Phương pháp lập trình

GVHD: Huỳnh Tuấn Anh

Nhóm : AD – SP

Lê Đăng Khôi – 15520385

Trương Thanh Nam – 15520526

Chapter 12 : Testing

Trước khi bạn lật qua chương này, "nhưng tôi không phải là người kiểm tra", hãy dừng lại và suy nghĩ. Thử nghiệm là một nhiệm vụ quan trọng đối với kỹ sư phần mềm và vì lý do này, các câu hỏi kiểm tra có thể xuất hiện trong cuộc phỏng vấn của bạn. Tất nhiên, nếu bạn đang áp dụng cho vai trò Kiểm tra (hoặcKỹ sư phần mềm trong thử nghiệm), thì đó là tất cả lý do tại sao bạn cần chú ý hơn.

Các vấn đề kiểm tra thường thuộc một trong bốn loại: (1) Thử nghiệm một đối tượng thế giới thực

(giống như một cây bút); (2) Kiểm tra một phần mềm; (3) Viết mã thử nghiệm cho một hàm; (4) Khắc phục sự cố một vấn đề tồn tại. Chúng tôi sẽ đề cập đến các phương pháp tiếp cận cho từng loại trong bốn loại này.

Hãy nhớ rằng tất cả bốn loại đều yêu cầu bạn không giả định rằng đầu vào hoặc người dùng sẽ chơi đẹp. Mong đợi sự lạm dụng và lên kế hoạch cho nó.

**Những gì người phỏng vấn đang tìm kiếm**

Trên bề mặt của họ, các câu hỏi kiểm tra có vẻ như họ đang chuẩn bị đưa ra một danh sách các trường hợp kiểm tra. Và ở một mức độ nào đó, điều đó đúng. Bạn cần phải tìm ra một danh sách các trường hợp thử nghiệm hợp lý.

Nhưng ngoài ra, người phỏng vấn muốn kiểm tra những điều sau đây:

• Hiểu biết về hình ảnh lớn: Bạn có phải là người hiểu phần mềm đó là gì không thực sự về? Bạn có thể ưu tiên các trường hợp thử nghiệm đúng cách không? Ví dụ: giả sử bạn yêu cầu thử nghiệm một hệ thống thương mại điện tử như Amazon. Thật tuyệt vời để đảm bảo rằng

hình ảnh sản phẩm xuất hiện ở đúng nơi, nhưng điều quan trọng hơn là

thanh toán hoạt động đáng tin cậy, các sản phẩm được thêm vào hàng đợi lô hàng và khách hang không bao giờ bị tính phí gấp đôi.

• Biết cách các mảnh phù hợp với nhau: Bạn có hiểu cách phần mềm hoạt động và làm thế nào nó có thể phù hợp với một hệ sinh thái lớn hơn? Giả sử bạn được yêu cầu thử nghiệm Google Bảng tính. Điều quan trọng là bạn thử mở, lưu và chỉnh sửa tài liệu.

Tuy nhiên, Google Spreadsheets là một phần của hệ sinh thái lớn hơn. Bạn cần thử nghiệm tích hợp với Gmail, với các trình cắm và các thành phần khác.

Tổ chức: Bạn có tiếp cận vấn đề một cách có cấu trúc hay bạn chỉ nghĩ ra bất cứ thứ gì đến đầu bạn? Một số ứng cử viên, khi được yêu cầu đến với các trường hợp thử nghiệm cho máy ảnh, sẽ chỉ nêu rõ mọi thứ và mọi thứ đến đầu họ. Một ứng cử viên tốt sẽ chia nhỏ các phần thành các danh mục như Tham gia Ảnh, Quản lý hình ảnh, Cài đặt, v.v. Cách tiếp cận có cấu trúc này cũng sẽ giúp bạn làm một công việc toàn diện hơn trong việc tạo ra các trường hợp thử nghiệm.

• Tính thực tiễn: Bạn có thực sự tạo ra các kế hoạch thử nghiệm hợp lý không? Ví dụ: nếu người dùng báo cáo rằng phần mềm gặp sự cố khi họ mở một hình ảnh cụ thể và bạn chỉ cần nói họ cài đặt lại phần mềm, thường không thực tế lắm. Kế hoạch thử nghiệm của bạn cần phải khả thi và thực tế để một công ty thực hiện.

Chứng minh những khía cạnh này sẽ cho thấy rằng bạn sẽ là một thành viên có giá trị của nhóm thử nghiệm.

**Thử nghiệm một đối tượng thế giới thực**

Một số ứng cử viên ngạc nhiên khi được hỏi những câu hỏi như cách kiểm tra bút. Xét cho cùng, bạn nên kiểm thử phần mềm, đúng không? Có thể, nhưng những câu hỏi "thế giới thực" này vẫn còn rất phổ biến. Hãy xem qua ví dụ này.

