

# 第十讲：WEB应用测试技术

Lian Yu  
The School of Software and Microelectronics  
Peking University  
No.24 Jinyuan RD, Beijing 102600

# 提 纲

- ➡ ■ 概述
  - Web应用测试概念
  - 测试过程概述
  - 内容测试
  - 用户界面测试
  - 组件级测试
  - 导航测试
  - 配置测试
  - 安全测试
  - 性能测试
  - 总结

## 概述-什么是WEB应用测试？

- Web应用测试是相关活动的集合，是为了发现存在于Web应用的内容、功能、可用性、导航、性能、容量、安全性方面的错误。
- 为了完成这些活动，在Web工程中使用包括静态评审和动态执行测试在内的测试策略。

## 概述-由谁来负责WEB应用测试？

- **Web**项目的工程师和其它与项目有关的涉众（管理者，客户，最终用户）都要参与**Web**应用测试。

# 概述-为什么WEB应用测试如此重要？

- 如果最终用户碰到错误并动摇他们对Web应用的信心，他们就要去其它地方寻找所需要的内容和功能，这样这个Web应用就失败了。
- 所以在Web应用上线之前，Web工程师必须努力消除尽可能多的错误。

## 概述- WEB应用测试步骤是什么？

- Web应用测试开始集中于用户可见的Web应用方面，然后测试技术与基础设施。
- 执行七个测试步骤：
  - 内容测试、界面测试、导航测试、组件测试、配置测试、性能测试、和安全性测试。

# 概述-有哪些工件形成？

- 在一些情况下，会生成Web应用测试计划。
- 在每一种情况下，要为每个测试步骤生成一组测试用例并将测试结果存档以便将来软件维护所有。

# 概述- 如何确保我们准确地完成了任务？

- 尽管永远不能保证你已经执行了所要求的每一个测试，可你能肯定测试已经发现了错误（并且已修正了这些错误）。
- 另外，如果已经制定了一个测试计划，你可以检查以保证所有计划测试已被完成。



# 提纲

- ➡ ■ 概述
  - Web应用测试概念
  - 测试过程概述
  - 内容测试
  - 用户界面测试
  - 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结

## WEB应用测试概念

- 测试是为了发现软件的错误(并最终修正错误)而运行软件的过程。
  - 对Web应用来说，这些最基本的原则不会变。
- 为了理解在Web工程中测试的目标，必须考虑Web应用质量的多个纬度。

## 质量的纬度

- 良好设计体现在**Web**应用所带来的质量。
- **Web**应用质量的评估是通过技术评审和测试。
  - 应用一系列的技术评审去评估设计模型的各个组成部分。
  - 并应用本章讨论的测试过程去评价**Web**应用实现。

## 质量的纬度

- 内容（**content**）是在句法和语义级别的评估。在句法级，要评估基于文本的文档中的拼写、标点、语法等；在语义级，要评估信息表现的正确性、整个内容对象和相关对象的一致性、无二义性。
- 功能（**function**）测试是要发现与客户需求不符的错误。每个**Web**应用功能要评估其正确性、不稳定性（**instability**）、与实现标准的符合性（比如，**Java**或者**XML**语言标准）。
- 结构（**structure**）评估是要确保正确发布了**Web**应用的内容和功能，并且是可扩展的，要支持新内容和新功能的增加。

## 质量的纬度

- 可用性（**usability**）测试是要确保对每一类用户都要有相应的界面支持；用户要能学习和应用所有需要的导航句法和语义。
- 导航（**navigability**）测试要确保所有的导航句法和语义都被测试，从而发现有关导航的任何错误（比如死链接、不合适的链接、错误的链接）。
- 性能（**performance**）测试确保在各种操作条件、配置、负载变化的情况下，系统在响应用户交互和处理极限负载时，性能没有出现不可接受的退化。

## 质量的纬度

- 兼容性（**compatibility**）测试通过在服务器和客户端不同的配置情况下，执行**Web**应用。目的是发现跟某种配置相关联的特殊错误。
- 互操作性（**interoperability**）测试是确保**Web**应用能正确地与其它应用或数据库进行交互。
- 安全（**security**）测试是评估潜在的易受攻击的弱点并尽量发现这些弱点。任何成功的渗透企图意味着安全性上的漏洞。

# WEB应用环境中的错误

1. 因为对很多种**Web**应用测试而言，首先在客户端发现问题出现的证据（比如，通过在一个特定浏览器或**PDA**或手机上实现的接口），所以**Web**工程师看到的是一个错误的症状，不是错误本身。
2. 因为**Web**应用是在许多不同的配置和不同的环境中实现的，所以发生在某个**Web**环境中的错误可能很困难或者不可能在该环境之外重现。
3. 尽管一些错误是由不正确的设计或不合适的**HTML**（或其它程序语言）代码造成的，但很多错误都能被追踪到**Web**应用配置。
4. 因为**Web**应用存在于客户端/服务器（**C/S**）体系结构中，所以很难跨越客户端、服务器、网络三个结构层追踪错误。
5. 一些错误发生在静态操作环境（也就是执行测试的特定配置）中，而其它一些错误发生在动态操作环境中（也就是实时资源负载或与时间有关的错误）。

