Aptech™
COMPUTER EDUCATION
Unleash your potential

SESSION 04

Xây dựng TEST CASE

Nội dung

Kiểm thử hộp đen

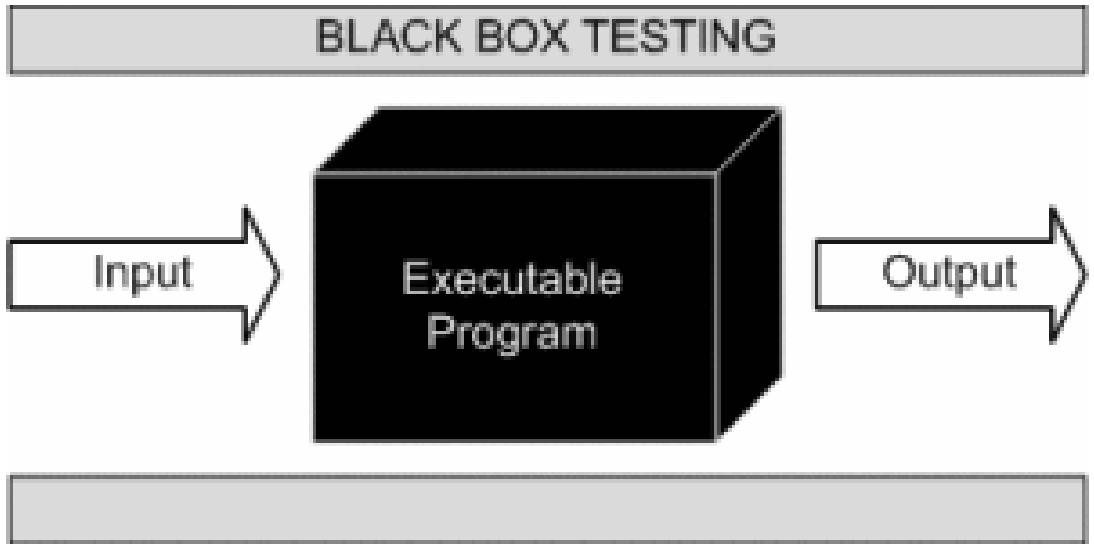
Kiểm thử hộp trắng

Khái niệm kiểm thử hộp đen

- Kiểm thử hộp đen: là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.
- Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out.
- Người kiểm thử nên xây dựng các nhóm giá trị đầu vào mà sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của chương trình.
- Cách tiếp cận của các tester đối với hệ thống là không dùng bất kỳ một kiến thức về cấu trúc lập trình bên trong hệ thống, xem hệ thống là một cấu trúc hoàn chỉnh, không thể can thiệp vào bên trong.

Khái niệm kiểm thử hộp đen

- Black Box Testing chủ yếu là được thực hiện trong Function test và System test.
- Phương pháp này được đặt tên như vậy bởi vì các chương trình phần mềm, trong con mắt của các tester, giống như một hộp đen; bên trong mà người ta không thể nhìn thấy. Phương pháp này cố gắng tìm ra các lỗi trong các loại sau:
  - Chức năng không chính xác hoặc thiếu.
  - Lỗi giao diện.
  - Lỗi trong cấu trúc dữ liệu hoặc truy cập cơ sở dữ liệu bên ngoài.
  - Hành vi hoặc hiệu suất lỗi.
  - Khởi tạo và chấm dứt các lỗi.



