试收集的信息

用户操作/事务处理的响应时间、WEB 服务器资源使用情况、CPU 使用情况 、内存使用情况、磁盘 I/O、……

2. 负载测试

负载测试是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，系统组成部分的相应输出项，例如通过量、响应时间、CPU 负载、内存使用等如何决定系统的性能。

3. 压力测试

压力测试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。通俗地讲，压力测试是为了发现在什么条件下应用程序的性能会变得不可接受。

4. 容量测试

容量测试(Volume Testing)的目的是使系统承受超额的数据容量来发现它是否能够正确处理。容量测试可以确定整个系统能够处理的业务容量，包括不同配置、不同场景下的最大用户数、最大处理数据记录数、最大处理文件容量。容量测试又称为大数据量测试。

容量测试是面向数据的，并且它的目的是显示系统可以处理目标内确定的数据容量。

容量测试的例子：

使用编译器编译一个极其庞大的源程序；

一个操作系统的任务队列被充满；

庞大的 Email 信息和文件充满了 Internet。

2.3.4 兼容性测试

兼容性测试是指所设计程序与硬件、软件之间的兼容性的测试。

兼容性测试分类主要可以分为：硬件兼容性测试、软件兼容性测试、数据库兼容性测试、数据兼容性测试。

1) **硬件兼容：**与整机兼容、与外设兼容

2) **软件兼容：**操作系统/平台、应用软件之间的兼容、不同浏览器的兼容

3) **数据库的兼容：**软硬件配合兼容，兼容不同类型数据库包括：MySQL、Oracle、MongoDB；

4) **数据兼容：**不同版本间的数据兼容、不同软件间的数据兼容

一般 Web 的兼容性测试考虑如下：操作系统平台兼容、浏览器兼容、分辨率兼容

1) 操作系统平台兼容

常见的操作系统有 windows、unix、linux 等，对于普通用户来讲，最常用的是 Windows 操作系统。Windows 操作系统包括 Windows XP，windows 2003，Windows7/8/10 等等，每个软件产品的兼容性测试是一个循序渐进的过程。

2) 浏览器兼容

主流浏览器有 IE,Firefox，Chrome、Opera、360 浏览器，搜狗浏览器等。

3) 分辨率兼容

分辨率的测试是为了页面在不同的分辨率下能正常显示，字体符合要求。现在常见的分辨率是 1920*1080、1280*1024、1024*768、800*600。对于常见的分辨率，测试必须保证测试通过，对于其他分辨率，原则上也应该尽量保证。

2.3.5 可用性测试

可用性测试(Usability Testing)是为了检测用户在理解和使用系统方面到底有多好。主要考虑产品是否符合实际应用情况，是否符合用户习惯或特殊要求，操作方式是否方便合理、设备和用户间的交互信息是否准确易于理解、是否遵从行业习惯、外观/界面是否美观等。应涉及到所有和用户有交互的功能或子系统。**一些测试人员应当关注的可用性问题：**

1) 可安装性测试

软件产品要易于安装，按照用户安装手册安装软件，评估安装过程的易用性和正确性。

2) 功能易用性测试

产品是否满足用户使用习惯，方便用户使用，用户体验较好；

- 业务符合性：主要测试软件是否符合其所服务的领域的业务逻辑。要求软件的界面风格、表格设计、业务流程、数据加密机制等必须符合相关的法律法规、业界标准规范以及使用人员的习惯。
- 功能定制性：为了适应用户需求的不断变化，软件功能应该能够灵活定制。如工资软件中部门结构和人员归属应可以灵活调整。
- 业务模块的集成度：在用户操作界面下，紧密关联的模块能够快捷的进行转换。
- 数据共享能力：测试软件数据库表的关联和数据重用问题，最大程度地减少用户的

