Diagrama de Interaccion

Diagrama de Interacción

- Muestra los objetos que participan en una **interacción** y el orden de los mensajes en el tiempo
- Captura el **comportamiento dinámico**

Tipos:

- Diagramas de Secuencia
- Diagramas de Colaboración

Diagramas de Interacción

- Diagramas de Interacción
  - **Usos comunes**
    - Modelar los aspectos dinámicos de un sistema.
    - El uso de estos diagramas es en el contexto del sistema como un todo, un subsistema, una operación, o una clase.
    - Podemos unir diagramas de interacción para casos de uso (para modelar un escenario) y para colaboraciones (para modelar los aspectos dinámicos de una sociedad de objetos).

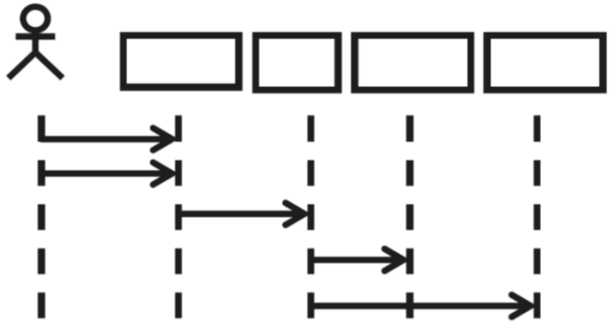
Diagramas de Interacción

Diagramas de Interacción

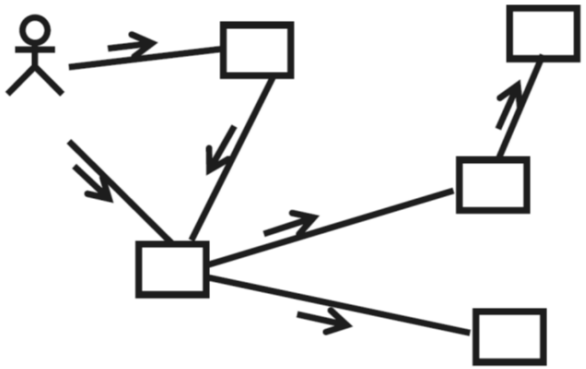
Usos comunes

- Cuando modelamos los aspectos dinámicos de un sistema, usamos diagramas de interacción de dos maneras:
  - **Para modelar flujos de control por orden de tiempo**
    - » Se usan *diagramas de secuencia*. Se hace énfasis en el paso de mensajes, en cómo se desenvuelven sobre el tiempo, lo cual es una manera útil para visualizar el comportamiento dinámico en el contexto de un escenario de un caso de uso.
  - **Para modelar flujos de control por organización**
    - » Se usan *diagramas de colaboración*. Se hace énfasis en las relaciones estructurales entre las instancias dentro de la interacción y junto con los mensajes que pueden ser pasados.
- Los diagramas de colaboración hacen un mejor trabajo para visualizar iteraciones y bifurcaciones complejas y para visualizar flujos de concurrencia múltiple de control.

Diagrama de Interacción



Diagramas de Secuencia



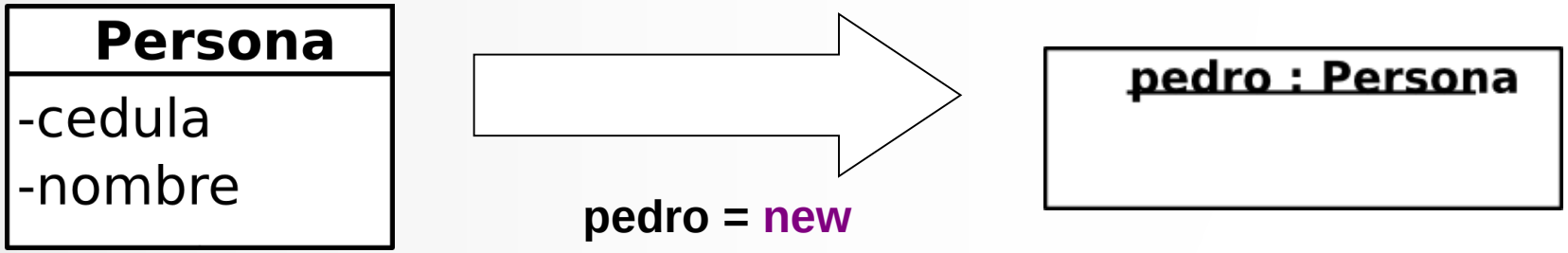
Diagramas de Colaboración

UML
Diagrama de Secuencia

Diagramas de Secuencia

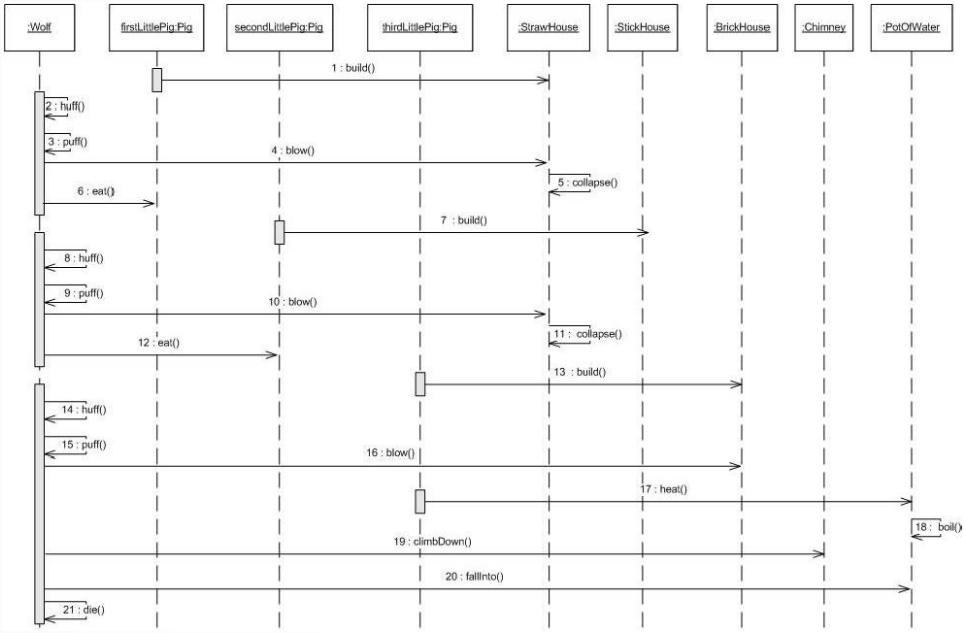
Los Diagramas de Secuencias muestran la forma en que *un grupo de objetos* se *comunican* (interactúan) entre sí a lo largo del *tiempo*

Un Diagrama de Secuencia consta de objetos, mensajes entre estos objetos y una línea de vida del objeto representada por una línea vertical



Es importante recordar la diferencia entre una clase y un objeto

Diagramas de Secuencia (Los tres cerditos)



