



第7章 系统测试

张圣林

zhangsl@nankai.edu.cn

泰达学院3区407

南开大学



7 系统测试

- 开发的软件只是实际投入使用系统的一个组成部分，还需要检测它与系统其他部分能否协调地工作，这就是**系统测试的任务**
- 系统测试实际上是针对系统中各个组成部分进行的综合性检验，很接近人们的日常测试实践



7 系统测试

- 视频：系统测试概述



7 系统测试

- 7.1 性能测试
- 7.2 压力测试
- 7.3 健壮性测试
- 7.4 安全性测试
- 7.5 可靠性测试
- 7.6 恢复性测试
- 7.7 备份测试
- 7.8 兼容性测试
- 7.9 安装性测试
- 7.10 可用性测试
- 7.11 配置测试
- 7.12 文档测试
- 7.13 GUI测试
- 7.14 验收测试
- 7.15 回归测试
- 7.16 网站测试



7.1 性能测试的基本概念（1）

■ 性能

- 性能是一种表明软件系统或构件对于**及时性**要求的符合程度的指标
- 性能是软件产品的一种**特性**，可以用时间来度量。性能的及时性通常用系统对请求做出**响应所需要的时间来衡量**

■ 性能测试

- 检验软件是否达到需求规格说明书中规定的各类性能指标，并满足一些性能相关的约束和限制条件

■ 性能测试的目的

- 通过测试，确认软件是否满足产品的**性能需求**，同时发现系统中存在的**性能瓶颈**，并对系统进行**优化**



7.1 性能测试的基本概念（2）

- 性能测试包括以下几个方面
 - 评估系统的能力
 - 测试中得到的负荷和响应时间等数据可以被用于验证所计划的模型的能力，并帮助做出决策
 - 识别系统中的弱点
 - 受控的负荷可以被增加到一个极端的水平并突破它，从而修复系统的瓶颈或薄弱的地方
 - 系统调优
 - 重复运行测试，验证调整系统的活动得到了预期的结果，从而改进性能，检测软件中的问题



性能测试方法

- 性能测试方法包括
 - 基准法
 - 性能下降曲线分析法等
- 基准法中关于性能测试的基准大体有以下几方面
 - 响应时间
 - 并发用户
 - 吞吐量
 - 性能计数器



响应时间

- 完成用户请求的时间，如从向系统发出请求开始，到客户端接收到最后一个字节数据为止所消耗的时间
- 对用户来讲，响应时间的长短并没有绝对的区别
 - 例如一个税务报账系统，用户每月使用一次该系统，每次进行数据录入等操作需要2小时以上的时间，当用户选择提交后，即使系统在**20分钟**后才给出处理成功的消息，用户仍然不会认为系统的响应时间不能接受
 - 因为相对于一个月才进行一次的操作来说，20分钟是一个可以接受的等待时间。所以在进行性能测试的时候，合理的响应时间取决于实际的**用户需求**，而不能根据测试人员自己的设想来决定



并发用户数

- 并发用户数一般是指同一时间段内访问系统的用户数量
- 在实际的性能测试中，经常接触到的与并发用户数相关的概念还包括“系统用户数”、“同时在线用户人数”和“同时操作用户数”
 - 例如，某大学的网站有邮件服务及学生信息、选课系统等业务，基于这些业务的用户共有10000人，则这10000个用户就称为系统用户数。假设整个网站在最高峰时有6000人同时在线，则这6000人可以称作同时在线用户人数。假设600人同时请求操作，则600人作为系统的并发用户数

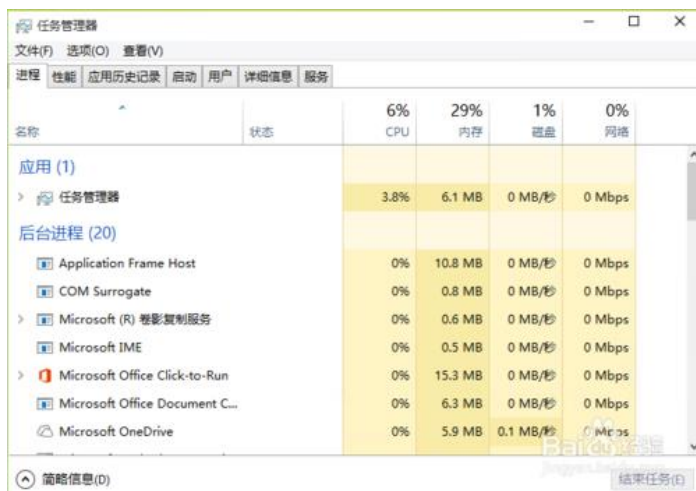


吞吐量

- 吞吐量指单位时间内系统处理的客户请求数量。例如每秒钟请求数或每秒钟处理**页面数/秒**，或者每分钟完成的**事务数/分**来衡量。吞吐量指标可以直接体现软件系统的性能

性能计数器

- 性能计数器是描述服务器或操作系统性能的一些**数据指标**。计数器在性能测试中发挥着**监控**和**分析**关键作用，尤其是在分析系统的可扩展性、进行性能瓶颈定位时，对计数器的取值的分析比较关键
 - ▣ 比如Windows系统任务管理器就是一个性能计数器，它提供了测试机CPU、内存、网络和硬盘的使用信息



The screenshot shows the Windows Task Manager Performance tab. At the top, there are four progress bars representing overall system usage: CPU at 6%, Memory at 29%, Disk at 1%, and Network at 0%. Below these, a table lists the resource usage for various applications and background processes.

名称	状态	CPU	内存	磁盘	网络
应用 (1)					
任务管理器		3.8%	6.1 MB	0 MB/秒	0 Mbps
后台进程 (20)					
Application Frame Host		0%	10.8 MB	0 MB/秒	0 Mbps
COM Surrogate		0%	0.8 MB	0 MB/秒	0 Mbps
Microsoft (R) 卷影复制服务		0%	0.6 MB	0 MB/秒	0 Mbps
Microsoft IME		0%	0.5 MB	0 MB/秒	0 Mbps
Microsoft Office Click-to-Run		0%	15.3 MB	0 MB/秒	0 Mbps
Microsoft Office Document C...		0%	6.3 MB	0 MB/秒	0 Mbps
Microsoft OneDrive		0%	5.9 MB	0.1 MB/秒	0 Mbps



性能测试执行（1）

- （1）计划阶段
 - 定义目标并设置期望值
 - 收集系统和测试要求
 - 定义工作负载
 - 选择要收集的性能度量值
 - 标出要运行的测试并决定什么时候运行它们
 - 决定工具选项和生成负载
 - 编写测试计划，设计用户场景并创建测试脚本