重复输入，同时保证数据传递的一致性。

- 约束性：对于流程性比较强的业务操作，上一步操作完成以后，要强制进行下一步操作，这时需要软件以向导或与屏蔽无关操作的方式来限制用户的操作；另外，应以屏蔽或提示的方式阻止用户输入非法字符或进行损害数据和系统的操作，有效的避免用户犯错误，减少系统出现异常的概率，提高系统的安全可靠性。
- 交互性：主要测试用户操作的可见性和系统对用户的反馈。也就是用户的每一步操作都应有所回应或者提示，使用户清晰地看到系统的运行状态。
- 错误提示：测试软件在关键操作完成后或数据删除等操作前给出明确提示，操作错误或系统出现错误时，给出的出错信息中提供差错产生的原因，并指示如何进行正确的步骤，帮助用户从错误中恢复。

3) 辅助系统

一般来说，辅助系统包括在线帮助、向导和信息提示。

1. 向导测试：向导是否正确、向导是否存在、向导是否一致、向导是否用在固定处理流程中。
2. 信息提示：信息提示是否具有可以理解的语言进行描述，对重要的、有破坏性的命令是否提供确认措施；信息是否具有判断色彩，任何情况下，信息提示只能是引导和帮助用户，而不是指责用户。信息提示是否具有统一的标记标准的缩写和隐含的颜色。

4) 用户体验测试：

用户体验测试寻找评估目标界面中是否存在违反规则的情况的方法。：

a) 系统状态的可视性：

系统可视性原则是指系统必须在一定的时间内做出适当的反馈，必须把现在正在执行的内容通知给用户。例如：下载软件的进度条；浏览器的页面加载图标；软件安装时的安装进度条等。

b) 系统和现实的协调：

系统和现实的协调原则是指系统必须遵循现实中用户的习惯，用自然且符合逻辑的顺序来把系统信息反馈给用户。例如：Windows 系统中的回收站，做成垃圾箱的形式；电商网站上的购物车，浏览器上的前进、后退等

c) 用户操控与自由程度：

用户操控与自由程度原则是指用户因为误解了功能的含义而做出错误的操作，能够尽快解脱，比如“取消（Undo/Cancel）”和“重新提交（Redo）”的功能。

这个规则要求不应该给用户一种被计算机操控的感觉。因此，必须提供任何时候都能从当前状态跳出来的出口，保证能够及时取消或者再运行执行过的操作。例如：网站的所有网页都有能够跳转首页的链接；App 和网站的广告都有跳过或者关闭按钮；图片以缩放形式显示时，一定要做成点击后即可放大的模式；QQ 发送消息后在一定时间内还可以撤回等。

d) 一贯性和标准化

一贯性和标准化原则是指不应该让用户出现不同的词语、状况、行为是否意味着相同的意思这样的疑问。一般应该遵循平台的惯例。

这个规则要求保证用户在相同的操作下得到相同的结果。例如：同一网站内，网页设计的风格应该统一；指向网页的链接文本应该与网页的标题一致；未访问与已访问的链接的颜色要加以区分，以便识别等。

e） 防止错误

防止错误原则是指能一开始就防止错误发生的这种防患于未然的设计要比适当的错误消息更重要。

这个规则要求相比完善错误发生后的应对方案，更应该做的是预防出错。另外，在执行会带来重大影响的操作前，应该先弹出确认对话框，让用户再次确认是否执行该操作。例如：设置默认值；只能输入数字的地方，在输入时就过滤数字以外的字符；注册和修改密码时，密码需要输入两次；在表单的必填项上加上标记，使其更醒目；当前无权或者不应该操作的按钮，灰显；表单提交后，在服务器返回响应前，提交按钮不可点击，防止用户多次重复提交等。

