软件质量工程 Week 8

Nite

2025-05-08T17:18:00+11:00

阅读 PDF

测试工具 (Tools for Testing)

概述

本周课程主要探讨用于软件测试的各种工具和方法，重点介绍**有限状态机 (Finite State Machines, FSMs)** 和**马尔可夫链 (Markov Chains)** 在测试中的应用，以及**测试管理工具 (Test Management Tools)** 的功能。

需要掌握

- **如何对面向对象的程序执行测试。**
- **什么是有限状态机 (Finite State Machine)，以及它如何用于测试。**
- **什么是马尔可夫链 (Markov Chain)，以及它如何用于测试。**
- 了解可用于测试管理的工具类型。

有限状态机 (Finite State Machines, FSMs)

有限状态机 (FSM) 是一种用于对程序执行或行为进行建模的中间形式化方法，它在表达能力和简洁性之间取得了平衡。FSM 是描述对象在一段时间内动态行为的模型。

FSM 的元素

- **状态 (States)**：对象值的一组集合；对象等待事件发生的一段时间；对象执行活动的一段时间。
- **转换 (Transitions)**：对象在特定状态下对事件发生的响应。包括事件触发器 (Event Trigger)（例如，启用转换的事件）、警戒条件 (Guard Condition)（例如，布尔表达式）、效果 (Effect)（例如，一个动作）和目标状态 (Target State)。
- **输入 (Inputs)**：一组有限的值；很大程度上取决于状态；可以包含条件。
- **输出 (Outputs)**：很大程度上取决于状态和输入；与输入条件的对应关系；与状态相关联的结果。

FSM 的表示方法

状态和转换信息的描述方式：

- **状态少时**：图形表示（节点和箭头），易于创建。

- **状态多时**：转换和输出关系的表格表示，输入和状态集合会产生影响，易于工具处理。
- **状态多，转换少时**：转换列表 (Transition lists)，典型的转换列表表示为：$(input\_A, output\_A, state\_A), ...,$

状态转换表 (State transition table) 示例： | 当前状态 (Current State) | 输入 (Input) | 下一状态 (Next State) | 输出 (Output) | | :————————–| :———–| :————————–| :———–| | | | | |

FSM 的主要局限性

- FSM 会随着设计的变化而变化。
- FSM 可能会因状态、转换、输入和输出的不可预见问题而出错。
- FSM 严重依赖于表示和操作。
- **FSM 状态的错误表示**：
  - **丢失状态 (Missing states)**：例如，缺少初始状态。
  - **不正确的状态 (Incorrect states)**：例如，指定了不正确的行为。
  - **多余的状态 (Extra states)**：例如，错误的输入组合。
- **FSM 转换的错误表示**：
  - **丢失转换 (Missing transitions)**：例如，状态序列不连贯。
  - **不正确的转换 (Incorrect transitions)**：例如，错误地连接了两个状态。
  - **多余的转换 (Extra transitions)**：例如，与产品不对应的转换。

马尔可夫链 (Markov Chains)

马尔可夫链 (Markov Chains) 是具有随机转换的 FSM，这些转换用概率值描述。

- 表示在时间 n 从状态“i”转换到时间 $n+1$ 状态“j”的概率。
- **从一个状态出发的所有出向转换的概率之和为 1。**
- 概率估计方法：
  - A - 基于对应用程序的专家知识进行初始概率估计。
  - B - 也可以通过先前收集的统计数据支持估计。
  - A 和 B 的结合。
- **注意**：
  - **对于生成状态和转换覆盖非常有用。**
  - **专注于产品最常用的部分。**

微软案例：测试软件系统中的异步回调行为

微软的一项专利 (US7500149B2) 描述了使用 FSM 来建模软件系统中的异步回调 (Asynchronous Callbacks)。

- **描述 (Description)**：使用 FSM 对软件系统中的异步回调进行建模。
- **主要动机 (Key Motivation)**：存在数百万行源代码分布在必须交互的不同模块或例程中。
- **主要成果 (Key Outcome)**：提高软件系统的性能和可靠性。

其他测试工具和方法

测试管理工具 (Test Management Tools)

测试管理工具用于支持软件测试的各个方面。在选择测试管理工具时，可以考虑以下功能点：

- 支持描述软件需求。
- **软件需求与测试之间的连接。**
- 将测试分组到测试套件 (Test Suites) 或类似单元中。
- 支持手动测试创建。
- 支持（自动化）测试创建。
- 多种测试环境支持。
- 报告功能。
- 支持团队成员之间的协作。
- 与项目管理的连接（里程碑 (Milestones)、截止日期 (Deadlines)、可交付成果 (Deliverables)、轻松创建新任务等）。
- 缺陷跟踪 (Defect Tracking) 功能或与辅助平台的连接。
- 价格（不一定总能获取到）。

从服务器日志到概率 (From Server Logs to Probabilities)

