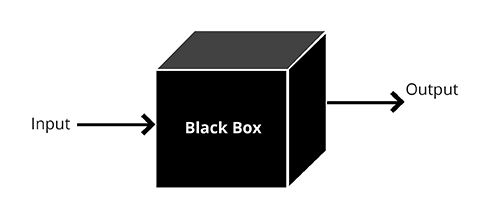
# PHÂN BIỆT BLACK BOX TEST VÀ WHITE BOX TEST, SƠ LƯỢC MỘT SỐ KỸ THUẬT TRONG BLACK BOX TEST

Trong kiểm thử phần mềm, có rất nhiều kỹ thuật kiểm thử được nhắc tới. Tuy nhiên ở bài viết này, tôi chỉ xin để cập đến 2 kỹ thuật đó là: Kỹ thuật kiểm thử hộp đen (Black Box test) và Kỹ thuật kiểm thử hộp trắng (White box test)

# 1. KHÁI NIỆM CỦA BLACK BOX TEST VÀ WHITE BOX TEST

## 1.1. BACK BOX TEST

**1.1.1. Định nghĩa** Kiểm tra hộp đen (Black box testing) là một phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó.



**1.1.2. Đối tượng được kiểm thử** Là thành phần phần mền (TPPM) có thể là 1 hàm chức năng, 1 modul chức năng, 1 phân hệ chức năng...

**1.1.3. Phương pháp thử nghiệm:** Dựa vào chức năng Kiểm thử hộp đen (Black box test) có thể được áp dụng hầu như đến mọi cấp độ của kiểm thử phần mềm:

* Kiểm thử đơn vị (Unit test)
* Kiểm thử tích hợp (Intergration test)
* Kiểm thử hệ thống (System test)
* Kiểm thử chấp nhận (Acceptance test).

Tuy nhiên, Black box test được sử dụng thích hợp nhất trong kiểm thử hệ thống (System test) và Kiểm thử chấp nhận (Acceptance test)

**1.1.4. Đặc điểm**

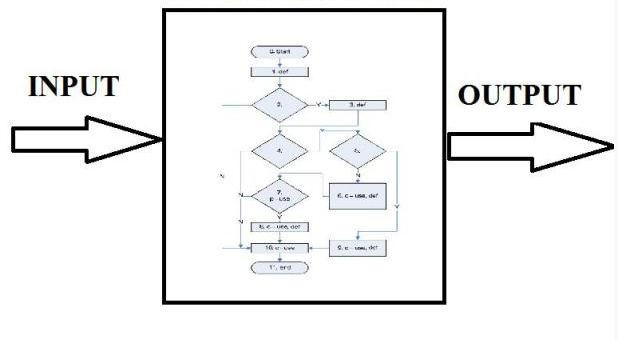
* Là chiến lược kiểm thử TPPM dựa vào thông tin duy nhất là các đặc tả về yêu cầu chức năng của TPPM tương ứng.
* Người kiểm thử không cần thiết phải có kiến thức về việc mã hoá, cấu trúc bên trong của TPPM, cũng như không yêu cầu phải biết lâp trình phần mềm.
* Việc kiểm thử được tiến hành dựa vào việc kiểm thử TPPM làm được gì, có phù hợp với yêu cầu của người dùng hay không. Các tester nhập số liệu vào phần mềm và chỉ cần xem kết quả của phần mềm và các mục tiêu kiểm tra.
* Mức test này thường yêu cầu các tester phải viết test case đầy đủ trước khi test; khi test, đơn giản chỉ cần thực hiện theo các bước mô tả trong test case thao tác và nhập data vào, sau đó xem kết quả trả về hoặc hành vi của phần mềm, rồi so sánh với kết quả mong đọi được viết trong testcase

**1.1.5. Tạo test case và Thực hiện test case**

* Khi viết test case: Dựa vào yêu cầu và giao diện bên ngoài của chương trình (Không can thiệp vào bên trong code của chương trình)
* Khi thực hiện test: Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code)

## 1.2. WHITE BOX TEST

**1.2.1. Định nghĩa** Kiểm thử hộp trắng (While box test) là phương pháp thử nghiệm phần mềm, trong đó các thiết kế, cấu trúc giải thuật bên trong, và việc thực hiện các công việc đều được biết đến



**1.2.2. Đối tượng kiểm thử** Là 1 thành phần của phần mềm (1 chức năng, 1 module chức năng, 1 phân hệ chức năng....)

**1.2.3. Phương pháp thử nghiệm:** Dựa vào thuật giải Kiểm thử hộp trắng dựa vào thuật giải cụ thể, vào cấu trúc dữ liệu bên trong của ₫ơn vị phần mềm cần kiểm thử ₫ể xác ₫ịnh ₫ơn vị phần mềm ₫ó có thực hiện ₫úng không.