# 测试策略

- Web应用测试策略采用对所有软件测试都适用的基本原则，并应用在面向对象系统中所使用的策略。
- 1. 评审Web应用的内容模型以发现错误。
- 2. 评审Web应用的接口模型以保证所有的用例都被考虑到了。
- 3. 评审Web应用的设计模型以发现导航错误。
- 4. 测试用户界面以发现在表现和导航机制中的错误。
- 5. 选择功能模块进行单元测试。
- 6. 测试贯穿整个体系结构的导航路径。
- 7. 在各种不同的环境配置下实现Web应用，并测试每种配置的兼容性。
- 8. 在Web应用及其环境中通过查找易受攻击的漏洞来进行安全性测试。
- 9. 进行性能测试。
- 10. 由一定数量的最终用户测试Web应用，他们和系统的交互结果用来评估内容和导航错误、可用性考虑、兼容性考虑、可靠性和Web应用的性能。



# 测试计划

- Web应用的测试计划应该确定
  - （1）测试开始时的任务集；
  - （2）每个测试任务完成后产生的工作结果（**work product**）；
  - （3）测试结果被评估、记录和在回归测试时重用的方式。
- 有时候测试计划集成在项目计划中；有时测试计划是单独的文档。

## WEB应用测试：任务集

1. 评审涉众的需求；识别用户的关键目的与目标。评审每一类用户的用例。
2. 制定优先级确保每个用户的目的是与目标都被充分测试。
3. 通过描述将要被执行的测试类型定义Web应用测试策略。

