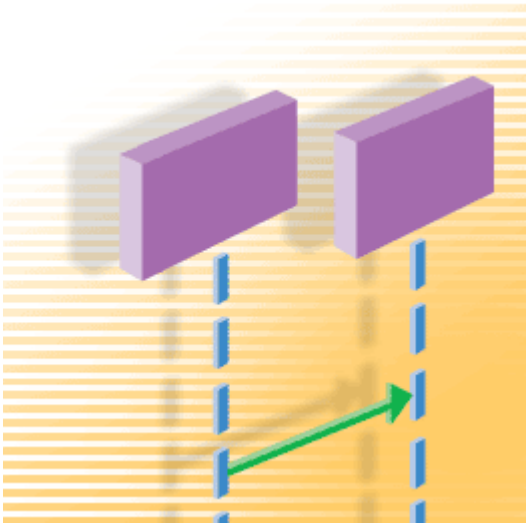
Căn bản về UML: Biểu đồ trình tự của UML

Mức độ: Nhập môn

Donald Bell, Chuyên gia IT, IBM

28 08 2009

Từ loạt bài viết The Rational Edge, UML căn bản, trên các biểu đồ cần thiết trong Unified Modeling Language (Mô hình hoá ngôn ngữ thống nhất), bài viết này đưa ra giới thiệu chi tiết đối với một biểu đồ trình tự. Nó cũng giới thiệu một vài thành phần chú thích mới trong đặc tả UML 2.0 gần đây.



Hiện đang là tháng Hai, và bây giờ bạn có thể đọc, hoặc nghe mọi người nói về việc tạo ra thay đổi đối với UML 2.0 - đặc tả mới cho UML chứa nhiều sự cải tiến. Căn cứ sự quan trọng của các đặc tả mới, chúng ta cũng thay đổi nền tảng của các loạt bài viết, chuyển sự quan tâm của chúng ta từ OMG's UML 1.4 Specification (đặc tả UML 1.4 của OMG) tới OMG's Adopted 2.0 Draft Specification of UML (Đặc tả dự thảo UML 2.0 được thông qua của OMG) (a.k.a. UML 2). Tôi không muốn thay đổi sự nhấn mạnh từ 1.4 tới 2.0 trong phần giữa của một loạt bài viết, nhưng Dự thảo Đặc tả UML 2.0 (UML 2.0 Draft Specification) là một bước tiến quan trọng, và tôi cảm thấy sự cần thiết để nói rộng hơn vấn đề này.

Đã có một vài nguyên nhân mà OMG phát triển UML. Nguyên nhân chính là họ muốn các mô hình UML đủ khả năng phân phối Kiến trúc định hướng Mô hình (Model Driven Architecture) (MDA), có nghĩa rằng UML phải hoạt động hơn là một chú thích định hướng mô hình. Tương tự, chú thích UML 1.x được tập hợp tại các thời điểm sẽ khó khăn để áp dụng vào các ứng dụng lớn hơn. Hơn nữa, các thành phần chú thích cần thiết nâng cấp để tạo các biểu đồ có thể đọc được. (Ví dụ, mô hình hoá dòng UML 1.x bị phức tạp và tại các thời điểm có thể. Các thay đổi đối với tập hợp chú thích của mô hình trình tự trong UML 2 đã tạo ra sự cải tiến rất mạnh trong mô hình hoá logic trong các trình tự).

Chú ý từ ngữ trong trình bày của tôi phía trên: "Adopted 2.0 Draft Specification of UML". Sự thật là đặc tả vẫn trong trạng thái dự thảo, nhưng quan trọng là Đặc tả Dự thảo đã được thông qua bởi OMG, một tập đoàn không thông qua các tiêu chuẩn mới nếu chúng không trở nên khá vững chắc. Sẽ có một vài thay đổi đối với đặc tả trước khi UML 2 được thông qua hoàn toàn, nhưng những thay đổi này nên ở mức tối thiểu. Sự thay đổi chính ở trong nội bộ của UML - bao gồm các tính năng được sử dụng điển hình bởi các công ty phần mềm sử dụng các công cụ UML.

Mục đích chính của bài viết này là tiếp tục sự tập trung của chúng ta vào các biểu đồ UML; tháng này, chúng ta sẽ xem kỹ hơn biểu đồ trình tự. Một lần nữa, lưu ý rằng, các ví dụ được đưa ra dưới đây dựa trên đặc tả UML 2.

Mục đích của biểu đồ

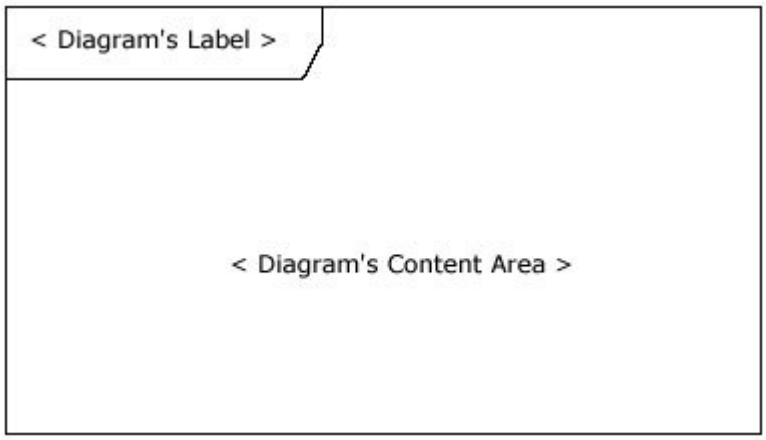
Biểu đồ trình tự được sử dụng chủ yếu để thể hiện mối tương tác giữa các đối tượng trong thứ tự trình tự mà các mối tương quan này xảy ra. Giống như biểu đồ lớp, các chuyên viên phát triển thường nghĩ các biểu đồ trình tự là dành riêng đối với họ. Tuy nhiên, nhân viên kinh doanh của một tổ chức có thể tìm ra biểu đồ trình tự hiệu quả để trao đổi các công việc gần đây hoạt động thế nào bằng cách trình bày các đối tượng công việc đa dạng tác động với nhau thế nào. Bên cạnh tập hợp tài liệu các sự kiện của một tổ chức, một biểu đồ trình tự ở cấp độ kinh doanh có thể được sử dụng như là một tài liệu cần thiết để trao đổi các yêu cầu cho việc thực thi hệ thống trong tương lai. Trong giai đoạn các yêu cầu của một dự án, các chuyên viên phân tích có thể tận dụng các trường hợp tới mức độ cao hơn bằng cách đưa ra một mức cải tiến chính thức hơn. Khi xảy ra như vậy, sử dụng các tình huống được cải tiến vào một hoặc nhiều các biểu đồ trình tự.

Nhân viên kỹ thuật của một tổ chức có thể tìm các biểu đồ trình tự hiệu quả qua việc chứng minh một hệ thống tương lai sẽ hoạt động thế nào. Trong giai đoạn thiết kế, các chuyên viên kiến trúc và chuyên viên phát triển có thể sử dụng biểu đồ để ép các mối tương quan đối tượng của hệ thống, do đó lọc ra các thiết kế hệ thống tổng thể.

Một trong các sử dụng chính của biểu đồ trình tự là trong sự chuyển đổi từ các yêu cầu được thể hiện như là sử dụng các tình huống tới các cải tiến tiếp theo và mức chính thức hơn của sự cải tiến. Sử dụng các trường hợp được cải tiến thường xuyên hơn vào một hoặc nhiều biểu đồ trình tự. Ngoài các sử dụng trong việc thiết kế hệ thống mới, biểu đồ trình tự có thể được sử dụng để dẫn chứng các đối tượng trong một hệ thống đang tồn tại (gọi là "hợp lệ") gần đây tương tác thế nào. Sự dẫn chứng này rất hữu ích khi chuyển đổi một hệ thống tới nhân sự hoặc tổ chức khác.

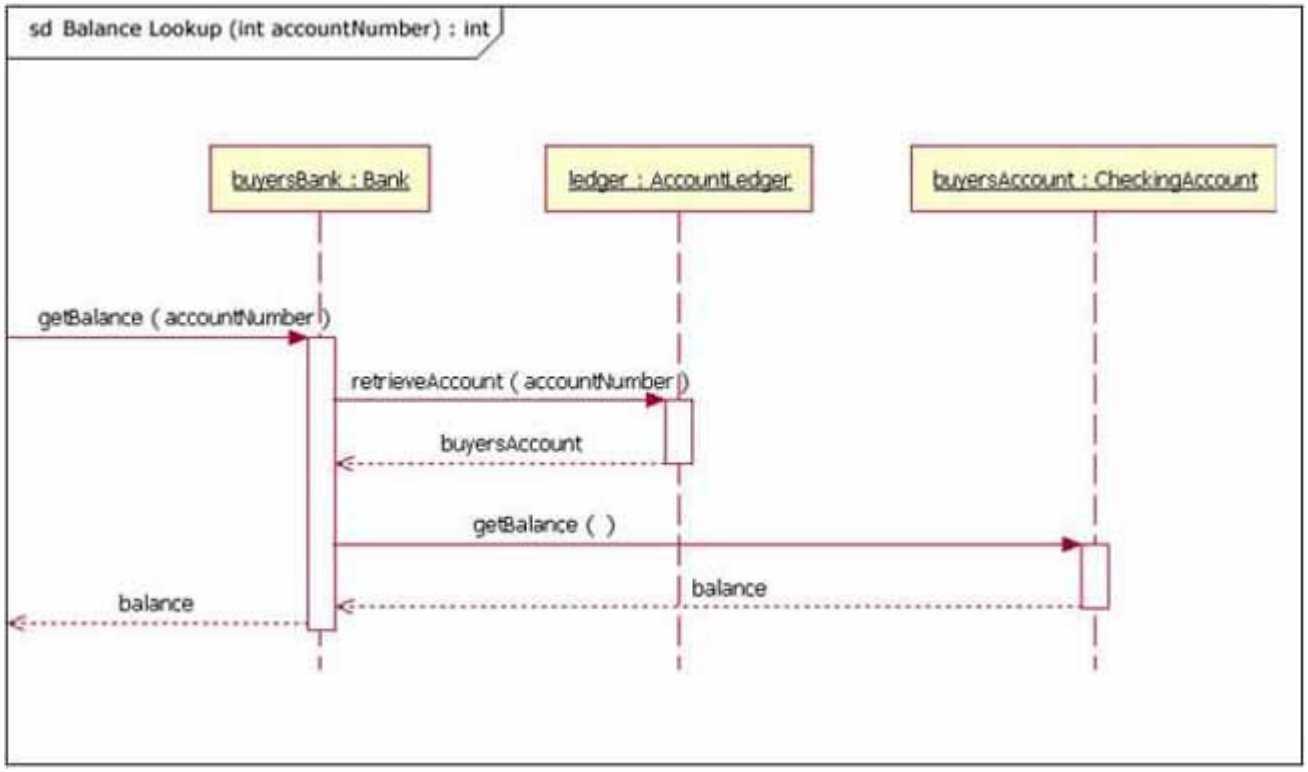
Chú thích

Do đây là bài viết đầu tiên trong loạt biểu đồ UML của tôi dựa trên UML 2, đầu tiên chúng ta cần thảo luận thêm về chú thích trong các biểu đồ UML 2, đặt tên một thành phần chú thích được gọi là một khung. Thành phần khung được sử dụng như là nền tảng cho các thành phần biểu đồ khác trong UML 2, nhưng nơi đầu tiên hầu hết mọi người bắt gặp một thành phần khung là ranh giới đồ hoạ của một biểu đồ. Một thành phần khung cung cấp một vị trí đồng nhất đối với một nhãn biểu đồ, trong khi cung cấp một ranh giới đồ hoạ cho biểu đồ. Thành phần khung là tuỳ chọn trong các biểu đồ UML; như bạn thấy trong Hình 1 và 2, nhãn của biểu đồ được đặt ở góc phía trên bên trái nơi Tôi sẽ gọi là "tên hộp" của khung, một kiểu của hình chữ nhật có dạng nếp quăn ở góc, và biểu đồ UML thực được định nghĩa trong phần thân của hình chữ nhật lớn hơn kèm theo.



