

Chương 4: Sơ đồ tương tác (Interaction Diagram)

Nội dung

- * Giới thiệu
- * Lợi ích của tương tác
- * Thông báo và sự kiện

Giới thiệu

- * Các sơ đồ tương tác cho thấy làm thế nào các thể hiện ở trong lòng hệ thống có thể tương tác với nhau để thực hiện một chức năng nào đó.
- * Các tương tác khá nhiều và rất đa dạng, cần có một ngôn ngữ phong phú để biểu diễn chúng. UML cung cấp nhiều sơ đồ cho mục đích này :
 - * Sơ đồ tuần tự (sequence diagram)
 - * Sơ đồ cộng tác (collaboration diagram)
 - * Sơ đồ truyền thông (communication diagram)
 - * Sơ đồ về thời gian (timing diagram)

Lợi ích của tương tác

- * Định nghĩa
- * Ký hiệu
- * Ứng dụng của sơ đồ tương tác

Định nghĩa

- * **Tương tác (interaction):** Một tương tác định nghĩa hành vi của một hệ có cấu trúc (một hệ thống con, một trường hợp sử dụng, một lớp, một thành tố...) bằng cách đặc biệt chú trọng đến việc trao đổi các thông tin giữa các thể hiện của nó.
- * **Sinh tuyến (lifeline):** Một sinh tuyến biểu diễn một thành phần duy nhất (có thể tương ứng với một người ngoài thế giới thực, hoặc một đối tượng trong lớp dữ liệu) tham gia vào một tương tác.

Định nghĩa

- * **Sơ đồ tương tác:**

- * Các sơ đồ tuần tự và sơ đồ cộng tác biểu diễn các tương tác giữa các sinh tuyến
 - * Một sơ đồ tuần tự chỉ ra các tương tác dưới góc độ thời gian, đặc biệt là các chuỗi thông báo (messages) trao đổi nhau giữa các sinh tuyến
 - * Một sơ đồ cộng tác biểu diễn về mặt không gian của các sinh tuyến
- * Sơ đồ thời gian mô hình hóa các hệ thống được khai thác dưới những điều kiện nghiêm ngặt về thời gian (sơ đồ tuần tự hoàn toàn có thể thực hiện chức năng này)

Ký hiệu

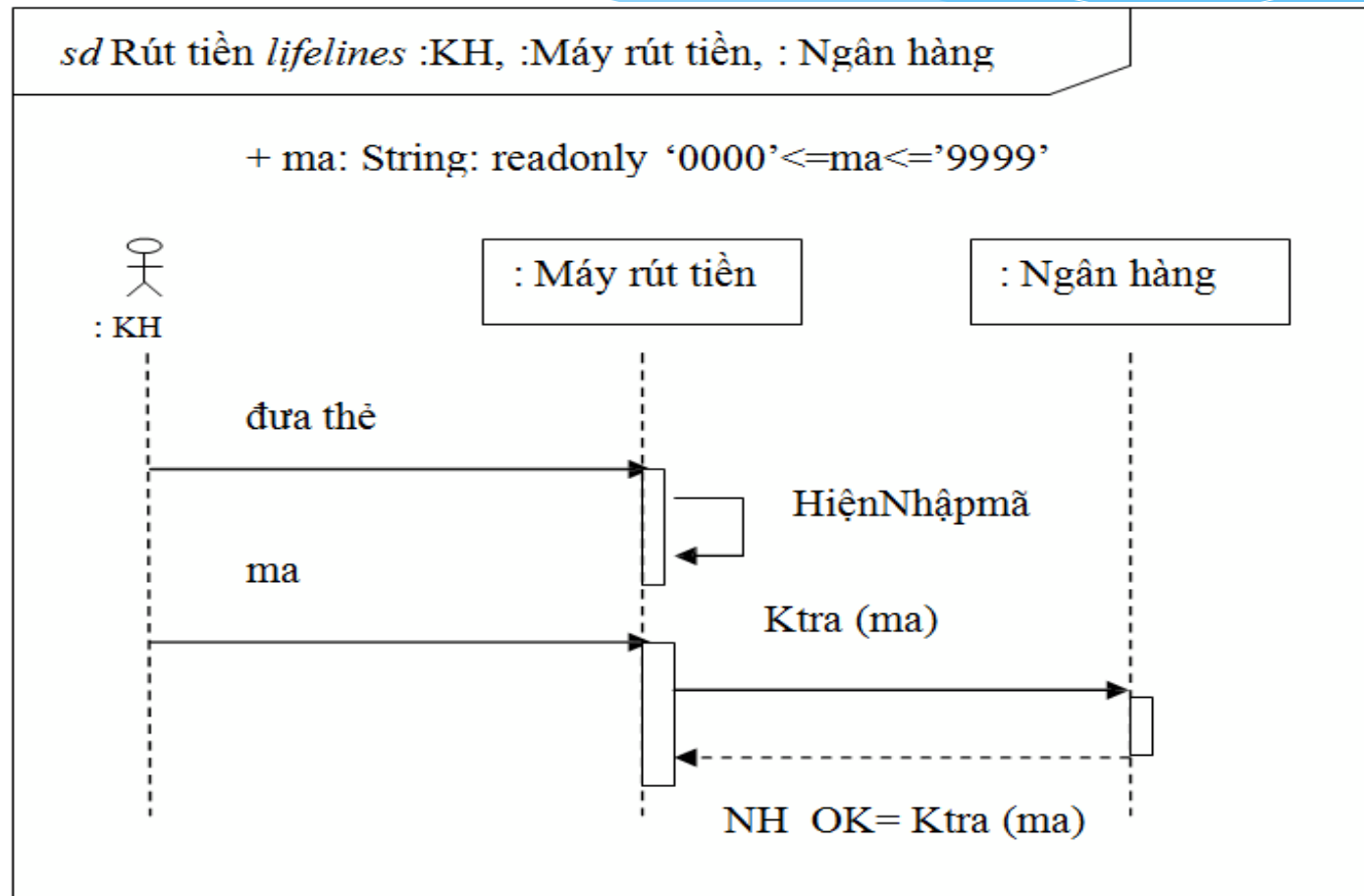
* Sơ đồ tương tác:

- * Biểu diễn bằng một hình chữ nhật, góc trái có tên của sơ đồ, đứng sau từ khóa :
 - * **sd** : nếu là sơ đồ tuần tự
 - * **com** : nếu là sơ đồ cộng tác
- * Có thể có :
 - * Danh sách các sinh tuyến đứng tiếp theo với từ khóa lifelines và dấu hai chấm
 - * Ràng buộc trên ít nhất một thuộc tính nào đó có dùng trong sơ đồ

* Sinh tuyến:

- * Một sinh tuyến được biểu diễn bằng một hình chữ nhật hoặc một hình người « treo » một đường thẳng dọc có chấm
- * Trong hình chữ nhật hoặc phía dưới hình người chứa một định danh có ngữ pháp như sau : [**<tên sinh tuyến> [<chỉ số>]**] : **<tên lớp>**

Ví dụ



Ứng dụng của sơ đồ tương tác

- * Các sơ đồ tương tác được sử dụng trong suốt một dự án, từ khi thu thập các nhu cầu cho đến khi thiết kế, nhằm:
 - * Mô tả các trường hợp sử dụng
 - * Mô hình hóa việc sử dụng một lớp hoặc một phương thức của lớp
 - * Thêm vào khía cạnh động cho việc mô hình hóa một hệ thống

Thông báo

- * Các dạng thông báo
- * Ký hiệu
- * Thông báo và sự kiện
- * Ngữ pháp của thông báo
- * Ràng buộc trên các sinh tuyến
- * Các kiểu phân đoạn của tương tác
- * Phân rã một sinh tuyến

Các dạng thông báo




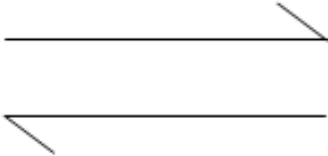
- * Một thông báo định nghĩa một liên lạc giữa các sinh tuyến.
- * Có nhiều dạng thông báo, trong đó phổ biến nhất là dạng:
 - * **Gửi một tín hiệu:** Thông báo khởi động một phản ứng nơi đối tượng nhận một cách không đồng bộ và không cần đợi trả lời
 - * **Kích hoạt thực hiện một thao tác trong một phương thức:** Đây là dạng thông báo được sử dụng nhiều nhất trong lập trình hướng đối tượng
 - * Trong thực tế, đa số thông báo loại này là đồng bộ
 - * Cũng có thể thực hiện thông báo dạng này một cách không đồng bộ qua các thread
- * **Tạo hoặc hủy một thể hiện (đối tượng)**

