# Chương 1: TỔNG QUAN KIỂM THỬ PHẦN MỀM

## 1.Các khái niệm cở bản về kiểm thử phần mềm

### 1.1 Kiểm thử phần mềm là gì?

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó. (Theo Bảng chú giải thuật ngữ chuẩn IEEE của Thuật ngữ kỹ nghệ phần mềm- IEEE Standard Glossary of Software Engineering Terminology).

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm lỗi. (Theo “The Art of Software Testing” – Nghệ thuật kiểm thử phần mềm).

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau. (Theo Bách khoa toàn thư mở Wikipedia).

Có thể định nghĩa một cách dễ hiểu như sau: Kiểm thử phần mềm là một tiến trình hay một tập hợp các tiến trình được thiết kế để đảm bảo mã hóa máy tính thực hiện theo cái mà chúng đã được thiết kế để làm, và không thực hiện bất cứ thứ gì không mong muốn. Đây là một pha quan trọng trong quá trình phát triển hệ thống, giúp cho người xây dựng hệ thống và khách hàng thấy được hệ thống mới đã đáp ứng yêu cầu đặt ra hay chưa?

## 1.2 Các phương pháp kiểm thử

Có 2 phương pháp kiểm thử chính là: Kiểm thử tĩnh và Kiểm thử động.

### 1.2.1 Kiểm thử tĩnh – Static testing

Là phương pháp thử phần mềm đòi hỏi phải duyệt lại các yêu cầu và các đặc tả bằng tay, thông qua việc sử dụng giấy, bút để kiểm tra logic, lần từng chi tiết mà không cần chạy chương trình. Kiểu kiểm thử này thường được sử dụng bởi chuyên viên thiết kế người mà viết mã lệnh một mình.

Kiểm thử tĩnh cũng có thể được tự động hóa. Nó sẽ thực hiện kiểm tra toàn bộ bao gồm các chương trình được phân tích bởi một trình thông dịch hoặc biên dịch mà xác nhận tính hợp lệ về cú pháp của chương trình.

### 1.2.2 Kiểm thử động – Dynamic testing

Là phương pháp thử phần mềm thông qua việc dùng máy chạy chương trình để điều tra trạng thái tác động của chương trình. Đó là kiểm thử dựa trên các ca kiểm thử xác định bằng sự thực hiện của đối tượng kiểm thử hay chạy các chương trình. Kiểm thử động kiểm tra cách thức hoạt động động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian. Trong kiểm thử động, phần mềm phải thực sự được biên dịch và chạy.

Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem liệu đầu ra có như mong muốn hay không. Các phương pháp kiểm thử động gồm có kiểm thử Unit – Unit Tests, Kiểm thử tích hợp – Intergration Tests, Kiểm thử hệ thống – System Tests, và Kiểm thử chấp nhận sản phẩm – Acceptance Tests.

## 1.3 Các chiến lược kiểm thử

Ba trong số những chiến lược kiểm thử thông dụng nhất bao gồm: Kiểm thử hộp đen, Kiểm thử hộp trắng, và Kiểm thử hộp xám.

### 1.3.1 Kiểm thử hộp đen – Black box testing

Một trong những chiến lược kiểm thử quan trọng là kiểm thử hộp đen, hướng dữ liệu, hay hướng vào/ra. Kiểm thử hộp đen xem chương trình như là một “hộp đen”. Mục đích của bạn là hoàn toàn không quan tâm về cách cư xử và cấu trúc bên trong của chương trình. Thay vào đó, tập trung vào tìm các trường hợp mà chương trình không thực hiện theo các đặc tả của nó. Theo hướng tiếp cận này, dữ liệu kiểm tra được lấy chỉ từ các đặc tả.

Các phương pháp kiểm thử hộp đen:

* Phân lớp tương đương – Equivalence partitioning.
* Phân tích giá trị biên – Boundary value analysis.
* Kiểm thử mọi cặp – All-pairs testing.
* Kiểm thử fuzz – Fuzz testing.
* Kiểm thử dựa trên mô hình – Model-based testing.
* Ma trận dấu vết – Traceability matrix.
* Kiểm thử thăm dò – Exploratory testing.
* Kiểm thử dựa trên đặc tả – Specification-base testing.

Kiểm thử dựa trên đặc tả tập trung vào kiểm tra tính thiết thực của phần mềm theo những yêu cầu thích hợp. Do đó, kiểm thử viên nhập dữ liệu vào, và chỉ thấy dữ liệu ra từ đối tượng kiểm thử. Mức kiểm thử này thường yêu cầu các ca kiểm thử triệt để được cung cấp cho kiểm thử viên mà khi đó có thể xác minh là đối với dữ liệu đầu vào đã cho, giá trị đầu ra (hay cách thức hoạt động) có giống với giá trị mong muốn đã được xác định trong ca kiểm thử đó hay không. Kiểm thử dựa trên đặc tả là cần thiết, nhưng không đủ để để ngăn chặn những rủi ro chắc chắn.