Ưu điểm của kiểm thử hộp đen

- Các **tester**s được thực hiện từ quan điểm của người dùng và sẽ giúp đỡ trong việc sáng tỏ sự chênh lệch về thông số kỹ thuật.
- Các **testers** theo phương pháp black box không có “mối ràng buộc” nào với code, và nhận thức của một **tester** rất đơn giản: một source code có nhiều lỗi. Sử dụng nguyên tắc, "Hỏi và bạn sẽ nhận" các tester black box tìm được nhiều bugs ở nơi mà các **developer** không tìm thấy.
- **Testers** có thể không phải IT chuyên nghiệp, không cần phải biết ngôn ngữ lập trình hoặc làm thế nào các phần mềm đã được thực hiện.
- Các **testers** có thể được thực hiện bởi một cơ quan độc lập từ các **developer**s, cho phép một cái nhìn khách quan và tránh sự phát triển thiên vị.
- Hệ thống thật sự với toàn bộ yêu cầu của nó được kiểm thử chính xác.
- Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

Nhược điểm của kiểm thử hộp đen

- Dữ liệu đầu vào yêu cầu một khối lượng mẫu (sample) khá lớn.
- Nhiều dự án không có thông số rõ ràng thì việc thiết kế test case rất khó và do đó khó viết kịch bản kiểm thử do cần xác định tất cả các yếu tố đầu vào, và thiếu cả thời gian cho việc tập hợp này.
- Kiểm thử black box được xem như "là bước đi trong mê cung tối đen mà không mang đèn pin" bởi vì tester không biết phần mềm đang test đã được xây dựng như thế nào. Có nhiều trường hợp khi một tester viết rất nhiều trường hợp test để kiểm tra một số thứ có thể chỉ được test bằng một trường hợp test và/hoặc một vài phần cuối cùng không được test hết.

Flow của Black Box Testing

Kế hoạch

Thiết kế

Tạo test case

Thực hiện test

Báo cáo test

Nội dung công việc trong công đoạn test

- **Kế hoạch test:** Chỉ ra rõ ràng mục đích và phạm vi của công đoạn test để kiểm tra xem là test bằng cách tiếp cận như thế nào. Điều chỉnh resource thành viên và quyết định cả schedule.
- **Thiết kế test:** Quyết định xem là sẽ sử dụng cái gì cho mục đích và loại test cần được thực hiện trong công đoạn test đó, chức năng đối tượng test, phương pháp test, import và export test. Ngoài ra cũng quyết định cụ thể hơn nguyên liệu cần thiết để thực hiện test hay tiêu chuẩn quyết định thành công/ không thành công.
- **Tạo testcase:** Tạo document ghi trạng thái trước khi bắt đầu test và kết quả mong đợi (kết quả chạy đối tượng test theo điều kiện và trình tự thao tác khi thực hiện test sẽ như thế nào) và cột trạng thái (cột ghi lại kết quả thao tác của đối tượng test).
- **Thực hiện test:** Vừa xem testcase vừa cho chạy phần mềm thực tế để tiến hành test, sau đó đánh dấu kết quả bằng dấu passed hoặc failed vào cột trạng thái của test case. Trường hợp có test case khác với kết quả mong đợi thì ghi dấu fail vào cột trạng thái, rồi tạo bản báo cáo lỗi. Trong bản báo cáo lỗi: trình bày nội dung mô tả hiện tượng khác với kết quả mong đợi và hiện tượng đó phát sinh trong trường hợp như thế nào (thao tác, giả nhập, điều kiện,...)
- **Báo cáo test:** Tóm tắt kết quả để báo cáo. Căn cứ vào các loại dữ liệu (mục thực hiện, hiệu quả của việc test, công số thực hiện,...) và dữ liệu lỗi (số lỗi được tìm ra, số lỗi theo mức độ quan trọng,...) để đánh giá xem có thỏa mãn tiêu chuẩn passed / failed của test không? Ngoài ra cũng đề xuất thêm risk có thể sinh ra sau khi release và mục cần bổ sung trong dự án cho giai đoạn tiếp theo.