Hình 1: Một thành phần khung UML 2 rỗng

Ngoài việc cung cấp một ranh giới thực, thành phần khung cũng có một chức năng hoạt động quan trọng trong biểu đồ, miêu tả các mối tương tác, như biểu đồ trình tự. Trong các thông điệp vào và ra của biểu đồ trình tự (a.k.a. interactions) đối với một trình tự có thể mô hình hoá bằng cách kết nối các thông điệp tới ranh giới của thành phần khung (như trong Hình 2). Điều này được trình bày chi tiết hơn trong phần "Trên mức cơ bản" dưới đây.



Hình 2: Một biểu đồ trình tự có các thông điệp vào và ra.

Lưu ý rằng trong Hình 2, nhãn của biểu đồ bắt đầu bằng các chữ "sd", dành cho Biểu đồ trình tự. Khi sử dụng một thành phần khung để kèm theo một biểu đồ, nhãn của biểu đồ cần theo định dạng của:

```
Diagram Type Diagram Name
```

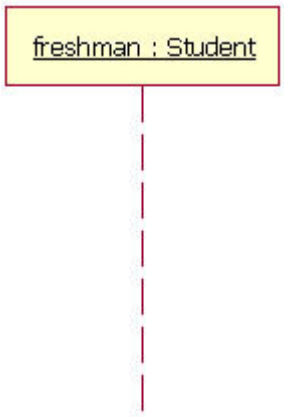
Đặc tả UML cung cấp các giá trị ký tự đặc thù đối với các loại biểu đồ (ví dụ: sd = Sequence Diagram (Biểu đồ trình tự), activity = Activity Diagram (Biểu đồ hoạt động), và use case = Use Case Diagram (Biểu đồ sử dụng tình huống)).

Cơ bản

Mục đích chính của một biểu đồ trình tự là để xác định các trình tự sự kiện dẫn đến một số kết quả mong muốn. Sự tập trung ít hơn vào chính các thông điệp và nhiều hơn vào thứ tự các thông điệp xảy ra; tuy nhiên hầu hết các biểu đồ trình tự sẽ truyền đạt những gì các thông điệp được gửi giữa các đối tượng của một hệ thống cũng như thứ tự trong đó chúng xảy ra. Biểu đồ truyền tải thông tin này theo chiều dọc và chiều ngang: Chiều đứng, từ trên xuống, thể hiện trình tự thời gian của các thông điệp/yêu cầu khi chúng xảy ra, và chiều ngang, từ trái sang phải, thể hiện các ví dụ đối tượng mà thông điệp được gửi đến.

Các sự trợ giúp

Khi vẽ một biểu đồ trình tự, các thành phần của sự trợ giúp (lifeline) được đặt chạy theo phía trên của biểu đồ. Các sự trợ giúp đại diện cho các vai trò hoặc các ví dụ đối tượng mà tham gia vào trình tự đang được mô hình hoá.[1] Các sự trợ giúp được vẽ như là một hộp với một đường ngắt quãng giảm dần từ trung tâm của đường viền phía dưới (Hình 3). Tên của sự trợ giúp được đặt bên trong hộp.



Hình 3: Ví dụ của lớp Sinh viên được sử dụng trong sự trợ giúp có tên ví dụ là freshman (sinh viên năm thứ nhất)

UML đại diện cho cách đặt tên một sự trợ giúp theo định dạng của:

```
Instance Name : Class Name
```

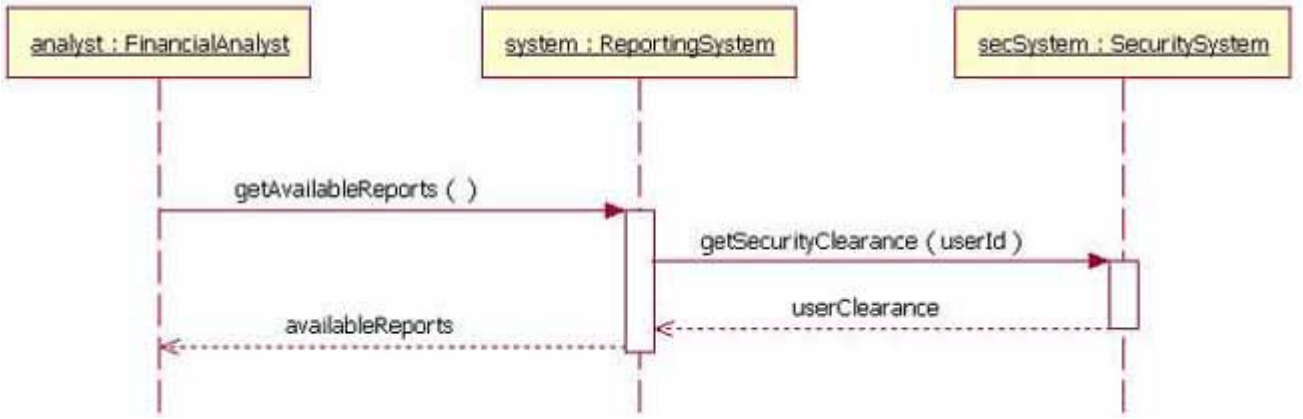
Trong ví dụ thể hiện trong Hình 3, sự trợ giúp đại diện một ví dụ của lớp Sinh viên, có ví dụ tên là freshman. Chú ý rằng, tại đây, tên sự trợ giúp được gạch chân. Khi một đường gạch chân được sử dụng, có nghĩa rằng sự trợ giúp đại diện một tình huống cụ thể của một lớp trong một biểu đồ trình tự, và không phải một loại cụ thể của tình huống (ví dụ: một vai trò). Trong một bài viết sau này, chúng ta sẽ xem xét cấu trúc mô hình hoá. Hiện tại, chỉ quan sát rằng các biểu đồ trình tự có thể bao gồm các vai trò (như *người mua* và *người bán*) mà không chỉ cụ thể ai đóng vai trò này (như **Bill** và **Fred**). Điều này cho phép biểu đồ sử dụng lại trong các ngữ cảnh khác nhau. Đơn giản đặt các tên ví dụ trong các biểu đồ trình tự được gạch chân; còn tên các vai trò thì không.

Ví dụ sự trợ giúp của chúng ta trong Hình 3 là một đối tượng được đặt tên, nhưng không phải tất cả các sự trợ giúp đại diện các đối tượng được đặt tên. Thay vì một sự trợ giúp có thể được để đại diện một ẩn danh hoặc một ví dụ không được đặt tên. Khi mô hình hoá một ví dụ không được đặt tên trên một biểu đồ trình tự, tên của sự trợ giúp theo mẫu tương tự như một ví dụ được đặt tên; nhưng thay vì cung cấp tên một ví dụ, phần tên của sự trợ giúp được để trống. Đề cập đến Hình 3 lần nữa, nếu sự trợ giúp đại diện một ví dụ ẩn danh của lớp Sinh viên, sự trợ giúp sẽ là: "Sinh viên". Cũng vậy, do biểu đồ trình tự được sử dụng trong giai đoạn thiết kế của dự án, điều đó hoàn toàn xác đáng để có một đối tượng có loại chưa được chỉ định: ví dụ, "freshman".

Các thông điệp

Thông điệp đầu tiên của một biểu đồ trình tự luôn bắt đầu phía trên cùng và thường được đặt phía bên trái của biểu đồ cho dễ đọc. Các thông điệp tiếp theo sau đó được dần dần thêm vào biểu đồ thấp hơn các thông điệp trước đó.

Để hiển thị một đối tượng (ví dụ: sự trợ giúp) đang gửi một thông điệp tới một đối tượng khác, bạn vẽ một đường để nhận đối tượng với mũi tên đặc ở phần đầu (nếu một hoạt động tín hiệu đồng bộ) hoặc với một mũi tên gắn ở phần đầu (nếu một dấu hiệu thiếu đồng bộ). Tên thông điệp/phương pháp được đặt trên dòng có hình mũi tên. Thông điệp được gửi tới đối tượng nhận sẽ đại diện một quá trình/phương pháp mà nhận thực thi lớp của đối tượng nhận đó. Trong ví dụ tại Hình 4, chuyên viên phân tích đối tượng tạo một yêu cầu tới đối tượng hệ thống nơi là một trường hợp của ReportingSystem. Chuyên viên phân tích đối tượng sẽ yêu cầu phương pháp getAvailableReports của đối tượng hệ thống. Sau đó, đối tượng hệ thống yêu cầu phương pháp getSecurityClearance với luận cứ của userld tại các đối tượng secSystem, nơi là một dạng của SecuritySystem.[2]



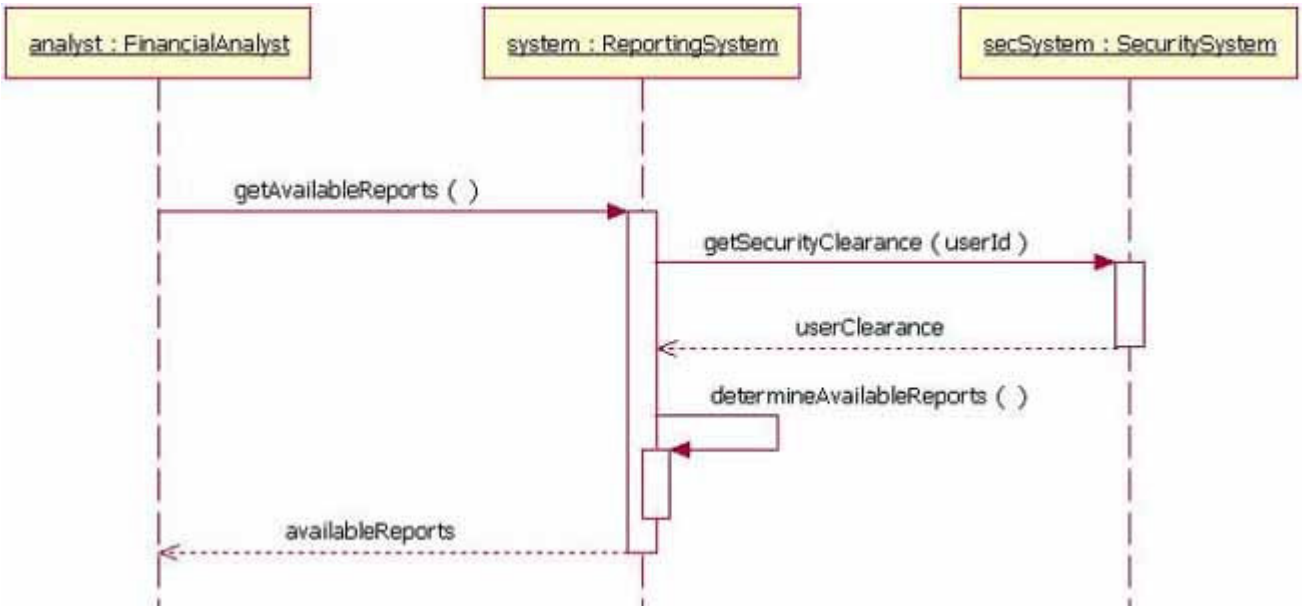
Hình 4: Một ví dụ về thông điệp được gửi đi giữa các đối tượng.