## WEB应用测试：任务集

### 4. 制定一个测试计划：

- 定义一个测试进度表和对每一个测试分配职责；
- 指定自动测试工具；
- 为每一类测试定义用户接受的标准；
- 指定缺陷追踪机制；
- 定义问题报告机制；

## WEB应用测试：任务集

### 5. 执行“单元”测试：

- 评审内容以发现句法和语义错误；
- 评审内容以保证适度的清晰和许可；
- 进行界面测试以保证正确的操作；
- 测试每一个组件以确保正确的功能。

### 6. 执行“集成”测试：

- 根据用例测试界面的语义；
- 执行导航测试。

## WEB应用测试：任务集

### 7. 执行配置测试：

- 评估客户端的配置和兼容性；
- 评估服务器端的配置。

### 8. 执行性能测试。

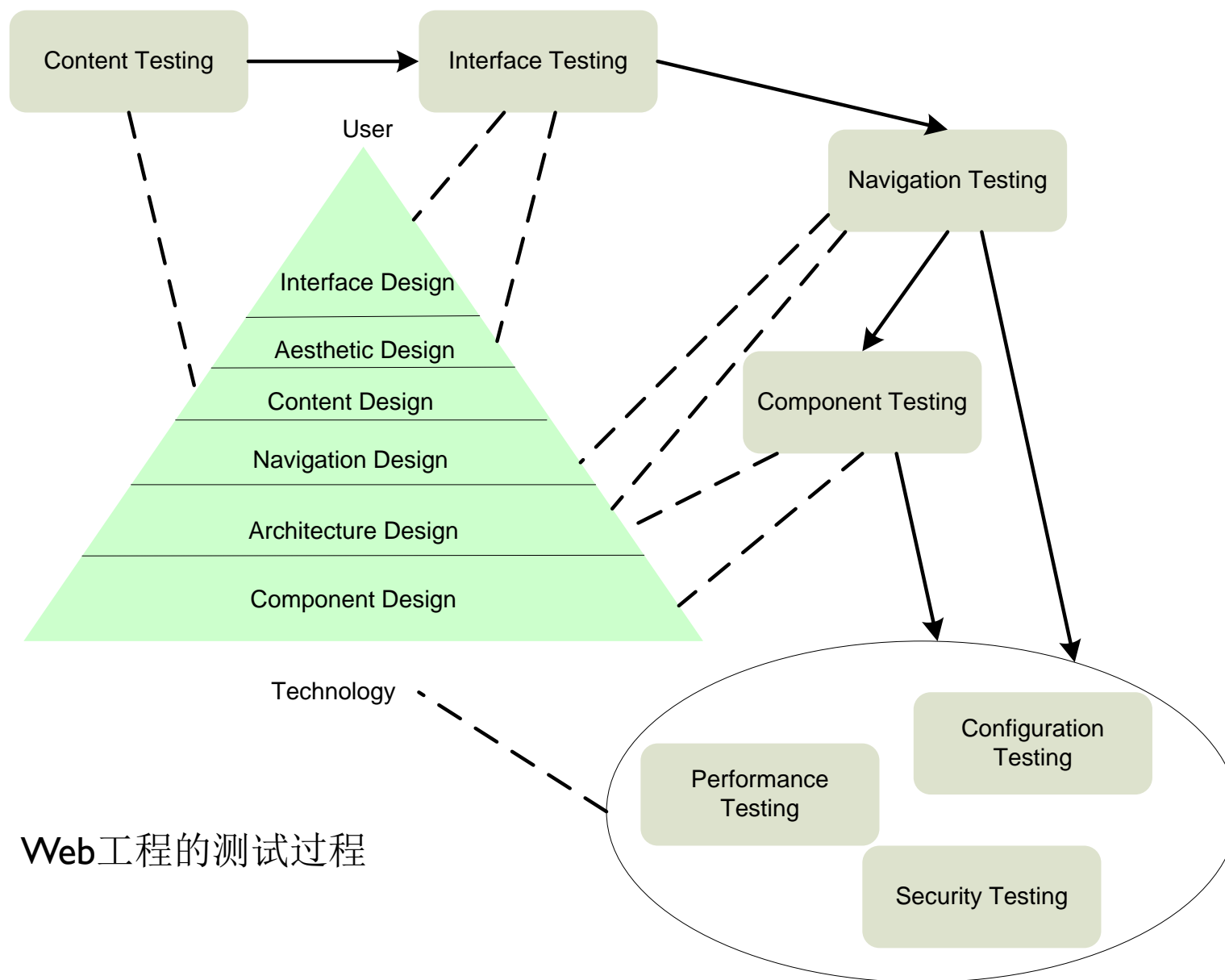
### 9. 执行安全测试。

# 提纲

- 概述
- Web应用测试概念
- ➡ ■ 测试过程概述
- 内容测试
- 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结

## 测试过程概述

- 对于**Web**工程的测试过程是从测试最终用户立即可见的内容和界面功能开始。
- 随后，体系结构设计和导航方面要被测试，用户可能理解也可能不理解这些**Web**应用元素。
- 最后，测试的重点转移到测试对最终用户来说不总是显而易见的技术能力上，如**Web**基础设施和安装/实现问题。



Web工程的测试过程



# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- ➡ ■ 内容测试
- 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结

## 内容测试

- **Web**应用内容的错误可能是微不足道的如排版错误，也可能是严重的错误，如不正确的信息、不合适的组织结构、违反知识产权法等。内容测试是在用户遇到这些和其它很多的问题之前，要先发现它们。
- 内容测试结合评审和生成的可执行的测试用例两种方法。
  - 评审是发现内容中的语义错误。
  - 可执行的测试用例用于发现从一个或多个数据库中获取的数据驱动生成的动态内容中的错误。

## 内容测试目标

- 内容测试有三个重要目标：
  - （1）发现文本文件、图像展示和其他媒体的句法错误（如排版错误，语法错误等）；
  - （2）发现当进行导航时呈现在任何内容对象中的语义错误（即信息准确性和完整性的错误）；
  - （3）找出呈现给最终用户的内容组织或结构上的错误。

## 内容测试目标

- 要完成第一个目标，可以使用自动拼写和语法检查工具。
  - 不过，有许多句法错误无法被这些工具检查出来，因而需要专门评审人员（测试人员）才能查出来。
  - 根据前面所述，审稿是发现句法错误的最好方法。

## 内容测试目标

- 语义测试关注于每个内容对象所呈现的信息。
- 审查者或测试者必须回答以下这些问题：
  - 这些信息准确无误吗？
  - 这些信息是否简明扼要？
  - 内容对象版面设计是否利于最终用户的理解？
  - 内容对象中的信息是否容易被找到？
  - 对于来自其他地方的所有信息是否已经提供恰当的引用说明？

## 内容测试目标

- 审查者或测试者必须回答以下这些问题（续）：
  - 展现的信息是否保持内部一致性， 并且是否和其它的内容对象所现的信息一致？
  - 内容是否具有攻击性、容易引起误解或者容易引起法律诉讼？
  - 内容是否侵犯了其他版权或商标？
  - 内容中是否包含内部链接，而这些链接提供相应的内容？这些链接正确吗？
  - 内容的审美风格是否与界面的审美风格冲突？

## 内容测试目标

- 存在于体系结构中的内容对象具有特殊的形式。
  - 在内容测试中，对内容组织和结构的测试，是为了确保所需的内容以恰当的顺序和关联关系呈现给最终用户。
  - 例如，**cfi.ss.pku.edu.cn** Web应用为项目经理、质量保证（QA）人员、以及社区开发人员提供一个平台。
  - 对**cfi.ss.pku.edu.cn**的内容架构进行测试是为了发现存在于这些信息中的错误（比如，对QA人员的问候语呈现给社区开发人员）。

## 数据库测试

- 现代**Web**应用不仅仅是显示静态的内容对象。
- 在很多应用领域中，**Web**应用和复杂的数据管理系统交互并且实时地从数据库中获取数据从而动态地创建内容对象。