2.3.6 安装卸载测试

软件产品能否进行正常的安装和卸载；包括，安装前后的检查，过程中的中断、异常；版本的升级；

现在软件系统越来越庞大，有可能使安装过程变得复杂，安装耗时越来越长，没有正确的安装就谈不上正确的使用。因此，可安装性测试是软件测试重要的一个环节。

系统的可安装性测试，主要是根据软件的测试特性列表、软件安装、配置文档，设计安装过程的测试用例，发现软件在安装过程中的错误。主要从以下几个方面进行测试：

1. **安装前的检查**
   - 检查安装文档是否齐全。
   - 检查被测试软件的安装文件是否齐全。
   - 检查软件的文件格式是否与安装指导中要求的文件格式相符。
2. **安装手册评估。**安装前，需要检查安装手册或用户文档中的安装说明。一般来说，安装手册需要对安装平台、安装过程需要注意的问题以及手动配置的部分进行详细说明。
3. **安装的自动化程度测试。**由于制作安装程序的软件很多，许多软件采用了自动安装方式。但由于部分软件的特殊性，有时必须采用一定的手动配置来完成安装。因此，测试软件安装过程的自动化程度，一般来说，软件安装尽可能做到“全自动化”，即使在不得已的情况下需要进行手动配置，也需要采取一些措施，比如选择框方式等，使手动配置变得简单和明确。
4. **安装选项和设置的测试。**在安装的过程中常常需要对安装的项目进行选择，比如推荐安装、完全安装、自定义安装等，也可能要设置不同的信息，比如：安装路径、安装向导、缺省设置等，安装测试时需要对不同的选项和设置方案进行测试，论证各种方案是否能够安装成功。
5. **安装过程的中断测试。**一个大型的软件有可能需要很长时间来进行安装，如果因为断电、文件冲突或读写错误导致安装过程的非正常中断，有可能使已进行的安装前功尽弃。一个好的自动化安装程序应该能够记忆安装的过程，当恢复安装时，安装程序能够自动进行检测，并从“断点”进行安装。

6. **安装过程的异常测试。**在安装测试过程中要设计异常的安装测试用例，包括配置参数的异常、安装选项和安装路径的异常等。
7. **安装顺序测试。**对于大多数应用系统，特别是分别式系统，常常需要安装软件系统的不同组成部分。不同的安装顺序会导致安装失败，或引起一些不可预料的错误，例如，先安装客户端后安装服务器，会导致某些软件的客户端与服务器连接不上。因此，如果安装手册中未明确指出安装顺序，则需要测试不同顺序的安装过程。
8. **多环境安装测试。**不同的应用环境下安装的情况也是不一样的。因此，测试时需要在主测试环境和辅测试环境中都进行安装测试。
9. **安装的正确性测试。**在上述安装测试后，还需要简单的使用软件功能以验证安装的正确性，同时，还要考察对其他应用程序的影响。
10. **安装后的检查**
    - 所有文件是否都已产生并确认有所需要的内容；
    - 程序文件的目录及子目录是否正确产生；
    - 各目录及子目录下的程序文件是否都正确产生；
    - 是否存在无用的目录、子目录、程序文件和无用的临时文件；
    - 程序文件的目录及子目录、以及程序文件本身的权限是否正确；
    - 对于 PC windows 下的软件，还要检查与应用软件相配套的动态联接库文件是否齐全；
    - 安装日志的检查。
11. **修复安装与卸载测试。**修复安装主要是在软件使用后，根据需要添加或删除软件的一些组件或者修复受损软件。修复安装和卸载应该是自动化的。修复、安装、卸载是一个完整安装程序中的不同选项。进行修复安装时，需要检查修复对软件有无不良影响。软件卸载后需要检查软件的关键信息或敏感信息是否已被自动清除。
12. **安装测试中的软件升级测试。**软件通过重新安装来达到升级目的，具体测试方法同上。同时要进行版本兼容性的测试。可通过 Patch 的方式实现软件的升级。

2.3.7 安全性测试

产品在使用过程具备数据的备份和恢复能力、权限的访问控制能力防止非法入侵，通信过程中数据加密能力，敏感信息的脱敏处理能力等。

安全测试(Security Testing)用来验证集成在系统内的保护机制是否能够在实际中保护系统不受到非法的侵入。用来保证系统本身数据的完整性和保密性。如当受到恶意攻击时，设备的自我保护能力，病毒防护能力，自定义通信协议安全性等。广义的还包括物理安全性测试、业务安全性测试、网络安全性测试等等。

我们这里讲的安全性主要是针对软件产品来说的，软件产品的安全性与功能、易用性有较大的关联，是通过某些功能的实现来体现的。针对软件产品的安全性测试内容，一般可以从以下方面考虑：