Ký hiệu

- * Thông báo đồng bộ và thông báo không đồng bộ
- * Tạo và hủy một đối tượng

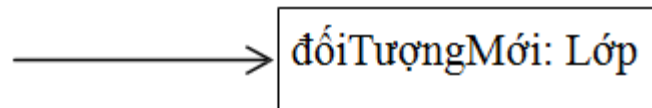
Thông báo đồng bộ và thông báo không đồng bộ

- * **Thông báo đồng bộ (sychrone)** : Ở cuối mũi tên, ở đối tượng nhận, có thể có trả lời phản hồi bằng mũi tên đứt đoạn
- * **Thông báo không đồng bộ (asynchrone)**

	Thông báo đồng bộ (sychrone)	Thông báo không đồng bộ (asynchrone)
Cách 1		
Cách 2		

Tạo và hủy một đối tượng

* Tạo đối tượng:



* Hủy đối tượng:

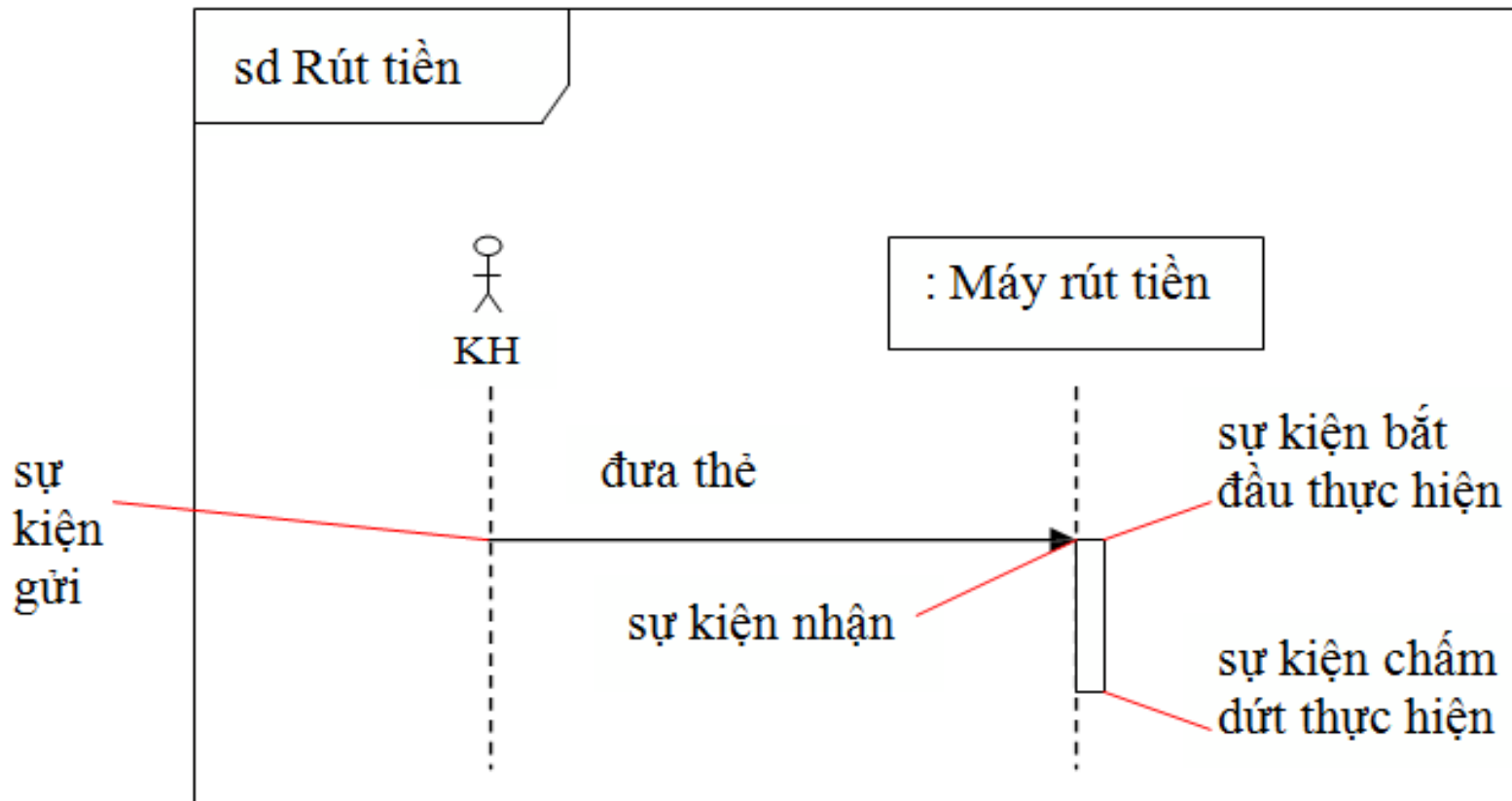


Thông báo và sự kiện

- * UML phân biệt việc gửi với nhận một thông báo, cũng như phân biệt việc bắt đầu với việc kết thúc thực hiện một phản ứng
- * Các sự kiện (event) được dùng để đánh dấu từng giai đoạn

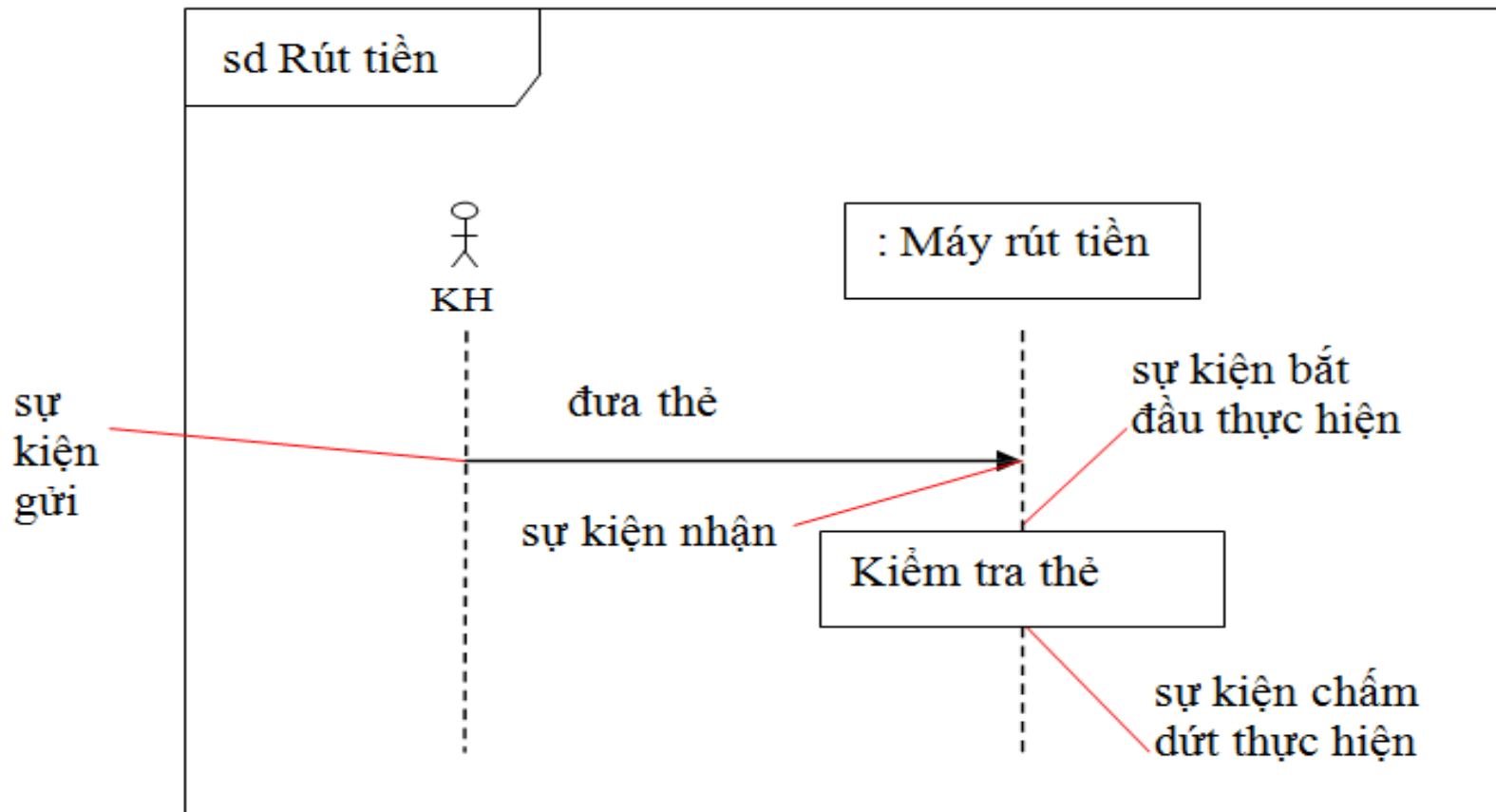
Thông báo và sự kiện

Ví dụ:





Thông báo và sự kiện

Hoặc:



Thông báo và sự kiện

- * Dựa vào các sự kiện gửi và nhận, UML định nghĩa 3 kiểu thông báo :
 - * **Thông báo đầy đủ:** các sự kiện gửi và nhận đều được biết
 - * **Thông báo bị lạc:** sự kiện gửi có được biết, nhưng sự kiện nhận thì không
 - * *Ký hiệu:* 
 - * **Thông báo tìm được:** ngược lại, sự kiện nhận có được biết, nhưng sự kiện gửi thì không
 - * *Ký hiệu:* 

Ngữ pháp của thông báo

* Thông báo gửi :

- * **Cú pháp:** <tên tín hiệu hoặc tên phương thức> [(<danh sách tham số>)]
- * Trong danh sách, các tham số cách nhau bởi dấu phẩy, và mỗi tham số có ngữ pháp như sau :

[<tên tham số> [<dấu>] <trị của tham số>]

- * <dấu> là dấu '=' nếu đó là tham số chỉ đọc
- * <dấu> là dấu ':' nếu đó là tham số có thể sửa được (đọc/ghi)

* Ví dụ:

- * `nhapMa(«1234»)`
- * `nhapMa(m)`
- * `ghiNhan(maSV, maSach)`
- * `suaSolg(sl : 10)`

Ngữ pháp của thông báo

- * **Thông báo trả lời :**

- * **Cú pháp:** [**<thuộc tính> =] <thông báo gửi> [: <trị trả về>]**

- * **Ví dụ:**

- * solgTon= ktraHgTon(« A001 »)