Phân loại quan điểm / cách tiếp cận Test

Phân loại lớn	Phân loại trung	Quan điểm test
Chức năng	**Thông thường**	- Chức năng cơ bản - Di chuyển (Trang thái, MH) (GUI) - Thiết định (bảo trì, thay đổi, phản ánh) - Security - Hiển thi - User interface
	Bất thường	- Nhập giá trị bất thường - Kiểm tra lỗi - Phục hồi lỗi - Dữ liệu bất thường - Thiết bị nhớ bất thường - Trạng thái bất thường - Môi trường bất thường - Thao tác bất thường
	Kết hợp	- Hoạt động đồng thời - Hoạt động gián đoạn - Xử lý loại trừ - Tính tương hỗ - Cấu trúc - Tính vận hành tương hỗ - Option/ hàng phụ thuộc
Phi chức năng		- Tốc độ xử lý - Loading - Dung lượng lớn - Hoạt động liên tục - Thiếu vùng thông tin - Thiếu resource
User		- Chất lượng output - Usability - Tính hấp dẫn - Kịch bản nghiệp vụ - install, bảo trì
Test		- Test chỉnh sửa - Khuyết điểm quá khứ - Test quy hồi - Testability

Phương pháp kiểm thử hộp đen

- Phân lớp tương đương – (Equivalence partitioning / class)
- Phân tích giá trị biên (Boundary Value Analysis)
- Đồ thị nguyên nhân - kết quả (Cause Effect Graphing)
- Sủ dụng bảng quyết định (Decision Tables)
- Đoán lỗi (Error Guessing)

Phân vùng tương đương

- Phân vùng tương đương là một kỹ thuật kiểm thử phần mềm có liên quan đến phân chia các giá trị đầu vào thành các phân vùng hợp lệ và không hợp lệ, sau đó chúng ta sẽ viết ra các kịch bản kiểm thử cho từng phần, chọn giá trị đại diện từ mỗi phân vùng làm dữ liệu thử nghiệm.

Phân vùng tương đương

- Phân vùng tương đương: là kỹ thuật thực hiện test theo từng class đồng giá trị (tập hợp điều kiện cùng một thao tác).
- Tập hợp giá trị đầu vào có cùng một kết quả xử lý, tập hợp thời gian có cùng một kết quả xử lý, tập hợp kết quả export được xử lý cùng một giá trị nhập.
- Mục đích : Giảm đáng kể số lượng test case cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện.
- Chọn tối thiểu một giá trị đại diện từ các class đồng giá trị để tiến hành test.

Phân vùng tương đương (Ví dụ)

- Yêu cầu nhập 3 (ký tự) <= mật khẩu <= 30 (ký tự)
  - Nhập đúng đưa ra thông báo hợp lệ.
  - Nhập sai: đưa ra thông báo yêu cầu nhập lại.
- Như vậy ta sẽ thực hiện 2 testcases: một giá trị cho phần **PASSED** và một giá trị cho phần **FAILED**.

Phân tích giá trị biên

- **Phân tích giá trị biên** là một kỹ thuật kiểm thử phần mềm có liên quan đến việc xác định biên (ranh giới) của điều kiện mô tả cho các giá trị đầu vào và chọn giá trị ở biên và bên cạnh giá trị biên làm dữ liệu kiểm thử.
- Phương pháp phân tích giá trị biên sẽ đưa ra các giá trị đặc biệt, bao gồm loại dữ liệu, giá trị lỗi, bên trong, bên ngoài biên giá trị, lớn nhất và nhỏ nhất.
- Những LTV nhiều kinh nghiệm chắc chắn đã từng gặp phải các lỗi của hệ thống ngay tại giá trị biên. Đó là lý do tại sao phân tích giá trị biên lại quan trọng khi kiểm thử hệ thống.

Phân tích giá trị biên

- Trình tự thực hiện:
  - Tìm ra đường biên
  - Quyết định giá trị biên
  - Quyết định giá trị để test:
    - Giá trị biên.
    - Dưới giá trị biên. (Nếu là class đồng giá trị)
    - Trên 1 giá trị biên. (Nếu là class đồng giá trị)

Phân tích giá trị biên (Ví dụ)

