软件测试与质量知识点整理

1、软件测试概述

1、 软件测试基本思想

（1） 软件生存周期：

软件生命周期一般包括以下阶段：软件计划与可行性研究（问题定义、可行性研究）、需求分析、软件设计（概要设计与详细设计）、编码、软件测试、运行与维护

（2） 软件测试的技术与过程

软件测试的过程包括以下阶段：测试设计、测试自动化、测试执行、测试评估

测试设计：

1） Criterial Based：设计测试数值去满足覆盖规则或者其他工程性目标
软件测试中最具技巧性的工作
需要的知识：离散数学、编程、测试
往往需要软件工程知识背景

2） Human Bases：基于程序的领域知识和测试的人工知识设计测试数据
基于规则的（Criteria-based）方法会忽略特殊情况，
需要的知识：域知识、测试技能和用户接口
几乎不需要传统的软件工程知识背景

测试自动化：将测试数据写入可运行脚本
需要很少的理论知识基础，对技术的要求不高，需要较低层次的编程技术即可
测试执行：在软件上运行测试并且记录结果
如果测试自动化程度比较高，这将是一个简单而又繁琐的工作。
需要基本的计算机技能：实习生、没有软件工程知识背景的人员
图形用户界面并不是很容易实现自动化，这需要许多人工劳作
测试执行者必须非常谨慎小心地对运行结果进行记录
测试评估：评估测试的结果，这比它看上去要难得多
需要的知识：领域知识、测试知识、用户接口和心理学
通常情况下几乎不需要软件工程（SE）学历：软件的领域（ domain ）背景很重要；有相关的经验背景是很有用的（生理学、心理学…）；具有逻辑很好的逻辑思维对胜任这项工作很有帮助（法律、哲学、数学）

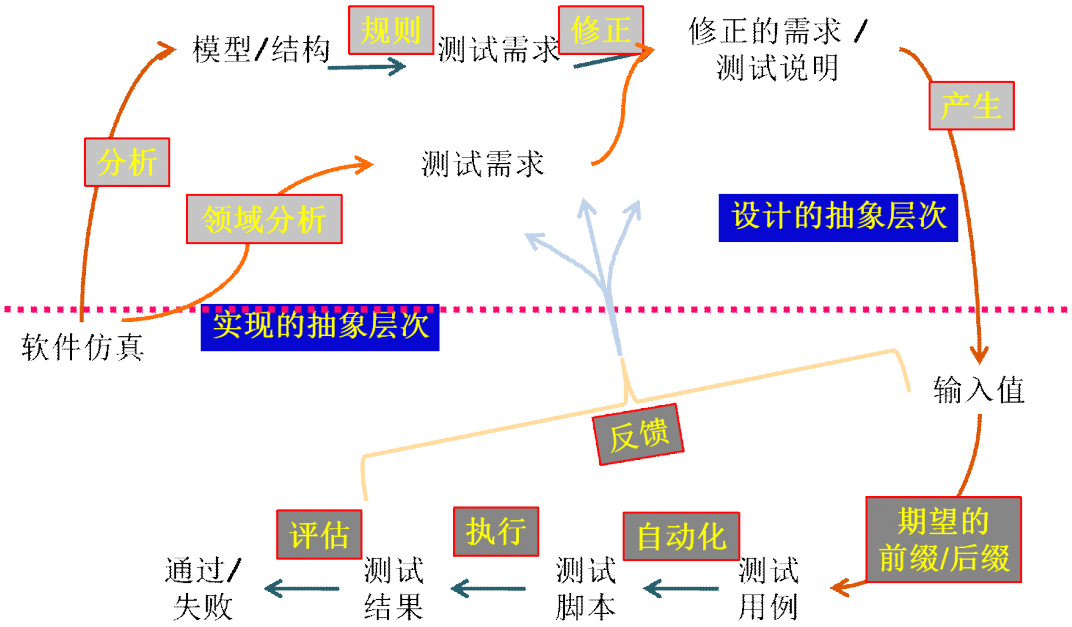
其他活动：

1） 测试管理：制定策略，组织团队，开发之间的接口，选择测试规则，决定所需要的自动化程度

2） 测试维护：保存测试用例以供软件衍化时的复用，需要测试设计人员和自动化人员的合作，决定何时整理测试套件既需要策略又需要技巧，测试必须纳入配置管理

3） 测试文档：需要各方参与，每个测试都需记录“为什么要这么做”---规则和满足测试需求或者人工设计的测试基本原理，保证整个过程可追溯，在自动化的测试中做记录

模型驱动式软件测试：



（3） 持续的软件测试

不同级别的测试：

1） 单元测试: 单独测试每一个单元（方法）
2） 模块测试: 测试每一个类、文件、模块或组件
3） 集成测试: 测试各个模块如何交互
4） 系统测试: 测试系统的总体功能性
5） 验收测试: 这个软件是否能够被用户所接受

测试活动：

1） 软件需求：定义测试目标（规则）、定义计划
2） 系统设计：设计系统测试、设计验收测试、设计可用性测试（如果合适的话）
3） 中层设计：将系统测试具体化、制定集成测试和单元测试的计划、获取测试的支持工具、确定类集成的顺序
4） 详细设计：构建测试或者将测试具体化
5） 实现：创建测试用例、当单元构成完成后运行测试
6） 集成：运行集成测试
7） 系统部署：运行系统测试、验收测试、可用性测试
8） 使用和维护：收集用户的问题、进行回归测试

2、 软件测试中的若干问题

（1） 重要的术语辨析：

1） Validation & Verification：

Validation（确认）：在软件开发末期评估软件以确保与先前需求相符合的过程

Verification（验证）：判断软件某一开发阶段的产品是否满足了在前面阶段建立的需求

2） 测试 & 质量保障(QA)：

测试的目的是寻找 bug，尽早地发现他们确保他们已经被修正。

质量保障（ QA ）的目的是创建一个强制执行的标准和方法去提高开发过

程以便了防止 Bug 的出现。

3） 静态测试&动态测试：

静态测试 ： 不实际运行程序的测试，这包括代码审查和一些形式的分析，在发现某些特定类型的问题上很有效---特别是“潜在”的缺陷，即在程序被改动时会引发的问题

动态测试: 用实际输入执行程序的测试

4） Faults, Errors & Failures：

Software Fault : 软件代码中的一个静态缺陷

Software Failure : 相对于需求或者其他队软件行为的描述而言，外部的不正确的行为

Software Error : 由缺陷导致的内部的不正确的状态

5） 测试 & 调试：

测试 ： 寻找使得软件出错的输入

调试 ： 根据 failure 寻找缺陷（fault）的过程

6） Fault & Failure 模型，缺陷被发现的 3 个必要条件：

可达性（reachability） ： 包含缺陷的代码地址必须在软件运行时刻到达

可感染（Infection） ： 必须出现程序的错误状态

可波及（Propagation）: 被感染的状态必须可以导致一些输出的错误

7） 测试用例：

测试用例的数值: 直接满足测试需求的数据

可预测的结果 ： 当软件满足预计的行为时，执行程序所产生的输出

8） 可观察性&可控制性：

软件可观察性（Software Observability） ： 以其输出、对环境以及其他硬件软件部分的作用作为考量观察程序的容易程度。对设备，数据库或者远程文件有影响的软件具有较低的可观察性

