软件测试

授课教师：张剑波

授课班级：111161-2班

2019年3月

Chapter2 软件测试技术

内容提要

- 静态测试
- 动态测试
  - 经典测试方法
  - 基于质量特征的测试
  - 基于经验的测试

案例分析

- 某网校考试平台：三层B/S架构的MIS系统

  客户端/应用服务器/数据库操作层
- 测试目的：进行应用服务器的压力测试，找出应用服务器能够支持的最大客户端数
- 测试方法：按照正常业务压力估算值的1~10倍进行测试，考察应用服务器的运行情况
- 用例设计
  - 登陆48个用户，每秒登录1个用户
  - 登陆48个用户，每秒同时登录10个用户
  - 登陆48个用户，所有用户同时并发操作
  - 登陆200个用户，所有用户同时并发操作

基于质量特征的测试

- 功能测试
  - 对产品的各功能进行验证，根据功能测试用例逐项测试，检查产品是否达到用户要求的功能。
- 非功能测试（14种）
  - 性能测试
  - 压力测试
  - 安全性测试
  - 可靠性测试
  - 兼容性测试
  - …

1. 功能测试

- 一般需要在完成单元测试后、集成测试前进行，而且是针对应用系统进行各功能测试。
- 功能测试是基于产品功能说明书，在已知产品所应具有的功能上，从用户角度来进行功能验证，以确认每个功能是否都能正常使用、是否实现了产品规格说明书的要求、是否能适当地接收输入数据而产生正确的输出结果等。
- 特点：
  - 重复：不同团队、不同阶段、不同角度
  - 不是一个完全的测试：可能有缺陷遗漏

功能测试的方法

- 体系结构验证
  - 针对整个系统的行为，保证被测系统的完备性
  - 对照设计和体系结构开发来检查测试用例，进而验证体系结构的可行性。
- 业务垂直测试
  - 针对不同业务深度，对被测软件进行测试
  - 例如： 银行、保险；
  - 定制：用户（负责人、工作人员）、角色（区分不同操作：录入、审核等）

功能测试的方法（续1）

- 业务垂直测试方法
  - 模拟：在业务垂直测试模拟中，客户和测试人员要测试需求和业务流。
  - 复制：获取客户数据和过程，对被测软件进行完全定制和测试，并把定制后经过测试的软件发布给客户。
- 部署测试
  - 离场部署测试：在软件开发组织内、模拟
  - 现场部署测试：客户场地中已有的资源和环境、事务记录
- β测试：交给客户收集反馈意见，由客户完成

2. 非功能测试

- 过程与功能测试类似，但在复杂性、所需知识、所需工作量和测试重复次数等方面有更高要求。
- 对于非功能测试来说，应该尽可能减少重复，有更严格的进入/退出准则和更为周密的测试计划与测试配置、数据。
- 14种非功能性测试