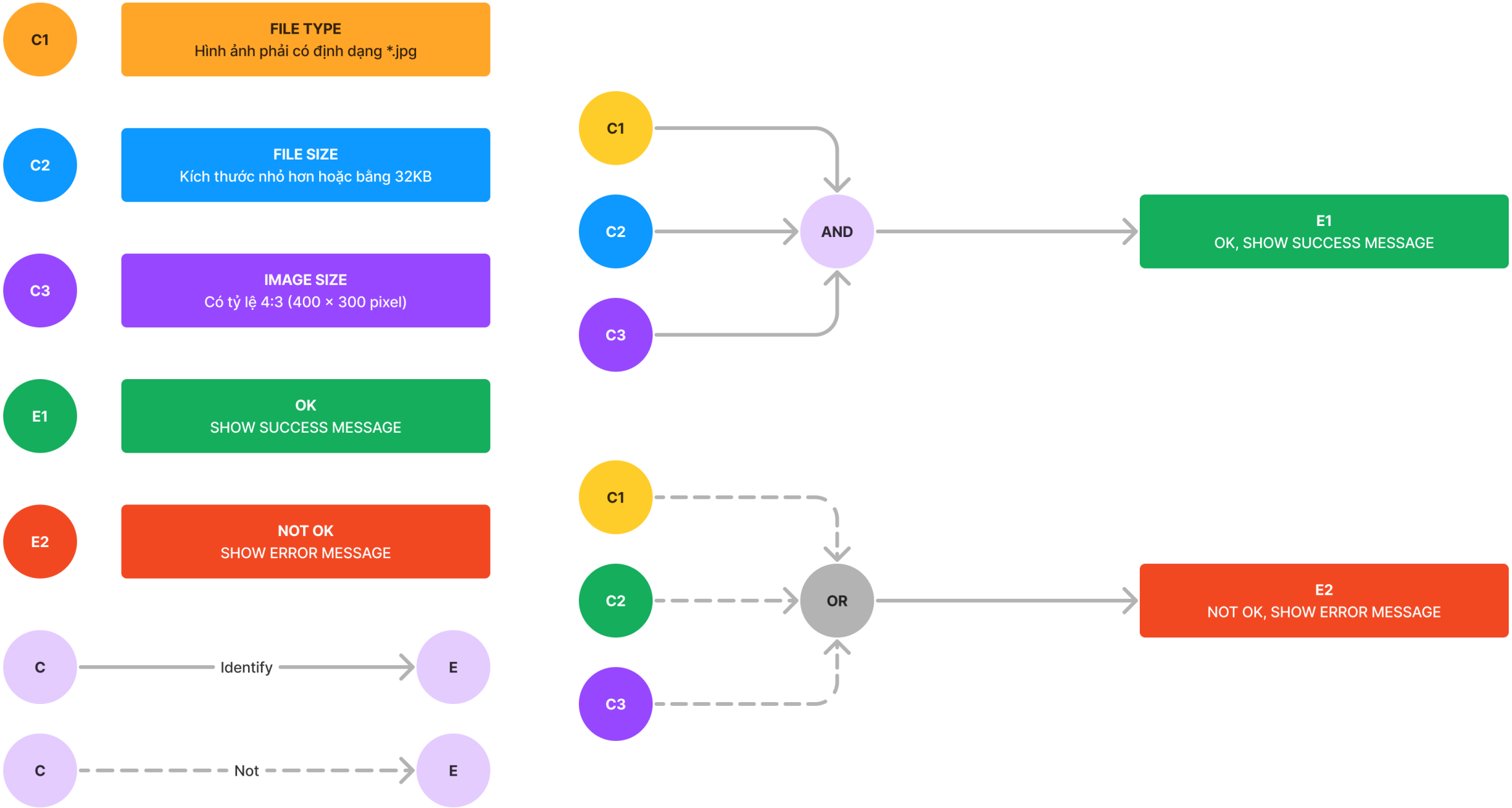
- Yêu cầu nhập 3 (ký tự) <= mật khẩu <= 30 (ký tự)
  - Giá trị biên là 3 và 30.
  - Giá trị dưới ngoài biên là 2 và trên ngoài biên là 31
  - Giá trị trên và dưới trong biên là 4 => 29 và được phép giản lược thành 2 đơn vị là 4 và 29
- Các class đồng giá trị vô hiệu là < 3 và > 30
- Các class đồng giá trị hữu hiệu là 3 đến 30