软件可控制性（Software Controllability） ： 以数值、操作和行为为考量，为程序提供输入的容易程度。通过键盘很容易对软件进行控制；通过硬件传感器或者远程的分布式软件输入会难一点；数据抽象会降低可观察性和可控制性

9） 对可控制性和可观察新具有影响的输入：

Prefix Values : 将软件切换至接受测试用例数据的正确状态的任何必要的输入。有两类 postfix values，确认值（Verification Values） ： 查看测试用例结果的必要输入之；退出命令（Exit Commands） ： 终止程序或者将其返回到稳定状态时所需要输入的数据

Postfix Values : 在测试用例数据之后必须被传到程序之中的数据

可运行测试脚本（Executable Test Script） :一个以在被测试软件上自动执行并且输出结果的形式编写的测试用例

10） 黑盒测试&白盒测试：

黑盒测试：通过外部描述（包括规格说明、需求、设计）而产生的测试

白盒测试：通过软件的内部源代码（具体包括分支、独立条件和语句）而产生的测试

基于模型的测试：从元件模型产生的测试（例如，UML 图）

11） 自顶向下测试&自底向上测试：

自顶向下（Top-Down Testing） ： 测试主干（main procedure），然后向下

贯穿其所调用的过程，以此类推

自底向上（Bottom-Up Testing）：先测试调用树的叶节点（没有调用其它过程的部分），然后逐渐向上直至根节点。每个过程不会被测试除非它的所有子节点都被测试过。

（2） 测试的级别：

Level 0：测试与调试没有什么区别；

Level 1：测试的目的是为了展示正确性

Level 2：测试的目的是为了使软件崩溃

Level 3：测试的目的不是为了验证任何具体的细节而是降低使用软件的风险

Level 4：测试是一个心理的规约用来帮助 IT 专家开发更高质量的软件

（3） 如何优化测试：

1） 测试人员需要更多更好的软件工具

2） 测试人员需要采用更高效率更具影响的测试实践和技巧

3） 测试/质量保障小组需要更多的技巧专业知识

4） 测试/质量保障小组需要更加特化

（4） 四个主要的障碍：

1） 缺乏测试的相关知识培训

2） 改变程序代码的必要性

3） 工具的使用 Usability of tools

4） 功能较弱、效率较低的工具

（5） 软件测试的前景：

1） 逐渐增长的测试特化趋势会导致更加高效和更具影响的测试

2） 测试和质量保障小组会有更多的技巧专门技能

3） 开发人员会更加了解测试，并且有把测试做得更好的动力

4） 灵巧的过程使得测试被放在了第一位---要求开发人员和测试人员去做出更好的测试

5） 测试和安全会开始合并

6） 我们会开发出用来测试基于软件的系统中关系的方法

2、 边界值测试

1、 边界值测试

（1） 人们从长期的测试工作经验得知，大量的错误是发生在输入或输出范围的边界上，而不是在输入范围的内部。

（2） 针对各种边界情况设计测试用例，可以查出更多的错误。

（3） 使用边界值分析方法设计测试用例，首先应确定边界情况。

（4） 通常输入等价类与输出等价类的边界，就是应着重测试的边界情况。

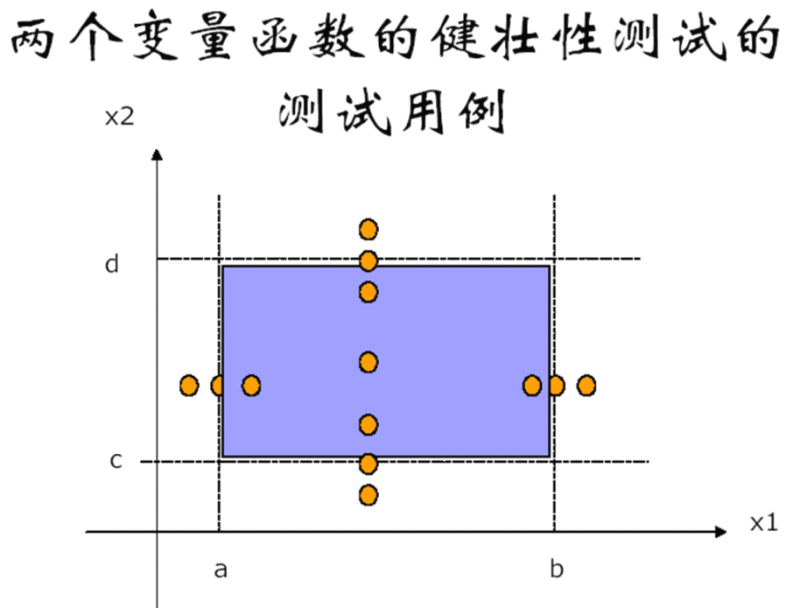
（5） 应当选取正好等于，刚刚大于，或刚刚小于边界的值做为测试数据，而不是选取等价类中的典型值或任意值做为测试数据。

2、 健壮性测试：

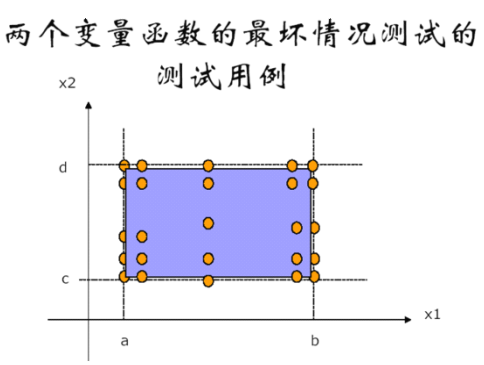
（1） 健壮性测试，又称为容错性测试，用于测试系统在出现故障时，是否能够自动恢复或者忽略故障继续运行。

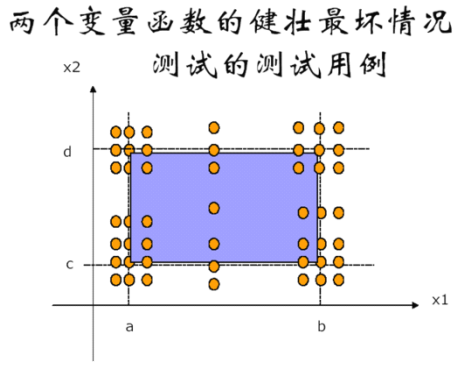
（2） 为了使系统具有良好的健壮性，要求设计人员在做系统设计时必须周密细致，尤其要注意妥善地进行系统异常的处理。