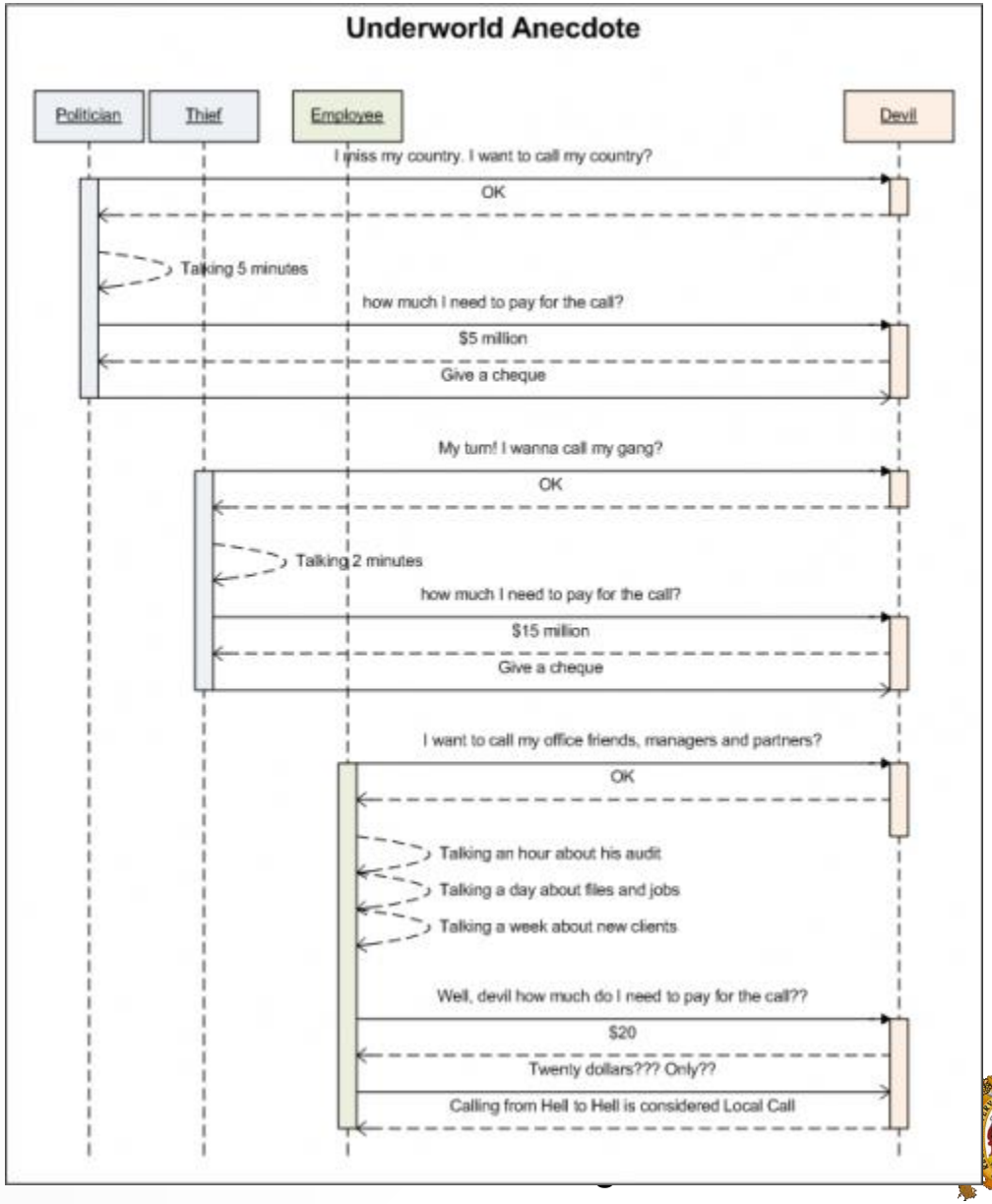
¿Qué tiene que ver un diagrama de secuencias con la fábula de los tres cerditos?

(Gracias Ken Howard)

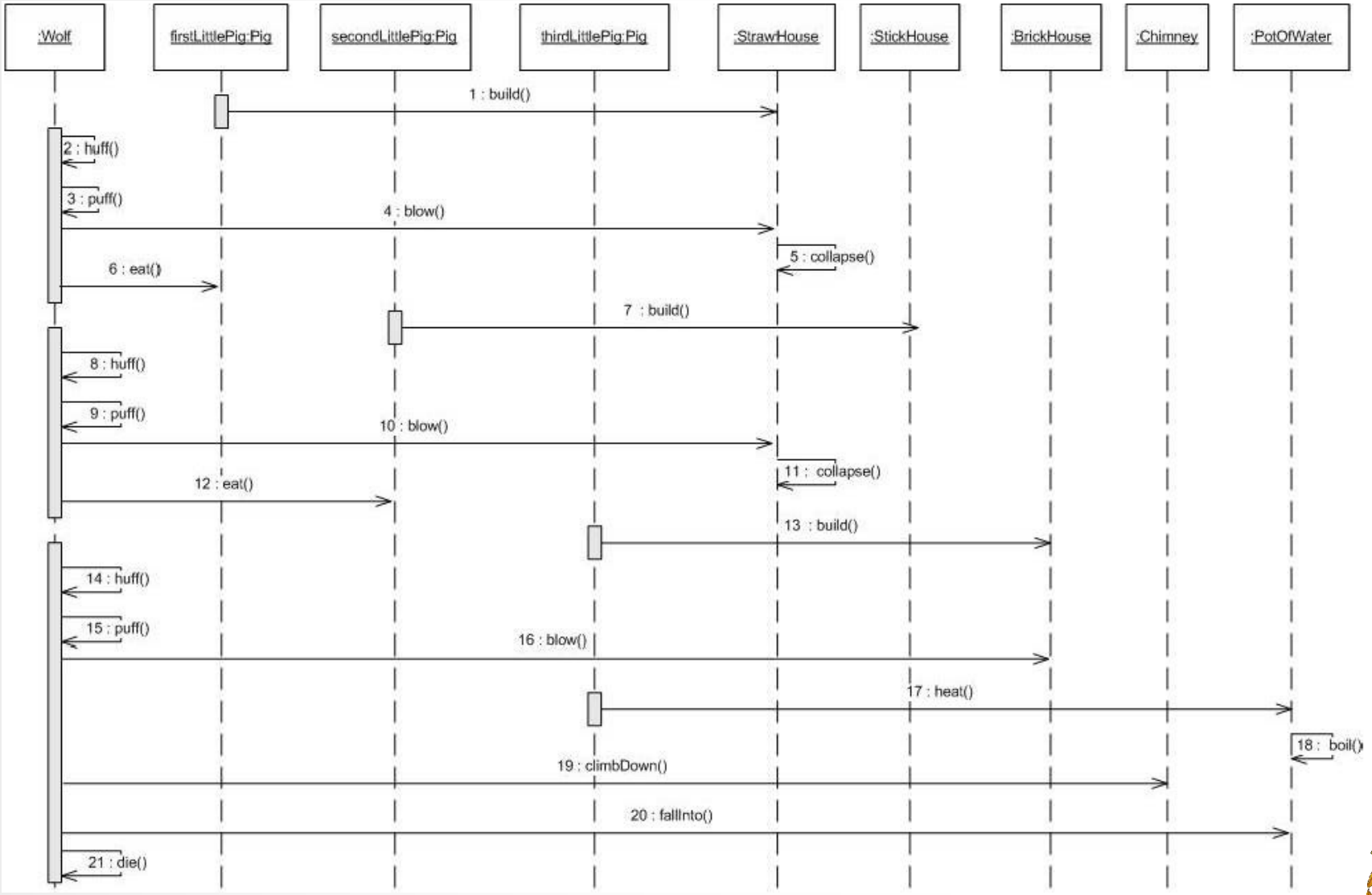
http://kenhoward01.blogspot.com/2008/06/three-little-pigs-in-uml.html