Đồ thị Nguyên nhân - Kết quả (Cause Effect Graph)

- Là một kỹ thuật thiết kế kiểm thử phần mềm liên quan đến việc xác định các trường hợp (điều kiện đầu vào: **Cause**) và các hiệu ứng (điều kiện đầu ra: **Effect**).
- Vì các hệ thống hiện nay đều được phát triển trên nền tảng OOP, do đó, chúng ta có thể có được một đồ thị các đối tượng mà hệ thống định nghĩa và kết nối.
- Từ đồ thị này, chúng ta dễ dàng biết các mối quan hệ của những đối tượng mà hệ thống xử lý, từ đó sẽ cho chúng ta các kịch bản kiểm thử phù hợp.

Các ví dụ

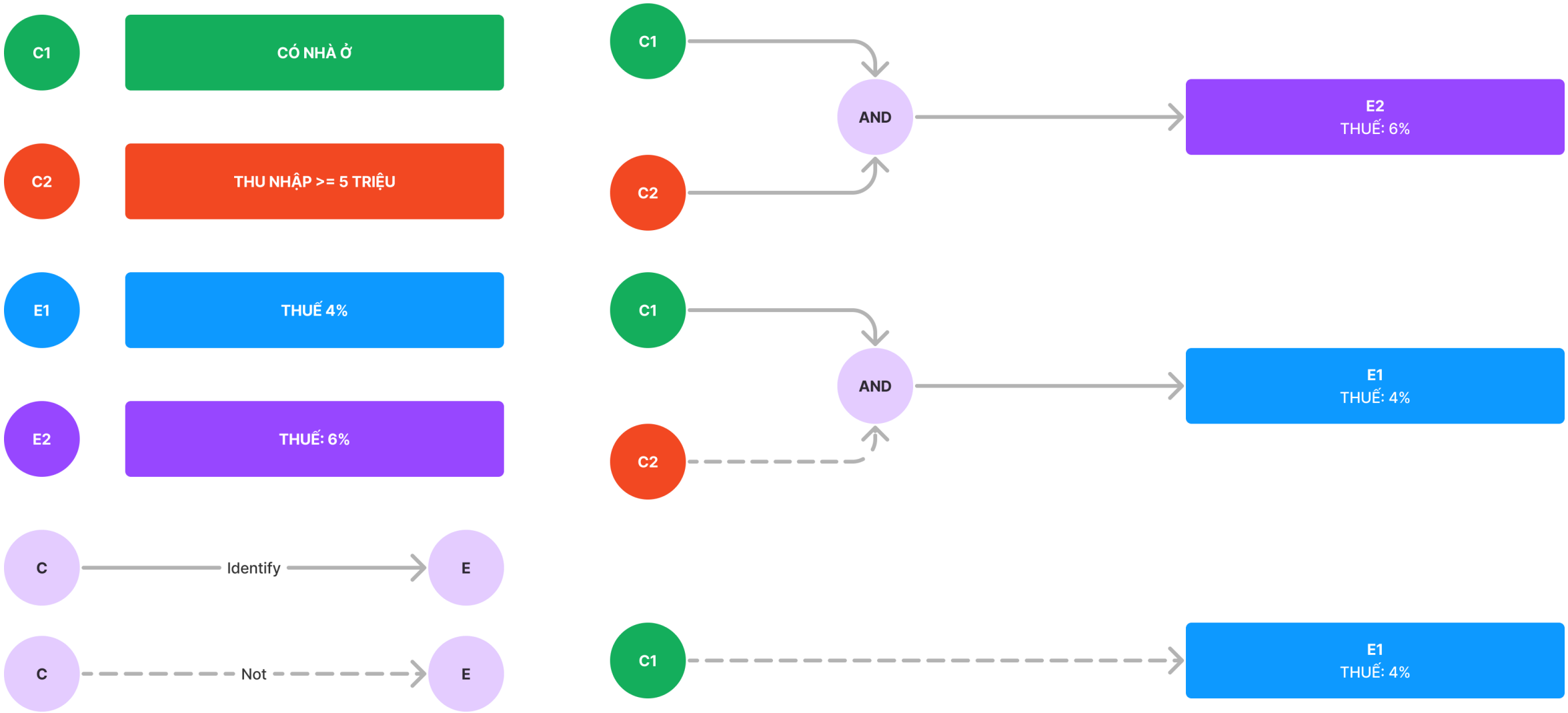
- **Điều kiện upload thành công là:**
  - Hình ảnh phải có định dạng *.jpg.
  - Kích thước của file hình ảnh từ 32Kb trở xuống.
  - Độ phân giải: 400*300 (pixel)