（1）性能测试

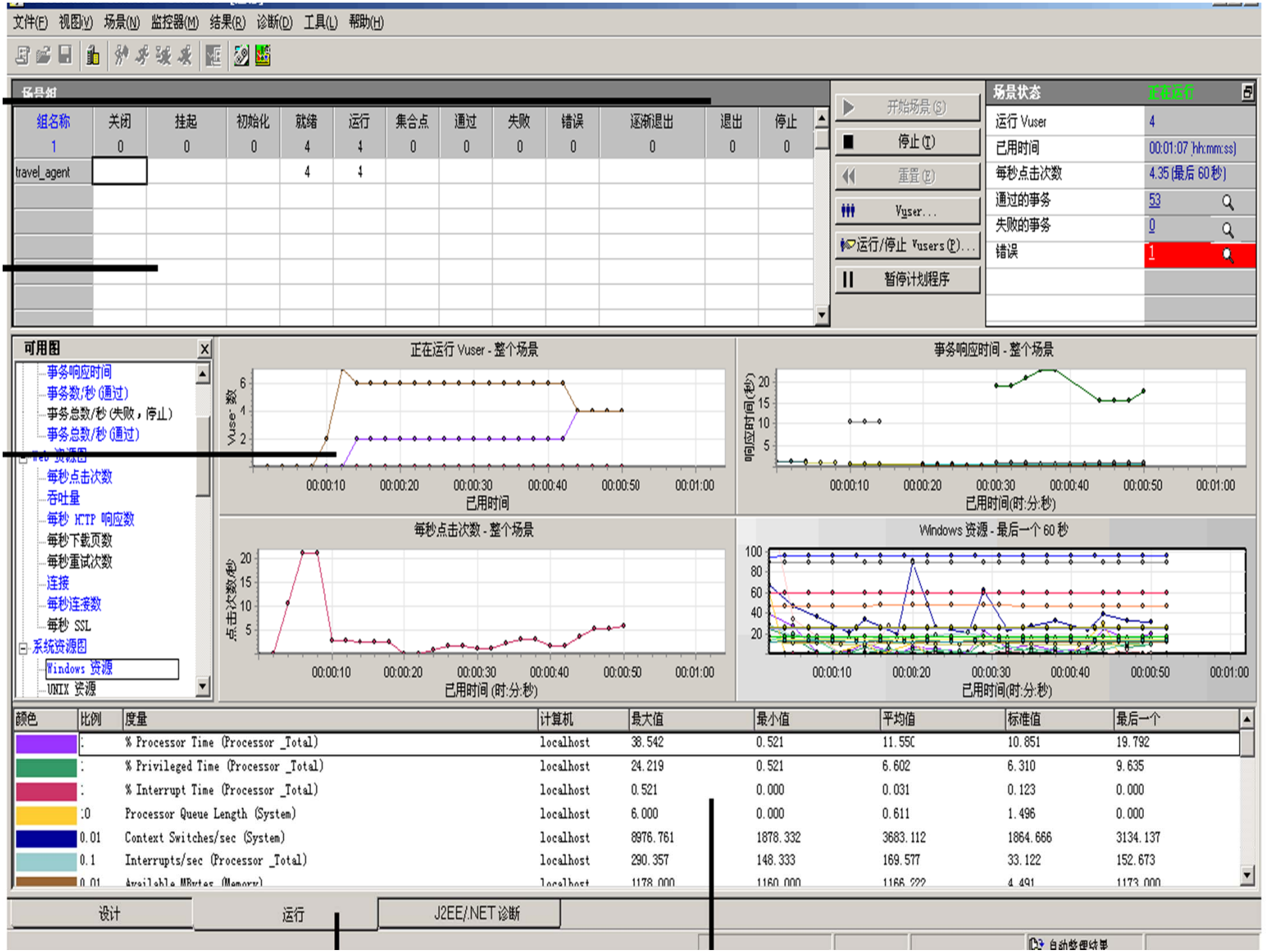
- 性能：表明软件系统或构件对于“及时性”要求的符合程度的指标
- 响应时间：对计算机系统的查询或请求开始到一个响应结束所使用的时间。
- 性能测试：检验软件是否达到需求规格说明书中规定的各类性能指标，并满足一些性能相关的约束和限制条件。
  - 评估系统的能力：负荷、响应时间
  - 识别系统中的弱点：瓶颈
  - 系统调优：改进性能

性能基准测试法

- 响应时间
  - 从应用系统发出请求开始，到客户端接收到最后一个字节数据为止所消耗的时间。
  - 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。
  - 例如：税务报账系统，录入/月
- 并发用户数
  - 同一时间段内访问系统的用户数量
  - 系统用户数
  - 同时在线用户人数

性能基准测试法（续1）

- 吞吐量
  - 单位时间内系统处理的客户请求数量
  - 每秒钟请求数
  - 每秒钟处理页面数
- 性能计数器
  - 描述服务器或操作系统性能的一些数据指标
  - 性能监控与分析：瓶颈定位
  - Windows系统的资源管理器


文件(F) 视图(V) 场景(N) 监控器(M) 结果(R) 诊断(D) 工具(L) 帮助(H)
场景组
组名称 关闭 挂起 初始化 就绪 运行 集合点 通过 失败 错误 逐渐退出 退出 停止
1 0 0 0 4 4 0 0 0 0 0 0 0
travel_agent 4 4
开始场景(S)
停止(T)
重置(E)
Vuser...
运行/停止 Vusers(P)...
暂停计划程序
场景状态 正在运行
运行 Vuser 4
已用时间 00:01:07 (hh:mm:ss)
每秒点击次数 4.35 (最后 60 秒)
通过的事务 53
失败的事务 0
错误 1
可用图
事务响应时间
事务数/秒 (通过)
事务总数/秒 (失败，停止)
事务总数/秒 (通过)
Web 资源图
每秒点击次数
吞吐量
每秒 HTTP 响应数
每秒下载页数
每秒重试次数
连接
每秒连接数
每秒 SSL
系统资源图
Windows 资源
UNIX 资源
正在运行 Vuser - 整个场景
Vuser 数
已用时间
事务响应时间 - 整个场景
响应时间(秒)
已用时间(时:分:秒)
每秒点击次数 - 整个场景
点击次数/秒
已用时间 (时:分:秒)
Windows 资源 - 最后一个 60 秒
已用时间(时:分:秒)
颜色 比例 度量 计算机 最大值 最小值 平均值 标准值 最后一个
: % Processor Time (Processor _Total) localhost 38.542 0.521 11.550 10.851 19.792
: % Privileged Time (Processor _Total) localhost 24.219 0.521 6.602 6.310 9.635
: % Interrupt Time (Processor _Total) localhost 0.521 0.000 0.031 0.123 0.000
:0 Processor Queue Length (System) localhost 6.000 0.000 0.611 1.496 0.000
0.01 Context Switches/sec (System) localhost 8976.761 1878.332 3683.112 1864.666 3134.137
0.1 Interrupts/sec (Processor _Total) localhost 290.357 148.333 169.577 33.122 152.673
设计
运行
J2EE/.NET 诊断