Câu hỏi: Làm thế nào bạn sẽ kiểm tra một cái kẹp giấy?

Bước 1: Ai sẽ sử dụng? Và tại sao?

Bạn cần thảo luận với người phỏng vấn đang sử dụng sản phẩm và cho những gì mục đích. Câu trả lời có lẽ không như bạn nghĩ. Câu trả lời có thể là "bởi giáo viên, để giữ giấy tờ với nhau, "hoặc nó có thể là" của các nghệ sĩ, để uốn cong thành hình dạng của động vật. "Hoặc, nó có thể là cả hai. Câu trả lời cho câu hỏi này sẽ định hình cách bạn xử lý phần còn lại

câu hỏi.

Bước 2: Các trường hợp sử dụng là gì?

Nó sẽ hữu ích cho bạn để thực hiện một danh sách các trường hợp sử dụng. Trong trường hợp này, trường hợp sử dụng có thể là chỉ cần buộc giấy lại với nhau theo cách không gây tổn hại (đến giấy).

Đối với các câu hỏi khác, có thể có nhiều trường hợp sử dụng. Nó có thể là, ví dụ, rằng sản phẩm cần có khả năng gửi và nhận nội dung hoặc viết và xóa, v.v.

Bước 3: Các giới hạn sử dụng là gì?

Các giới hạn sử dụng có thể có nghĩa là giữ tối đa ba mươi tờ giấy trong một lần sử dụng không có hư hỏng vĩnh viễn (ví dụ: uốn) và ba mươi đến năm mươi tờ với mức tối thiểu uốn cong vĩnh viễn.

Các giới hạn cũng mở rộng đến các yếu tố môi trường là tốt. Ví dụ, nên

kẹp giấy hoạt động trong nhiệt độ rất ấm (90 - 110 độ F)? Gì về cực lạnh.

Bước 4: Các điều kiện căng thẳng / thất bại là gì?

Không có sản phẩm nào không được chứng minh, vì vậy việc phân tích các điều kiện thất bại cần phải là một phần của thử nghiệm của bạn.

Một cuộc thảo luận tốt để có với người phỏng vấn của bạn là về thời điểm có thể chấp nhận (hoặc thậm chí là cần thiết) cho sản phẩm thất bại, và những gì thất bại nên có nghĩa là.

Ví dụ, nếu bạn đang thử nghiệm một máy giặt, bạn có thể quyết định rằng máy nên có thể xử lý ít nhất 30 áo sơ mi hoặc quần. Đang tải 30 - 45 chiếc quần áo có thể dẫn đến thất bại nhỏ, chẳng hạn như quần áo được làm sạch không đầy đủ. Nhiều hơn hơn 45 chiếc quần áo, sự thất bại nghiêm trọng có thể chấp nhận được. Tuy nhiên, cực đoan thất bại trong trường hợp này có lẽ có nghĩa là máy không bao giờ quay trên mặt nước. Nó chắc chắn có nghĩa là lũ lụt hoặc hỏa hoạn.

Bước 5: Bạn sẽ thực hiện thử nghiệm như thế nào?

Trong một số trường hợp, nó cũng có thể có liên quan để thảo luận chi tiết về việc thực hiện thử nghiệm.

Ví dụ: nếu bạn cần đảm bảo ghế có thể chịu được mức sử dụng bình thường trong năm năm, bạn có thể không thực sự đặt nó trong nhà và chờ năm năm. Thay vào đó, bạn cần để xác định mức sử dụng "bình thường" (bao nhiêu "ngồi" mỗi năm trên ghế? armrest?). Sau đó, ngoài việc thực hiện một số kiểm tra thủ công, bạn có thể sẽ muốn

máy để tự động hóa một số cách sử dụng.

**Thử nghiệm một phần mềm**

Thử nghiệm một phần mềm thực sự rất giống với việc thử nghiệm một đối tượng thế giới thực. Các sự khác biệt chính là kiểm thử phần mềm thường đặt trọng tâm lớn hơn vào chi tiết về thử nghiệm thực hiện.

Lưu ý rằng kiểm thử phần mềm có hai khía cạnh cốt lõi đối với nó:

• Thử nghiệm bằng tay và tự động: Trong một thế giới lý tưởng, chúng tôi có thể yêu thích tự động hóa mọi thứ, nhưng điều đó hiếm khi khả thi. Một số điều đơn giản hơn nhiều với hướng dẫn sử dụng thử nghiệm vì một số tính năng quá chất lượng cho máy tính hiệu quả kiểm tra (chẳng hạn như nếu nội dung đại diện cho nội dung khiêu dâm). Ngoài ra, trong khi một máy tính nói chung có thể chỉ nhận ra các vấn đề mà nó được yêu cầu tìm kiếm, con người quan sát có thể tiết lộ những vấn đề mới chưa được kiểm tra cụ thể. Cả hai con người và máy tính tạo thành một phần thiết yếu của quá trình thử nghiệm.