C1
FILE TYPE
Hình ảnh phải có định dạng *.jpg
C2
FILE SIZE
Kích thước nhỏ hơn hoặc bằng 32KB
C3
IMAGE SIZE
Có tỷ lệ 4:3 (400 × 300 pixel)
E1
OK
SHOW SUCCESS MESSAGE
E2
NOT OK
SHOW ERROR MESSAGE
C
Identify
E
C
Not
E
C1
C2
C3
AND
E1
OK, SHOW SUCCESS MESSAGE
C1
C2
C3
OR
E2
NOT OK, SHOW ERROR MESSAGE

Các ví dụ

Để tính thuế thu nhập, người ta có mô tả sau:

- **Người chưa có nhà nộp 4% thuế thu nhập.**
- **Người có nhà ở nộp thuế theo bảng sau:**
  - Nếu tổng thu nhập <= 5 triệu đồng thì chịu thuế 4%
  - Nếu tổng thu nhập > 5 triệu đồng thì chịu thuế 6%


C1
CÓ NHÀ Ở
C2
THU NHẬP >= 5 TRIỆU
E1
THUẾ 4%
E2
THUẾ: 6%
C
Identify
E
C
Not
E
C1
C2
AND
E2
THUẾ: 6%
C1
C2
AND
E1
THUẾ: 4%
C1
E1
THUẾ: 4%

Bài tập (Cause Effect Graph)

- **Điều kiện rút tiền thành công tại ATM là:**
  - Thẻ hợp lệ.
  - Nhập số PIN hợp lệ.
  - 3 lần sai số PIN là khóa thẻ.
  - Số tiền cần rút <= số dư tài khoản.

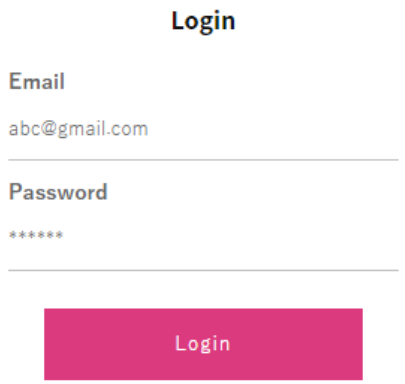
Bảng quyết định

- Là dùng bảng để hiển thị danh sách các thao tác phần mềm được quyết định trên các điều kiện khác nhau.
- Decision table testing chú trọng vào nhiều điều kiện để thực hiện test.
- Kỹ thuật test này sẽ trở nên quan trọng khi yêu cầu test trở nên phức tạp, phải kết hợp nhiều điều kiện khác nhau. Nó cũng giúp cho việc test các business logic phức tạp một cách dễ dàng hơn.
- **Giá trị biên (Boundary value) và Phân vùng tương đương (Equivalence partition)** là các kỹ thuật được sử dụng nếu hệ thống hiển thị cùng một kết quả đầu ra của một tập hợp lớn các input - đầu vào. Tuy nhiên, trong một hệ thống với mỗi bộ giá trị đầu vào khác nhau, kết quả đầu ra của hệ thống khác nhau thì kỹ thuật giá trị biên và phân vùng tương đương không hiệu quả trong việc đảm bảo phạm vi test.
- Bảng quyết định có thể được sử dụng làm tài liệu tham khảo, tài liệu ghi lại các yêu cầu của dự án hoặc được dùng để phát triển các chức năng.