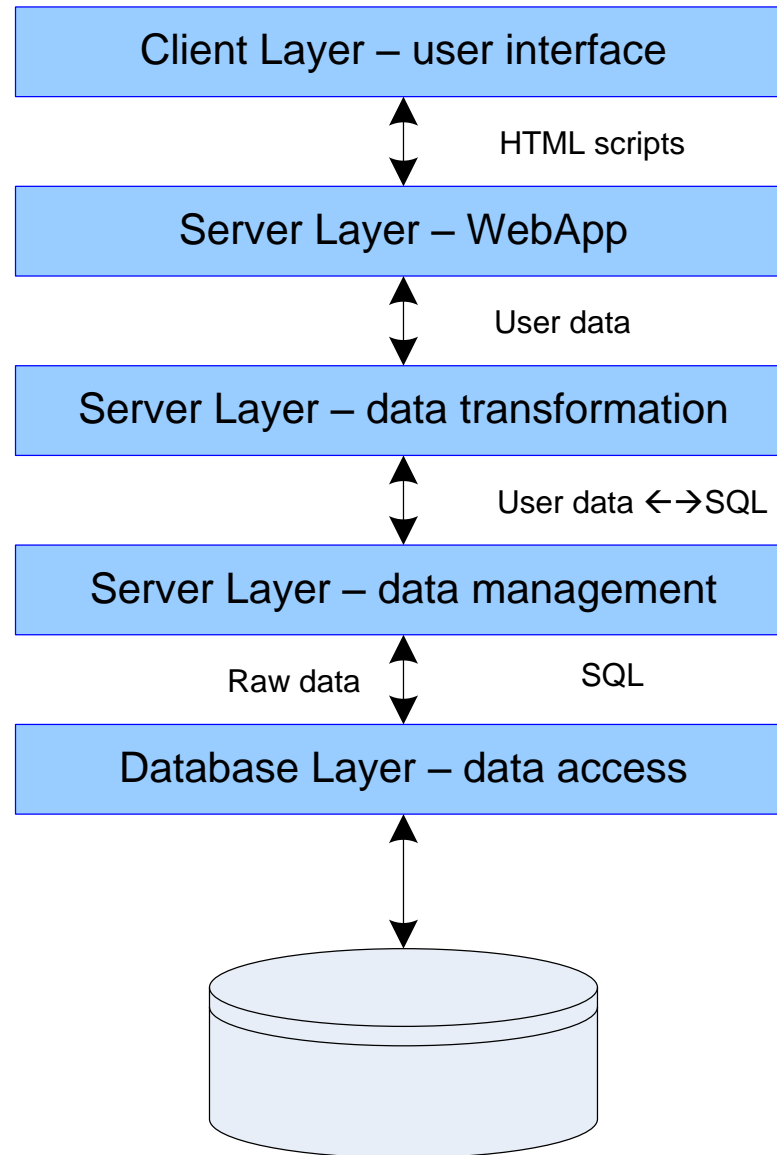
# 数据库测试

- Web应用的数据库测试由于以下因素变的很复杂：
  - 1. 客户端对于信息的原始请求很少能够表现成直接输入数据库管理系统的格式（例如SQL）。所以，要设计测试来查找把用户请求转换成能够被数据库处理的格式过程中产生的错误
  - 2. 数据库可以被远程服务器上的Web应用调用。所以，要测试Web应用和远端数据库通信中的错误。
  - 3. 从数据库得到的原始数据必须传送到服务器应用程序并且转换成恰当的格式以便将来传送给用户。因此，要设计测试方案来说明Web应用服务器接收到的原始数据的有效性，还要生成额外的测试来验证对原始数据进行有效地转换并创建有效的内容对象。
  - 4. 动态的内容对象必须能以某种能展示的形式传送给最终用户。所以，需要设计一系列的测试来（a）找出内容对象格式中的错误；（b）测试不同客户端环境配置下的兼容性。

## 数据库测试 (CONT'D)

### ■ 测试应该确保

- (1) 服务器和客户端之间通过界面层传输的信息有效;
- (2) **Web**应用正确地处理脚本并恰当的抽取或者格式化用户数据;
- (3) 用户数据可以正确地传送到服务器端, 服务器端的数据转换功能能够生成正确的查询 (如**SQL**);
- (4) 此查询被传输到负责与数据库存取例程 (一般在另一台机器上) 通信的数据管理层。



# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- 内容测试
- ➡ ■ 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结

# 用户界面测试

- 在**Web**工程中，有三个不同的地方需要对**Web**应用进行用户界面的确认和验证。
  - 在需求建模和分析阶段：评审界面模型以确保它们符合用户需求并符合分析模型的其他因素。
  - 在设计阶段：评审界面设计模型以确保达到适用所有用户的一般性质量标准，并且妥当地解决了与应用有关的特殊界面的设计问题。
  - 在测试阶段：重点转移到和应用相关的特殊的用户交互方面的执行，如界面的语法和语义所显示的那样。另外，测试也对可用性作最后评价。

# 界面测试策略

- 界面测试的总体策略是：
  - （1）找出与特定界面机制相关的错误（如，菜单链接的不能适当执行错误或数据输入表格的方式错误等）和
  - （2）找出在界面实现导航语义、**Web**应用功能或内容显示方法中存在的错误。

# 界面测试策略

- 为了完成这个策略，必须实现下面的一系列目标：
  - 测试界面特性（**feature**）以确保设计规则、美观以及视觉内容对用户来说是可用的，不存在错误。
    - 特性包括字体形状、颜色、结构、形象、边界、表格以及**Web**应用执行中所生成的相关元素。
  - 类似于单元测试方式，对单个界面机制的进行测试。比如，设计测试来检查所有的表单 (**Forms**)、客户端脚本、动态**HTML**、**CGI**脚本、流内容和应用相关的界面机制（例如用于电子商务应用的购物车）。
    - 在很多情况下，测试可以完全集中在一个界面单元上，而不考虑其他界面的特性和功能。