Diagramas de Secuencia (Los tres cerditos)

Los diagramas de Secuencias "cuentan" historias

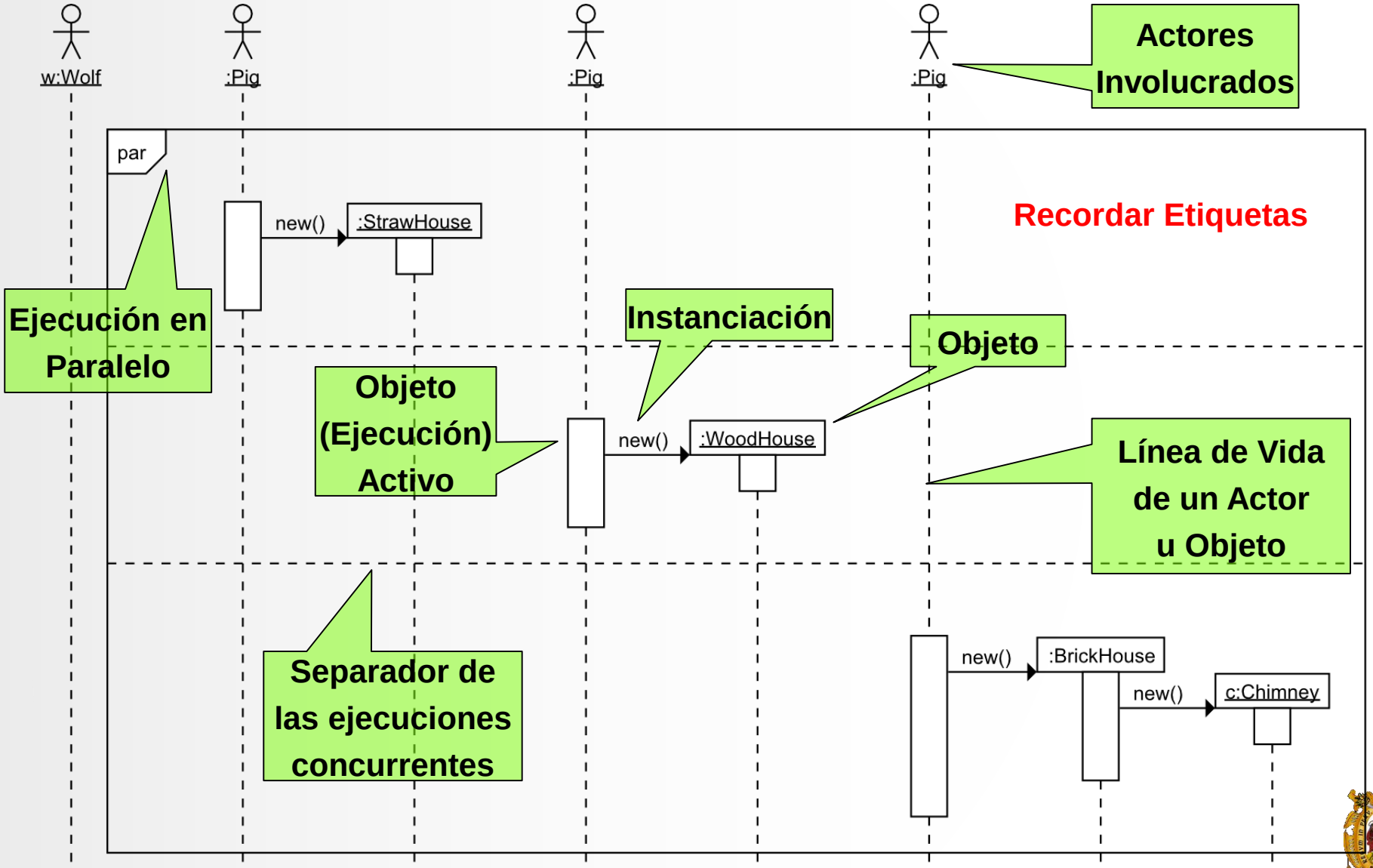


Diagramas de Secuencia (Los tres cerditos)



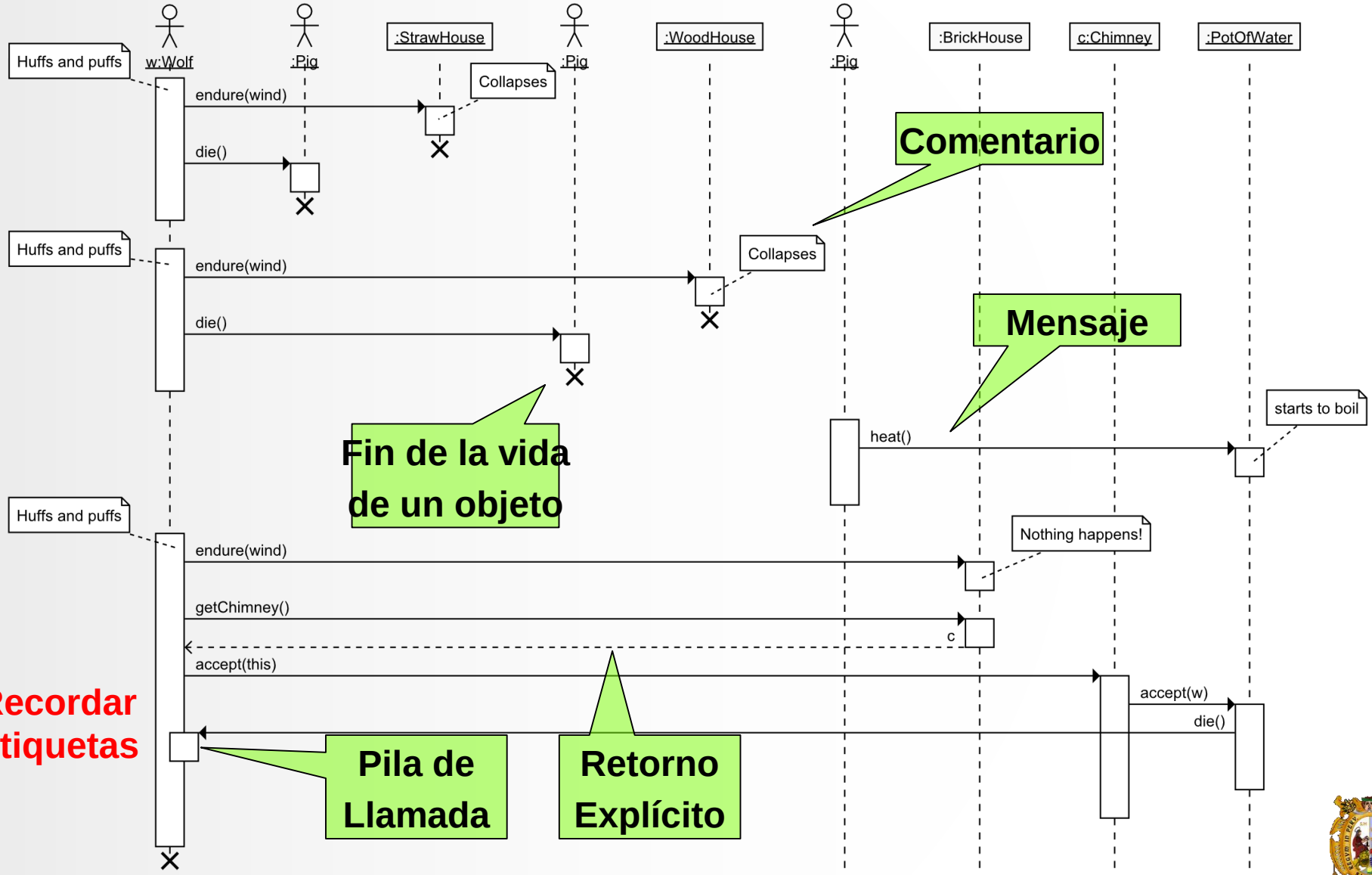
Fuente: http://kenhoward01.blogspot.com/2008/06/three-little-pigs-in-uml.html