性能测试执行（2）

- （2）测试阶段
 - 做准备工作（如建立测试服务器或布置其他设备）
 - 运行测试
 - 收集数据
- （3）分析阶段
 - 分析结果
 - 改变系统以优化性能
 - 设计新的测试





7.2 压力测试

- 压力测试基本概念
- 压力测试的特点
- 压力测试方法
- 压力测试执行



压力测试基本概念

- 压力测试（Stress Testing）是指模拟巨大的工作负荷，以查看系统在峰值使用情况下是否可以正常运行
- 压力测试是通过逐步增加系统负载来测试系统性能的变化，并最终确定在什么负载条件下系统性能处于失效状态，以此来获得系统性能提供的最大服务级别的测试



压力测试特点（1）

- 压力测试是检查系统处于压力情况下的能力表现。
- 通过不断增加系统压力，来检测系统在不同压力情况下所能够到达的工作能力和水平
 - ▣ 比如，通过增加并发用户的数量，检测系统的服务能力和水平；通过增加文件记录数来检测数据处理的能力和水平等等
- 压力测试一般通过模拟方法进行



压力测试特点（2）

- 压力测试是一种极端情况下的测试，所以为了捕获极端状态下的系统表现往往采用模拟方法进行。通常在系统对内存和CPU利用率上进行模拟，以获得测量结果
 - ▣ 如将压力的基准设定为：内存使用率达到75%以上、CPU使用率达到75%以上，并在此观测系统响应时间、系统有无错误产生。除了对内存和CPU的使用率进行设定外，数据库的连接数量、数据库服务器的CPU利用率等等也都可以作为压力测试的依据



压力测试特点（3）

- 压力测试一般用于测试系统的稳定性
- 如果一个系统能够在压力环境下稳定运行一段时间，那么该系统在普遍的运行环境下就应该可以达到令人满意的稳定程度。在压力测试中，通常会考察系统在压力下是否会出现错误等方面的问题



压力测试特点（4）

■ 与性能测试的联系与区别

- ▣ 压力测试用来保证产品发布后系统能否满足用户需求，关注的重点是系统整体；性能测试可以发生在各个测试阶段，即使是在单元层，一个单独模块的性能也可以进行评估
- ▣ 压力测试是通过确定一个系统的瓶颈，来获得系统能提供的最大服务级别的测试。性能测试是检测系统在一定负荷下的表现，是正常能力的表现；而压力测试是极端情况下的系统能力的表现
 - 例如对一个网站进行测试，模拟10到50个用户同时在线并观测系统表现，是常规性能测试；当用户增加到系统出项瓶颈时，如1000乃至上万个用户时，则为压力测试



压力测试方法

- 重复
- 并发
- 量级增加
- 随机



重复压力测试

- 重复测试就是一遍又一遍地执行某个操作或功能，比如重复调用一个Web服务
- 压力测试的一项任务就是确定在极端情况下一个操作能否正常执行，并且能否持续不断地在每次执行时都正常。这对于推断一个产品是否适用于某种生产情况至关重要，客户通常会重复使用产品。重复测试往往与其它测试手段一并使用



并发压力测试

- 并发是同时执行多个操作的行为，即在同一时间执行多个测试线程
 - ▣ 例如，在同一个服务器上同时调用许多Web服务。并发测试原则上不一定适用于所有产品，但多数软件都具有某个并发行为或多线程行为元素，这一点只能通过执行多个代码测试用例才能得到测试结果



量级增加压力测试

- 压力测试可以重复执行一个操作，但是操作自身也要尽量给产品增加负担
 - ▣ 例如一个Web服务允许客户机输入一条消息，测试人员可以通过模拟输入超长消息来使操作进行高强度的使用，即增加这个操作的量级。这个量级的确定总是与应用系统有关，可以通过查找产品的可配置参数来确定量级。例如，数据量的大小、延迟时间的长度、输入速度以及输入的变化等



随机压力测试

- 该手段是指对上述测试手段进行随机组合，以便获得最佳的测试效果
 - ▣ 例如，使用重复时，在重新启动或重新连接服务之前，可以改变重复操作间的时间间隔、重复的次数，或者也可以改变被重复的Web服务的顺序
 - ▣ 使用并发时，可以改变一起执行的Web服务、同一时间运行的Web服务数目，也可以改变关于运行许多不同的服务还是运行许多同样的实例的决定
 - ▣ 量级测试时，每次重复测试时都可以更改应用程序中出现的变量（例如发送各种大小的消息或数字输入值）



压力测试执行

- 压力测试用例选取可以从以下几个方面考虑
 - 检查是否有足够的磁盘空间
 - 检查是否有足够的内存空
 - 创造极端的网络负载
 - 制造系统溢出条件





7.3健壮性测试基本概念（1）

- 健壮性测试（Robustness Testing）
 - 主要用于测试系统抵御错误的能力
 - 这里的错误通常是指由于设计缺陷而带来的系统错误。测试的重点为当出现故障时，是否能够自动恢复或忽略故障继续运行
- 健壮性测试的现状
 - 由于受到开发成本、时间和人员等条件的约束，企业经常把软件测试的关注点放在功能正确性上面，往往分配少量的资源用于确定系统的异常处理，从而忽略系统健壮性。
 - 该矛盾随着软件应用的日益普遍而异常突出，所以一个好的软件系统必须经过健壮性测试之后才能最终交付给用户



7.3健壮性测试基本概念（2）

- 健壮性有两层含义：一是高可靠性，二是从错误中恢复的能力
 - 前者体现了软件系统的**质量**；后者体现了软件系统的**适应性**。二者也给测试工作提出了不同的测试要求
 - 前者需要根据**符合规格说明**的数据选择测试用例，用于检测在正常情况下系统输出的正确性
 - 后者需要在**异常数据**中选择测试用例，检测非正常情况下的系统行为



健壮性测试方法（1）

- 可以根据以下几个方面评价系统的健壮性
 - 通过
 - 系统调用运行输入的参数产生预期的正常结果
 - 灾难性失效
 - 这是系统健壮性测试中最严重的失效，这种失效只有通过系统重新引导才能得到解决
 - 重启失效
 - 一个系统函数的调用没有返回，使得调用它的程序挂起或停止



健壮性测试方法（2）

- 夭折失效
 - 程序执行时由于异常输入，系统发出错误提示使程序中
止
- 沉寂失效
 - 异常输入时，系统应当发出错误提示，但是测试结果却
没有发生异常
- 干扰失效
 - 指系统异常时返回了错误提示，但是该错误提示不是期
望中的错误



健壮性测试方法（3）

- 自动化实现上述测试内容时需要把握的原则
 - 可移植性
 - 健壮性测试基准程序是用来比较不同系统的健壮性，因此移植性是测试基准程序的基本要求
 - 覆盖率
 - 理想的基准程序能够覆盖所有的系统模块，然而这种开销是巨大的。因此一般选取使用频度最高的模块进行测试