# 界面测试策略

- 为了完成这个策略，必须实现下面的一系列目标 (cont'd):
  - 针对一个特殊用户类别，使用某个用例或**NSU (Navigation semantic unit)**对每个界面机制都进行测试。这种测试方法类似于集成测试，因为若干界面机制集合在一起以便执行那个用例或**NSU**。
  - 选定若干用例和**NSUs**，测试完整的界面以找出接口中的语义错误。这种测试方法类似于确认测试，因为它的目的就是证明符合特定的用例或**NSU**语义。在这个阶段，要进行一系列易用性测试。
  - 界面在各种环境中（比如浏览器）进行测试以确保是兼容的。实际上，这一系列测试也可以看作是配置测试的一部分。



# 测试界面机制

- **链接**:测试每个导航链接确保能到达恰当的内容对象或者实现相应的功能。
- **表单**:标签,域,默认的选项,浏览器功能（比如“后退”）,脚本,宽度和数据类型,输入的字符串长度,下拉式菜单,自动填写,tab键.
- **客户端脚本**:附带着表单测试.
- **动态HTML**: 进行兼容性测试
- **弹出窗口**: 大小合适, 位置恰当; 不要覆盖原来的窗口; 美观设计一致; 滚动条和其它控制机制
- **CGI脚本**: 黑盒测试的重点放在数据的完整性.
- **流内容**:流数据是最新的,正确显示,毫无错误地挂起和毫无困难地重新开始.
- **Cookies**:服务器端和客户端都需要进行测试.
- **应用相关的界面机制**:测试遵照一个由界面机制定义的功能和特性的检查表.

## 测试界面语义

- 一旦每个界面机制被单元测试过, 界面测试的重点就转到对界面语义的考虑。
- 界面语义测试 “评价设计是否很好地考虑了使用者、提供清楚的方向、传递反馈、并保持语言和方法的一致性”

# 易用性测试

- 易用性测试目的不在交互对象的语义上，易用性评审与测试是要确定**Web**应用界面方便用户使用的程度。
- 步骤
  - 1. 定义一组易用性测试类型及其目标。
  - 2. 为评估每一个目标设计测试。
  - 3. 选择将要进行测试的参与者。
  - 4. 当执行测试时为参与者和**Web**应用之间的交互提供设备。
  - 5. 开发一种评价**Web**应用易用性的机制。

# 易用性测试抽象层次

- 易用性测试可以用于多种不同抽象层次：
  - （1）评价特定的界面机制（如表单）的易用性。
  - （2）评价完整的网页（包括界面机制、数据对象和相关功能）的易用性。
  - （3）评价整个Web应用的易用性。

