

信息6030

系统分析与设计

欢迎

第十周

澳大利亚联邦

1969 年版权条例

警告

本材料已由纽卡斯尔大学或代表该大学根据1968 年版权法第 VB 部分（以下简称“该法”）复制并传达给您。

本通讯中的材料可能受该法案的版权保护。您对本材料的任何进一步复制或传播可能受该法案的版权保护。

请不要删除此通知。

讲座幻灯片内容

- 这些讲座幻灯片是讲座的基础。
- 换句话说,实际讲座可能涵盖的内容比这些幻灯片。

·为什么

- 可能会问一个与评估相关的问题
- 可能会提出一个观点,然后我们就这个想法进行讨论
- 我可能会在讲座中提出问题并讨论答案
- 我可能会将内容与评估或时事联系起来
- 可能会根据上述行动讨论例子

指示性课程表

| 星期 | 一周开始 | 话题 | 学习活动 | 评估到期 |
|----|---------|----------------|----------------------|------------------|
| 1 | 5月13日 | 介绍 这是指示性课程表 | 实验室： 检视问题 团队组建 | |
| 2 | 5月20日 | 统一过程 (UP) 和模型 | 检视问题 案例分析 团队组建 | |
| 3 | 5月27日 | UP 信息收集和 造型 | 检视问题 案例分析 | |
| 4 | 6月 3 日 | 建模用例和类别 图表 | 检视问题 案例分析 | |
| 5 | 6月 10 日 | 可行性 | 检视问题 | 测验 1 (实验室) |
| 6 | 6月 17 日 | 修订 | 检视问题 | 评估 1:星期五晚上 11:59 |
| 7 | 6月 24 日 | 状态图 | 简短陈述 (每个团队) 案例分析 | |
| 8 | 7月 1 日 | 设计类图 | 检视问题 案例分析 | |
| 9 | 7月 8 日 | 序列图 | 检视问题 | 测验 2 (实验室中) |
| 10 | 7月 15 日 | 测试控制和 HCI 及修订 | 检视问题 案例分析 | |
| 11 | 7月 22 日 | 实施与部署 | 检视问题 | |
| 12 | 7月 29 日 | 修订 | 简短陈述 (每个团队) | 任务 2:周五晚上 11:59 |

联系信息

课程协调员: eugene

讲师:尤金·卢顿

电子邮件: eugene.lutton@newcastle.edu.au

电子邮件主题标题: .

INFO6030 / 地点 / 原因 · 例如:

INFO6030 / 在线 / 测验 2 · 例如:INFO6030 /
Callaghan / 视频第 9 周

·咨询:请发送电子邮件预约



(金属展开电阻器,nd)

测试

intended number
include Institute
tools
methods separate
expected also time cycle
may analysis example oriented
team black executed failure
system Compatibility following
Assurance Assurance
process development steps
output regression plan acceptance
often requirements defects used
tester box quality product defect beta
integrated one failures usually faults
needed Certification offered non-functional
case program testers developers value vs
two tests end white Input exist tested
cases usually reports different conditions well information
project common results functionality considered Certifications
SQA latest level coverage performance inputs
found functions automated faults slips



(360Logica, 第 201-215 页)

讲座概要

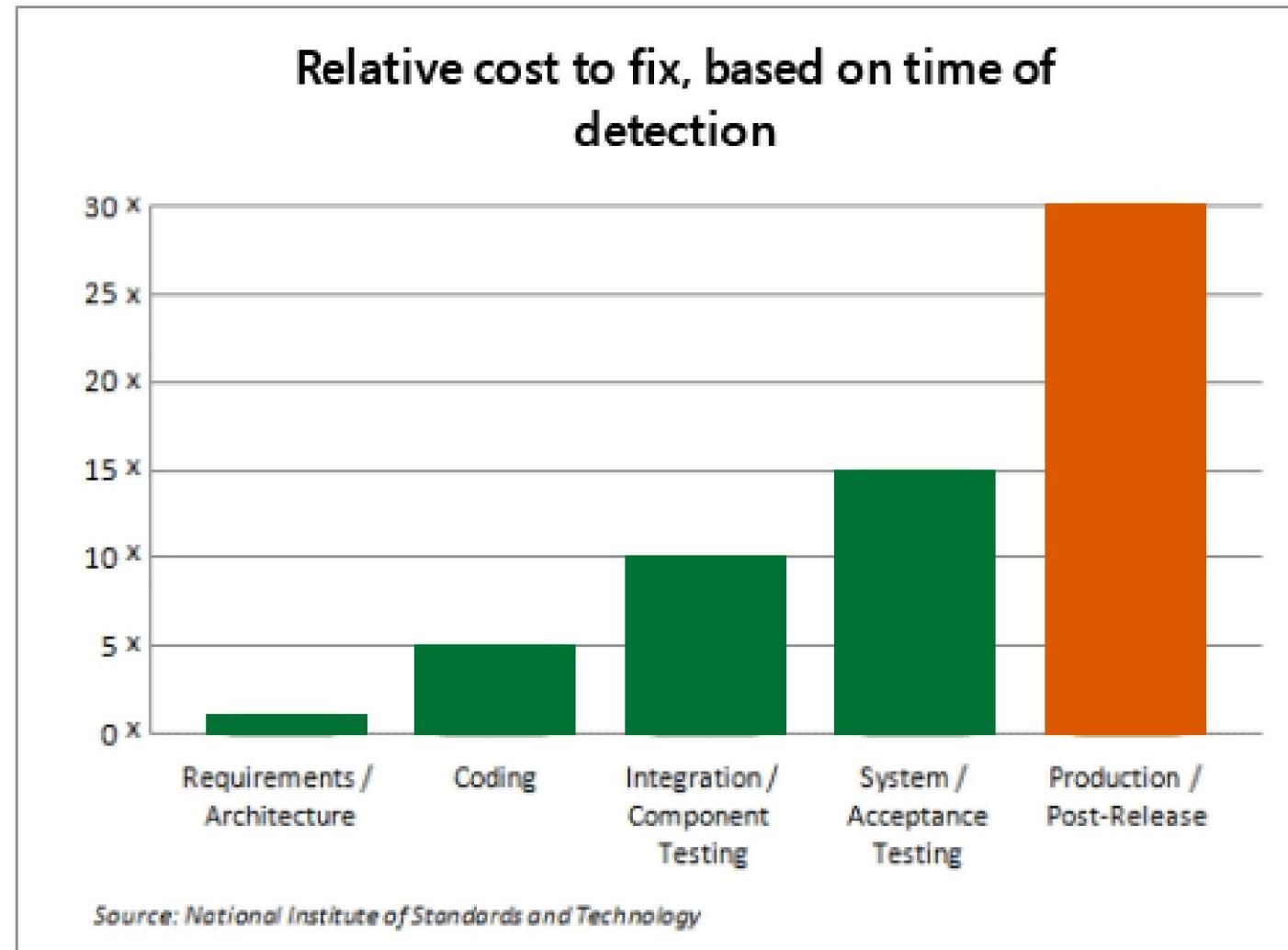
需要用例开发中的活动来建立系统：

- 实施 · 测试 · 部署 · 配置
和变更管理

测试很重要

- 开发人员将整个项目时间的50%用在测试上是很常见的
- 对于生命攸关的软件（飞行控制）,测试成本可能是所有其他活动总成本的 3 至 5 倍
- 测试的破坏性要求开发人员放弃
对自己开发的软件的正确性的先入为主的观念

测试很重要



(Guru99,第 39 页)