Diagramas de Secuencia (Los tres cerditos)



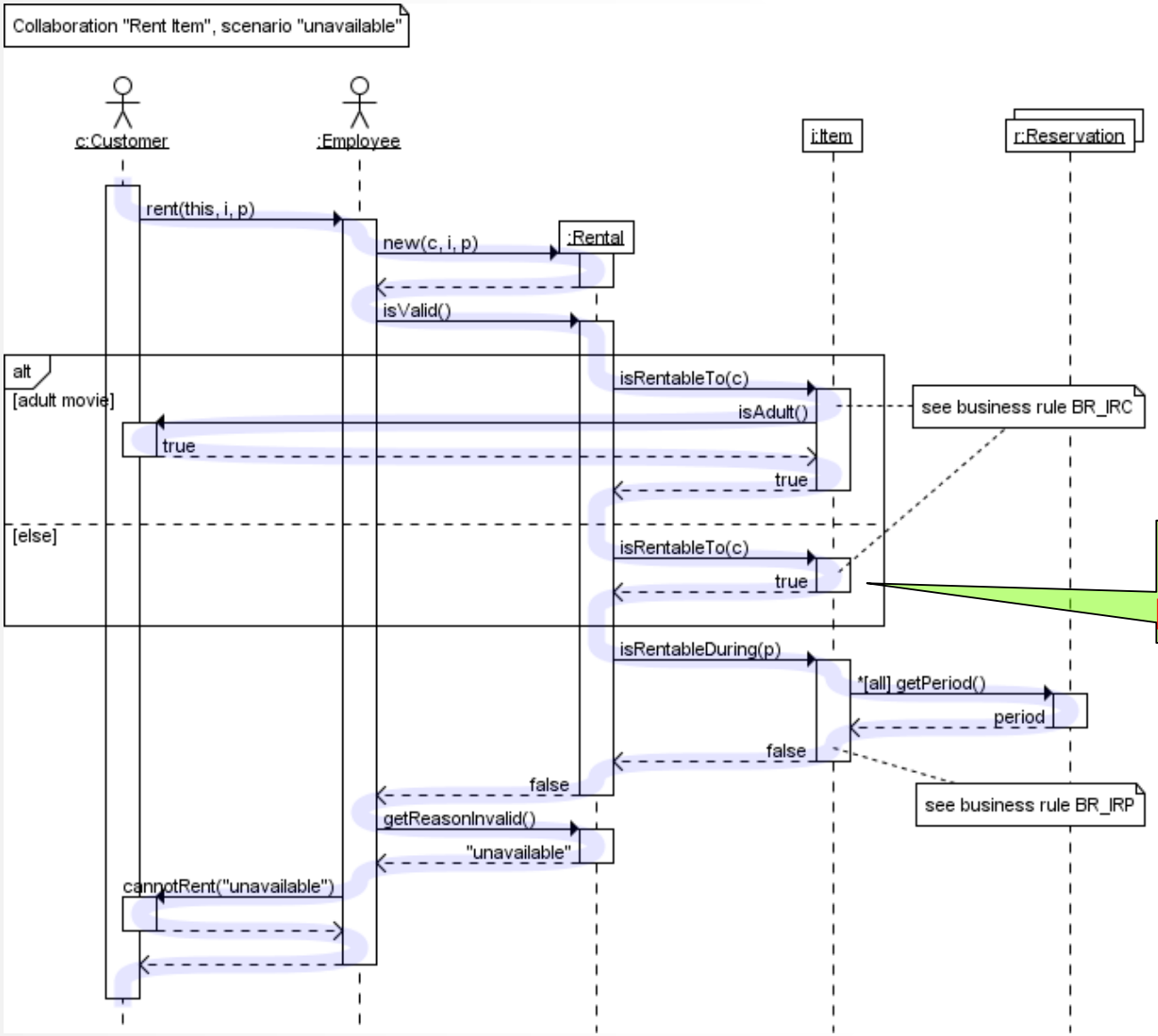
Fuente: http://www.tracemodeler.com/articles/pimp-my-diagram-three-little-pigs/

Diagramas de Secuencia (Los tres cerditos)



Fuente: http://www.tracemodeler.com/articles/pimp-my-diagram-three-little-pigs/

Diagramas de Secuencia (Los tres cerditos)



Fuente: http://www.tracemodeler.com/articles/pimp-my-diagram-three-little-pigs/

Diagramas de Secuencia (Relación con Casos de Uso)

Flujo Normal:

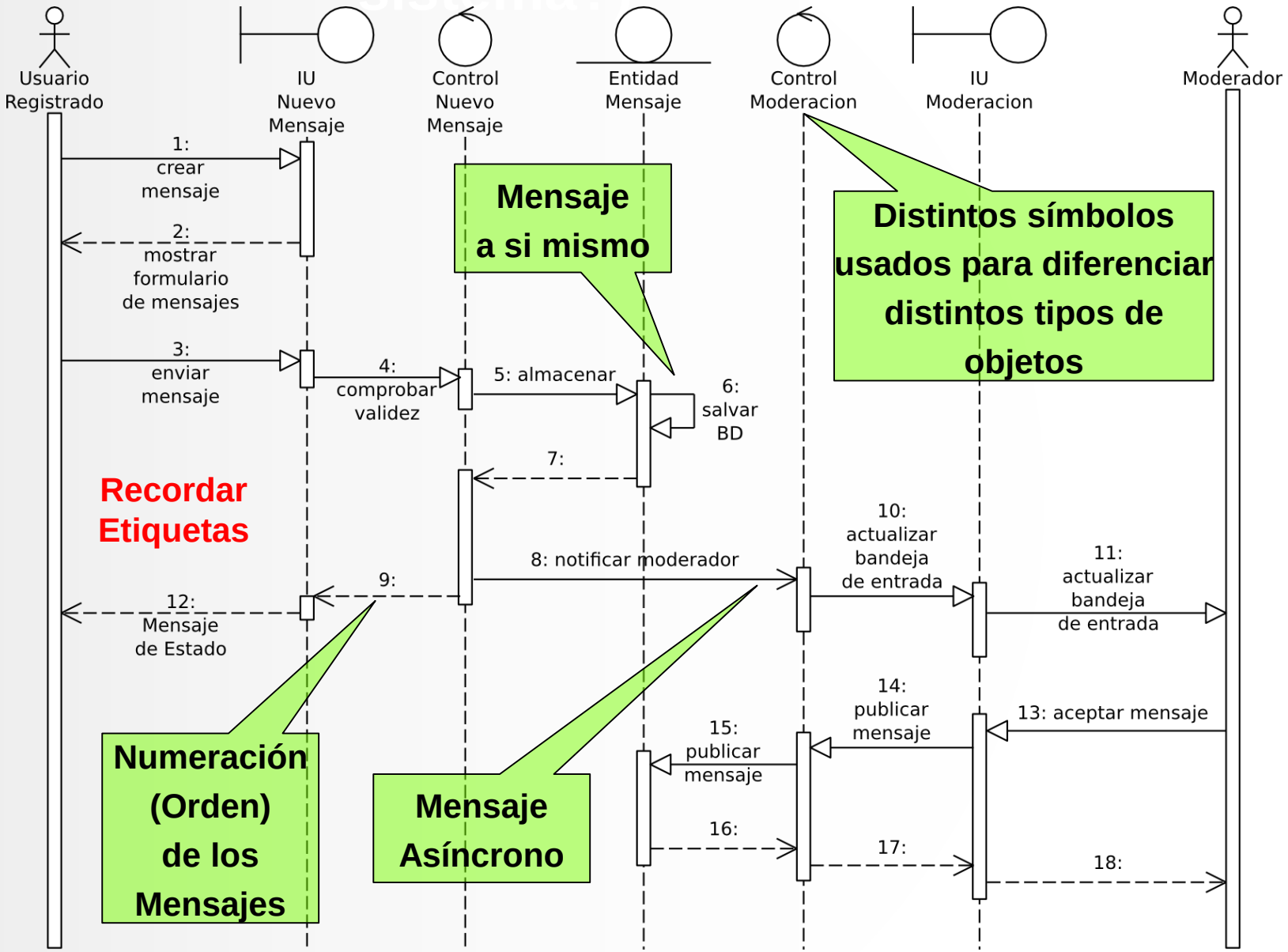
1.- El actor pulsa sobre el botón para crear un nuevo mensaje.
2.- El sistema muestra una caja de texto para introducir el título del mensaje y una zona de mayor tamaño para introducir el cuerpo del mensaje.
3.- El actor introduce el título del mensaje y el cuerpo del mismo.
4.- El sistema comprueba la validez de los datos y los almacena.
5.- El moderador recibe una notificación de que hay un nuevo mensaje.
6.- El moderador acepta y el sistema publica el mensaje si éste fue aceptado por el moderador.

Flujo Alternativo:

4.A.- El sistema comprueba la validez de los datos, si los datos no son correctos, se avisa al actor de ello permitiéndole que los corrija.

7.B.- El moderador rechaza el mensaje, de modo que no es publicado sino devuelto al usuario.

Descripción Textual de un Caso de Uso (Requerimientos: ¿Qué debe hacer el sistema?)

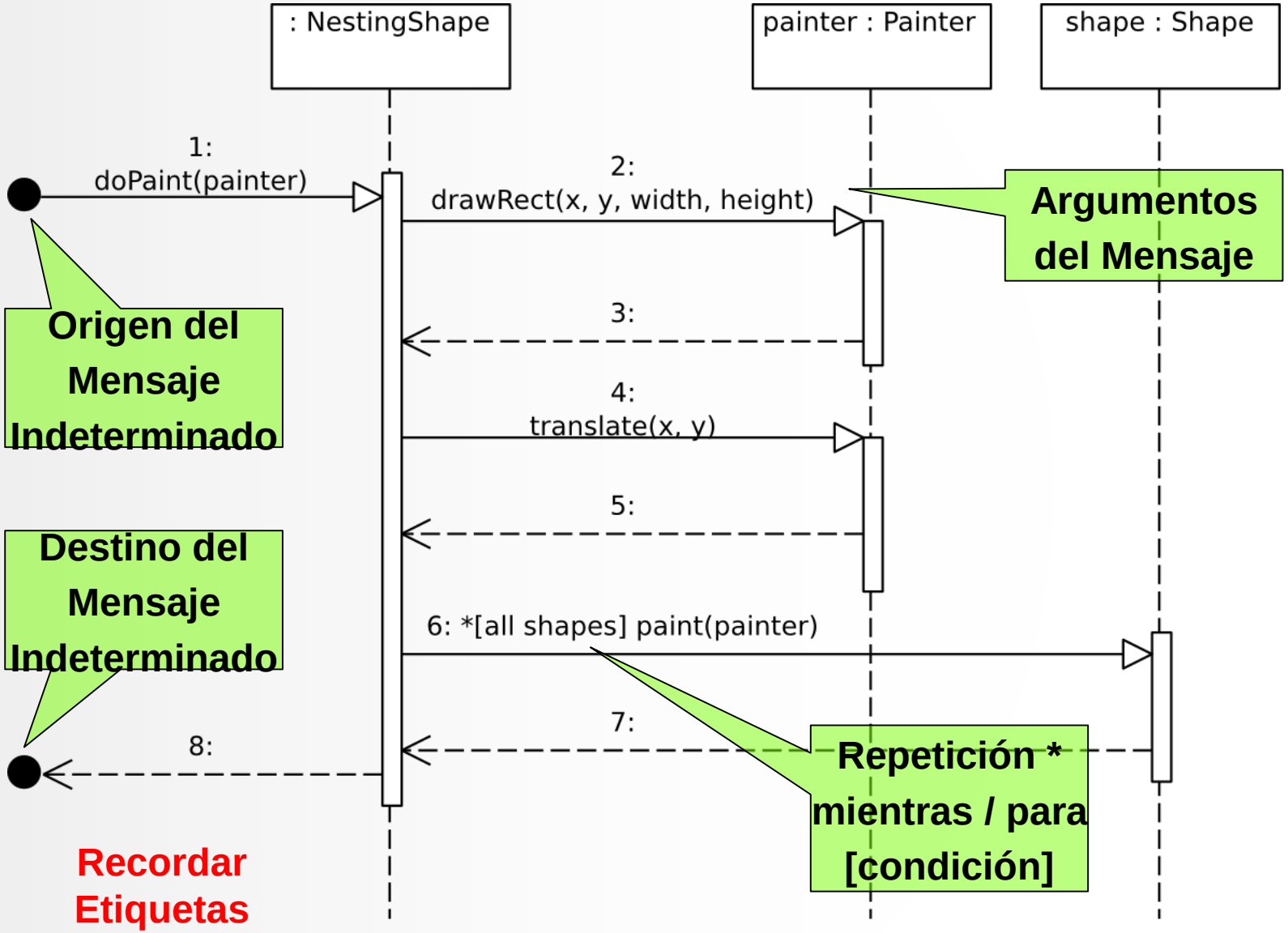


Diagramas de Secuencia (Implementación)

```
protected void doPaint(Painter painter) {
  painter.drawRect(x, y, width, height);

  // Cause painting of shapes to be relative to this shape
  painter.translate(x, y);

  for (Shape s : shapes) {
    s.paint(painter);
  }
}
```

Es posible utilizar un diagrama de secuencia para modelar el método anterior

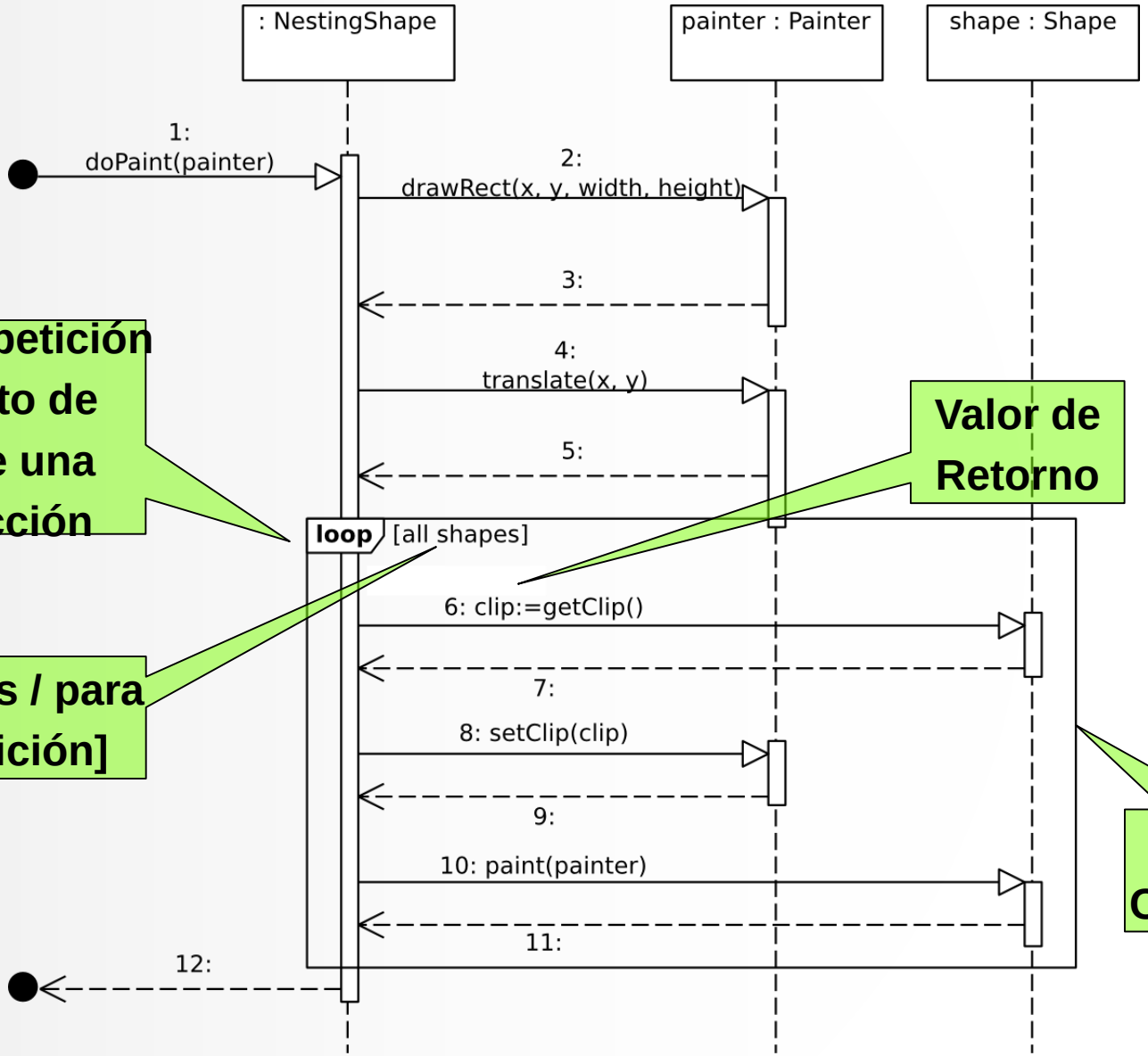
Diagramas de Secuencia (Implementación)



Diagramas de Secuencia (Implementación)

```
protected void doPaint(Painter painter, Config config) {
  painter.drawRect(x, y, width, height);

  // Cause painting of shapes to be relative to this shape
  painter.translate(x, y);

  for (Shape s : shapes) {
    Rectangle clip = s.getClip();
    painter.setClip(clip);
    s.paint(painter);
  }

  // Restore graphics origin
  painter.translate(-x, -y);
}
```

Diagramas de Secuencia (Implementación)



Lazo / Repetición Explícito de más de una instrucción

Valor de Retorno

Mientras / para [condición]