* Với những TPPM quá lớn sẽ tốn rất nhiều thời gian và công sức để kiểm thử nếu như dùng kiểm thử tích hợp (Integration test) hay kiểm thử chức năng (Functional test)).
* Kỹ thuật white box test thích hợp dùng để kiểm thử đơn vị (Unit test)

**1.2.4. Đặc điểm**

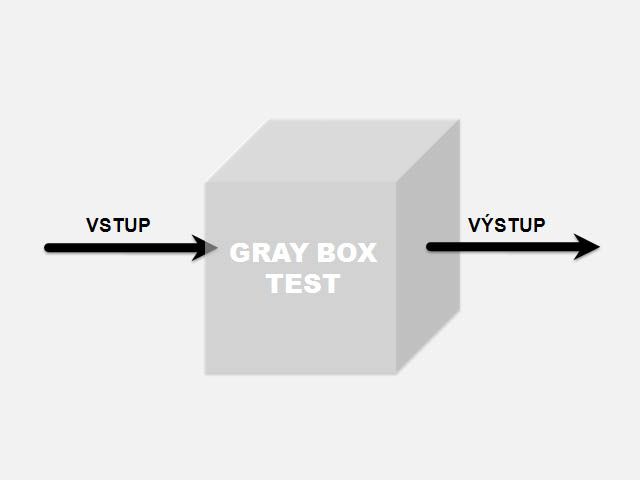
* Là chiến lược kiểm thử TPPM dựa vào giải thuật, cấu trúc bên trong chức năng của TPPM tương ứng.
* Người kiểm thử phải có kiến thức nhất định về việc mã hoá, cấu trúc bên trong của chức năng, biết lâp trình phần mềm.
* Việc kiểm thử được tiến hành dựa vào việc kiểm xem giải thuật, mã lệnh đã làm có đúng không.
* Mức test này thường yêu cầu các tester phải viết test case đầy đủ các nhánh trong code; khi test, sẽ set điều kiện và data để chạy vào đủ tất cả các nhánh trong giải thuật, đảm bảo thực hiện đầy đủ.

**1.2.5. Tạo testcase và thực hiện test**

* Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)
* Khi thực hiện test: 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 (Ví dụ như test kiểu debug)
* Trong kiểm tra này, đòi hỏi người tester phải có kiến thức và kỹ năng nhất định về ngôn ngữ lập trình được dùng, hiểu thuật giải trong thành phần phần mềm, để có thể hiểu được chi tiết về đoạn code cần kiểm thử .

## 1.3. GRAY BOX TEST

Ngoài 2 kỹ thuật đã được nhắc đến: Black box test và white box test, thì có 1 kỹ thuật, Gray box test là sự kết hợp giữa black box test và white box test. **1.3.1. Định nghĩa** Gray Box Testing là một phương pháp kiểm thử phần mềm được kết hợp giữa Phương pháp Kiểm thử Black Box (hộp đen) và White Box (hộp trắng). Trong Kiểm thử Hộp xám, cấu trúc bên trong sản phẩm chỉ được biết một phần



**1.3.2. Phương pháp thử nghiệm:** Dựa vào giải thuật và chức năng

Gray box test có thể được sử dụng ở nhiều mức kiểm thử khác nhau. Tuy nhiên, chủ yếu được ứng dụng trong Kiểm thử tích hợp (Intergration test)

**1.3.3. Tạo testcase và thực hiện test**

* Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)
* Khi thực hiện test: Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code)

# 2. ƯU ĐIỂM, NHƯỢC ĐIỂM CỦA BLACK BOX TEST VÀ WHITE BOX TEST

| **Nội dung** | **Black box test** | **White box test** |
| --- | --- | --- |
| 1. Ưu điểm | - Thích hợp trong việc kiểm tra từng phân đoạn lớn các mã lệnh, chức năng lớn  - Người thử nghiệm không cần hiểu biết về mã lệnh được viết trong chương trình - Tách biệt giữa quan điểm của người sử dụng và người phát triển phần mềm | - Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh  - Biết được yêu cầu bên trong của phần mềm, kiểm tra sẽ sát hơn  - Cho phép tìm kiếm các lỗi ẩn bên trong  - Các lập trình viên có thể tự kiểm tra  - Giúp tối ưu việc mã hoá  - Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất |
| 2. Nhược điểm | - Độ bao phủ hạn chế vì chỉ có một phần nhỏ trong số các kịch bản thử nghiệm được thực hiện  - Kiểm tra không hiệu quả do người thử nghiệm không hiểu biết gì về cấu trúc bên trong phần mềm.  - Tester có hạn chế về hiểu biết về ứng dụng | - Không thể tìm thấy tính năng chưa thực hiện hoặc bỏ sót  - Đòi hỏi hiểu sâu về cấu trúc bên trong của phần mềm được thử nghiệm  - Yêu cầu truy xuất mã lệnh bên trong chương trình |