Ngoài việc chỉ đưa thông điệp yêu cầu đối với biểu đồ trình tự, biểu đồ Hình 4 bao gồm các thông điệp phản hồi. Các thông điệp phản hồi này là tuỳ chọn; một thông điệp phản hồi được vẽ bằng một đường nét đứt có đầu hình mũi tên, phía sau sự trợ giúp ban đầu, và phía trên đường nét đứt này bạn đặt giá trị phản hồi từ phương thức hoạt động. Trong Hình 4, đối tượng the secSystem phản hồi userClearance tới đối tượng hệ thống khi phương pháp getSecurityClearance được yêu cầu. Đối tượng hệ thống phản hồi availableReports khi phương pháp getAvailableReports được yêu cầu.

Một lần nữa, các thông điệp phản hồi là một mảng tuỳ chọn của một biểu đồ trình tự. Việc dùng thông điệp phản hồi phụ thuộc vào mức độ chi tiết/thiếu hụt đang được mô hình hoá. Các thông điệp phản hồi rất hữu dụng nếu các chi tiết tốt hơn được yêu cầu, ngược lại, thông điệp dẫn chứng là đầy đủ. Cá nhân tôi muốn gộp các thông điệp phản hồi bất cứ khi nào một giá trị sẽ được phản hồi, vì tôi thấy rằng các chi tiết phụ tạo ra một biểu đồ trình tự dễ đọc hơn.

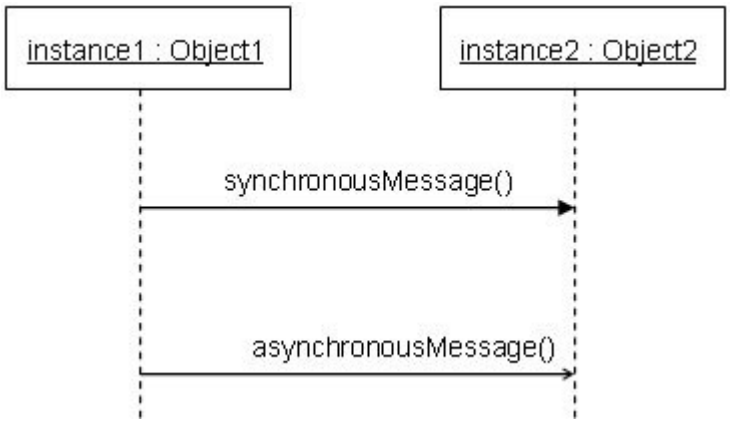
Khi mô hình hoá một biểu đồ trình tự, sẽ có những lúc một đối tượng cần gửi một thông điệp đến chính nó. Khi nào thì một đối tượng yêu cầu chính nó? Một người nhìn cách thuần tuý sẽ lập luận rằng một đối tượng sẽ không bao giờ gửi một thông điệp tới chính nó. Tuy nhiên, mô hình hoá một đối tượng gửi một thông điệp tới chính nó có thể có tác dụng trong một số trường hợp. Ví dụ, Hình 5 là một phiên bản nâng cấp của Hình 4. Phiên bản Hình 5 thể hiện đối tượng hệ thống yêu cầu phương pháp determineAvailableReports của nó. Bằng cách đưa ra hệ thống gửi tới chính nó thông điệp "determineAvailableReports", mô hình đã gây sự chú ý một thực tế rằng quá trình này xảy ra trong đối tượng hệ thống.

Để vẽ một đối tượng yêu cầu tới chính nó, bạn vẽ một thông điệp như cách bạn làm thông thường, nhưng thay vì kết nối nó tới một đối tượng khác, bạn kết nối thông điệp ngược lại chính đối tượng đó.



Hình 5: Đối tượng hệ thống yêu cầu phương pháp determineAvailableReports

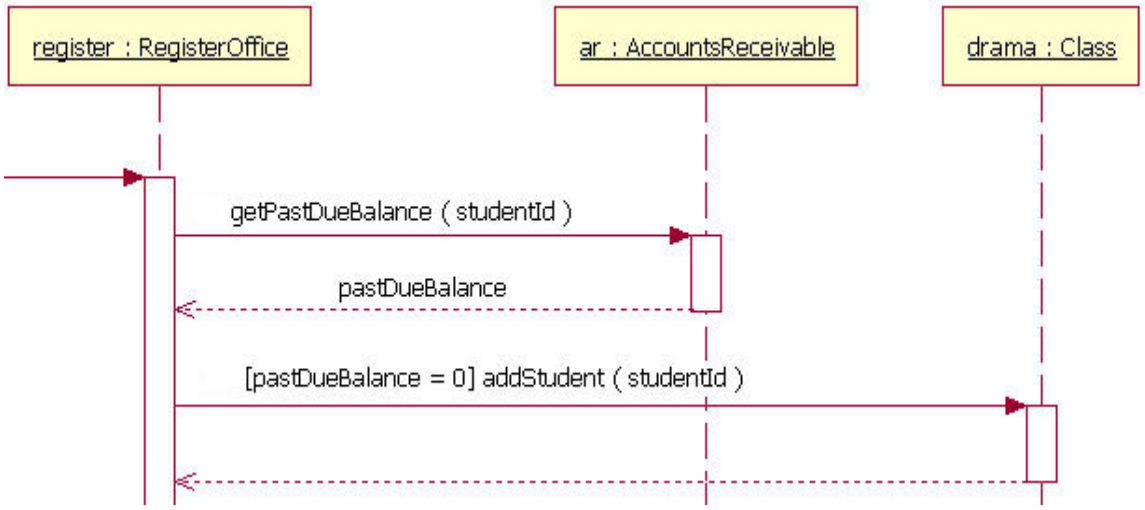
Các thông điệp ví dụ trong Hình 5 biểu hiện các thông điệp đồng bộ, tuy nhiên trong các biểu đồ trình tự bạn có thể mô hình các thông điệp không đồng bộ. Một thông điệp không đồng bộ được vẽ tương tự như đồng bộ, nhưng dòng của thông điệp được vẽ với một mũi tên gắn ở đầu, như trong Hình 6.



Hình 6: Một mảng của biểu đồ trình tự thể hiện một thông điệp không đồng bộ đang được gửi tới trường hợp 2

Trạng thái bảo vệ

Khi mô hình hoá các mối tương tác đối tượng, sẽ có lúc khi một điều kiện phải được thoả mãn để một thông điệp được gửi tới đối tượng. Trạng thái bảo vệ được sử dụng trong các biểu đồ UML để kiểm soát sự lưu thông. Tới đây, tôi sẽ thảo luận các trạng thái bảo vệ trong cả UML 1.x cũng như UML 2.0. Trong UML 1.x, một trạng thái bảo vệ chỉ có thể được thiết lập đối với một thông điệp riêng lẻ. Để vẽ một trạng thái bảo vệ trên một biểu đồ trình tự trong UML 1.x, bạn đặt thành phần trạng thái bảo vệ phía trên đường thông điệp đang được bảo vệ và phía trước tên của thông điệp. Hình 7 thể hiện một phần của biểu đồ trình tự với một trạng thái bảo vệ trên phương pháp thông điệp addStudent.



Hình 7: Một mảng của biểu đồ trình tự UML 1.x trong đó thông điệp addStudent có trạng thái bảo vệ

Trong Hình 7, trạng thái bảo vệ là dòng "[pastDueBalance = 0]". Bằng cách bảo vệ thông điệp này, thông điệp addStudent sẽ chỉ được gửi nếu hệ thống các tài khoản phải thu đưa ra một số dư tài khoản bằng không (0). Chú thích của một trạng thái bảo vệ rất đơn giản, định dạng là:

```
[Boolean Test]
```

Ví dụ,

```
[pastDueBalance = 0]
```

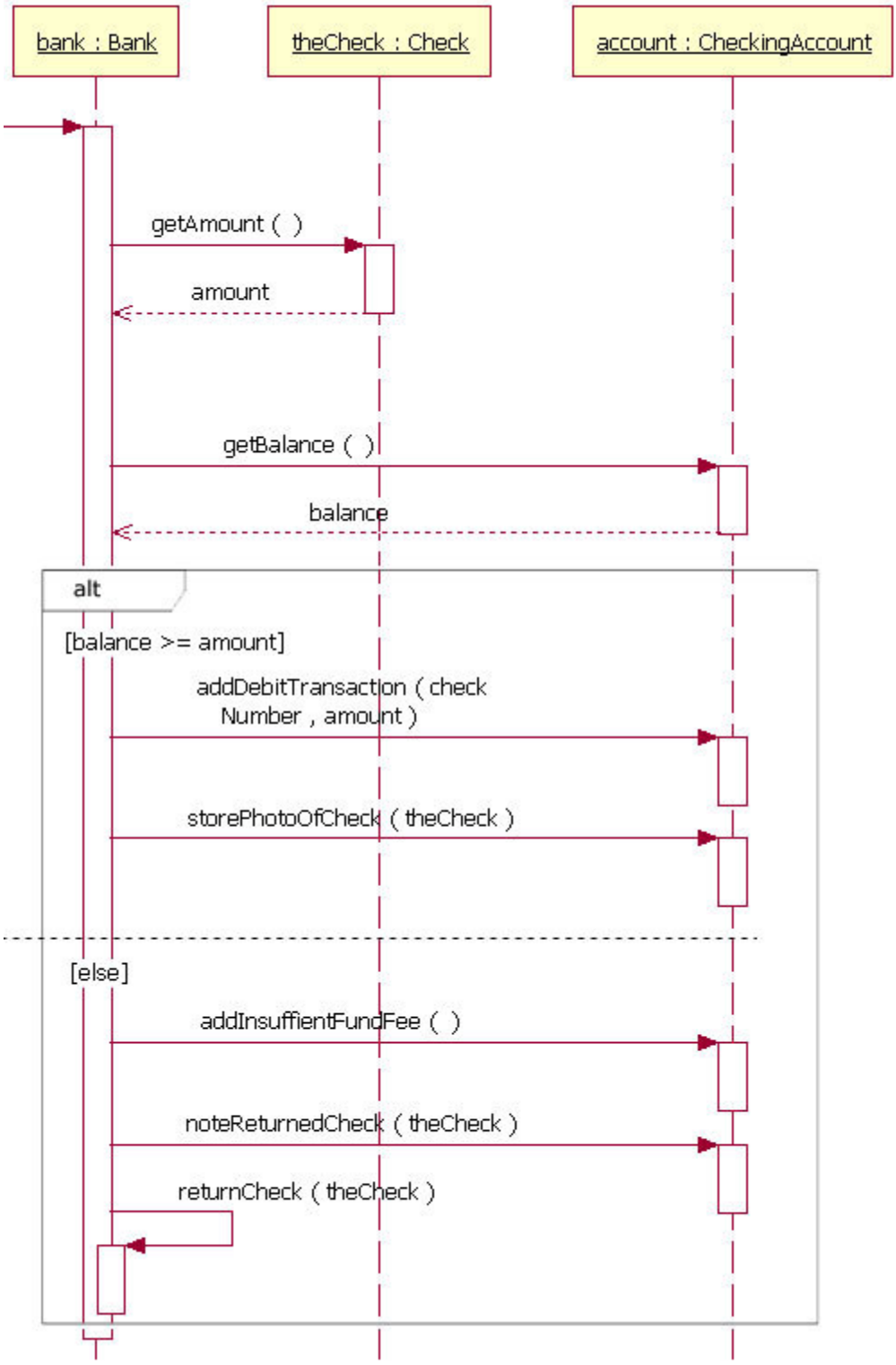
Các mảng kết hợp (các sự thay thế, tuỳ chọn, và vòng lặp)

