软件测试基础与实践

Software Testing: Foundations and Practices

第5讲 系统测试 确认测试 回归测试

教师：汪鹏 廖力

软件工程专业 主干课程

本讲内容

系统测试	功能测试：业务、部署、 **Alpha、Beta、……** 非功能系统测试: 可靠性、压力、 性能、互操作、……
确认测试	确认准则　测试用例选择 确认测试执行
回归测试基础	基本概念、波及效应 回归测试工具

一 系统测试

1. 功能测试和非功能测试
2. 系统测试方法

系统测试概述

系统测试概念

定义：

对完整集成后的产品和解决方案进行测试，用来评价系统对具体需求规格说明的功能和非功能的符合性的测试

目的/作用：

1.发现可能难以直接与模块或接口关联的缺陷

2.发现产品设计、体系和代码的基础问题（产品级缺陷）

系统测试概述

系统测试概念

引入时机：

集成测试之后（基础的程序逻辑错误和缺陷已更正后）

实施人员：

独立测试团队
（引入独立视角，有助于发现遗漏缺陷）

特点：

是既测试产品功能也测试产品非功能的唯一测试阶段

系统测试概述

系统测试的实施原因

1. 在测试中引入独立视角

独立测试团队；直接向高层报告

2. 在测试中引入客户视角

消除对产品的偏见；验证完整产品

3. 在测试中模拟客户使用环境

正式、完备和现实的测试环境

系统测试概述

■系统测试的实施原因

4. 测试产品功能和非功能的问题

产品交付给客户之前发现各种残余产品缺陷的最后机会。

5. 建立对产品的信心

把握系统测试发现缺陷的度。

6. 分析和降低产品发布的风险

对发现的缺陷进行分析和分类，修复高影响风险的缺陷。

7. 保证满足所有需求，产品具备交付确认测试条件

系统测试概述

系统测试vs. 组件和集成测试

1.系统测试是对其他阶段测试的互补性测试。

系统测试阶段通过明确关注客户对早期测试阶段形成补充。

2.组件和集成测试主要关注技术和产品实现；而客户场景和使用模式是系统测试的基础。

系统测试阶段有助于将关注点从产品开发团队转移到客户及其对产品的使用上。

系统测试概述

- 系统测试=功能测试+非功能测试

功能测试VS非功能测试

	功能测试	非功能测试
作用/关注点	验证产品功能和特性，代码缺陷检测	验证产品质量参数
范围	所有测试阶段	系统测试
方法	检查预期结果的简单步骤	采集并分析大量数据
预备知识	产品和领域	产品、领域、设计、体系结构、分析技能

系统测试概述

- 系统测试=功能测试+非功能测试

功能测试VS非功能测试

	功能测试	非功能测试
用例可重复性	可重复多次	只在未通过时和针对不同配置时重复
配置	对一组测试用例建立配置一次	每个测试用例都可能改变配置
影响结果因素	产品实现	产品实现、资源、配置
未通过原因	代码缺陷	体系结构、设计、代码的缺陷
工作量	建议50%:50% ~ 30%:70%	

功能系统测试

廖 力

功能系统测试

■为何还要进行功能系统测试？

1.在系统测试阶段，主要进行产品级的功能测试。

2.进行功能系统测试，可以覆盖前面各阶段均遗漏掉的测试。

3.在各个阶段，由不同的人不同的团队进行功能测试，有助于从不同视角发现新的缺陷。

功能系统测试

功能系统测试方法

1. 设计/体系结构测试
2. 业务垂直测试
3. 部署测试
4. Alpha测试和Beta测试
5. 符合性的认证、标准和测试

功能系统测试

设计/体系结构测试

原理：

对照设计和体系结构开发和检查测试用例，从而整理出产品级测试用例。

集成测试用例关注模块或组件间交互，而功能系统测试用例关注整个产品的行为。

功能系统测试

设计/体系结构测试

原理：

从产品角度出发，检查软件体系结构的总体拓扑结构、构件模型的结构是否合理？构件的接口和连接件的角色之间是否匹配？是否满足相应的约束？等

功能系统测试

设计/体系结构测试

方法：

- 体系结构的静态测试
  - 即体系结构分析
  - 对体系结构的特征进行建模、分析，如：对类定义的一致性分析
- 体系结构的动态测试

功能系统测试

设计/体系结构测试

测试用例特征	建议
测试用例关注代码逻辑、数据结构和产品单元	单元测试
测试用例关注组件接口	集成测试
测试用例关注的是不能为客户所看到的产品实现	单元测试/集成测试
测试用例综合了客户使用和产品实现	系统测试

功能系统测试

■业务垂直测试

原理：

针对不同业务纵深的产品，根据业务定制测试用例，验证业务运作和使用。

应用范围：

通用的工作流自动化系统在不同商业领域的应用。

方法：

模拟：测试需求和业务流。

复制：获取客户数据和过程，针对特殊业务进行定制。

功能系统测试

■业务垂直测试

定制：

改变系统的一般工作流，以适用于不同业务纵深。

术语：

尽量使用各业务领域的专属名词。

例如：电子邮件 **vs.** 索赔申请（保险）**vs.** 贷款申请（银行）**vs.** 血浆需求（血库）

功能系统测试

部署测试

验证系统是否能满足客户的部署需求。

目的：

特定产品版本短期内是否能成功使用

离场部署：

在产品开发组织内进行，以确保客户部署需求的（模拟）部署测试。

功能系统测试

部署测试

现场部署（离场部署的扩展）：

- 现场部署是指在客户场地中的资源和环境都发布后，实施的一种部署方案。
- 第1阶段：采集实际系统真实数据，建立镜像测试环境，重新执行用户操作。
  - 可以在不影响用户的情况下了解产品功效，并可以用智能记录器来记录并比对事务处理。
- 第2阶段：引入新产品，进行新业务操作，同时比对事务处理情况，以确定新系统是否能够替代老系统。