1） 测试软件是否具有数据的备份和恢复功能

为了防范系统的意外崩溃造成的数据丢失，备份与恢复手段是一个应用系统的必备功能。备份与恢复根据应用系统对安全性的要求可以采用多种手段，如数据库增量备份、数据库完全备份、系统完全备份等。出于更高的安全性要求，某些实时系统经常会采用双机热备或多

机热备。除了对于这些备份与恢复方式进行验证测试以外，还要评估这种备份与恢复方式是否满足应用系统的安全性需求。

2） 用户管理和访问控制

1. 用户权限控制

对于应用软件来说，用户权限的控制是比较重要的。一个用户能够访问一个应用系统的权限主要来源于三个方面：应用软件本身、操作系统和数据库。

应用软件在在开发过程中，主要采用用户名和密码登录的方式完成。安全强度高的软件也可采用指纹认证、智能卡认证等方式进行。

⑴ 用户名称的测试：用户名称应该具有唯一性。

① 同时存在的用户名称在不考虑大小写的状态下，不能够同名；

② 对于关键领域的软件产品和安全性要求较高的软件，应当同时保证使用过的用户在用户删除或停用后，保留该用户记录，并且新用户不得与该用户同名。

⑵ 用户口令测试：用户口令应该满足当前流行的控制模式，主要测试口令的强度、口令存储的位置和加密强度。

① 最大口令时效：指定用户可以保留当前口令的时间；

② 最小口令时效：指定用户修改口令之前，用户必须保留口令的最少时间；

③ 口令历史：确定系统将要记住的口令数量，如果用户选择的口令存在于口令数据库中，系统将强制用户选择其他口令；

④ 最小口令长度：对于用户口令可以包含的最少的可以接受的；

⑤ 口令复杂度：在口令中要求用户使用非字母数字的字符或大小写字母；

⑥ 加密选项：可以加密本地存储的口令；

⑦ 口令锁定：在输入非法口令达到规定的次数之后，系统将禁用这个帐户；

⑧ 账户复位：账户锁定后定义是否可以在规定的时间间隔后自动恢复。

2. 操作系统安全性测试

⑴ 是否关闭或卸载了不必要的服务和程序；

⑵ 是否存在不必要的帐户；

⑶ 权限设置是否合理；

⑷ 安装相应的安全补丁程序的情况；

⑸ 操作系统日志管理。

3. 数据库权限测试

⑴ 数据库管理用户的设置应当注意对帐户的保护，超级用户的口令不得为空或默认口令，对数据库的帐号和组的权限应作相应设置；

⑵ 数据库中关于应用软件用户权限和口令存储的相关表格，尽量采用加密算法进行加密；

⑶ 根据不同的程序访问数据库的功能，使用不同的数据库用户进行连接，并且必须设置密码。

3） 通信加密

通信加密是保证数据在传输过程中数据的保密性和一致性。软件产品在技术上通常使用链路加密、数据加密的方式进行。目前，使用的加密技术包括 VPN 技术、对称加密算法、非

对称加密算法、HASH 算法。

测试主要按照客户设计要求所使用的加密技术，采用验证的方式进行测试。

4) **安全日志测试**

安全日志是软件产品被动防范的措施，但也是重要的防范功能。

⑴ 日志应当记录所有用户访问系统的操作内容，包括登录用户名称、登录时间、浏览数据动作、修改数据动作、删除数据动作、退出时间等；

⑵ 日志的完整性、正确性；

⑶ 大型应用软件，系统是否提供了安全日志的统计分析能力。

2.3.8 异常测试

产品在使用中通过人工干预的手段产生软硬件的异常，检查系统是否具备容错，排错和自动恢复能力；

概念：系统异常测试又叫系统容错和可恢复性测试，它是通过人工干预手段使系统产生软、硬件异常，通过验证系统异常前后的功能和运行状态，达到检验系统的容错、排错和恢复的能力。它是系统可靠性评价的重要手段。