Bảng quyết định (Ưu điểm và Khuyết điểm)

- **ƯU ĐIỂM:**
  - Dễ dàng xây dựng và chuyển đổi thành một bộ quy tắc. Có thể được sử dụng trong quá trình tạo và test các case test hoặc kiểm tra logic của hệ thống dựa trên knowledge-based của hệ thống.
  - Dựa vào bảng quyết định có thể phát hiện ra một số case test mà khi xây dựng test case theo cách thông thường tester sẽ dễ bị thiếu.
  - Được dùng làm tài liệu khi làm việc với stakeholders - các bên liên quan và các thành viên nontechnical trong team dự án vì bảng quyết định trình bày, minh họa các vấn đề dưới dạng bảng giúp cho mọi người dễ hiểu hơn.
- **Khuyết điểm**
  - Khi số lượng cái input đầu vào tăng thì bảng quyết định sẽ trở nên phức tạp hơn.
  - Không có các bước chi tiết step by step để thực hiện test.

Bảng quyết định



Điều kiện	Quy tắc 1	Quy tắc 2	Quy tắc 3	Quy tắc 4
Email (T/F)	T	T	F	F
Mật khẩu (T/F)	T	F	T	F
Kết quả (E/H)	H	E	E	E

Chú thích:

- **T - True:** Nhập đúng email và mật khẩu.
- **F - False:** Email hoặc mật khẩu bị sai.
- **E - Error:** Hiển thị lỗi.
- **H - Home:** Hiển thị trang chủ.

Diễn giải:

- Trường hợp 1: Email và mật khẩu đúng, người dùng sẽ được chuyển hướng đến trang chủ.
- Trường hợp 2: Email đúng, mật khẩu sai; người dùng sẽ nhận được thông báo lỗi.
- Trường hợp 3: Email sai, mật khẩu đúng; người dùng sẽ nhận được thông báo lỗi.
- Trường hợp 4: Email và mật khẩu sai, người dùng sẽ nhận được thông báo lỗi.

Bảng quyết định (Cách tính số lượng các cột trường hợp)

- Số lượng các cột trường hợp trong bảng được tính bằng công thức **2^n**. Trong đó n là số lượng các input đầu vào.
- Với form login ở ví dụ 1 chúng ta có 2 input đầu vào (email và mật khẩu) -> Số lượng cột trường hợp sẽ là 2^2 = 4 cột.

Bảng quyết định (Cách tính số lượng các cột trường hợp)

- Điều kiện đăng nhập thành công 1 hệ thống là:
  1. ID (username) hợp lệ.
  2. Password hợp lệ.
  3. 3 lần nhập sai password => Lock
  4. Có quyền truy cập vào hệ thống.

	Step	Description
Main Success Scenario **A: Actor** **S: System**	1	A: Enters user ID
	2	S: Validates user ID and asks for password
	3	A: Enters password
	4	S: Validates password
	5	S: Allows user into system
Message Scenarios	2a	User ID is invalid S: Display message and ask for user ID again
	4a	Password is invalid S: Display message and ask for password retry (twice, if necessary)
	4b	Account is locked S: Displayed locked user ID message

Bảng quyết định (Cách tính số lượng các cột trường hợp)

4 INPUT ⇒ 2^4 = 16 CASES

Điều kiện (Rule)	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
ID hợp lệ	N	N	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	N	N	Y
Password hợp lệ	N	N	N	Y	Y	Y	Y	N	Y	Y	N	N	Y	N	Y	N
Sai password 3 lần	N	N	Y	Y	Y	Y	N	N	N	Y	N	Y	N	Y	N	Y
Truy cập vào được	N	Y	Y	Y	Y	N	N	N	N	N	Y	Y	Y	N	Y	N
	TC1	TC1	TC1	TC1			TC4	TC2	TC1	TC1	TC2	TC3	TC5	TC1	TC1	TC3