功能系统测试

Alpha测试和Beta测试

Alpha测试：

定义：用户在开发环境下进行的受控测试

特点：不由程序员或测试员完成，但开发者会在现场

功能系统测试

Alpha测试和Beta测试

在Alpha测试达到一定程度后进行Beta测试

Beta测试：

定义：用户在实际使用环境下进行测试
一种可以把待测产品交给客户收集反馈意见的机制

特点：开发者通常不在现场

挑战：客户数量；客户充分了解产品

功能系统测试

符合性的认证、标准和测试

产品需要通过主流硬件、操作系统、数据库和其他基础设施构件上进行的验证，并符合相关法规和行规。

主流基础设施：操作系统、硬件、数据库...

约定和法律要求：质量行业标准、法规、技术领域标准...

- 例如：
  - 符合FDA食品药品管理法。
    - 医疗科学方面的产品需进行充分测试。
  - 508可获得性指南
    - 残障人士使用的产品要进行充分测试。
  - SOX（Sarbanes-Oxley）法案
    - 经济类产品要检查所有事务，防止经济诈骗。

非功能系统测试

廖 力

非功能系统测试

非功能系统测试方法

非功能测试用于验证系统的质量因素。

非功能测试要求理解产品行为、设计和体系结构；针对不同配置和资源对产品进行测试，并收集和分析相应数据。评判产品的质量。

非功能系统测试

■非功能系统测试方法

非功能测试的最大挑战：设置配置

原因：

1. 难以预测客户使用的环境
2. 对配置进行组合测试的代价太高
3. 建立测试环境成本高
4. 很难准确预测客户使用的数据

设置配置的两种方法：模拟环境 和 真实客户环境

非功能系统测试

非功能系统测试方法

1. 可伸缩性测试/容量测试
2. 可靠性测试
3. 压力测试
4. 互操作性测试/兼容性测试
5. 可使用性与易获得性测试
6. 国际化测试
7. 性能测试
8. 安全性测试

非功能系统测试

■可伸缩性测试/容量测试

系统的接受或容纳能力，也可以指某项功能的最大承受能力

目标：

确认产品参数的最大值，确定产品的主要瓶颈。

例如：确认多少客户机可以同时登录到服务器上执行某些操作。

非功能系统测试

■可伸缩性测试/容量测试 （Volume testing）

测试方法：

1.设计和体系结构会提出理论值，客户会提出期望的最大能力值

2.首先验证二者较低的参数；若设计不满足合理的客户期望值，直接返工；否则逐步更改参数直到最大能力

测试失效：

系统不能响应或者系统崩溃

非功能系统测试

■可伸缩性测试/容量测试 （Volume testing）

失效改正：

1.对容量缺陷的改正可能会对产品的其它非功能特性产生影响。
2.不能在牺牲其它质量因素的前提下获得容量的改进。
3.测试失效可能意味着产品策划不够，对产品行为也不够了解，需要寻求其他技术方案以提高系统容量。

测试注意点：

此类测试需要大量资源并且开销大。

非功能系统测试

■可靠性测试

可靠性测试：

在一定时间段内持续不断地测试软件产品，评价该产品在规定时间和条件下，维持其正常的功能操作、性能水平的程度。

应用领域：

银行交易系统、铁路交通管制系统、导弹防御系统...

测试目标：

评价产品在给定条件下在给定时间段内或很多轮迭代内，执行所要求功能的能力

非功能系统测试

可靠性测试

测试失效：

由于重复执行某些操作而引起错误，如：内存泄露（典型）、系统无响应

例子：

持续查询数据库72小时

执行登陆操作10000次

可恢复性：

系统快速从错误状态中恢复到正常状态的能力或时间

容错性

对外界错误（如：误操作）隔离的能力

非功能系统测试

可靠性测试

可靠性指标：

平均失效等待时间：连续失效之间的平均间隔时间

失效率（风险参数）：单位时间内发生的失效次数

平均发现下k个缺陷的时间：预测下K个缺陷的平均时间

非功能系统测试

可靠性测试

经可靠性测试产品特征：

1.重复执行事务操作没有或极少有错误
2.零宕机
3.资源的优化使用
4.具有一致的性能和时间响应
5.重复执行事务操作没有副作用

非功能系统测试

■负载测试 (Load Testing)

负载：

并发用户数、上载文件大小、数据库的记录数…

负载测试：

通过模拟实际软件系统所承受的负载条件、改变系统负载大小和负载方式来发现系统中所存在的问题。

例如：

逐渐增加模拟用户数来观察系统响应时间和数据吞吐量、系统占用的资源等，以发现系统存在的性能瓶颈、内存泄漏等问题

非功能系统测试

负载测试 vs. 容量测试:

- 负载测试通过改变系统负载方式、增加负载等来发现系统的性能问题。

- 容量测试是在预先定义（或分析）的极限值下，看系统是否能正常工作。考虑到预期业务处理量的增加，容量测试还将确定测试对象在给定时间内能够持续处理的最大负载

非功能系统测试

压力测试 (Stress Testing)

压力来自于系统的极端情况：

不足的内存、有限的网络带宽、磁盘读写被占用……

目的：

1.评价系统超过所描述的需求或资源限制时的情况，保证系统不崩溃

2. 有助于了解系统在极端和现实环境中的行为，期望随着负载增加产品性能平稳下降，但任何时刻都不应该崩溃

非功能系统测试

压力测试 (Stress Testing)

压力测试方法：

- 负载通过各种方式逐步增加，产品会达到一个压力点，此时有些事务会因为资源不可用而失效，超过这个压力点失效率会增加。
- 略微降低负载至压力点以下，观察产品是否能恢复到正常状态，失效率是否降低。
- 升降负载2-3次，观察产品行为是否与预期一致。
- MTTR（平均恢复时间）：由失效恢复的平均时间

非功能系统测试

压力测试

压力测试用例选择：

1.重复性测试用例：如反复执行的操作
2.并发性测试用例：如多用户并发操作
3.负载量级要对系统形成压力
4.随机变化的输入和量级以对系统产生压力

压力测试可发现的典型缺陷：

内存泄露、死锁等并发和同步问题

压力测试可以看作负载测试的一种，压力测试更强调持续加压，测试系统极限值。

非功能系统测试

性能测试

为了获取或验证系统性能指标而进行的测试。

性能测试评价响应时间、吞吐率和系统的使用情况，执行所要求的功能以对同一产品的不同版本或不同的竞争产品进行比较

事先有明确的性能指标，在严格的测试环境和所定义的负载下进行，获得不同负载情况下的性能指标数据。

性能测试可使用负载测试的技术和工具

非功能系统测试

对比分析

1. 可靠性测试在保持恒定的负载条件，在一定时间内不断持续测试产品，根据结果评判产品的可靠性状况。

2. 压力测试中负载以各种方式逐步增加，发现压力点，保证系统在超过压力点的情况下不崩溃。

3. 负载测试通过观察不同负载下系统响应时间和数据吞吐量、系统占用的资源等，以发现系统存在的性能瓶颈、内存泄漏等问题。

非功能系统测试

对比分析

4. 容量测试确定在不同配置下系统的最大能力，有助于确定产品中的瓶颈。

5. 性能测试通过自动化测试工具模拟多种正常、峰值及异常负载条件来对系统的各项性能指标进行测试。

案例：　针对某公司办公自动化（OA）系统的负载压力测试，采用专业的负载压力测试工具来执行测试。系统采用B/S架构，服务器是一台PC Server（4路2.7GHz 处理器，4GB内存），安装的平台软件包括Microsoft Internet Information Server5.0，ASP.NET，SQLServer 2000。

使用2台笔记本电脑安装测试工具模拟客户端执行“登录”业务操作。

1）测试系统分别在2M、4M网络宽带下，能够支持用户登录的最大并发用户数；

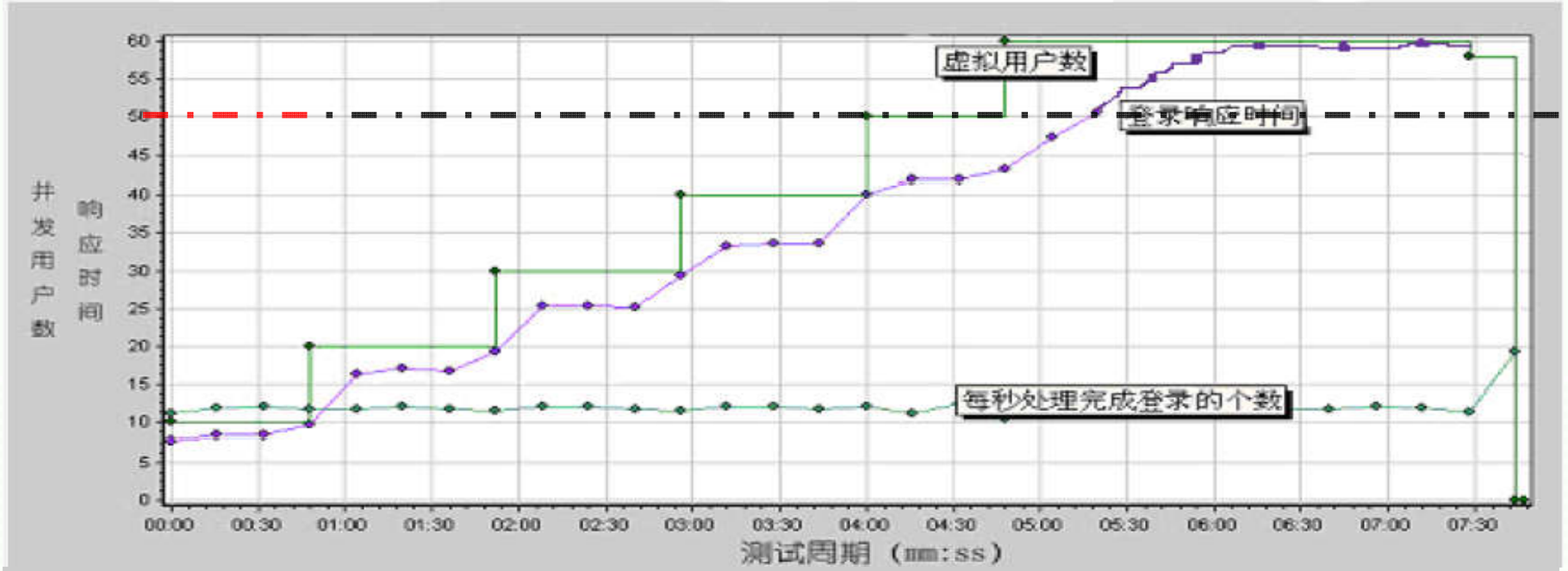
2）测试服务器的吞吐量（即：每秒可以处理的交易数），主要包括服务器CPU平均使用率达到85%时系统能够支持的最大吞吐量和服务器CPU平均使用率达到100%时系统能够支持的最大吞吐量。