Ưu, nhược điểm:

Kiểm thử hộp đen không có mối liên quan nào tới mã lệnh, và kiểm thử viên chỉ rất đơn giản tâm niệm là: một mã lệnh phải có lỗi. Sử dụng nguyên tắc “ Hãy đòi hỏi và bạn sẽ được nhận”, những kiểm thử viên hộp đen tìm ra lỗi mà những lập trình viên đã không tìm ra. Nhưng, mặt khác, người ta cũng nói kiểm thử hộp đen “giống như là đi trong bóng tối mà không có đèn vậy”, bởi vì kiểm thử viên không biết các phần mềm được kiểm tra thực sự được xây dựng như thế nào. Đó là lý do mà có nhiều trường hợp mà một kiểm thử viên hộp đen viết rất nhiều ca kiểm thử để kiểm tra một thứ gì đó mà đáng lẽ có thể chỉ cần kiểm tra bằng 1 ca kiểm thử duy nhất, và/hoặc một số phần của chương trình không được kiểm tra chút nào.

Do vậy, kiểm thử hộp đen có ưu điểm của “một sự đánh giá khách quan”, mặt khác nó lại có nhược điểm của “thăm dò mù”.

### 1.3.2 Kiểm thử hộp trắng – White box testing

Là một chiến lược kiểm thử khác, trái ngược hoàn toàn với kiểm thử hộp đen, kiểm thử hộp trắng hay kiểm thử hướng logic cho phép bạn khảo sát cấu trúc bên trong của chương trình. Chiến lược này xuất phát từ dữ liệu kiểm thử bằng sự kiểm thử tính logic của chương trình. Kiểm thử viên sẽ truy cập vào cấu trúc dữ liệu và giải thuật bên trong chương trình (và cả mã lệnh thực hiện chúng).

Các phương pháp kiểm thử hộp trắng

* Kiểm thử giao diện lập trình ứng dụng - API testing (application programming interface): là phương pháp kiểm thử của ứng dụng sử dụng các API công khai và riêng tư.
* Bao phủ mã lệnh – Code coverage: tạo các kiểm tra để đáp ứng một số tiêu chuẩn về bao phủ mã lệnh.
* Các phương pháp gán lỗi – Fault injection.
* Các phương pháp kiểm thử hoán chuyển – Mutation testing methods.
* Kiểm thử tĩnh – Static testing: kiểm thử hộp trắng bao gồm mọi kiểm thử tĩnh.

Phương pháp kiểm thử hộp trắng cũng có thể được sử dụng để đánh giá sự hoàn thành của một bộ kiểm thử mà được tạo cùng với các phương pháp kiểm thử hộp đen. Điều này cho phép các nhóm phần mềm khảo sát các phần của 1 hệ thống ít khi được kiểm tra và đảm bảo rằng những điểm chức năng quan trọng nhất đã được kiểm tra.

### 1.3.3 Kiểm thử hộp xám – Gray box testing

Kiểm thử hộp xám đòi hỏi phải có sự truy cập tới cấu trúc dữ liệu và giải thuật bên trong cho những mục đích thiết kế các ca kiểm thử, nhưng là kiểm thử ở mức người sử dụng hay mức hộp đen. Việc thao tác tới dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, giống như một chiếc “hộp xám”, bởi vì đầu vào và đầu ra rõ ràng là ở bên ngoài “hộp đen” mà chúng ta vẫn gọi về hệ thống được kiểm tra. Sự khác biệt này đặc biệt quan trọng khi quản lý kiểm thử tích hợp – Intergartion testing giữa 2 modun mã lệnh được viết bởi hai chuyên viên thiết kế khác nhau, trong đó chỉ giao diện là được đưa ra để kiểm thử.

Kiểm thử hộp xám có thể cũng bao gồm cả thiết kế đối chiếu để quyết định, ví dụ, giá trị biên hay thông báo lỗi.

