



软件开发环境国家重点实验室
State Key Laboratory of Software Development Environment

质量保障和测试

定义

- **测试 (Testing) – 提供输入、度量输出，在实现之后进行**
 - 单元测试
 - 黑盒测试
 - 白盒测试
- **质量保障 (Quality Assurance) – 所有增加对软件质量信心的活动**
 - 测试只是质量保障的一种活动
 - 质量保障涉及软件生命周期的各个阶段

为什么质量很重要？



- **低质量软件的支持太昂贵**
 - ❑ 支持电话
 - ❑ 补丁 / 快速修复
 - ❑ 重新发布 (Vx.1)
- **低质量软件增加你的法律责任**
- **低质量软件降低你的声誉**
 - ❑ 信任容易失去，需要十倍以上的努力去恢复
- **对于专业人士来说：**
 - ❑ 低质量软件 → 你公司的市场份额 → 股票价格 → 你的利益

什么是质量

- 质量的某些方面是**客观的**
 - 稳定性 / 没有bug
 - 是否遵从规格说明书
- 质量的某些方面是**主观的**
 - 对客户的总体价值 / 是否满足课题的需要
 - 好的终端用户体验、情感价值
 - 使得客户想要更多...
- 所以没有Bug != 高质量
 - 为什么BMW比Hyundai更好？

测试是否足够保证质量？

- **经典测试技术只是度量客观质量，比如稳定性和与规格说明书的符合性**
- **测试通常是在设计完成很长一段时间后进行的，所以设计问题很可能不能及时检测出来**
- **测试用例生成是一门艺术，所以覆盖的可靠性差别很大**
- **测试时间经常被牺牲**

• QA角色和头衔

- ❑ 软件测试工程师 / STE
- ❑ 面向测试的软件设计工程师 / SDE/T
- ❑ 技术带头人
- ❑ 测试带头人
- ❑ 测试经理
- ❑ 集团测试经理
- ❑ 测试主管
- ❑ 测试架构师
- ❑ 卓越的工程设计大师

- **开发人员/测试人员比例目标：1:1**
- **时间安排**
 - 实现和稳定的时间1:1
 - 质量控制里程碑
 - 代码完成
 - 可视代码(UI)冻结
 - 代码冻结
 - 零Bug反弹
 - 第三方托管(Escrow)
 - 发布候选 1... / RC1...
 - 制造商发布/网上发布 / RTM/RTW



• 产品的生命周期模型中包含质量控制活动

- ❑ 功能计划：包含对上一版本的质量反馈 / bug修复
- ❑ 规格说明书：规格建模
- ❑ 实现：代码复审、伤检分类、签入测试、构建验证测试
- ❑ 稳定：测试通过、锁定、第三方托管

• 安全开发生命周期 / SDL

- ❑ 威胁建模
- ❑ 遗留代码的安全提升
- ❑ 二进制/静态代码扫描
- ❑ 安全团队的遵从性复审

典型的软件开发工程师/测试的一天



- **Depends on where we are in the PCM**
 - ❑ During Planning: Spec Reviews, Test infrastructure work, establish EE best practices
 - ❑ During Implementation: Test Engineering, Code Reviews
 - ❑ During Stabilization: Test Passes, Security work
- **Major components of SDE/T Work**
 - ❑ Reviews of specs (requires PM skills / customer empathy)
 - ❑ Reviews of Code (requires Dev++ skills)
 - ❑ Generating Test Cases (requires Creativity, hacker mentality)
 - ❑ Writing automation / Test Case Generation (requires Engineering skills, creativity)
 - ❑ Investigating and debugging issues (requires system knowledge, persistence)