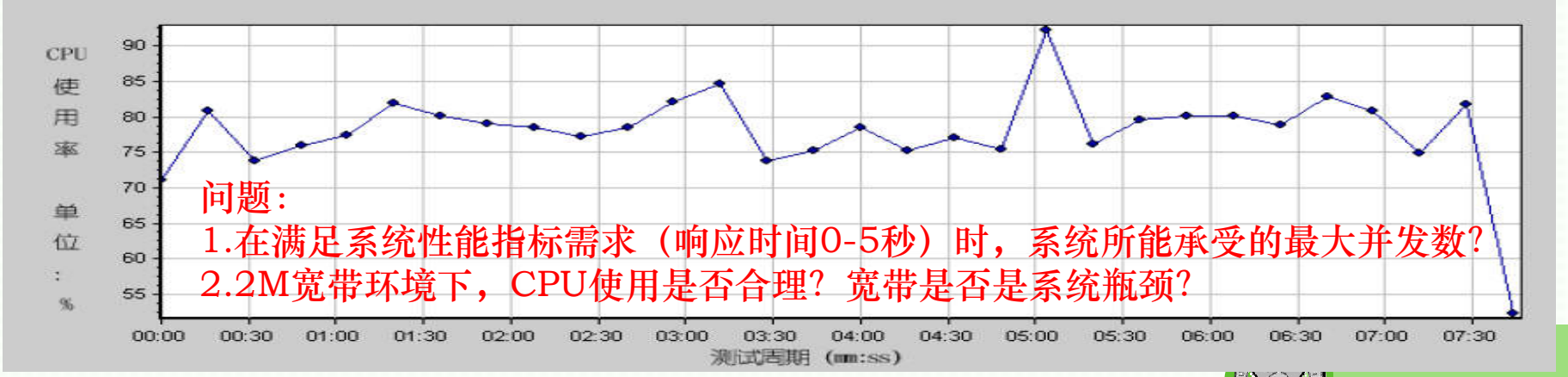
性能需求：指标“响应时间”合理范围为0~5秒。

1）设计出两种场景2M网络和4M网络环境下进行模拟测试。

2）其中选定登录业务进行测试，加压策略采取逐步加压的方式。

注：图中登录响应时间的纵坐标单位是**0.1**秒



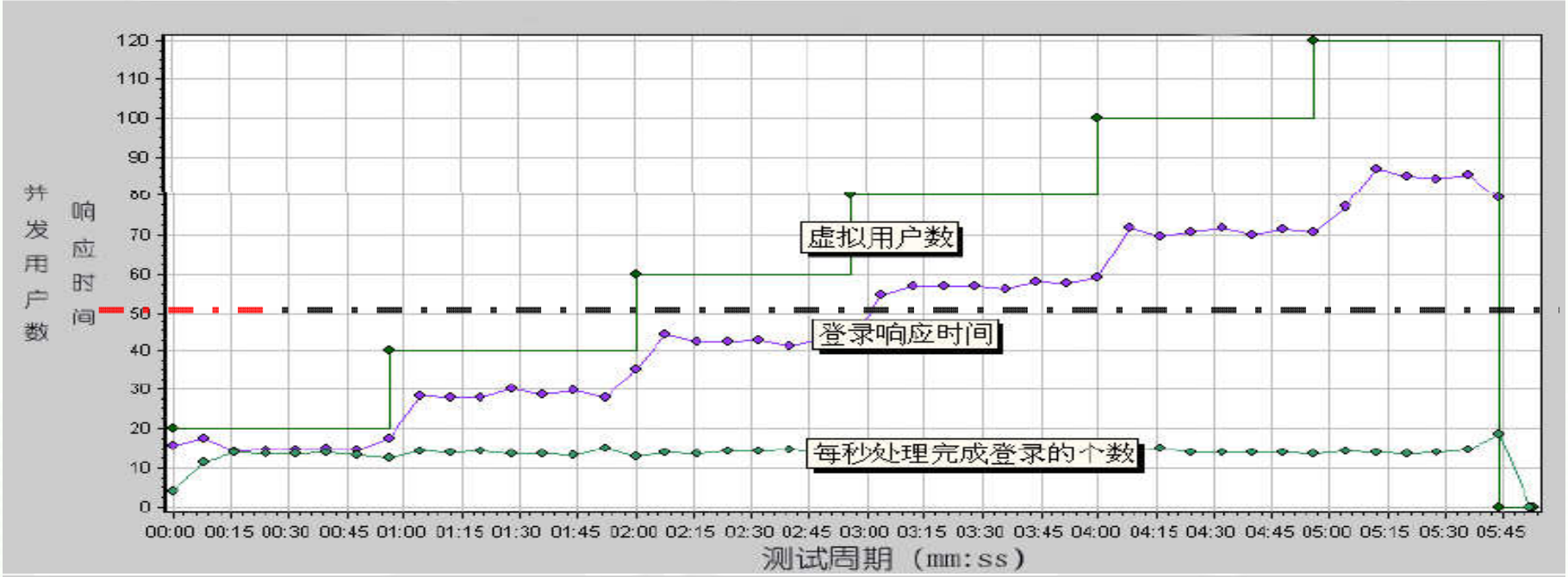


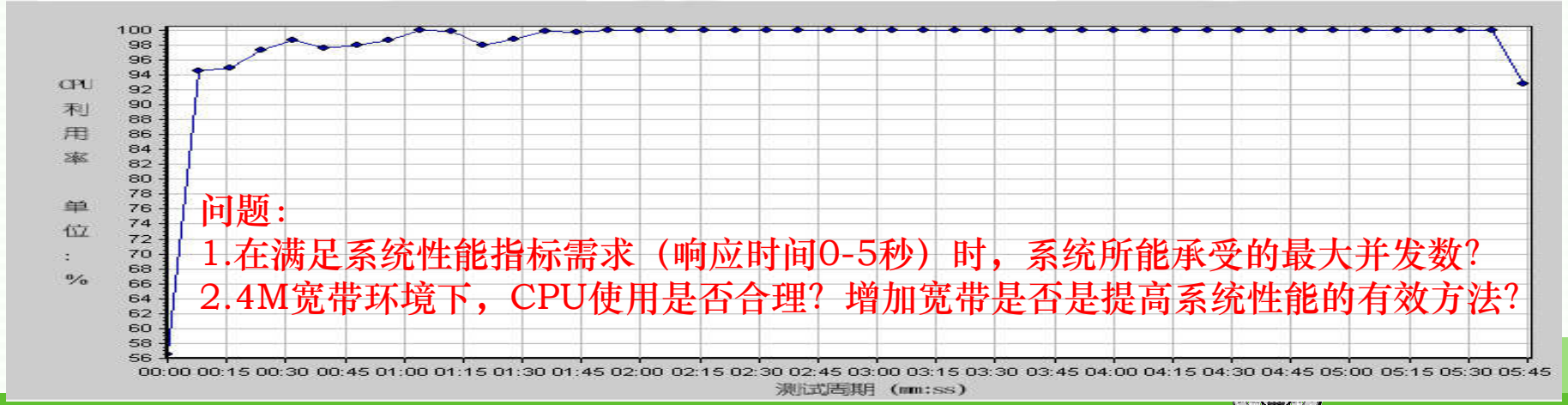
问题：

1.在满足系统性能指标需求（响应时间0-5秒）时，系统所能承受的最大并发数?

2.2M宽带环境下，CPU使用是否合理？宽带是否是系统瓶颈?

注：图中登录响应时间的纵坐标单位是0.1秒





问题：

1.在满足系统性能指标需求（响应时间0-5秒）时，系统所能承受的最大并发数?

2.4M宽带环境下，CPU使用是否合理？增加宽带是否是提高系统性能的有效方法?

1） 通过 Case 1 中的并发用户数和响应时间的监控图，发现登录响应时间随虚拟并发用户的增加而增长。在 50 个虚拟并发用户的负载下，登录响应时间达到 5 秒（注：图形中响应时间指标的比例为 10：1），当负载超过 50 个虚拟用户时，响应时间超过 5 秒或与 5 秒持平。因此可推断当系统满足性能指标需求时，系统能够承受的并发用户登录的最大数量是 50。