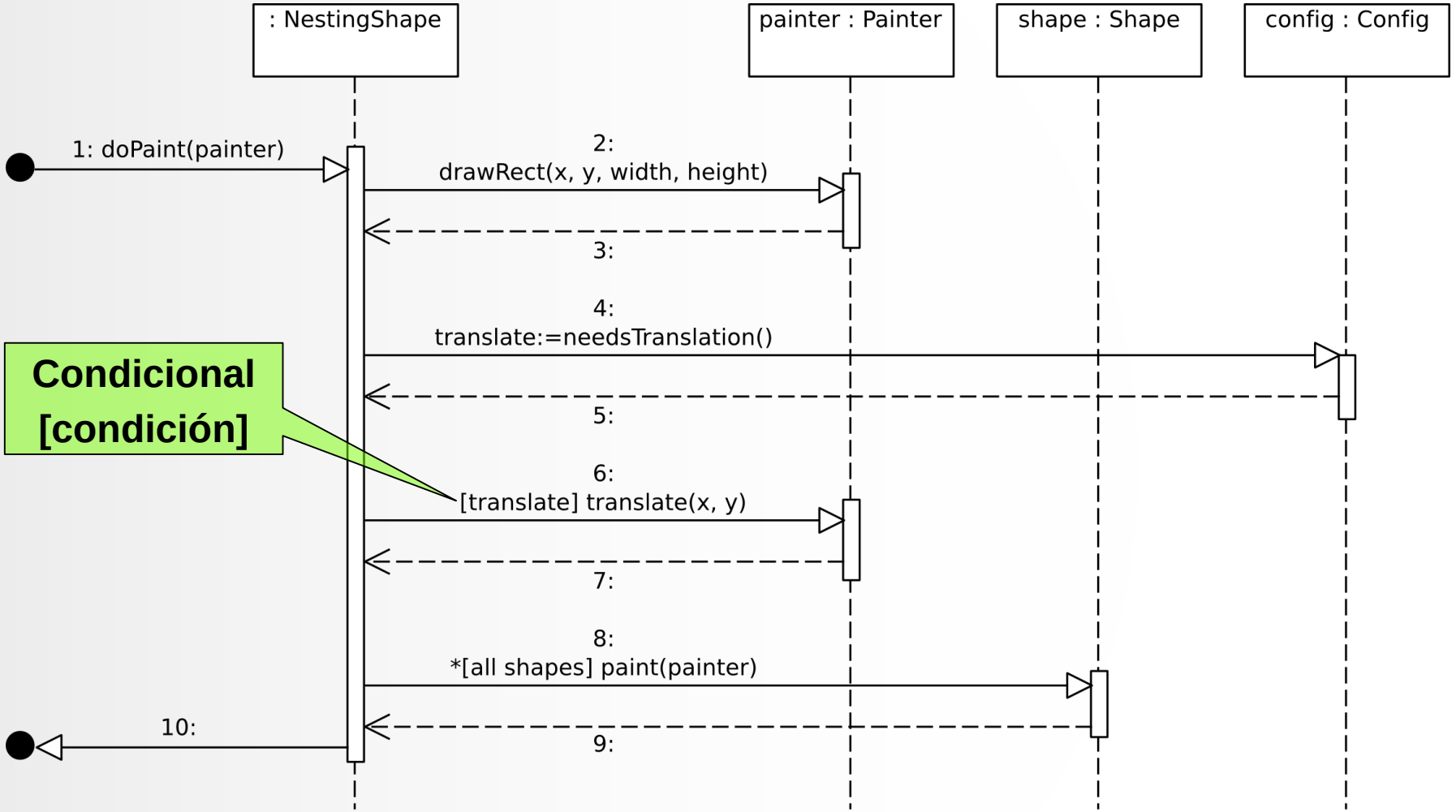
Marco Compuesto

Recordar Etiquetas

Diagramas de Secuencia (Implementación)

```
protected void doPaint(Painter painter, Config config) {
  painter.drawRect(x, y, width, height);

  // Cause painting of shapes to be relative to this shape
  boolean translate = config.needsTranslation();

  if (translate) {
    painter.translate(x, y);
  }

  for (Shape s : shapes) {
    s.paint(painter);
  }
}
```

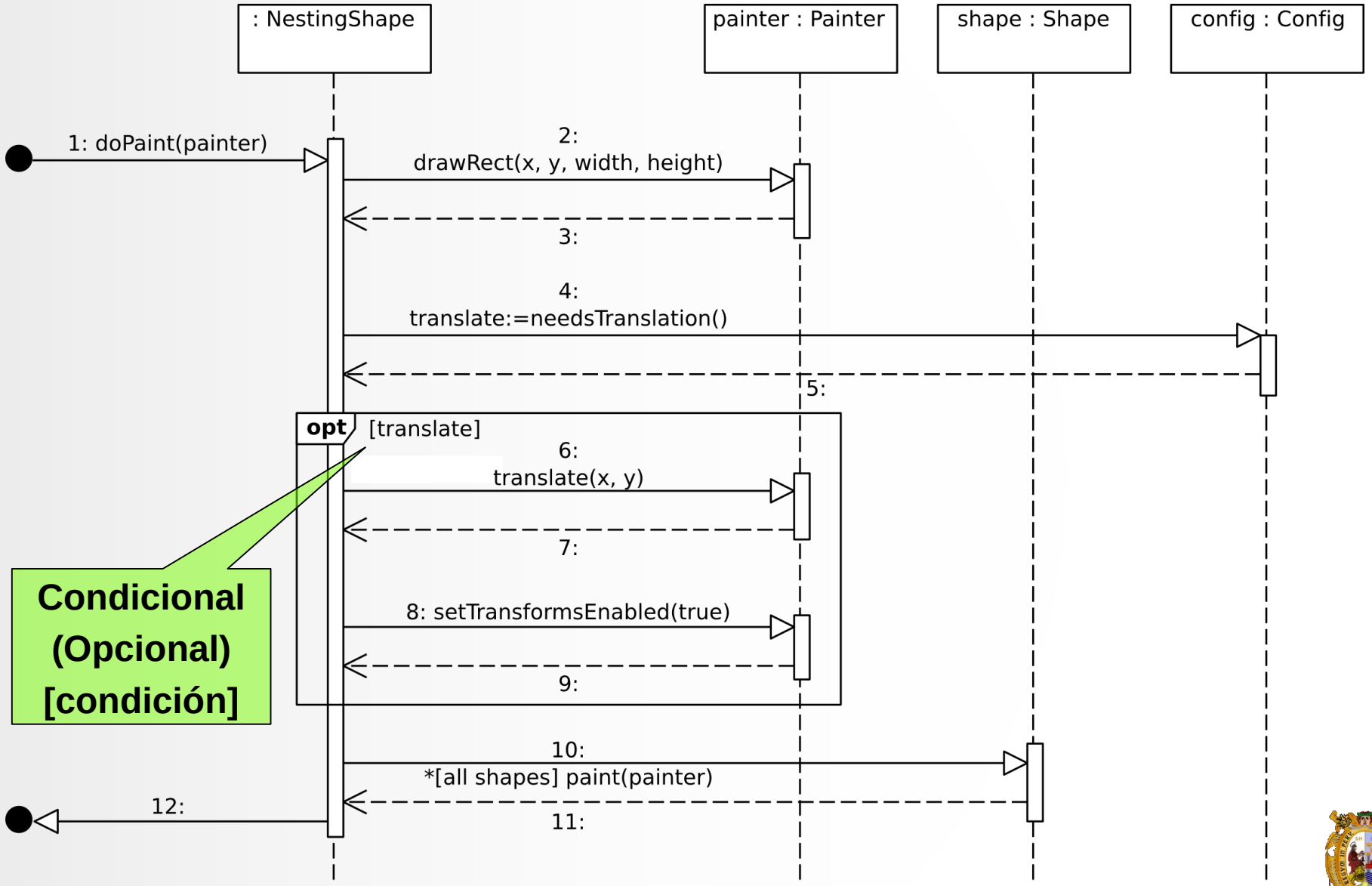
Diagramas de Secuencia (Implementación)



Diagramas de Secuencia (Implementación)

```
protected void doPaint(Painter painter, Config config) {
  painter.drawRect(x, y, width, height);

  // Cause painting of shapes to be relative to this shape
  boolean translate = config.needsTranslation();

  if (translate) {
    painter.setTransformsEnabled(true);
    painter.translate(x, y);
  }

  for (Shape s : shapes) {
    s.paint(painter);
  }
}
```