健壮性测试方法（4）

- 可扩展性

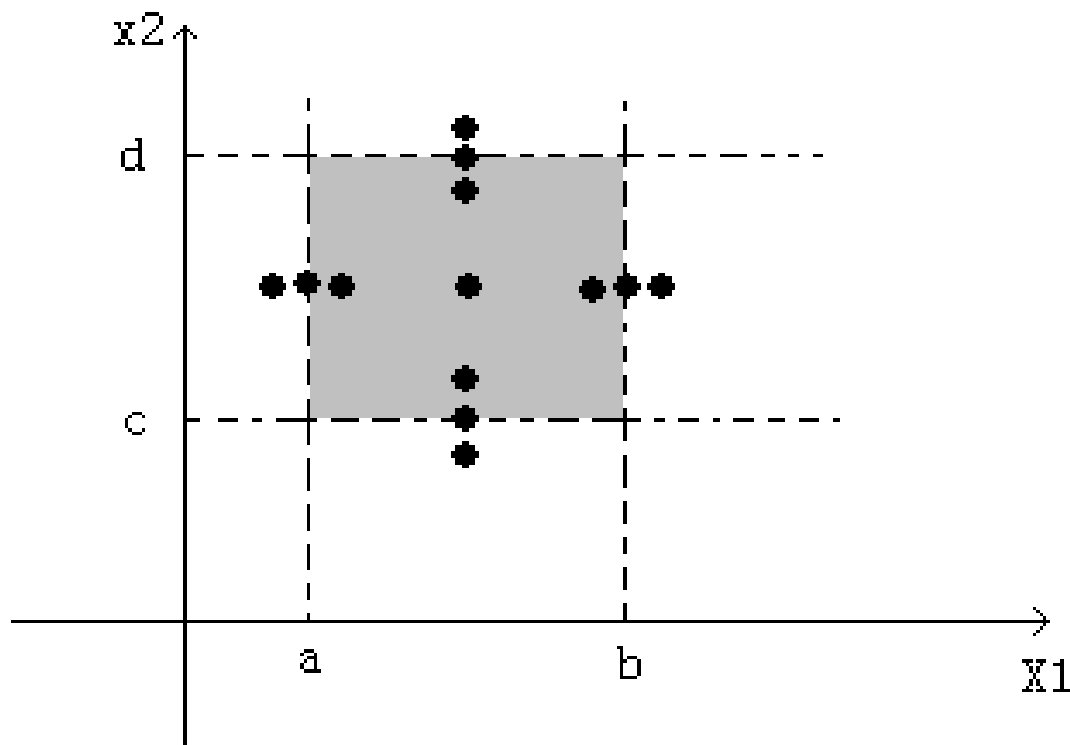
- 可扩展性体现在当需要扩展测试集时能够前后一致。这种可扩展性不仅指为已有模块增加测试集，还包括为新增加的模块增加测试集

- 测试结果的记录

- 健壮性测试的目的是找出系统的不健壮性因素，因此应详细的记录测试结果

健壮性测试案例

- 假如定义了两个变量X1和X2，两个变量都有自己的取值范围，则写成如下这种形式： $a < X1 < b$ ； $c < X2 < d$ ；用坐标的形式表示如下图所示





7.4 安全性测试

- 安全性测试基本概念
- 安全性测试手段
- 安全性测试层次
- 安全性测试方法
 - 功能验证
 - 漏洞扫描
 - 模拟攻击
 - 侦听技术



安全性测试基本概念

■ 安全性测试

- 检查系统对非法侵入的**防范**能力，其目的是为了发现软件系统中是否存在**安全漏洞**。软件安全性是指在**非正常条件**下**不发生安全事故**的能力

■ 系统安全设计的准则

- 使非法侵入的**代价**超过被保护的信息的价值，从而令非法侵入者**无利可图**
- 一般来讲，如果黑客为非法入侵花费的代价（考虑时间、费用、危险等因素）高于得到的好处，那么这样的系统可以认为是安全的系统



安全性测试手段

- 测试者扮演一个试图攻击系统的角色
 - ▣ 尝试通过外部的手段来获取系统的密码
 - ▣ 使用能够瓦解任何防护的客户软件来攻击系统
 - ▣ 把系统“制服”，使别人无法访问
 - ▣ 有目的地引发系统错误，期望在系统恢复过程中侵入系统
 - ▣ 通过浏览非保密的数据，从中找到进入系统的钥匙等



安全性测试层次

- 安全性一般分为两个层次

- 应用程序级别的安全性：对数据或业务功能的访问
- 系统级别的安全性：对系统的登录或远程访问

- 二者的关系

- 应用程序级别的安全性可确保在预期的安全性情况下，操作者只能访问特定的功能或用例，或者只能访问有限的数
据
 - 例如，某财务系统可能会允许所有人输入数据，创建新账户，但只有管理员才能删除这些数据或账户
- 系统级别的安全性对确保只有具备系统访问权限的用户才能访问应用程序，而且只能通过相应的入口来访问



安全性测试方法

- 功能验证
- 漏洞扫描
- 模拟攻击
- 侦听技术



功能验证（1）

- 功能验证是采用软件测试当中的黑盒测试方法，对涉及安全的软件功能，如用户管理模块、权限管理模块、加密系统、认证系统等进行测试，主要是验证上述功能是否有效



功能验证（2）

- 功能性的安全性问题包括
 - 控制特性是否工作正确？
 - 无效的或者不可能的参数是否被检测并且适当的处理？
 - 无效的或者超出范围的指令是否被检测并且适当的处理？
 - 错误和文件访问是否适当的被记录？
 - 是否有变更安全性表格的过程？
 - 系统配置数据是否能正确保存，系统故障时是否能恢复？
 - 系统配置数据能否导出，在其他机器上进行备份？



功能验证（3）

- 功能性的安全性问题包括
 - 系统配置数据能否导入，导入后能否正常使用？
 - 系统配置数据保存时是否加密？
 - 没有口令是否可以登录到系统中？
 - 有效的口令是否被接受，无效的口令是否被拒绝？
 - 系统对多次无效口令是否有适当的反应？
 - 系统初始的权限功能是否正确？
 - 各级用户权限划分是否合理？
 - 用户的生命期是否有限制？
 - 低级别的用户是否可以操作高级别用户命令？



功能验证（4）

- 功能性的安全性问题包括
 - 高级别的用户是否可以操作低级别用户命令？
 - 用户是否会自动超时退出，超时的时间是否设置合理，用户数据是否会丢失？
 - 登录用户修改其他用户的参数是否会立即生效？
 - 系统在最大用户数量时是否操作正常？
 - 对于远端操作是否有安全方面的特性？
 - 防火墙是否能被激活和取消激活？
 - 防火墙功能激活后是否会引起其他问题？