（3） 考虑到了软件使用过程中的非法输入值



3、 最坏情况测试





4、 特殊值测试

在测试中考虑到特殊的输入值和特殊情况

5、 随机测试

（1） 随机测试是根据测试说明书执行样例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。随机测试主要是对被测软件的一些重要功能进行复测，也包括测试那些当前的测试样例没有覆盖到的部分。

（2） 对于软件更新和新增加的功能要重点测试。重点对一些特殊点情况点、特殊的使用环境、并发性、进行检查。尤其对以前测试发现的重大 Bug，进行再次测试，可以结合回归测试一起进行。

（3） 理论上，每一个被测软件版本都需要执行随机测试，尤其对于最后的将要发布的版本更要重视随机测试。

（4） 随机测试最好由具有丰富测试经验的熟悉被测软件的测试人员进行测试。对于被测试的软件越熟悉，执行随机测试越容易。

6、 边界值测试方针

（1） 如果输入条件规定了值的范围,则应取刚达到这个范围的边界的值,以及刚刚超越这个范围边界的值作为测试输入数据。

（2） 如果输入条件规定了值的个数,则用最大个数,最小个数,比最小个数少一,比最大个数多一的数作为测试数据。

（3） 应用于输出条件，即设计测试用例使输出值达到边界值及其左右的值。

（4） 如果程序的规格说明给出的输入域或输出域是有序集合,则应选取集合的第一个元素和最后一个元素作为测试用例。

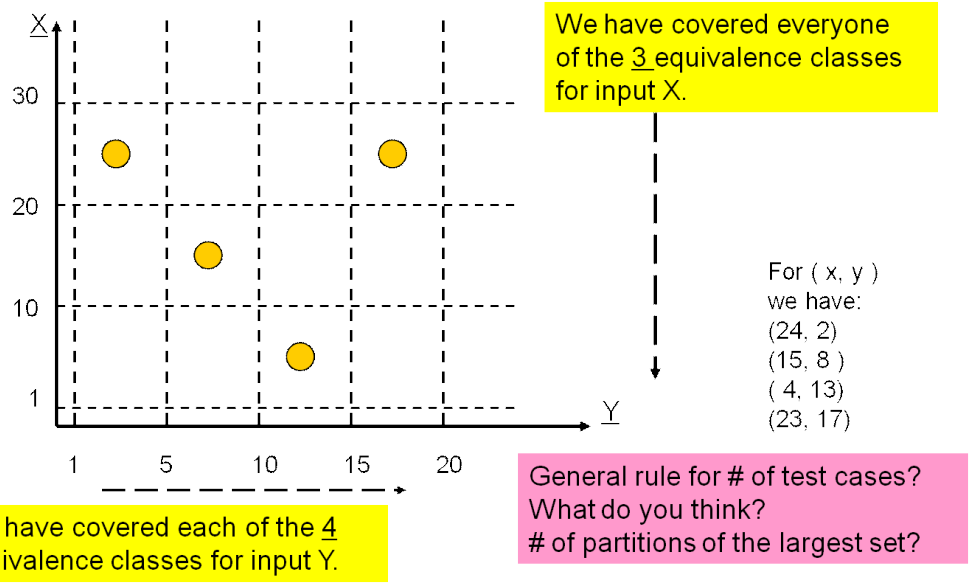
（5） 如果程序中使用了一个内部数据结构,则应当选择这个内部数据结构的边界上的值作为测试用例。

3、等价类测试

1、 等价类

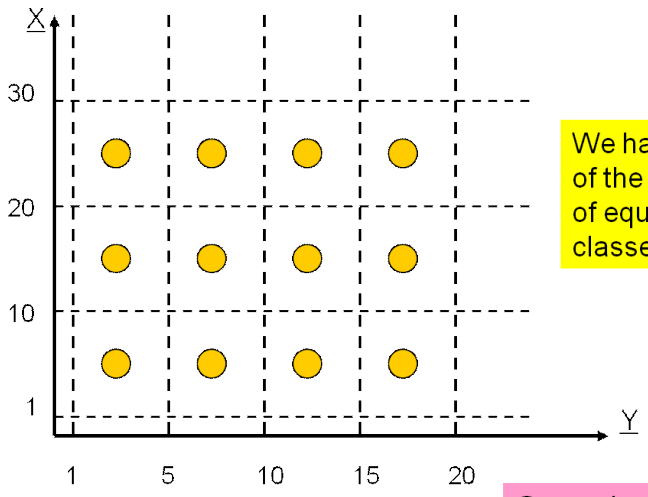
（1） 弱一般等价类测试：假设只有一个缺陷或者输入变量相互独立

Assume the equivalence partitioning of input X is: 1 to 10; 11 to 20, 21 to 30
and the equivalence partitioning of input Y is: 1 to 5; 6 to 10; 11;15; and 16 to 20



（2） 强一般等价类测试：与弱一般等价类相似，但是它关注多种缺陷因素和变量之间的依赖，各种变量的每个等价类组合都需要被包括

Assume the equivalence partitioning of input X is: 1 to 10; 11 to 20, 21 to 30 and the equivalence partitioning of input Y is: 1 to 5; 6 to 10; 11;15; and 16 to 20

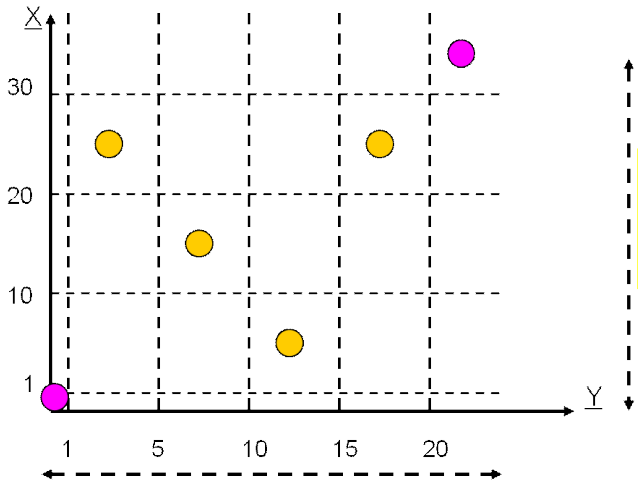


We have covered everyone of the 3 x 4 Cartesian product of equivalence classes

General rule for # of test cases? What do you think?