性能测试的阶段

- 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

性能测试的阶段（续1）

- 测试阶段
  - 做准备工作，如建立测试服务器
  - 运行测试
  - 收集数据
- 分析阶段
  - 分析结果
  - 改变系统以优化性能
  - 设计新的测试

（2）压力测试

- 在保证系统不崩溃的前提下，用于评价系统超过所描述的需求或资源限制时的情况。
- 测试人员可以故意制造过载情况、模拟资源问题等，然后观察被测软件的行为。
- 期望：随着负载的增加，被测软件的性能应该平稳地下降，但在任何时刻系统都不应该崩溃。
- 设计测试用例
  - 检验系统的能力在参数、条件等异常的情况下最高所能达到的限度
  - 一般取比平常限度高5~10倍的限度
  - 例如：100个用户并发——〉500个用户

压力测试

- 压力测试中应该模拟真实的运行环境
- 使用测试文档，参与人员一致
- 批处理的压力测试可以利用大批量的事务进行
- 被测事务中应该包括错误条件
- 压力测试中使用的事务来源
  - 测试数据生成器；
  - 由测试小组创建的测试事务；
  - 原来在系统环境中处理过的事务。

以“WEB压力测试”为例

- 重复测试
  - 确定一个操作能否持续不断地在每次执行时都正常
  - 例如：重复执行一个Web服务
- 并发测试
  - 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
  - 不一定适用于所有产品（例如：无状态服务)

WEB压力测试

- 量级增加
  - 增加某个操作的量级
  - 例如：模拟用户输入超长消息
  - 量级的确定：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、延迟时间的长短、输入速度以及输入的变化等。
- 随机变化
  - 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