关于测试的误区

- 测试在项目的最后进行就可以了。
- 这是远远不够的。当你在项目后期发现了问题，问题的根源往往是项目的早期的一些决定和设计，这时候，再要对进行修改就比较困难了。这要求测试人员从项目开始就要积极介入，从源头防止问题的发生。
- 有人会说- 我是一个小小的测试人员，项目开始的时候我能做什么？这就是小小测试人员努力的方向。
- 一个软件项目的各个功能都可以有自己的测试计划，它们可以在不同的阶段发挥作用。但是针对整个项目的总测试计划（又叫测试总纲）要在计划阶段大致定下来，并指导所有测试工作的进行。
- What is “good enough”



- **单元测试应该在最基本的功能/参数上验证程序的正确性**
- **单元测试必须由最熟悉代码的人（程序的作者）来写**
- **单元测试过后，机器状态保持不变**
- **单元测试要快**
- **单元测试应该产生可重复，一致的结果**
- **单元测试的运行/通过/失败不依赖于别的测试，可以人为构造数据，以保持单元测试的独立性**
- **单元测试应该覆盖所有代码路径**
- **单元测试应该继承到自动测试的框架中**
- **单元测试必须和产品代码一起保护和维护**

关于测试的误区

- **测试就得根据规格说明书（spec）来测，所以是很机械的。**
- **那不一定，即使你的软件产品功能100% 符合spec的要求，但是用户也可能非常恨你的软件。这时，测试人员就没有尽到责任，因为测试人员要从用户的角度出发，测试软件。**

关于测试的误区

- **测试人员当然也写代码，但是质量不一定要很高。**
- **开发人员的代码没写好，可以依赖于测试人员，来发现问题。但是如果测试人员的代码没写好，我们依赖谁来测试，改错呢？这就要求我们测试人员的代码质量特别高，因为我们是最后一道防线，如果我们的代码和测试工作有漏洞，那么bug 就会跑到用户那里去。**

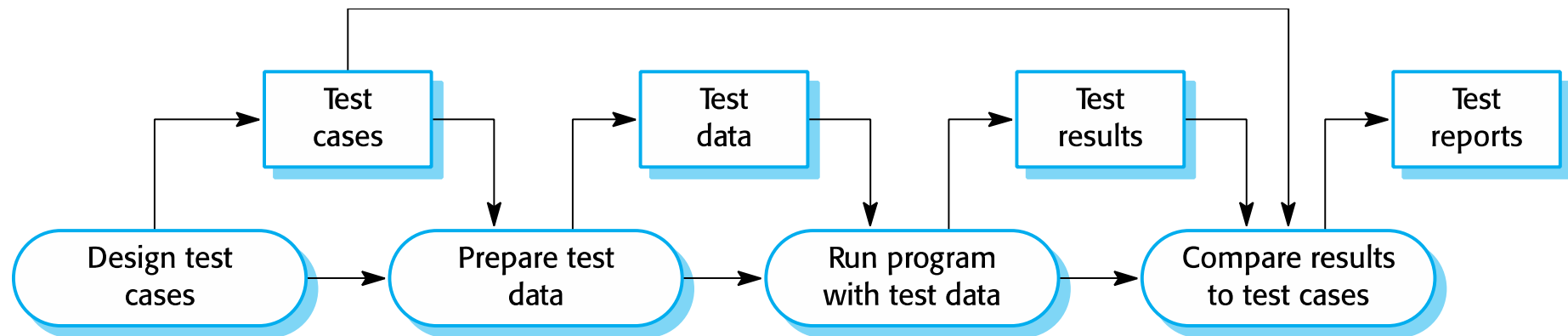
关于测试的误区

- **测试的时候尽量用Debug版本，便于发现 bug**
- **如果你的目的是尽快让问题显现，尽快找到问题，那建议用Debug版本，“尽快发现问题”在软件开发周期的早期特别重要。**
- **如果你的目的是尽可能测试用户所看到的软件，则用Release版本，这在软件开发的后期很有价值，特别是在运行效能 (performance) 和压力 (stress) 测试的时候。**

