# **Thợ lành nghề #28: Phá vỡ tính đóng gói (Giám sát liều lượng – Phần 5)**

**Robert C. Martin**

**29/6/2004**

*…Tiếp tục từ phần trước*

**21/2/2002, 1100**

*Khi sự căng thẳng ở châu Âu tăng lên và thậm chí là bùng nổ thành chiến tranh, Clyde vẫn hiện hữu nhưng đã bị công chúng lãng quên, ngoại trừ Tombaugh. Ông tiếp tục theo dõi Clyde, tính toán đi tính toán lại quỹ đạo của nó. Kết quả của ông được công bố trên các tạp chí chuyên ngành, và nhận được sự quan tâm ngày càng lớn của các nhà khoa học ở cả hai phương diện của sự phân chia chính trị đang tăng lên. Tỷ lệ cho một vụ va chạm vào Trái Đất, mặc dù vẫn rất thấp, tiếp tục tăng dần với mỗi phép đo mới.*

*Đầu năm 1939, khi hầu hết các nhà khoa học chắc chắn rằng họ sẽ sớm không thể liên lạc được với các đồng nghiệp ở phía bên kia Đại Tây Dương, Tombaugh quyết định tổ chức một hội nghị thiên văn ở Stockholm. Hội nghị nhận được sự quan tâm rất lớn, không chỉ từ các nhà thiên văn học. Có vẻ các nhà khoa học ở nhiều lĩnh vực khác nhau cho rằng hội nghị này có thể là cơ hội cuối cùng của họ ở một cuộc họp quốc tế.*

*Clyde không phải chủ đề trọng tâm của buổi thảo luận chính thức, nhưng xung quanh quán bar, trong các phòng chờ, cuộc nói chuyện thường theo chiều hướng: “Nếu như?” Những cuộc thảo luận không quá nghiêm trọng khi mọi người đang tiêu khiển; ngoại trừ việc họ dùng để phát động các ý tưởng thú vị. Một số nói về việc phá hủy Clyde, số khác thì nghĩ cách làm chệch hướng nó, và thậm chí có một số người nói về việc trốn lên sao Kim hay sao Hỏa.*

*Mặc dù có rất nhiều ý tưởng, mọi người đều đồng ý rằng vấn đề căn bản là năng lượng. Tất cả các giải pháp đều đỏi hỏi nguồn năng lượng vượt qua khả năng con người tạo ra; và vây nên tất cả chỉ được coi là những điều phù phiếm bên quầy rượu. Sau đó, trong đêm cuối cùng của hội nghị, Leo Szilard, Neils Bohr và Lise Meitner công bố tin chấn động rằng các cuộc nghiên cứu của họ về phản ứng hạt nhân cho thấy một nguồn năng lượng như vậy có thể nằm trong tầm tay. Với cách nhìn của tình hình chính trị, rõ ràng mọi người không hẳn coi đây là tin tức tốt lành. Một vài người ở lại rất muộn đêm đó để thảo luận… các lựa chọn. Họ tự gọi mình với cái tên kì quái: Đội ngũ Stockholm.*

Jerry, Avery và tôi đi đến phòng quan sát ở đầu gút cao cấp của cánh tay gamma. Mỗi người một tách cà phê và ngồi ở một cái bàn quan sát quả cầu sao đang xoay tít dưới chân chúng tôi.

Jerry dời ánh nhìn từ cái bàn qua chỗ Avery và nói: “Ừm, Avery, mày kêu ca về cách dùng biến công khai của tao ở kiểm thử fixture mà chúng ta đang viết.”

Avery nhăn nhó và nói: “Ồ, đó không phải là thứ OO thực sự làm.”

“Mày có thể định nghĩa OO cho tao không?”

“Tại sao biến công khai không phải OO?”

“Vì OO có tính đóng gói, và biến công khai thì không được đóng gói.”

“Mày nghĩ OO yêu cầu tất các biến phải được đóng gói?”

“Dĩ nhiên!” Avery trả lời.

Jasmine đang ở bàn gần đó với Adelade. Cô ấy có vẻ hứng thú với cuộc nói chuyện này.

Jerry hỏi: “Điều tồi tệ nào sẽ đến nếu mày có một biến không được đóng gói?”

Avery trưng ra bộ mặt khinh thường và nói: “Những chương trình khác có thể thay đổi biến của anh!”

“Nếu như đó là điều mày đã dự tính? Nếu như mày muốn các chương trình khác thay đổi biến nào đó?”