漏洞扫描（1）

- 安全漏洞扫描通常都是借助于特定的漏洞扫描器完成
 - ▣ 漏洞扫描器是一种能自动检测远程或本地主机安全性弱点的程序，通过使用漏洞扫描器，系统管理员能够发现所维护信息系统存在的安全漏洞，从而在信息系统网络安全防护过程中做到有的放矢，及时修补漏洞
- 安全漏洞扫描是可以用于日常安全防护，同时可以作为对软件产品或信息系统进行测试的手段，可以在安全漏洞造成严重危害前发现漏洞并加以防范



漏洞扫描（2）

- 按常规标准，可以将漏洞扫描器分为**两种**类型：主机漏洞扫描器（Host Scanner）和网络漏洞扫描器（Network Scanner）
 - 主机漏洞扫描器是指在系统**本地运行**检测系统漏洞的程序，如著名的COPS、Tripewire、Tiger等自由软件
 - 网络漏洞扫描器是指基于网络**远程检测**目标网络和主机系统漏洞的程序，如Satan、ISS Internet Scanner等



模拟攻击试验

- 模拟攻击试验是一组特殊的黑盒测试案例，通常以模拟攻击来验证软件或信息系统的安全防护能力
 - 冒充
 - 重演
 - 消息篡改
 - 拒绝服务
 - 内部攻击
 - 外部攻击
 - 陷阱
 - 木马



冒充

■ 口令猜测

- 一旦黑客识别了一台主机，而且发现了基于NetBIOS、Telnet或NFS服务的可利用的用户账号，并成功地猜测出了口令，就能对机器进行控制

■ 缓冲区溢出

- 在服务程序中，如果程序员使用类似于strcpy()、strcat()等字符串函数，由于这些函数不进行有效位检查，所以可能导致恶意用户编写一小段程序来进一步打开缺口，将该代码放在缓冲区有效载荷末尾。这样，当发生缓冲区溢出时，返回指针指向恶意代码，执行恶意指令，就可以得到系统的控制权



重演

- 当一个消息或部分消息为了产生非授权效果而被重复时，就出现了重演
 - ▣ 例如，一个含有鉴别信息的有效消息可能被另一个实体所重演，目的是鉴别它自己



消息篡改

■ DNS高速缓存污染

- 由于DNS服务器与其他名称服务器**交换信息**的时候并不进行身份验证，这就使黑客可以加入不正确的信息，并把用户引向黑客自己的主机

■ 伪造电子邮件

- 由于SMTP并不对邮件**发送附件**的身份进行鉴定，因此黑客可以对内部客户伪造电子邮件，声称是来自某个客户认识并相信的人，并附上可安装的特洛伊木马程序，或者是一个指向恶意网站的链接



拒绝服务 (Dos)

- Denial of service -> Distributed DoS
- 当一个实体不能执行它的正常功能，或它的动作妨碍了别的实体执行它们的正常功能的时候，便发生服务拒绝
 - 可能是一般性的，比如一个实体抑制所有的消息
 - 也可能是有具体目标的，例如一个实体抑制所有流向某一特定目的端的消息，如安全审计服务
 - 死亡之Ping、泪滴、UDP泛洪、SYN泛洪、Land攻击、Smurf攻击、Fraggle攻击、电子邮件炸弹、畸形消息攻击



拒绝服务 (Dos)

■ 死亡之Ping

- 由于在早期的阶段，路由器对包的最大尺寸都有限制，许多操作系统对TCP/IP栈的实现在ICMP包上都是规定64KB，并且在对包的标题头进行读取之后，要根据该标题头里包含的信息来为有效载荷生成缓冲区，当产生畸形的，声称自己的尺寸超过ICMP上限的包也就是加载的尺寸超过64K上限时，就会出现内存分配错误，导致TCP/IP堆栈崩溃，致使接受方当机。

■ SYN泛洪

- SYN攻击利用的是TCP的三次握手机制，攻击端利用伪造的IP地址向被攻击端发出请求，而被攻击端发出的响应报文将永远发送不到目的地，那么被攻击端在等待关闭这个连接的过程中消耗了资源，如果有成千上万的这种连接，主机资源将被耗尽，从而达到攻击的目的。



内部攻击

- 当系统的合法用户以非故意或非授权方式进行动作时就成为内部攻击
- 防止内部攻击的保护方法
 - 所有管理数据流进行加密
 - 利用包括使用强口令在内的多级控制机制和集中管理机制来加强系统的控制能力
 - 为分布在不同场所的业务部门划分VLAN，将数据流隔离在特定部门
 - 利用防火墙为进出网络的用户提供认证功能，提供访问控制保护
 - 使用安全日志记录网络管理数据流等



外部攻击

- 外部攻击可以使用的办法
 - 搭线窃听（主动的与被动的）
 - 截取辐射
 - 冒充为系统的授权用户
 - 冒充为系统的组成部分
 - 为鉴别或访问控制机制设置旁路等

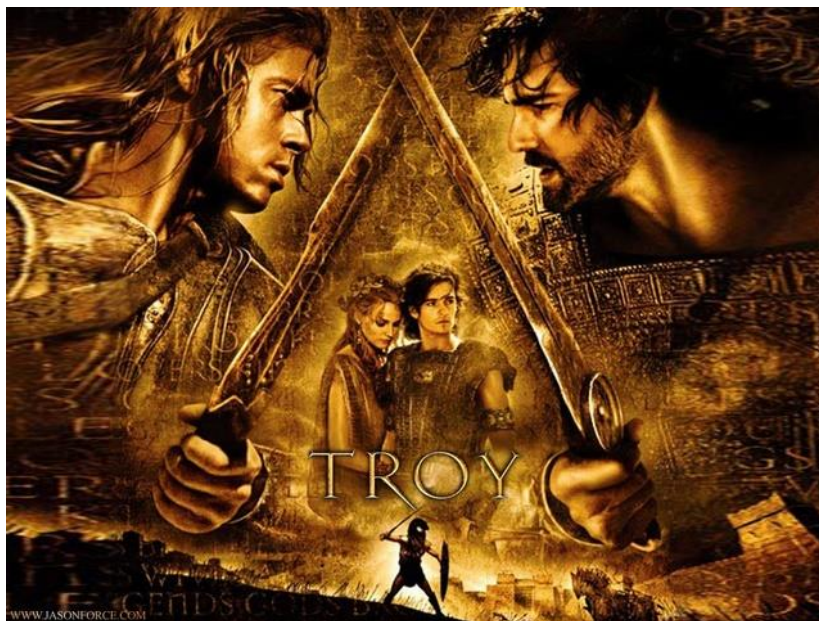


陷阱

- 当系统的实体受到改变，致使一个攻击者能对命令或对预定的事件或事件序列产生非授权的影响时，其结果就称为陷阱门
 - ▣ 例如：口令的有效性可能被修改，使其除了正常效力之外也使攻击者的口令生效

