软件质量

第8讲

鲁赞娜·奇奇安 /乔恩·伯德 / 皮特·贝内特
助教：Alex Elwood、Alex Cockrean、Casper Wang

概述

·软件质量以及如何实现它 ·测试驱动开发

·白盒测试

·黑盒测试

软件质量

为什么软件质量相关：

吟游诗人的案例（双子座）

Google Bard AI mistake just cost Google over $100 billion

By Philip Michaels last updated 8 days ago

AI-powered chatbot makes costly error in early demo

Comments (0)

https://www.tomsguide.com/news/google-bard-ai-is-off-to-an-embarrassing-start

Isabel Angelo
@IsabelNAngelo · **Follow**

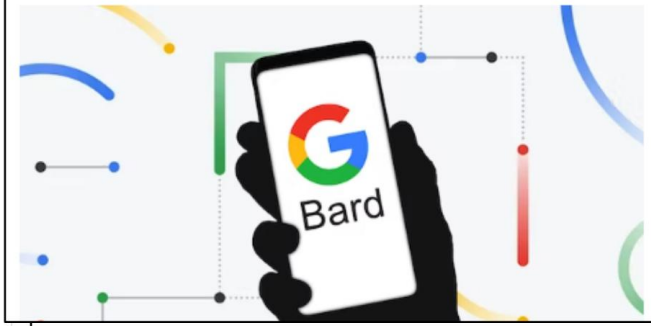
Unfortunately a simple google search would tell us that JWST actually did not "take the very first picture of a planet outside of our own solar system" and this is literally in the ad for Bard so I wouldn't trust it yet

Google @Google
Bard is an experimental conversational AI service, powered by LaMDA. Built using our large language models and drawing on information from the web, it's a launchpad for curiosity and can help simplify complex topics → goo.gle/3HBZQtu

Introducing Bard,
an experimental conversational AI service
powered by LaMDA

You can use Bard to —
Plan a friend's baby shower
Compare two Oscar nominated
Get lunch ideas based on what's in your fridge

The chatbot generated an incorrect fact about the James Webb Space Telescope in its very first public demo. The incident has dramatically highlighted one of the most pertinent dangers for marketers using AI: it doesn't always tell the truth.



https://www.thedrum.com/news/2023/02/09/attention-marketers-google-s-1000bn-bard-blunder-下划线-current-dangers-using-ai

为什么软件质量很重要？

- 声誉
- 产品和维护成本
- 软件认证
- 组织认证
- 合法性
- 道德/伦理行为准则

软件质量是多维的

·主观或“适用性”:由单个用户感知（例如,GUI 的美观、功能缺失……）

·目标或“符合要求”:可以作为产品的属性来衡量（例如,详细文档、错误数量、遵守法规……）

·实用:这对您的团队和客户意味着什么？

质量模型：ISO/IES25010

- **Functional Suitability**
  - Functional Completeness
  - Functional Correctness
  - Functional Appropriateness
- **Performance Efficiency**
  - Time Behaviour
  - Resource Utilisation
  - Capacity
- **Compatibility**
  - Co-existence
  - Interoperability
- **Usability**
  - Appropriateness
  - Realisability
  - Learnability
  - Operability
  - User Error Protection
  - User Interface Aesthetics
  - Accessibility
- **Reliability**
  - Maturity
  - Availability
  - Fault Tolerance
  - Recoverability
- **Security**
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Authenticity
  - Accountability
- **Maintainability**
  - Modularity
  - Reusability
  - Analysability
  - Modifiability
  - Testability
- **Portability**
  - Adaptability
  - Installability
  - Replaceability

迈向软件质量的步骤：

- 使用标准开发流程
- 使用编码标准
  - 符合行业标准（例如ISO、安全等）
  - 一致的代码质量
  - 从一开始就确保安全
  - 降低开发成本并加快上市时间
- 定义和监控指标（缺陷指标和复杂性指标）
  - 高复杂性导致更多缺陷
- 识别并消除缺陷
  - 进行人工审核
  - 使用测试