Tuy nhiên, trong hầu hết các biểu đồ trình tự, trạng thái bảo vệ theo dòng "in-line" UML 1.x không đủ để có được tính logic theo yêu cầu đối với một trình tự được mô hình. Sự thiếu sót chức năng là một vấn đề của UML 1.x. UML 2 đã chỉ ra vấn đề này bằng cách loại bỏ trạng thái bảo vệ theo dòng "in-line" và thêm một thành phần chú thích gọi là Mảng được kết hợp "Combined Fragment". Một mảng được kết hợp được sử dụng để nhóm các gói thông điệp với nhau, thể hiện điều kiện trong một biểu đồ trình tự. Đặc tả UML 2 xác định 11 loại tương tác của các phần kết hợp. Ba trong số mười một loại này sẽ được đề cập tại đây, trong phần "Cơ bản", hai loại khác sẽ được nói đến trong phần "Trên mức cơ bản", và sáu phần còn lại tôi sẽ nói trong một bài viết sau. (Nào, đây là một bài viết, chứ không phải một cuốn sách. Tôi muốn bạn kết thúc phần này trong một ngày!)

Các sự thay thế

Các sự thay thế được sử dụng để tìm kiếm một sự lựa chọn tổng thể có thể thay thế lẫn nhau giữa hai hoặc nhiều trình tự thông điệp.[3] Các sự thay thế cho phép mô hình hoá lập luận cổ điển "nếu thì" (ví dụ: **nếu (if)** Tôi mua 3 thứ, **thì (then),** Tôi được giảm 20% giá tiền; **hoặc (or)** Tôi được giảm 10% giá tiền).

Như bạn chú ý trong Hình 8, một thành phần kết hợp khác được vẽ bằng một khung. Từ "alt" được đặt bên trong khung tên hộp. Hình chữ nhật lớn hơn sau đó được chia thành những gì mà UML 2 yêu cầu các toán hạng.[4] Các toán hạng được phân chia bởi một đường có nét gạch. Mỗi toán hạng được gán một trạng thái bảo vệ để kiểm tra, và trạng thái bảo vệ này được đặt hướng lên phần trên bên trái của toán hạng, phía trên của một sự trợ giúp (lifeline).[5] Nếu một trạng thái bảo vệ của toán hạng có giá trị "true", thì toán hạng đó là toán hạng để kiểm tra, theo dõi.



Hình 8: Một mảng biểu đồ trình tự chứa một phần kết hợp thay thế

Như ví dụ thể hiện một mảng kết hợp thay thế (alternative combination fragment), Hình 8 cho thấy trình tự bắt đầu tại đỉnh, cùng với đối tượng ngân hàng kiểm tra số liệu và số dư tài khoản. Tại đây, trong trình tự phần kết hợp thay thế sẽ tiếp tục. Do trạng thái bảo vệ "[balance >= amount]" (số dư >= số giá trị), nếu số dư tài khoản lớn hơn hoặc bằng số giá trị, thì trình tự tiếp tục cùng với đối tượng ngân hàng gửi đi thông điệp addDebitTransaction storePhotoOfCheck tới đối tượng tài khoản. Tuy nhiên, nếu số dư không lớn hơn hoặc bằng số giá trị, thì trình tự xử lý với đối tượng ngân hàng gửi đi thông điệp addInsuffientFundFee và noteReturnedCheck tới đối tượng tài khoản và thông điệp returnCheck tới chính nó. Trình tự thứ hai được yêu cầu khi số dư không lớn hơn hoặc bằng số giá trị vì trạng thái bảo vệ ""[else]" (hoặc). Trong các mảng kết hợp thay thể, trạng thái bảo vệ "[else]" không cần yêu cầu; và nếu một toán

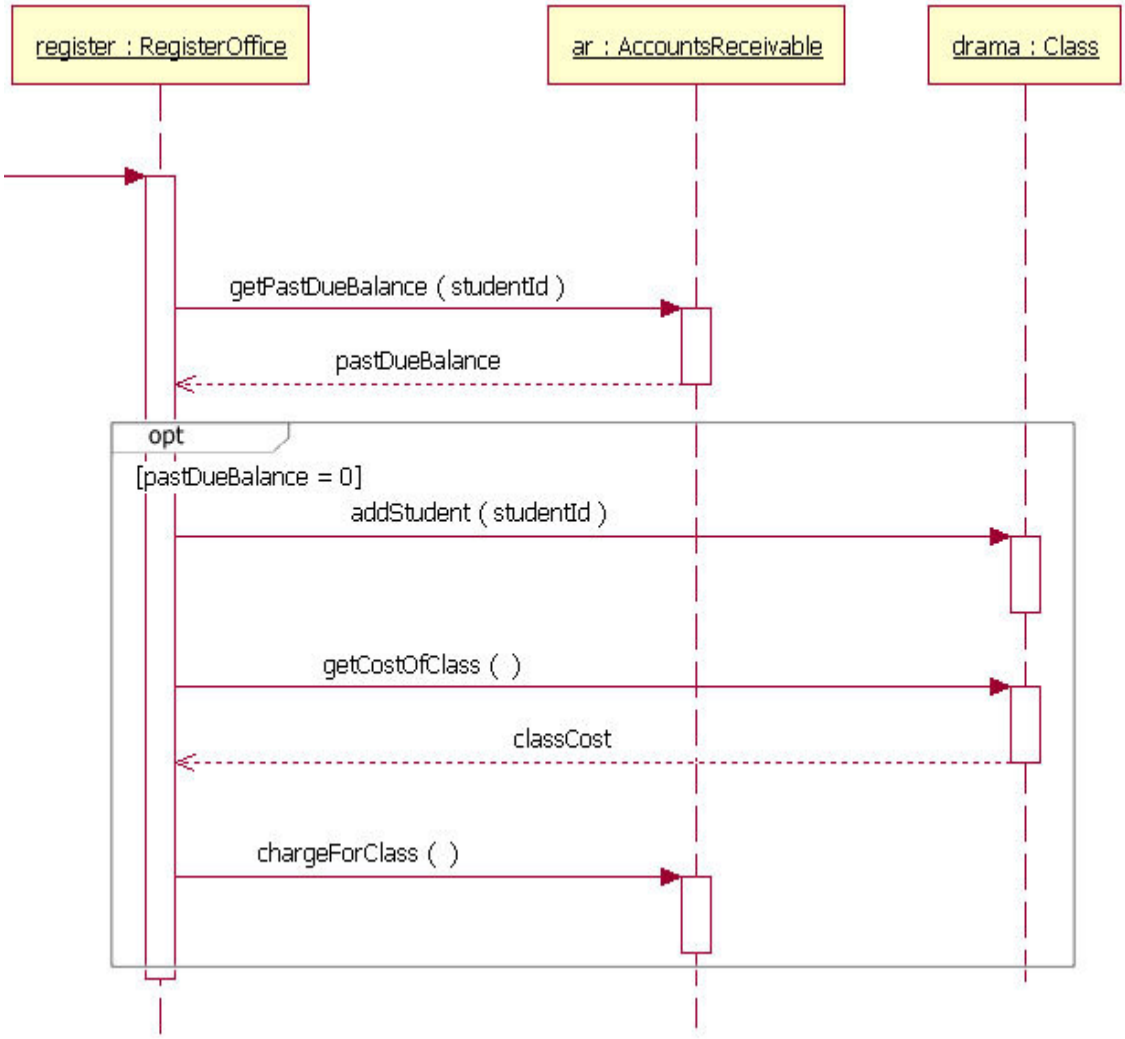
hạng không có một trạng thái bảo vệ rõ ràng, thì trạng thái bảo vệ "[else]" không được có trong giả định.

Các mảng kết hợp thay thế không giới hạn trong kiểm tra đơn giản ""if then else" (nếu thì). Sẽ có nhiều cách lựa chọn thay thế nếu cần. Nếu cần thêm các sự lựa chọn thay thế, bạn chỉ phải thêm một toán hạng vào hình chữ nhật với các thông điệp và trạng thái bảo vệ của trình tự.

Sự tuỳ chọn

Mảng kết hợp tuỳ chọn được sử dụng để mô hình một trình tự sẽ xảy ra, giả sử trong điều kiện nhất định; nếu không, trình tự sẽ không xảy ra. Một tuỳ chọn được sử dụng để mô hình một mệnh đề đơn giản "if then" (nếu thì) (ví dụ: nếu có ít hơn năm cái bánh trên kệ, thì làm thêm hai tá bánh nữa).

Chú thích của mảng kết hợp tuỳ chọn tương tự đối với mảng kết hợp thay thế, ngoại trừ nó chỉ có một toán hạng và không bao giờ có trạng thái bảo vệ "else" (nó chỉ không có ý nghĩa tại đây). Để vẽ một kết hợp tuỳ chọn bạn vẽ một khung. Chữ "opt" được đặt trong khung tên hộp, và trong khu vực nội dung của khung trạng thái bảo vệ được đặt hướng lên góc phía trên bên trái, bên trên của sự trợ giúp (lifeline). Sau đó trình tự tuỳ chọn của thông điệp được đặt trong phần còn lại khu vực nội dung của khung. Các thành phần này được minh hoạ trong Hình 9.



Hình 9: Một mảng biểu đồ trình tự bao gồm một mảng kết hợp tuỳ chọn.