• Thử nghiệm B / ackBox so với thử nghiệm hộp trắng: Sự khác biệt này đề cập đến mức độ truy cập chúng ta có vào phần mềm. Trong thử nghiệm hộp đen, chúng tôi chỉ đưa ra phần mềm như là cần kiểm tra nó. Với thử nghiệm hộp màu trắng, chúng tôi có quyền truy cập có lập trình bổ sung để kiểm tra các chức năng riêng lẻ. Chúng tôi cũng có thể tự động hóa một số thử nghiệm hộp đen, mặc dù nó chắc chắn khó hơn nhiều.

Hãy đi qua một cách tiếp cận từ đầu đến cuối.

Bước 1: Có phải chúng tôi đang thực hiện Kiểm tra hộp đen hoặc Kiểm tra hộp trắng không?

Mặc dù câu hỏi này thường có thể bị trì hoãn đến một bước sau, tôi muốn loại bỏ nó sớm. Kiểm tra với người phỏng vấn xem bạn có đang làm thử nghiệm hộp đen hay không thử nghiệm hộp trắng - hoặc cả hai.

Bước 2: Ai sẽ sử dụng? Và tại sao?

Phần mềm thường có một hoặc nhiều người dùng đích và các tính năng được thiết kế với trong tâm trí. Ví dụ: nếu bạn được yêu cầu kiểm tra phần mềm để kiểm soát của phụ huynh trên web trình duyệt, người dùng mục tiêu của bạn bao gồm cả cha mẹ (người đang triển khai chặn)

và trẻ em (là người nhận chặn). Bạn cũng có thể có "khách" (người

những người không nên thực hiện cũng như không nhận được chặn).

Bước 3: Các trường hợp sử dụng là gì?

Trong trường hợp chặn phần mềm, các trường hợp sử dụng của cha mẹ bao gồm cài đặt phần mềm, cập nhật điều khiển, xóa điều khiển và tất nhiên là cá nhân của riêng họ sử dụng internet. Đối với trẻ em, các trường hợp sử dụng bao gồm truy cập nội dung pháp lý cũng như nội dung "bất hợp pháp".

Hãy nhớ rằng bạn không chỉ quyết định các trường hợp sử dụng một cách kỳ diệu. Đây là cuộc trò chuyện để có với người phỏng vấn của bạn.

Bước 4: Các giới hạn sử dụng là gì?

Bây giờ chúng ta đã xác định được các trường hợp sử dụng mơ hồ, chúng ta cần tìm ra chính xác cái này có nghĩa. Trang web bị chặn là gì? Chỉ nên trang "bất hợp pháp" bị chặn hoặc toàn bộ trang web? Ứng dụng có phải là "tìm hiểu" những gì là xấu nội dung, hoặc là nó dựa trên một danh sách trắng hoặc danh sách đen? Nếu nó phải tìm hiểu những gì không phù hợp nội dung là, mức độ dương tính giả hoặc âm tính giả nào có thể chấp nhận được?

Bước 5: Điều kiện căng thẳng / điều kiện thất bại là gì?

Khi phần mềm thất bại - chắc chắn nó sẽ như thế nào - cái gì sẽ thất bại như thế nào?

Rõ ràng, lỗi phần mềm không làm hỏng máy tính. Thay vào đó, có khả năng là phần mềm chỉ cho phép trang web bị chặn hoặc cấm trang web cho phép. Trong trường hợp sau, bạn có thể muốn thảo luận về khả năng ghi đè chọn lọc với mật khẩu từ bố mẹ.

Bước 6: Các trường hợp thử nghiệm là gì? Bạn sẽ thực hiện thử nghiệm như thế nào?

Đây là nơi phân biệt giữa thử nghiệm thủ công và tự động và giữa hộp đen và thử nghiệm hộp màu trắng, thực sự đi vào chơi.

Các bước 3 và 4 cần phải xác định rõ các trường hợp sử dụng. Ở bước 6, chúng tôi xác định thêm và thảo luận cách thực hiện thử nghiệm. Bạn đang thử nghiệm những tình huống chính xác nào?

Bước nào sau đây có thể được tự động? Yêu cầu can thiệp của con người?