2） 在 Case 1 中的服务器资源监控图中分析，得出服务器的 CPU 资源使用是合理的。

3） 将 Case 1 和 Case 2 结合起来比较，发现 4M 带宽下，系统每秒处理完成的登录个数固定在 13.5 个左右，登录响应时间随虚拟用户增加而增长。在 60 个虚拟用户的压力下，登录响应时间在 4.2 秒左右。在 80 个虚拟用户的压力下，登录响应时间在 5.8 秒左右，所以在合理登录响应时间 5 秒内预计同时登录用户数是 70 左右。服务器 CPU 使用已成为新的瓶颈。这说明随着带宽的提高，系统的处理能力进一步提高，充分说明了 2M 网络会成为系统的瓶颈，但是增加网络带宽又会造成 CPU 资源利用紧张，造成新的瓶颈带来更严重的后果。

非功能系统测试

■互操作测试/兼容性测试

目的：

保证两个或多个产品可以交换和使用信息，并恰当地在一起运行

典型兼容目标：

操作系统、数据库、网络、浏览器、应用软件

不仅是不同版本之间，而且是不同系统之间

非功能系统测试

互操作测试/兼容性测试

例如：

- 从Web页面剪切文字，在word文档中粘贴；
- 使照片修饰软件在同一操作系统下的不同版本中正常工作；
- 新的数据库程序对现存所有数据库，能像老程序一样处理；
- 从电子表格程序保存账目数据，在另一个完全不同的电子表格程序中读入。