测试:术语

- 测试是识别缺陷的过程

- 制定测试计划、测试用例和测试数据

- **测试计划**描述测试策略、目标、时间表、估算、可交付成果、以及执行软件产品测试所需的资源。

- **测试数据**是一组用于测试模块、模块组或整个系统的起始状态和事件

- **测试用例**是

- 起始状态

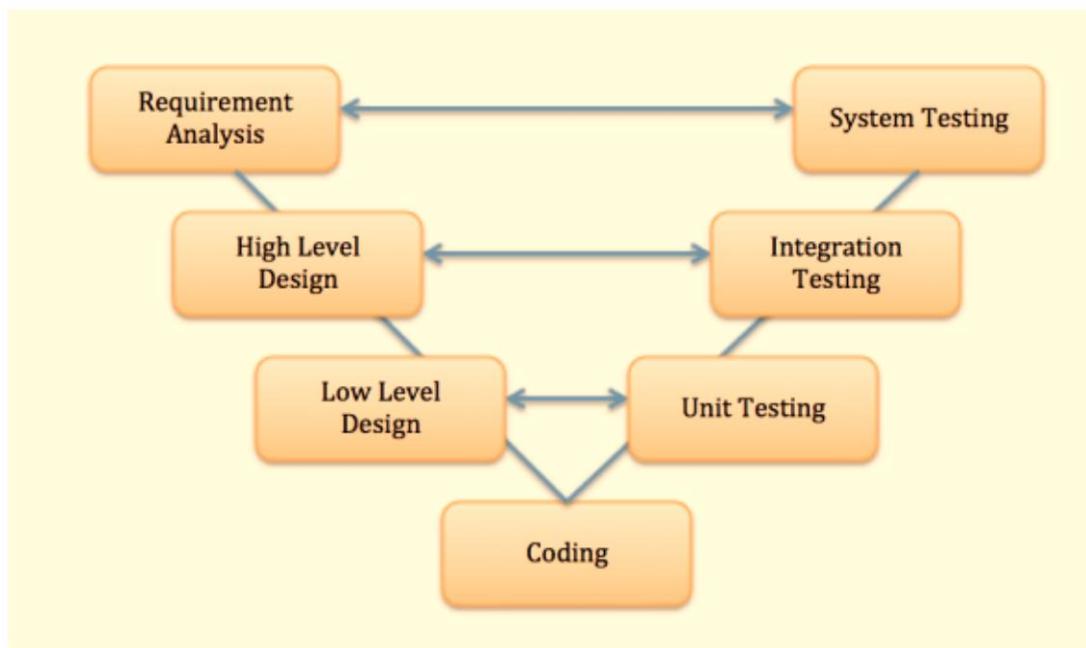
- 软件必须响应的一个或多个事件

- 预期反应或结束状态

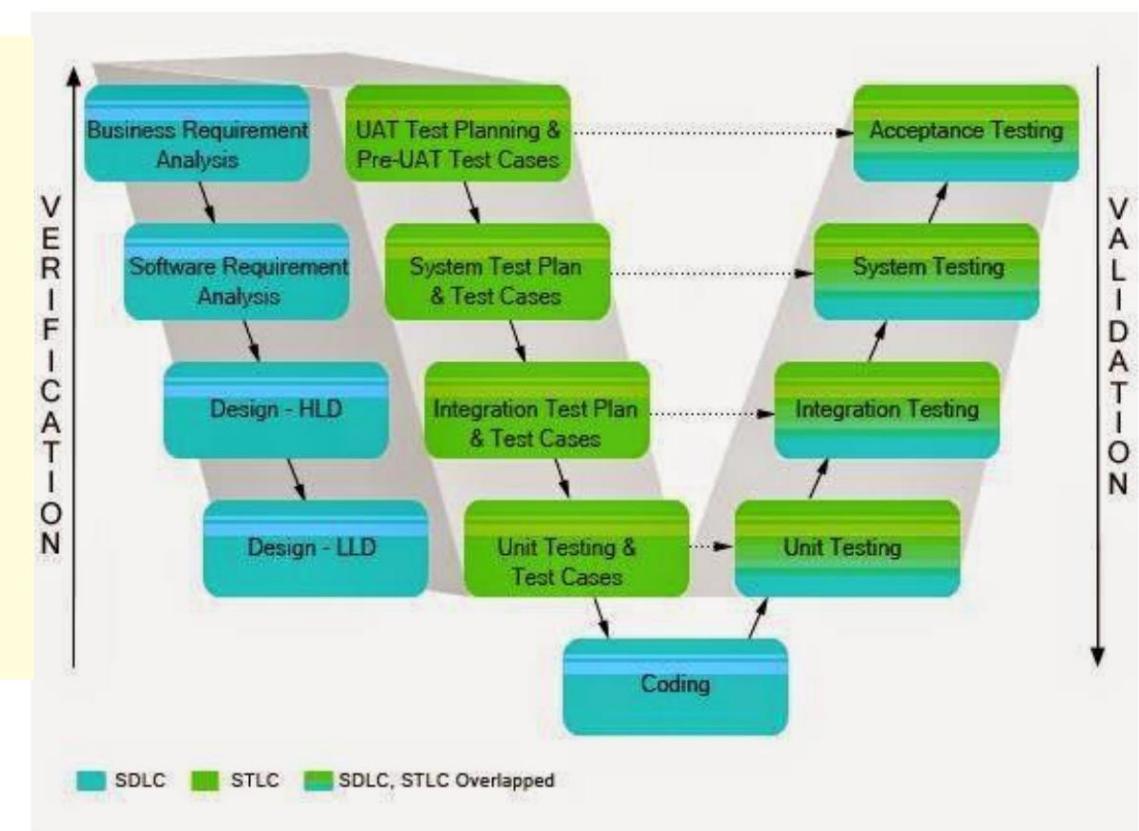
- 参见讲座末尾的每周阅读材料

SDLC 和 STLC:V 模型

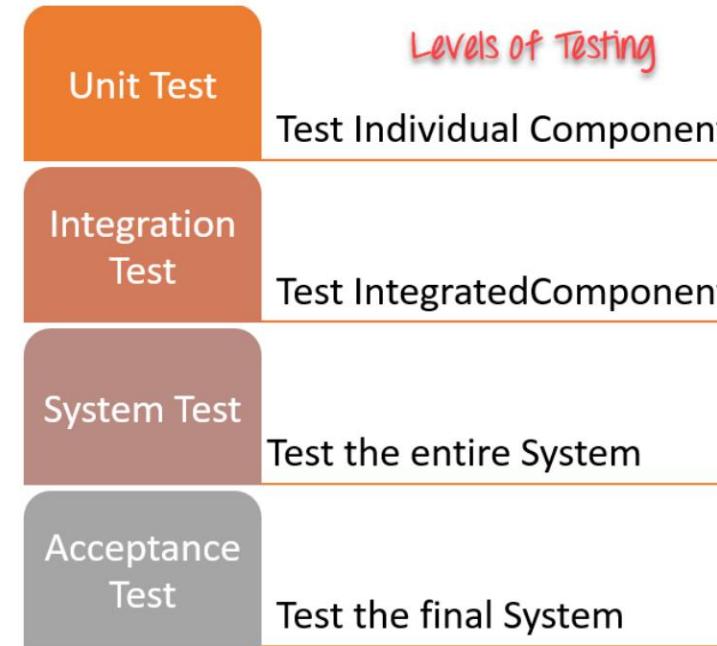
(Guru99, 第 39 页)



(Soni, 2015 年)