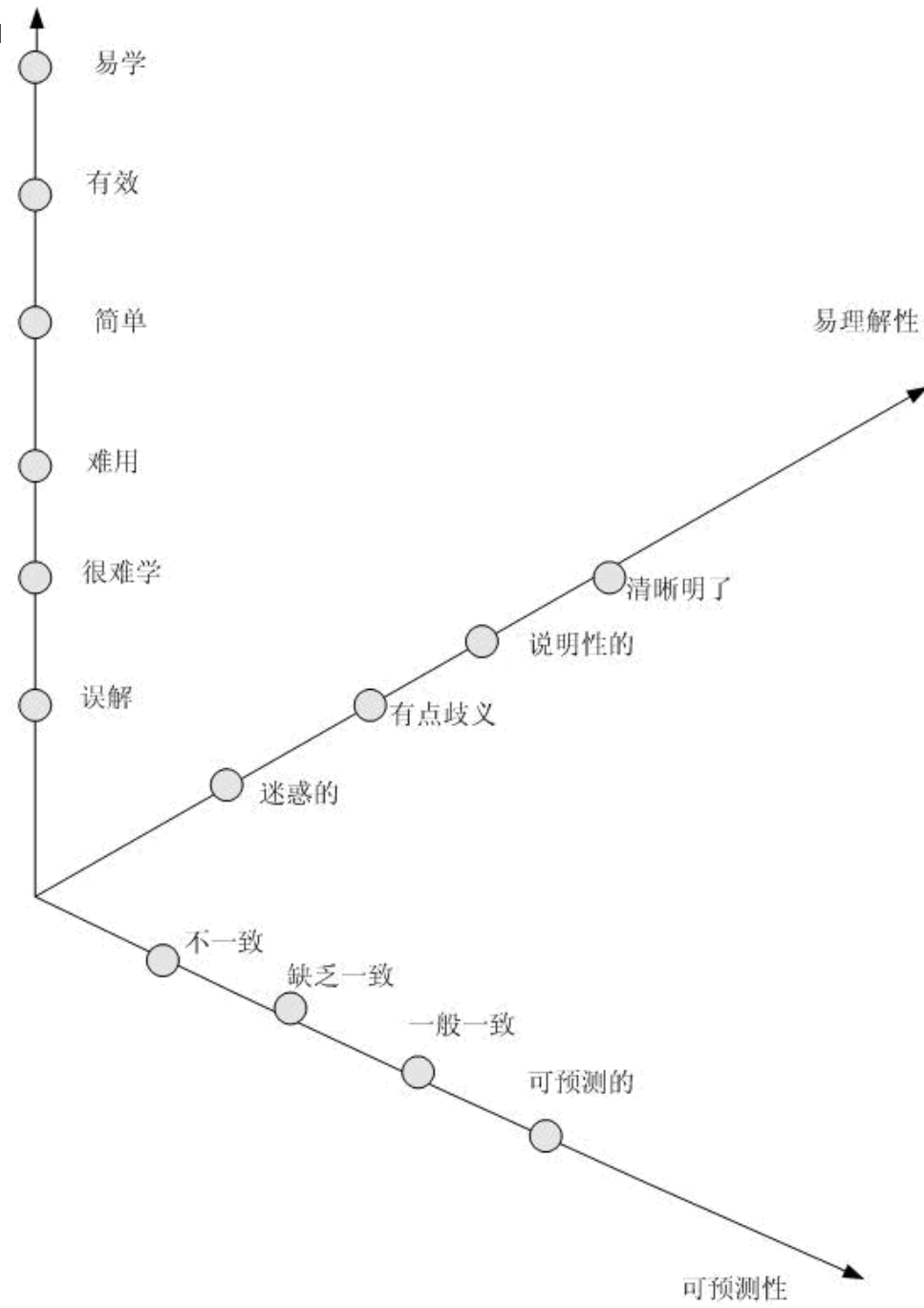
# 易用性类别

- 交互性 — 交互机制（例如：下拉式菜单、按钮、光标）是否易于理解和使用？
- 布局 — 布局风格是否使用户快速找到导航机制、内容、功能。
- 可读性 — 文档是否书写正确，便于理解？图形表述是否易于理解？
- 美观 — 布局、颜色、字体和相关特性是否易于使用？用户是否对**Web**应用感觉舒服？

# 易用性类别 (续)

- 展示特性 — **Web**应用是否使用了最优的屏幕尺寸和分辨率？
- 时间敏感性 — 重要的特性、功能和内容是否可以及时的使用和获得？
- 个性 — **Web**应用是否调整自身以适应不同类别、不同用户的需要？
- 可接近性 — 有身体障碍的人是否可以理解**Web**应用？

易用性



由用户选择的评估“级别”集合

# 兼容性测试

- **Web**应用必须运行于不同的环境。不同的计算机、显示设备、操作系统、浏览器和网络连接速度对**Web**应用的运行有重要的影响。
  - 兼容性测试的第一步是定义一个“经常遇到的”客户端计算机配置及其变化的集合。
  - 下一步，**Web**开发小组导出一系列兼容性确认测试，包括从现有的界面测试、导航测试、性能测试和安全测试。这些测试的目的是发现由配置不同引发的错误或运行时问题。



# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- 内容测试
- 用户界面测试
- ➡ ■ 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结

# 组件级测试

- 组件级测试也叫功能测试，用来发现Web应用功能方面错误。
- 每个Web应用功能是一个软件模块（用一种程序或脚本语言实现），可以使用黑盒测试技术（有时候用白盒技术）进行测试。

## 测试用例设计方法

- 等价类划分——功能的输入域被分成输入类别，由这些类别生成测试用例。评估输入形式从而确定什么数据类别和功能有关。当其它的输入类别不变时，为每一输入类别生成测试用例并执行测试用例。
- 边界值分析——测试表单数据的边界值。
- 路径测试——如果功能的逻辑复杂度很高，使用路径测试（一种白盒测试用例设计方法）确保程序中每个独立的路径都被测试到。

# 组件级测试

- 每个组件级别的测试用例指定组件提供的所有输入值和期望的输出。记录下测试所产生的实际结果输出，以便将来在技术支持和软件维护时参考。
- 在很多情况下，**Web**应用功能的正确执行是与连接外部数据库的交互紧密联系的，因此，数据库测试成为组件级测试不可缺少的组成部分。

# 提纲

- 概述
- Web应用测试概念 → ■ 导航测试
- 测试过程概述
- 配置测试
- 内容测试
- 安全测试
- 用户界面测试
- 性能测试
- 组件级测试
- 总结

# 导航测试

- 导航测试的目的是：
  - （1）保证任何用户可以使用的路径都处于可工作状态；
  - （2）确认每个导航语义单元都可被适当类型的用户使用。

# 测试导航语法

- 导航测试的第一个阶段实际上始于界面测试。对各种导航机制进行测试以保证每一个都完成它们本身的功能。
  - 导航链接,重定向,书签,框架,网站地图,内部搜索引擎
- 上述的一些测试可以通过自动化工具完成（例如链接检验），而其它一些则需要人工设计和执行。

