**Structure Based or Whitebox Testing Techniques**

**Định nghĩa.**

Kiểm thử cấu trúc, còn được gọi là kiểm thử hộp kính hoặc kiểm thử hộp trắng là một cách tiếp cận mà các bài kiểm thử bắt nguồn từ kiến ​​thức về cấu trúc của phần mềm hoặc việc triển khai nội bộ. Các tên gọi khác của kiểm thử cấu trúc bao gồm kiểm thử hộp rõ ràng, kiểm thử hộp mở, kiểm thử theo hướng logic hoặc kiểm thử hướng theo đường dẫn.

**Công dụng**

Trọng tâm chính của thử nghiệm hộp trắng là đảm bảo rằng mã của ứng dụng là chắc chắn và an toàn, đồng thời luồng của mỗi quy trình là tự nhiên và thân thiện với người dùng.

**Áp dụng khi nào?**

* Đối tượng được kiểm thử là 1 thành phần phần mềm (có thể là 1 hàm chức năng, 1 module chức năng, 1 phân hệ chức năng…).
* Phương pháp Kiểm tra Hộp trắng áp dụng cho các mức độ kiểm tra phần mềm sau đây:
  + Unit Testing (Kiểm thử đơn vị): Để kiểm tra đường dẫn trong một đơn vị.
  + Integration Testing (Test tích hợp): Để kiểm tra đường dẫn giữa các đơn vị.
  + System Testing (Test hệ thống): Để kiểm tra các đường dẫn giữa các hệ thống con.

Tuy nhiên, nó là chủ yếu áp dụng cho các kiểm thử đơn vị .

**Áp dụng ở giai đoạn nào của dự án?**

Khi viết test case sẽ dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình). Khi thực hiện test sẽ thực thi test trong code (không cần thực thi chương trình, vì thực hiện test white box sẽ sử dụng framework nào đó hỗ trợ - như Junit, Nunit). *Kiểm thử hộp trắng có thể bắt đầu ngay sau khi một thành phần phần mềm được code xong.*

**Ai là người thực hiện?**

Do các lập trình viên, nhà phát triển phần mềm thực hiện.

**Các kỹ thuật, phương pháp được sử dụng**

* Statement testing (kiểm thử câu lệnh): đảm bảo mỗi dòng lệnh được thực thi ít nhất một lần, qua đó có thể test được toàn bộ cách vận hành của source code.
* Decision testing (kiểm thử quyết định): test những quyết định dựa trên decision result (kết quả quyết định). Để làm điều này, test case sẽ đi theo các control flow từ decision point (điểm quyết định). Decision testing giúp kiểm thử xem có câu lệnh không thể truy cập hay gây bất thường không.
* Condition testing (kiểm thử điều kiện): test các biểu thức Boolean có dạng True (đúng) hoặc False (sai). Mỗi biểu thức Boolean sẽ được thực thi ít nhất một lần bằng cả tham số True và False.
* Multiple condition testing (kiểm thử đa điều kiện): kiểm thử mọi tổ hợp điều kiện có thể của quyết định. Công thức tính số tổ hợp này là 2 lũy thừa bậc N, với N là số biến điều kiện. Số lượng tổ hợp này cũng chính là số lượng test case mà bạn phải dùng.
* Path testing (kiểm thử lộ trình): tester sẽ test từng câu lệnh có trong source code để tìm lỗi. Việc này giúp xác định lỗi tiềm ẩn trong một đoạn code.

**Các công cụ, chương trình được sử dụng**

* JBehave: <http://jbehave.org/>
* Cucumber: <https://cucumber.io/>
* Junit: <https://junit.org/>
* Cfix: <http://www.cfix-testing.org/>
* Parasoft Jtest: <https://www.parasoft.com>

**So sánh tổng hợp**

* Ưu điểm:
  + Buộc testers phải suy luận cẩn thận về việc triển khai kiểm thử
  + Phát hiện lỗi trong code 'ẩn'
  + Phát hiện ra Dead code hoặc các vấn đề khác liên quan đến các phương pháp lập trình tốt nhất.
* Nhược điểm:
  + Đắt tiền vì người ta phải dành cả thời gian và tiền bạc để thực hiện.
  + Có khả năng một vài dòng code bị vô tình bỏ qua.
  + Yêu cầu kiến thức tốt về ngôn ngữ lập trình.

Ref:

* [What is Structural Testing? Software Testing Class](https://www.softwaretestingclass.com/what-is-structural-testing/)
* [Structural Software Testing - GeeksforGeeks](https://www.geeksforgeeks.org/structural-software-testing/)
* [Structural Testing Tutorial - What Is Structural Testing (softwaretestinghelp.com)](https://www.softwaretestinghelp.com/structural-testing-tutorial/)