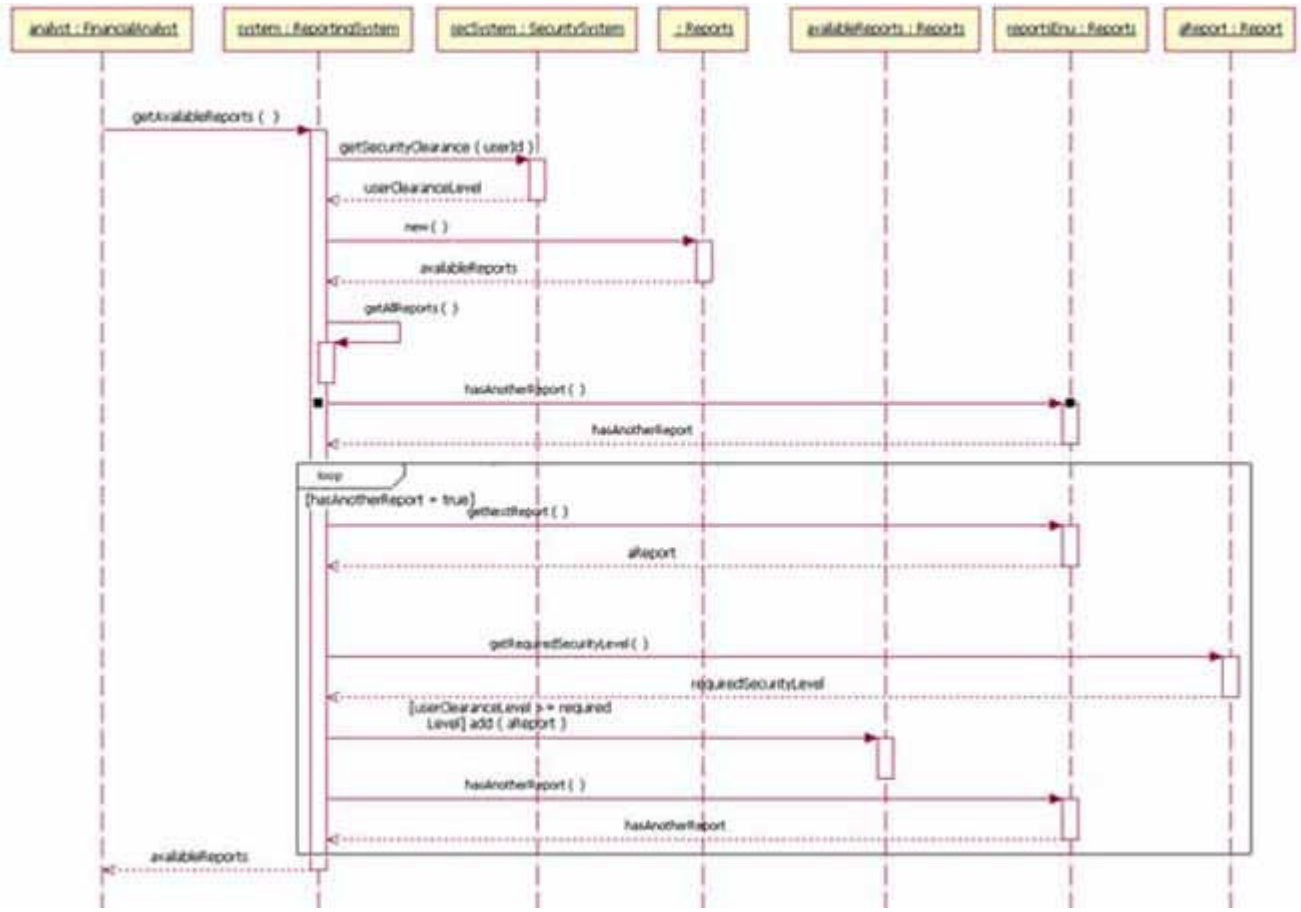
Đọc một mảng kết hợp tuỳ chọn là chuyện đơn giản. Hình 9 là hình vẽ lại mảng biểu đồ trình tự trong Hình 7, nhưng lúc này nó sử dụng một mảng kết hợp tuỳ chọn vì nhiều thông điệp cần được gửi đi nếu số dư tài khoản của sinh viên bằng không (0). Theo biểu đồ trình tự trong Hình 9, nếu số dư tài khoản của một sinh viên bằng không, thì thông điệp addStudent, getCostOfClass, và chargeForClass được gửi đi. Nếu số dư tài khoản của sinh viên không bằng không, thì trình tự sẽ ngừng gửi các thông điệp trong mảng kết hợp tuỳ chọn.

Ví dụ trong Hình 9, mảng biểu đồ trình tự bao gồm một trạng thái bảo vệ đối với sự tuỳ chọn; tuy nhiên trạng thái bảo vệ không phải là yếu tố bị yêu cầu. Ở mức cao hơn, những biểu đồ trình tự khó hiểu làm bạn không muốn xác định điều kiện của tuỳ chọn. Bạn có thể chỉ muốn xác định rằng các mảng là tuỳ chọn.

Vòng lặp

Đôi khi bạn sẽ cần mô hình một trình tự lặp đi lặp lại. Trong UML 2, mô hình hoá một biểu đồ lặp lại đã được cải tiến cùng với điều kiện của mảng kết hợp vòng lặp.

Mảng kết hợp vòng lặp rất đơn giản trong giao diện đối với mảng kết hợp tuỳ chọn. Bạn vẽ một khung, và trong phần tên hộp của khung, chữ "loop" (vòng lặp) được đặt vào. Trong khu vực nội dung của khung, trạng thái bảo vệ của vòng lặp [6] được đặt hướng lên góc phía trên bên trái, bên trên của sự trợ giúp (lifeline). Sau đó trình tự các thông điệp của vòng lặp được đặt trong khu vực nội dung còn lại của khung. Trong vòng lặp, một trạng thái bảo vệ có hai điều kiện đặc biệt được kiểm tra ngoài kiểm tra theo tiêu chuẩn Boolean. Các điều kiện bảo vệ đặc biệt là sự lặp lại tối thiểu được viết dưới dạng "minint = [the number]" (ví dụ: "minint = 1") và sự lặp lại tối đa được viết dưới dạng "maxint = [the number]" (ví dụ: "maxint = 5"). Với sự bảo vệ lặp lại tối thiểu, vòng lặp phải thực thi tối thiểu một số lần được chỉ định, trong khi với sự bảo vệ lặp lại tối đa, số thực thi vòng lặp không thể vượt quá một số chỉ định.



Hình 10: Ví dụ biểu đồ vòng lặp cùng với mảng kết hợp vòng lặp.
nhấn vào để xem to hơn

Vòng lặp thể hiện trong Hình 10 thực thi cho đến khi thông điệp hasAnotherReport của đối tượng reportsEnu đưa ra lỗi. Vòng lặp trong biểu đồ trình tự này sử dụng kiểm tra của Boolean để xác nhận xem trình tự lặp có chạy được không. Để đọc biểu đồ này, bạn phải bắt đầu từ phía trên, như cách thông thường. Khi bạn kết thúc kiểm tra mảng kết hợp vòng lặp để xem giá trị của hasAnotherReport có đúng hay không. Nếu giá trị của hasAnotherReport là đúng, thì trình tự chạy tiếp mảng vòng lặp. Bạn có thể xem tiếp các thông điệp trong vòng lặp như khi bạn làm với biểu đồ trình tự.

Trên mức cơ bản

Tôi đã nói về phần cơ bản của biểu đồ trình tự, phần cho phép bạn mô hình hầu hết các sự tương tác xảy ra trong một hệ thống thông thường. Phần tiếp theo sau đây sẽ đề cập nâng cao hơn các thành phần chú thích có thể được sử dụng trong một biểu đồ trình tự.

Đề cập một biểu đồ trình tự khác

Khi làm việc với các biểu đồ trình tự, các chuyên viên phát triển muốn sử dụng lại các biểu đồ trình tự hiện có trong các trình tự biểu đồ của họ.[7] Bắt đầu với UML 2, thành phần "Interaction Occurrence" (Xảy ra Tương tác) đã được giới thiệu. Các phần bổ sung của Xảy ra Tương tác đang bị tranh luận có phải là các sáng kiến quan trọng nhất trong mô hình hoá các tương tác UMl 2. Xảy ra Tương tác thêm các khả năng để tạo ra các biểu đồ trình tự nguyên thuỷ trong các biểu đồ trình tự phức tạp. Với những thứ này, bạn có thể kết hợp (tái sử dụng) các trình tự đơn giản hơn để tạo ra các trình tự phức tạp hơn. Có nghĩa rằng bạn có thể tách một trình tự hoàn chỉnh, có thể là phức tạp như là một đơn vị khái niệm đơn nhất.

Một thành phần Xảy ra Tương tác được vẽ sử dụng một khung. Chữ "ref" được đặt bên trong tên hộp của khung, và tên của biểu đồ trình tự đề cập được đặt bên trong khu vực nội dung của khung cùng với các tham số đối với biểu đồ trình tự. Chú thích của tên biểu đồ trình tự được đề cập theo mẫu sau:

```
tên bi□u đ□ trình t□ [(arguments)] [: return value]
```

Hai ví dụ:

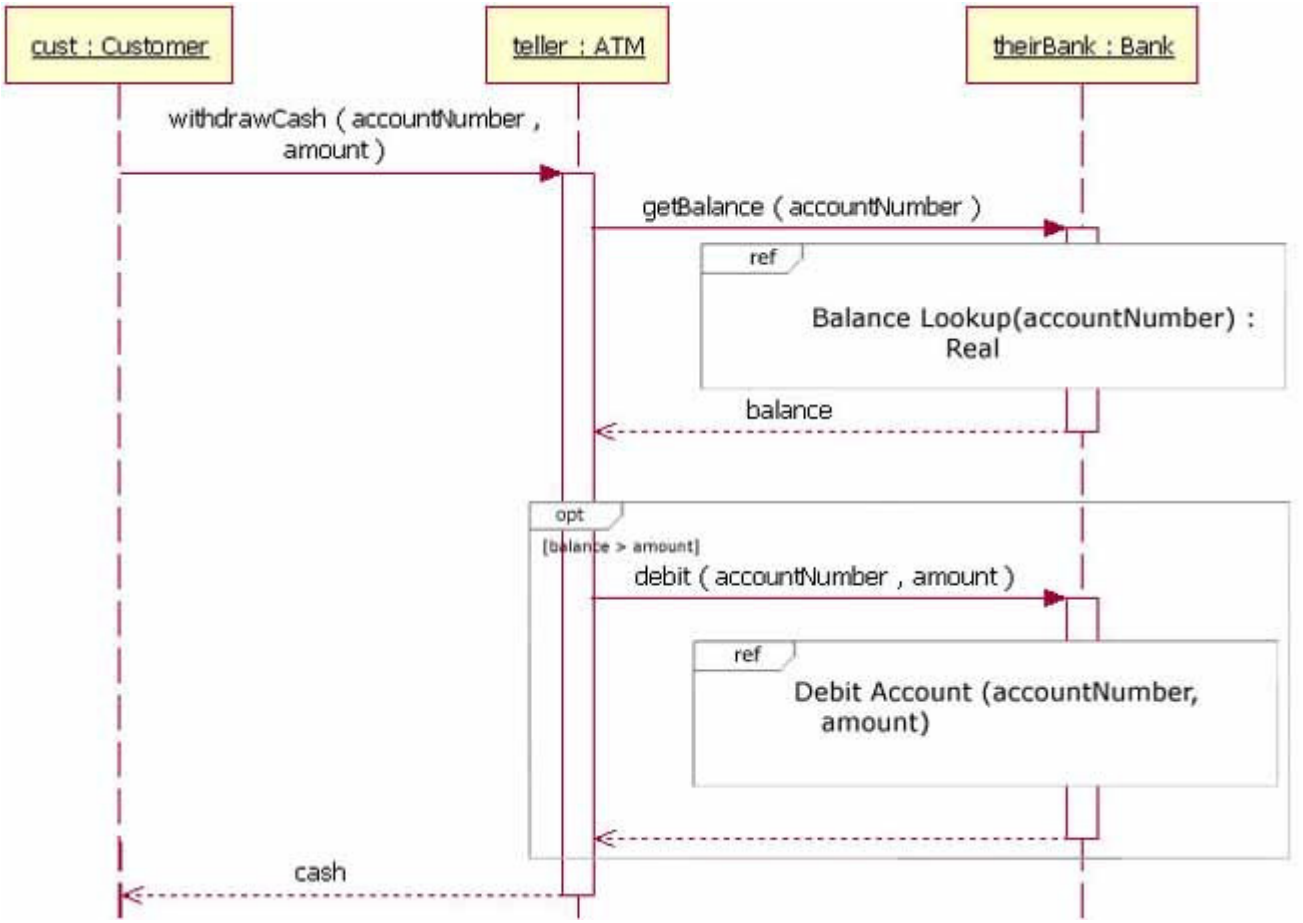
1. `Truy vấn Báo cáo tín dụng của người vay(ssn) : borrowerCreditReport`

hoặc

2. `Xử lý thẻ tín dụng (tên, số, hạn sử dụng, số tiền : 100)`

Trong ví dụ 1, các yêu cầu cú pháp về biểu đồ trình tự đòi hỏi Truy vấn Báo cáo tín dụng của người vay và chuyển nó các thông số ssn. Truy vấn Báo cáo Tín dụng của người vay phản hồi một báo cáo borrowerCreditReport.