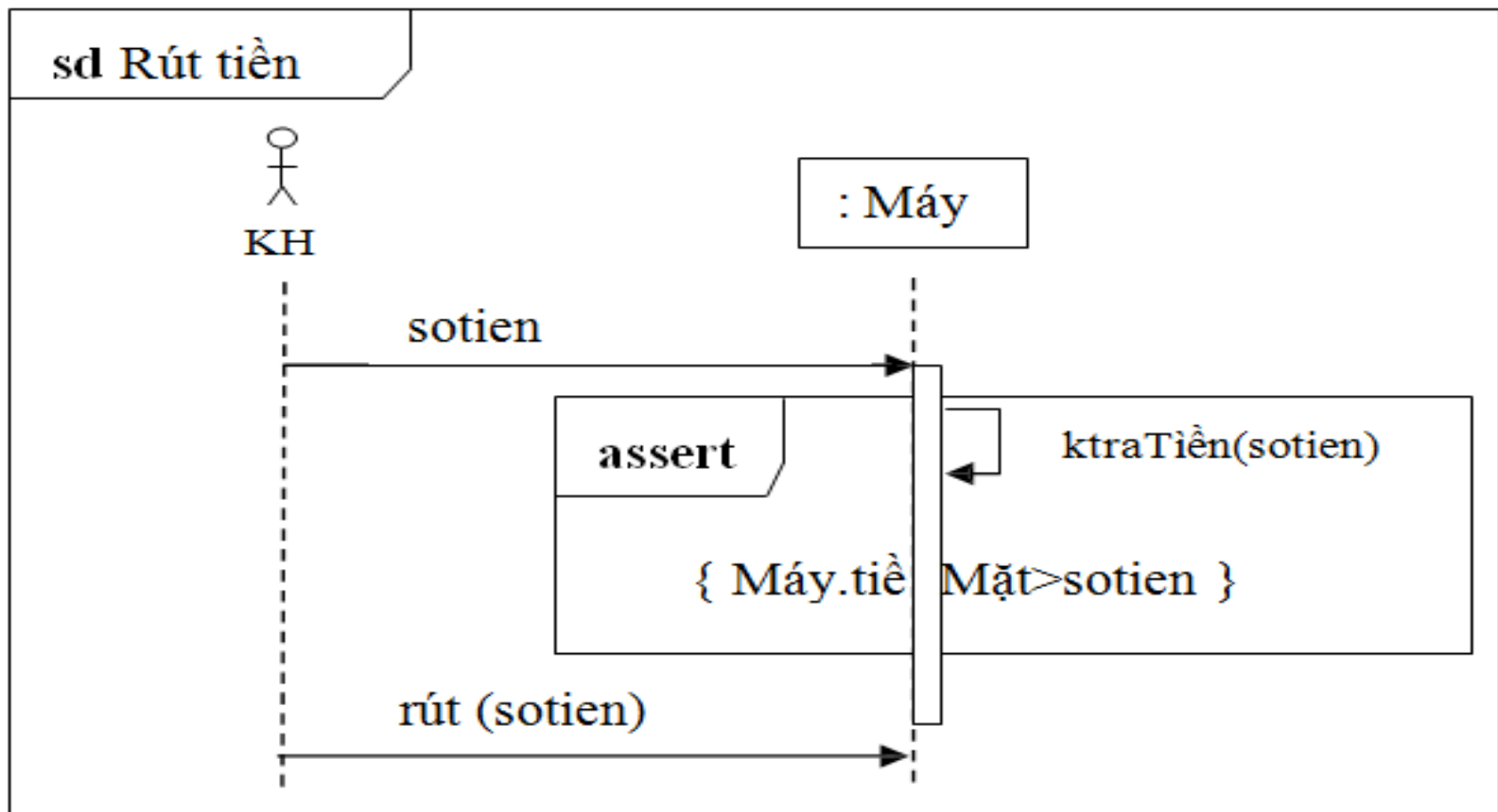
- * maOK=ktraMa(ma) : true

Ràng buộc trên các sinh tuyến

- * Các sinh tuyến của một tương tác có thể mang đủ loại ràng buộc
- * **Ký hiệu:**
 - * Một ràng buộc được biểu diễn trên một sinh tuyến bằng một văn bản giữa dấu móc ({ })
 - * Một ràng buộc cũng có thể được chứa trong một ghi chú gắn với thể hiện của sự kiện liên quan

Ràng buộc trên các sinh tuyến

Ví dụ:



Ràng buộc trên các sinh tuyến

* Chú ý:

- * Một ràng buộc được đánh giá khi khai thác tương tác
- * Nếu ràng buộc không được thỏa mãn, các thể hiện của sự kiện đi theo sau ràng buộc nay sẽ được xem là không hợp lệ, ngược lại với khi ràng buộc được thỏa mãn
- * Như vậy, một sơ đồ tương tác có thể mô tả các thông báo không hợp lệ, nghĩa là không bao giờ được gửi đi