测试

毛罗·佩泽和米哈尔·杨。软件测试和分析　过程、原理和技术。威利，2007。

测试过程:关键要素和关系

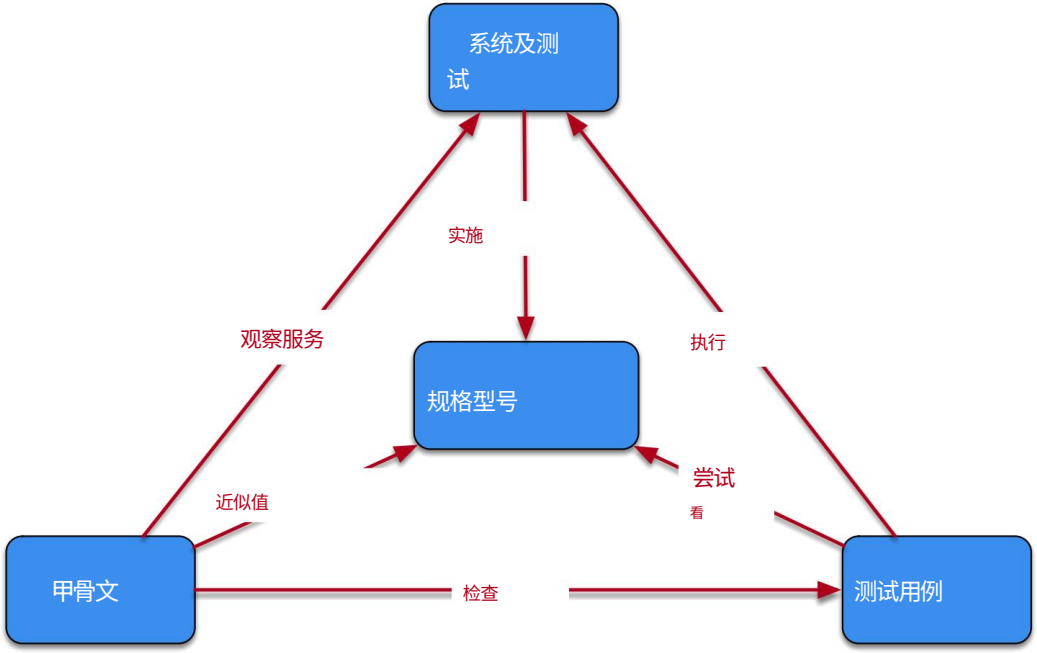


图 6.1:控制测试过程的主要元素和关系。

来自:M. Staats、MW Whalen 和 MPE Heimdahl。程序、测试和预言:重新审视测试的基础。软件工程 (ICSE),2011 年第 33 届国际会议,第 391-400 页。 IEEE,2011。

测试:白盒

毛罗·佩泽和米哈尔·杨。软件测试和分析　过程、原理和技术。威利,2007。

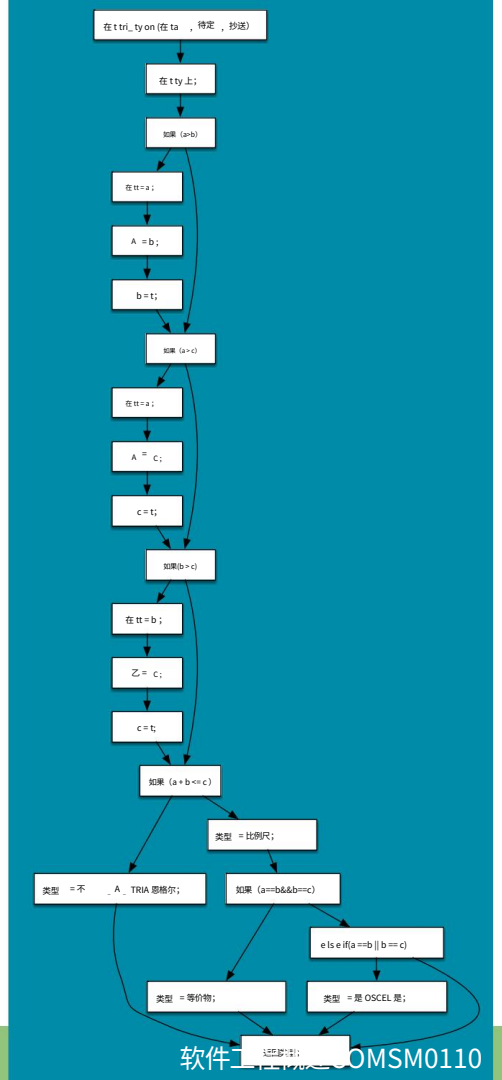
白盒测试

- 访问软件“内部结构”：
  - 源代码
  - 运行时状态
  - 可以跟踪执行情况。
- 白盒测试利用这一点
  - 使用代码来衡量
    - 覆盖范围
      - 多种方式
  - 推动测试的生成–以最大化覆盖范围

```
        int tri_type(int a, int b, int c) {
1           int 类型;
2           如果 (a > b)
3               { intt=a;a=b;b=t; }
4           如果 (a > c)
5               { intt = a;a=c;c=t; }
6           如果 (b > c)
7               {intt=b;b=c;c=t; }
8           如果（a + b <= c）
91              类型= NOT_A_TRIANGLE;
01          别的 {
11              类型=比例尺;
21              如果（a==b&&b==c）
31                  类型=等距;
41              否则如果（a == b || b == c）
51                  类型=等腰;
61          }
71      返回类型;
819     }
```

白盒测试

- 访问软件“内部结构”：
  - 源代码
  - 运行时状态
  - 可以跟踪执行情况。
- 白盒测试利用这一点
  - 用代码来衡量–
    - 覆盖范围
      - 多种方式
  - 推动测试的生成–以最大化覆盖范围。



白盒测试

·覆盖率指标：

·语句覆盖率 ·分支覆盖率 ·

Def-Use 或数据流覆盖率

· MC/DC（修改条件/决策覆盖率）

·突变覆盖率……

· 规定的指标,例如民用飞机软件的DO178-B/C标准

· 非关键-语句覆盖

· 安全关键 - MC/DC 覆盖范围

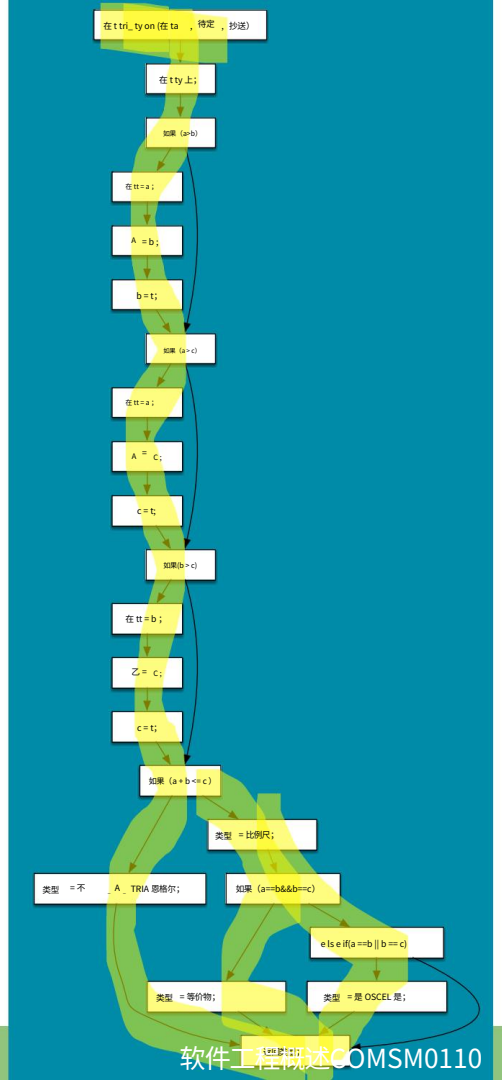
报表覆盖范围

·测试输入应共同具有-
执行每条语句

·如果某个语句总是出现错误–
执行时会被检测到

·计算公式为：

$$覆盖率 = \frac{|执行的语句|}{|总陈述|}$$



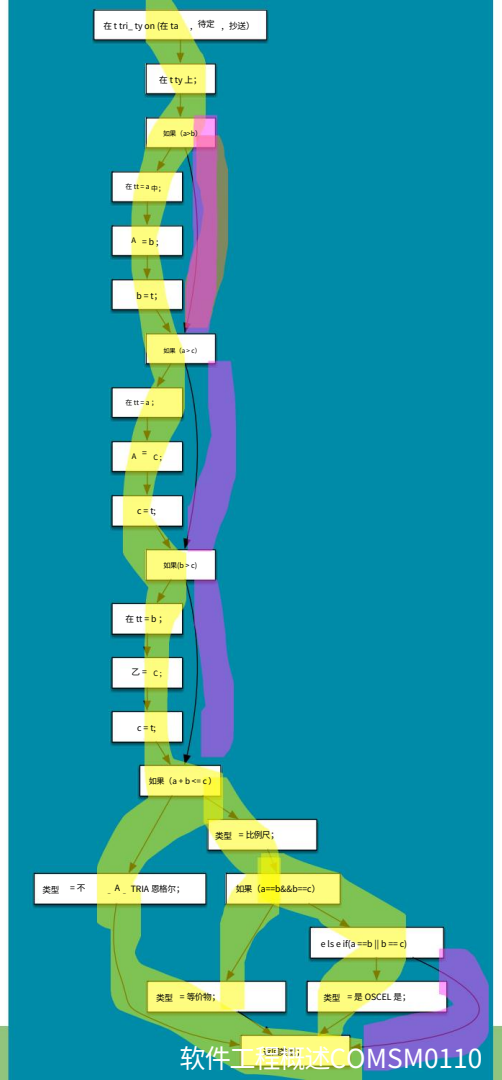
分支机构覆盖范围

·测试输入应共同具有-
执行每个分支

·包含报表覆盖范围

·计算公式为：

$$\text{覆盖率} = \frac{|\text{已执行的分支}|}{|\text{分支机构总数}|}$$

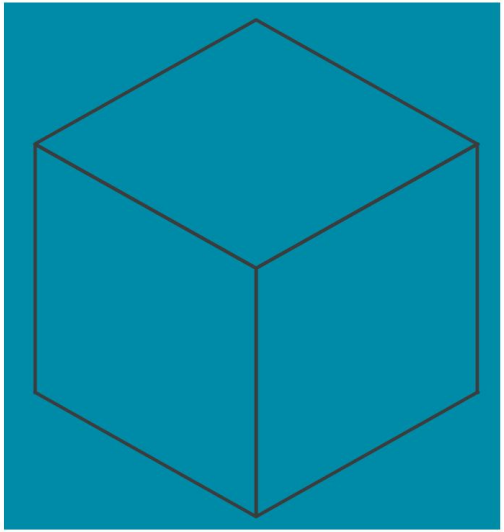


测试:黑盒

毛罗·佩泽和米哈尔·杨。软件测试和分析　过程、原理和技术。威利,2007。

黑盒测试

- 无法访问“内部”
  - 可能可以访问,但不想　　想要
- 我们了解接口
  - 参数
  - 可能的功能/方法
- 我们可能有某种形式的规范文件



测试挑战

·多种不同类型的输入

·输入选择可以通过多种不同方式影响输出

·几乎无限数量的可能输入和组合

等价划分（EP)方法

通过分析程序接口来识别测试

1. 将程序分解为“功能单元”
2. 确定这些单元的输入/参数
3. 对于每个输入

    a) 确定其局限性和特征

    b) 定义“分区” 值类别

    c) 确定类别之间的约束

    d) 编写测试规范

示例 – 生成评分组件

该组件通过了考试分数（满分 75 分)和课程作业 (c/w) 分数（满分 25 分），并据此生成课程的等级,范围为“A”到“D”。

成绩根据总分计算,总分是考试分数和 C/W 分数的总和,如下所示： 大于或等于 70 - A 大于或等于 50,但小于 70 - B 大于或等于 30,但小于 50 - C

小于 30 - D

如果标记超出其预期范围,则会生成故障消息（“FM”）。所有输入都作为整数传递。

EP – 1. 分解为功能单元

- 分成更小的单元是很好的做法
  - 可以生成更严格的测试用例。
  - 发现故障时更容易调试。
- 例如：将大型Java应用程序划分为其核心模块/包
- 已经是评分组件示例的功能单元

EP – 2. 识别输入和输出

·对于某些系统,这很简单 ·例如,三角程序： ■
输入:3 个数字， ■输出:1 个字符串
·例如,评分组件

■输入:2 个整数:考试分数和课程作业分数■输出:1 个成绩
字符串 ·对于其他不太如此的字
符串。请考虑以下事项： ·手机应用程序。 ·带有
Flash 组件的网
页。