服务器日志记录了用户访问页面的信息，这些信息可以用来分析用户行为，并为将 FSM 转换为马尔可夫链提供概率数据。例如，一条典型的服务器日志条目可能如下（为简化可视化，每字段一行）：

```
97.185.220.155
user
[09/Oct/2012:17:24:11 +0900]
"GET /platform/folders/user/document.html HTTP/1.1"
401
6012
"https://perfectsoftware.com/platform/folders/user"
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.202 Saf
```

其中，字段 5（用户访问的页面）和字段 8（包含所选链接的源页面）对于分析页面转换非常重要。通过处理大量的日志数据，可以统计不同页面之间的转换频率，从而估算出马尔可夫链中的转换概率。**这对于在组合爆炸导致无法进行全覆盖测试的系统中，根据功能的使用频率来分配测试资源非常有用。**

可能的考点与示例考题

可能考点

- **有限状态机 (FSM) 的定义、元素（状态、转换、输入、输出）及其在测试中的应用。**
- FSM 的不同表示方法（图形、表格、转换列表）及其适用场景。
- **FSM 的主要局限性，特别是状态和转换的错误表示。**
- **马尔可夫链 (Markov Chain) 的定义，与 FSM 的关系（随机转换、概率）。**
- **马尔可夫链中转换概率的意义和估计方法。**
- 如何利用服务器日志数据为马尔可夫链生成转换概率。
- **测试管理工具 (Test Management Tools) 的主要功能和选择标准。**
- 面向对象程序的测试策略（如类测试、自底向上集成测试）。

示例考题

问题 1：什么是有限状态机 (FSM)？请描述其基本元素，并说明 FSM 在软件测试中如何被使用。

参考答案：

- **中文：**
  - 定义：有限状态机 (FSM) 是一种数学计算模型，用于设计计算机程序和数字逻辑电路。它是一个抽象机器，可以处于有限数量的状态之一。机器在任何给定时间只能处于一个状态。当它接收到输入时，它可以从一个状态转换到另一个状态，并且可以产生输出。
  - 基本元素：
    1. **状态 (States)**：系统可能存在的条件或情况的集合。
    2. **输入 (Inputs)**：可能触发状态改变的事件或信号的集合。
    3. **转换 (Transitions)**：基于当前状态和接收到的输入，从一个状态到另一个状态的规则或函数。
    4. **输出 (Outputs)**：在转换发生时或处于特定状态时产生的动作或信号的集合（取决于 FSM 的类型，如 Mealy 机或 Moore 机）。
    5. 初始状态 (Initial State)：机器开始操作时的状态。
  - 在软件测试中的使用：FSM 可以用来对软件的行为进行建模，特别是那些具有明确状态和转换的系统（例如，用户界面、协议处理、设备控制器）。测试人员可以基于 FSM 模型设计测试用例，以确保：
    * 所有指定的状态都可以达到。
    * 所有指定的转换都可以被正确触发。
    * 对于每个状态和输入组合，系统都能转换到正确的下一个状态并产生正确的输出。
    * **覆盖不同的状态序列和转换路径，包括有效和无效的场景。**这有助于系统地测试软件的动态行为。
- **English:**
  - Definition: A Finite State Machine (FSM) is a mathematical model of computation used to design computer programs and digital logic circuits. It is an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time. It can transition from one state to another when it receives an input, and it may produce an output.
  - Basic Elements:
    1. **States**: A set of conditions or situations in which the system can exist.
    2. **Inputs**: A set of events or signals that can trigger a change of state.
    3. **Transitions**: Rules or functions that define the move from one state to another based on the current state and the received input.
    4. **Outputs**: A set of actions or signals produced when a transition occurs or while in a particular state (depending on the type of FSM, e.g., Mealy machine or Moore machine).
    5. Initial State: The state in which the machine begins operation.
  - Use in Software Testing: FSMs can be used to model the behavior of software, especially systems with well-defined states and transitions (e.g., user interfaces, protocol handlers, device controllers). Testers can design test cases based on the FSM model to ensure that:
    * All specified states are reachable.
    * All specified transitions can be correctly triggered.
    * For each state and input combination, the system transitions to the correct next state and produces the correct output.
    * **Different state sequences and transition paths are covered, including valid and invalid scenarios.** This helps in systematically testing the dynamic behavior of the

software.