（3）容量测试

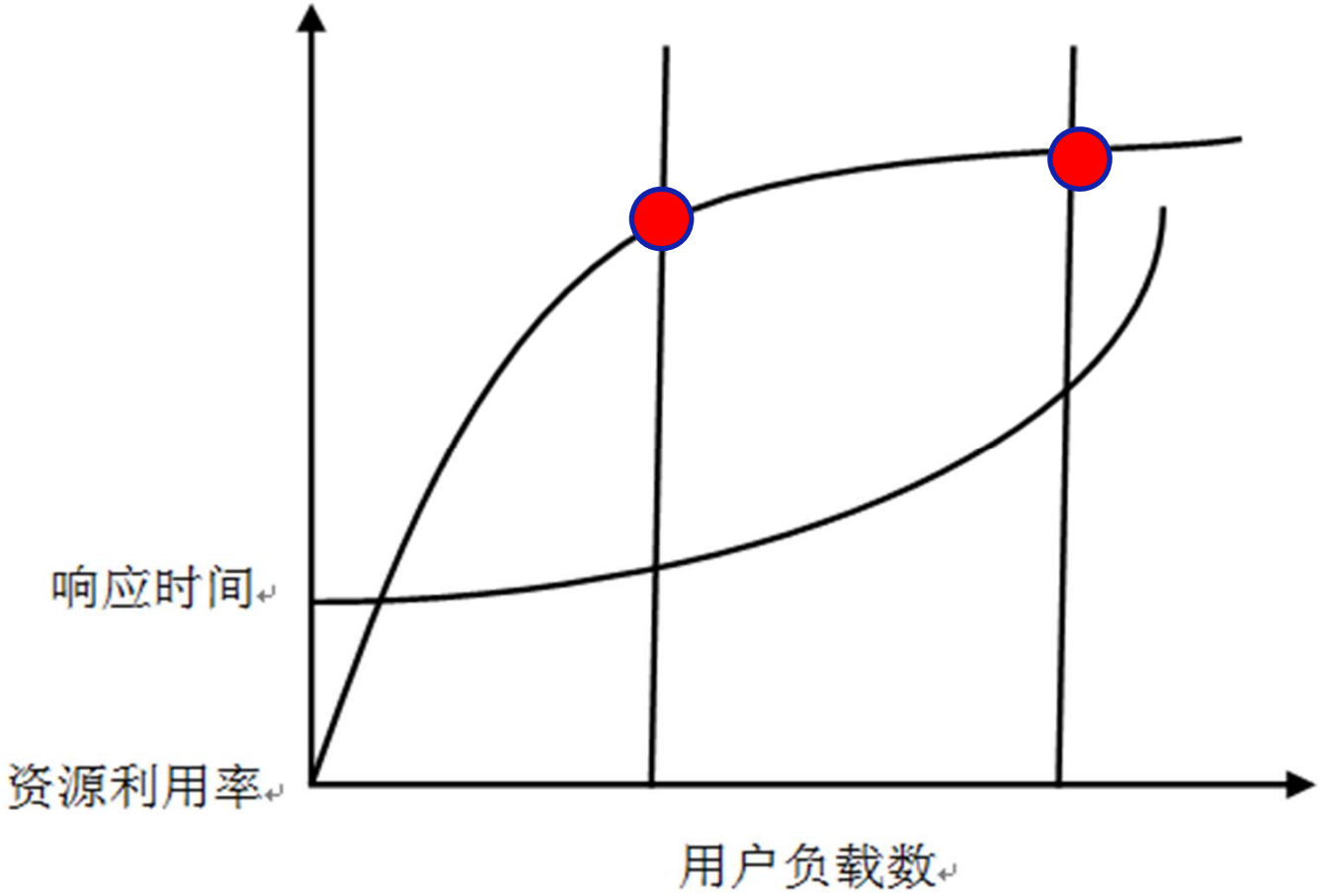
- 采用特定的手段，检测系统能够承载处理任务的极限值所进行的测试工作。
- 分析出反映软件系统应用特征的某项指标的极限值(如最大并发用户数、数据库记录数等)，确定系统在其极限值状态下还能否保持其主要功能正常运行。
- 容量测试关注的是数据方面的承受能力，显示系统可以处理的数据容量。
  - 数据库容量测试
  - 确定被测对象在给定时间内能够持续处理的最大负载或工作量

容量测试

- 压力测试的重点：发现功能性测试所不易发现的系统方面的缺陷
- 容量测试和性能测试：系统测试的主要目标内容，也就是确定软件产品或系统的非功能性方面的质量特征
- 三者的测试方法相通，在实际测试工作中，往往结合起来以提高测试效率。
- 容量测试的任务：
  - 确定被测系统数据量的极限，即容量极限

容量测试

- 找到“饱和点”：是指所有性能指标都不满足，随后应用发生恐慌的时间点



（4）健壮性测试

- 用于测试系统抵御因“设计缺陷”带来的错误的能力
- 测试重点：当出现故障时，系统是否能够自动恢复或忽略故障继续运行
  - 高可靠性
  - 从错误中恢复的能力：非正常情况下
- 评价依据
  - 通过：系统运行输入的参数，产生预期的正常结果。
  - 灾难性失效：系统需重新引导

健壮性测试的评价依据

- 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。
- 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提示。

❖ 测试用例设计

- 故障插入测试、变异测试和错误猜测法
- 构造一些不合理的输入来引诱软件出错

（5）安全性测试

- 检查系统对非法侵入的防范能力
- 目的：发现软件系统中是否存在安全漏洞
- 系统安全设计的准则：使非法侵入的代价超过被保护信息的价值，从而令非法侵入者无利可图。
  - 黑客为非法入侵花费的代价(时间、费用、危险等因素)高于得到的好处
- 安全性两个层次：
  - 应用程序级别：对数据或业务功能的访问
  - 系统级别：对系统的登录或远程访问

1）功能验证

- 功能验证：采用黑盒测试方法，对涉及安全的软件功能进行测试，验证其功能是否有效
  - 用户管理模块、权限管理模块、加密系统、认证系统等
  - 典型的问题包括：
    - 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被适当地处理?
    - 无效的或者超出范围的指令是否被检测并且被适当地处理?
    - 错误和文件访问是否适当地被记录?
    - 是否有变更安全性表格的过程?