EP – 3.a 识别类别

类别	描述
有效的	有效考试成绩
	有效的课程作业分数
	有效总分
无效的	考试成绩无效
	无效的课程作业分数
	总分无效

EP:3.b 定义“分区”-值类别

·输入的有效值范围/值特征

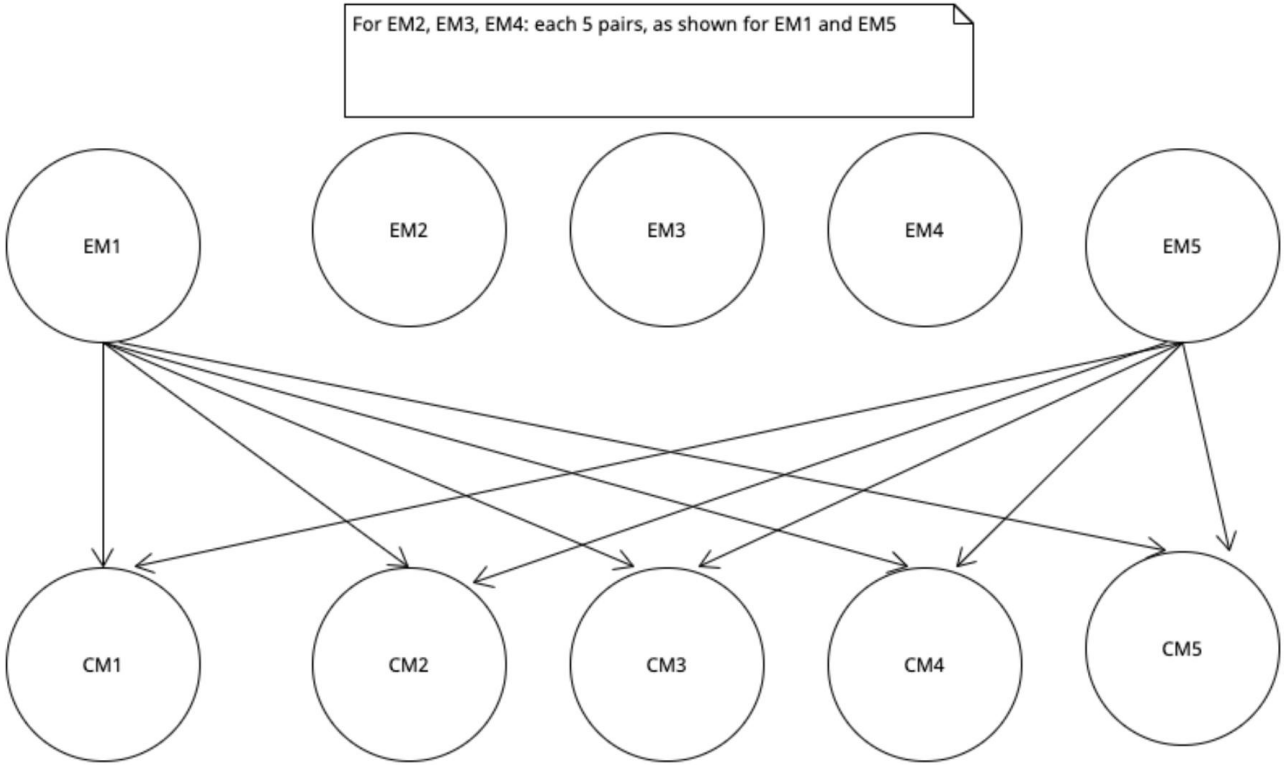
类别	描述	分割
有效的	EM_1 有效考试分数	0 ⩽ 考试分数 ⩽ 75
	CM_1 有效课程作业分数	0 ⩽ 课程作业分数 ⩽ 25
无效的	EM_2 无效考试分数	考试分数 > 75
	EM_3 无效考试分数	考试分数 < 0
	EM_4 无效考试分数	字母的
	EM_5 无效考试分数	其他实数（EM_1 之外）
	CM_2 课程作业分数无效	课程作业分数 > 25
	CM_3 课程作业分数无效	课程作业分数 < 0
	CM_4课程作业标记无效	字母的
	CM_5 课程作业分数无效	其他实数（CM_1 之外）

EP – 3.c 识别类别之间的约束

· 并非所有类别都可以相互组合

类别		健康)状况
有效考试成绩	输入1	0 ⩽ 考试分数 ⩽ 75
考试成绩无效	输入2	考试分数 > 75
考试成绩无效	EM_3	考试分数 < 0
考试成绩无效	EM_4	字母的
考试成绩无效	EM_5	其他实数
有效的课程作业分数	CM_1	0 ⩽ 课程作业分数 ⩽ 25
无效的课程作业分数	CM_2	课程作业分数 > 25
无效的课程作业分数	CM_3	课程作业分数 < 0
无效的课程作业分数	CM_4	字母的
无效的课程作业分数	CM_5	其他实数

EP – 3.d 编写测试规范



帕隆斯和埃斯凯斯。 oreac oese 明确地避开了 ony e snge paron eng arge。确定了十九个分区，导致十九个测试用例。

示例：输入和预期输出

与从输入测试标记派生的分区相对应的测试用例是：

测试用例	1	2	3
输入（考试分数）	44	-10	93
输入(c/wmark)	15	15	15
总分（计算得出）	59	5	108
分区测试（ofexam 标记）	0⩽e⩽75	且 < 0	且 > 75
输出输出	B	调频	调频

边界值

- 最常见的错误发生在“边缘”情况下
  - 测试刚好低于边界值
  - 测试刚好高于边界值
  - 测试边界值

我们如何使用它？

- Java中应用的测试：使用JUnit

  ·使用“断言”来测试代码

  ·请允许我们说明应该是什么情况

  ·如果断言不成立，JUnit 的日志记录机制会报告失败

  ·可以使用多种类型的断言，例如，assertEquals(expected,actual);断言真（条件）；断言假（条件）；断言（值，匹配函数）

审查

·什么是软件质量？ ·测试规范的关键要素和关系是什么？ ·如何进行白盒测试？ ·如何进行黑盒测试？