软件测试的级别

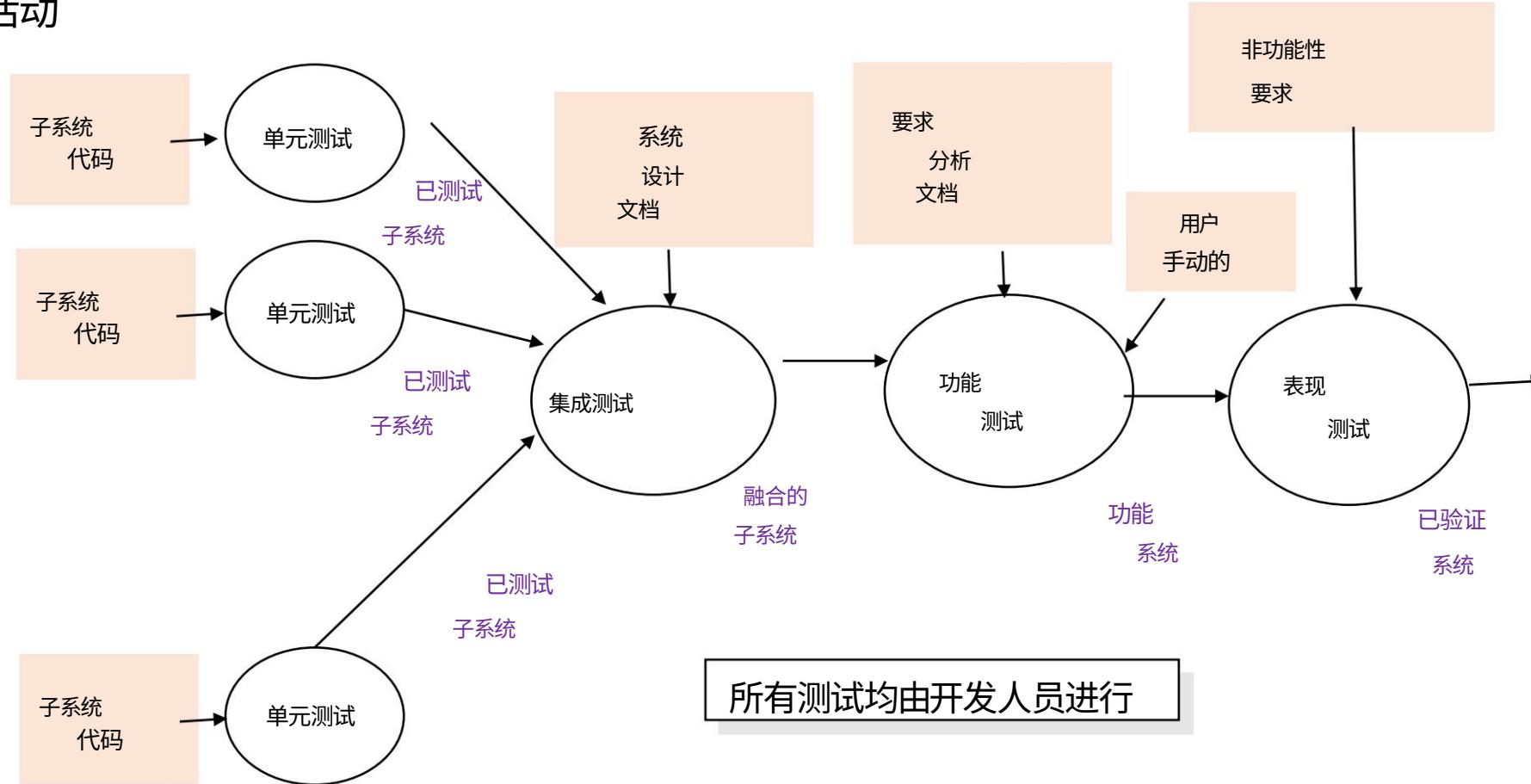


- 测试通常从功能 (黑盒)测试开始,
- 辅以结构 (白盒)测试,
- 通过一个或多个集成步骤从单元级别向系统级别进行工作。

(汉密尔顿,2023年5月13日)

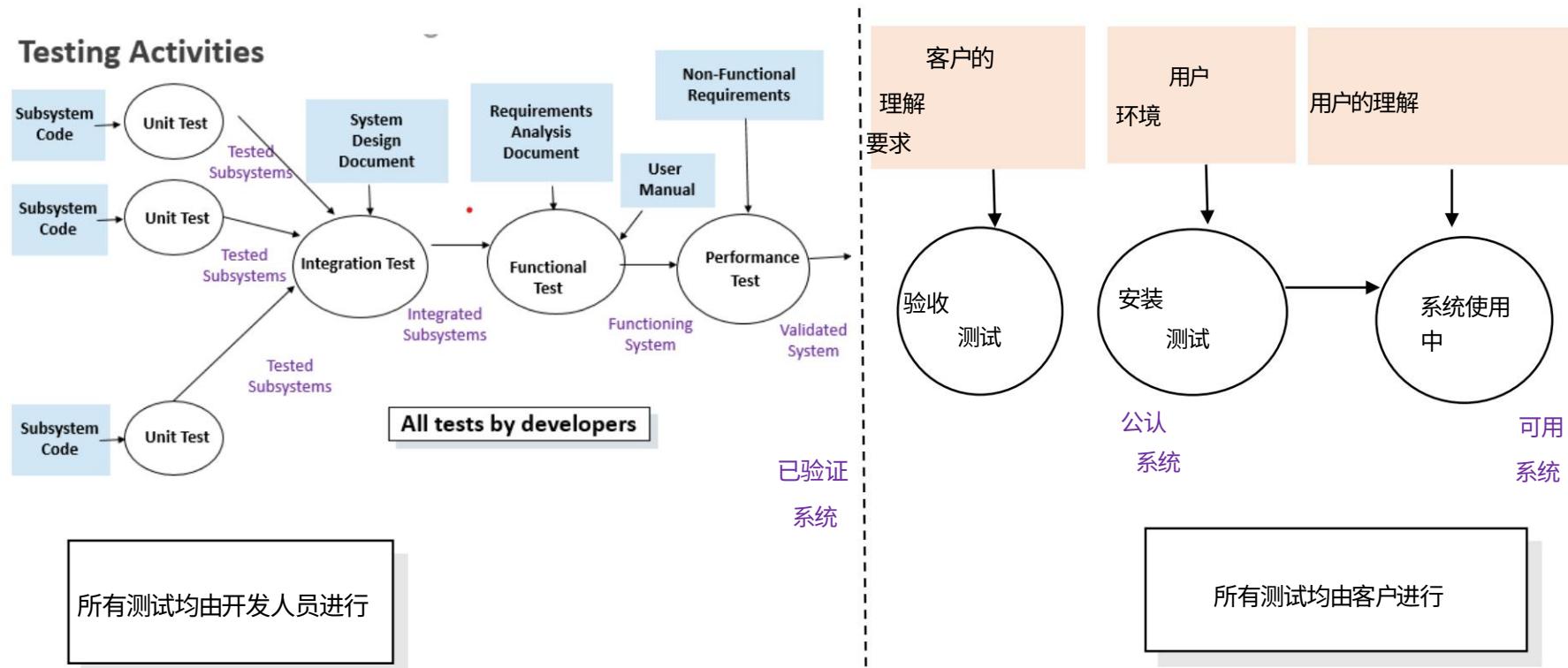
软件测试的级别

- 测试活动



软件测试的级别

- 测试活动



测试级别:单元

· 单元测试:

- 一个单位是系统或应用程序的最小可测试部分,可以编译、加载和执行。这种测试有助于单独测试每个模块。
- 目的是通过分离软件的每个部分来测试软件。它检查组件是否实现功能。这种测试由开发人员执行。

查找单个子系统或对象中的错误。

由开发人员进行。

黑盒测试:检查输出,根据用例进行测试,使用开发人员的功能知识。

白盒测试:检查代码中的逻辑路径,包括决策点、循环和边界值。

目标:确认子系统编码正确并执行预期功能。

(汉密尔顿,2023年5月13日)

测试级别:单元示例

- 测试类型:单元测试测试用例

测试案例 1:存款

输入 :BankAccount account = new BankAccount(100);, double amount = 50; 预期输出:

account.getBalance() returns 150.0 测试代码 :assertEquals(150,
account.getBalance()); 测试方法 :testDepositMoney()

测试级别:单元示例

- 测试类型:单元测试

```
导入org.junit.Test;导入静态  
org.junit.Assert.assertEquals;
```

公共类MyTest {

```
    _____  
  
    @Test public void testAddition() {  
        加法 addition = new Addtion (); int result = addtion.calculate(2, 3);  
        assertEquals(5, result);  
  
    }  
}
```