## 测试导航语义

- 它的定义“导航语义单元是信息及其相关导航结构的集合，它们为了完成相关的用户需求而相互协作”
- 每个**NSU**被一组导航路径（称作“导航路”）所定义，导航路径链接导航节点（比如**Web**页面、内容对象、功能）。
- 从整体看，每个**NSU**允许用户完成特定的需求，这种需求由某类用户的一个或多个用例定义。
- 导航测试对每个**NSU**进行测试，确保这些需求被满足。



# 测试导航语义

- 当每个NSU被测试时，Web项目组必须回答下面的问题：
  - NSU是不是无误地达到了它的完整性？
  - 在为NSU定义的导航路径上下文中每个导航节点是不是可达的？
  - 如果NSU可以被多个导航路径达到，是不是每个相关路径都被测试到了？
  - 如果用户界面为辅助导航提供了指导帮助，那么在导航时这些指导是否正确而可理解的吗？
  - 除了浏览器的“back”按钮之外，有返回上一层节点和路径首节点的机制吗？
  - 一个大的导航节点（如很长的网页）内部的导航机制工作正常吗？
  - 如果在一个节点某个功能被执行，而用户没有提供输入，余下的NSU会被完成吗？
  - 如果在某个节点执行功能发生了错误，那么这个NSU能否被完成？
  - 有没有方法可以在所有节点被到达之前暂停导航？并且能不能返回暂停的节点继续导航？
  - 网站地图的节点是否都可以达到？最终用户明白这些节点名称的意思吗？
  - 如果从某个外部的节点到达一个NSU内部的节点，能在导航路径上到达下一个节点吗？能返回到先前的导航路径上的节点吗？
  - 当执行NSU时，用户能理解当前所在内容结构的位置吗？

# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- 内容测试
- 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结

# 配置测试

- Web应用具有很大的挑战性，其中配置的变化和不稳定是很重要的因素。
- 配置测试的工作不是要把每一种可能的客户端配置都测试到，而是测试一组最可能的客户端和服务端配置以保证在所有的配置上用户体验都相同，并把和某一种配置相关的错误隔离出来。

# 服务器端问题



- 在服务器端，设计配置测试用例是为了验证服务器端的计划配置（比如**Web**服务器、数据库服务器、操作系统、防火墙、并发应用等）能否正确无误地支持**Web** 应用。
- 本质上，**Web**应用安装在服务器端，测试是为了在服务器端发现配置相关的错误。

## 服务器端问题 (CONT'D)

- 服务器端配置测试中需要提出并要回答的问题：
  - Web应用完全与服务器操作系统兼容吗？
  - 当Web应用运行时，系统文件、目录、相关的系统数据会被正确创建吗？
  - 系统安全措施允许Web应用执行，对用户的服务没有受到干扰或造成性能下降吗？
  - 当选择分布式服务器配置后Web应用被测试了吗？
  - Web应用能与数据库软件集成吗？Web应用对不同版本的数据库软件敏感吗？
  - Web应用脚本会被正确执行吗？
  - 系统管理员的错误对Web应用的影响被测试了吗？
  - 如果使用代理服务器，是否考虑了在线测试时它们的不同配置会带来什么影响？

# 客户端问题

- 在客户端，配置测试集中在**Web**应用配置的兼容性上。
- 配置包括下列组件的一个或多个置换排列组合：
  - 硬件：**CPU**，内存，存储器，打印设备；
  - 操作系统：**linux**操作系统，**macintosh**操作系统，**Microsoft Windows**操作系统，基于移动设备的操作系统；
  - 浏览器软件：**Internet Explorer**, **Mozilla/Netscape**, **opera**, **safari**, 以及其它的；
  - 用户界面组件：**Active X**, **Java applets**, **SVG**，以及其它的；
  - 插件：**QuickTime**, **RealPlayer**, **SVG Viewer**，以及很多其它的；
  - 网络连接设备及技术：有线，**DSL**，调制解调器，**T1**。

- 
- 
- 为了设计客户端配置测试，Web工程小组必须减少配置变量数目到一个可管理的程度。为了完成这些测试，评估每一类用户从而确定这类用户可能遇到的配置。
  - 并且，可以使用行业共享的数据预测最可能的组件组合，从而在这些环境里进行配置测试。

# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- 内容测试
- 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结







# 安全测试

- Web应用以及服务器端和客户端的运行环境吸引了很多人的注意，如
  - 外部的黑客，抱怨的雇员，不诚实的竞争者，或任何人希望盗取敏感信息、恶意更改内容、降低（系统）性能、使功能不起作用、或使一个人、组织、业务染上麻烦。
- 设计安全测试是为了探查在客户端的漏洞，网络传输数据时的漏洞，以及服务器端的漏洞。
  - 这些域的每一部分都可能被攻击，安全测试人员就是为了发现那些可能会被某些带有企图的人利用的漏洞。

## 安全测试 (CONT'D)

- 客户端的漏洞经常能追溯到存在于浏览器、邮件程序、通信软件里面的bug。
- 另外一个客户端的漏洞是未经授权去访问浏览器中的cookies。带恶意目的的网站可以获取包含在合法的cookies里面信息，利用这些信息危害用户的隐私，更糟糕的是，达到盗取用户身份的目的。