木马

- 对系统而言的特洛伊木马，是指它不但具有自己的授权功能，而且还有非授权功能
 - ▣ 一个向非授权信道拷贝消息的中继就是一个特洛伊木马
 - ▣ 典型的特洛伊木马有NetBus、BackOffice和BO2k等





侦听技术

- 侦听技术实际上是在数据通信或数据交互过程，对数据进行**截取分析**的过程
- 目前最为流行的是网络数据包的捕获技术，通常称为Capture，黑客可以利用该项技术实现数据的盗用，而测试人员同样可利用该项技术实现安全测试
- 该项技术主要用于对网络加密的验证





7.5 可靠性测试

- 可靠性测试概念
- 可靠性测试基本参数



基本概念

■ 软件可靠性

- 在规定的时间内、规定的运行剖面上运行规定的软件，测试其是否能够正常执行的能力

■ 度量指标

- 平均无故障时间是否超过规定时限
- 因故障而停机的时间在一年中应不超过多少时间等

■ 可用**软件可靠性模型**来评估这些指标的有效性

- 目前，已建立了六十余种不同类型的可靠性模型
- 分别为指数失效时间模型、Weibull和Gamma分布失效时间模型、无限失效时间模型、Bayes模型等

基本参数

- 假设系统S投入测试或运行后，工作一段时间 t_1 后，软件出现错误，系统被停止并进行修复，经过 T_1 时间后，故障被排除，又投入测试或运行。假设 t_1, t_2, \dots, t_n 是系统正常的工作时间， T_1, T_2, \dots, T_n 是维护时间，如下图所示



系统工作状态图



基本参数

- 故障率（风险函数）

$$\lambda = \frac{\text{总失效次数}}{\text{总工作时间}} = \frac{n}{\sum_{i=1}^n t_i}$$

- λ 的单位是FIT，1FIT=10⁻⁹/小时



基本参数

- 维修率

$$u = \frac{\text{总失效次数}}{\text{总维护时间}} = \frac{n}{\sum_{i=1}^n T_i}$$

- 平均无故障时间

$$MTBF = \frac{\text{总工作时间}}{\text{总失效次数}} = \frac{\sum_{i=1}^n t_i}{n} = \frac{1}{\lambda}$$



基本参数

■ 平均维护时间

$$MTTR = \frac{\text{总维护时间}}{\text{总失效次数}} = \frac{\sum_{i=1}^n T_i}{n} = \frac{1}{u}$$

■ 有效度

$$A = \frac{\text{总工作时间}}{\text{总工作时间} + \text{总维护时间}} = \frac{MTBF}{MTBF + MTTR} = \frac{u}{\lambda + u}$$



基本参数

- 残留的缺陷数目 N_0 ：可以采用第二章叙述的方法进行估计
- 可靠性

$$R(t) = e^{-\int_0^t \lambda(t) dt}$$





7.6恢复性测试

■ 恢复性测试

- 主要检查系统的容错能力
- 当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。恢复测试首先要采用各种办法强迫系统失败，然后验证系统是否能尽快恢复
- 对于自动恢复需验证重新初始化、检查点、数据恢复和重新启动等机制的正确性
- 对于人工干预的恢复系统，还需估测平均修复时间，确定其是否在可接受的范围内

7.6恢复性测试

- 在设计恢复性测试用例时，需要考虑的问题
 - 测试是否存在潜在的灾难，以及它们可能造成的损失？
消防训练式的布置灾难场景是一种有效的方法
 - 保护和恢复工作是否为灾难提供了足够的准备？
评审人员应该评审测试工作及测试步骤，以便检查对灾难的准备情况。评审人员包括主要事件专家和系统用户
 - 当真正需要时，恢复过程是否能够正常工作？
模拟的灾难需要和实际的系统一起被创建以验证恢复过程。用户、供应商应当共同完成测试工作





7.7 备份测试

- 备份测试是恢复性测试的一个补充，也是恢复性测试的一个部分
- 备份测试的目的是验证系统在软件或者硬件失败时备份数据的能力



7.7 备份测试

- 备份测试需要从以下几个角度来进行设计
 - 备份文件，并且比较备份文件与最初的文件的区别
 - 存储文件和数据
 - 完善系统备份工作的步骤
 - 检查点数据备份
 - 备份引起系统性能衰减程度
 - 手工备份的有效性
 - 系统备份“触发器”的检测
 - 备份期间的安全性
 - 备份过程日志





7.8兼容性测试

- 兼容性测试是指检查软件之间是否能够正确地进行交互和共享信息
- 软件兼容性测试需要解决的问题
 - 软件设计需求与运行平台的兼容性
 - 如果被测软件本身是一个支持平台，那么还要设计一个在该平台上运行的应用程序
 - 软件的行业标准或规范，以及如何达到这些标准和规范的条件
 - 被测软件与其它平台、其它软件交互或共享的信息



7.8兼容性测试

- 举例1：Microsoft Windows认证软件要求
 - 支持三键以上的鼠标
 - 支持在C： 和D： 以外的磁盘上安装
 - 支持超过DOS 8.3格式文件名长度的文件名
 - 不能读、写，或者以其它形式使用旧系统中的win.ini、system.ini、autoexec.bat和config.sys文件



7.8兼容性测试

■ 举例2：浏览器测试

- ActiveX是Microsoft为Internet Explorer而设计的产品
- JavaScript是Sun为Netscape设计的Java产品
- 浏览器的测试就是要测试Netscape对ActiveX，以及Explorer对JavaScript兼容性的测试

	Explorer浏览器	Netscape浏览器
ActiveX	√	√
JavaScript	√	√



7.8兼容性测试

- 向前和向后兼容
 - 向后兼容是指可以使用软件的以前版本
 - 向前兼容是指可以使用软件的未来版本
- 不同版本之间的兼容性
 - 测试平台和应用软件多个版本之间是否能够正常工作
- 标准和规范
 - 高级标准是产品普遍应遵守的规章
 - 低级标准是对产品开发细节的描述
- 数据共享兼容性





7.9 安装性测试

- 软件运行的第一件事就是安装（除嵌入式软件外），安装测试是软件测试首先需要解决的问题
- 安装测试需要不仅要考虑在不同的操作系统上运行，还要考虑与现有软件系统的配合使用问题