测试级别:集成

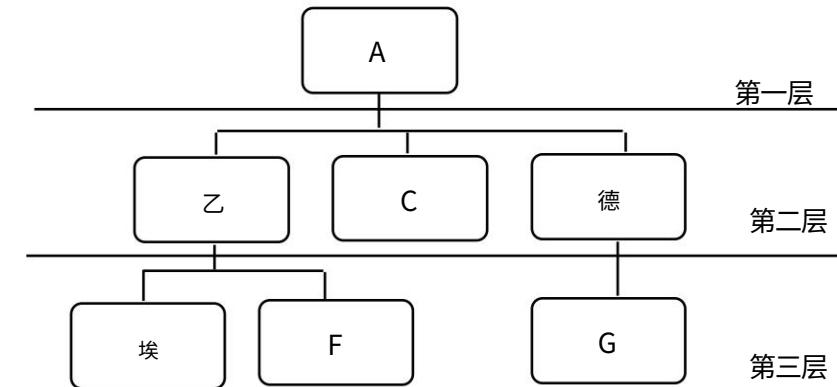
· 集成测试:

· 一体化意思是组合。例如,在这个测试阶段,不同的软件模块被组合在一起并作为一个整体进行测试,以确保集成系统已准备好进行系统测试。

· 集成测试检查从一个模块到另一个模块的数据流。这种测试由测试人员执行。

- 查找子系统连接在一起的错误。
- 由开发人员执行。
- 可以采用自上而下和自下而上的方法。
- 可能需要写存根。

· 目标:测试子系统之间的接口。



(汉密尔顿,2023年5月13日)

测试级别:集成示例

- 测试类型: `Integration Calculator.java`
- 测试用例: 验证BankingSystem 类是否可以从BankAccount 类正确地存入和提取资金。

- 先决条件:

- BankAccount 类已初始化其余额为 100。
- BankingSystem 类已使用 BankAccount 对象初始化。

- 脚步:

- 使用 BankingSystem 类向账户存入 50。
- 验证BankAccount类的余额是否增加了50。
- 使用 BankingSystem 类从账户中提取 20。
- 验证BankAccount类的余额是否减少了20。
- 验证BankAccount 类的余额仍然大于 0。

- 预期成绩:

- 存入50后,BankAccount类的余额为150。
- 提取20后,BankAccount类的余额为80。
- BankAccount类的余额仍然大于0。

测试级别:集成示例

测试类型: [Integration Calculator.java](#)

```
公共类计算器{ public int  
    add(int a, int b) {  
        返回a + b;  
    }  
}
```

```
公共类 Logger { 公共  
    void log (String message)  
    { System.out.println (message) ;  
    }  
}
```

测试级别:集成示例

- 测试类型:测试类型: Integration Calculator.java

```
public class CalculatorIntegrationTest {  
    @Test  
    public void testAddition() {  
        Calculator calculator = new Calculator();  
        Logger logger = new Logger();  
        // Call the calculator method and log the result  
        int result = calculator.add(2, 3);  
        logger.log("Result: " + result);  
        // Verify that the result was logged correctly  
        assertEquals("Result: 5", logger.getLog());  
    }  
}
```



测试级别:集成示例

- 测试类型: `Integration Calculator.java`
- 创建Calculator 和Logger 类的实例,然后调用Calculator 实例上的add 方法执行加法运算。
- 使用 Logger 实例将结果记录到控制台。
- 通过使用 `assertEquals` 方法检查 Logger 实例的输出来验证结果是否被正确记录。
- “结果:5” ,这是加法运算的预期结果。

测试级别:系统

· 系统测试

· 系统测试在完整的集成系统上执行。它允许根据要求检查系统的合规性。它测试组件的整体交互。它涉及负载、性能、可靠性和安全性测试。

- 系统测试通常是最终测试,用于验证系统是否符合规范。它评估功能性和非功能性测试都需要测试。

· 根据场景和要求测试整个系统行为（将系统视为一个黑匣子）。

- 由开发人员执行

- 对所有系统功能进行功能测试

- 非功能测试 性能测试。

- 目标:确定系统是否满足功能性和非功能性要求。

(汉密尔顿,2023年5月13日)

测试级别:系统

- 测试类型:系统测试
- 这有点复杂,但总的来说,我们正在测试系统转换按要求输入/输出或系统功能。
- 我们可以对火灾管理系统进行哪些测试?

测试级别:系统示例

测试类型:系统测试

- 测试案例 1:登录和注销
 - 前提条件:用户未登录
 - 步骤 1:输入有效的用户名和密码
 - 第 2 步:验证登录成功
 - 步骤 3:点击注销按钮
 - 步骤 4:验证注销是否成功
 - 预期结果:用户已注销并且无法访问系统
- 测试案例2:存款和取款
 - 前提条件:用户拥有有效账户
 - 步骤 1:输入有效存款金额
 - 第 2 步:验证存款是否成功
 - 步骤 3:输入有效的提款金额
 - 步骤4:验证提现是否成功
 - 预期结果:用户账户余额正确更新

测试级别:验收

· 验收测试:

· 验收测试是为了确定规范或合同的要求是否符合而进行的测试

按照交付要求进行。验收测试基本上由用户或客户完成。但是,其他股东可以参与此过程。这是黑盒测试

· 评估开发人员交付的系统

· 由客户执行。在开发人员在场的情况下在实验室中执行典型交易 (Alpha 测试)或在没有开发人员的情况下进行现场测试 (Beta 测试)

· 作为交付后维护的一部分,在现场进行回归测试

· **目标:**证明系统满足客户要求并可供使用

(汉密尔顿,2023 年 5 月 13 日)

测试级别:验收

- 测试类型:验收测试
- 测试案例1:存款和取款
- 测试用例编号: 1.1
- 描述:系统应允许用户存入和提取资金。
- 脚步:
 - 使用有效的用户帐户登录系统。
选择“存款”选项。
 - 输入要存入的金额（例如 100 美元）。
 - 点击“提交”按钮。
 - 验证帐户余额是否已更新以反映存款。
选择“提款”选项。
 - 输入要提款的金额（例如 50 美元）。
 - 点击“提交”按钮。
 - 验证帐户余额是否已更新以反映提款。

静态测试和动态测试

- 静态测试**是一种重要的测试技术,其形式包括业务需求评审、功能需求评审、设计评审、代码演练和测试文档评审。这是一项持续的活动,并非仅由测试人员完成。
- 验证**,动态测试部分更加实际,发生在产品本身而不是产品工件或产品表示上。测试用例/条件识别、覆盖范围考虑、执行和缺陷报告等更为正式的流程都标志着动态测试方法。
- 举个例子:**假设这是一份针对某个高度重视安全性的银行网站的业务需求文档(BRD)。BRD中有一节讨论了在该网上银行网站创建账户的各种用户的密码规则。其中一条规则是:**用户不能使用他/她用于其他账户的密码**。这是不可行的。
- 因为网站只能建议用户如何设置登录凭据,而无法施加此限制。因此,此要求不可行-换句话说,无法通过软件实现。

单元测试

请观看此视频：

标题：什么是单元测试？为什么你应该学习它 + 易于理解的示例链接：

<https://www.youtube.com/watch?v=3kzHmaeoDl> 时长：

10:42

分钟

代码示例,4 分钟 - 9 分钟



(Sterkowitz, 2018 年)

黑盒测试与白盒测试



(动画视频机构,2012 年)

请观看此简短的视频：

(虽然有些过时,但涵盖了基础知识)

标题:软件测试教程 - 动画视频时长: 1 分 53 秒链接: <https://www.youtube.com/watch?v=Y7Wg450八十>

黑盒测试与白盒测试

对特定输入的预期结果（输出）进行测试称为**黑盒测试** · 我们不关心模块的内部描述，只关心其

对外部刺激的外部反应 ·**功能测试**

测试模块的内部逻辑和工作称为**白盒测试** ·**结构测试**

黑盒测试与白盒测试

黑盒测试尝试获取能够充分满足系统所有功能要求的输入集。

它不是白盒测试的替代品。

它尝试查找以下类别的错误：

- 功能不正确或缺失， · 界面错误， ·

数据结构或外部数据库

访问错误， · 性能错误, 以及 · 初始化和终止错误。

白盒测试

- 基于内部逻辑（设计、代码等）分析的测试。也称为结构测试。
- 预期结果仍然来自于需求。
- 白盒测试技术主要应用于较低级别的测试（例如，单元和组件）。
- 白盒测试应在测试过程的早期进行，而黑盒测试则倾向于在后期进行。