- 
- 
- 客户和服务端之间通信的数据很容易被骗取。
    - 当一端的通信路径被一个怀有恶意的实体破坏就会发生骗取，比如，一个用户可以会被一个恶意的网站骗取，这个恶意网站可能外观跟合法的**Web**应用服务器一样，目的是盗取客户的密码，私有信息，信用卡数据。

## 安全测试 (CONT'D)

- 服务器端的漏洞包括拒绝服务攻击和恶意脚本。恶意脚本可以传给客户或使服务器操作不起作用。并且服务器数据库可能会被未经授权访问（数据盗取）。
- 为了防止这些（或更多的）漏洞，要实现下面的一个或更多安全要素：

## 安全测试 (CONT'D)

- 防火墙：是一个软硬件结合的过滤机制，检测每一个输入包确保它来源的合法性，阻止可疑的包。
- 认证：是一个验证机制，确认客户和服务器的真实身份，只有两边都被确认之后才能允许通信。
- 加密：是一个编码机制，保护敏感数据不被怀有恶意目的地读写和更改。数字认证增强了加密的功能，允许客户验证数据被传输到的目的地。
- 授权：是一个过滤机制，只有那些具有合法授权代码（比如用户ID和密码）的用户才能访问客户端和服务端。

## 安全测试 (CONT'D)

- 安全测试的目的是暴露那些可能被怀有恶意目的的人利用的安全要素中的漏洞。
- 安全测试设计要求对每个安全要素的内在工作（原理）有很深的理解和对网络技术有很广泛深入的理解。
- 在很多情况下，安全测试一般都外包给专门的公司。

# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- 内容测试
- 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结



# 性能测试

- 性能测试是要发现性能问题。性能问题可以来自服务器端缺乏资源，不足的网络带宽，数据库的性能不足，操作系统不完善，糟糕的**Web**应用功能设计和其它的软硬件问题，这些都会导致客户端—服务器性能的下降。
- 测试有两方面的目的：（1）理解系统怎样响应负载（也就是说用户数、事务数或数据量）；（2）收集为了提高性能而更改设计的指标。



# 性能测试目标

- 性能测试能帮助回答下面的问题：
  - 服务器的响应时间会下降到值得注意和不可接受的程度吗？
  - 在哪一点（根据用户数，事务数或数据负载量）性能变得不可接受了？
  - 系统的哪些组件导致性能下降？
  - 在负载变化时用户的平均响应时间是多少？
  - 性能下降对系统安全有影响吗？
  - 随着系统负载增长时**Web**应用的可靠性和准确性会受到影响吗？
  - 当负载大于系统的最大容量时会发生什么情况？

## 性能测试目标

- 为了回答上面这些问题，采用两种性能测试方法：
  - 负载测试：根据负载级别和组合的变化，对现实的负载进行测试；
  - 压力测试：负载增加到转折点时确定**Web**应用环境能处理多大的容量。

# 负载测试

- 负载测试的目的是确定**Web**应用和它的服务器端环境如何响应各种各样的负载条件。
- 下面变量的排列组合定义了一个测试条件集：
  - **N**，并发的用户数
  - **T**，单位时间每个用户的在线事务数
  - **D**，每个事务被服务器处理的数据负载量

## 负载测试

- 用下面的方法计算吞吐量P值。
  - $P = N \times T \times D$
- 考虑一个流行的体育新闻网站，在一个给定的时刻，平均每2分钟20,000个并发用户同时提交一个请求（事务T），每个事务要求Web应用下载一篇平均大小3K的新文章。因此，这样计算吞吐量：
  - $P = (20,000 \times 0.5 \times 3Kb) / 60 = 500Kbytes/秒 = 4M \text{ bits/秒}$ （注：1bytes=8bits）

# 压力测试

- 压力测试是负载测试的继续，但是在这里N，T，D变量要满足和超过操作限制。
- 这些测试的目的是要回答下面的问题：
  - 当超出系统容量时系统是逐渐退化还是服务器关闭吗？
  - 服务软件会出现“服务器不可用”的信息吗？更一般的说，用户会意识到服务器不能访问吗？
  - 服务器对请求进行排队吗？当服务请求取消后服务器会清空请求队列吗？

# 压力测试

- 这些测试的目的是要回答下面的问题：（续）
  - 容量超出限制时事务会丢失吗？
  - 容量超出限制时数据完整性会受到影响吗？
  - 多大的N，T，D值会引起服务器环境失败？怎么声明这些失败？警告信息会自动发给服务器网站的技术支持人员吗？
  - 如果系统失败了，多长时间才能恢复？
  - 当服务器容量达到**80%**到**90%**时，**Web**应用程序的某些功能（比如计算加强功能，数据流能力）会不会不能继续？

# 提纲

- 概述
- Web应用测试概念
- 测试过程概述
- 内容测试
- 用户界面测试
- 组件级测试
- 导航测试
- 配置测试
- 安全测试
- 性能测试
- 总结



# 总结

- **Web**应用测试是测试每个**Web**应用质量的纬度，目的是发现错误或者发现导致质量失败问题。
- 测试集中在内容、功能、结构、可用性、导航、性能、兼容性、互操作、容量、安全等方面。
- 测试应该和**Web**应用设计的评审相结合。