不同类型的测试



A model of the software testing process



Bug : 缺陷



- **Bug有3方面特征:**

- 症状 (Symptom)
- 程序错误 (Fault)
- 根本原因 (Root cause)

- **症状: 即从用户的角度看, 软件出了什么问题**

- 例如, 在输入 (3 2 1 1) 的时候, 程序错误退出。

- **程序错误: 从代码的角度看, 代码的什么错误导致了软件的问题**

- 例如, 代码在输入为 (3 2 1 1) 情况下访问了非法的内存地址——0X0000000C。

- **根本原因: 导致代码错误的根本原因**

- 例如, 代码对于id1==id2的情况没有做正确判断, 从而引用了未赋初值的变量, 产生了以上的情况。

例子

- **症状：**用户报告，一个Windows应用程序有时在启动时报错，程序不能运行。
- **程序错误：**有时候一个子窗口的handle为空，导致程序访问了非法内存地址，此为代码错误。
- **根本原因：**代码并没有确保创建子窗口（在CreateSubWindow()内部才做）发生在调用子窗口之前（在OnDraw()时调用），因此子窗口的变量有时在访问时空，导致上面提到的错误。

好的错误报告



- 1. bug的标题，要简明地说明问题。**
- 2. bug 的内容要写在描述中，包括：**
 - 1. 测试的环境和准备工作；**
 - 2. 测试的步骤，清楚地列出每一步做了什么；**
 - 3. 实际发生的结果；**
 - 4. （根据spec和用户的期望）应该发生的结果。**
- 3. 如果需要其他补充材料，例如相关联的bug、输出文件、日志文件、调用堆栈的列表、截屏等，都要保存在bug 相应的附件或链接中。**
- 4. 还可以设置bug 的严重程度（Severity）、功能区域等，这些都可在不同的字段中记录。**

严重程度

- **Sev = 1**

- ☐ 数据丢失
- ☐ 崩溃、界面锁死
- ☐ 安全问题
- ☐ 阻碍主要功能的使用

- **Sev = 2**

- ☐ 阻碍部分功能的使用
- ☐ 界面不是所见即所得
- ☐ 可扩展性
- ☐ 场景不完整

- **Sev = 3**

- ☐ 小的易用性（可用性）问题
- ☐ 性能问题、间歇性的停顿

我们能修复什么

- 修复症状

- ▣ 别让程序退出，吃掉异常

- 修复程序错误

- ▣ 修改代码

- 修复根本原因

- ▣ 找到根本原因
 - Spec 对某种情况没有考虑
 - 设计没有考虑支持多语言
 - ▣ 把所有受到根本原因影响的设计都改正

Bug的生命周期



- **测试者 / 用户：报告症状**
 - 创建一个bug
- **PM：理解影响，确定修复什么，什么时候修复**
 - 设置优先级和伤检分类
- **开发者：修复bug，修复根本原因**
 - Bug处于“working”的状态
- **代码复审者：确保质量**
 - Bug的修复签入，bug得到解决
- **测试者：回归测试**
 - Bug被关闭

测试人员做什么

- 对规格说明的遵从性

- (注意：大量的新人通常到此为止)

- 反例 / 错误处理

- 是否会崩溃？是否能记录？是否能恢复？错误消息是否有用？

- 例子：

- 网络断开
 - 接口超时
 - 某个资源被锁定
 - 引用某个已被销毁的对象 / 对象初始化失败
 - XML损坏 / 与schema不符合
 - 用户不是管理员

- 用户体验

- 是否能够交付好的体验给用户（在提供价值的同时）？

不同的测试



- **测试设计有两类方法：**
 - ▣ **黑盒、白盒**
- **黑盒：**
 - ▣ 在设计测试的过程中，把软件系统当作一个“黑盒”，无法了解或使用系统的内部结构及知识。一个更准确的说法是“Behavioral Test Design”，从软件的行为，而不是内部结构出发来设计测试。
- **白盒：**
 - ▣ 在设计测试的过程中，设计者可以“看到”软件系统的内部结构，并且使用软件的内部知识来指导测试数据及方法的选择。“白盒”并不是一个精确的说法，因为把箱子涂成白色，同样也看不见箱子里的东西。有人建议用“玻璃箱”来表示。
- **在实际的测试中，当然是对系统了解得越多越好。**