## 1.4 Nguyên tắc kiểm thử phần mềm

Để kiểm thử đạt hiệu quả thì khi tiến hành kiểm thử phần mềm cần phải tuân thủ một số quy tắc sau:

* Quy tắc 1: Một phần quan trọng của 1 ca kiểm thử là định nghĩa của đầu ra hay kết quả mong muốn.
* Quy tắc 2: Lập trình viên nên tránh tự kiểm tra chương trình của mình.
* Quy tắc 3: Nhóm lập trình không nên kiểm thử chương trình của chính họ.
* Quy tắc 4: Kiểm tra thấu đáo mọi kết quả của mỗi kiểm tra.
* Quy tắc 5: Các ca kiểm thử phải được viết cho các trạng thái đầu vào không hợp lệ và không mong muốn, cũng như cho các đầu vào hợp lệ và mong muốn.
* Quy tắc 6: Khảo sát 1 chương trình để xem liệu chương trình có thực hiện cái mà nó cần thực hiện chỉ là 1 phần, phần còn lại là xem liệu chương trình có thực hiện cái mà nó không cần phải thực hiện hay không.
* Quy tắc 7: Tránh các ca kiểm thử bâng quơ trừ khi chương trình thực sự là 1 chương trình bâng quơ.
* Quy tắc 8: Không dự kiến kết quả của kiểm thử theo giả thiết ngầm là không tìm thấy lỗi.
* Quy tắc 9: Xác suất tồn tại lỗi trong 1 đoạn chương trình là tương ứng với số lỗi đã tìm thấy trong đoạn đó.
* Quy tắc 10: Kiểm thử là 1 nhiệm vụ cực kỳ sáng tạo và có tính thử thách trí tuệ.

# Chương 2: THIẾT KẾ TEST-CASE

## 2.1 Khái niệm

Thiết kế test – case trong kiểm thử phần mềm là quá trình xây dựng các phương pháp kiểm thử có thể phát hiện lỗi, sai sót, khuyết điểm của phần mềm để xây dựng phần mềm đạt tiêu chuẩn.

## 2.2 Vai trò của thiết kế test-case

* Tạo ra các ca kiểm thử tốt nhất có khả năng phát hiện ra lỗi, sai sót của phần mềm một cách nhiều nhất.
* Tạo ra các ca kiểm thử có chi phí rẻ nhất, đồng thời tốn ít thời gian và công sức nhất.

## 2.3 Quy trình thiết kế test-case

Một trong những lý do quan trọng nhất trong kiểm thử chương trình là thiết kế và tạo ra các ca kiểm thử - các Test case có hiệu quả. Với những ràng buộc về thời gian và chi phí đã cho, thì vấn đề then chốt của kiểm thử trở thành: Tập con nào của tất cả ca kiểm thử có thể có khả năng tìm ra nhiều lỗi nhất?

Thông thường, phương pháp kém hiệu quả nhất là kiểm tra tất cả đầu vào ngẫu nhiên – quá trình kiểm thử một chương trình bằng việc chọn ngẫu nhiên một tập con các giá trị đầu vào có thể. Về mặt khả năng tìm ra nhiều lỗi nhất, tập hợp các ca kiểm thử được chọn ngẫu nhiên có rất ít cơ hội là tập hợp tối ưu hay gần tối ưu.

Sau đây là một số phương pháp để chọn ra một tập dữ liệu kiểm thử một cách thông minh. Để kiểm thử hộp đen và kiểm thử hộp trắng một cách thấu đáo là không thể. Do đó, một chiến lược kiểm thử hợp lý là chiến lược có thể kết hợp sức mạnh của cả hai phương pháp trên:

Phát triển 1 cuộc kiểm thử nghiêm ngặt vừa bằng việc sử dụng các phương pháp thiết kế ca kiểm thử hướng hộp đen nào đó và sau đó bổ sung thêm những ca kiểm thử này bằng việc khảo sát tính logic của chương trình, sử dụng phương pháp hộp trắng. Những chiến lược kết hợp đó bao gồm:

|  |  |
| --- | --- |
| Hộp đen | Hộp trắng |
| Phân lớp tương đương | Bao phủ câu lệnh |
| Phân tích giá trị biên | Bao phủ quyết định |
| Đồ thị nguyên nhân – kết quả | Bao phủ điều kiện |
| Đoán lỗi | Bao phủ điều kiện – quyết định |
|  | Bao phủ đa điều kiện |

Mỗi phương pháp có những ưu điểm cũng như khuyết điểm riêng, do đó để có được tập các ca kiểm thử tối ưu, chúng ta cần kết hợp hầu hết các phương pháp. Quy trình thiết kế các ca kiểm thử sẽ bắt đầu bằng việc phát triển các ca kiểm thử sử dụng phương pháp hộp đen và sau đó phát triển bổ sung các ca kiểm thử cần thiết với phương pháp hộp trắng.