非功能系统测试

互操作测试/兼容性测试

后向兼容：

老版本中的数据能否在当前版本中使用

前向兼容：

当前版本中的数据能否在未来版本中使用

测试描述	测试类型
测试产品组件之间的接口	集成测试
测试两个或多个产品之间的信息交换	互操作性测试
以不同基础设施部件，如OS，DB等测试产品	兼容性测试
测试产品老版本上的目标代码等是否能在当前版本上使用	后向兼容测试
测试产品接口是否能在基础设施部件的未来版本上使用	前向兼容测试
测试产品的API接口是否能在客户开发的组件上使用	API/集成测试

非功能系统测试

可使用性和易获得性测试

可使用性：

以用户视点确认产品的易用性、速度和美感的测试

可使用性质量因素：

可理解性、一致性、导航、响应

非功能系统测试

■可使用性和易获得性测试

易获得性：

检查产品对于行动不便的用户也可使用的测试

基本易获得性：

粘贴键、声响键、多种模拟鼠标功能、朗读器、触摸...

非功能系统测试

国际化测试

目的：

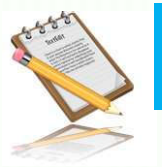
确保软件支持不同国家语言的测试手段

国际化问题：

语言的消息显示
语言和习俗的消息处理
日期格式、货币格式
对话框等文本显示区域至少0.5倍宽度的扩展空间
……

- 本地化测试注意点
  - 1.注意软件内度量单位、日期、时间、货币等的表示方式；
  - 2.对不同的度量单位等建立不同的等价划分；
  - 3.针对硬件平台和软件平台都要考虑进行平台划分；

二 确认测试



1. 确认测试概念

确认测试

概念

定义：

检查产品是否满足在项目的需求阶段定义的确认准则，或者说是否具备在真实环境中使用的条件

实施者：

客户或客户代表

引入时机：

系统测试之后

确认测试

目的：

验证和接受产品。

测试用例：

测试用例数量较少，目的也不是为了发现缺陷。

测试环境：

近似实际场景下执行。

确认测试

■确认准则

产品确认：

对现有测试用例进行分类形成确认准则

如：所有高优先级的需求要100%通过。

所有性能测试用例都必须满足所要求的响应时间。

规程确认：

根据交付规程进行定义。

如：产品源代码也要通过光盘交付。

在部署之前，要提供至少20人的产品使用培训。

确认测试

确认准则

服务等级约定：

服务等级约定通常是由客户和产品组织签署的合同的一部分。可选择重要的合同条款作为确认测试的一部分进行检验。

- 例如：系统不能工作的时间不高于0.1%。
- 再如：从报告时间起，所有重要缺陷的修改时间不得超过48小时。（确认测试要保证具有满足这些约定的资源)

确认测试

选择确认准则的测试用例

建议：

- 端到端的功能验证：所有事务测试都在客户端进行
- 领域测试用例：反映业务领域知识
- 用户场景测试用例：反映真实用户场景
- 完备性测试用例：全面包含各项功能

确认测试

选择确认准则的测试用例

建议：

- 新功能：检验新功能
- 少量非功能测试用例：再次验证非功能特性
- 符合法律义务和服务约定的测试用例
- 确认测试数据：选择客户真实数据

确认测试

执行确认测试

开发组织：

辅助客户完成确认测试

确认测试团队：

产品管理层+支持团队+咨询团队

90%成员：具有产品业务过程知识

10%成员：技术测试团队

开发组织的测试团队：

与确认测试团队不断沟通，提供采集测试数据和分析测试结果的帮助

三 回归测试

1. 回归测试概念
2. 波及效应

Microsoft

1

开发人员的一天

Rainbow Build: 2261 has been build SUCCESSFULLY and PASSED BVT tests - Message (Rich Text)

File Edit View Insert Format Tools Actions Table Help

From: Rainbow Build Automation
To: Raibow Project Team All Members
Cc:
Subject: Rainbow Build: 2261 has been build SUCCESSFULLY and PASSED BVT tests

Build Quick Status

Rainbow Build **2261** has been build SUCCESSFULLY and PASSED the BVT tests.

Drop Location

Build Information

Build No.	2261
Build Started	06/29/2004 04:00 AM
Build Finished	06/29/2004 04:20 AM
Build Result	Passed
Location	\\rainbow02\build\drops\2261\
MSI Tested	\\rainbow02\build\drops\2261\distrib\retail\RainbowInstaller.msi
Other Comments	

Summary Results

Summary of BVT Tests

No. of Test Cases Executed	4
No. of Test Cases Passed	4
No. of Test Cases Failed	0

Detailed Results URL

For Detailed Results, visit:
http://rainbowteam/build/BVT/Default.aspx

Thank you,
Rainbow Test Team

This is an automated Daily Build message generated from an unmonitored alias.
DO NOT REPLY to this alias.
For Further information on results, please contact Rainbow Test Team.
For Build Information, please contact Rainbow Build Distribution.

Microsoft

接下来，开发人员会...