Bảng quyết định (Cách tính số lượng các cột trường hợp)

Điều kiện (Rule)	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
ID hợp lệ	N	N	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	N	N	Y
Password hợp lệ	N	N	N	Y	Y	Y	Y	N	Y	Y	N	N	Y	N	Y	N
Sai password 3 lần	N	N	Y	Y	Y	Y	N	N	N	Y	N	Y	N	Y	N	Y
Truy cập vào được	N	Y	Y	Y	Y	N	N	N	N	N	Y	Y	Y	N	Y	N
	TC1	TC1	TC1	TC1			TC4	TC2	TC1	TC1	TC2	TC3	TC5	TC1	TC1	TC3

Bảng quyết định (Cách tính số lượng các cột trường hợp)

Điều kiện (Rule)					
ID hợp lệ	N	Y	Y	Y	Y
Password hợp lệ	-	N	N	Y	Y
Sai password 3 lần	-	N	Y	-	-
Truy cập vào được	-	-	-	N	Y
	TC1	TC2	TC3	TC4	TC5

Bảng quyết định (Cách tính số lượng các cột trường hợp)

Collapsed Decision Table	TC1	TC2	TC3	TC4	TC5
Conditions					
ID valid?	N	Y	Y	Y	Y
Password valid?	-	N	N	Y	Y
Third invalid password attempt?	-	N	Y	-	-
Access to system?	-	-	-	N	Y
Actions					
Reject ID with message	Y	N	N	N	N
Ask for new password	N	Y	N	N	N
Lock account	N	N	Y	N	N
Display error message	N	N	N	Y	N
Allow user to login	N	N	N	N	Y

Các ví dụ

Để tính thuế thu nhập, người ta có mô tả sau:

- **Người chưa có nhà nộp 4% thuế thu nhập.**
- **Người có nhà ở nộp thuế theo bảng sau:**
  - Nếu tổng thu nhập <= 5 triệu đồng thì chịu thuế 4%
  - Nếu tổng thu nhập > 5 triệu đồng thì chịu thuế 6%

2 INPUT ⇛ 2^2 = 4 CASES

Điều kiện (Rule)	R1	R2	R3	R4
Có nhà ở	N	N	Y	Y
Thu nhập > 5 triệu	Y	N	N	Y
Thuế 4%	✓	✓	✓	-
Thuế 6%	-	-	-	✓

R1 và R2 gộp chung thành 1 CASE

(Nếu là xét tuần tự Có nhà rồi xét thu nhập)

Điều kiện (Rule)	R1	R2	R3	R4
Có nhà ở	N	N	Y	Y
Thu nhập > 5 triệu	Y	N	N	Y
Thuế 4%	✓	✓	✓	-
Thuế 6%	-	-	-	✓

Bảng quyết định (Cách tính số lượng các cột trường hợp)

- Điều kiện upload thành công là:
  - Hình ảnh phải có định dạng *.jpg.
  - Kích thước của file hình ảnh từ 32Kb trở xuống.
  - Độ phân giải: 400*300 (pixel)

Bảng quyết định (Cách tính số lượng các cột trường hợp)

- Điều kiện rút tiền thành công tại ATM là:
  - Thẻ hợp lệ.
  - Nhập số PIN hợp lệ.
  - 3 lần sai số PIN là khóa thẻ.
  - Số tiền cần rút <= số dư tài khoản.

Đoán lỗi

- Đoán lỗi là một kỹ năng quan trọng của tester, thậm chí có thể gọi là nghệ thuật.
- Một kiệt tác của trực giác. Phương pháp này đặc biệt dựa vào kinh nghiệm và kiến thức của tester.
- Nhiều tester cố gắng đoán xem phần nào của hệ thống mà có khả năng ẩn chứa lỗi.
- Với phương pháp này, họ không cần một công cụ hay một kịch bản kiểm thử nào khi bắt đầu vào việc.