功能验证

- 功能验证的典型问题还包括：
  - 系统配置数据是否能正确保存，系统出现故障时是否能恢复?
  - 系统配置数据能否导出，并在其他机器上进行备份?
  - 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?

功能验证

- 功能验证的典型问题还包括：
  - 低级别的用户是否可以操作高级别用户命令?
  - 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，超时的时间设置是否合理，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?

功能验证示例

- SQL注入测试
  - 就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。
  - 具体来说，它是利用现有应用程序，将（恶意）的SQL命令注入到后台数据库引擎执行。它可以通过在Web表单中输入（恶意）SQL语句，得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。
  - 主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。

SQL注入测试步骤

- 首先找到带有参数传递的URL页面，如搜索页面、登录页面、提交评论页面等等
  - http://domain/index.asp?ID=10
- 对 于未明显标识在URL中传递参数的，可以通过查看HTML源代码中的“FORM”标签来辨别是否还有参数传递
  - 在<FORM> 和</FORM>的标签中间的每一个参数传递都有可能被利用

```
<form id="form_search" action="/search/" method="get">
<div>
<input type="text" name="q" id="search_q" value="" />
<input name="search" type="image" src="/media/images/site/search_btn.gif" />
<a href="/search/" class="fl">Gamefinder</a>
</div>
</form>
```

SQL注入测试步骤（续1）

- 其次，在URL参数或表单中加入某些特殊的SQL语句或SQL片断
  - 如在登录页面的URL中输入
    - http://domain/index.asp?username=HI' or 1=1--
  - 根据实际情况，SQL注入请求可以使用以下语句：
    - ' or 1=1- -
    - " or 1=1- -
    - or 1=1- -
    - ' or 'a'='a
    - " or "a"="a
    - ') or ('a'='a

SQL注入测试步骤（续2）

- 在登录进行身份验证时，通常使用如下语句来验证：
  - sql=select * from user where username='username' and pwd='password'
  - 如 输入http://duck/index.asp?username=**admin' or 1='1**&pwd=**11**，SQL语句会变成：
  - sql=select * from user where username='**admin**' **or 1='1'** and password='**11**'
  - '与admin前面的 '组成了一个查询条件，即username='admin'，后续语句将按下一个查询条件来执行。
  - 接下来是OR查询条件，OR是一个逻辑运算符，在判断多个条件的时候，只要一个成立，则等式就成立，后面的and就不再进行判断。即**绕过了密码验证**，只用用户名就可登录

SQL注入测试步骤（续3）

- 另一个例子
  - 输入http://duck/index.asp?username=**admin'--**&pwd=**11**
  - SQL语 句会变成：sql=select * from user where name='**admin' --**' and password='11',
  - ’与admin前面的‘组成了一个查询条件，即username=‘admin’，接下来的语句将按下一个查询条件来执行； 接下来是“--”查询条件， “--” 是忽略或注释，上述通过连接符注释掉了后面的密码验证

- 最后，验证是否能入侵成功或是出错的信息是否包含关于数据库服务器的相关信息；如果能，说明存在SQL安全漏洞。

如何防止SQL注入

- 从应用程序的角度
  - 转义敏感字符及字符串
    - SQL的敏感字符包括“exec”, ”xp_”, ”sp_”, ”declare”, ”Union”, ”cmd”, ”+”, ”//”, ”..”, ”;”, ”‘”, ”--”, ”%”, ”0x”, ”><=!-*/()|” 和”空格”
  - 屏蔽出错信息：阻止攻击者知道攻击的结果
  - 在服务端正式处理之前提交数据的合法性
    - 合法性检查主要包括三项：数据类型、数据长度、敏感字符的校验
    - 最根本的解决手段：在确认客户端的输入合法之前，服务端拒绝进行关键性的处理操作

如何防止SQL注入（续）

- 从测试人员的角度
  - 在程序开发前(即需求阶段)， 有意识地将安全性检查应用到需求测试中
  - 例如，对一个表单需求进行检查时，一般检验以下几项安全性问题:
    - 需求中应说明表单中某一field的类型,长度以及取值范围(主要作用就是禁止输入敏感字符)
    - 需求中应说明若超出表单规定的类型,长度以及取值范围，应用程序应给出不包含任何代码或数据库信息的错误提示
  - 在执行测试过程中，也需要对上述两项内容进行测试

