



# SOFTWARE TESTING


苏临之

[sulinzhi029@nwu.edu.cn](mailto:sulinzhi029@nwu.edu.cn)



# Shared Testing

- ❖ 测试共享：在一定时间内互换测试任务，可理解为“你执行我的测试，我执行你的测试”。
- ❖ 这样可以让他他人想出其他的测试用例来测试，如在动态黑盒测试里至少可以帮助审查等价类划分和测试用例，为测试提供新思路。
- ❖ 请求协助寻找软件缺陷的最佳伙伴是产品支持或客户服务小组。这些人对软件缺陷非常敏感（特别是易用性缺陷）。



# $\alpha$ - and $\beta$ -Testing

- ❖  $\alpha$ 测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，不能由程序员或测试员完成。
- ❖  $\beta$ 测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，不能由程序员或测试员完成。所以 $\beta$ 测试是在开发者无法控制的环境下进行的软件现场应用。



# Differences

- ❖ 两者主要区别是测试的场所、环境和时序不同：
  - $\alpha$ 测试可以是指把用户请到开发方的场所来测试， $\beta$ 测试是指在一个或多个用户的场所进行的测试。
  - $\alpha$ 测试可以的环境是受开发方控制的，用户的数量相对比较少，时间比较集中。而 $\beta$ 测试的环境是不受开发方控制的，且用户数量相对比较多，时间不集中。
  - 一般 $\alpha$ 测试可以先于 $\beta$ 测试执行。



# Outsourcing Testing

- ❖ 许多公司采用的一种常用做法是向擅长各方面软件测试的其他公司外包或提交部分测试工作。这虽然比有产品小组的测试员来完成更麻烦、更昂贵，但是一种更有效的共享测试途径。
- ❖ 配置和兼容性测试通常是外包测试的理想选择。
- ❖ 本地化测试是另一个通常被外包测试的例子。



# Defect Management

- ❖ 软件缺陷管理是在软件生命周期中识别、管理、沟通任何缺陷的过程。该过程从缺陷的识别开始，直到到缺陷的解决关闭为止，确保缺陷被跟踪管理而不丢失。一般的，需要跟踪管理工具来帮助进行缺陷全流程管理。
- ❖ 每一个软件组织需要明确必须妥善处理软件中的缺陷，因为这关系到软件组织机构生存、发展的质量根本。但是并非所有的软件组织都明确应当如何有效地管理自己软件中的缺陷。





# Description of a Defect

- 缺陷ID
- 缺陷标题
- 缺陷详细描述
- 缺陷提交人和提交时间
- 缺陷状态
- 缺陷严重程度
- 缺陷紧急程度
- 缺陷等级
- 缺陷所属项目/模块
- 缺陷指定解决人和解决时间
- 缺陷复核情况
- 测试环境
- 必需附件



# Seven Defect Statuses

- New (发现错误 • 新建)
- Open (打开)
- Fixed (已修复)
- Reopen (再次打开)
- Closed (关闭)
- Rejected (拒绝)
- Delayed (延迟)





# Seven Defect Statuses

状态	状态说明	操作人员
发现错误	首次录入	测试人员
打开	已经确认是缺陷，并等待修改	开发经理/项目经理
已修复	修改完成后待回归测试验证	开发经理/开发人员
再次打开	回归测试验证不通过，再次等待修改	测试人员
关闭	回归测试验证通过	测试人员
拒绝	讨论后认为不是缺陷或拒绝修改	开发经理/开发人员
延迟	延迟修改	开发经理/开发人员/缺陷 评审委员会



# Process of Defect Management

- 缺陷管理的流程可以概括为：测试人员提交新的错误入库，缺陷状态为“发现错误·新建”；项目经理将缺陷分配给相应的开发人员，缺陷状态为“打开”；开发人员查重现缺陷，做如下处理：如果不是缺陷，则置状态为“拒绝”；如果是缺陷则修复并置状态为“已修复”；如果不能解决的错误，要留下文字说明并保持错误为“拒绝”状态。测试人员查询状态为“已修复”的错误，验证错误是否已解决，做如下处理：如问题解决了置错误的状态为“关闭”，如果问题得不到立即解决则需要置状态为“延迟”。