# 3. SỰ KHÁC NHAU GIỮA BLACK BOX TEST VÀ WHITE BOX TEST

| **Tiêu chuẩn** | **Black box test** | **White box test** |
| --- | --- | --- |
| 1. Định nghĩa | - Kiểm tra hộp đen là phương pháp thử nghiệm phần mềm được sử dụng để kiểm tra các phần mềm mà không quan tâm đến cấu trúc bên trong của chương trình. | - Kiểm tra hộp trắng là phương pháp kiểm thử phần mềm, sử dụng để kiểm tra phần mềm mà yêu cầu phải biết cấu trúc bên trong của chương trình. |
| 2. Trách nhiệm | - Thử nghiệm được thực hiện bên ngoài, không liên quan đến nhà phát triển phần mềm. | - Thông thường, các thử nghiệm được thực hiện bởi nhà phát triển phần mềm. |
| 3. Cấp độ test sử dụng | - Thử nghiệm áp dụng ở cấp độ cao như: kiểm tra hệ thống (System test), kiểm tra chấp nhận (Acceptance test) | - Thử nghiệm được áp dụng ở mức độ thấp hơn như thử nghiệm đơn vị (Unit Test), thử nghiệm hội nhập (Integration test) |
| 4. Biết lập trình | - Không yêu cầu hiểu biết về Lập trình | - Yêu cầu hiểu biết nhất định về lập trình. |
| 5. Biết việc thực hiện chương trình | - Không yêu cầu hiểu về cấu trúc bên trong chức năng, và không cẩn hiểu làm thế nào để có được chức năng đó | - Yêu cầu hiểu cấu trúc bên trong chức năng được thực hiện như nào. |
| 6. Cơ sở tạo Test Cases | - Kiểm tra hộp đen được bắt đầu dựa trên tài liệu yêu cầu kỹ thuật | - Kiểm tra hộp trắng được bắt đầu dựa trên các tài liệu thiết kế chi tiết |

# 4. SƠ LƯỢC VỀ CÁC KỸ THUẬT KIỂM TRA HỘP ĐEN

## 4.1. Quy trình kiểm thử hộp đen

Với đặc điểm của Kiểm thử hộp đen là chỉ dựa vào chức năng của phần mềm, do đó quy trình kiểm thử qua các bước chính như sau:

* Phân tích đặc tả về các yêu cầu chức năng mà TPPM cần thực hiện
* Dùng 1 kỹ thuật đinh nghĩa các testcase xác định để định nghĩa các testcase, gồm 3 thông tin sau: + Giá trị dữ liệu để TPPM xử lý (hoặc hợp lệ, hoặc không hợp lệ) + Trạng thái của TPPM cần có để thực hiện testcase + Giá trị dữ liệu xuất mà TPPM phải tạo được
* Kiểm thử các testcase đã định nghĩa
* So sánh kết quả thu được với kết quả kỳ vọng trong từ testcase, từ đó lập báo cáo về kết quả kiểm thử

## 4.2. Các kỹ thuật phổ biến trong kiểm thử hộp đen

* Kỹ thuật phân lớp tương ₫ương (Equivalence Class Partitioning).
* Kỹ thuật phân tích các giá trị biên (Boundary value analysis).
* Kỹ thuật dùng các bảng quyết ₫ịnh (Decision Tables)
* Kỹ thuật kiểm thử các bộ n thần kỳ (Pairwise)
* Kỹ thuật dùng bảng chuyển trạng thái (State Transition)
* Kỹ thật phân tích vùng miền (domain analysis)
* Kỹ thuật dựa trên ₫ặc tả Use Case (Use case)
* Kỹ thuật dùng lược ₫ồ quan hệ nhân quả (Cause-Effect Diagram) Trong bài viết này tôi chỉ nêu sơ lược về một số kỹ thuật kiểm thử trên. Chi tiết sẽ được cụ thể trong các bài viết sau

### 4.2.1. Kỹ thuật phân lớp tương ₫ương

Ý tưởng của kỹ thuật này là cố gắng phân các testcase ra thành nhiều nhóm khác nhau : các testcase trong mỗi nhóm xác định TPPM thực hiện cùng 1 hành vi. Mỗi nhóm testcase thỏa mãn tiêu chuẩn trên ₫ược gọi là 1 lớp tương ₫ương, ta chỉ cần xác ₫ịnh 1 testcase ₫ại diện cho nhóm và dùng testcase này ₫ể kiểm thử TPPM. Như vậy ta ₫ã giảm rất nhiều testcase cần ₫ịnh nghĩa và kiểm thử, nhưng chất lượng kiểm thử không bị giảm sút bao nhiêu so với vét cạn. Điều này là dựa vào kỳ vọng: - Nếu 1 testcase trong lớp tương ₫ương nào ₫ó gây lỗi TPPM thì các testcase trong lớp này cũng sẽ gây lỗi như vậy.