### 2.3.1 Kiểm thử hộp trắng – Kiểm thử phủ logic

Kiểm thử hộp trắng có liên quan tới mức độ mà các ca kiểm thử thực hiện hay bao phủ tính logic (mã nguồn) của chương trình. Kiểm thử hộp trắng cơ bản là việc thực hiện mọi đường đi trong chương trình, nhưng việc kiểm thử đầy đủ đường đi là một mục đích không thực tế cho một chương trình với các vòng lặp. Các tiêu chuẩn trong kiểm thử bao phủ logic gồm có:

* Bao phủ câu lệnh – Statement Coverage

Tư tưởng: Thực hiện mọi câu lệnh trong chương trình ít nhất 1 lần.

* Bao phủ điều kiện – Condition Coverage

Tư tưởng: Viết đủ các ca kiểm thử để đảm bảo rằng mỗi điều kiện trong một quyết định đảm nhận tất cả các kết quả có thể ít nhất một lần.

* Bao phủ quyết định/ điều kiện – Decision/condition coverage

Tư tưởng: Thực hiện đủ các ca kiểm thử mà mỗi điều kiện trong 1 quyết định thực hiện trên tất cả các kết quả có thể ít nhất 1 lần, và mỗi điểm vào được gọi ít nhất 1 lần.

* Bao phủ đa điều kiện – Multiple condition coverage

Tư tưởng: Viết đủ các ca kiểm thử mà tất cả những sự kết hợp của các kết quả điều kiện có thể trong mỗi quyết định, và tất cả các điểm vào phải được gọi ít nhất 1 lần.

### 2.3.2 Kiểm thử hộp đen

#### 2.3.2.1 Phân lớp tương đương – Equivalence Patitioning

Phân lớp tương đương là một phương pháp kiểm thử hộp đen chia miền đầu vào của một chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các ca kiểm thử. Phương pháp này cố gắng xác định ra một ca kiểm thử mà làm lộ ra một lớp lỗi, do đó làm giảm tổng số các trường hợp kiểm thử phải được xây dựng. Thiết kế ca kiểm thử cho phân lớp tương đương dựa trên sự đánh giá về các lớp tương đương với một điều kiện vào. Lớp tương đương biểu thị cho tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào. Một cách xác định tập con này là để nhận ra rằng 1 ca kiểm thử được lựa chọn tốt cũng nên có 2 đặc tính khác:

* Giảm thiểu số lượng các ca kiểm thử khác mà phải được phát triển để hoàn thành mục tiêu đã định của kiểm thử “hợp lý”.
* Bao phủ một tập rất lớn các ca kiểm thử có thể khác. Tức là, nó nói cho chúng ta một thứ gì đó về sự có mặt hay vắng mặt của những lỗi qua tập giá trị đầu vào cụ thể.

Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

(1). Xác định các lớp tương đương

(2). Xác định các ca kiểm thử.

**Xác định các lớp tương đương**

Các lớp tương đương được xác định bằng bằng cách lấy mỗi trạng thái đầu vào (thường là 1 câu hay 1 cụm từ trong đặc tả) và phân chia nó thành 2 hay nhiều nhóm.

|  |  |  |
| --- | --- | --- |
| Điều kiện bên ngoài | Các lớp tương đương hợp lệ | Các lớp tương đương không hợp lệ |
|  |  |  |

Chú ý là hai kiểu lớp tương đương được xác định: lớp tương đương hợp lệ mô tả các đầu vào hợp lệ của chương trình, và lớp tương đương không hợp lệ mô tả tất cả các trạng thái có thể khác của điều kiện (ví dụ, các giá trị đầu vào không đúng). Với 1 đầu vào hay điều kiện bên ngoài đã cho, việc xác định các lớp tương đương hầu như là 1 quy trình mang tính kinh nghiệm. Để xác định các lớp tương đương c có thể áp dụng tập các nguyên tắc dưới đây:

* Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ.
* Nếu 1 trạng thái đầu vào xác định số giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương bất hợp lệ.
* Nếu 1 trạng thái đầu vào chỉ định tập các giá trị đầu vào và chương trình sử dụng mỗi giá trị là khác nhau, xác định 1 lớp tương đương hợp lệ cho mỗi loại và 1 lớp tương đương không hợp lệ.
* Nếu 1 trạng thái đầu vào chỉ định một tình huống “chắc chắn – must be”, xác định 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ.

Nếu có bất kỳ lý do nào để tin rằng chương trình không xử lý các phần tử trong cùng 1 lớp là như nhau, thì hãy chia lớp tương đương đó thành các lớp tương đương nhỏ hơn.