“Ờ thì đó là có thứ gì đó bị lỗi với thiết kế của anh!” Avery tuyên bố với sự phẫn nộ chính đáng.

Jasmine tiến gần đến bàn của chúng tôi và nói: “Ù ôi, Avery, thật vớ vẩn.”

Avery không chú đến Jasmine cho đến giây phút này. Khi hắn ta thấy nàng, biểu hiện trên mặt hắn thay đổi với sự trộn lẫn giữa bối rối và hốt hoảng. Hắn lắp bắp như muốn trả lời nàng, nhưng Jasmine vẫn tiếp tục nói.

“Nhìn cậu đi, đồ bị thịt, đó không phải là nguyên tắc, đó là lý do. Mục đích của OO là giúp cậu quản lý các dependency. Thông thường giữ biến của cậu nội bộ sẽ giúp cậu làm điều này. Nhưng đôi khi biến nội bộ lại là vật cản đường.”

Jerry ngắt lời: “Giờ đợi một chút nào, Jasmine. Tôi đồng ý rằng các ngôn ngữ OO giúp cô quản lý các dependency, nhưng có một lợi ích quan trọng hơn là tính biểu hiện. Sẽ dễ hơn nhiều nếu diễn tả các khái niệm bằng ngôn ngữ OO.”

“Ừ, ừ, đúng rồi, đúng rồi.” Jasmine trả lời. “Nhưng nó không thiếu tính biểu hiện cái mà biến hệ thống phần mềm thành một khối hỗn độn với các module được kết nối với nhau; các dependency được quản lý không tốt cũng làm như thế. Tôi có thể tạo ra các hệ thống biểu đạt mà hỗn độn như địa ngục. Tôi đồng ý rằng tính biểu hiện là quan trọng, nhưng giữ sự liên kết còn quan trọng hơn nhiều!”

“Sao cô có thể nói thế được? Cô thật thiển cận! Nghệ thuật ở đâu chứ? Sự sắc sảo đâu rồi? Tất cả những gì cô quan tâm là các dependency đi đúng hướng.”

“Thế nào là khéo léo với các module được đặt cái tên đẹp đẽ hợp thành một mớ hỗn độn? Nghệ thuật nằm ở cấu trúc, Jerry. Nghệ thuật ở trong sự quản lý cẩn thận của các dependency giữa các module với nhau.”

“Cấu trúc quan trọng, chắc chắn rồi, nhưng mà….”

Tôi nghĩ có lẽ điều này sẽ kéo dài thêm một lúc nữa, và tôi thực sự muốn biết về biến công khai, vì họ thật sự làm phiền tôi. Nên tôi đã ngắt lời họ: “Xin lỗi, Jasmine, Jerry, nhưng nó phải làm cái gì với các biến công khai?”

Hai người họ nhìn Avery và tôi như thể họ chợt nhớ ra rằng còn chúng tôi ở đó. Jerry nói:

“Ơ, ừm, xin lỗi về việc này. Đây là cuộc tranh luận cũ của tao và Jasmine. Xem này, chúng tôi làm các biến nội bộ để nói cho các lập trình viên khác phớt lờ chúng. Đó là cách thể hiện cho người khác thấy các biến không phải là vấn đề của họ, và rằng chúng tôi muốn họ tập trung vào việc của bản thân họ.”

Jasmine đảo mắt. “Ồ, chúng ta bắt đầu lại nào! “Nhìn này, các cậu bé, (Tôi không thích bị gọi là cậu bé, đặc biệt là với Jasmine. Tôi thấy Avery cũng ngần ngại.) chúng tôi làm các biến nội bộ để giảm sự kết hợp. Các biến nội bộ là một tường lửa mà các depency không thể vượt qua. Không một module nào có thể kết hợp với một biến nội bộ.”

“Đó quả là một cách nhìn sắc lạnh.” Jerry nói. “Cô có thấy vậy là vô nhân tính không?”

Tôi muốn dừng cuộc tranh luận trước khi nó lại bùng phát trở lại, nên tôi buột miệng hỏi câu hỏi tiếp theo: “Được rồi, nhưng khi nào thì thuận lợi cho một biến trở thành công khai?”

Avery có vẻ đã hoàn hồn trở lại. Hắn ta nhìn tôi và phát ngôn: “Không bao giờ!”

Điều này khiến Jerry và Jasmine dừng lại giữa chừng. Cả hai đều quay sang Avery và nói: “Linh tinh – Vớ vẩn.”