白盒测试

例如：“逻辑覆盖率”

- 语句:每个语句至少执行一次
- 分支:每个分支至少遍历一次（并且每个入口点都至少被访问一次）
- 条件:每个条件至少为 True 一次,且至少为 False 一次
- 复合条件:每个分支处条件值的所有组合
语句被覆盖（并且每个入口点都被占用）
- 路径:所有程序路径至少遍历一次

白盒 测试

例如：“逻辑覆盖率”

```
FindMean(FILE ScoreFile) { float SumOfScores
= 0.0; int NumberOfScores = 0; float Mean=0.0;
    float Score;读取(ScoreFile, Score); while (!
EOF(ScoreFile) { if (Score > 0.0 )

}

    分数总和 = 分数总和 + 分数;
    分数++; }

    读取 (ScoreFile,Score) ;
}

/* 计算平均值并打印结果 */ if (NumberOfScores > 0) {

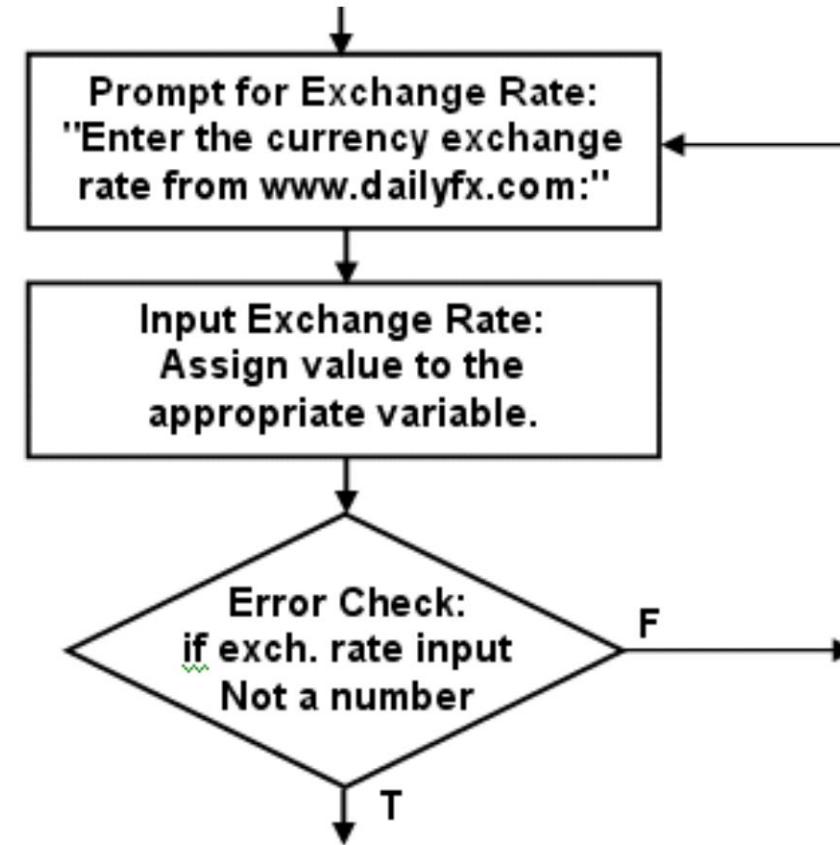
    平均值 = 分数总和 / 分数数量;printf( “ 平均分数为 %f\n” ,
意思是) ;

} 别的

printf ( “文件中未找到分数\n” );
}
```

白盒测试

构建逻辑流程图



http://www.ciese.org/pathways/rwlo/rwlos/2868/Cprogramming%20RWLO/How_to_Create_a_Logic_Flow_Diagram.doc

| Black Box Testing | White Box Testing |
|---|--|
| 1. Black box testing techniques are also called functional testing techniques. | 1. White box testing techniques are also called structural testing techniques. |
| 2. Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester | 2. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. |
| 3. It is mainly applicable to higher levels of testing such as Acceptance Testing and System Testing | 3. Mainly applicable to lower levels of testing such as Unit Testing and Integration Testing |
| 4. Black box testing is generally done by Software Testers | 4. White box testing is generally done by Software Developers |
| 5. Programming knowledge is not required | 5. Programming knowledge is required |
| 6. Implementation knowledge is not required. | 6. Implementation knowledge is required |

(拉姆纳特,nd)

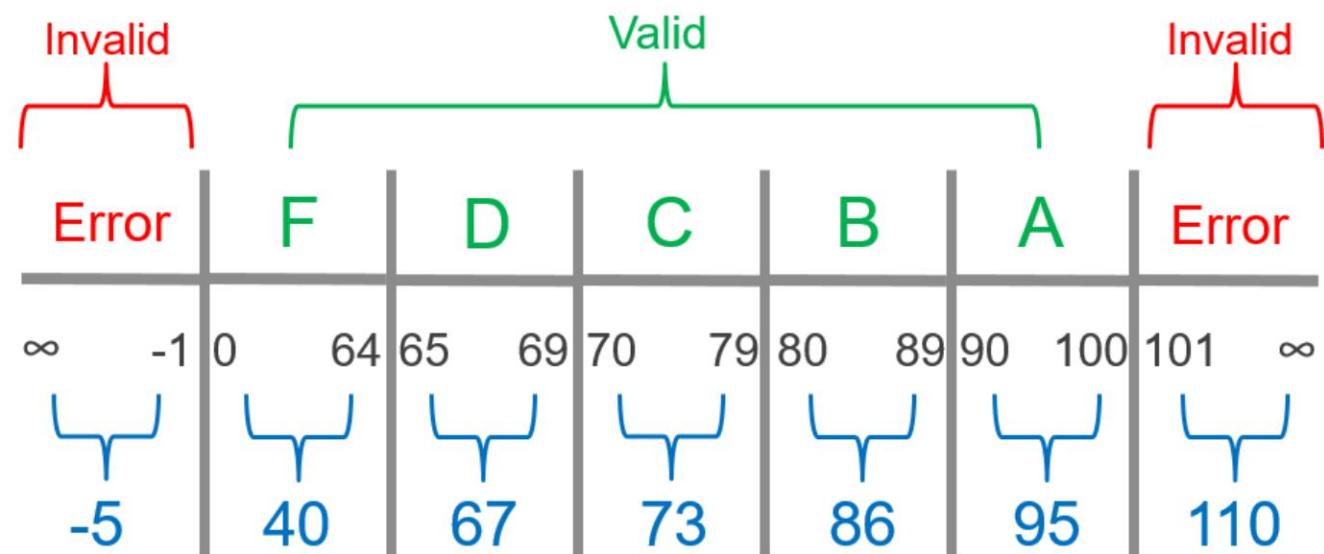
| | |
|---|---|
| 7. Basis for test cases is requirement specification | 7. Basis for test cases is detailed design |
| 8. Techniques that are used in Black Box testing are Boundary-value analysis, Error guessing, Syntax testing, Equivalence partitioning. | 8. Techniques that are used in white box testing Code coverage, data flow testing, path testing. |
| 9. Not suited for algorithm testing | 9. Suitable for algorithm testing |
| 10. It is least time consuming and exhaustive | 10. It is exhaustive and most time consuming type of technique. |
| 11. Best example of Black box testing is Search on Google. User just enters keywords and gets the expected results in turn. | 11. Example: The tester chooses inputs to verifying loops, control flows, return types etc. through the code and determine the suitable outputs |

(拉姆纳特,nd)

常用技术

- 分区测试
- 边界值分析
- 直觉和经验

Equivalence Partitioning



(Derisk 问答, 2017 年)