容错处理：系统自动处理、人工干预处理

注意：

系统异常测试还与系统的指标测试有关系，当系统需要提供的服务能力大于系统的设计指标时，也属于系统异常的情况，因此应该结合起来加以考虑。

系统的可靠性是设计出来的，而不是测试出来的。测试出来的数据有助于为我们进行进一步的系统优化设计积累经验，设计和测试是一个互为反馈的过程

2.3.9 网络测试

产品在使用过程中，在不同的网络环境下，包括数据网络，WiFi，弱网下，及网络间切换过程中处理能力，能否正常使用；

概念：网络测试是在网络环境下和其他设备对接，进行系统功能、性能与指标方面的测试，保证设备对接正常。数据网络、WiFi 网络、弱网

内容：网络测试考察系统的处理能力、系统兼容性、系统稳定可靠性及用户使用等方面。如通信产品，主要进行协议测试：

- 一致性测试：检测所实现的系统与协议规范符合程度
- 性能测试：检测协议实体或系统的性能指标（数据传输率、联接时间，执行速度、吞吐量、并发度等）
- 互操作性测试：检测同一协议不同实现厂商之间，同一协议不同实现版本之间、或同一类协议不同实现版本之间互通能力和互连操作能力
- 坚固性测试：检测协议实体或系统在各种恶劣环境下运行的能力（信道被切断、通信设备掉电、注入干扰报文等）

2.3.10 文档测试

在项目中我们主要接触到的文档有开发文档、测试文档、管理文档和用户文档。通常我们所说的文档测试主要是针对用户文档的测试。

为了使用户了解软件的使用、操作和对软件进行维护，软件开发者为用户提供的详细资料，称为用户文档，包括用户手册、操作手册、维护手册。

用户文档测试项目如下：

1. 读者群：用户文档面向的读者要明确。对于初级用户，需要从鼠标的用法、点击、确定等讲起。对于中级用户，重要界面的截图、关键步骤、每一个参数的选择方法都需要介绍，对于高级用户，则没有必要给出太多的界面截图，但对重要参数的讲解一定要深入，用词要专业。
2. 术语：用户文档中用到的术语要适应定位的读者群。用法一致，标准定义与业界规范相吻合。如果有索引或交叉引用，所有的术语都应该能够进行索引和引用。如果术语较多，在用户手册的末尾应给出术语的索引。如果被测的软件提供二次开发功能，则有必要编写开发指南。
3. 正确性：测试过程中，检查所有信息是否真实正确，检查所有的目录、索引和章节引用是否已更新，尝试链接是否正确。产品支持电话、地址、邮政编码等信息是否正确。
4. 完整性：不作任何假设，完全根据提示操作，对照软件界面检查是否有重要的分支没有描述到。
5. 一致性：根据文档描述的操作执行后，检查软件返回的结果是否与文档的描述相同，软件界面上出现的版本号与手册、帮助上的信息是否一致。
6. 易用性：检查用户文档是否条理清晰、结构合理，对关键的步骤是否以粗体或背景色给用户提示。文档对用户看到的错误信息应当有更详细的文档解释，采用合理的页面布局、适量的图表描述，提高文档的易用性。
7. 图表与界面截图：检查所有的图表与截面截图是否与发行版本相同，注意图表标题的正确性。
8. 样例和示例：像用户一样载入和使用样例，以每一个模版制作文件，确认他们的正确性。
9. 语言：检查是否有二义性的描述。
10. 印刷与包装：检查用户文档的印刷与包装质量，力求为用户提供精美、实用的用户手册。