“Nhìn đây, Avery” Jasmine “thỉnh thoảng cậu muốn kết hợp với một biến.”

“Đúng vậy” Jerry nói. “Thỉnh thoảng một biến công khai là cách tốt nhất để liên kết với ý định của mày.”

“Khi nào thì thế?”

“Chà, giống như trong kiểm thử fixture.” Jerry đáp. “Dũ liệu được sử dụng bởi fixture đến từ một nguồn, nhưng kiểm thử được kích hoạt bởi một nguồn khác. Một bên phải tải dữ liệu, bên còn lại vận hành nó. Chúng tôi muốn giữ chúng riêng biệt.”

“Đúng thế.” Jasmine nói. “Đó là tất cả về sự kết nối. Điều quan trọng là nguồn dữ liệu không được kết hợp với ứng dụng đang được thử nghiệm. Thực tế là các biến trong fixture công khai thì cũng vô hại.”

Nhưng nếu như có ai đó sừ dụng chúng?” Avery phàn nàn.

“Ai sẽ làm?” Jerry đáp. “Chúng là các biến trong kiểm thử fixture. Mọi người trong dự án đều biết rằng những biến đó nằm dưới sự kiểm soát của FitNess1 và không ai khác được dùng.”

“Nhưng họ có thể!”

Jasmine liếc mắt và nói” “Ừ, và họ có thể lấy một trong những biến nội bộ của cậu và làm nó công khai. Có gì khác biệt sao?”

Avery thêm phần ấp úng, nhưng có vẻ vẫn không thể đối mặt với cô ấy. Vậy nên tôi hỏi một câu hỏi hiển nhiên:

“Nhưng cô không thể dùng setter và getter sao? Tại sao lại làm các biến công khai?”

“Chính xác!” Avery lắp bắp.

“Sao phải bận tâm làm gì?” Jerry hỏi.

“Đúng thế,” Jasmine nói. “Đó chỉ là mã thêm vào, cái không cho cậu thêm bất cứ thứ gì.”

“Nhưng tôi nghĩ getter và setter là hướng đi đúng để cung cấp quyền truy cập tới các biến?” Tôi đáp trả.

“Getter và setter có thể hữu dụng với các biến mà mày thực sự muốn đóng gói.” Jerry nói, “nhưng mày không bắt buộc phải dùng chúng cho tất cả các biến. Thêm vào đó, dùng chúng cho tất cả các biến là mùi của code thối.”

“Chính xác.” Jasmine tươi cười đáp lại. “PeeYew! Không có gì tệ hơn là một lớp có một getter và setter cho mỗi biến, và không có một phương thức nào khác!”

“Nhưng đó không phải là cách cô định thực hiện một cấu trúc dữ liệu sao?” Tôi hỏi.

“Không, không hề.” “Jasmine đáp. “Một lớp có thể đóng hai vai trò trong Java. Đầu tiên là giống như một đối tượng, trong trường hợp này chúng ta thường tao các biến nội bộ và cung cấp một vài getter và setter cho các biến quan trọng nhất. Vai trò còn lại của lớp là được coi như một cấu trúc dữ liệu, lúc này chúng ta tạo các biến công khai và không cung cấp các phương thức. Kiểm thử fixture là cấu trúc dữ liệu căn bản với một vài phương thức để di chuyển qua lại giữa FitNesse và ứng dụng dưới dạng kiểm thử.”

“Đúng thế.” Jerry nói. “Một lớp mà có getter và setter nhưng không có phương thức nào khác thì hoàn toàn không phải là một đối tượng. Nó chỉ là một cấu trúc dữ liệu ai đó đã lãng phí thời gian và công sức cố gắng đóng gói nó. Việc đóng gói đó rốt cuộc chẳng mang lại điều gì. Cấu trúc dữ liệu, theo định nghĩa, không được đóng gói.”

“Đồng ý.” Jasmine nói. “Các đối tượng được đóng gói bởi vì chúng chứa các phương thức có thể thay đổi biến. Chúng không có nhiều getter và setter vì dữ liệu trong đối tượng vẫn ẩn trong hầu hết các phần.”

“Chính xác.” Jerry nói.

“Đúng thế.” Jasmine nói.

Và rồi cả hai bọn họ cười với nhau theo cách làm tôi muốn ói. Avery nhìn giống như hắn ta vừa ăn phải quả chanh vậy.