（3） 弱健壮等价类测试：与弱一般等价类测试相似，但是考虑了输入的非法值

Assume the equivalence partitioning of input X is 1 to 10; 11 to 20, 21 to 30 and the equivalence partitioning of input Y is 1 to 5; 6 to 10; 11;15; and 16 to 20

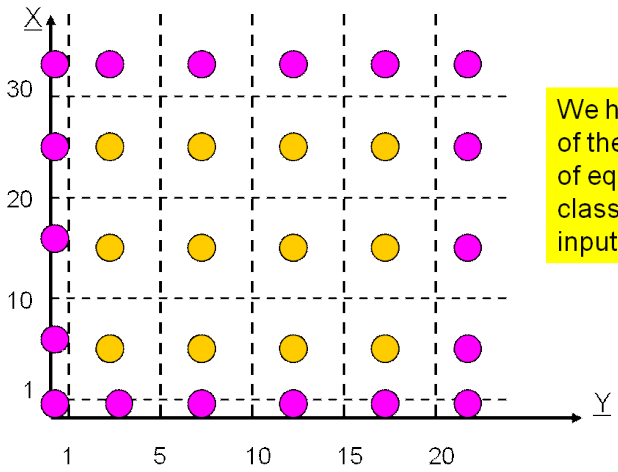


We have covered everyone of the 5 equivalence classes for input X.

We have covered each of the 6 equivalence classes for input Y.

（4） 强健壮等价类测试：与强一般等价类测试类似，但是考虑了输入的非法值

Assume the equivalence partitioning of input X is: 1 to 10; 11 to 20, 21 to 30 and the equivalence partitioning of input Y is: 1 to 5; 6 to 10; 11;15; and 16 to 20



We have covered everyone of the 5 x 6 Cartesian product of equivalence classes (including invalid inputs)

2、 一般基于等价类的测试方法

（1） 按照规格说明中的“输入条件”（或者输出条件）划分等价类
确定等价类的原则：有效等价类、无效等价类

（2） 设计一个新的测试用例，使其尽可能多的覆盖尚未覆盖的有效等价类；重复这一步骤，直到所有的有效等价类都被覆盖为止

（3） 设计一个新的测试用例，使其仅覆盖一个无效等价类；重复这一步骤，直到所有的无效等价类都被覆盖为止

3、 等价类划分方法

（1） 如果输入条件规定了取值范围，可定义一个有效等价类和两个无效等价类。

（2） 如规定了输入数据的一组值，且程序对不同输入值做不同处理，则每个允许的输入值是一个有效等价类，并有一个无效等价类(所有不允许的输入值的集合)。

（3） 如果规定了输入数据的个数，则类似地可以划分出一个有效等价类和两个无效等价类。

（4） 如果规定了输入数据必须遵循的规则，可确定一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则)。

4、 输入域的两种建模方法

（1） 基于接口/输入的建模方法

1） 根据独立的输入参数来生成特点，最简单的建模方法，在某些情况下可以被部分自动化生成。

2） 一些语义知识和领域知识不会被用到；忽略了参数之间的关系。

（2） 基于功能的建模方法

1） 根据程序在测试中的行为视角生成特点，在生成测试用例的时候更困难，需要花费更多的精力，会有更好的测试，或者在同样有效的条件下更少的用例。

2） 可以使语义知识和领域知识相互结合，可以考虑到参数之间的关系，建模基于功能而非实现，同样的参数可以出现在不同的特征中并且很难将数值转化为测试用例。

4、静态测试

1、 静态测试技术：

定义：不执行程序代码而寻找代码中可能存在的错误或评估程序代码的过程。静态测试可以手工进行，也可以借助软件工具自动进行。

特点：静态测试不必动态的执行程序，也就是不必进行测试用例设计和结果判读等工作；静态测试可以由人手工方式进行，充分发挥人的优势，行之有效；静态测试实施不需要特别条件，容易开展。

（1） 代码审查：

测试内容：检查代码和设计的一致性；检查代码对标准的遵循、可读性；检查代码的逻辑表达的正确性；检查代码结构的合理性。

组成和方式：代码审查由一组程序和错误检查技术组成，以代码审查组方式组织。代码审查组一般为 4 个人，其中一人为组长，还包括资深程序员、程序编写者与专职测试人员

步骤：准备、程序阅读、审查会议、跟踪及报告

以第三方测试的方式进行代码审查：应就发现的缺陷及错误与软件开发人员讨论；避免由于理解不一致产生问题，形成共同认可的审查结果

（2） 代码走查：

代码走查与代码审查相似，它也是由一组程序和错误检查技术组成，只是程序和错误检查技术不完全相同。

组成和方式：代码走查以小组方式进行。代码走查组包括组长、秘书（负责记录）、测试人员。走查过程与审查相似。

走查会议内容：与代码审查不同，不是读程序和使用代码审查单，而是由被指定的作为测试员的小组成员提供若干测试用例，让参加会的成员当计算机，在会议上对每个测试用例用头脑来执行程序，也就是用测试用例沿程序逻辑走一遍，并由测试人员讲述程序执行过程，在纸上或黑板上监视程序状态。如果发现问题由秘书记下来，中间不讨论任何纠错问题，主要是发现错误

缺点：代码走查使用测试用例启发检测错误，人们注意力会相对集中在随测试用例游历的程序逻辑路径上，不如代码审查检查的范围广，错误覆盖面全。

2、 静态测试的内容

（1） 需求定义的静态测试：对需求定义的测试着重于测试对用户需求的描述和解释是否完整、准确

对照条例：兼容性、完备性、一致性、正确性、可行性、易修改性、健壮性、易追溯性、易理解性、易测试性和可验证性

（2） 设计文档的静态测试：对设计文档的静态测试着重于分析设计是否与需求定义一致，所采用的数值方法和算法是否适用于待解问题，程序的设计中对程序的划分是否与待解问题相适应，需求是否都被满足了等等

对照条例：完备性、一致性、正确性、可行性、易修改性、模块性、可预测性、结构化、易追溯性、易理解性、可验证性/易测试性

（3） 源代码的静态测试：对源代码的静态测试着重于分析实现是否正确、完备

对照条例：完备性、一致性、正确性、易修改性、可预测性、健壮性、结构化、易追溯性、易理解性、可验证性

5、结构性测试-控制流测试

1、 语句覆盖 SC

（1） **定义：**设计若干测试用例，运行被测程序，使程序中每个可执行语句至少执行一次

（2） **优点：**可以很直观地从源代码得到测试用例，无须细分每条判定表达式。

（3） **缺点：**由于这种测试方法仅仅针对程序逻辑中显式存在的语句，但对于隐藏的条件是无法测试的。如在多分支的逻辑运算中无法全面的考虑。语句覆盖是最弱的逻辑覆盖。

2、 判定覆盖 DC

（1） **定义：**设计若干测试用例，运行被测程序，使得程序中每个分支的取真值和取假值至少一次，即判断真假值均曾被满足。

（2） **优点：**判定覆盖具有比语句覆盖更强的测试能力。同样判定覆盖也具有和语句覆盖一样的简单性，无须细分每个判定就可以得到测试用例。

（3） **缺点：**往往大部分的判定语句是由多个逻辑条件组合而成，若仅仅判断其整个最终结果，而忽略每个条件的取值情况，必然会遗漏部分测试路径。判定覆盖仍是弱的逻辑覆盖。