# Five Defect Levels

等级	说明	举例
低	可在发布后再商量是否改进	A. 某些测试的建议
中	不影响功能的正常使用,可以在时间和资源允许的情况下再解决	A. 辅助性说明描述不清楚 B. 显示格式不规范 C. 长时间操作未给用户进度提示 D. 提示窗口文字未采用行业术语 E. 可输入区域和只读区域没有明显的区分标志 F. 系统处理未优化
高	事件是重要的,但是由于解决问题需要花费一定的时间,所以可以用较长的时间解决,如果时间紧可以留下个版本解决	A. 界面文字等错误 B. 打印内容、格式错误 C. 简单的输入限制未放在前台进行控制 D. 删除操作未给出提示



# Five Defect Levels

等级	说明	举例
很高	事件是重要的，并且应该在紧急的事件处理之后尽快得到解决，必须在发布前解决	A. 功能不符 B. 缺少功能，与需求不符 C. 数据流错误 D. 程序接口错误 E. 轻微的数值计算错误
紧急	事件非常重要，测试工作无法继续进行，需要马上给予关注解决。	A. 由于程序所引起的死机，非法退出，运行中断，应用程序崩溃 B. 死循环 C. 导致数据库发生死锁 D. 数据通讯错误 E. 严重的数值计算错误



# Planning a Testing Effort

- ❖ 软件测试中最后一个要点就是把先前的软件测试知识（去哪里找、如何测试、如何有效测试）联系起来，说明和软件测试有关的所有工作如何计划、如何组织。
- ❖ 通过利用精心组织的测试计划、测试用例和测试报告，可以对测试工作进行正确的记录以及交流，将使达到目标变得更加可能。



# Goals

- ❖ 如果测试员之间不交流计划测试的对象，需要什么资源，进度如何安排，整个项目就很难成功。
- ❖ 软件测试计划是软件测试员与产品开发小组交流意图的主要方式。
- ❖ 测试计划的目标就是：规定测试活动的范围、方法、资源和进度；明确正在测试的项目、需要测试的特性、要执行的测试任务、每个任务的负责人，以及与计划相关的风险。



# Role of Documentation

- ❖ 根据电气和电子工程师协会 (IEEE) 的标准，测试计划采用的形式是书面文档。这虽然很重要，但是并不是测试计划的全部内容。
- ❖ 测试计划文档只是创建详细计划过程的一个副产品，重要的是计划过程，而不是产生的结果文档。
- ❖ 如果计划工作的目标从建立文档转移到建立过程，从撰写测试计划到计划测试任务，就不会存在束之高阁的文档了。
- ❖ 因此测试计划过程的最终目标是交流 (而不是记录) 软件测试小组的意图、期望，以及对将要执行的测试任务的理解。





# Main Theme

- ❖ 测试计划主题是一套清单，该清单应该在整个项目小组中全面讨论、相互沟通并达成一致。这比测试计划文档模版更加重要。
- ❖ 测试计划的主题主要包括以下方面：高级期望、人员（包括地点和事物）、责任、测试必要性、测试阶段、测试策略、资源需求、任务分配、测试进度、测试用例、缺陷报告、度量统计、风险问题等。



# High-Level Expectation

- ❖ 测试计划过程和软件测试计划的目的是什么？这一点不但测试人员要清楚，还要保证程序员、文档编写人员、管理部门都要知道，因为测试要获得他们的同意和支持。
- ❖ 测试的是什么产品？因为软件产品不断的升级换代，下一版本是完全重写还是维护升级？它是一个独立的程序还是有若干个小程序构成的？它是自行开发的还是由第三方开发的？
- ❖ 产品的质量 and 可靠性目标是什么？如销售代表说软件运行要尽量快（怎么快）；程序员说软件要使用最先进（怎样算是先进）的技术；产品支持说软件不能有引起任何（什么程度）冲突的缺陷，等等。这些都要有一致通过的定义。



# Who? Where? What're They Doing?

- ❖ 测试计划需要明确在项目中工作的人员，该人员需要做或正在做什么，怎样和他联系。
  - 因为测试小组有可能跟所有的人都需要打交道。
  - 内容包括人员姓名、职务、地址、电话号码、电子邮件地址和职责范围。
  - 还要包含相关文档放在哪里？
  - 如果在执行测试时硬件不可缺少，那么它放在哪里、如何得到？如果有进行配置测试的外部测试实验室，那么它们具体在哪里？
  - 找到所有的问题，把它们记录下来。