- 从源代码管理工具中Check out代码
- 修改代码（解决Bug或实现新功能）
- 取得源代码管理工具中最新变化，在本机Build和单元测试
- 请开发组同事作Code Review
- Check in代码
- 在Bug管理工具中修改Bug的状态
- 开发人员以一封Daily Report结束一天的工作

启示：软件每天都在成长变化

问题1：软件修改会对周围/之前的模块有何影响？

问题2：用什么测试手段保证软件在不断修改过程中的功能完整性和正确性？

回归测试基础

回归测试背景

琢磨不定的软件

1.它总是在变化：软件不断演化——开发、维护、升级
2.对它的要求从未停止：软件需求不断变更
3.它还要适应不断变化的环境：技术更新和软/硬件升级

软件的每次改变都会引入潜在风险——缺陷

回归测试基础

回归测试背景

对软件变化的要求

1.修改后的功能是否达到预期？
2.原有功能是否被损害？

解决方案：回归测试是一种验证已变更系统的完整性与正确性的测试技术

回归测试基础

回归测试概念

回归测试(Regression Testing)：

定义1：回归测试是对之前已测试过、经过修改的程序进行的重新测试，以保证该修改没有引入新的错误或者由于更改而发现之前未发现的错误。

定义2：回归测试要保证增强型或改正型修改使软件正常进行并且不影响已有的功能。

回归测试基础

回归测试概念

回归测试意义：

1.保证软件维护时未更改的代码功能不会受到影响

2.保证软件模块区域和持续维护过程与回归测试的协作关系，使回归测试成为一个每月/每周/每日的常规活动

3.实现软件整个生命周期的测试

回归测试基础

回归测试概念

回归测试引入时机：

单元测试
集成测试
系统测试

引入原则：开发过程发生修改或维护，就有必要进行回归测试

回归测试基础

回归测试概念

回归测试特点：

1.测试计划

常规测试：已有的带有测试用例的测试计划

回归测试：更改的规格说明书、修改过的程序和需要更新的旧测试计划

2.测试范围

常规测试：整个程序

回归测试：被修改部分正确性以及它与原有功能的整合

3.时间分配

常规测试：测试时间实现有预算

回归测试：测试时间不包含在进度表中

回归测试基础

回归测试概念

回归测试特点：

4.开发信息

常规测试：随时可获得开发信息

回归测试：需要保留开发信息保证回归测试正确

5.完成时间

常规测试：所需时间长

回归测试：只需测试软件的一部分，所需时间短

6.执行频率

常规测试：高频率的活动

回归测试：由系统被修改而触发的周期性活动

回归测试基础

回归测试概念

回归测试过程

1.提出软件修改需求

2.进行软件修改

3.选择测试用例
获取正确的测试用例集

4.执行测试
通常以自动化方式执行大量测试用例

回归测试基础

回归测试概念

回归测试过程

5.识别失败结果

失败原因：测试用例错误？代码错误？

6.识别错误

精确定位哪些组件和哪些修改导致测试失败

7.排除错误

错误的处理：修改、移除、忽略

回归测试策略

策略1：全部重新测试：重新执行之前所有测试用例
优点：不用进行测试用例选择
缺点：不灵活（对系统微小改动就要全部重新测试）

策略2：有选择地重新测试：选择和使用现有测试用例子集
优点：灵活、实用于测试用例较多的情形
缺点：需要耗费精力选择测试用例

回归测试

重新确认测试用例

方式：手工操作

目的：识别出对于新版本软件不再有效的测试用例

步骤：检查需求规格说明书、测试策略和已有测试用例

特点：费时耗力

回归测试

识别错误——组测试技术

原理：一个模块可能单独工作正常，多个模块集成工作时却出现错误；组测试(Group Testing)是寻找此类错误的有效方法；

A	B	C	D	E	原先版本：所有测试用例全部通过
A_1	B_1	C_1	D	E	新版本：有几个测试用例失败
A_1	B	C	D	E	运行失败测试用例，只有A是新组件
A	B_1	C	D	E	运行失败测试用例，只有B是新组件
A	B	C_1	D	E	运行失败测试用例，只有B是新组件