2）漏洞扫描

- 漏洞扫描：通过使用漏洞扫描器，系统管理员能够发现所维护的信息系统存在的安全漏洞
- 漏洞扫描器：一种能自动检测远程或本地主机安全性弱点的程序
  - 主机漏洞扫描器：系统本地运行，如Cops
  - 网络漏洞扫描器：基于网络远程，如satan

3）模拟攻击试验

- 模拟攻击试验：设计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
- 在数据处理与数据通信环境中，特别关注：
  - 冒充
  - 重演
  - 消息篡改
  - 服务拒绝
  - 内部攻击
  - 外部攻击
  - …

可能的攻击方法

- 冒充：特权很少的实体为了得到额外的特权，可能使用冒充成为具有这些特权的实体
  - 口令猜测
  - 缓冲区溢出：服务程序中strcpy等字符串函数
- 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
  - 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（把它当作其他实体）

可能的攻击方法（续1）

- 消息篡改：数据所传送的内容被改变而未被发觉，并导致非授权后果
  - DNS高速缓存污染
  - 伪造电子邮件：SMTP
- 服务拒绝：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
  - Ping of Death
  - Teardrop
  - UDP Flood
  - SYN Flood
  - …

可能的攻击方法（续2）

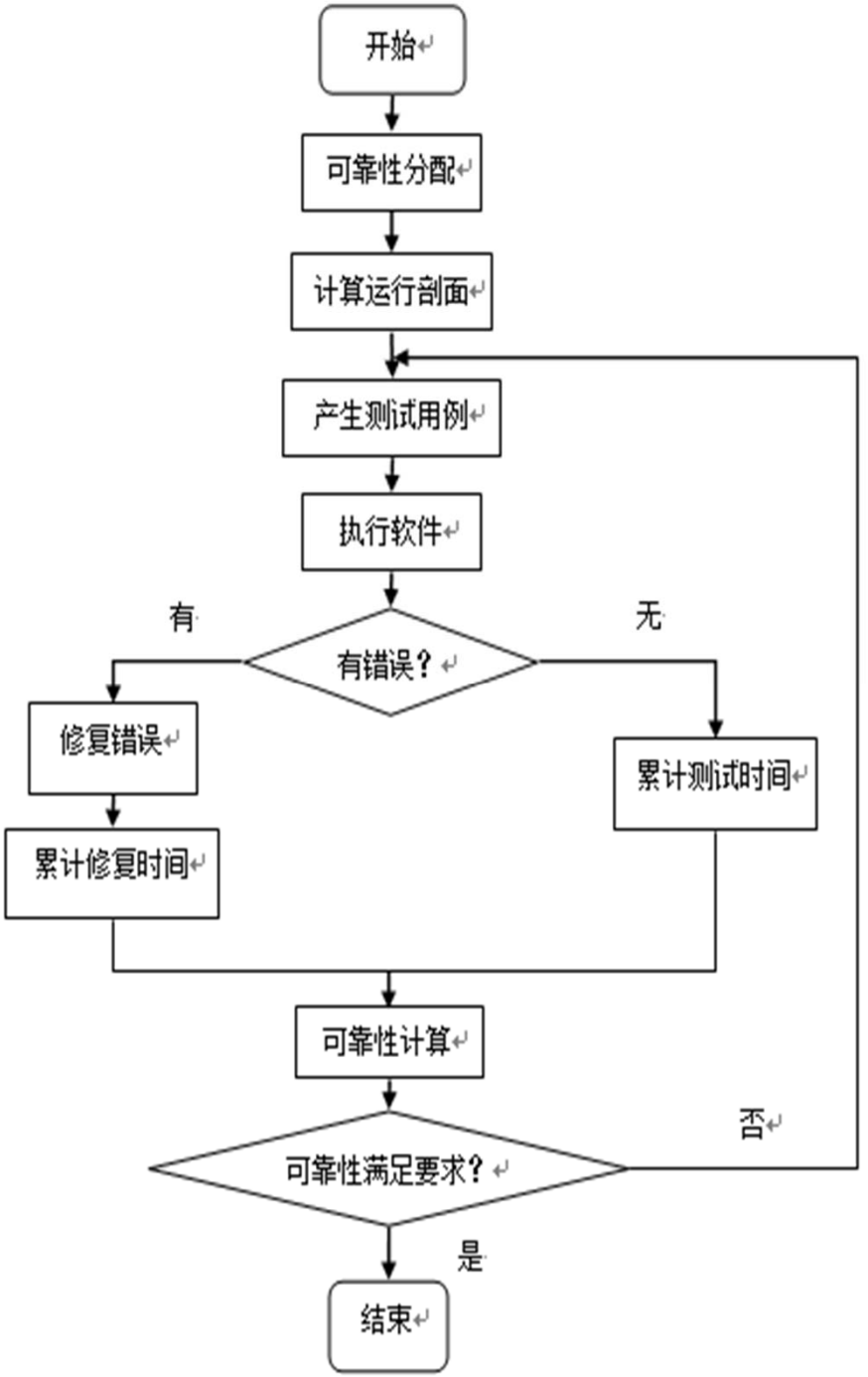
- 内部攻击：系统的合法用户以非故意或非授权方式进行操作
  - 保护方法：数据流加密、强口令、多级权限控制、防火墙、安全日志记录等
- 外部攻击
  - 搭线、截取、冒充（用户或系统的组成部分）
- 陷阱门：系统的实体发生改变，致使一个攻击者能对命令或对预定的事件或事件序列产生非授权的影响
- 特洛伊木马