问题 2：马尔可夫链 (Markov Chain) 与有限状态机 (FSM) 有何不同？在测试中，为什么以及如何使用马尔可夫链？

参考答案：

- **中文：**
  - 不同点：
    - **主要区别在于转换的性质。**在标准的 FSM 中，从一个状态到另一个状态的转换通常是确定性的，由特定的输入触发。
    - **马尔可夫链是一种随机过程，其状态转换是概率性的。**每个可能的转换都关联一个概率，表示从当前状态转移到下一个特定状态的可能性。一个关键特性是“无记忆性”，即下一个状态只取决于当前状态，而与到达当前状态的历史无关。
  - 在测试中的使用原因和方法：
    - **原因：**对于复杂的系统，用户行为往往不是确定性的，而是具有一定的随机性或遵循某种使用模式。马尔可夫链可以更好地模拟这种概率性行为。**它可以帮助测试人员优先测试那些最常被用户执行的路径或功能，从而在有限的测试资源下最大化缺陷发现的效率。**
    - **方法：**
      1. 首先，可以基于系统的功能或用户场景构建一个 FSM 模型。
      2. 然后，通过分析用户数据（如服务器日志、使用统计）或基于专家经验，为 FSM 中的每个转换分配概率，将其转换为马尔可夫链。
      3. **使用马尔可夫链生成测试序列。**可以根据转换概率随机游走状态图，或者优先选择概率较高的路径。
      4. 这种方法有助于生成更符合实际用户使用情况的测试用例，特别适用于基于使用模型的测试 (Usage-Based Testing) 或统计测试 (Statistical Testing)。
- **English:**
  - Differences:
    - **The primary difference lies in the nature of transitions.** In a standard FSM, transitions from one state to another are typically deterministic, triggered by specific inputs.
    - **A Markov Chain is a stochastic process where state transitions are probabilistic.** Each possible transition is associated with a probability, indicating the likelihood of moving from the current state to a specific next state. A key property is “memorylessness,” meaning the next state depends only on the current state and not on the history of how the current state was reached.
  - Why and How to Use Markov Chains in Testing:
    - **Why:** For complex systems, user behavior is often not deterministic but rather exhibits some randomness or follows certain usage patterns. Markov chains can better model this probabilistic behavior. **It can help testers prioritize testing paths or functionalities that are most frequently executed by users, thereby maximizing defect detection efficiency with limited testing resources.**
    - **How:**
      1. First, an FSM model can be built based on the system’s functionalities or user scenarios.
      2. Then, by analyzing user data (e.g., server logs, usage statistics) or based on expert

experience, probabilities are assigned to each transition in the FSM, converting it into a Markov chain.

3. **Test sequences are generated using the Markov chain.** This can involve randomly traversing the state diagram according to transition probabilities or prioritizing paths with higher probabilities.
4. This approach helps generate test cases that are more representative of actual user usage, particularly useful for Usage-Based Testing or Statistical Testing.

问题 3：假设你正在评估一款测试管理工具 (Test Management Tool) 用于你的团队。你会关注该工具的哪些关键功能？请列举至少五项，并解释为什么它们很重要。

参考答案：

- **中文：**
  - 关键功能（至少五项）：
    1. **需求可追溯性 (Requirements Traceability)：**工具应能将测试用例与软件需求关联起来。
       - 重要性：确保每个需求都有相应的测试覆盖，防止需求遗漏；当需求变更时，能快速定位受影响的测试用例。
    2. **测试用例管理 (Test Case Management)：**包括创建、编辑、组织（如分组成测试套件）、版本控制和复用测试用例。
       - 重要性：高效地管理大量测试用例，方便维护和更新，提高测试效率。
    3. **测试执行管理 (Test Execution Management)：**支持计划测试周期、分配测试任务、记录测试结果（通过/失败/阻塞等）、以及跟踪执行进度。
       - 重要性：系统化地执行测试，实时监控测试进展，确保测试按计划进行。
    4. **缺陷跟踪集成 (Defect Tracking Integration)：**工具应能与缺陷跟踪系统（如 Jira, Bugzilla）无缝集成，或者自身具备强大的缺陷管理功能。
       - 重要性：当测试失败时，能方便地创建和关联缺陷报告，跟踪缺陷的生命周期直至修复和验证，形成闭环管理。
    5. **报告和分析 (Reporting and Analytics)：**提供各种可定制的报告，如图表、仪表盘，展示测试覆盖率、通过率、缺陷趋势等。
       - 重要性：为项目管理者和团队提供关于测试状态和产品质量的清晰视图，帮助做出明智决策，并识别改进领域。
- **English:**
  - Key Features (at least five):
    1. **Requirements Traceability:** The tool should be able to link test cases to software requirements.
       - Importance: Ensures that every requirement has corresponding test coverage, preventing requirement omissions; allows for quick identification of affected test cases when requirements change.
    2. **Test Case Management:** Includes creating, editing, organizing (e.g., into test suites), versioning, and reusing test cases.
       - Importance: Efficiently manages a large number of test cases, facilitates maintenance and updates, and improves testing efficiency.
    3. **Test Execution Management:** Supports planning test cycles, assigning test tasks, recording test results (pass/fail/blocked, etc.), and tracking execution progress.

    * Importance: Systematically executes tests, monitors testing progress in real-time, and ensures tests proceed as planned.
4. **Defect Tracking Integration:** The tool should seamlessly integrate with defect tracking systems (e.g., Jira, Bugzilla) or have robust built-in defect management capabilities.
    * Importance: Allows for easy creation and linking of defect reports when tests fail, and tracks the defect lifecycle through to resolution and verification, forming a closed-loop management process.
5. **Reporting and Analytics:** Provides various customizable reports, such as charts and dashboards, displaying test coverage, pass rates, defect trends, etc.
    * Importance: Offers project managers and the team a clear view of the testing status and product quality, aiding in informed decision-making and identifying areas for improvement.