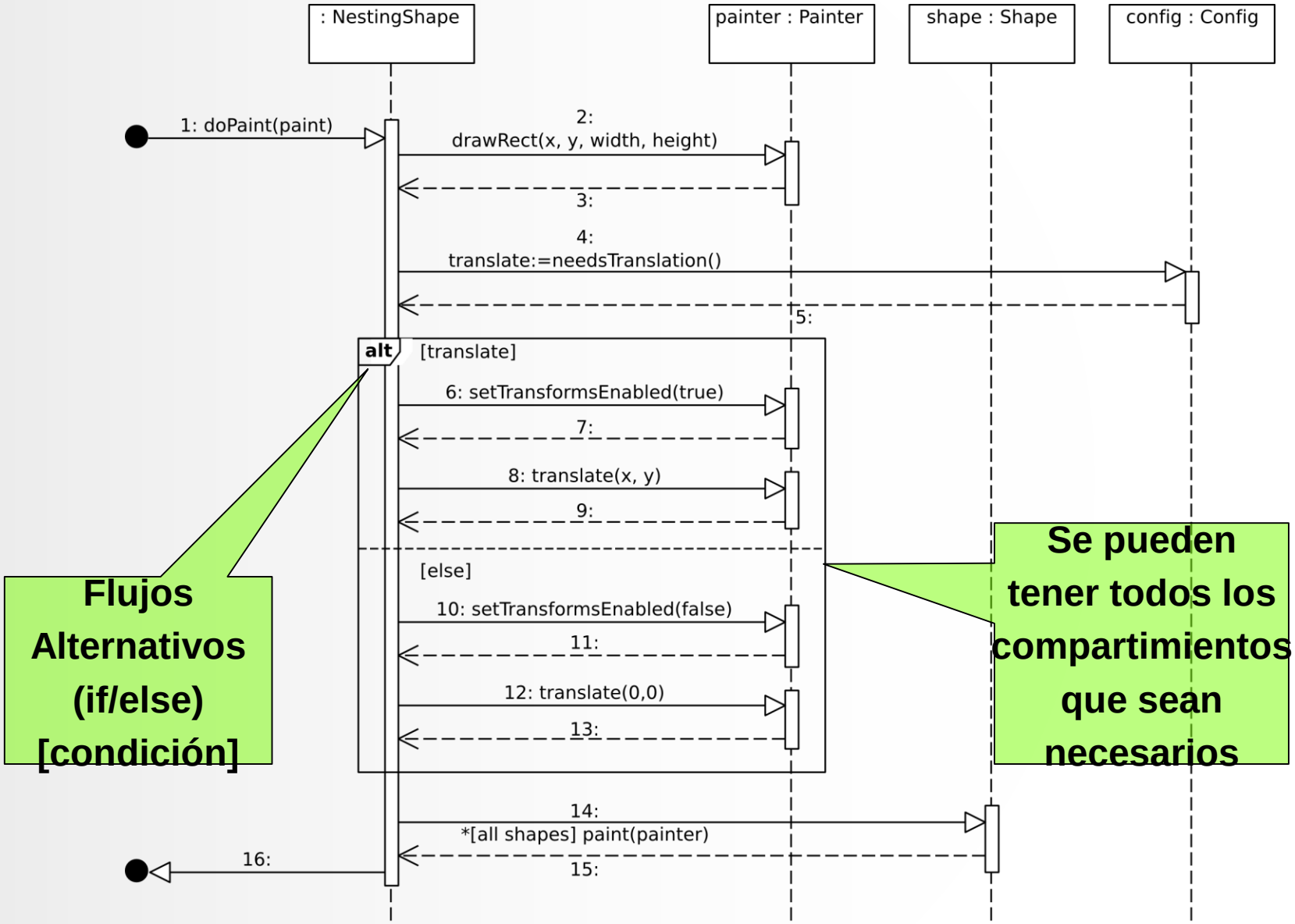
Diagramas de Secuencia (Implementación)



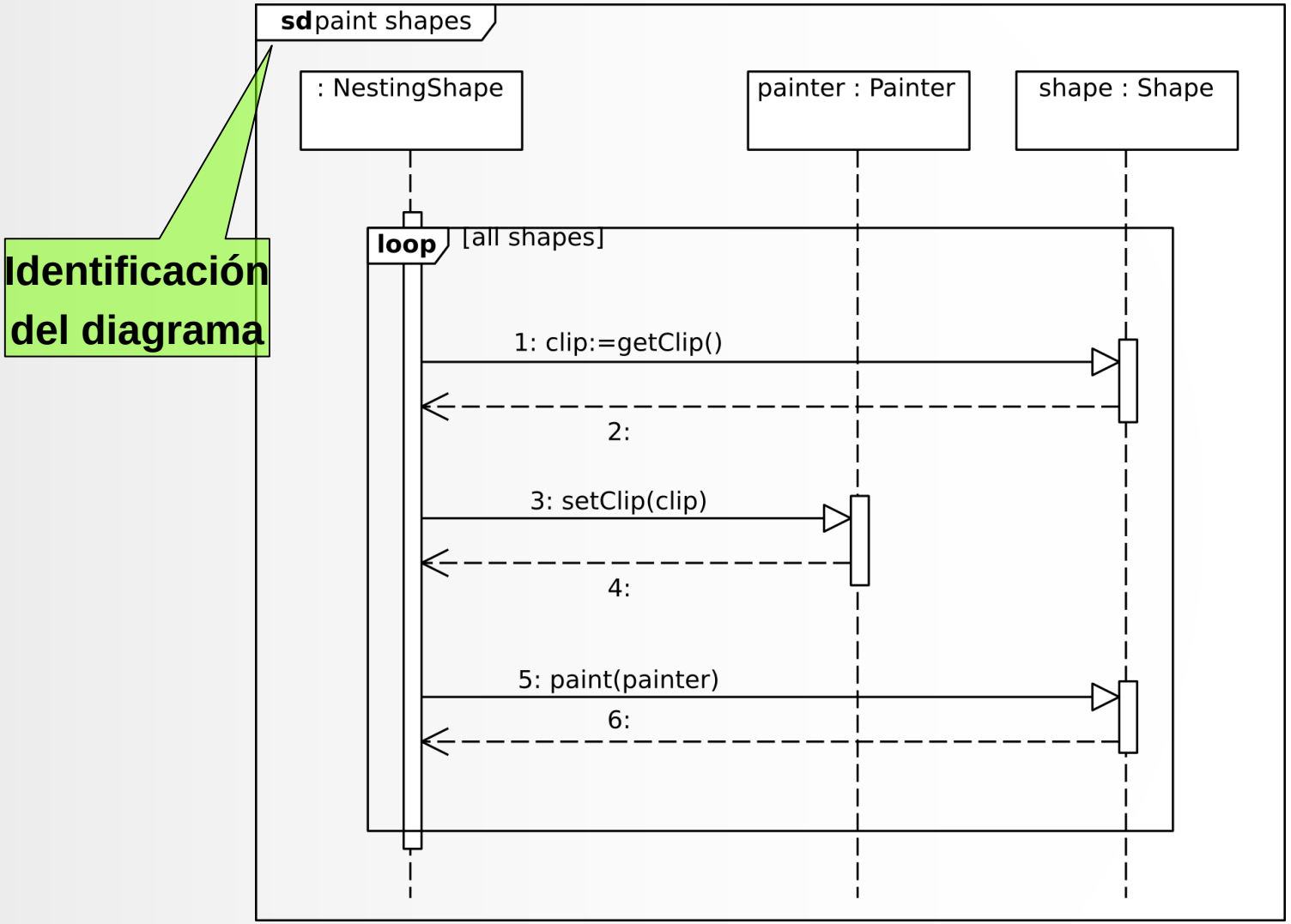
Diagramas de Secuencia (Implementación)

```
protected void doPaint(Painter painter, Config config) {
  painter.drawRect(x, y, width, height);

  // Cause painting of shapes to be relative to this shape
  boolean translate = config.needsTranslation();

  if (translate) {
    painter.setTransformsEnabled(true);
    painter.translate(x, y);
  } else {
    painter.setTransformsEnabled(false);
    painter.translate(0, 0);
  }

  for (Shape s : shapes) {
    s.paint(painter);
  }
}
```

Diagramas de Secuencia (Implementación)



Diagramas de Secuencia (Implementación)



Diagramas de Secuencia (Implementación)