分区测试（黑盒）

这个想法是对程序的输入空间进行分区,给定分区的每个元素都将在

同样的方式。

例如接口。

需要确定两类情况：

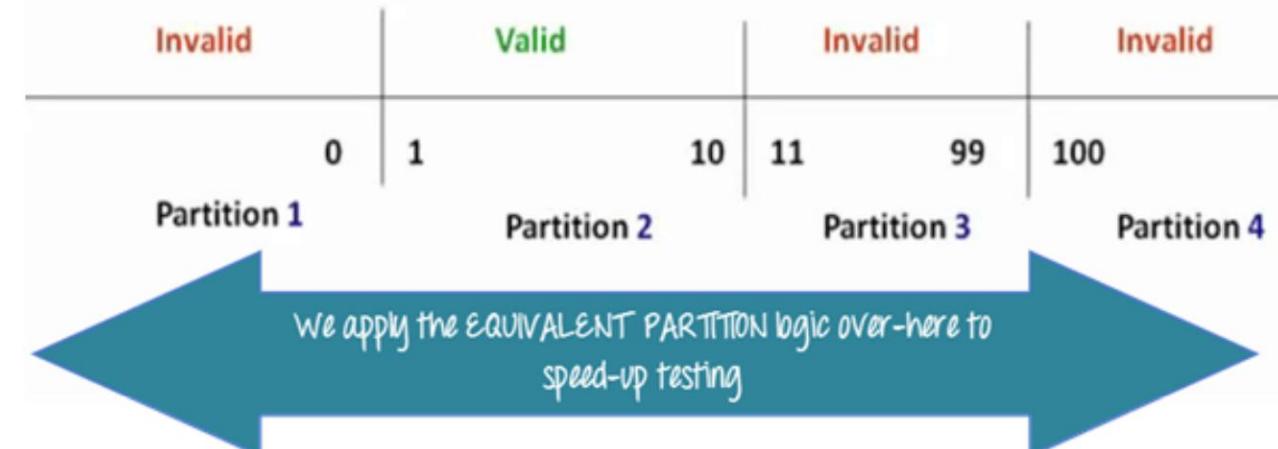
1. **有效**案例（对应有效输入）

2. **无效**

案例（对应无效输入）

如果有兴趣,请看一下:标题:边界值分析和等价划分示例

链接: <https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html>



(Guru99,第 39 页)

边界值分析（黑箱）

一种基于识别和生成测试用例来探索边界条件的技术。

基于自然语言的边界规范通常具有歧义,例如“分数在 0 到 40 之间”

- 范围:确定
范围的端点,略低于或略高于。
- 值的数量:一个文件将包含 1-25 条记录
 - 边界值:空文件 (IV)、包含 1 条记录的文件 (V)、包含 25 条记录的文件 (V) 和包含 26 条记录的文件 (IV)

请观看此视频:标题:软件测试中的边界

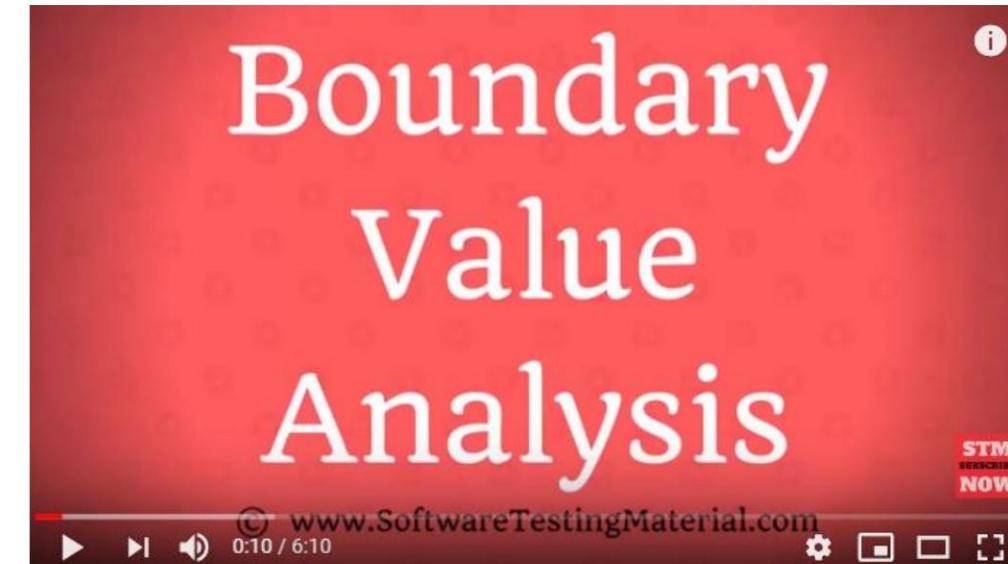
值分析 -

测试设计技术时长: 6 分 10
秒

链接:

<https://www.youtube.com/watch?v=21wOCNHsSU4>

(软件测试材料,2017 年)



为什么要进行等效性和边界分析测试

- 此测试用于将大量测试用例减少为可管理的块。
- 非常明确的指导方针,确定测试用例,而不会影响测试的有效性测试。
- 适用于具有大量变量/输入的计算密集型应用

概括:

- 当实际上不可能测试大量测试用例时,使用边界分析测试单独地
- 使用**两种技术**- 等价分割和边界值分析测试技术
- 在**等价划分中**,首先将一组测试条件划分为可经过考虑的。
- 在边界值分析中,你可以测试等价划分之间的边界
- 适用于计算密集型应用,其中变量表示物理量

直觉与经验

也称为错误猜测、临时测试等。测试人员使用直觉和经验来识别潜在错误并设计测试用例来揭示它们。

对开发人员可能做出的合理但不正确的假设进行设计测试。

- 字符串中的空白或空字符
- 数字字段中的非数字值
- 输入太长或太短

测试是否遵守业务规则约束

敏捷测试生命周期

Impact Assessment
Gather inputs from stakeholders and users, this will act as feedback for next deployment cycle.



Start-Test Plan Meet Up
All stakeholders come together to plan the schedule of testing process, meeting frequency and deliverables.

Daily Standup Meetings
This includes the everyday standup morning meeting to catch up on the status of testing and set the goals for whole day.

Agility Review Meeting
Weekly review meetings with stakeholders meet to review and assess the progress against milestones.

(ReQtest, 2018 年)

敏捷测试方法

测试方法：

- 行为驱动开发 (BDD)
- 验收测试驱动开发 (ATDD)
- 探索性测试

(ReQtest,2018 年)

行为驱动开发 (BDD)

- BDD 改善了项目利益相关者之间的沟通,以便所有成员正确地在开发过程开始之前了解每个功能。
- 开发人员、测试人员和业务分析师之间持续进行基于示例的交流。这些示例称为“场景”,以 Gherkin Given/When/Then语法编写。
- Given-When-Then公式是一个用于指导验收测试编写的模板对于用户故事: ·

(给定)某些背景 · (当)

执行某些操作时 · (然后)应该获得一组特定的可观察后果

一个例子:

- 鉴于我的银行账户有余额,并且我最近没有提款,· 当我尝试提取的金额低于我的卡限额时,· 然后提款应该会完成,不会出现错误或警告

- <https://www.agilealliance.org/glossary/gwt/>

(ReQtest,2018 年)

验收测试驱动开发 (ATDD)

- ATDD 注重让团队成员从不同的角度参与进来,例如客户、开发人员和测试人员。
- 召开三次 Amigos 会议,制定结合客户、开发和测试观点的验收测试。
 - 客户关注的是需要解决的问题,开发关注的是问题如何解决,而测试关注的是可能出现的问题。
 - 验收测试代表了用户的观点,并且描述系统如何运行。
 - 它还有助于验证系统是否按预期运行。在某些情况下实例验收测试都是自动化的。

(ReQtest,2018 年)

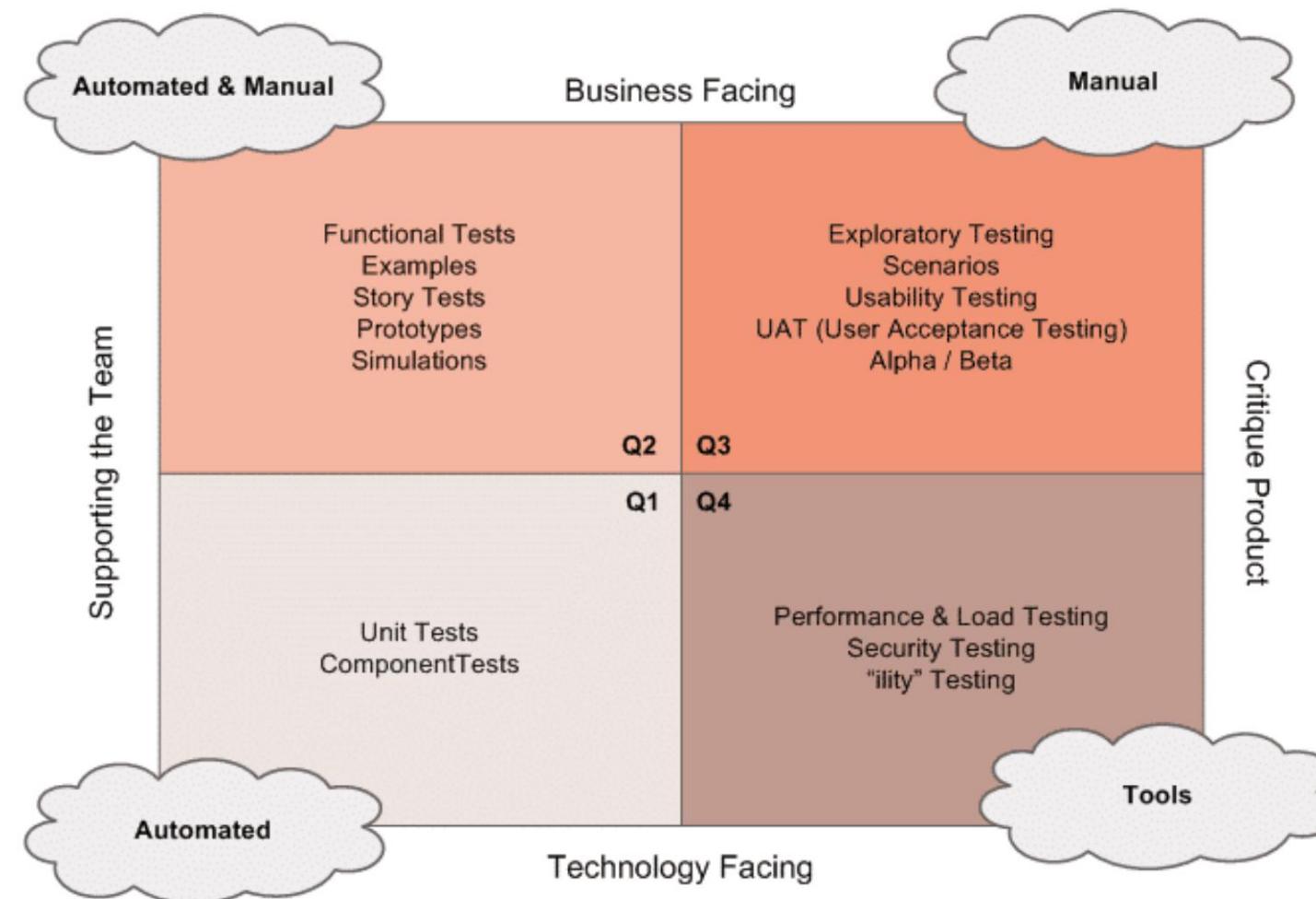
探索性测试

- 在这种类型的测试中,测试设计和测试执行阶段齐头并进。 · 探索性测试强调工作软件,而不是综合性文档。个人和互动比流程和工具更重要。
- 客户合作比合同谈判更有价值。
- 探索性测试更能适应变化。在此过程中,测试人员识别通过探索应用程序来了解应用程序的功能。
- 测试人员尝试学习应用程序,并设计和执行测试计划根据他们的发现。

(ReQtest,2018 年)

敏捷测试策略

Agile Testing Quadrants



(Ghahrai, 2018 年)

敏捷测试的优势

优点：

- 成本和时间效率
- 减少文档
- 灵活且高度适应变化
- 提高质量和
- 提高准确性
- 更加强大
- 它提供了一种从最终用户那里接收定期反馈的方法
- 更好的协作
- 通过每日会议更好地解决问题

(Ishwaran,2016;ReQtest,2018;Gordan,2017)

回归测试

请观看此视频

标题:什么是回归测试?软件测试常见问题解答 - 为什么?
如何做?
什么时候?

时长: 11 分 14 秒

链接:

<https://www.youtube.com/watch?v=xmQuLTarGI4>



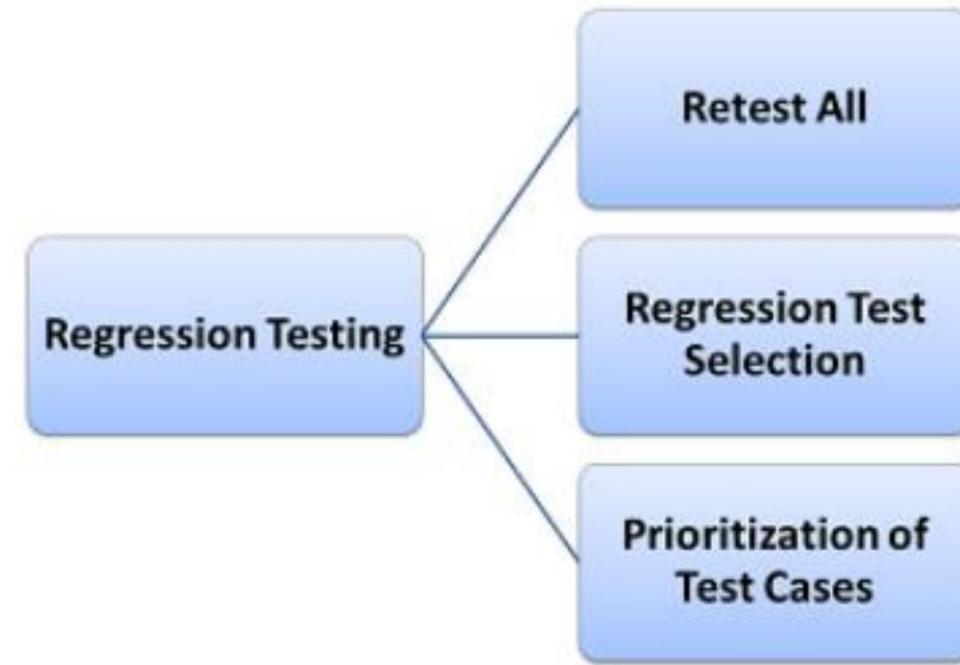
(EvilTester - 软件测试,2017)

回归测试

回归测试:是一种软件测试,用于确认最近的程序或代码更改不会对现有功能产生不利影响。

当有以下需求时,需要进行回归测试:

- 需求发生变化,并根据需求修改代码
- 软件添加了一项新功能
- 缺陷修复
- 修复性能问题



©guru99.com

(Guru99,第 39 页)

为什么回归测试很重要

- 它是质量流程的重要组成部分,可确保代码更改不会损害现有的功能。
- 测试团队应该遵循 “**更改后测试和经常测试**”的方法,这将提供指示新代码是否与现有代码库兼容的结果。
- 包括执行越来越多的测试以及覆盖现有的功能直至产品完成。
- 持续回归测试帮助团队构建表现如下的软件并保持稳定。

(库马尔,2016 年)

集成测试

集成测试

评估一组方法或类的行为

识别接口兼容性、意外参数值或状态交互以及运行时异常

系统测试

行为整合测试
整个系统或独立子系统

构建和冒烟测试

每天或每周进行几次系统测试



(软件测试基础,nd)

可用性测试

确定方法、类、子系统或系统是否满足用户要求

性能测试就是一个例子

- 确定系统或子系统是否能满足基于时间的性能标准

·响应时间指定了所需或最大允许时间限制

软件对查询和更新的响应

·吞吐量指定所需的或最小的查询数量,以及

每分钟或每小时必须处理的交易

用户验收测试 (UAT)