Hãy nhớ rằng trong khi tự động hóa cho phép bạn thực hiện một số thử nghiệm rất mạnh mẽ, nó cũng

có một số hạn chế đáng kể. Kiểm tra thủ công thường là một phần của thử nghiệm thủ tục của bạn.

Khi bạn đi qua danh sách này, không chỉ rattle tắt mỗi kịch bản bạn có thể nghĩ đến. nó là vô tổ chức và bạn chắc chắn sẽ bỏ lỡ các danh mục chính. Thay vào đó, hãy tiếp cận điều này trong một cấu trúc cách thức. Chia nhỏ thử nghiệm của bạn thành các thành phần chính và đi từ đó.

Không chỉ bạn sẽ cung cấp danh sách đầy đủ hơn các trường hợp kiểm tra, nhưng bạn cũng sẽ cho thấy rằng bạn một người có cấu trúc, có phương pháp.

**Kiểm tra chức năng**

Theo nhiều cách, thử nghiệm một hàm là loại thử nghiệm đơn giản nhất. Cuộc trò chuyện thường là ngắn và ít mơ hồ hơn, vì thử nghiệm thường được giới hạn trong việc xác thực đầu vào và đầu ra.

Tuy nhiên, đừng bỏ qua giá trị của một số cuộc trò chuyện với người phỏng vấn của bạn. Bạn nên thảo luận về bất kỳ giả định nào với người phỏng vấn của bạn, đặc biệt là về cách thức để xử lý các tình huống cụ thể.

Giả sử bạn được yêu cầu viết mã để kiểm tra sắp xếp (mảng int], sắp xếp mảng các số nguyên. Bạn có thể tiến hành như sau.

Bước 7: Xác định các trường hợp kiểm tra

Nói chung, bạn nên suy nghĩ về các loại trường hợp kiểm tra sau đây:

• Trường hợp bình thường: Liệu nó có tạo ra đầu ra chính xác cho các đầu vào điển hình không? Nhớ suy nghĩ về các vấn đề tiềm ẩn ở đây. Ví dụ, vì việc phân loại thường đòi hỏi một số loại phân vùng, nó là hợp lý để nghĩ rằng các thuật toán có thể thất bại trên mảng với số lượng phần tử lẻ, vì chúng không thể được phân chia đồng đều. Bài kiểm tra của bạn trường hợp nên liệt kê cả hai ví dụ.

• Cực đoan: Điều gì sẽ xảy ra khi bạn vượt qua trong một mảng trống? Hoặc rất nhỏ (một

phần tử)? Điều gì nếu bạn vượt qua trong một rất lớn?

• Nulls và đầu vào "bất hợp pháp": Nó là đáng giá để suy nghĩ về cách mã nên hoạt động khi đưa ra đầu vào bất hợp pháp. Ví dụ: nếu bạn đang thử nghiệm một hàm để tạo ra thứ n Số Fibonacci, các trường hợp thử nghiệm của bạn có lẽ nên bao gồm tình huống trong đó n tiêu cực.

• Đầu vào lạ: Một loại đầu vào thứ tư đôi khi xuất hiện: đầu vào lạ. Gì xảy ra khi bạn vượt qua trong một mảng đã được sắp xếp? Hoặc một mảng được sắp xếp theo thứ tự ngược lại?

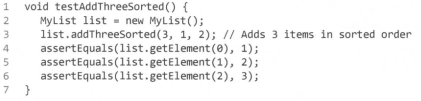
Việc tạo ra các kiểm tra này đòi hỏi kiến ​​thức về hàm bạn đang viết. nếu bạn không rõ ràng về những ràng buộc, bạn sẽ cần phải hỏi người phỏng vấn của bạn về điều này trước tiên.

Bước 2: Xác định kết quả mong đợi

Thông thường, kết quả mong đợi là hiển nhiên: đầu ra đúng. Tuy nhiên, trong một số trường hợp, bạn có thể muốn xác thực các khía cạnh bổ sung. Ví dụ, nếu phương thức sắp xếp trả về một bản sao mới được sắp xếp của mảng, bạn có lẽ nên xác nhận rằng mảng ban đầu chưa được sử dụng.

Bước 3: Viết mã kiểm tra

Khi bạn đã xác định các trường hợp thử nghiệm và kết quả, hãy viết mã để thực hiện kiểm tra các trường hợp phải khá đơn giản. Mã của bạn có thể trông giống như :



**Câu hỏi khắc phục sự cố**

Một loại câu hỏi cuối cùng là giải thích cách bạn gỡ lỗi hoặc khắc phục sự cố hiện có vấn đề. Nhiều ứng cử viên balk tại một câu hỏi như thế này, đưa ra câu trả lời không thực tế như "cài đặt lại phần mềm". Bạn có thể tiếp cận những câu hỏi này theo cách có cấu trúc, như còn gì nữa không.