# Responsibility of the Group

- ❖ 有时确认职责划分非常麻烦，有些任务有多个负责者，一个负责人可能同时负责多个任务。
- ❖ 划分团队之间责任的有效方法是构造简表。用不同的符号区分任务的责任者、参与者，而不负责该任务的人员留以空白。



# Necessities of Testing

- ❖ 软件产品中包含某些内容不必测试。
  - 以前发布过或者测试过的软件部分
  - 来自其他软件公司（如外包公司）并已经测试过的内容
- ❖ 如果某部分内容不需要测试，需要阐明这样做的理由，因为确定对某内容不测试存在风险。



# Testing Stage

- ❖ 要计划测试的阶段，测试小组就会查看预定的开发模式，并决定在项目期间是采用一个测试阶段还是分阶段测试。
- ❖ 在边写边改的开发模式中只有一个测试阶段。在螺旋模型中可能有若干测试阶段，其中计划测试就是一个。
- ❖ 与测试阶段相关联的两个重要概念是进入和退出规则。每一阶段都必须有可观定义的规则，明确定义本阶段的结束和下一阶段开始。
  - 如说明书审查阶段可能在正式说明书审查公布时结束。
  - $\beta$ 测试阶段可能在测试员完成验收测试时开始。



# Testing Strategy

- ❖ 定义测试策略与定义测试阶段相关联。
- ❖ 测试策略描述测试小组用于测试整体和每个阶段的方法，决定使用黑盒还是白盒，或是两种方式结合使用，用在软件的什么部分，怎么运用？
- ❖ 如某些产品适合手工测试，而其他部分适合动态测试。如果使用工具，那么怎么获得？当然还可以选择把整个测试工作外包给专业测试公司等等。
- ❖ 测试策略的决策问题，责任重大，非同小可，要有经验相当丰富的测试员来做。





# Resource Demand

- ❖ 计划资源需求是确定实现测试策略的必要过程。  
由于项目后期想要获取资源非常困难，因此在项目测试期间可能用到的所有资源都要考虑。
- 人员：数量、经验和专长？全职、兼职？
- 设备：计算机、测试硬件、工具等
- 办公室和实验空间
- 软件
- 外包测试公司：用他们吗？怎么用？
- 其他配备



# Task Allocation

- ❖ 一旦定义了测试阶段、测试策略和资源要求，这些信息加上产品说明书就可以分配每个测试员的任务。
- ❖ 与团队之间的责任划分不一样，测试员的任务分配指的是，明确测试员负责软件的那些部分、那些可测试特性。
- ❖ 责任分配的越详细，就越能确保每一部分都有人测试，每一个测试员都明确自己负责什么，并且有足够的信息开始设计测试用例。



# Progress of Testing

- ❖ 测试进度需要以上所述的全部信息，并将其反映到整个进度安排中。
- ❖ 测试进度非常重要，有时甚至会根据测试进度决定砍掉产品的一些特性，或者推迟到下一版本中推出。
- ❖ 测试工作的进度安排通常都不是平均分布的，因为测试任务的数量、人员的数量和测试花费的时间随着项目的进展不断增长，在产品发布之前会形成短期的高峰。
- ❖ 持续增长的结果是测试进度受到先前事件的影响越来越大。即产生进度破坏的问题。
- ❖ 有效的解决进度破坏问题的方式是：避免定死启动和停止任务的时间。



# Final Examination

- 目标1：掌握基软件测试的基本概念、基本理论、常用测试方法与测试技术。
  - 目标2：能够使用常见的测试自动化工具以提高测试的效率和准确性。
  - 目标3：能够针对实际问题选择合适的测试策略，综合应用测试理论、测试方法与技术并运用测试工具进行软件部件测试的设计与实施。
- 
- 基本概念和基本理论，对应目标1（ $27 \div 0.6 = 45$ ）
  - 综合大型题目，对应目标3（ $27 \div 0.6 = 45$ ）
  - JUnit测试工具的使用，对应目标2（ $6 \div 0.6 = 10$ ）



# Final Examination

- 全英文试题，务必熟悉专业术语
- 熟悉课堂上讲过的概念，尤其是近年考试反复出现的概念和多年未考的概念
- 测试技术方法必须复习全面不留死角，理解方法的本质
- JUnit测试掌握其用法，尤其是如何使用注解、断言



**THANK YOU!**