3、 条件覆盖 CC

（1） **定义：**设计若干测试用例，执行被测程序以后要使每个判断中每个条件的可能取值至少满足一次。

（2） **优点：**增加了对条件判定情况的测试，增加了测试路径。

（3） **缺点：**条件覆盖不一定包含判定覆盖。条件覆盖只能保证每个条件至少有一次为真，而不考虑所有的判定结果。

4、 条件判定组合覆盖 CDC

（1） **定义：**设计足够的测试用例，使得判断条件中的所有条件可能至少执行一次取值，同时，所有判断的可能结果至少执行一次。

（2） **测试用例要满足以下条件：**

1） 所有条件可能至少执行一次取值；

2） 所有判断的可能结果至少执行一次。

（3） **优点：**能同时满足判定、条件两种覆盖标准

（4） **缺点：**判定/条件覆盖准则的缺点是未考虑条件的组合情况。

5、 多条件覆盖 MCC

（1） **定义：**设计足够的测试用例，使得所有可能的条件取值组合至少执行一次，又称条件组合覆盖

（2） **优点：**条件组合覆盖准则满足判定覆盖、条件覆盖和判定/条件覆盖准则。

（3） **缺点：**线性地增加了测试用例的数量。

6、 修正条件判定覆盖 MCDC

（1） **定义：**要求在一个程序中每一种输入输出至少得出现一次，在程序中的每一个条件必须产生所有可能的输出结果至少一次，并且每一个判定中的每一个条件必须能够独立影响一个判定的输出，即在其他条件不变的前提下仅改变这个条件的值，而使判定结果改变.

（2） **条件的含义：**不含有布尔操作符号的布尔表达式

（3） **判定的含义：**判定表示由条件和零或者很多布尔操作符号所组成的一个布尔表

达式

（4） **优点：**MC/DC 测试错误的准确率却是特别地高，因此 MC/DC 适合那些大型的并且要求测试非常精确的软件测试所用

（5） **缺点：**在测试时为了寻找出测试集合所花费的时间多，由此而引起的人力、物力、财力三重开销也会直接攀升；并且有可能存在一些条件在任何情况下都不独立，使得 MC/DC 的要求没法实现

7、 路径覆盖

（1） **定义：**设计所有的测试用例，来覆盖程序中的所有可能的执行路径

（2） **特点：**在以路径为特点的软件程序代码中的起点和终点之间经常会有许多可能路径。每一个决策都会使潜在的路径数量变成原先的两倍；每一个 Case 语句都会使潜在的路径数量变为原先的数量乘以 Case 的分支数量；每一个循环都会使潜在的路径数量变为原先数量乘以循环中迭代器可能值的个数那么多[Beizer-90]

（3） **优点：**这种测试方法可以对程序进行彻底的测试，比前面几种的覆盖面都广。

（4） **缺点：**需要设计大量、复杂的测试用例，使得工作量呈指数级增长，不见得把所有的条件组合都覆盖。

8、 基本路径测试方法

（1） 绘制程序的控制流图

（2） 计算 McCabe 圈复杂度（设为 n）

V（G）= e – n + 2 = d + 1

（3） 确定基本路径集的确定，即构造 n 条独立路径

1） 任意构造一条从（唯一）入口结点到（唯一）出口结点的路径，将该路径加入基本路径集

2） 修改基本路径集中路径，至少经过一条以前未走过的边，将新路径加入基本路径集

3） 重复第(2)步，直到基本路径集中包含 n 条路径

（4） 设计测试用例，使基本路径集中的路径能走通

6、集成测试

将经过单元测试的模块按照设计要求连接起来，组成所规定的软件系统的过程称为“集成”。集成测试又称组装测试、联合测试、部件测试。

1、 集成测试的必要性：实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。由于在接口部分存在问题，程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。

主要任务：

将各模块连接起来，检查模块相互调用时，数据经过接口是否丢失；

将各个子功能组合起来，检查能否达到预期要求的各项功能；

一个模块的功能是否会对另一个模块的功能产生不利的影响；

全局数据结构是否有问题，会不会被异常修改；

单个模块的误差积累起来，是否被放大，从而达到不可接受的程度。

驱动模块：用以模拟被测模块的上级模块。接受测试数据，把相关的数据传送给被测模块，启动被测模块，并获得相应的结果。

桩模块：用以模拟被测模块工作过程中所调用的模块。由被测模块调用，一般只进行很

少的数据处理，例如打印入口和返回，以便于检验被测模块与其下级模块的接口

与单元测试和系统测试的比较

测试类型	对象	目的	测试依据	测试方法
单元测试	局部模块	消除局部模块内的逻辑和功能上的错误和缺陷	模块逻辑设计，外部说明	白盒为主
集成测试	模块间的集成和调用关系	找出与设计相关的程序结构、模块调用关系、模块间接口方面的问题	程序结构设计	白盒+黑盒，采用较多黑盒方法构造测试用例
系统测试	整个系统(包括硬件等)	对整个系统进行一系列的整体、有效性测试	系统结构设计、需求规格说明等	黑盒

2、 集成测试的方法

（1） 逐步集成（增量式集成）

自顶向下增量式测试：按结构图自上而下逐步集成和逐步测试；首先集成主控模块（主程序），然后按照软件控制层次结构向下进行集成。

3） 步骤：1、以主控模块作为测试驱动器，把对主控模块进行单元测试时引入的被调用模拟子模块用实际模块替代；2、依照所选用的模块集成策略，下层的被调用模拟子模块一次一个地被替换为真正的模块；3、在每个模块被集成时，都必须立即进行一遍测试。

4） 优点：测试和集成可以较早开始；减少驱动器的开发；底层接口修改灵活

5） 缺点：桩模块的开发代价较大；底层模块无法意料的需求可能促使顶层修改；底层模块测试可能不充分

6） 适用范围：增量式开发；并行软件开发

自底向上增量式测试：从最底层的模块开始，按结构图自下而上逐步进行集成和测试。

1） 步骤：1、从最底层的模块开始，按结构图自下而上逐步进行集成和测试；2、依照所选用的模块集成策略，将驱动模块(Driver) 用实际模块代替；3、在每个模块被集成时，都必须立即进行一遍测试。

2） 优点：无需构造桩模块，桩模块往往千差万别；驱动模块具有某种统一性，且随着测试层次的提供，驱动模块数量减少；涉及复杂算法和真正输入、输出的模块一般在底层，且是较易出错的模块。可以尽早发现错误；各子树的集成和测试可以并行