Trong ví dụ 2, các yêu cầu cú pháp biểu đồ trình tự đòi hỏi Xử lý thẻ Tín dụng và chuyển nó các thông số như tên, số, hạn sử dụng và số tiền. Tuy nhiên, trong ví dụ 2, tham số số tiền là trị số của 100. Và do ví dụ 2 không có một giá trị phản hồi được đánh nhãn, trình tự không phản hồi một giá trị (giả sử, trình tự đang được mô hình không cần giá trị phản hồi).



Hình 11: Biểu đồ trình tự đề cập hai biểu đồ trình tự khác nhau.

Hình 11 thể hiện một biểu đồ trình tự đề cập đến các biểu đồ trình tự "Balance Lookup" (truy tìm số dư) và "Debit Account" (tài khoản nợ). Trình tự bắt đầu phía trên bên trái, cùng với khách hàng gửi đi một thông điệp tới đối tượng nhân viên giao dịch. Đối tượng nhân viên giao dịch gửi một thông điệp tới đối tượng ngân hàng của họ. Tại đây, biểu đồ trình tự Truy tìm số dư được yêu cầu, với số tài khoản được chuyển như là một tham số. Biểu đồ trình tự Truy tìm tham số đưa ra biến số dư. Sau đó điều kiện trạng thái bảo vệ của mảng kết hợp tuỳ chọn được kiểm tra để xác nhận số dư là lớn hơn khi số tiền đã có. Trong các trường hợp khi số dư lớn hơn số tiền, biểu đồ trình tự Tài khoản Nợ được yêu cầu, chuyển nó đến số tài khoản và số tiền như là các tham số. Sau đó, trình tự hoàn thành, thông điệp rút tiền sẽ phản hồi số tiền mặt tới khách hàng.

Rất quan trọng để lưu rằng trong Hình 1, sự trợ giúp của theirBank bị che khuất bởi xảy ra sự tương tác Truy vấn số dư. Do xảy ra sự tương tác che khuất các sự trợ giúp, có nghĩa rằng sự trợ giúp tài khoản theirBank được đề cập trong biểu đồ trình tự "Truy vấn số dư". Ngoài việc giấu sự trợ giúp trong việc xảy ra sự tương tác, UML 2 cũng chỉ ra rằng sự trợ giúp phải có theirBank trong trình tự "Truy vấn số dư" của chính nó.

Sẽ có những lúc khi bạn mô hình các biểu đồ trình tự mà một sự tương tác xảy ra sẽ trùng với sự trợ giúp *không* được đề cập trong sự xảy ra tương tác. Trong các trường hợp như vậy, sự trợ giúp được thể hiện như là một sự trợ giúp thông thường và không bị giấu bởi sự xảy ra tương tác chồng chéo.

Trong Hình 11, trình tự đề cập biểu đồ trình tự "Truy vấn số dư". Biểu đồ trình tự "Truy vấn số dư" được thể hiện trong Hình 12. Do ví dụ biểu đồ có các tham số và giá trị phản hồi, nhãn của nó được đặt trong phần tên của biểu đồ, theo mẫu cụ thể sau:

```
Tên bi□u đ□ Lo□i bi□u đ□ [(Parameter Type : Parameter Name)]:
```

```
[: Return Value Type] (loại giá trị phản hồi)
```

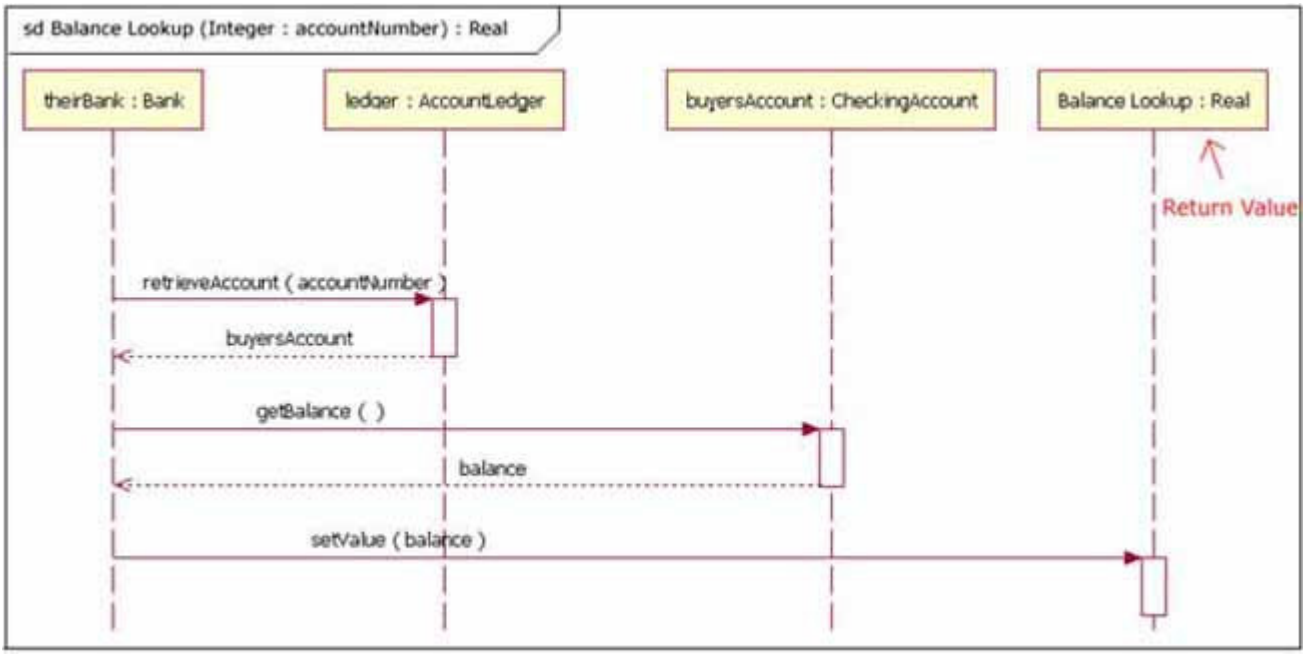
Hai ví dụ:

1. `SD Truy vấn số dư (Số nguyên: Số tài khoản): Số thực`

hoặc

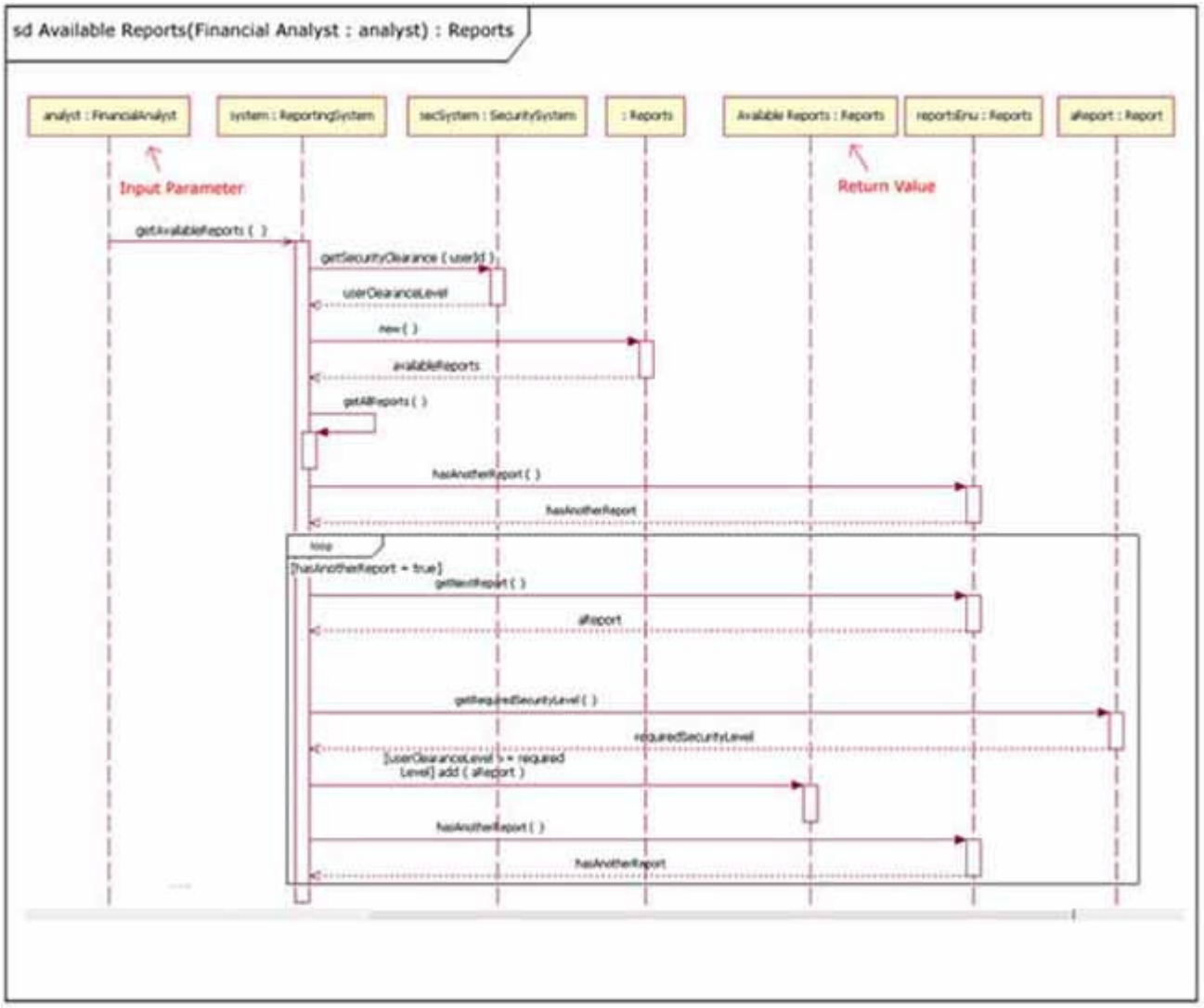
2. `SD Báo cáo hiện có (Chuyên viên phân tích Tài chính: chuyên viên phân tích): Báo cáo`

Hình 12 minh hoạ ví dụ 1, trong đó trình tự Truy vấn số dư sử dụng tham số số tài khoản (accountNumber) như là một biến trong trình tự, và biểu đồ trình tự thể hiện một đối tượng số thực được phản hồi. Trong các trường hợp như thế này, khi trình tự phản hồi một đối tượng, đối tượng phản hồi được giả sử một tên của biểu đồ trình tự.