确定系统是否满足用户的要求

- 让**最终用户**参与进来

系统测试验收：· 进行测试
以确保系统已“准备好”进入系统级测试阶段

验收测试是大多数开发项目中非常正式的活动



(Zephyr,nd)

其他类型的测试

Alpha:

在开发环境中进行的实际最终用户测试

Beta

版:在正式发布之前,在用户环境中进行的最终用户测试

性能测试:

- 重点关注与吞吐量、响应时间、内存利用率、输入/输出率等
 - 需求必须以可量化的形式表述。
-

压力测试:

- 重点关注系统在过载或接近过载条件下的行为 (即“推动系统达到失败”)。
- 通常与性能测试一起进行。
- 总体而言,产品应表现出“优雅”的故障和非突然的性能降解。

其他类型的测试

安全测试：

重点关注资源受到未经授权的访问或操纵的脆弱性。

- 计算机资源、媒体等的物理安全，· 登录和密码程序/政策，· 数据或程序访问的授权级别等。

恢复测试：

重点关注从与硬件、软件或人员相关的异常情况中恢复的能力。 · 检测异常情况， · 切换到备用系统，

- 数据恢复，· 维护
审计跟踪等。 · 还可能涉及外部
程序,如存储备份等。

谁测试软件？

程序员

- 单元测试 · 测试
- 伙伴可以测试其他程序员的代码 · 老虎队可以尝试“破解”其他程序员的软件

用户

- 可用性和验收测试 · 经常使用志愿者来测试
- beta 版本

质量保证人员 · 除单元和验收之外的所有测试类型 · 制定测试计划并确定所需的更改

测试政策文件

目的:

代表组织的测试理念，并为测试人员提供应遵循的方向。这可能涵盖新项目和现有系统的维护。

内容

可能包含以下元素：

- 测试的定义
- 员工指南
- 测试过程的描述或目标
- 风险缓解部分
- 手动或自动测试的测试评估（或两个都）
- 要达到的质量水平
- 测试流程改进方法



(Ghahrai, 2018 年)

测试政策文件

请看以下详细示例

- 利物浦大学：
计算服务部测试政策和指南
 - 关联：
<https://www.liverpool.ac.uk/media/livacuk/computingservices/regulations/testingpolicyandstrategies.pdf>
- 测试政策
- 对测试方法所依据的价值观的高层看法 · 测试指南
 - 提供有关如何进行测试、测试内容以及
需要考虑的因素,以及可以采用的各种测试阶段

(利物浦大学,2017 年)

软件错误



<https://reqtest.com/testing-blog/how-to-reduce-the-cost-of-bugs/>

每周活动

除了讲座幻灯片中使用的视频和文章外,这里还有一些供您参考的阅读材料:

1. 标题:测试计划:什么是测试计划,如何创建 (附示例)

链接: <https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html>

2. 标题:如何编写测试用例

链接: <https://www.guru99.com/test-case.html>

频链接: <https://www.youtube.com/watch?v=BBmA5Qp6Ghk>

参考

- 360Logica (nd) 有效的软件测试技巧 随时随地的测试人员![图片]。检索日期:2024 年 7 月 12 日
来自<https://www.360logica.com/blog/effective-software-testing-tips-testers-go/>
- AnimatedVideoAgency (2012 年 11 月 27 日) 软件测试教程-动画 [视频] 已检索
2024 年 7 月 12 日,来自<https://www.youtube.com/watch?v=Y7Wg4508tHo>
- Derisk QA (2017 年 12 月 22 日) 测试设计技术:边界值分析 [图片]。
2024 年 7 月 12 日取自<http://www.deriskqa.com/Article-Boundary-Value-Analysis.html>
- Evil-Tester,软件测试 (2017 年 10 月 20 日) 什么是回归测试?软件测试常见问题解答
- 为什么? 怎么做? 什么时候? [视频] 2024 年 7 月 12 日检索自<https://www.youtube.com/watch?v=xmQuLTarGI4>
- Ghahrai, Amir (2017 年 7 月 2 日) 什么是测试政策文件? 2019 年 6 月 6 日取自<https://www.testingexcellence.com/test-policy-document/>
- Gordan,Alycia (2023 年 1 月 1 日) 敏捷测试自动化的 6 大好处 | 软件测试
材料。2024 年 7 月 12 日检索自<https://www.softwaretestingmaterial.com/6-benefits-agile-testing-automation/>
- Guru99 (nd) 什么是 V 模型?通过 SDLC 和 STLC 案例研究进行学习。[图片] 2024 年 7 月 12 日从<https://www.guru99.com/software-testing-lifecycle.html>检索

参考

- Guru99 (nd) 边界值分析与等价划分示例 [图片]
2024 年 7 月 12 日取自<https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html>

- Guru99 (nd) 什么是回归测试? 测试用例、工具和示例。检索日期: 2024 年 7 月 12 日
来自<https://www.guru99.com/regression-testing.html>
- 汉密尔顿。(2023 年 5 月 13 日)。软件测试中的测试级别。2024 年 7 月 12 日检索自
<https://www.guru99.com/levels-of-testing.html>
- Ishwaran, Anuradha (2016 年 6 月 21 日)《敏捷环境中软件测试的 5 大好处》。
2024 年 7 月 12 日取自<http://www.tothenew.com/blog/5-benefits-of-software-testing-in-an-agile-environment/>
- Kumar, Pavan (2016 年 1 月 20 日) 回归测试的重要性。2018 年 11 月 10 日检索自<https://www.utest.com/articles/the-importance-of-regression-testing>

- 金属部署电阻器 (nd) 联系我们 [图片] 2018 年 9 月 8 日检索自
<http://www.mdresistor.com/en/harmonic-filter-resistor/contact-us-email-meta-deploye-resistor-4/>

参考

- Pexels.com (nd) analysis-blackboard-board-bubble [图片] 于 2024 年 7 月 12 日从<https://www.pexels.com/photo/analysis-blackboard-board-bubble-355952/> 检索
- Ramnath (nd) 问题:借助示例描述黑盒测试和白盒测试之间的区别。2024 年 7 月 12 日检索自<http://www.ques10.com/p/2150/describe-the-difference-between-black-box-and-wh-1/>
- ReQtest (2018 年 7 月 18 日)敏捷测试 原则、方法和优势。2024 年 7 月 12 日从<https://reqtest.com/testing-blog/agile-testing-principles-methods-advantages/>检索
- Satzinger,Jackson 和 Burd。(2016 年)《不断变化的世界中的系统分析与设计》， Cengage 学习。
- 软件测试基础 (nd)集成测试。[图片] 2023 年 7 月 7 日检索自 <http://softwaretestingfundamentals.com/integration-testing/>
- 软件测试帮助 (2023 年 6 月 18 日) 静态测试和动态测试 这两种重要测试技术之间的区别。2024 年 7 月 12 日从[https://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/](http://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/) 检索

参考

- Soni, Sujit (2015 年 3 月 27 日) 逻辑工程师 – STLC 生命周期。[图片] 2023 年 7 月 12 日检索
摘自<http://automatetesterzone.blogspot.com/2015/>

- Sterkowitz, Andy (2018 年 11 月 4 日) 于 2024 年 7 月 12 日检索自 “什么是单元测试?为什么
你应该学习它 + 易于理解的示例 [视频]。2018 年 11 月 10 日检索自<https://www.youtube.com/watch?v=3kzHmaeoZDI>

- 利物浦大学 (2017 年 3 月 29 日) 计算服务系 : 测试政策和
指南。2018 年 11 月 10 日检索自<https://www.liverpool.ac.uk/media/livacuk/computingservices/regulations/testingpolicyandstra特吉.pdf>

- Zephyr (nd) 运行有效用户验收测试周期的科学 [图片] 2018 年 11 月 10 日取自<https://www.getzephyr.com/resources/whitepapers/science-running-effective-user-acceptance-testing-cycles>

问题？



(pexels.com,nd)

谢谢