功能测试



测试名称	测试内容
Unit Test	单元测试——在最低的功能/参数上验证程序的正确性
Functional Test	功能测试——验证模块的功能
Integration Test	集成测试——验证几个互相有依赖关系的模块的功能
Scenario Test	场景测试——验证几个模块是否能够完成一个用户场景
System Test	系统测试——对于整个系统功能的测试
Alpha/Beta Test	外部软件测试人员（Alpha/Beta 测试员）在实际用户环境中对软件进行全面的测试

非功能测试



测试名称	测试内容
Stress/load test	测试软件在负载情况下能否正常工作
Performance test	测试软件的效能
Accessibility test	软件辅助功能测试——测试软件是否向残疾用户提供足够的辅助功能
Localization/Globalization Test	本地化/全球化测试
Compatibility Test	兼容性测试
Configuration Test	配置测试——测试软件在各种配置下能否正常工作
Usability Test	易用性（可用性）测试——测试软件是否好用
Security Test	软件安全性测试

测试的目的



测试名称	测试内容
Smoke Test	“冒烟” ——如果测试不通过，则不能进行下一步工作
Build Verification Test	验证构建是否通过基本测试
Acceptance Test	验收测试，为了全面考核某方面功能/特性而做的测试

测试的方法

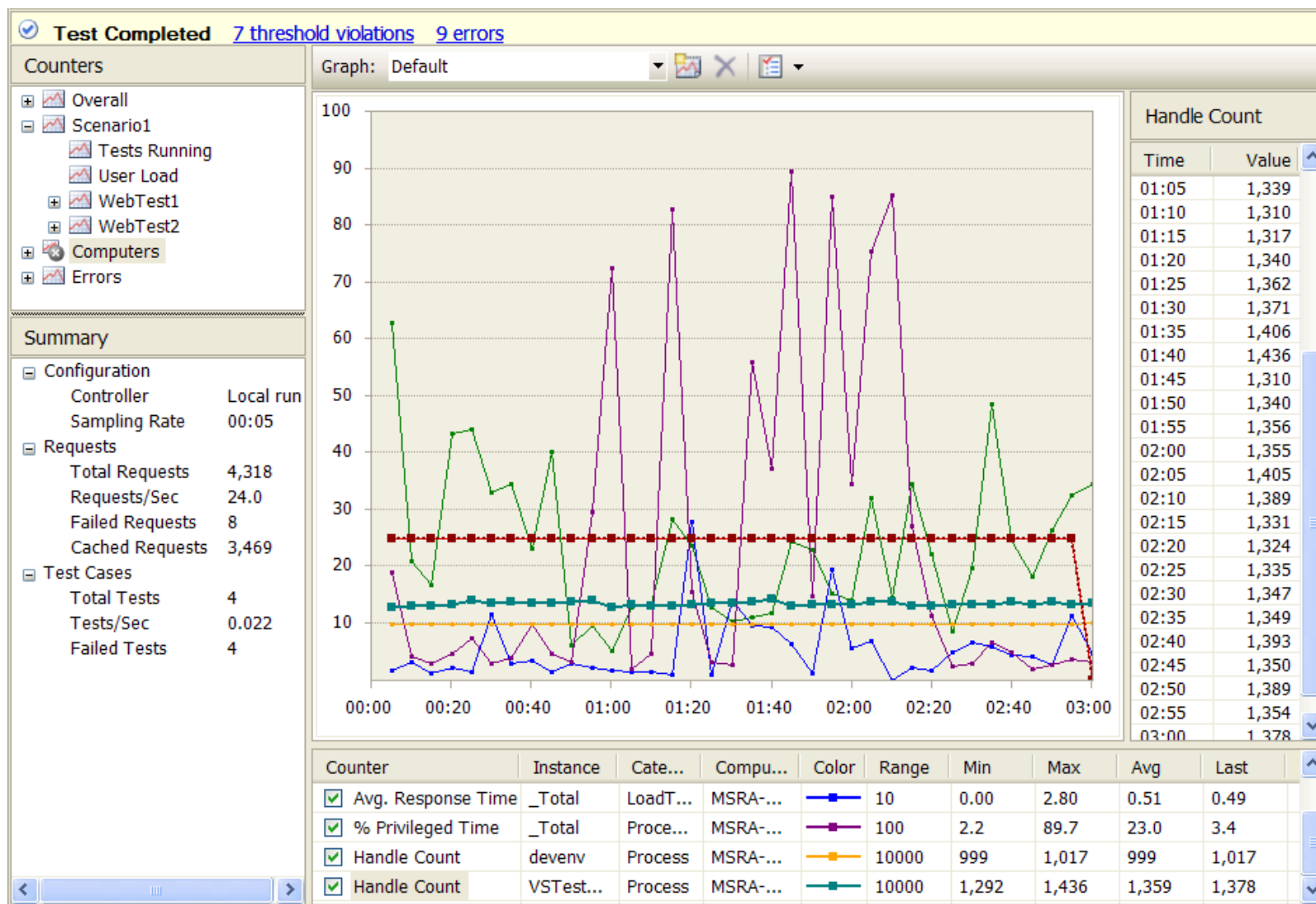


测试名称	测试内容
Regression Test	“回归”测试——对一个新的版本，重新运行以往的测试用例，看看新版本和已知的版本相比是否有“退化”（regression）
Ad hoc (Exploratory) Test	随机进行的、探索性的测试
Bug Bash	Bug大扫荡——全体成员参加的找“小强”活动
Buddy Test	伙伴测试——测试人员为开发人员（伙伴）的特定模块作的测试

例子 - 如何测试效能

- **效能测试**：在100个用户的情况下，产品搜索必须在3秒钟内返回结果。
- **负载测试**：在2 000 用户的情况下，产品搜索必须在5秒钟内返回结果。
- **压力测试**：在高峰压力（4 000 用户）持续48小时的情况下，产品搜索的返回时间必须保持稳定。系统不至于崩溃。