Xác định các ca kiểm thử

Với các lớp tương đương xác định được ở bước trên, bước thứ hai là sử dụng các lớp tương đương đó để xác định các ca kiểm thử. Quá trình này như sau:

* Gán 1 số duy nhất cho mỗi lớp tương đương.
* Cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi (hợp nhất thành) các ca kiểm thử, viết 1 ca kiểm thử mới bao phủ càng nhiều các lớp tương đương đó chưa được bao phủ càng tốt.
* Cho đến khi các ca kiểm thử của bạn đã bao phủ tất cả các lớp tương đương không hợp lệ, viết 1 ca kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ.
* Lý do mà mỗi ca kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm tra đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm tra đầu vào không đúng khác.

Mặc dù việc phân lớp tương đương là rất tốt khi lựa chọn ngẫu nhiên các ca kiểm thử, nhưng nó vẫn có những thiếu sót. Ví dụ, nó bỏ qua các kiểu test – case có lợi nào đó. Hai phương pháp tiếp theo, phân tích giá trị biên và đồ thị nguyên nhân – kết quả , bao phủ được nhiều những thiếu sót này.

#### 2.3.2.2 Phân tích giá trị biên

Kinh nghiệm cho thấy các ca kiểm thử mà khảo sát tỷ mỷ các điều kiện biên có tỷ lệ phần trăm cao hơn các ca kiểm thử khác. Các điều kiện biên là những điều kiện mà các tình huống ngay tại, trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp tương đương đầu ra. Phân tích các giá trị biên là phương pháp thiết kế ca kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở 2 khía cạnh:

* Phân tích giá trị biên không lựa chọn phần tử bất kỳ nào trong 1 lớp tương đương là điển hình, mà nó yêu cầu là 1 hay nhiều phần tử được lựa chọn như vậy mà mỗi cạnh của lớp tương đương đó chính là đối tượng kiểm tra.
* Ngoài việc chỉ tập trung chú ý vào các trạng thái đầu vào (không gian đầu vào), các ca kiểm thử cũng nhận được bằng việc xem xét không gian kết quả (các lớp tương đương đầu ra).

Phân tích giá trị biên yêu cầu óc sáng tạo và lượng chuyên môn hóa nhất định và nó là một quá trình mang tính kinh nghiệm rất cao. Tuy nhiên, có một số quy tắc chung như sau:

* Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, hãy viết các ca kiểm thử cho các giá trị cuối của giới hạn, và các ca kiểm thử đầu vào không hợp lệ cho các trường hợp vừa ra ngoài phạm vi.
* Nếu 1 trạng thái đầu vào định rõ số lượng giá trị, hãy viết các ca kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và một giá trị trên, một giá trị dưới những giá trị này.
* Sử dụng quy tắc 1 cho mỗi trạng thái đầu vào. Ví dụ, nếu 1 chương trình tính toán sự khấu trừ FICA hàng tháng và nếu mức tối thiểu là 0.00$, và tối đa là 1,165.25$, hãy viết các ca kiểm thử mà khấu trừ 0.00$ và 1,165.25, khấu trừ âm và khấu trừ lớn hơn 1,165.25$. Chú ý là việc xem xét giới hạn của không gian kết quả là quan trọng vì không phải lúc nào các biên của miền đầu vào cũng mô tả cùng một tập sự kiện như biên của giới hạn đầu ra (ví dụ, xét chương trình con tính SIN). Ngoài ra, không phải lúc nào cũng có thể tạo ra 1 kết quả bên ngoài giới hạn đầu ra, nhưng tuy nhiên rất đáng để xem xét tiềm ẩn đó.
* Sử dụng nguyên tắc 2 cho mỗi trạng thái đầu ra.
* Nếu đầu vào hay đầu ra của 1 chương trình là tập được sắp thứ tự ( ví dụ,1 file tuần tự hay 1 danh sách định tuyến hay 1 bảng) tập trung chú ý vào các phần tử đầu tiên và cuối cùng của tập hợp.
* Sử dụng sự khéo léo của bạn để tìm các điều kiện biên.

# Chương 3: HIỆN THỰC

## 3.1. Mô tả yêu cầu dự án

### 3.1.1. Chức năng chính

* Đăng nhập.
* Thêm khóa học.
* Đánh giá khóa học.