2.3.11 可靠性测试

软件系统可靠性是指软件在规定的条件下和规定的时间内完成规定的功能的能力。软件可靠性测试是对软件产品的可靠性进行调查、分析和评价的一个过程。

软件的可靠性度是指在规定的时间内，规定的条件下，软件不发生失效的概率。

系统可靠性的指标有：系统平均失效时间间隔（MTBF）、系统平均恢复时间（MTTR）

测试软件在系统（包括硬件、要求的软件及属于该产品的程序）运行过程中，不应出现用户无法控制的状态，即不应崩溃也不应丢失数据。即使出现下列情况也应满足上述要求：

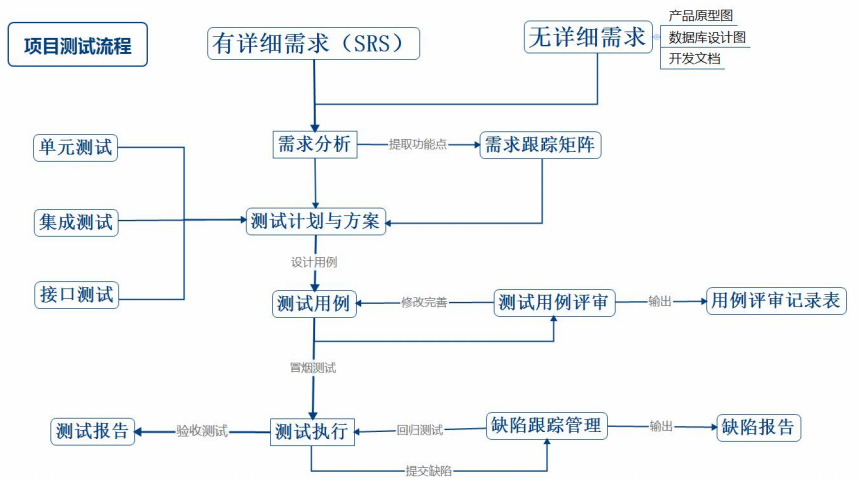
1) 使用的容量达到规定的极限。
2) 企图使用的容量超出规定的极限。
3) 其他程序或用户造成的错误输入。
4) 使用用户手册中明确规定的非法指令。

2.3.12 稳定性测试

系统稳定性测试目的是评价系统在一定负荷情况下、长时间的运行情况。包括系统在一定负荷下，再增加新的业务，原有的业务是否受影响，新的业务是否能正常工作，系统资源有无泄漏，数据有无不一致的情况，系统性能是否会降下来，关键点是长时间的运行后，系统的状况如何，系统平均无故障时间 MTBF 是否满足系统设计要求。（不变的负载；变化的负载）

