计算机学院软件工程与实践课程实验报告

|  |  |  |
| --- | --- | --- |
| 实验题目：实验十三 软件测试（二） | | 学号：201905130197 |
| 日期：2022/6/11 | 班级： 计科19级1班 | 姓名： 赵一帆 |
| 实验目的：   1. 深入理解白盒测试和黑盒测试 2. 熟悉单元测试工具 3. 了解符号测试 4. 了解差分测试 | | |
| 实验软件和硬件环境：  Windows10 | | |
| 实验步骤与内容：   1. 阅读下面白盒测试和黑盒测试相关资料（或查阅其它相关资料），深入理解白盒测试和黑盒测试，总结其特点 （保存到每个小组选定的协作开发平台上，以组为单位）(陈文盼)   WhiteBox.pdf  BlackBox.pdf  **白盒测试：**是一种测试用例的设计方法，盒子指的是被测试的软件，盒子是可视的可以清楚知道盒子内部里的东西是如何运作的，因此白盒测试需要对系统内部的结构和工作原理有个清除的了解，基于这个知识来设计用例。  白盒测试技术一般可以分为静态分析和动态分析：   1. 静态分析：控制流分析技术、数据流分析技术、信息流分析技术； 2. 动态分析：逻辑覆盖率测试（分支测试、路径测试等）、程序插装等； 3. 优点：迫使测试人员去仔细思考软件的实现；可以检测代码中的每条分支和路径；揭示隐藏在代码中的错误；对代码的测试比较彻底；最优化。 4. 缺点：昂贵；无法检测代码中遗漏的路径和数据敏感性错误；不验证规格的正确性。   **黑盒测试：又叫功能测试，**这是因为主要关注被测软件的功能实现，而不是内部逻辑。在黑盒测试中，被测对象的内部逻辑，测试人员不可见，测试人员对被测产品的验证主要是根据其规格，验证其与规格的一致性。  绝大多数没有用户参与的黑盒测试中，最常见的测试有：功能性测试、容量测试、安全性测试、负载测试、恢复性测试、标杆测试、稳定性测试、可靠性测试等。   1. 优点：比较简单，不需要了解程序内部代码及实现；与软件内部实现无关；从用户角度出发，很容易的知道用户会用到哪些功能遇到哪些问题；在做软件自动化测试时比较方便。 2. 缺点：不可能覆盖所有的代码，覆盖率较低，大概只能达到总代码量的30%；   自动化测试的复用性较低。   1. 阅读下面单元测试相关工具的介绍（或查阅其它单元测试工具相关资料），学习单元测试工具的使用 （赵一帆、郑晓旭）   JUnit-Tutorial.pdf  junit\_tutorial.pdf  Embunit User Guide.pdf  （1）为什么要使用单元测试工具？  在应用程序中，单元是具有一个或多个输入和单个输出的软件中最小可测试部分。单元测试是一种测试软件代码单元的方法，通常包括一个或两个输入，产生一个输出。单元测试主要关注独立模块的功能正确性，目的是确保每个单元都按照预期的方式运行。  要进行单元测试，开发人员需要编写测试代码。单元测试有手动和自动化测试两种类型，自动化通常是首选的方法，可以为开发人员节省大量的时间和精力。  单元测试是自动化测试金字塔模型中占比最大的测试类型，做好单元测试对于保证软件产品的质量非常重要，单元测试可以：  及早发现软件中的缺陷并及早修复  修复单元测试发现的缺陷时，代码更改不会影响其他模块  模块集成变得更容易  减少缺陷率和时间成本  （2）现在让我们来总结一下针对Java、C++和Python语言的单元测试中，受欢迎的测试工具，它们不仅包括单元测试框架，还包括了Mock工具、代码覆盖率工具，以及两个智能化的单元测试用例自动生成工具：  Junit  TestNG  GoogleTest  pytest  unittest  Jmockit  JaCoCo  gcov、lcov、gcovr  Coverage.py  EvoSuite  Diffblue Cover  （3）对junit的进一步了解：  **·Junit简介**  JUnit是一个Java语言的单元测试框架。多数Java的开发环境都已经集成了JUnit作为单元测试的工具。Junit测试是程序员测试，即所谓白盒测试，因为程序员知道被测试的软件如何（How）完成功能和完成什么样（What）的功能。Junit是一套框架，继承TestCase类，就可以用Junit进行自动测试了。    **JUnit通过注解的方式来识别测试方法**。目前支持的主要注解有：  @BeforeClass:  修饰的方法会在所有方法被调用前执行，且该方法时静态的，所以当测试类被加载后就接着运行它，而且在内存中他只会存在一份实例，他比较适合加载配置文件（针对所有测试，只执行一次 ）  @AfterClass:  所修饰的方法通常用来对资源管理，如关闭数据库连接（针对所有测试，只执行一次 ）  @Before和@After  会在每个测试方法前后各执行一次  @Test：测试方法，在这里可以测试期望异常和超时时间  @Ignore：忽略的测试方法    **·Junit特性**  ①    用于测试期望结果的断言（Assertion）  ②    用于共享共同测试数据的测试工具  ③    用于方便的组织和运行测试的测试套件  ④    图形和文本的测试运行器    **·安装**  下载完以后解压到你喜欢的目录下，假设是JUNIT\_HOME，然后将JUNIT\_HOME下的junit.jar包加到你的系统的CLASSPATH环境变量中，对于IDE环境，对于需要用到的junit的项目增加到lib中，其设置不同的IDE有不同的设置，这里不多讲。  而**eclipse中自带JUnit**，我们可以直接使用。   1. 阅读下面符号测试 （Symbolic Testing）相关资料（或查阅其它相关资料），了解符号测试的基本概念、主要技术、重要挑战等（刘诗婷）   A Survey of Symbolic Execution Techniques.pdf  Symbolic Execution and Program Testing.pdf  Symbolic Execution for Software Testing-Three Decades Later.pdf  符号执行的核心思想:输入时用符号值(symbolic values)来表征而不是具体值(concrete data values)；程序变量根据输入符号值表征成符号表达式(symbolic expressions)；输出是根据程序计算出的函数表达式。普通测试执行的是算术运算，符号测试则是执行代数运算，可以代表一类的测试。 符号值可以是初等符号值，也可以是表达式。初等符号是任何变量值的字符串，表达式则是数字、算术运算符和符号值的组合。在条件语句中，判断条件就是谓词，可能是符号表达式，可取真假。不断构造下去，构成了一个二叉树，称为符号执行树。将各个分支点的谓词条件累积在一起，用逻辑乘符号联接在一起，得到的这个逻辑表达式称为路径条件。  符号执行能在给定时间里，探索尽量多的、不同的程序路径，生成一个具体输入的集合，检查是否存在错误（断言违规、未捕获的异常、安全漏洞和内存损坏）。其中，生成具体输入集合的优势在于：从测试用例角度来看，允许创建高覆盖率的测试套件；从查找bug角度来看，针对bug提供触发此bug的输入，用于验证和debug。另外，在给定程序路径中寻找错误时，符号执行比传统的动态执行技术更强大，传统动态执行技术的表现则要看那些触发错误的具体输入的可用性。最后，与某些其他程序分析技术不同，符号执行既可以找出“缓冲区溢出”这样的一般性错误，也可以引出更高级的程序属性，例如“复杂的程序断言”。   1. 阅读下面差分测试 （Differential Testing）相关资料（或查阅其它相关资料），了解差分测试的基本原理、主要应用等（史子涵）   Differential Testing for Software.pdf  Feedback-Directed Differential Testing of Interactive Debuggers.pdf  基本原理：  差分测试，也称为差分模糊测试，是一种流行的软件测试技术，它试图通过为一系列相似的应用程序（或同一应用程序的不同实现）提供相同的输入，并观察不同的应用程序来检测错误。他们的执行。差异测试是对传统软件测试的补充，因为它非常适合发现不表现出明显错误行为（如崩溃或断言失败）的语义或逻辑错误。差异测试有时称为背靠背测试。  差异测试通过使用与交叉引用[预言](https://en.wikipedia.org/wiki/Oracle_machine" \o "甲骨文机器)机相同的功能的不同实现来发现语义错误，查明它们在相同输入上的输出差异：相同输入上的程序行为之间的任何差异都被标记为潜在错误。  主要应用：  差异测试已被用于在不同领域成功地发现语义错误，例如SSL/TLS实现、C编译器、Java 反编译器、JVM实现、Web应用程序防火墙、API安全策略、和防病毒软件。差分测试也被用于从不同的网络协议实现中自动生成指纹。  记录项目及小组每个人工作的进度、里程碑、工作量的跟踪图或表，将其保存到每个小组选定的协作开发平台上，每周更新。 | | |
| 结论分析与体会：  本周认识到了测试的重要性，查阅资料，了解了各种测试：黑盒白盒测试，单元测试工具（主要是junit），符号测试，差分测试。 | | |