回归测试中的波及效应

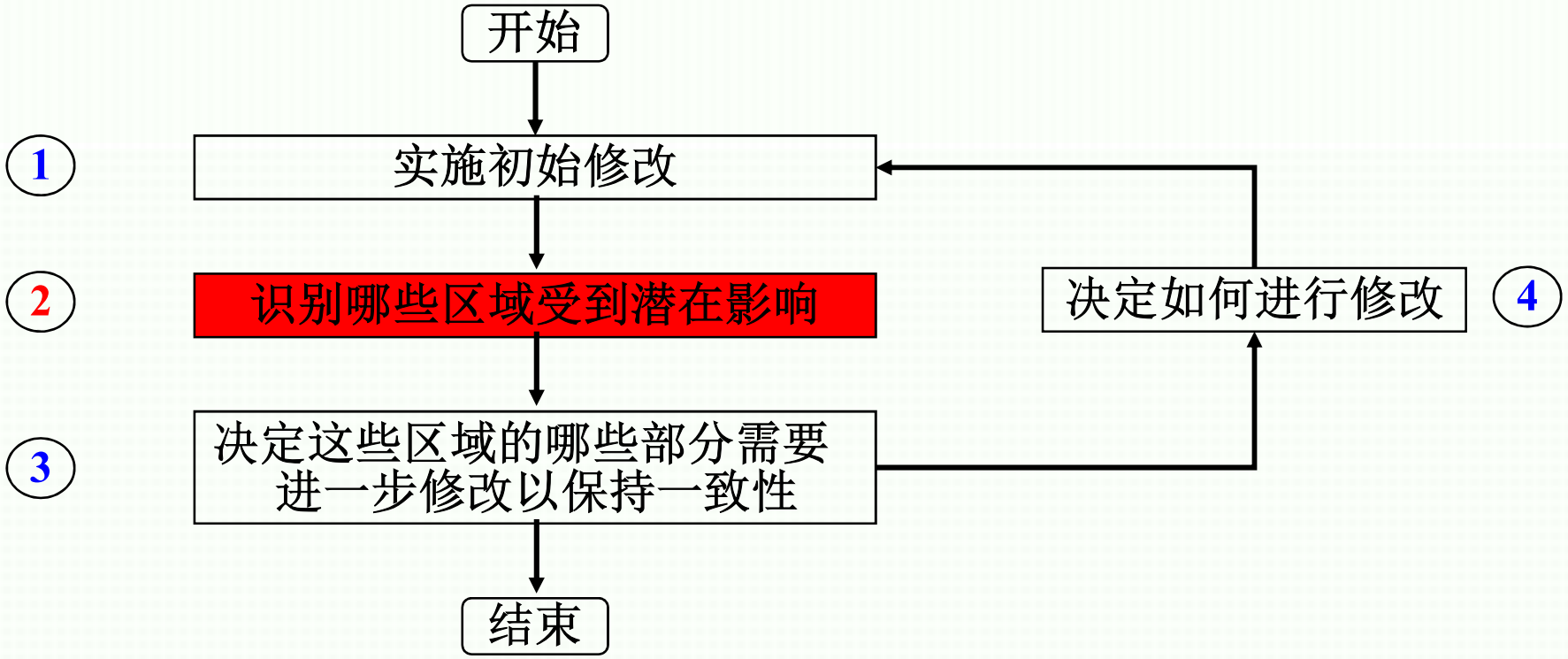
波及效应是为了发现所有受影响部分和发现潜在的受影响部分，以保证软件发生改变后仍然保持一致性与完整性。

四种波及效应：

1.需求的波及效应
2.设计的波及效应
3.代码的波及效应
4.测试用例的波及效应

回归测试中的波及效应

波及效应分析步骤：



回归测试中的波及效应

两种识别技术：

1.字符串匹配或交叉引用

2.程序切片(Program Slice)

国内程序切片领域：

东南大学：徐宝文 教授（现 南京大学）

李必信 教授

上海交通大学：赵建军 教授

回归测试工具

QTP与回归测试

QuickTest Professional是著名的自动化测试工具，被广泛应用于软件功能测试和回归测试中

QTP是早期WinRunner的替代品

QTP9.2基础知识

QTP 9.2 安装

- QTP 9.2支持的环境
  - 操作系统：windows 2000，windows XP，windows server 2003，windows Vista
  - 浏览器：IE6，IE7，FireFox1.5/2.0，Netscape 8.1.2
  - **Web页面**
  - **标准windows应用程序**

QTP9.2基础知识

QTP 9.2的基本配置

- QTP的帮助文档—QuickTest Professional | Documentation，F1
- QTP自带的样例程序
  - Windows 程序—“Flight ”机票预定系统
  - Web 程序---“Mercury Tours Web Site”

QTP9.2基础知识

QTP 的运行原理

- 通过查找应用程序界面中的各个控件的属性来判断是否与测试对象匹配，还可以根据控件的类型，把其拥有的可操作的方法列举出来。

QTP9.2基础知识

- QTP的使用
  - 测试脚本录制和编辑
  - 测试脚本调试和运行
  - 分析测试结果

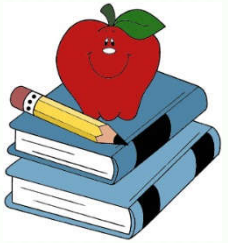
QTP9.2基础知识

- QTP的常用概念
  - 关键字视图/专家视图
  - 检查点
  - 参数化
  - 对象识别(Object Spy)
  - 关键字驱动/数据驱动

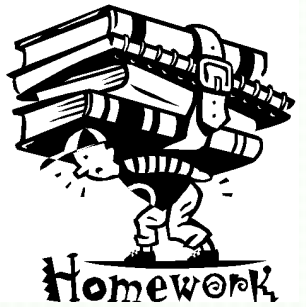
参考文献

1. **Srinivasan Desikan, Gopalaswamy Ramesh,**（韩柯，李娜 译），软件测试：原理与实践，**p80-p104**，机械工业出版社
2. 宫云战，软件测试教程，**p155-p206**, 机械工业出版社
3. **Srinivasan Desikan, Gopalaswamy Ramesh,**（韩柯，李娜 译），软件测试：原理与实践，p120-p130，机械工业出版社
4. 郁莲，软件测试方法与实践，p149-p158，清华大学出版社
5. 陈能技，QTP自动化测试实践，电子工业出版社

课后习题



无



本讲总结

- 系统测试
- 确认测试
- 回归测试

衷心感谢各位老师莅临指导！

欢迎各位老师同学批评指正！

Email: pwang@seu.edu.cn

系统测试_review

- 功能系统测试方法
  1. 设计/体系结构测试
  2. 业务垂直测试
  3. 部署测试
  4. Alpha测试和Beta测试
  5. 符合性的认证、标准和测试

非功能系统测试_review

非功能系统测试方法

1. 可伸缩性测试/容量测试
2. 可靠性测试
3. 负载测试
4. 压力测试
5. 性能测试
6. 互操作性测试/兼容性测试
7. 可使用性与易获得性测试
8. 国际化测试