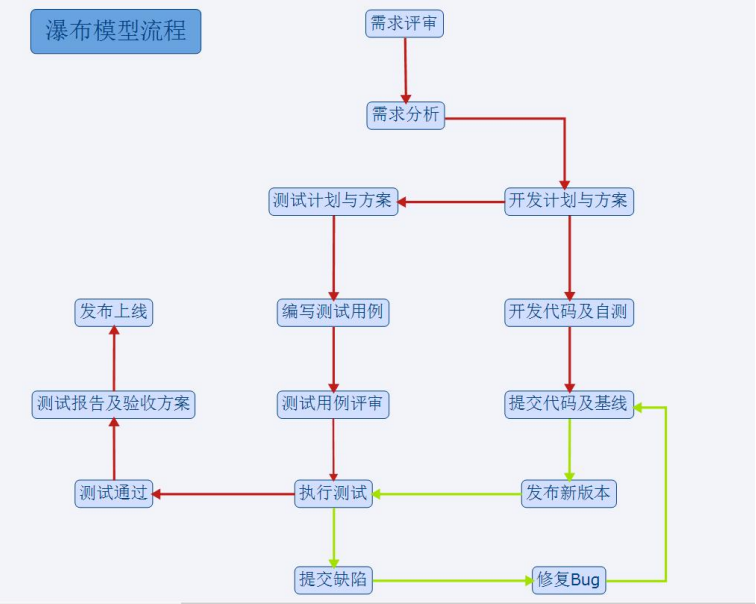
2.4 系统测试流程

2.4.1 项目测试流程



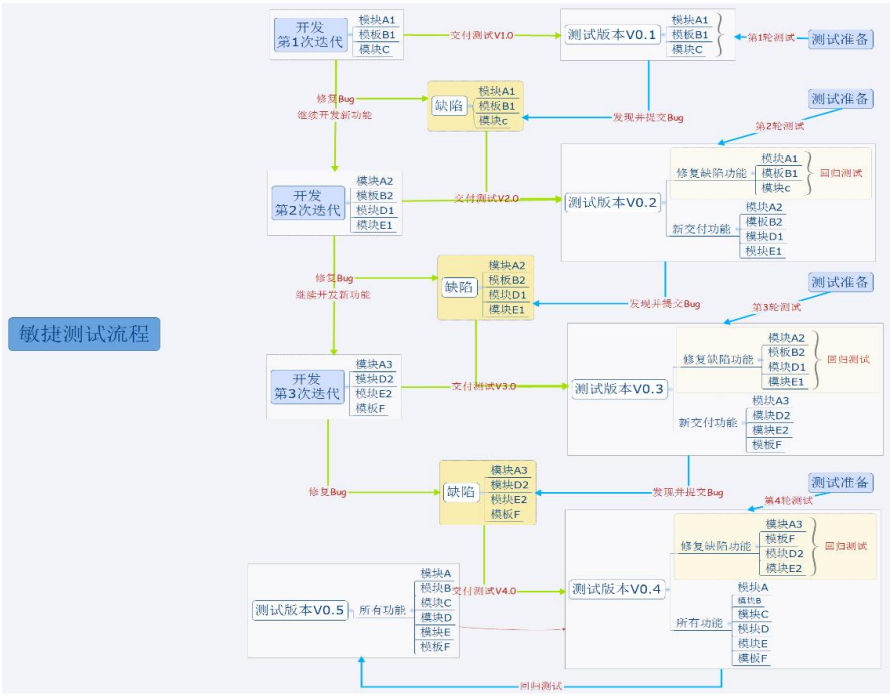
1) **测试需求分析：**在拿到项目后要先进行测试需求分析，如果有详细的需求文档 SRS，就根据它进行测试需求分析，提取功能点和测试输出功能矩阵；如果没有详细的需求，则我们就根据产品原型图、或者开发的开发文档及数据库设计图等来进行测试需求分析；
2) **测试计划方案：**在测试需求分析之后就开始制定软件测试计划与方案，测试计划与方案要包括单元测试、集成测试和系统测试。
3) **用例设计评审：**在计划和方案制定完成后，就开始设计测试用例，用例设计完成后要进行用例的评审，评审过程中要输出一份用例评审记录表。评审若存在问题，则要进行用例的修改和完善，并再次评审，直到评审通过；
4) **测试执行：**接下来要进行测试执行，但在测试执行之前要先做一次冒烟测试，冒烟通过后才正式进行系统测试执行。测试执行中发现问题要提交缺陷并进行跟踪管理，输出缺陷报告单。缺陷修复完成后还要进行回归测试，若回归测试过程中发现问题还要继续提交并修复，直到回归测试通过。最后在进行验收测试，通过后提交测试报告。到此完整的项目测试流程就结束了。

2.4.2 传统测试流程



1) 原始需求 ->产品需求分析(PM/产品经理)-> PRD -> 需求评审(相关人员)-> SRS ；SRS -> 开发/测试需求分析 ->开发/测试计划方案；
2) 开发计划 ->代码编写->开发自测/UT ->提交代码 ->代码评审形成基线->发布新版本；
3) 测试计划 -> 编写测试用例->用例评审->执行新版本测试；
4) 测试发现问题->提交缺陷->开发修复缺陷->提交修复代码并评审->评审通过发布新版->在新版本上进回归测试；
5) 测试回归通过->进行验收测试->提交测试报告->发布上线；

2.4.3 敏捷测试流程



1) 在第一迭代交付前，开发人员对第一次迭代的功能进行开发，测试人员则进行第一次迭代的测试用例等相关的测试准备工作；开发在第一次迭代开发完成后发布一个测试版本供测试人员进行测试；
2) 在测试人员进行第一个版本的测试过程中，开发人员则继续进行第二次功能的迭代开发，并同时修复测试人员提交的上一版本的缺陷，然后将修复后的缺陷及第二次迭代开发新功能一同发布到新版本交付测试；
3) 重复上面的迭代过程，当所有新功能开发完成后，则进行缺陷的修复和回归测试，直至所有缺陷修复完成，则发布最终版本交付用户；