4）侦听技术

- 在数据通信或数据交互过程对数据进行截取分析的过程
- Capture：网络数据包的捕获技术
- 安全测试：用于对网络加密进行验证
- 采用技术
  - 密码学的应用
  - 信任管理和口令认证
  - 客户端安全性
  - 安全控制/构架
  - …

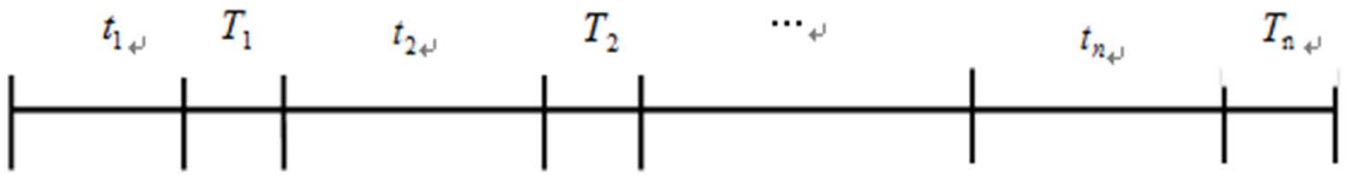
（6）可靠性测试

- 计算出软件的可靠性
- 判断软件可靠性是否达到了规定的指标要求
- 还可以给出软件测试应该何时结束？



描述软件可靠性的参数

假设系统S投入测试或运行一段时间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，…，t_n是系统正常的工作时间，T_1，T_2，…T_n是维护时间，如图4-13所示。



故障率（风险函数）：$\lambda = \dfrac{\text{总失效次数}}{\text{总工作时间}} = \dfrac{n}{\sum_{i=1}^{n} t_i}$

λ的单位是FIT，1FIT=10^{-9}/小时

维修率：$\mu = \dfrac{\text{总失效次数}}{\text{总维护时间}} = \dfrac{n}{\sum_{i=1}^{n} T_i}$

描述软件可靠性的参数（续1）

美国国家标准ANSI/AIAA R-013-1992要求最合理的故障率大约在10^{-4} /小时左右，对于安全用计算机，推荐的MTBF大约为1000~10000小时。

平均无故障时间：$MTBF=\frac{\text{总失效次数}}{\text{总工作时间}}=\frac{\sum_{i=1}^{n} t_i}{n}=\frac{1}{\lambda}$

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

有效度：$A=\frac{\text{总工作时间}}{\text{总工作时间}+\text{总维护时间}}=\frac{MTBF}{MTBF+MTTR}=\frac{\mu}{\lambda+\mu}$

可靠性：$R(t)=e\,e^{-\int_0^t \lambda(t)dt}$

（7）恢复性测试

- 恢复性测试：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。
  - 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
  - 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
- 备份测试：是恢复性测试的一个补充。目的：是验证系统在发生软件或者硬件失败时备份数据的能力。

（8）协议一致性测试

- 针对：目前的网络协议基本上是以自然语言描述的文本，实现者对于协议文本的不同理解以及实现过程中的非形式化因素等都会导致不同的协议实现，有时甚至是错误的协议实现
- 协议测试是一种黑盒测试，它按照协议标准，通过控制观察被测协议实现的外部行为，对其进行评价。
- 研究方向
  - 一致性测试：ISO 9646针对OSI协议
  - 互操作性测试
  - 性能测试

（9）兼容性测试

- 指检查软件之间是否能够正确地进行交互和共享信息
- 需要解决以下问题：
  - 如何测试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。
  - 采用什么软件行业标准或规范，以及如何达到这些标准和规范的条件？
  - 被测软件如何与其他平台、其他软件交互或共享信息？