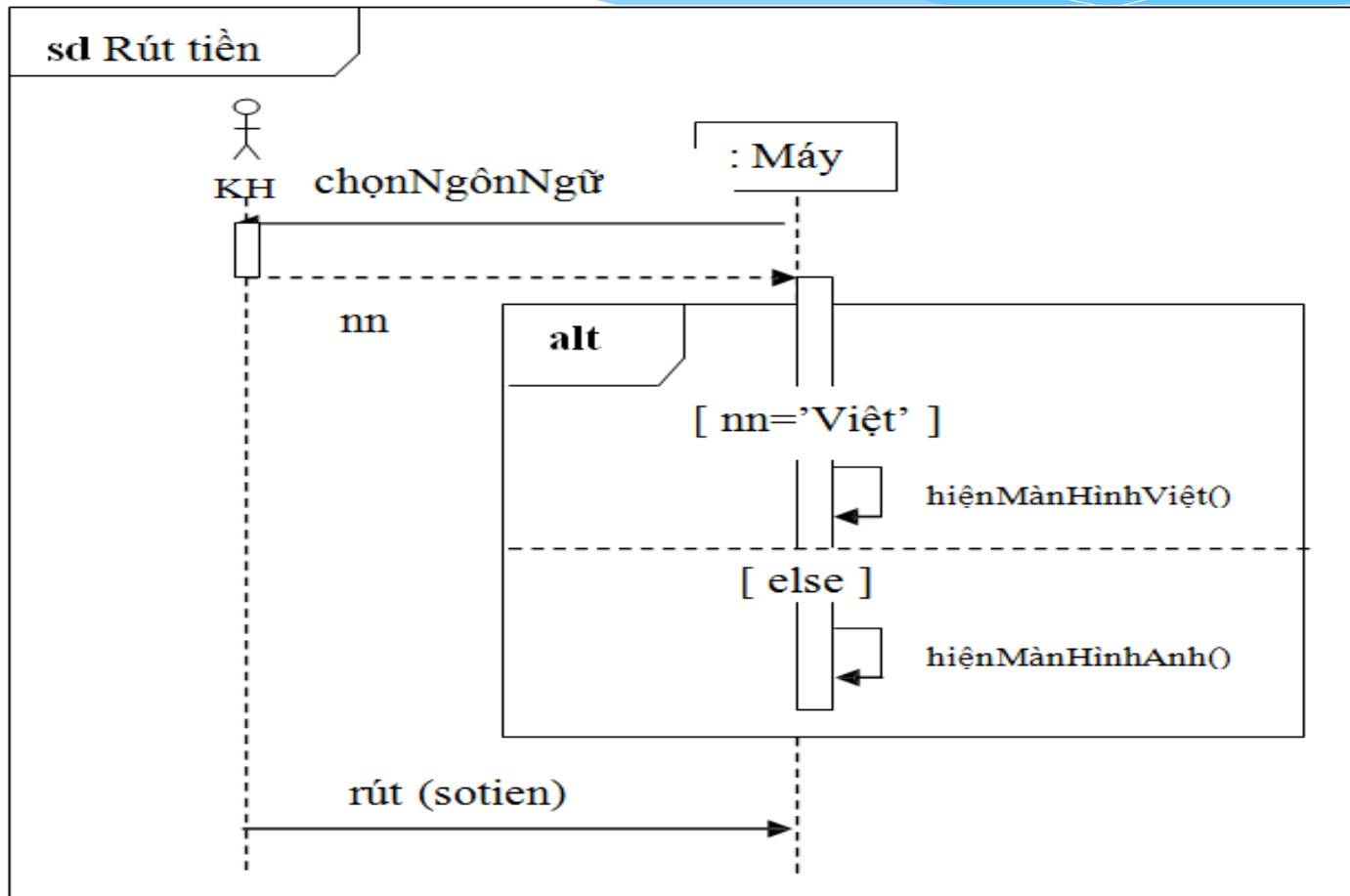
Các kiểu phân đoạn của tương tác

- * Rẽ nhánh
- * Vòng lặp
- * Xử lý song song
- * Các toán tử assert, ignore và consider

Rẽ nhánh

- * Có 2 từ khóa là :
 - * « **alt** » (**alternative**) và “**else**”: rẽ nhánh 1 nếu điều kiện đúng và rẽ nhánh 2 nếu ngược lại.
 - * « **opt** » (**optional**) : thực hiện hành vi tiếp theo nếu điều kiện đúng.