3） 缺点：驱动模块开发量大；高层模块的可操作性和互操作性测试不充分

4） 适用范围：重要模块在底层

混合集成（三明治集成）：以中间层为目标层，以上用自顶向下，以下用自底向上。

1） 优点：它将自顶向下和自底向上的集成方法有机地结合起来，不需要写桩程序因为在测试初自底向上集成已经验证了底层模块的正确性

2） 缺点：在真正集成之前每一个独立的模块没有完全测试过。

3） 改进的三明治集成方法：不仅自两头向中间集成，而且保证每个模块得到单独的测试，使测试进行得比较彻底

（2） 一次性集成

优点：迅速完成集成测试；测试用例较少

缺点：错误难以定位；即使通过测试，许多接口错误也可能隐藏

适用范围：小的、良构的系统，其模块已经接受了充分测试；一个已经存在的系统，只有少量修改；通过复用可信赖的模块构造系统

（3） 非增量式测试 VS 增量式测试：

1） 非增量式测试是先分散测试，然后集中起来再一次完成集成测试。如果在模块的接口处存在错误，只会在最后的集成测试时一下暴露出来；很难确定出错的真正位置、所在的模块、错误的原因。这种方法适合在规模较小的应用系统中使用。

2） 增量式测试的逐步集成和逐步测试的方法，把可能出现的差错分散暴露出来，便于找出问题和修改。模块在逐步继承的测试中得到了较为频繁的考验，因而可能取得较好的测试效果。

3） 总的来说，增量式测试比非增量式测试具有一定的优越性。

（4） 集成测试原则：

1） 所有公共接口都要被测试到

2） 关键模块必须进行充分的测试（对应几条需求；具有高层控制功能；复杂，易出错；有特殊的性能要求）

3） 集成测试应当按一定的层次进行

4） 集成测试的策略选择应当综合考虑质量、成本和进度之间的关系

5） 集成测试应当尽早开始，并以总体设计为基础

6） 在模块与接口的划分上，测试人员应当和开发人员进行充分的沟通

7） 当接口发生修改时，涉及的相关接口必须进行再测试

8） 测试执行结果应当如实记录

（5） 集成测试的技术：

1） 集成测试主要测试软件的结构问题，因为测试建立在模块的接口上，所以多为黑盒测试，适当辅以白盒测试。有时又把集成测试称作灰盒测试。

2） 集成测试一般也不使用真实数据，测试人员可以使用手工制作一部分代表性的测试数据。在创建测试数据时，应保证数据充分测试软件系统的边界条件。

3） 在集成测试时可适当地重用单元测试时生成的数据。

（6） 集成测试分析：

1） 体系结构分析：确定基本模块的大小，合理划分测试的模块，为测试策略的确定提供依据

2） 模块分析：划分关键模块（基本模块、共享模块等）；分清高危模块、一般模块和低危模块

3） 接口分析：集成测试的重点是接口的功能性、可靠性、安全性、完整性、稳定性等

4） 可测试性分析：一般可测试性会随集成范围的增加而下降；尽早评估可测试性、有利于集成测试的可操作性分析

5） 集成策略分析：分析测试策略的优劣；好的测试策略应该是测试充分、总成本低

（7） 集成的依据：功能分解树、函数调用图、类关系图、进程调用图、网络服务调用图、UML 协作图、UML 顺序图

（8） 集成测试步骤：制定集成测试计划、集成测试分析和设计（确定测试需求、确定集成策略、评估测试风险、确定测试优先级、确定测试方法、集成测试代码

设计、集成测试用例设计、集成测试工具和资源）、集成测试实现、集成测试执行、集成测试评估

7、构造测试用例

1、 集成测试用例设计：

（1） 为正向测试设计用例（正向：接口、功能的正确性；输入输出域分析、等价分类法、状态转换）；

（2） 为逆向测试设计用例（逆向：接口、功能的错误、异常、多余、遗漏）

（3） 不执行其不应该完成的工作（错误猜测、边界值、状态转换、基于风险）

2、 其他测试用例设计：参考 PPT 中的例子

5、 系统和外部测试

1、 系统测试的目的：系统测是的目的是验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。

2、 系统测试的各种类型：比较常见的、典型的系统测试包括恢复测试、安全测试、压力测试。

（1） 恢复测试：恢复测试作为一种系统测试，主要关注导致软件运行失败的各种条件，并验证其恢复过程能否正确执行。在特定情况下，系统需具备容错能力。另外，系统失效必须在规定时间段内被更正，否则将会导致严重的经济损失。

（2） 安全测试：安全测试用来验证系统内部的保护机制，以防止非法侵入。在安全测试中，测试人员扮演试图侵入系统的角色，采用各种办法试图突破防线。因此系统安全设计的准则时要想方设法使侵入系统所需的代价更加昂贵。

（3） 压力测试：压力测试是指在正常资源下使用异常的访问量、频率或数据量来执行系统。在压力测试中可执行以下测试：

1） 如果平均中断数量是每秒一到两次，那么设计特殊的测试用例产生每秒十次中断。

2） 输入数据量增加一个量级，确定输入功能将如何响应。

3） 在虚拟操作系统下，产生需要最大内存量或其它资源的测试用例，或产生需要过量磁盘存储的数据。

3、 α和β测试：

（1） α测试：α测试是指软件开发公司组织内部人员模拟各类用户对即将面市软件产品（称为α版本）进行测试，试图发现错误并修正。α测试的关键在于尽可能逼真地模拟实际运行环境和用户对软件产品的操作并尽最大努力涵盖所有可能的用户操作方式

（2） β测试：β测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用β版本，并要求用户报告异常情况、提出批评意见，然后软件开发公司再对β版本进行改错和完善。β测试也是黑盒测试。黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。

6、 专项测试活动

1、 配置和兼容性测试

（1） **性能测试包括：**负载测试、压力测试、吞吐量测试、尖峰突击测试、配置测试、隔离测试

（2） **配置式测试的定义：**配置式测试是检查被测软件在各种硬件设备上的操作情况的过程，一个标准的商用或者家用的 PC 机有许多种可能的配置

（3） **如何确定是配置 Bug:** 最准确的方法就是在另外一个配置完全不同的机器上一步一步执行产生 Bug 的操作，如果这种问题没有重现，那么很有可能是配置 Bug，否则，有可能是普通缺陷

（4） **配置 Bug 的种类：**

1） 软件有可能会有在很多种配置上都会出现的 Bug

2） 软件的 Bug 有可能只出现在某一种特定的配置中

3） 硬件设备或者它的驱动本身的错误，只在当前被测的软件运行时被揭示出来

4） 硬件设备或者它的驱动本身的错误，有可能在很多其他软件中被发现

（5） **配置式测试的步骤：**

1） 确定所需要的硬件

2） 确定现在能够得到的硬件品牌、模型和驱动设备

3） 确定所有可能用到的硬件特点、模式和选择（features, modes, and options）

4） 将所有确认过的硬件配置缩减到可管理的数量

5） 标记出软件在特定的配置下独特的风格

6） 设计在各个配置下运行的测试用例

7） 在每种配置下执行测试用例

8） 重复运行测试用例直到结果符合要求

（6） **获取硬件的方法：**

1） 只够买经常时候或者将经常使用的配置

2） 联系硬件生产厂家，问其是否能够借用或者免费赠予配置

3） 发送备忘录或者邮件询问办公室或者家里有什么样的硬件，是否介意在这些设备上运行几个测试

4） 如果预算允许，可以和项目经理一起联系一个专门负责配置兼容测试的实验室，将这一部分外包出去

（7） **识别硬件标准：**了解一些硬件设备的细节可以帮助自己更好地做等价划分的决策

（8） **配置测试其他硬件：**硬件、软件以及它们所连接的东西并不重要；如果它连接到其他设备，配置需要被测试

（9） **兼容性测试定义：**检测被测软件能否与其他软件正确地交互和共享信息

（10） **兼容性测试需要考虑的问题：**被测软件需要和哪些其他软件相互兼容？定义被测软件和其他软件交互的兼容性标准和指导方针是什么？被测软件与其他软件或者平台之间共享信息的数据类型是什么？