* Nếu 1 testcase trong lớp tương ₫ương nào ₫ó không gây lỗi TPPM thì các testcase trong lớp này cũng sẽ không gây lỗi.
* Với các giá trị không hợp lệ: Ta nên tạo 1 lớp tương đương đại diện các testcase chứa các giá trị không hợp lệ theo đặc tả để xem TPPM phản ứng như nào với những trường hợp này

### 4.2.2. Kỹ thuật phân tích các giá trị ở biên

Khi tạo testcase, ta chỉ dùng Kỹ thuật phân lớp tương đương thì hẳn là chưa đủ. Kinh nghiệm cho thấy rằng lỗi thường nằm ở biên (₫ầu hay cuối) của 1 khoảng liên tục nào ₫ó (lớp tương ₫ương). Do ₫ó với Kỹ thuật phân tích giá trị biên tập trung tạo các testcase ứng với những giá trị ở biên này. Nên thông thường là có sự kết hợp cả 2 kỹ thuật: Phân lớp tương đương và Phân tích giá trị biên để viết các testcase. Ý tưởng của kỹ thuật là chỉ ₫ịnh nghĩa các testcase ứng với các giá trị ngay trên biên hay lân cận biên của từng lớp tương ₫ương. Do ₫ó kỹ thuật này chỉ thích hợp với các lớp tương ₫ương xác ₫ịnh bởi các giá trị liên tục (số nguyên, số thực), chứ nó không thích hợp với lớp tương ₫ương ₫ược xác ₫ịnh bởi các giá trị liệt kê mà không có mối quan hệ lẫn nhau. Quy trình cụ thể ₫ể thực hiện kiểm thử dựa trên các giá trị ở biên: - Nhận dạng các lớp tương ₫ương dựa trên ₫ặc tả về yêu cầu chức năng của TPPM.

* Nhận dạng 2 biên của mỗi lớp tương ₫ương. Tạo các testcase cho mỗi biên của mỗi lớp tương ₫ương :
* 1 testcase tại giá trị biên.
* 1 testcase ngay dưới biên.
* 1 testcase ngay trên biên. Ý nghĩa ngay trên và ngay dưới biên phụ thuộc vào ₫ơn vị ₫o lường cụ thể : Số nguyên , số thập phân...

### 4.2.3. Kỹ thuật dùng bảng quyết ₫ịnh (decision table)

Bảng quyết ₫ịnh là 1 công cụ rất hữu ích ₫ể ₫ặc tả các yêu cầu phần mềm hoặc ₫ể ₫ặc tả bảng thiết kế hệ thống phần mềm. Nó miêu tả các qui tắc nghiệp vụ phức tạp mà phần mềm phải thực hiện dưới dạng dễ ₫ọc và dễ kiểm soát :

|  | **Rule 1** | **Rule 2** | **...** | **Rule p** |
| --- | --- | --- | --- | --- |
| **Conditions** |  |  |  |  |
| Conditions-1 |  |  |  |  |
| ... |  |  |  |  |
| Conditions-m |  |  |  |  |
| **Actions** |  |  |  |  |
| Actions-1 |  |  |  |  |
| ... |  |  |  |  |
| Actions-n |  |  |  |  |

Trong đó:

* Condition-1 tới Condition-m miêu tả m ₫iều kiện dữ liệu nhập khác nhau có thể có.
* Action-1 tới Action-n miêu tả n hoạt ₫ộng khác nhau mà hệ thống có thể thực hiện phụ thuộc vào tổ hợp ₫iều kiện dữ liệu nhập nào.
* Mỗi cột miêu tả 1 luật cụ thể : tổ hợp ₫iều kiện nhập cụ thể và các hoạt ₫ộng cụ thể cần thực hiện. Lưu ý các hoạt ₫ộng cần thực hiện không phụ thuộc vào thứ tự các ₫iều kiện nhập, nó chỉ phụ thuộc vào giá trị các ₫iều kiện nhập. Tương tự, các hoạt ₫ộng cần thực hiện không phụ thuộc vào trạng thái hiện hành của TPPM, chúng cũng không phụ thuộc vào các ₫iều kiện nhập ₫ã có trước ₫ó.