Hình 12: Một biểu đồ trình tự lấy ra một số tài khoản và phản hồi một đối tượng số thực

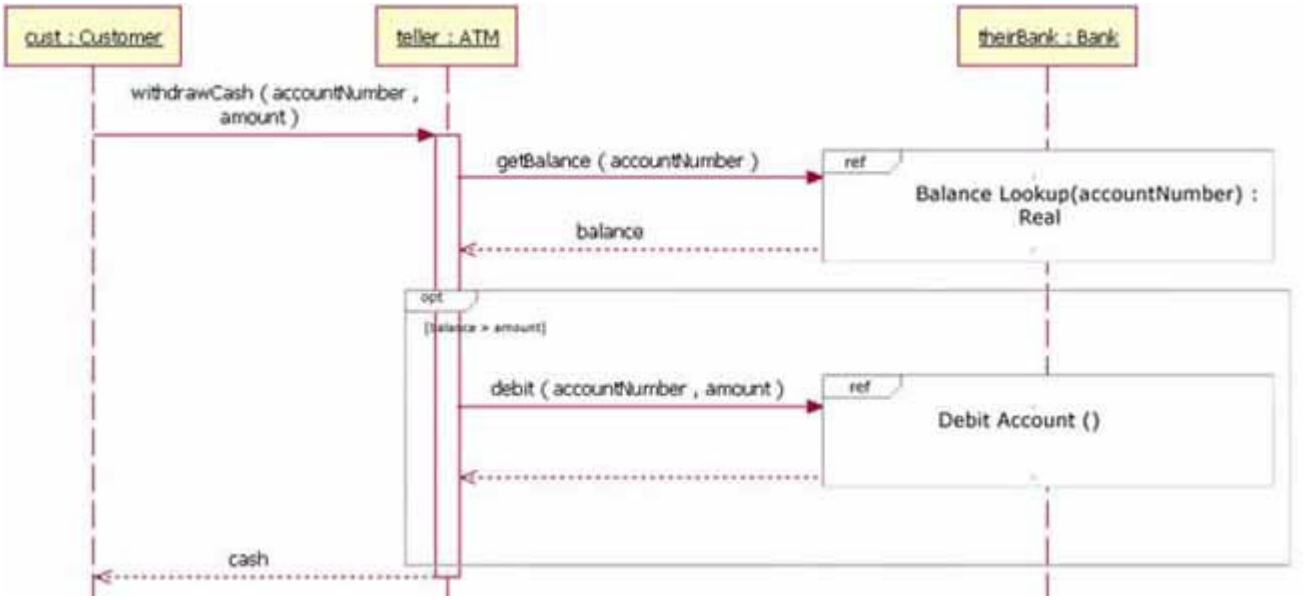
Hình 12 minh hoạ ví dụ 2, trong đó một trình tự lấy ra một tham số và phản hồi một đối tượng. Tuy nhiên, trong Hình 3 tham số được sử dụng trong sự tương tác của trình tự.



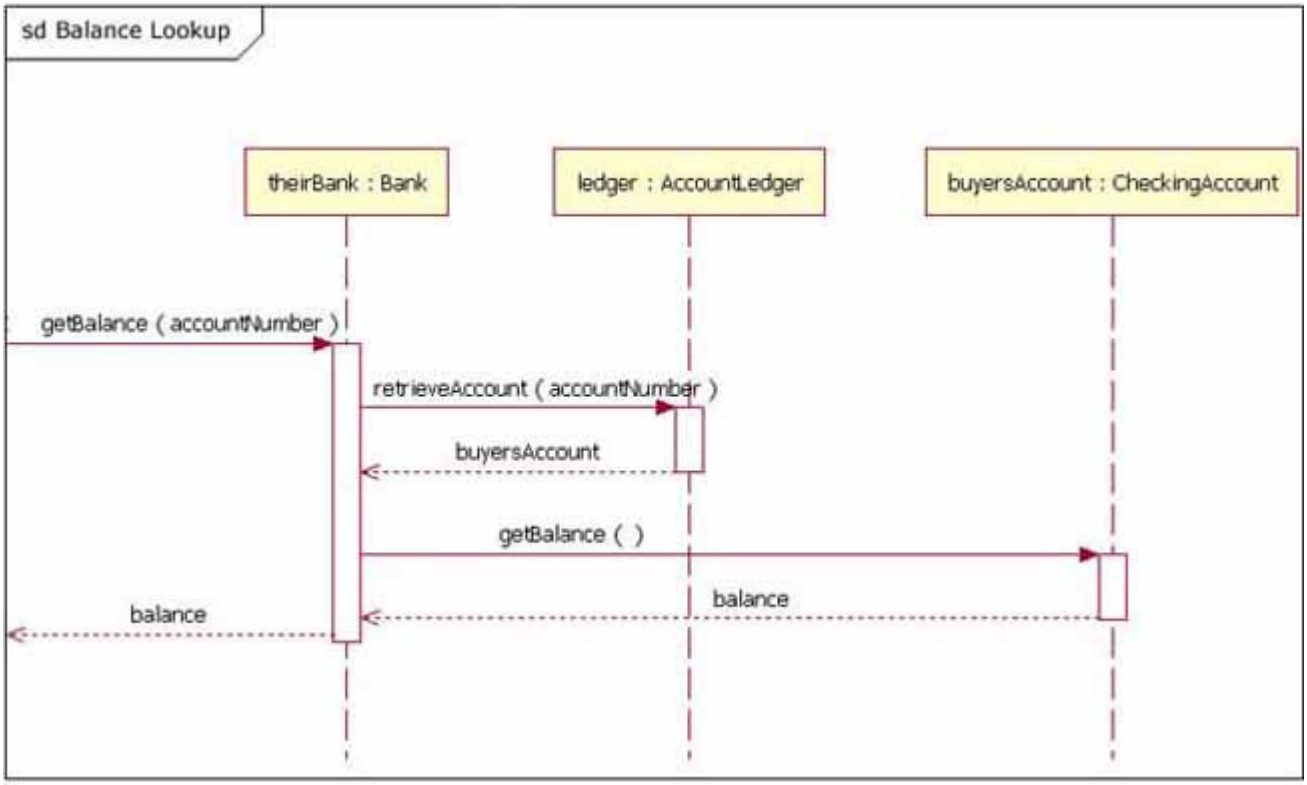
Hình 13: Một biểu đồ trình tự sử dụng tham số của nó trong sự tương tác của nó và phản hồi một đối tượng Báo cáo.
Nhấn vào để xem to hơn

Các cổng

Phần trước đã cho thấy cách để tham chiếu một biểu đồ trình tự khác bằng cách chuyển thông tin qua các tham số và giá trị phản hồi. Tuy nhiên, có một cách khác để chuyển thông tin giữa các biểu đồ trình tự. Các cổng có thể là một cách dễ dàng để mô hình việc chuyển thông tin giữa một biểu đồ trình tự và ngữ cảnh của nó. Một cổng đơn giản là một thông điệp, được minh hoạ bằng phần kết thúc nối với phần viền của khung của biểu đồ trình tự và phần kết thúc khác nối với một sự trợ giúp. Hình 11 và Hình 12 được vẽ lại sử dụng các cổng trở thành các Hình 14 và Hình 15. Biểu đồ trình tự trong Hình 15 có một cổng vào được gọi lại getBalance (số dư) mà nó lấy các tham số ra khỏi số tài khoản (accountNumber). Thông điệp getBalance là một cổng vào, vì nó là một đường mũi tên nối khung của biểu đồ với sự trợ giúp (lifeline). Biểu đồ trình tự cũng có một cổng ra mà phản hồi biến số dư. Cổng ra được biết, vì nó là một thông điệp phản hồi được nối từ một sự trợ giúp tới khung của biểu đồ bằng đường mũi tên nối tới khung.



Hình 14: Vẽ lại của Hình 11, sử dụng các cổng vào



Hình 15: Vẽ lại của Hình 12, sử dụng các cổng

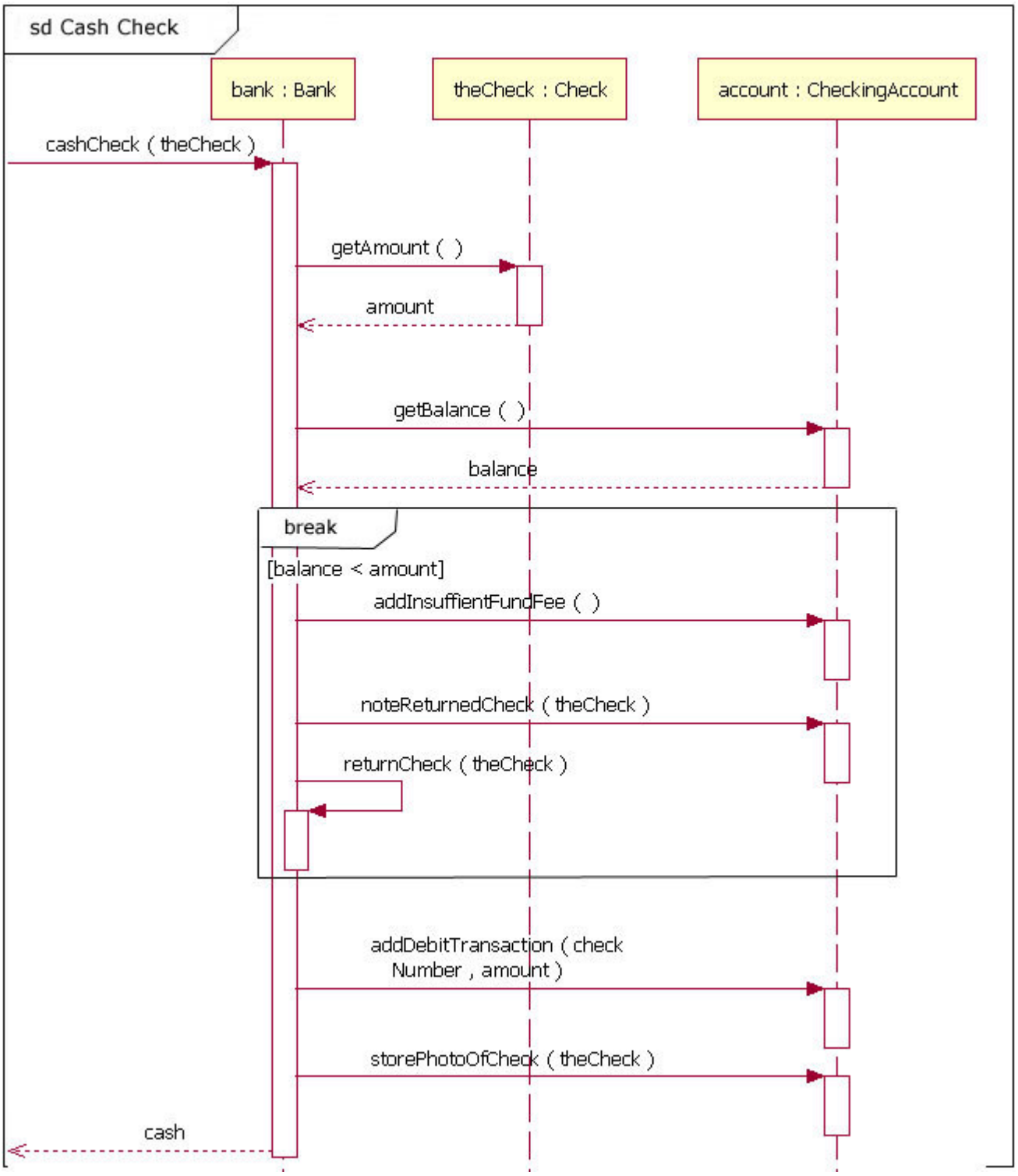
Kết hợp các mảng (vỡ và song song)

Trong phần "cơ bản" được giới thiệu trước đây trong bài viết này, Tôi đã nói về các mảng kết hợp được biết như là "thay thế", "tuỳ chọn", và "vòng lặp". Ba mảng kết hợp này được mọi người sử dụng nhiều nhất. Tuy nhiên, có hai mảng kết hợp khác một phần lớn người dùng sẽ tìm thấy sự hữu ích - vỡ và song song.