（11） **平台和应用的版本：**选择目标平台或者兼容的应用实际上是一个项目管理或者市场任务，他们会识别软件需要兼容的版本

1） 向后兼容/向前兼容：如果一个软件被称为向后/向前兼容，那么就说明这个软件可以与之前/之后的版本一起工作

2） 多版本测试的影响：我们不能测试在操作系统中成百上千的软件，所以我们只挑选重要的软件进行测试

（12） OATS（正交矩阵测试策略，orthogonal matrix testingstrategy）

（13） 标准和指导方针：

1） 高层标准和指导方针：知道产品的总体运行情况、外观和使用体验、支持的风格等等

2） 底层标准是本质细节，比如文件格式和网络交换协议，我们应将底层标准看作是软件具体细节的拓展

（14） 数据共享兼容性：一个支持遵循标准并且允许用户容易地发送信息到其他软件或者从其他软件获取信息的程序，是具有很高兼容性的产品（文件的保存、加载、导入、导出、复制、剪切、粘贴）

（15） 兼容性测试环境：

1） 使用可移除的硬盘驱动（HDD）和分区工具

2） 使用驱动映像工具（ghost）为配置创建一个映像文件

3） 虚拟机

在目标平台上的安装方法也很重要，必须和最终用户所用的一样

2、 外国语言测试

（1） 国际化 i18N (internationalization)：

1） 定义：是设计一个可以不用在工程上进行改变便可适用于各个语言和地区的软件应用的过程。国际化是软件开发者的任务。

2） 国际化软件的特点：

1、通过添加本地化的数据，这个软件可以在全球各个地方运行；

2、文本元素，如状态信息和图形用户界面组件的 label，不是通过硬编码实现的，而是保存在源代码之外可以动态提取的；

3、添加新的语言支持不需要重新编译

4、与文化相关的数据，如日期等，以符合某个语言和地区的格式类型出现

5、可以很快地被本地化

3） 如何使软件国际化：

1、 对于网页来说，所有需要本地化的字符串都应由 tag 括起来

2、 对于二进制的客户端，所有的 GUI 字符串都应保存在资源文件中，而不是通过硬编码写在程序代码中

4） 语言翻译的问题：

1、 热键和快捷方式

2、 拓展的字符

3、 字符数量的估算

4、 从左向右阅读还是从右向左阅读

5、 图形中的文本

6、 保留代码中的文本

5） 网页中的国际化中的问题：

1、 布局和用户接口

2、 无序的代码和混乱的文本

3、 字符索引

4、 欧亚的全名差异

（2） 本地化 L10N (localization)：

1）定义：将一个国际化的软件翻译和调整到某个特定的语言和文化的过程 。本地化需要实现翻译文本，修改 GUI、声音和图片，测试产品

2）总的来说，国际化被认为是一个工程的过程，而本地化被认为是一个翻译的过程。

3）本地化的问题：

1、 内容（Contents）：内容是除了源代码以外，进入产品的其他“资料”。我们需要考虑组成最终产品的所有成分

2、 数据格式：不同的地区使用不同的格式来组织数据单元。所以在测试的过程中，我们需要对测量的单元很熟悉

（3） 全球化 G11N (Globalization)：

1）定义：是一个包括了国际化和本地化的广义的称呼，G11N = I18N + L10N

（4） G11N 测试：为了确定软件的全球化程度

（5） 测试包含的内容：国际化测试、本地化测试、语言/翻译测试、用户界面测试、功能测试、更多的功能测试、发布测试

3、 易用性测试：

（1） 易用性的含义：易被发现、易于学习、易于使用、易于获得

（2） 用户界面/易用性测试（UI/Usability Testing）：

1） UI 被称作用户界面：从用户那里获取输入、将输出反馈给用户

2） 可用性反映了软件交互的合理性、功能性和有效性

3） 一个好的 UI 决定了一个软件的易用性

4） GUI 需要经过可用性测试

5） 易用性测试的主要部分便是用户界面的测试

（3） 好的用户界面所具备的品质：遵循标准和指导、满足直觉特征、正确性、一致性、灵活性、使用舒适、易于使用、简单

1）遵循标准和指导：

1、如果没有很好的不遵循标准的理由，那么最好遵循已有的标准；

2、经过大量的测试、尝试和错误，已经产生了许多衡量一个软件是否用户友好的标准；

3、可以为自己的软件单独制定标准

2）直觉：

1、 用户界面简洁；

2、 良好的布局；

3、 功能之间易于切换；

4、 没有过度堆砌的功能

5、 帮助系统真的能够起到帮助作用

3）正确性：市场差别、语言和拼写、媒介问题、所见即所得

4）一致性：

1、 快捷键和菜单选项

2、 使用正确的术语和命名、

3、 音频

4、 键盘、按钮等价和布局

5）灵活性：状态的跳转、状态的终止和跳过、数据的输入输出

6）使用舒适：界面的合理性、错误处理、性能

7）简单

（4） **残障人员对软件的使用：**法律已经有相关条例规定，在软件设计过程中需要考虑

4、 文档测试

（1） **系统文档与用户文档**

文档的两种读者：在软件产品周期中负责维护系统的信息系统管理员、在日常生活中使用系统的用户

1） **系统文档：**系统的设计规格、内部原理和其功能的详细描述

2） **用户文档：**关于应用系统的文字或者图像描述，讲解系统如何工作、如何使用

（2） **软件文档的类型：**

1） **文档类型：**用户手册、操作手册、系统概述指导、教程和自动化系统介绍、其他系统文档

2） **用户帮助和问题处理手册：**错误信息参考索引、在线帮助、快速参考指导、安装和启动指示

3） **其他文档：**包装文字和图片；商标、广告和其他信息；担保/注册信息；最终用户许可协议（EULA）；标志；示例和模板

4） **示例-用户手册的主要内容：**

1、 **系统概述：**系统的设计目标；功能和能力；风格、特征和优势，包括关于系统实现了什么的清晰描述

2、 **功能描述：**

1） 主要功能的概括图，以及各个功能之间如何联系

2） 每个功能的屏幕展示、每个功能的目的和每个菜单选项和功能键选项的执行结果

3） 每个功能的期望输入

4） 每个功能的实际输出

5） 每个功能可以引发的特殊情况

5） **示例-错误信息参考索引内容：**

1、 当错误发生时，所运行的代码部分的名字

2、 所执行部分的代码行号

3、 错误的严重性和对系统的影响

4、 任何相关的系统内存和数据指针（如注册表和栈指针）的内容

5、 故障的属性，或者故障的编号

（3） **文档测试的重要性**

1） **好的文档在以下三个方面提高了软件的质量：**提高可用性、提高可靠性、提供更多的支持

2） 为了发现文档的错误，文档检查很重要

