TEMA 6

INDICE

[ENTORNOS DE DESARROLLO]

Entornos de desarrollo del curso "*Desarrollo de Aplicaciones Web*"

DISEÑO ORIENTADO A OBJETOS. ELABORACIÓN DE DIAGRAMAS DE COMPORTAMIENTO.

CASO PRÁCTICO

En BK Programación continúan inmersos en el mundo de UML. A pesar de que han trabajado duro y han aprendido bastante acerca de este lenguaje de especificación, Ada se ha dado cuenta de que apenas han empezado a arañar la superficie de todas las posibilidades que les ofrece. De momento ya saben como crear un diagrama de clases bastante completo y como analizar un problema propuesto, sin embargo hay muchos aspectos del problema que no pueden modelar todavía, por ejemplo con solo el diagrama de clases no pueden saber qué se espera del sistema que van a construir, o en qué se deben basar para codificar los métodos, o simplemente, ¿Cómo colaboran los objetos de las clases que han creado para hacer alguna tarea que sea útil?

Ada decide que no pueden parar ahora, y que hay que hacer un esfuerzo final para que los conocimientos del equipo sean globales y puedan enfrentarse a cualquier desarrollo software con solvencia.

Al momento, Ada pone a su equipo manos a la obra.

1. Diagramas de comportamiento.

Caso práctico
—Compañeros, creo que ahora no debemos conformarnos con modelar diagramas de clase y nada más, esto no nos da posibilidades de incluir ninguna información acerca del comportamiento de nuestro sistema. ¿Cómo especificamos la funcionalidad? O ¿qué acciones se realizan?, o ¿las restricciones? Necesitamos seguir estudiando diagramas que nos ayuden a especificar la dinámica del sistema.
¿Empezamos ahora mismo?

En el tema anterior vimos como crear un diagrama de clases para un problema determinado, esto nos ayuda a ver el problema con otra perspectiva y descubrir información nueva, sin embargo no tiene en cuenta elementos como la creación y destrucción de objetos, el paso de mensajes entre ellos y el orden en que deben hacerse, qué funcionalidad espera un usuario poder realizar, o como influyen elementos externos en nuestro sistema. Un diagrama de clases nos da información estática pero no dice nada acerca del comportamiento dinámico de los objetos que lo forman, para incluir éste tipo de información utilizamos los diagramas de comportamiento que incluyen:

- ✓ Diagramas de casos de uso.
- ✓ Diagramas de actividad.
- ✓ Diagramas de máquinas de estado.
- ✓ Diagramas de interacción.
  - ➔ Diagramas de secuencia.
  - ➔ Diagramas de comunicación.
  - ➔ Diagramas de interacción.
  - ➔ Diagramas de tiempo.

En el siguiente enlace tienes una descripción y algunos ejemplos de todos los diagramas UML, puedes usarlo como complemento a lo que vamos a ver en la unidad.
http://jms32.eresmas.net/tacticos/UML/UMLIndex.html

2. Diagramas de casos de uso.

CASO PRÁCTICO.
—Empezaremos por el principio. Cuando estamos diseñando software es esencial saber cuales son los requerimientos del sistema que queremos construir, y necesitamos alguna herramienta que nos ayude a especificarlos de una manera clara, sistemática, y que nuestros clientes puedan entender fácilmente, ya que es imprescindible que nos pongamos de acuerdo en lo que realmente queremos hacer. ¿Alguna idea?
—¿No bastaría con hacer una lista de requerimientos y describirlos exhaustivamente?
—No, una descripción textual puede inducir a errores de interpretación y suele dejar cabos sueltos, y no contempla otra información, como quien realiza las acciones que describimos, por ejemplo. Necesitamos algo más específico.
Lo que Ada necesita son los diagramas de Casos de uso.

Los diagramas de casos de uso son un elemento fundamental del análisis de un sistema desde la perspectiva de la orientación a objetos porque resuelven uno de los principales problemas en los que se ve envuelto el proceso de producción de software: la falta de comunicación entre el equipo de desarrollo y el equipo que necesita de una solución software. Un diagrama de casos de uso nos ayuda a determinar **QUÉ** puede hacer cada tipo diferente de usuario con el sistema, en una forma que los no versados en el mundo de la informática o, más concretamente el desarrollo de software, pueda entender.

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los **requisitos funcionales del sistema** (*acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados. Suelen venir definidos por el cliente*), es decir, representan las funciones que un sistema puede ejecutar.

Un diagrama de casos de uso es una visualización gráfica de los requisitos funcionales del sistema, que está formado por casos de uso (se representan como elipses) y los actores que interactúan con ellos (se representan como monigotes). Su principal **función** es dirigir el proceso de creación del software, definiendo qué se espera de él, y su **ventaja** principal es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente.