Vỡ

Trong mọi trường hợp, mảng kết hợp vỡ hầu như giống mảng kết hợp tuỳ chọn, với hai trường hợp ngoại trừ. Thứ

nhất, một khung vỡ có một hộp tên với chữ "break" (vỡ) thay vì "option" (tuỳ chọn). Thứ hai, khi một thông điệp của mảng kết hợp vỡ được thực thi, các thông điệp còn lại của sự tương tác kèm theo sẽ không được thực thi vì trình tự vỡ ra khỏi sự tương tác kèm theo. Theo cách này, mảng kết hợp vỡ giống từ khoá trong một ngôn ngữ lập trình như C++ hoặc Java.



Hình 16: Xử lý lại mảng biểu đồ trình tự từ Hình 8, với mảng sử dụng một đoạn vỡ thay vì một sự thay thế.

Các đoạn vỡ được sử dụng thông dụng nhất để mô hình hoá ngoại trừ sự kiểm soát. Hình 16 được xử lý lại của Hình 8, nhưng lúc này Hình 16 sử dụng một đoạn mảng kết hợp vỡ thay vì như là một sự thay thế. Để đọc Hình 16, bạn bắt đầu từ góc phía trên, bên trái của trình tự và đọc xuống. Khi trình tự có giá trị phản hồi "balance" (số dư), nó kiểm tra xem số dư có ít số tiền không. Nếu số dư không ít hơn số tiền, thông điệp tiếp theo là thông điệp addDebitTransaction,

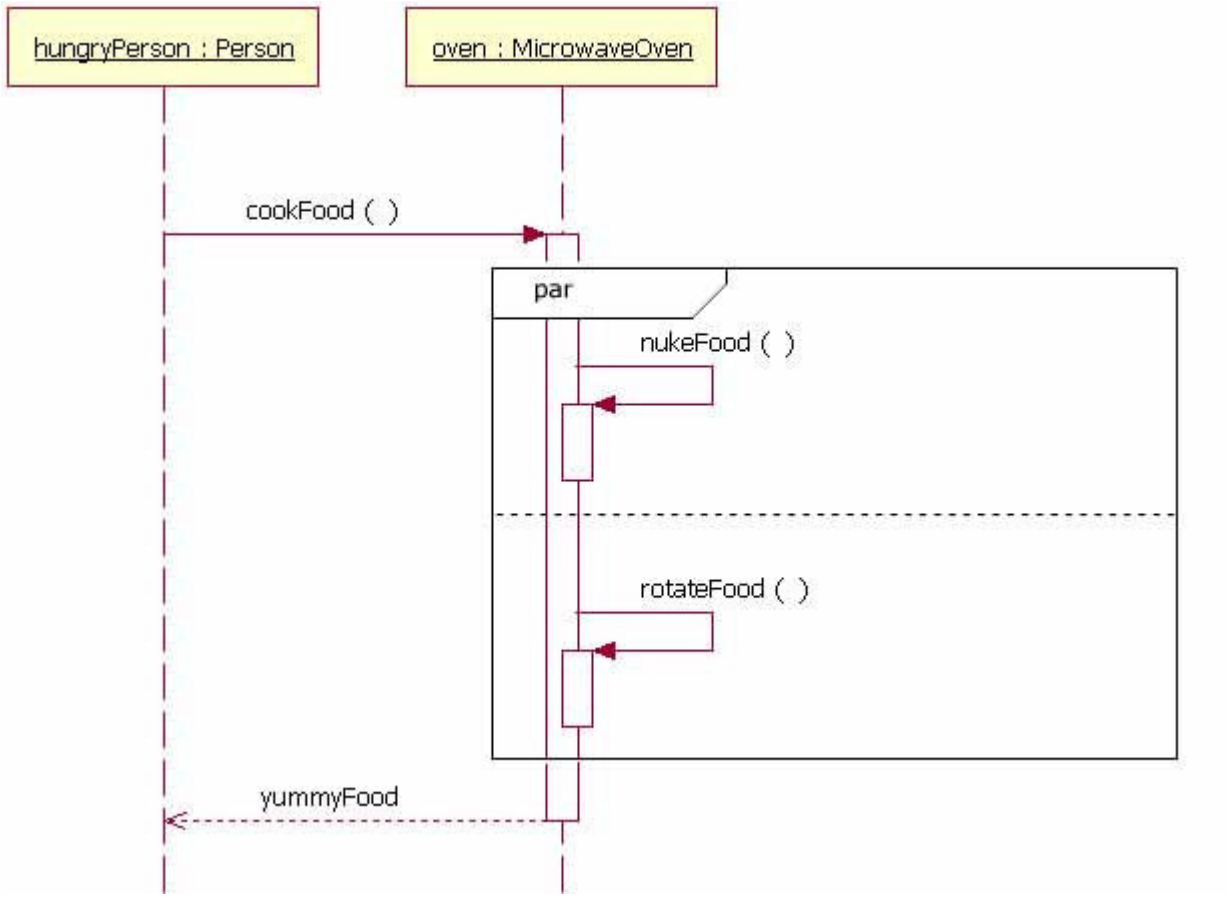
và trình tự tiếp tục một cách thông thường. Tuy nhiên, trong các trường hợp khi số dư ít hơn số tiền, thì trình tự kích hoạt mảng kết hợp vỡ và các thông điệp của nó được gửi đi. Khi tất cả các thông điệp trong sự kết hợp vỡ được gửi đi, trình tự thoát ra mà không gửi đi một thông điệp còn lại nào (ví dụ: addDebitTransaction).

Một điều quan trọng cần lưu ý về các đoạn vỡ là chúng chỉ gây sự thoát ra của trình tự của sự tương tác kèm theo và không nhất thiết trình tự hoàn chỉnh được miêu tả trong biểu đồ. Trong các trường hợp khi một sự kết hợp vỡ là một phần của sự thay thế hay vòng lặp, thì chỉ sự thay thế hay vòng lặp là được kết thúc.

Song song

Các hệ thống máy tính hiện đại ngày nay rất tiên tiến về mức độ mức tạp và những lúc thực hiện công việc cùng một lúc. Khi xử lý thời gian yêu cầu để hoàn thành các phần của công việc phức tạp dài hơn dự kiến, một số hệ thống kiểm soát các phần của bộ xử lý theo cách song song. Thành phần mảng kết hợp song song cần được sử dụng khi tạo một biểu đồ trình tự mà biểu hiện các hoạt động xử lý song song.

Mảng kết hợp song song được vẽ sử dụng một khung, và bạn đặt chữ "par" trong hộp tên của khung. Sau đó bạn chia phần nội dung của khung thành các toán hạng chiều ngang được chia tách bởi một đường có nét gạch. Mỗi toán hạng trong khung đại diện một mạch xử lý hoàn tất trong song song.



Hình 17: Lò vi sóng là ví dụ một vật làm hai việc song song.

Hình 17 có thể không minh hoạ ví dụ hệ thống xử lý tốt nhất một đối tượng làm nhiều việc song song, mà nó đưa ra một ví dụ dễ hiểu của một trình tự với các hoạt động song song. Trình tự giống như sau: một người đói (hungryPerson) gửi một thông điệp cookFood (nấu thức ăn) tới lò vi sóng. Khi lò vi sóng nhận được thông điệp, nó gửi hai thông điệp tới chính nó cùng một lúc (nukeFood and rotateFood) (làm chín và quay thức ăn). Sau khi cả hai thông điệp hoàn tất, đối tượng người đói được phản hồi yummyFood (thức ăn đã sẵn sàng) từ lò vi sóng.

Kết luận

Biểu đồ trình tự là một biểu đồ tốt cho sử dụng để dẫn chứng một yêu cầu của hệ thống và để phát hiện một thiết kế của hệ thống. Nguyên nhân biểu đồ trình tự rất có tác dụng vì nó thể hiện sự tương tác logic giữa các đối tượng trong hệ thống tại thời điểm được yêu cầu mà sự tương tác xảy ra.

Tham khảo

- UML 2.0 Superstructure Final Adopted Specification (đặc biệt là Chương 8) http://www.omg.org/cgi-bin/doc?ptc/2003-08-02
- Tổng quan Biểu đồ trình tự UML 2 http://www.agilemodeling.com/artifacts/sequenceDiagram.htm
- Bài viết về UML 2 http://www.omg.org/news/meetings/workshops/ UML%202003%20Manual/Tutorial7-Hogg.pdf

---

Lưu ý

1 Trong các hệ thống được mô hình hoá đầy đủ, các đối tượng (trường hợp của các lớp) cũng sẽ được mô hình trên biểu đồ lớp của hệ thống.

2 Khi đọc biểu đồ trình tự này, giả sử rằng chuyên viên phân tích đã đăng nhập vào hệ thống.

3 Xin lưu ý rằng, có thể có hai hoặc hơn các điều kiện trạng thái bảo vệ được gắn với các toán hạng thay thế khác nhau, là đúng tại cùng thời điểm, nhưng hầu như chỉ một toán hạng sẽ thực sự diễn ra khi chạy (cái mà sự thay thế "chiến thắng" trong các trường hợp như vậy không được xác định bởi tiêu chuẩn UML).

4 Mặc dù các toán hạng trông rất giống các làn đường trên đường cao tốc, Tôi đặc biệt không gọi chúng là các làn đường. Các hàng song song (swim lanes) là một chú thích UML được sử dụng trong các biểu đồ hoạt động. Xin tham khảo tại bài viết trước của *The Rational Edge's* về các biểu đồ trình tự.

5 Thông thường, sự trợ giúp đối với trạng thái bảo vệ được đính kèm là sự trợ giúp sở hữu biến số mà được bao gồm trong trạng thái bảo vệ.

6 Cùng với mảng kết hợp tuỳ chọn, mảng kết hợp vòng lặp không yêu cầu rằng một điều kiện trạng thái bảo vệ được đặt vào đó.

7 Có thể sử dụng lại một biểu đồ trình tự với các thể loại (ví dụ: lập trình hoặc kinh doanh). Tôi mới thấy rằng các chuyên viên phát triển muốn chia tách chức năng các biểu đồ của họ hơn nữa.

Đôi nét về tác giả

Donald Bell là một chuyên gia CNTT tại IBM Global Services, ở đây ông làm việc với các khách hàng của IBM để thiết kế và phát triển giải pháp phần mềm dựa trên J2EE