Hãy xem xét vấn đề này với một ví dụ: Bạn đang làm việc trên Google Chrome

khi bạn nhận được báo cáo lỗi: Chrome gặp sự cố khi khởi chạy. Bạn sẽ làm gì?

Việc cài đặt lại trình duyệt có thể giải quyết sự cố của người dùng này, nhưng nó sẽ không giúp ích cho người khác người dùng có thể gặp phải sự cố tương tự. Mục tiêu của bạn là để hiểu những gì thực sự xảy ra, để các nhà phát triển có thể khắc phục nó.

Bước 1: Hiểu kịch bản

Điều đầu tiên bạn nên làm là đặt câu hỏi để hiểu nhiều về tình hình

càng tốt.

• Người dùng đã gặp sự cố này trong bao lâu?

• Phiên bản của trình duyệt là gì? Hệ điều hành nào?

• Vấn đề có xảy ra một cách nhất quán hay không hay tần suất xảy ra? Khi nào nó

xảy ra?

• Có báo cáo lỗi phát hành không?

Bước 2: Chia nhỏ sự cố

Bây giờ bạn đã hiểu chi tiết của kịch bản, bạn muốn chia nhỏ vấn đề thành các đơn vị có thể kiểm chứng. Trong trường hợp này, bạn có thể hình dung luồng chảy của tình huống sau:

1. Vào menu Bắt đầu của Windows.

2. Nhấp vào biểu tượng Chrome.

3. Trình duyệt bắt đầu.

4. Trình duyệt tải cài đặt.

5. Trình duyệt đưa ra yêu cầu HTTP cho trang chủ.

6. Trình duyệt nhận phản hồi HTTP.

7. Trình duyệt phân tích trang web.

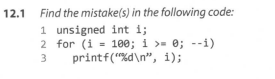
8. Trình duyệt hiển thị nội dung.

Tại một số thời điểm trong quá trình này, một cái gì đó không thành công và nó làm cho trình duyệt sụp đổ. Mạnh mẽ tester sẽ lặp qua các phần tử của kịch bản này để chẩn đoán vấn đề.

Bước 3: Tạo thử nghiệm cụ thể, có thể quản lý

Mỗi thành phần trên phải có hướng dẫn thực tế — những thứ bạn có thể yêu cầu người dùng thực hiện hoặc những việc bạn có thể tự làm (chẳng hạn như các bước lặp lại trên máy riêng). Trong thế giới thực, bạn sẽ giao dịch với khách hàng và b n không thể cung cấp họ hướng dẫn rằng họ không thể hoặc sẽ không làm.

**Câu hỏi phỏng vấn**



**Cách giải quyết**

Có hai lỗi trong mã này.

Đầu tiên, lưu ý rằng một int không dấu, theo định nghĩa, luôn luôn lớn hơn hoặc bằng không.

Do đó, điều kiện vòng lặp sẽ luôn đúng, và nó sẽ lặp vô hạn.

Mã đúng để in tất cả các số từ 100 đến 1, là i> 0. Nếu chúng ta thực sự muốn in không, chúng tôi có thể thêm một tuyên bố printf bổ sung sau vòng lặp for.



Một điều chỉnh bổ sung là sử dụng% u thay cho% d, vì chúng tôi đang in int không dấu.



Mã này bây giờ sẽ in chính xác danh sách tất cả các số từ 100 đến 1, theo thứ tự giảm dần

**12.2**Bạn được cấp nguồn cho một ứng dụng gặp sự cố khi nó chạy. Sau khi chạy nó mười lần trong một trình gỡ lỗi, bạn thấy nó không bao giờ bị treo ở cùng một nơi. Ứng dụng là một chuỗi duy nhất và chỉ sử dụng thư viện chuẩn C. Lập trình gì lỗi có thể gây ra sự cố này? Làm thế nào bạn sẽ kiểm tra mỗi một?

**Cách giải quyết**

Câu hỏi phần lớn phụ thuộc vào loại ứng dụng được chẩn đoán. Tuy nhiên, chúng tôi có thể đưa ra một số nguyên nhân chung gây ra sự cố ngẫu nhiên.

1. "Biến ngẫu nhiên:" Ứng dụng có thể sử dụng một số hoặc biến số ngẫu nhiên thành phần có thể không được sửa cho mỗi lần thực hiện chương trình. Ví dụ bao gồm đầu vào của người dùng, một số ngẫu nhiên do chương trình tạo hoặc thời gian trong ngày.

2. Vorinit / e không được khởi tạo: Ứng dụng có thể có biến chưa được khởi tạo, trong đó một số ngôn ngữ, có thể khiến nó mất giá trị tùy ý. Các giá trị của biến này có thể dẫn đến mã sử dụng một đường dẫn hơi khác nhau mỗi lần.

3. Rò rỉ bộ nhớ: Chương trình có thể hết bộ nhớ. Thủ phạm khác là hoàn toàn ngẫu nhiên cho mỗi lần chạy vì nó phụ thuộc vào số lượng quy trình đang chạy tại thời điểm cụ thể đó. Điều này cũng bao gồm tràn bộ nhớ heap hoặc tham nhũng của dữ liệu trên ngăn xếp.

4. Phụ thuộc bên ngoài: Chương trình có thể phụ thuộc vào ứng dụng, máy khác, hoặc tài nguyên. Nếu có nhiều phụ thuộc, chương trình có thể gặp sự cố bất kỳ lúc nào.

Để theo dõi vấn đề, chúng ta nên bắt đầu với việc học càng nhiều càng tốt về ứng dụng. Ai đang chạy nó? Họ đang làm gì với nó? Loại ứng dụng nào là nó?

Ngoài ra, mặc dù ứng dụng không bị lỗi chính xác ở cùng một nơi, đó là

có thể là nó được liên kết với các thành phần hoặc kịch bản cụ thể. Ví dụ, nó có thể là rằng ứng dụng không bao giờ bị treo nếu nó đơn giản được khởi chạy và không bị ảnh hưởng, và sự cố chỉ xuất hiện tại một số điểm sau khi tải tệp. Hoặc, có thể là tất cả các sự cố diễn ra trong các thành phần cấp thấp hơn, chẳng hạn như tệp I / O.

Nó có thể hữu ích để tiếp cận điều này bằng cách loại bỏ. Đóng tất cả các ứng dụng khác trên hệ thống. Theo dõi sử dụng tài nguyên rất cẩn thận. Nếu có các phần của chương trình, chúng tôi có thể vô hiệu hóa, làm như vậy. Chạy nó trên một máy khác và xem chúng ta có gặp vấn đề tương tự không.

Chúng ta càng có thể loại bỏ (hoặc thay đổi), chúng ta càng dễ dàng theo dõi vấn đề.

Ngoài ra, chúng tôi có thể sử dụng các công cụ để kiểm tra các tình huống cụ thể. Ví dụ: để điều tra vấn đề # 2, chúng tôi có thể sử dụng các công cụ thời gian chạy kiểm tra các biến chưa được khởi tạo.

Những vấn đề này càng nhiều về khả năng động não của bạn vì chúng là về tiếp cận. Bạn có nhảy khắp nơi, hét lên những gợi ý ngẫu nhiên không? Hay bạn tiếp cận nó một cách hợp lý, có cấu trúc? Hy vọng rằng, đó là phần sau.

**12.3** Chúng tôi có phương pháp sau được sử dụng trong trò chơi cờ vua: boolean canMoveTofint x, int

y). Phương thức này là một phần của lớp Piece và trả về xem phần đó có thể

di chuyển đến vị trí (x, y). Giải thích cách bạn sẽ kiểm tra phương pháp này

Cách giải quyết

Trong bài toán này, có hai loại thử nghiệm chính: xác thực trường hợp cực đoan (đảm bảo rằng chương trình không gặp sự cố về đầu vào xấu) và kiểm tra trường hợp chung. Tốt nhất nên bắt đầu với loại đầu tiên.

**Loại thử nghiệm # 1: Xác thực trường hợp cực đoan**

Chúng tôi cần đảm bảo rằng chương trình xử lý thông tin đầu vào xấu hoặc không bình thường một cách duyên dáng. Điều này

có nghĩa là kiểm tra các điều kiện sau:

• Thử nghiệm với số âm cho x và y

• Kiểm tra với x lớn hơn chiều rộng

• Thử nghiệm to hơn so với chiều cao

• Kiểm tra với một bảng đầy đủ

• Thử nghiệm với một bảng trống hoặc gần như trống

• Thử nghiệm với nhiều mảnh trắng hơn màu đen

• Thử nghiệm với nhiều mảnh màu đen hơn màu trắng

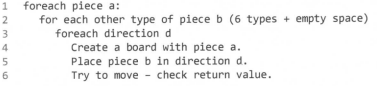
Đối với các trường hợp lỗi ở trên, chúng tôi nên hỏi người phỏng vấn xem có muốn quay lại không sai hoặc ném một ngoại lệ, và chúng ta nên kiểm tra cho phù hợp.

**Loại thử nghiệm # 2: Kiểm tra chung:**

Thử nghiệm chung rộng hơn nhiều. Lý tưởng nhất, chúng tôi sẽ kiểm tra mọi bảng có thể, nhưng có quá nhiều bảng. Tuy nhiên, chúng tôi có thể thực hiện mức độ phù hợp hợp lý của các bảng khác nhau.

Có 6 miếng trong cờ vua, vì vậy chúng tôi có thể kiểm tra từng mảnh chống lại mọi mảnh khác, trong mọi

có thể hướng.Điều này sẽ trông giống như mã dưới đây :



Chìa khóa cho vấn đề này là nhận ra rằng chúng tôi không thể kiểm tra mọi tình huống có thể xảy ra, ngay cả khi chúng ta muốn. Vì vậy, thay vào đó, chúng ta phải tập trung vào các lĩnh vực thiết yếu.

**12.4** Bạn sẽ tải thử nghiệm một trang web mà không sử dụng bất kỳ công cụ kiểm tra nào?

**Cách giải quyết**

Kiểm tra tải giúp xác định khả năng hoạt động tối đa của một ứng dụng web, như bất kỳ tắc nghẽn nào có thể ảnh hưởng đến hiệu suất của nó. Tương tự, nó có thể kiểm tra như thế nào ứng dụng phản hồi các biến thể trong tải.

Để thực hiện kiểm tra tải, trước tiên chúng tôi phải xác định các tình huống quan trọng về hiệu suất và các chỉ số đáp ứng các mục tiêu hiệu suất của chúng tôi. Tiêu chí điển hình bao gồm:

• Thời gian đáp ứng

• Thông lượng

• Tận dụng nguồn tài nguyên

• Tải trọng tối đa mà hệ thống có thể chịu.

Sau đó, chúng tôi thiết kế các thử nghiệm để mô phỏng tải trọng, cẩn thận để đo lường từng tiêu chí này.

Trong trường hợp không có các công cụ kiểm tra chính thức, chúng ta về cơ bản có thể tạo ra công cụ của riêng mình. Ví dụ: chúng tôi có thể mô phỏng người dùng đồng thời bằng cách tạo hàng nghìn người dùng ảo. Chúng tôi sẽ viết một chương trình đa luồng với hàng nghìn luồng, trong đó mỗi luồng hoạt động như người dùng trong thế giới thực đang tải trang. Đối với mỗi người dùng, chúng tôi sẽ lập trình đo lường thời gian đáp ứng, dữ liệu I / O, v.v.

Sau đó chúng tôi sẽ phân tích các kết quả dựa trên dữ liệu thu thập được trong các thử nghiệm và so sánh nó với các giá trị được chấp nhận.

**12.5** Bạn sẽ kiểm tra bút bằng cách nào?

**Cách giải quyết**

Vấn đề này phần lớn là tìm hiểu các ràng buộc và tiếp cận

vấn đề một cách có cấu trúc.

Để hiểu những khó khăn, bạn nên hỏi rất nhiều câu hỏi để hiểu "ai, cái gì, ở đâu, khi nào, làm thế nào và tại sao" của một vấn đề (hoặc như nhiều người trong số đó là áp dụng cho vấn đề). Hãy nhớ rằng một người kiểm tra giỏi hiểu chính xác những gì anh ta đang thử nghiệm trước khi bắt đầu công việc.

Để minh họa kỹ thuật trong vấn đề này, hãy để chúng tôi hướng dẫn bạn qua một cuộc trò chuyện giả.

**Người phỏng vấn**: Bạn sẽ kiểm tra bút bằng cách nào?

**Ứng cử viên**: Hãy để tôi tìm hiểu một chút về cây bút. Ai sẽ dùng bút?

**Người phỏng vấn**: Có thể là trẻ em.

**Ứng cử viên**: Được rồi, điều đó thật thú vị. Họ sẽ làm gì với nó? Họ sẽ viết, vẽ, hoặc làm cái gì khác với nó?

**Người phỏng vấn**: Vẽ.

**Ứng cử viên**: Ok, tuyệt vời. Về những gì? Giấy? Quần áo? Bức tường?

**Người phỏng vấn**: Trên quần áo.

**Ứng cử viên**: Tuyệt. Bút có loại đầu nào? Cảm thấy? Điểm bóng? Nó có nhằm rửa sạch, hoặc nó được dự định là vĩnh viễn?

**Người phỏng vấn**: Nó có ý định rửa sạch.

Nhiều câu hỏi sau, bạn có thể nhận được điều này:

**Ứng cử viên**: Được rồi, vì vậy khi tôi hiểu nó, chúng tôi có một cây bút đang được nhắm mục tiêu từ 5 đến 10 tuổi. Cây bút có đầu phớt và có màu đỏ, lục, lam và đen. Nó dự định rửa sạch khi quần áo được rửa sạch. Đúng không?