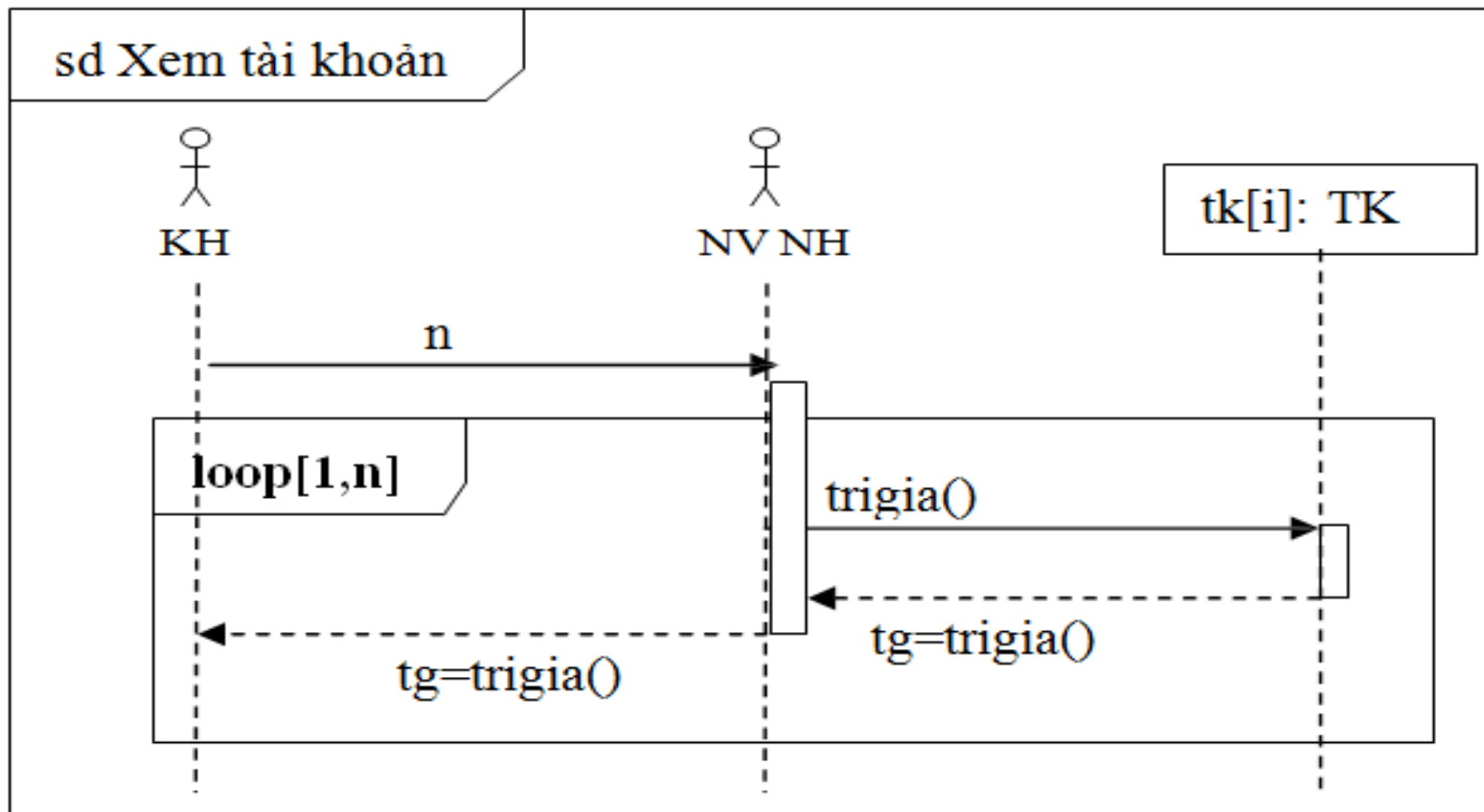
Ví dụ



Vòng lặp

- * Sử dụng 2 từ khóa :
 - * « **loop** » : có 3 trường hợp
 - * **Loop** : lặp mãi đến khi có lệnh « break »
 - * **Loop <điều kiện>** : còn lặp lại khi điều kiện còn thỏa
 - * **Loop [<chỉ số min>, <chỉ số max>]** : thường có dạng loop [1, n], lặp lại các bước lặp theo chỉ số nhận trị từ <chỉ số min> đến <chỉ số max>
 - * Lưu ý : các biến trong điều kiện và các chỉ số phải được gán trị trước khi đặt tả loop
 - * « **break** » : để thoát ra khỏi vòng lặp

Ví dụ



Xử lý song song

- * Dùng từ khóa « par » (parallel)

Các toán tử **assert**, **ignore** và **consider**

- * Toán tử **assert**: khẳng định sự cần thiết của việc gửi thông báo tiếp theo
- * Các phép **ignore** và **consider** cho phép tập trung chú ý để mô hình hóa một số thông báo có thể được gửi trong một tương tác
 - * **Ignore** định nghĩa các thông báo có thể bỏ qua
 - * **Consider** định nghĩa các thông báo cần phải chú ý đến.
- * Các toán tử **assert**, **ignore** và **consider** được đặt sau tên của sơ đồ tuần tự, với ngữ pháp như sau :
 - ignore { danh sách thông báo }
 - consider { danh sách thông báo }

Phân rã một sinh tuyến

- * Đôi khi, một tương tác quá phức tạp để có thể được mô tả trong một sơ đồ thôi, nên ta có thể cắt một sinh tuyến ra trên nhiều sơ đồ
- * Một phần của sinh tuyến được phân rã ra sẽ được thay thế bằng một hình chữ nhật với từ khóa *ref* (reference), và được trình bày ở sơ đồ khác

Ví dụ