负载测试结果



例子 - 旅客列车



- **效能测试：**

- 在80%上座率的情况下，
- 期望：列车按时到达，并且乘客享受到优质服务（每小时清洁，保障水，食物，卫生）。乘务员不要太累。

- **负载测试：**

- 在100%上座率的情况下，
- 期望：列车大部分按时到达，乘客享受到基本服务。乘务员的疲劳在可恢复范围内。

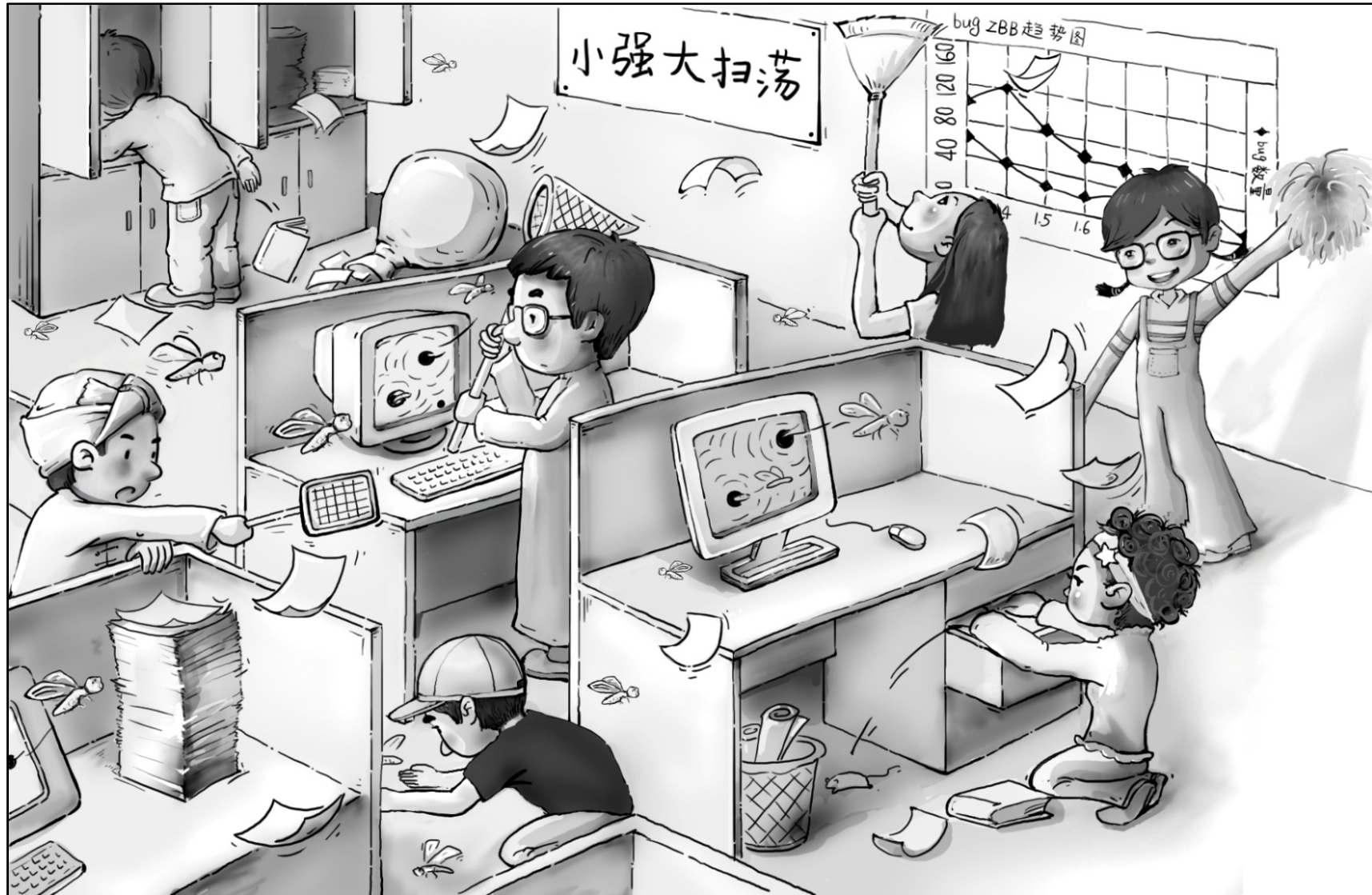
- **压力测试：**

- 在高峰压力是200%上座率，全国铁路系统增加20%列车，持续15天的情况下（春运）
- 期望：列车能到站，无出轨；乘客能活着下车，系统不至于崩溃。乘务员也能活着下车。

质量保障和测试小结



- 只在最后进行测试并不能保证质量
- 仅仅针对规格说明书进行测试是不够的
- 并不是只有测试者对质量负责，每一个人都需要负责
- 需要掌握大量的知识和工具才能把测试做好
- 测试是软件工程中的一种伟大的训练
- QA覆盖了软件的整个生命周期




What to do



- **PM (of product team)**

- ❑ What' s ready for testing
- ❑ What' s not ready (incomplete feature)
- ❑ If your software is not ready, share with them your spec, and test plan, UI design

- **Tester (from testing team)**

- ❑ Focus on the “testable areas”
 - ❑ Open bugs from user' s perspective
 - ❑ If software is not ready, open bugs against spec, UI design, etc.
- 



谢谢！