（4） **阅读文档的要点：**

1） **优秀文档的编写：**

1、 **好的用户文档包括：**

1） 一个不常见术语的对照表

2） 错误信息、问题处理、和恢复信息

3） 关键问题索引

4） 详细的内容列表

2、 **首先编写一个能够概括整个文档的提纲，保证所有的关键功能被说明**

3、 **包含基于任务的文档描述：**

1）“How to’s…”
2）常见问题列表
3）系统信息和含义
4）示例

4、使用易理解的简单句式

2）文档测试准则：

1、文档测试是为了确保：正确性、完整性、可理解性
2、文档测试是判断信息是否与模型逻辑的规格同步

3）文档测试问题对照表：

1、总体考虑方面：读者、术语、内容和主题
2、正确性：反应事实、步骤具体清晰、图标和截屏、示例、拼写与语法

（5）文档测试的现状：

使文档编写和测试与软件开发产生差异的原因：

1）文档通常受到较少的关注、有较少的预算和资源
2）文档编写人员通常不是软件方面的专家
3）纸质文档的产生需要时间

5、安全性测试

（1）入侵电脑的动机：为了挑战和赢得别人的崇拜；出于好奇；为了使用别人的计算机设备；破坏别人的系统（3-D：defacing, destruction, denial of service - DoS）；窃取信息。

（2）威胁建模：

1、步骤：

1）组建威胁建模小组
2）标识所发现有讨论价值的东西
3）创建一个总体的架构
4）将应用分解
5）将威胁标识、评级、文档化

2、评估标准：潜在风险、可复现性、可利用性、对用户的影响、可发现性

（3）测试的要点：

1、身份认证（Authentication）：登录、超时、修改密码、上限/下限、存储加密、旁路捕获 URL、超时探测处理、
2、完备性（Integrity）：对篡改/欺骗的的防护
3、私密性（Privacy）：防止窃听
4、可靠性（Non-Repudiation）：提升可靠度
5、可用性（Availability）：RAID、簇、冗余备份（RAID,clusters,cold standbys）

（4）关键术语：

1、SERVERS: web, app, database server
2、CLIENT: browser, other apps, components
3、Cookies
4、Open Systems Interconnect
5、Protocols
6、NETWORK

1）Router Tools: Lancope StealthWatch
2）Network Scanning Tools

7、 DMZ

8、 VPN

9、 WEB Vulnerabilities

10、 Host/Network Identification

11、 Viruses and Worms

12、 Password Cracking

13、 Valid Remote Apps vs Rogue

6、 Web 站点测试

（1） 网页基础知识：

1、 网页的组成部分：布局、导引框架、相关文件的链接、具体内容（文字、图像、声音、媒体）

2、 网站是由很多文件组成的：

1） 主要的文件是.htm 文件（这是放有代码、连接、具体内容的框架）

2） 每个图像都是被.htm 文件调用的文件

3） 每个链接与其他文件连接

（2） Web 站点测试知识：

1、 **测试的主要内容：**

1） 网页的内容：文字的拼写；不同的字体、颜色、字号；图形、图片、界面

2） 功能性：连接、按钮、导航条；登录、登出、会话、 cookie；页面之间的逻辑关系

3） 可用性

4） 安全性

5） 性能

2、 网站中的技术：

1） HTML/DHTML/XML

2） JavaScript, Java

3） VBScript

4） ActiveX, Plug-in

5） Perl, CGI

6） PHP/ASP/JSP

（3） 黑盒测试

测试的关注点：文本和替代文本（ATL text）；超链接；图片；对象和其他各项功能

（4） 灰盒测试：黑盒测试和白盒测试的有效结合，按照黑盒测试的方法测试软件，但是在测试过程中会参考部分使被测部分工作的代码（知识部分查看代码而非全部查看）

（5） 白盒测试：

条件：要求掌握一些网站系统结构的知识；需要掌握一些编程知识

测试关注点：动态内容、数据库驱动网页、编程生成的网页、服务器性能和负载、安全性

（6） 配置与兼容测试

1） 目的：为了确保网站在各种操作系统平台和各种浏览器配置可以正常运行；确保特殊的代码，如 as Active X, Java, Javascript 和 CGI 可以在不同的配置上

正常运行

2） 硬件平台：不同的操作系统、不同的浏览器配置、不同的连接速度、不同的浏览器选项、不同的 Java 虚拟机版本、不同的 SSL 代理（SSL w/t proxy, SSL w/o proxy）、不同的显示器解析（滚动条、字体）

3） 兼容性测试：向前兼容、向后兼容

4） 网站测试要点：

1、 需要运行测试确保站点已网络化（networkable）；

2、 产品同样需要在不同的操作系统下运行测试，以确保其能够在不同的操作系统下正常运行；

3、 需要进行压力测试来确保一定用户数量条件下，站点能够正常运行

（7） 可用性测试：侧重于网页的人机交互方面内容

1、 网页设计中的十个最常见的错误：

1） 最前沿技术的滥用

2） 带有滚动条的文本框、一直运行的动画效果

3） 页面过长、有一个长长的滚动条

4） 非标准颜色的链接

5） 过时的信息

6） 下载时间过长

7） 缺少导航指示

8） 幽灵页面（即没有链接指向的、永远访问不到的页面）

9） 复杂的 URL

10） 过度使用框架

2、 设计中的专注点：页宽、颜色、字体、页长（滚动条）、导航（导航条、按钮、路径图）、页面命名

（8） 自动化介绍

1、 合理的站点测试：

1） 互联网和内联网测试

2） 检查坏掉的连接、结构、等等

3） 可以使用 Site check 将网站发布到目标站点上

2、 GUI 录制和回放：捕捉键盘输入及其他输入，在测试运行时作出反馈，将实际输出与预期输出对比，并且在录制过程中自动生成脚本

工具：Visual Test Suite、Visual Test、TeamTest、XRunner,、WinRunner、QARun、SilkTest、Robot

3、 负载测试：目标是要求成千上万的用户同时访问并且与网站交互

工具: WebLoad（通过不同的脚本可以模拟出用户访问进行压力测试和负载测试）

4、 **测试工具：（参考 PPT STQ12）**