Các ứng cử viên bây giờ có một vấn đề khác biệt đáng kể so với những gì nó ban đầu dường như được. Đây không phải là không phổ biến. Thực tế, nhiều người phỏng vấn cố ý đưa ra vấn đề có vẻ rõ ràng (mọi người đều biết bút là gì!), chỉ để cho bạn khám phá ra đó là một vấn đề khác với những gì nó có vẻ. Niềm tin của họ là người dùng cũng làm như vậy điều, mặc dù người dùng vô tình làm như vậy.

Bây giờ bạn hiểu những gì bạn đang thử nghiệm, đó là thời gian để đưa ra một kế hoạch tấn công.

Chìa khóa ở đây là cấu trúc, Hãy xem xét các thành phần khác nhau của đối tượng hoặc vấn đề và đi từ đó. Trong trường hợp này, các thành phần có thể là:

• Kiểm tra sự thật: Xác minh rằng bút có đầu nhọn và mực là một trong những màu được cho phép.

• Mục đích sử dụng: Vẽ. Bút có viết đúng trên quần áo không?

• Mục đích sử dụng: Giặt. Nó có giặt quần áo không (ngay cả khi nó ở đó cho một khoảng thời gian dài)? Nó có rửa sạch trong nước nóng, ấm và lạnh không?

• An toàn: Bút có an toàn (không độc hại) cho trẻ em không?

• Sử dụng không chủ định: Trẻ em có thể sử dụng bút như thế nào? Họ có thể viết trên khác bề mặt, vì vậy bạn cần phải kiểm tra xem liệu hành vi có chính xác hay không. Họ cũng có thể stomp trên cây bút, ném nó, và như vậy. Bạn sẽ cần phải chắc chắn rằng bút giữ trong những điều kiện này.

Hãy nhớ rằng trong bất kỳ câu hỏi kiểm tra nào, bạn cần kiểm tra cả dự định và không mong muốn kịch bản. Mọi người không phải lúc nào cũng sử dụng sản phẩm theo cách bạn muốn.

**12.6** Bạn sẽ kiểm tra máy ATM trong hệ thống ngân hàng phân phối như thế nào?

**Cách giải quyết**

Điều đầu tiên cần làm trong câu hỏi này là làm rõ các giả định. Đặt các câu hỏi sau:

• Ai sẽ sử dụng máy ATM? Câu trả lời có thể là "bất kỳ ai" hoặc có thể là "bị mù" mọi người, "hoặc bất kỳ số câu trả lời nào khác.

• Họ sẽ sử dụng nó để làm gì? Câu trả lời có thể là "rút tiền", "chuyển tiền , "" kiểm tra số dư của họ "hoặc nhiều câu trả lời khác.

• Chúng ta phải thử những công cụ nào? Chúng tôi có quyền truy cập vào mã hoặc chỉ với máy ATM không?

Hãy nhớ rằng: một người kiểm tra tốt chắc chắn rằng cô ấy biết những gì cô ấy đang thử nghiệm!

Khi chúng tôi hiểu hệ thống trông như thế nào, chúng tôi sẽ muốn giải quyết vấn đề thành các thành phần có thể kiểm tra khác nhau. Các thành phần này bao gồm:

• Đăng nhập

• Việc rút tiền

• Gửi tiền

• Kiểm tra số dư

• Chuyển tiền

Chúng tôi có thể muốn sử dụng kết hợp kiểm tra thủ công và tự động.

Kiểm tra thủ công sẽ liên quan đến việc thực hiện các bước trên, đảm bảo kiểm tra tất cả các trường hợp lỗi (số dư thấp, tài khoản mới, tài khoản không tồn tại, v.v.).

Thử nghiệm tự động phức tạp hơn một chút. Chúng tôi sẽ muốn tự động hóa tất cả các tiêu chuẩn kịch bản, như được hiển thị ở trên và chúng tôi cũng muốn tìm một số vấn đề rất cụ thể, chẳng hạn như điều kiện chủng tộc. Lý tưởng nhất, chúng tôi có thể thiết lập một hệ thống kín với giả mạo và đảm bảo rằng, ngay cả khi ai đó rút tiền và gửi tiền nhanh chóng từ các địa điểm khác nhau, anh ta không bao giờ nhận tiền hoặc mất tiền mà anh ta không nên.

Trên hết, chúng ta cần ưu tiên bảo mật và độ tin cậy. Tài khoản của mọi người phải luôn là được bảo vệ và chúng tôi phải đảm bảo rằng tiền luôn được tính đúng. Không một ai muốn bất ngờ mất tiền! Một người kiểm tra tốt hiểu được các ưu tiên của hệ thống.