7.9 安装性测试

■ 安装测试应从以下方面考虑

- 应参照安装手册中的步骤进行安装，主要考虑到安装过程中所有的缺省选项和典型选项的验证。安装前应先备份测试机的注册表
- 安装有自动安装和手工配置之分，应测试不同的安装组合的正确性，最终使所有组合均能安装成功
- 安装过程中异常配置或状态情况要进行测试
- 检查安装后能否产生正确或是多余的目录结构和文件，以及文件属性是否正确

7.9 安装性测试

- 安装测试应从以下方面考虑：
 - 安装测试应该在所有的运行环境上进行验证，如操作系统、数据库、硬件环境、网络环境等
 - 至少要在一台笔记本上进行安装测试，台式机和笔记本硬件的差别会造成其安装时出现问题
 - 安装后应执行卸载操作，检测系统是否可以正确完成任务
 - 检测安装该程序是否对其他的应用程序造成影响
 - 如有web服务，应检测会不会引起多个web服务的冲突





7.10可用性测试

- 可用性测试 (Usability Testing)
 - ▣ 对于用户友好性的测试，是指在设计过程中被用来改善易用性的一系列方法
 - ▣ 测试人员为用户提供一系列操作场景和任务让他们去完成，这些场景和任务与产品或服务密切相关
 - ▣ 通过观察来发现完成过程中出现了什么问题，用户喜欢或不喜欢哪些功能和操作方式，原因是什么，针对问题所在提出改进的建议



可用性测试方法

■ 一对一用户测试

- ▣ 一个可用性测试部分包括测试人员（主持人/助理）和一个目标用户，这个目标用户会在测试人员的陪同下完成一系列的典型任务

■ 启发式评估

- ▣ 邀请5~8名用户作为评估人员来评价产品使用中的人机交互状况，发现问题，并根据可用性设计原则提出改进方案

■ 焦点小组

- ▣ 依据群体动力学原理由6-12个参试人组成的富有创造力的小群体，在一名专业主持人的引导下对某一主题或观念进行深入讨论。焦点小组实施之前，通常需要列出一张清单，包括要讨论的问题及各类数据收集目标。小组藉由参与者之间的互动来激发想法和思考，从而使讨论更加深入、完整



可用性测试

- 测试人员应当关注的可用性问题包括
 - 过分复杂的功能或者指令
 - 困难的安装过程
 - 错误信息过于简单，例如“系统错误”
 - 语法难于理解和使用
 - 非标准的GUI接口
 - 用户被迫去记住太多的信息
 - 难以登录



可用性测试

- 测试人员应当关注的可用性问题包括
 - 帮助文本上下文不敏感或者不够详细
 - 和其他系统之间的连接太弱
 - 默认不够清晰
 - 接口太简单或者太复杂
 - 语法、格式和定义不一致
 - 没有给用户提供所有输入的清晰的认识





7.11 配置测试

- 配置测试（Configuration Testing）
 - ▣ 验证系统在不同的系统配置下能否正确工作
 - ▣ 配置包括：软件，硬件，网络等
 - ▣ 理想状况：所有生产厂家都严格遵照一套标准来设计硬件，那么所有使用这些硬件的软件就能正常运行
 - ▣ 实际情况：标准并没有被严格遵守，一般都是由各个组织或公司自行定义规范
- 配置测试的目的
 - ▣ 促进被测软件在尽可能多的硬件平台上运行。有时经常会与兼容性测试或安装测试一起进行



7.11 配置测试

- 计划配置测试时一般采用的过程
 - 确定所需的硬件类型
 - 确定哪些硬件型号和驱动程序可以使用
 - 确定可能的硬件特性、模式和选项
 - 将硬件配置缩减到可以控制的范围内
 - 明确使用硬件配置的软件的特性
 - 设计在每种配置中执行的测试用例
 - 反复测试直到对结果满意为止





7.12 文档测试

- 软件产品由可运行的程序、数据和文档组成
- 文档是软件的一个重要组成部分。在软件的整个生命周期中，会产生许多文档，在各个阶段中以文档作为前阶段工作成果的总结和后阶段工作的依据
- 软件文档的分类如下表所示

用户文档	开发文档	管理文档
用户手册、 操作手册、维护 修改建议	软件需求说明书、数据库设计说明书、概要设计说明书、详细设计说明书、可行性研究报告	项目开发计划、 测试计划、测试报告、 开发进度月报、开发 总结报告



7.12 文档测试

- 以下一些是可归于软件文档的部分
 - 包装文字、标签和不干胶条
 - 市场宣传材料、广告以及其他插页
 - 如软件的屏幕抓图、特性清单、系统要求和版权信息等
 - 授权 / 注册登记表
 - 最终用户许可协议、用来解释使用软件的法律条款
 - 安装和设置指导
 - 用户手册、联机帮助、指南和向导
 - 样例、示例和模板
 - 错误提示信息



7.12文档测试

- 从用户的角度看，文档都是软件的一部分
 - 对于严肃对待它们的用户而言，这些信息必须正确，所有这些文档都应该得到很好的检测
- 一个好的软件文档能从以下3方面提高软件产品的整体质量。
 - **提高可用性**：可用性大都与软件文档有关
 - **提高可靠性**：可靠性是指软件平稳运行的程度
 - **降低支持费用**：好的文档能够通过恰当的解释和引导帮助用户克服困难，尽可能预防这种情况发生



7.12 文档测试

- 文档测试（Documentation Testing）
 - ▣ 针对系统提交给用户的文档进行验证，目标是验证软件文档是否正确记录系统的开发全过程的技术细节。通过文档测试可以改进系统的可用性、可靠性、可维护性和安装性



7.12文档测试

- 用户文档测试的内容
 - 把用户文档作为测试用例选择依据
 - 确切的按照文档所描述的方法使用系统
 - 测试每个提示和建议，检查每条陈述
 - 查找容易误导用户的内容
 - 把缺陷并入缺陷跟踪库
 - 测试每个在线帮助超链接
 - 测试每条语句，不要想当然
 - 表现的像一个技术编辑而不是一个被动的评审者



7.12 文档测试

■ 用户文档测试的内容

- 首先对整个文档进行一般的评审，然后进行一个详细的评审
- 检查所有的错误信息
- 测试文档中提供的每个样例
- 保证所有索引的入口有文档文本
- 保证文档覆盖所有关键用户功能
- 保证阅读类型不是太技术化
- 寻找相对比较弱的区域，这些区域需要更多的解释



7.12 文档测试

■ 开发文档测试的内容

- 系统定义的目标是否与用户的要求一致
- 系统需求分析阶段提供的文档资料是否齐全
- 文档中的所有描述是否完整、清晰，准确地反映用户要求
- 与所有其他系统成份的重要接口是否都已经描述
- 被开发项目的数据流与数据结构是否足够、确定
- 所有图表是否清楚，在不补充说明时能否理解
- 主要功能是否已包括在规定的软件范围之内，是否都已充分说明
- 软件的行为和它必须处理的信息、必须完成的功能是否一致