### 3.1.2. Mô tả chức năng

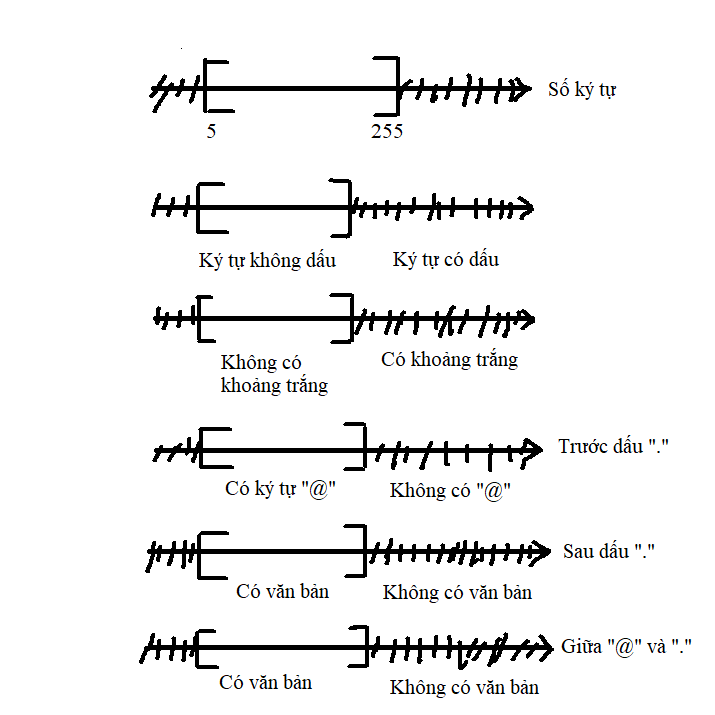
* Đăng nhập: Khi người dùng đăng nhập phải nhập email và password đã đăng ký trước đó.
* Email: không được để trống hoặc chỉ có khoảng trắng. Độ dài email phải từ 5 đến 255 ký tự, không được nhập ký tự có dấu, không được có khoảng trắng trong mail, phải có 1 ký tự “@” ở đâu đó trong chuỗi nằm trước dấu “.”, phải có văn bản sau “@” nhưng trước dấu “.”, phải có dấu “.” và chữ sau dấu “.”.
* Password: Mật khẩu không được chứa bất kỳ khoảng trống nào. Mật khẩu phải chứa ít nhất một chữ số (0-9). Độ dài mật khẩu phải từ 8 đến 15 ký tự. Mật khẩu phải chứa ít nhất một chữ cái viết thường (az). Mật khẩu phải chứa ít nhất một chữ cái viết hoa (AZ). Mật khẩu phải chứa ít nhất một ký tự đặc biệt ( @, #, %, &, !, $, v.v…). Mật khẩu không được viết có dấu.
* Thêm khóa học: người dùng nhập tên khóa học và hình ảnh khóa học.
* Tên khóa học: không được để trống hoặc chỉ có khoảng trắng**,** độ dài ký tự phải từ 1 đến 255 ký tự.
* Hình ảnh khóa học: không được để trống hoặc chỉ có khoảng trắng.
* Đánh giá khóa học: người dùng nhập số sao, nội dung.
* Số sao: không được để trống hoặc chỉ có khoảng trắng, phải là số nguyên từ 1 đến 5.
* Nội dung: không được để trống hoặc chỉ có khoảng trắng, số ký tự nhập từ 1 đến 255.

## 3.2. Thiết kế Test-case cho dự án

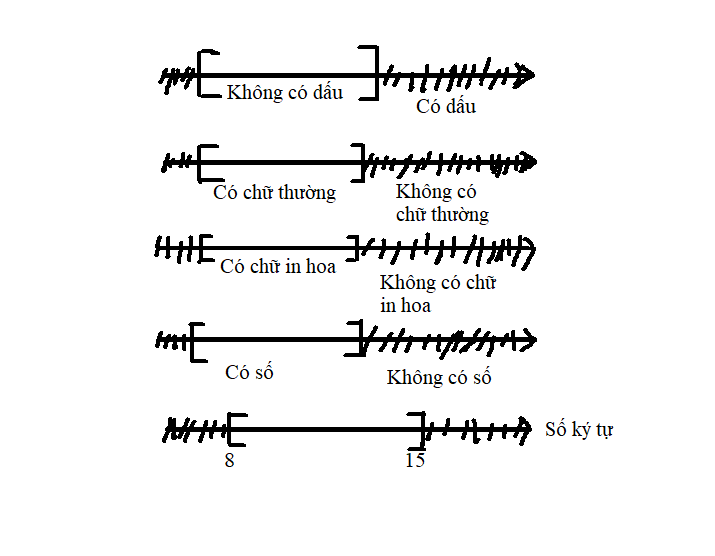
### 3.2.1. Kiểm thử hộp đen

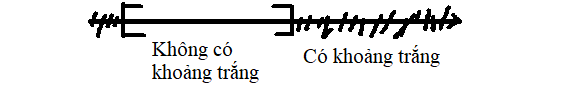
#### 3.2.1.1. Phân vùng tương đương

* Đăng nhập:
* Email:

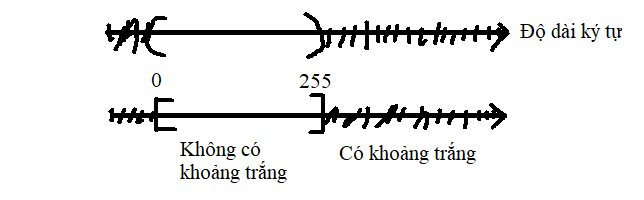


* Password:

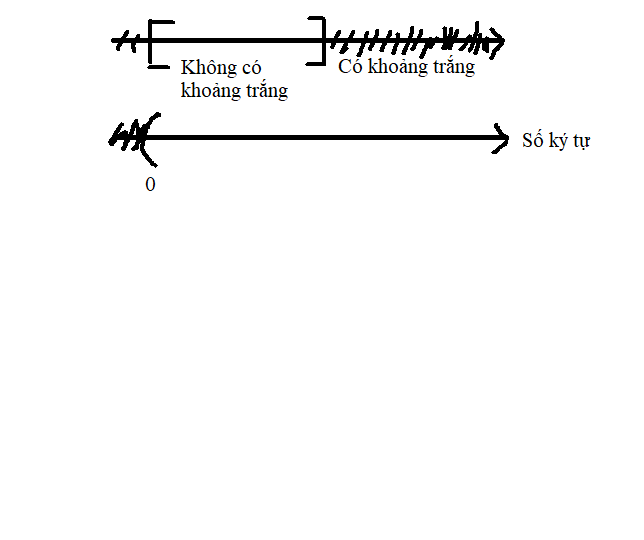




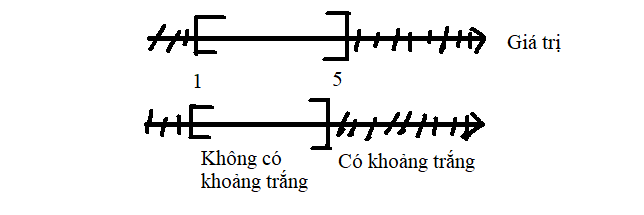
* Thêm khóa học:
* Tên khóa học:



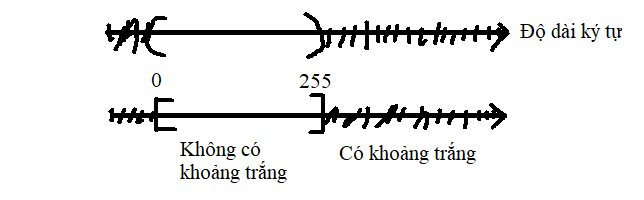
* Hình ảnh:



* Đánh giá khóa học:
* Số sao



* Nội dung



#### 3.2.1.2. Phân vùng giá trị biên

* Đăng nhập:
* Email:
* Kiểm tra định dạng:

🗹 Valid:

* Example@gmail.com

🗷 Invalid:

* “Khoảng trắng”
* ẽample@gmail.com
* exa mple@gmail.com
* examplegmail.com
* example@.com
* example@gmailcom
* example@gmail.
* Kiểm tra độ dài giá trị

🗹 Valid:

* Example@gmail.com

🗷 Invalid:

* “Trống”
* chieudangdanbuonghatmuaroixuongkhonggianlangimsuytuvanvuongngoimongmodonphuongnhodenmotnangthogiodonguavemangnhungechengolatraitimkhocanheouasethoibuondaunhungthatcaydangkhibietlauoaminhchilamotnguoidensaubietemdangconguoioganbennhunganhsevandungnoidayvachoem@gmail.com
* Password:
* Kiểm tra định dạng:

🗹 Valid:

* Example@123
* @123Abcde
* 123@Abcde

🗷 Invalid:

* “Khoảng trắng”
* Example@
* Example123
* example@123
* exampletest
* Exam ple@123
* Kiểm tra độ dài giá trị

🗹 Valid:

* Example@123
* @123Abcde
* 123@Abcde

🗷 Invalid:

* “Trống”
* Exam@1
* Exampletest@123456
* Thêm khóa học:
* Tên khóa học:
* Kiểm tra định dạng

🗹 Valid:

* Java swing
* CCNA
* CEHv12

🗷 Invalid:

* “Khoảng trắng”
* Kiểm tra độ dài giá trị

🗹 Valid:

* CISSP
* CISM
* API.NET

🗷 Invalid:

* “Trống”
* Lập trình…..(dài hơn 255 ký tự)
* Hình ảnh khóa học
* Kiểm tra định dạng

🗹 Valid:

* https://www.cisco.com/c/en/us/training-events/training-certifications/certifications/associate/ccna/jcr:content/Grid/category\_atl\_df1b/layout-category-atl/blade\_836154571\_copy/bladeContents/halves/H-Half-2/tile.img.png/1643317912632.png

🗷 Invalid:

* “Khoảng trắng”
* Kiểm tra độ dài giá trị

🗹 Valid:

* https://www.cisco.com/c/en/us/training-events/training-certifications/certifications/associate/ccna/jcr:content/Grid/category\_atl\_df1b/layout-category-atl/blade\_836154571\_copy/bladeContents/halves/H-Half-2/tile.img.png/1643317912632.png

🗷 Invalid:

* “Trống”
* Đánh giá khóa học
* Số sao:
* Kiểm tra định dạng

🗹 Valid:

* 1
* 2
* 3
* 4
* 5

🗷 Invalid:

* Một
* Năm
* Chín
* “Khoảng trắng”
* Kiểm tra độ dài giá trị

🗹 Valid:

* 1
* 2
* 3
* 4
* 5

🗷 Invalid:

* “Trống”
* 0
* 6
* Nội dung:
* Kiểm tra định dạng

🗹 Valid:

* Hay
* Tốt

🗷 Invalid:

* “Khoảng trắng”
* Kiểm tra độ dài giá trị

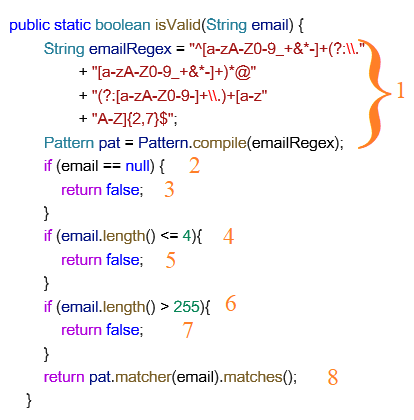
🗹 Valid:

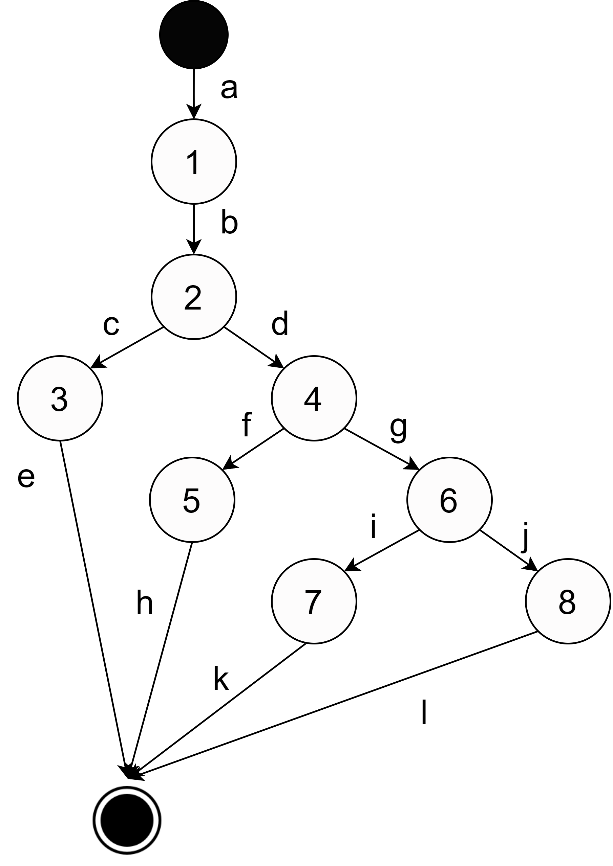
* Tuyệt vời

🗷 Invalid:

* “Trống”
* Thật cảm động…(nhiều hơn 255 ký tự)

### 3.2.2. Kiểm thử hộp trắng

* Đăng nhập
* Email:



Độ phức tạp:

Tập đường dẫn cơ sở:

* a, b, c, e
* a, b, d, f, h
* a, b, d, g, i, k
* a, b, d, g, j, l

|  |  |  |
| --- | --- | --- |
| Đường dẫn | Đầu vào | Đầu ra mong muốn |
| a, b, c, e | “Trống” | Lỗi |
| a, b, d, f, h | @e.c | Lỗi |
| a, b, d, g, i, k | chieudangdan...@gmail.com(hơn255 ký tự) | Lỗi |
| a, b, d, g, j, l | example@gmail.com | Thành công |

* Password: