

# **Análisis y diseño orientación a objetos. Diagramas de comportamiento**

Diagramas de casos de uso

Diagramas de secuencia

Diagrama de comunicación

# 1. Introducción

En este último tema vamos a centrarnos en cómo modelar lo que sucede en un sistema de software por medio de **diagramas de comportamientos**.

Los diagramas de clases nos dan información sobre la estructura estática del sistema, no nos dan **información sobre el comportamiento dinámico** del mismo. Para modelar esta información utilizamos los diagramas de comportamiento.

Dentro de estos diagramas podemos encontrar los de **casos de uso, actividad, estado e interacción**. Los diagramas de interacción incluyen el diagrama de secuencia, de comunicación, de tiempos y de vista de interacción.

Los diagramas de comportamiento mostrarán, como su propio nombre indica, el comportamiento de un sistema. Se clasifican en:

| DIAGRAMA                        | RESUMEN  |
|---------------------------------|--|
| Diagrama de casos de uso        | Describe el comportamiento del sistema desde el punto de vista de un usuario que interactúa con él |
| Diagrama de actividad           | Parecido a diagramas de flujo, muestra pasos, puntos de decisión y bifurcaciones                   |
| Diagrama de estado              | Muestra estados de un objeto y transiciones de un estado a otro                                    |
| Diagrama de secuencia           | Muestra interacción de unos objetos con otros  |
| Diagrama de comunicación        | Interacciones entre elementos en tiempo de ejecución   |
| Diagrama de tiempos             | Define comportamiento de objetos en una escala de tiempo   |
| Diagrama de vista e interacción | Cooperación entre otros diagramas de interacción   |

## 2. Diagramas de casos de uso. Actores, escenario, relación de comunicación.

Los casos de uso van a modelar el sistema desde el punto de vista del usuario.

Con esta herramienta vamos a poder obtener los requisitos de software en la fase de análisis de un proyecto. Tendrá que cumplir los siguientes objetivos:

- Definir los requisitos funcionales y operativos del sistema, diseñando un escenario que nos haga más fácil describir cómo se usará el sistema.
- Proporcionar una descripción clara de cómo el usuario interactúa con el sistema y viceversa.
- Facilitar una base para la validación de las pruebas.

Un caso de uso se escribe en lenguaje sencillo y sin tecnicismos para que todos los participantes del proceso lo entiendan. El conjunto completo de casos de uso, especifican todas las diferentes formas de utilizar el sistema y por tanto definen todo el comportamiento requerido del mismo.

### 2.1. Elementos del diagrama de casos de uso

Usando UML podemos crear una representación visual de los casos de uso llamada diagrama de casos de uso.

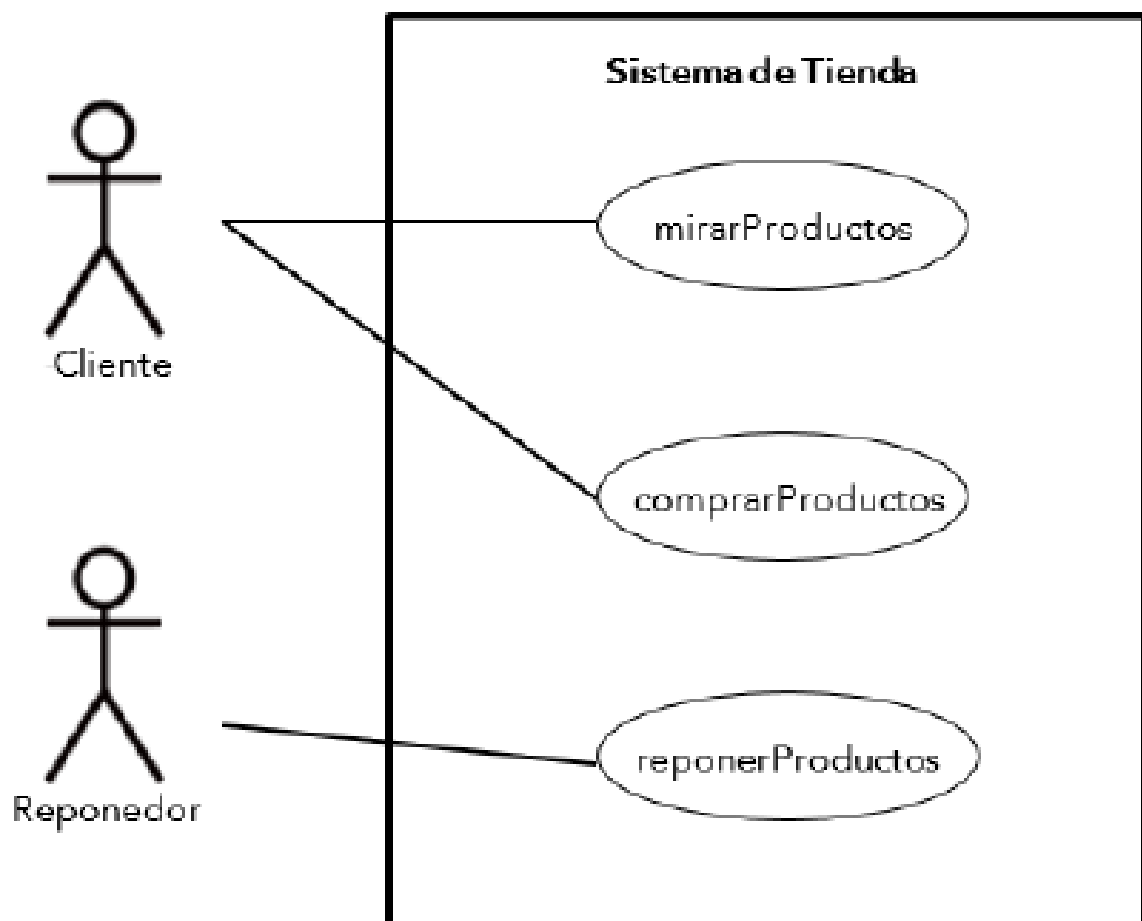
Los componentes de un diagrama de casos de uso son:

– **Actores:** cualquier cosa que interactúa con el sistema y es externo a él. No tiene que ser una persona, puede ser un dispositivo u otro sistema. Se representa con un monigote y un nombre debajo.

– **Casos de uso:** representa una unidad funcional del sistema que realizará una orden de algún agente externo, tanto de un agente como de otro caso de uso. Será iniciado por un *actor* y otros actores podrán participar de él. Se representan con un óvalo o eclipse y una descripción textual.

– **Relaciones:** existen varios tipos. La más común es entre actores y casos de uso representada con una línea continua.

– También podemos tener un rectángulo que delimite el sistema.



## 2.2. Identificar a actores

Los casos de uso siempre serán iniciados por actores que pueden solicitar o modificar información del sistema. El nombre debe coincidir con el objetivo del actor principal que será el que normalmente comience el caso de uso.

Los actores son entidades externas que van a interactuar con el sistema. Normalmente son personas, pero pueden ser otros sistemas o incluso dispositivos. Para poder interactuar con el sistema hay que conocer la información de cada elemento para saber qué y quién interactúa con el sistema y qué rol tendrá cuando se interactúe con él. Es necesario tener en cuenta los siguientes puntos a la hora de definir los actores:

- Serán siempre externos al sistema.
- Van a interactuar directamente con el sistema.
- Representan roles de personas o elementos que desempeñan en relación con el sistema.
- Necesitan un nombre que describa la función que desempeñan.

– Una misma persona o elemento puede desempeñar varios roles como actores distintos.

Ejemplo tienda online.

Actores: usuario normal, usuario registrado, administrativo que comprueba el stock, realiza pedidos y da de alta productos, el encargado de almacén que tramita los pedidos, etc.

## 2.3. Identificar casos de uso

Para identificarlos será necesario entender el funcionamiento del sistema y lo que quiere hacer. Para ello tendremos que **buscar los actores que participan y saber cómo lo usarán.**

Para documentar los casos de uso, podremos hacerlo a través de una plantilla que nos describa lo que hace el actor y lo que ocurre cuando se interactúa con dicho sistema.

**Una plantilla sencilla podrá ser:**

- **Nombre** del caso de uso.
- **ID del** caso de uso.
- Pequeña **descripción** de lo que se espera del caso de uso.
- **Actores implicados.** Existen principales y secundarios. Los primeros activan el sistema y los segundos usan el caso de uso una vez iniciado.
- **Precondiciones.** Serán las condiciones que se deberán cumplir antes de que empiece el caso de uso.
- **Curso normal.** Pasos del caso de uso para que finalice correctamente.
- **Postcondiciones.** Las condiciones que se deberán cumplir cuando finalice el caso de uso.
- **Alternativas.** Errores o excepciones.

Hacer ejemplos pg 266-269

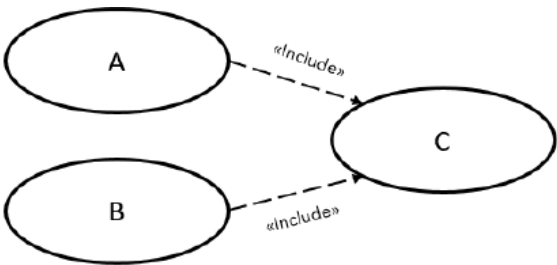
## 2.4. Relaciones en un diagrama de casos de uso

Podremos encontrar **varios tipos** de relaciones:

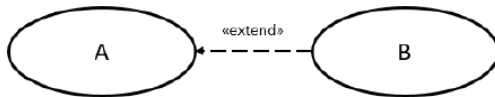
| RELACIÓN                            | FUNCIÓN   | NOTACIÓN              |
|-------------------------------------|---|-----------------------|
| Asociación                          | Línea de comunicación entre actor y caso de uso   | —————                 |
| Extensión<br><<extend>>             | Especifica que el comportamiento de un caso de uso es diferente según las circunstancias. Apuntará a la relación de caso de uso que extenderá | <<extend>><br>----->  |
| Generalización de casos de uso      | Generalización entre clases. El caso de uso hijo hereda el comportamiento y significado del padre.  | —————>                |
| Inclusión <<include>><br>o <<uses>> | Permite que un caso de uso base incluya el comportamiento de otro caso de uso   | <<include>><br>-----> |

Suele ocurrir que las clases <<extend>> e <<include>> se confunden. Para aclarar el funcionamiento de cada relación vamos a especificar el funcionamiento de cada una:

**Include:** Un caso de uso base incorpora explícitamente el comportamiento de otro en algún lugar de su secuencia. La relación de inclusión sirve para enriquecer un caso de uso con otro y compartir una funcionalidad común entre varios casos de uso. Si tenemos, por ejemplo, dos casos de uso X e Y que poseen una serie de pasos en común, podrán ponerse esos pasos en un mismo caso de uso Z, y X e Y lo incluyen para usarlo.

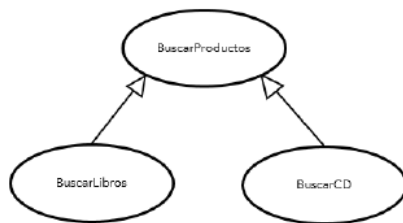


**Extend:** se usará esta relación cuando un caso de uso extiende la funcionalidad de otro caso de uso. Si tenemos un caso de uso Y extenderá la funcionalidad del caso de uso X añadiendo algunos pasos.



En este caso, el caso de uso extendido no sabe nada del caso de uso que lo extiende. Por tanto, el caso de uso de extensión (Y) no será indispensable que ocurra, y si lo hace, ofrece un valor extra (extiende) al objetivo original del caso de uso base

La **relación de generalización** se usa cuando se poseen uno o más casos de uso que son especificaciones de un caso de uso más general.



Ver ejemplos pg 271-280

**HASTA AQUÍ ENTRA PARA EL EXÁMEN**

### 3. Diagramas de interacción:

Estos diagramas **describen el intercambio de secuencias de mensajes entre las partes de un sistema**, es decir muestran el **flujo de control** a través de varios objetos. Dentro de estos tenemos dos:

- El de **secuencia** que se centra en la **ordenación temporal** de los mensajes.
- El de **comunicación** centrado en la **organización estructural** de los objetos que envían y reciben mensajes.

#### 3.1 Diagramas de secuencia.


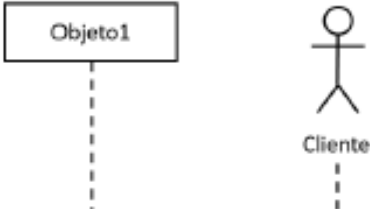

El **diagrama de secuencia de sistema** nos mostrará gráficamente los eventos que fluyen de los actores del sistema. Partimos de los casos de uso elaborados en la etapa de análisis. El diagrama tendrá **dos dimensiones: la dimensión vertical, que representa el tiempo; y la dimensión horizontal, que representa los roles de la interacción.**

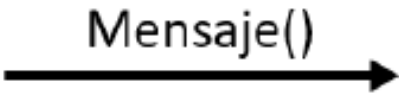
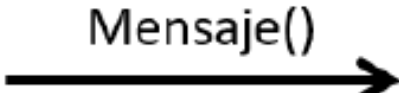
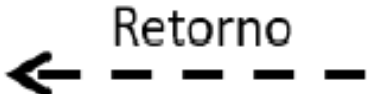

Cada **rol** será representado por un **rectángulo** en la parte superior del diagrama, y cada uno tendrá una línea vertical llamada **línea de vida**, la cual describe la interacción a lo largo del tiempo. Cada línea vertical tendrá **flechas horizontales** que mostrarán la interacción, y encima de ellas habrá un mensaje.

**Es importante ver la secuencia porque nos indicará el orden en que van ocurriendo los eventos.** A veces la secuencia puede ir acompañada de una descripción del curso normal de eventos del caso de uso.



Los elementos principales serán:

| SÍMBOLO       | FUNCIÓN   | NOTACIÓN   |
|---------------|---|--|
| Marco         | Da borde visual al diagrama de secuencia. A la izquierda del marco se escribe la etiqueta sd seguida de un nombre |    |
| Línea de vida | Representa un participante durante la interacción   |    |
| Actor         | Representa el papel desempeñado por un usuario  |  |

|            |   |   |
|------------|---|---|
| Mensaje    | Mensaje síncrono  |   |
|            | Mensaje asíncrono   |   |
|            | Mensaje de retorno  |   |
| Activación | Opcionales. Representa el tiempo durante el cual se ejecuta una función. Se suele poner cuando está activo un método. |  |

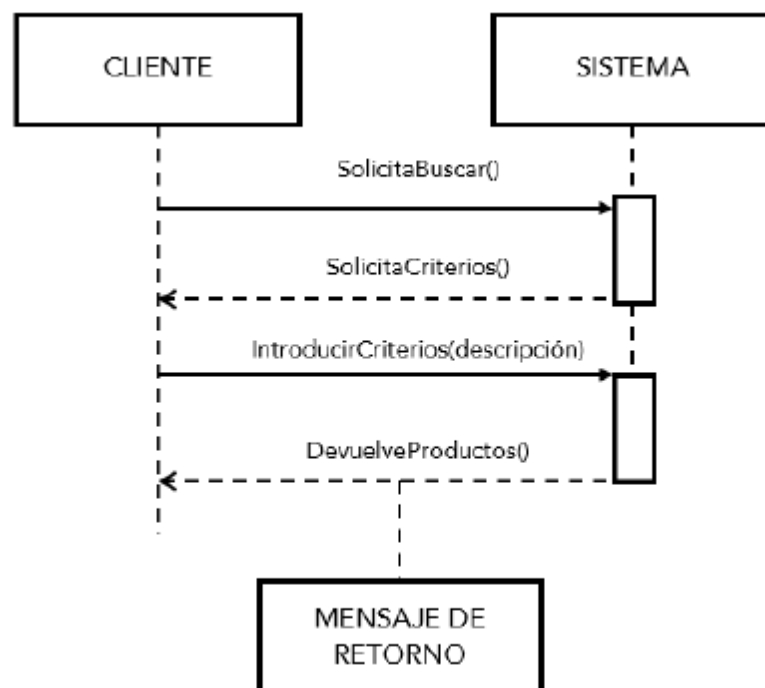
**Los mensajes** representan la comunicación entre participantes y se dibujará con una **flecha** que irá dirigida desde el participante que lo envía hasta el que lo ejecuta. **Tendrá un nombre, acompañado o no de parámetros.**

– **Mensaje síncrono**: cuando se envía un mensaje a un objeto, no se recibe el control hasta que el objeto receptor ha finalizado la ejecución.

– **Mensaje asíncrono**: cuando el emisor que envía un mensaje asíncrono continúa con su trabajo después de ser enviado, es decir, no espera que el receptor finalice la ejecución del mensaje. Su utilización la podemos ver en sistemas multi-hilos donde se producen procesos concurrentes.

– **Mensaje de retorno**: representa mensaje de confirmación. Su uso es opcional.

En estos diagramas hemos visto solamente eventos que fluyen desde el cliente hacia el sistema. En la dimensión horizontal solo se incluían esos dos roles. Dado que trabajamos en un sistema orientado a objetos en el que tenemos varias clases y varios objetos que se comunican unos con otros, las clases y los objetos se representarán en la dimensión horizontal.

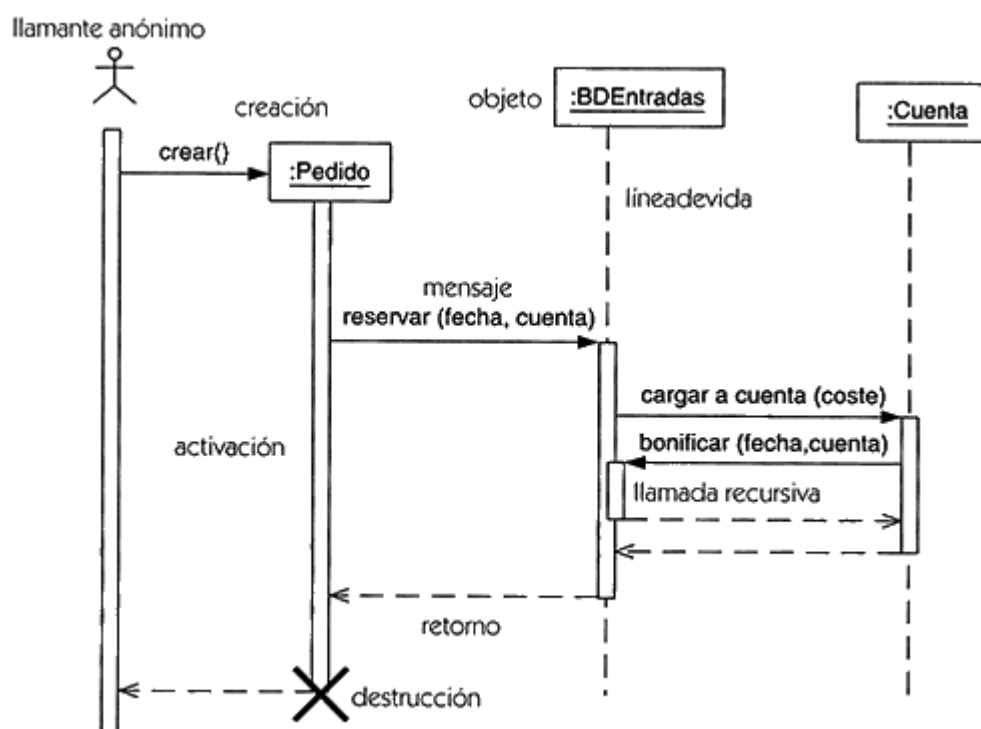


- CREACIÓN Y DESTRUCCIÓN DE OBJETOS**

En el diagrama de secuencia podemos ver la creación y destrucción de objetos.

La creación se representa mediante un mensaje que termina en el objeto que será creado. Este mensaje puede llevar la identificación <<create>>.

La destrucción finalizará la línea de vida del objeto y se representa mediante una X grande en su línea de vida. El mensaje puede llevar la identificación <<destroy>>.



La autodelegación o mensaje reflexivo es un mensaje que un objeto se envía a sí mismo. Por ello, la flecha del mensaje de vuelta regresa a la misma línea de vida.

## ALTERNATIVAS Y BUCLES

En estos tipos de diagramas podemos introducir extensiones para dar soporte a bucles y alternativas. Podemos llamarlas fragmentos combinados, y las hay de varios tipos, entre ellos las alternativas y los bucles.

Se pueden representar mediante un marco o caja los eventos que se repiten. Podemos modelar varios tipos de alternativas:

- Usando operador *opt* seguido de una condición. Si esa condición se cumple, el contenido se ejecuta.
- Usando operador *alt*, seguido de varias condiciones y a final la palabra clave *else*. Se dividirá en varias zonas según las condiciones que haya. La parte *else* se ejecutará si no se ejecuta ninguna condición.

Si queremos representar un bucle lo haremos con el operador *loop*, seguido de una condición. Lo que está encerrado en el marco se ejecutará siempre que dicha condición se cumpla.






## 3.2 Diagramas de colaboración. Objetos, mensajes.

En el diagrama de colaboración, cada objeto se representa con una caja, las relaciones entre los objetos con líneas y los mensajes con flechas que indican la dirección. **Este diagrama muestra los objetos junto con los mensajes que se envían entre ellos.** Se representará la misma información que en el diagrama de secuencia. De hecho, hay herramientas que permiten convertir un diagrama de secuencia en un diagrama de colaboración y viceversa.

Sin embargo, este diagrama, a diferencia del anterior, **se centrará en el contexto y la organización general de los objetos que envían y reciben mensajes.** También muestra los objetos y sus relaciones, es decir, los mensajes que se envían entre sí.

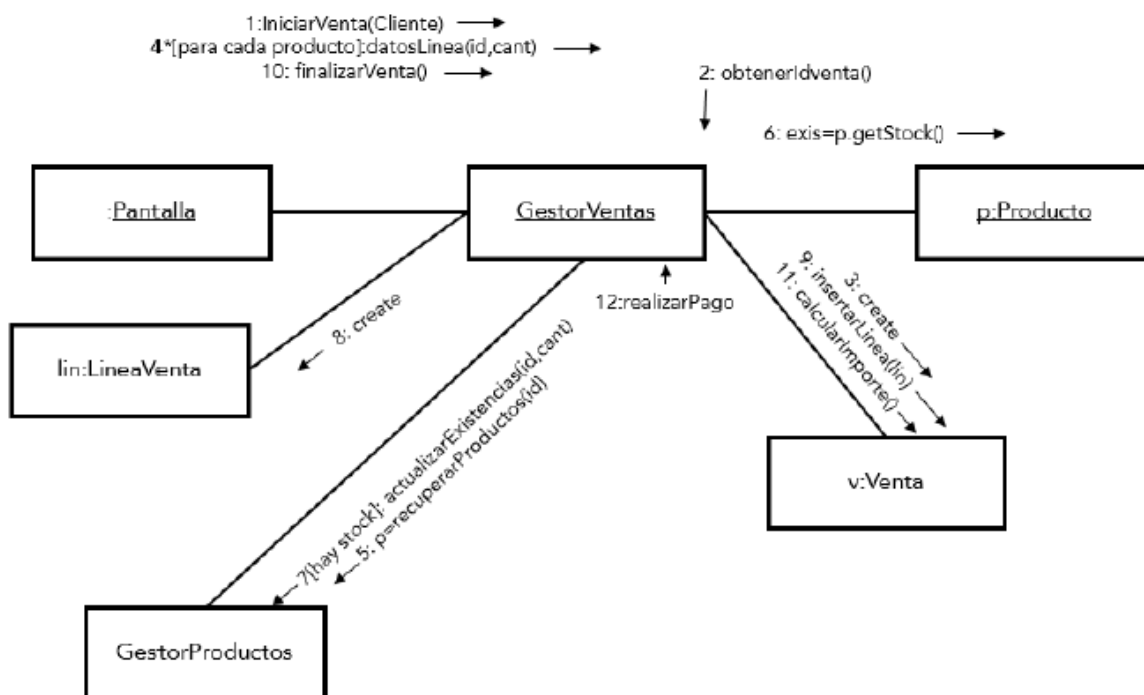
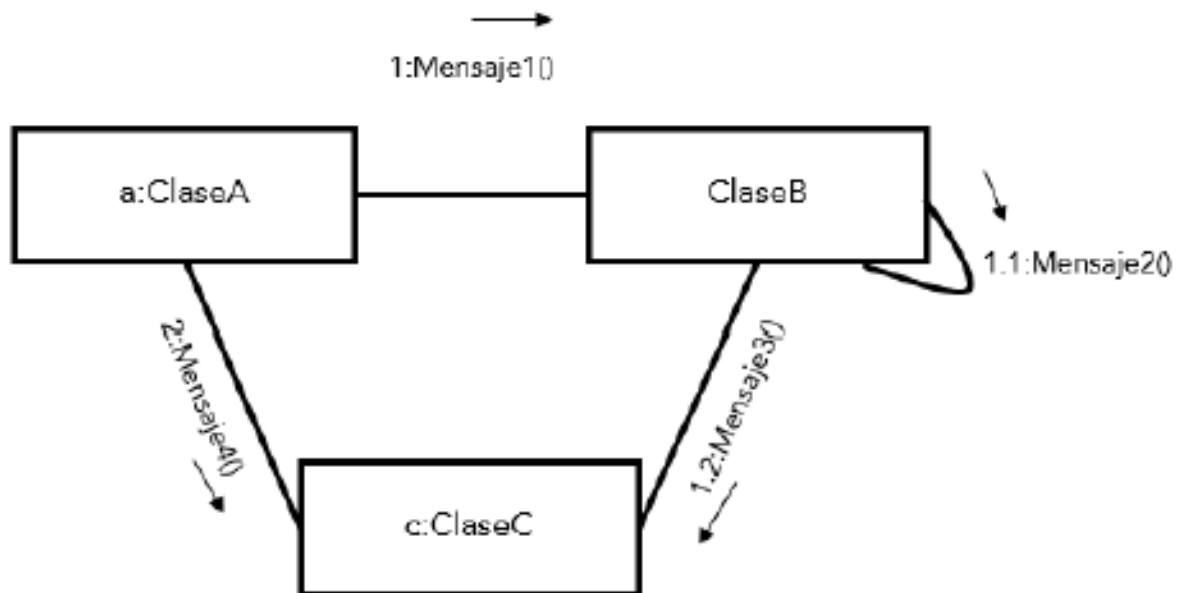
El diagrama de colaboración posee los siguientes elementos:

| SÍMBOLO         | FUNCIÓN   | NOTACIÓN   |
|-----------------|---|--|
| Objetos o roles | Se representa con un rectángulo que contiene el nombre y la clase del objeto en el siguiente formato objeto:Clase.                              |  |
| Enlaces         | Arcos del grafo que conecta ambos objetos. Se podrán mostrar muchos mensajes en un mismo enlace, pero cada uno con un número de secuencia único |  |

|                     |  |   |
|---------------------|--|---|
| Mensajes            | Se representa mediante una flecha dirigida con un nombre y un número de secuencia  | 1:Mensaje()  |
| Número de secuencia | Indica el orden de un mensaje dentro de la iteración (repetición). Comenzará con el 1 y se incrementará conforme se envíen mensajes. El primer mensaje no lleva número de secuencia. |   |
| Iteración           | Se representa colocando un * después del número de secuencia y una condición encerrada entre corchetes.  |   |
| Alternativa         | Se indican con condiciones entre corchetes. Los caminos alternativos tendrán el mismo número de secuencia, seguido del número de subsecuencia.                                       |   |
| Anidamiento         | Se puede mostrar el anidamiento de mensajes con números de secuencia y subsecuencia.   |   |

Para numerar los mensajes se puede usar varios esquemas:

- Numeración simple: comienza en 1 y se incrementa una unidad y no hay nivel de anidamiento.



- Numeración decimal: tiene varios subíndices para indicar anidamiento de operaciones.