7.12 文档测试

■ 开发文档测试的内容

- 设计的约束条件或限制条件是否符合实际
- 是否考虑了开发的技术风险
- 是否考虑过软件需求的其他方案
- 是否考虑过将来可能会提出的软件需求
- 是否详细制定了检验标准，它们能否对系统定义是否成功进行确认
- 有没有遗漏、重复或不一致的地方
- 用户是否审查了初步的用户手册或原型
- 项目开发计划中的估算是否受到了影响



7.12 文档测试

■ 开发文档测试的内容

□ 接口

- 分析软件各部分之间的联系，确认软件的内部接口与外部接口是否已经明确定义。模块是否满足高内聚低耦合的要求。模块作用范围是否在其控制范围之内

□ 风险

- 确认该软件设计在现有的技术条件下和预算范围内是否能按时实现

□ 实用性

- 确认该软件设计对于需求的解决方案是否实用

□ 技术清晰度

- 确认该软件设计是否以一种易于翻译成代码的形式表达



7.12 文档测试

■ 开发文档测试的内容

□ 可维护性

- 从软件维护的角度出发，确认该软件设计是否考虑了方便未来的维护

□ 质量

- 确认该软件设计是否表现出良好的质量特征

□ 各种选择方案

- 看是否考虑过其他方案，比较各种选择方案的标准是什么

□ 限制

- 评估对该软件的限制是否实现，是否与需求一致

□ 其他具体问题

对于文档，可测试性，设计过程等进行评估



7.12 文档测试

- 文档测试的内容

- 文档的正确性

- 不要写错软件的功能和操作，不允许文档内容前后矛盾

- 文档的完备性

- 不可以虎头蛇尾，更不许漏掉关键内容

- 文档的可理解性

- 文档能否让大众用户看得懂，能否理解术语？缩写用户是否理解？内容和主题是否一致？与文档作者密切合作，对文档仔细阅读，跟随每个步骤，检查每个图形，尝试每个示例是进行文档测试的基本方法

7.12 文档测试

■ 两类文档测试方法

- ❑ 走查：只通过阅读文档，不必执行程序
- ❑ 验证：对比文档和程序执行结果

用户文档测试技术与Bug类型的关系

	语言类 错误	版面类 错误	逻辑类 错误	一致性 错误	联机文档 功能错误
文档走查	√	√	√	√	√
数据校对			√	√	
操作流程检查			√		√
引用测试				√	
链接测试					√
可用性测试			√		
界面截图测试				√	



7.12 文档测试

■ 文档测试方法

□ 文档走查

- 熟悉软件特性的人，通过阅读文档，来检查文档的质量

□ 数据校对

- 只需检查文档中数据所在部分，不必检查全部文档。数据有：边界值、程序的版本、硬件配置、参数缺省值等

□ 操作流程检查

- 安装/卸载过程、参数配置过程、功能操作和向导功能

□ 引用测试

- 文档之间的相互引用，如术语、图、表和示例等，是Bug的





7.13 GUI测试

- 图形用户界面(Graphical User Interface, GUI) 测试是功能测试的一种表现形式，不但要考虑GUI本身的测试，也要考虑GUI所表现系统功能的测试
 - 一般来说，当一个软件产品**完成GUI设计**后，就确定了它的外观架构和GUI元素，GUI本身的测试工作就可以进行
 - 而GUI对应的**功能完成**之后，才进入功能测试阶段
 - 有时可以把上述两个阶段加以合并，待功能代码完成后一并进行。GUI测试可以采用**手工测试**方法和**自动化测试**方法完成



7.13 GUI测试

- GUI的手工测试

- ▣ 按照软件产品的文档说明设计测试用例，依靠人工点击鼠标的方式输入测试数据，然后把实际运行结果与预期的结果相比较后，得出测试结论

- 随着软件产品的功能越来越复杂，越来越完善，一般一套软件包括丰富的用户界面，每个界面里又有相当数量的对象元素，所以GUI测试完全依靠手工测试方法是难以达到测试目标的

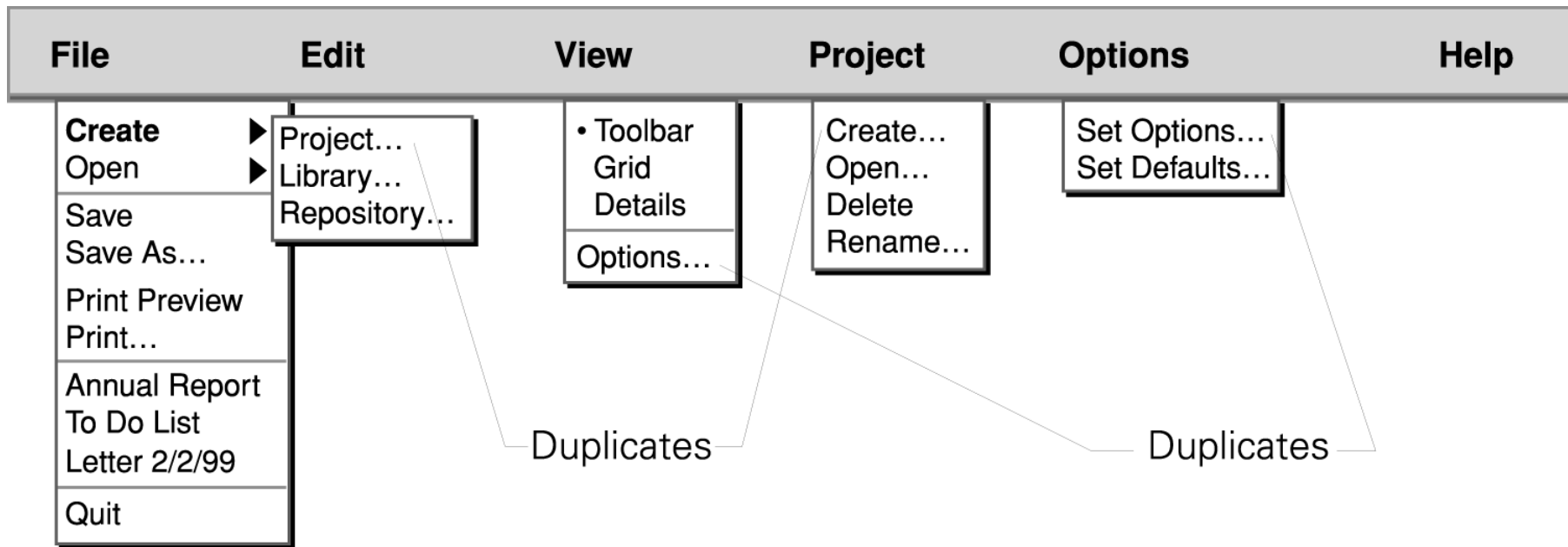


7.13 GUI测试

■ GUI的自动化测试

- 一是选择一个能够完全满足测试自动化需要的测试工具
- 二是使用编程语言，如Java、C++等编写自动化测试脚本
- 但是，任何一种工具都不能够完全支持众多不同应用的测试，常用的做法是使用一种主要的**自动化测试工具**，并且使用编程语言**编写自动化测试脚本**以弥补测试工具的不足
- 自动化测试的引入大大提高了测试的效率和准确性，而且专业测试人员设计的脚本可以在软件生命周期的各个阶段重复使用