（10）安装测试

- 不仅需要考虑在不同的操作系统上运行，而且还要考虑与现有软件系统的配合使用问题
- 需要注意的问题：
  - 参照安装手册中的步骤进行安装，进行安装过程中所有的默认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(继电等)要进行测试。

安装测试

- 还需要注意的问题：
  - 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。
  - 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会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。

（11）可用性测试

- 可用性：指产品在特定使用环境下为特定用户用于特定用途时所具有的有效性、效率和用户主观满意度。
- 可用性测试:是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。
  - 与实用性相比，可用性重视了人的因素，重视了产品是要被最终用户使用的。
  - 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，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，针对问题提出改进的建议。

可用性测试

- 包含内容
  - 任务操作的成功率
  - 任务操作效率
  - 任务操作前的用户期待
  - 任务操作后的用户评价
  - 用户满意度
  - 各任务出错率
  - 二次操作成功率
- 测试文档
  - 日程安排
  - 用户协议
  - 测试脚本
  - 测试前问卷
  - 测试后问卷
  - 任务卡片
  - 测试过程检查文档
  - 过程记录文档
  - 测试报告

可用性测试

- 测试人员应当关注的可用性问题：
  - 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。
  - 帮助文本上下文不敏感或者不够详细。

可用性测试

- 测试人员还应当关注的可用性问题：
  - 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认识。

可用性测试方法

- 一对一用户测试
  - 目标用户会在测试人员的陪同下完成一系列的典型任务，了解用户的操作过程、思维过程
- 启发式评估
  - 发现用户界面设计中的可用性问题（75%）
  - 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，提出改进方案

可用性测试方法（续）

焦点小组

- 用于产品功能的界定、工作流程的模拟、用户需求的发现、用户界面的结构设计和交互设计、产品原型的接受度测试、用户模型的建立等
- 由6~12个参试人组成的富有创造力的小团体，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
- 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
- 通常需要2周的前期沟通和准备，1个星期测试，2个星期提交分析报告

（12）配置测试

- 验证系统在不同的系统配置下能否正确工作，这些配置包括：软件、硬件和网络等。
- 目的就是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。
- 提前准备工作
  - 分离配置缺陷：判断来源于哪种硬件配置？
  - 计算配置测试工作量
  - 例如：3D游戏软件，对各种图形显卡、声卡、网卡和打印机进行配置测试

配置测试的用例选择需考虑问题

- 确定所需的硬件类型
- 确定有哪些厂商的硬件、型号和驱动程序可用
- 确定可能的硬件特性、模式和选项
- 将确定后的硬件配置缩减到可控制的范围
- 明确与硬件配置有关的软件的唯一特性
- 在每种配置中执行测试
- 反复测试直到小组对结果满意为止

（13）文档测试

- 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开发全过程的技术细节。
- 用户文档
  - 用户手册，操作手册，维护修改建议
- 开发文档
  - 可行性研究报告
  - 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

用户文档测试

- 把用户文档作为测试用例选择依据
- 确切地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
- 检查所有的错误信息
- 测试每个在线帮助超级链接
- 测试文档中提供的每个样例
- 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
- …

开发文档测试

- 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。
- 与所有其他系统成分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
- 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
- 是否考虑了开发的技术风险
- 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。

开发文档测试（续1）

- 确认软件的内部接口与外部接口是否已经明确定义
- 风险：即确认软件设计在现有的技术条件和预算范围内是否能按时实现。
- 实用性：即确认软件设计对于需求的解决方案是否实用。
- 技术清晰度：即确认软件设计是否以一种易于翻译成代码的形式表达。
- 可维护性：从软件维护的角度出发，确认软件设计是否考虑了方便将来的维护。
- 质量：即确认软件设计是否表现出良好的质量特征

文档测试方法

- 文档走查
- 数据校对
  - 检查的数据主要有：边界值、软件版本、硬件配置、参数默认值等。
- 操作流程检查
- 引用测试
- 链接测试：电子文档中的超级链接
- 可用性测试：文档的可用性
- 界面截图测试

（14）GUI测试

- 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
  - 手工测试
  - 自动化测试
- GUI测试用例的选择规范
  - 窗口相关标准
  - 下拉式菜单和鼠标
  - 数据项

本节小结

- 功能测试
- 非功能测试（14种）
  - 性能测试
  - 压力测试
  - 安全性测试
  - 可靠性测试
  - 兼容性测试
  - 可用性测试
  - 文档测试
  - …