7.13 GUI测试





7.14验收测试

- 验收测试是部署软件之前的最后一项测试，目的是确保软件准备就绪，并且可以让最终用户使用
- 验收测试的常用策略有3种
 - 正式验收测试
 - 非正式验收测试
 - Beta测试
- 策略的选择通常建立在合同需求、公司标准以及应用领域的基础上



7.14验收测试

■ 正式验收测试

- 正式验收测试是一项管理严格的过程，它通常是系统测试的延续。计划和设计这些测试的周密和详细程度不低于系统测试
- 选择的测试用例应当是系统测试中所执行测试用例的子集
- 在很多项目中，正式验收测试是通过自动化测试工具执行的



7.14验收测试

■ 非正式验收测试

- 非正式测试过程的限制不像正式验收测试中那样严格，是仅对需要**重点**解决的功能和业务进行的测试，测试内容由各测试人员决定
- 多数情况下，非正式验收测试是由**内部测试人员组织**执行的测试



7.14验收测试

■ Beta测试

- ❑ 开发人员模拟用户进行的测试，可以分别用于上面两种测试工作中
- ❑ 采用的数据、方法和测试环境完全由各测试人员决定，并决定要研究的功能、特性和任务
- ❑ 测试人员负责确定自己对于系统当前状态的接受标准

验收测试方法

- 验收测试用例的设计可以从以下几个方面来考虑：
 - ▣ 验收测试用例的组织应当面向客户，从客户使用和业务场景的角度出发，而不是从开发者实现的角度出发
 - ▣ 设计验收测试用例应当充分把握客户的关注点。在保证系统完整性的基础上，把客户关心的主要功能点和性能点作为测试的重点，其它的功能点可以忽略，避免画蛇添足
 - ▣ 验收测试用例可以适当地展示软件的某些独有特性，引导和激发客户的兴趣，达到超出客户预期效果的目的





7.15 回归测试

- 回归测试是在**软件发生变动时**保证原有功能正常运作的一种测试策略和方法
 - 不需要进行全面的测试，而是根据修改的情况进行有选择性的测试
 - “保证软件原有功能正常运作” 的含义
 - 所做的修改达到了**预期的目的**，例如缺陷得到了修改，新增加的功能得到了实现
 - 软件的修改**没有引入新的缺陷**，没有影响原有的功能实现



回归测试方法

■ 测试用例库的维护

- ❑ 为了最大限度地满足客户的需要和适应应用的要求，软件在其生命周期中会频繁地被修改和不断推出新的版本
- ❑ 为了保证测试用例库中测试用例的有效性，必须对测试用例库进行维护
- ❑ 被修改的或新增添的软件功能仅仅依靠重新运行以前的测试用例并不足以揭示其中的问题，有必要追加新的测试用例来测试这些新的功能或特性



回归测试方法

■ 回归测试包的选择

- 在软件生命周期中，即使一个得到良好维护的测试用例库也可能会变得相当庞大，使得每次回归测试都重新运行完整的测试包变得不切实际
- 采用了代码相依性分析等安全的缩减技术，就可以决定哪些测试用例可以被删除而不会让回归测试的意图遭到破坏

回归测试方法

■ 回归测试的基本过程

- ❑ 1.识别出软件中被修改的部分
- ❑ 2.从原基线测试用例库 T 中排除所有不再适用的测试用例，确定对新版本依然有效的测试用例，其结果是建立一个新的基线测试用例库 T_0
- ❑ 3.依据一定的策略从 T_0 中选择测试用例测试被修改的软件
- ❑ 4.生成新的用例集 T_1 ，用于测试 T_0 无法充分测试的部分
- ❑ 5.用 T_1 执行修改后的软件。
 - 第2和第3步测试验证修改是否破坏了现有的功能
 - 第4和第5步测试验证修改后的功能





7.16 网站测试

■ 文字测试

- 网页文字可以看做是软件文档，可以用测试文档的方法进行测试，检查用户等级、术语、内容、准确度——特别是可能过期的信息

■ 链接测试

- 链接是Web页的一个主要特征，它是在页面之间进行切换和指导用户去一些不知道地址的页面的主要手段
- 链接测试可分为三个方面
 - 所有链接是否按指示的那样确实链接到了目的页面
 - 测试所链接的页面是否存在
 - 保证网站上没有孤立的页面



7.16 网站测试

■ 图形测试

- ❑ 确保图形有明确的用途，图片或动画不能胡乱地堆放在一起，以免浪费传输时间
- ❑ 图片的大小和质量也是一个很重要的因素，一般采用JPG或GIF压缩。图片尺寸应小，但又能清楚地说明某件事情
- ❑ 检测是否所有图形正确载入和显示

■ 表单测试

- ❑ 表单是指网页上用于输入和选择信息的文本框、列表框和其他域
- ❑ 表单测试检测域的大小是否正确，数据接受是否正确，可选域是否真正可选等



7.16 网站测试

- 动态内容测试

- 动态内容是根据当前条件而发生变化的文字和图形

- 数据库测试

- 使用了数据库的Web应用系统中，一般可能出现两种故障，一是数据一致性故障，二是输出故障。前者主要是由于用户提交的表单信息不正确而引起的，后者主要是由于网络速度或程序设计等问题引起的，应分别进行测试



7.16 网站测试

■ 服务器性能和加载测试

- ▣ 流行网站每天可能要接受数百万次点击。每一次点击都要从网站的服务器下载数据到浏览器的计算机。必须找到一种方法来模拟数百万个连接和下载以测试系统的性能

■ 安全性测试

- ▣ 必须测试有效和无效的用户名和密码，检测是否可以不登陆而直接浏览某个页面等
- ▣ 检测网页是否有超时的限制
- ▣ 当使用了安全套接字时，检测加密是否正确，信息是否完整
- ▣ 检测在服务器端放置和编辑脚本等问题