Los diagramas de casos de uso se crean en las primera etapa de desarrollo del software, y se enmarcan en el proceso de análisis, para definir de forma detallada la funcionalidad que se espera cumpla el software, y que, además, se pueda comunicar fácilmente al usuario, pero, ¿termina aquí su función?
En absoluto, de los diagramas de casos de uso se desprenden otros (normalmente se realiza antes que el diagrama de clases) que describen tanto la estructura del sistema como su comportamiento, lo que influye directamente en la implementación (*Paso a código de las especificaciones que se han definido durante la fase de análisis y diseño de un sistema software*) del sistema y en su arquitectura (*Conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema, y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores y el estilo arquitectónico que guía esta organización: estos elementos y sus interfaces, sus colaboraciones y su composición*) final. Por otra parte al describir específicamente qué se espera del software también se usa en la fase de prueba, para verificar que el sistema cumple con los requisitos funcionales, creándose muchos de los casos de prueba (pruebas de caja negra *(Se realizan cuando una aplicación es probada usando su interfaz externa sin preocuparnos de la implementación de la misma)*) directamente a partir de los casos de uso.

2.1 Actores

Caso práctico

—Como decía, uno de los principales problemas de una descripción textual es que no permite especificar adecuadamente qué personas o entidades externas interactúan con el sistema. Ahora tenemos una herramienta que tiene esto muy en cuenta...

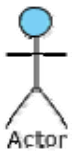
Los actores representan un tipo de usuario del sistema. Se entiende como usuario cualquier cosa externa que interactúa con el sistema. No tiene por qué ser un ser humano, puede ser otro sistema informático o unidades organizativas o empresas.

Siempre hay que intentar independizar los actores de la forma en que se interactúa con el sistema. Por ejemplo, un usuario del sistema puede interpretar diferentes roles según la operación que esté ejecutando, cada uno de estos roles representará un actor diferente, es decir, un actor en un diagrama de casos de uso representa un rol que alguien puede estar jugando, no un individuo particular por lo tanto puede haber personas particulares que puedan estar usando el sistema de formas diferentes en diferentes ocasiones. Suele ser útil mantener una lista de los usuarios reales para cada actor.

Tipos de actores:

- ***Primarios***: interaccionan con el sistema para explotar su funcionalidad. Trabajan directa y frecuentemente con el software.
- ***Secundarios***: soporte del sistema para que los primarios puedan trabajar. Son precisos para alcanzar algún objetivo.
- ***Iniciadores***: no interactúan con el sistema pero desencadenan el trabajo de otro actor.

Los actores se representan mediante la siguiente figura:



Es posible que haya casos de uso que no sean iniciados por ningún usuario, o algún otro elemento software, en ese caso se puede crear un actor “Tiempo” o “Sistema”.

¿Un sistema software externo, como una entidad para la autentificación de claves, podría considerarse como un actor en un diagrama de casos de uso?

- [x] Verdadero
- [ ] Falso

un actor no tiene porqué ser una persona física, es cualquier entidad que interaccione con el sistema

2-2 Casos de uso

Caso práctico

—Vale, pero lo que verdaderamente queremos es identificar la funcionalidad del sistema ¿no?, ¿cómo hace esta herramienta la descripción de la funcionalidad?

Se utilizan casos de uso para especificar tareas que deben poder llevarse a cabo con el apoyo del sistema que se está desarrollando.

Un **caso de uso** especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto. El conjunto de casos de uso forma el "**comportamiento requerido**" de un sistema.

El objetivo principal de elaborar un diagrama de casos de uso no es crear el diagrama en sí, sino la descripción que de cada caso se debe realizar, ya que esto es lo que ayuda al equipo de desarrollo a crear el sistema a posteriori. Para hacer esto utilizamos, sobre todo otros diagramas que permiten describir la dinámica del caso de uso, como el diagrama de secuencia que veremos después, y una descripción textual, en la que se deben incluir, al menos, los siguientes datos (a los que se denomina **contrato**):

- ✓ **Nombre:** nombre del caso de uso.
- ✓ **Actores**: aquellos que interactúan con el sistema a través del caso de uso.
- ✓ **Propósito**: breve descripción de lo que se espera que haga.
- ✓ **Precondiciones**: aquellas que deben cumplirse para que pueda llevarse a cabo el caso de uso.
- ✓ **Flujo normal**: flujo normal de eventos que deben cumplirse para ejecutar el caso de uso exitosamente, desde el punto de vista del actor que participa y del sistema.
- ✓ **Flujo alternativo**: flujo de eventos que se llevan a cabo cuando se producen casos inesperados o poco frecuentes. **No** se deben incluir aquí errores como escribir un tipo de dato incorrecto o la omisión de un parámetro necesario.
- ✓ **Postcondiciones**: las que se cumplen una vez que se ha realizado el caso de uso.

La representación gráfica de un caso de uso se realiza mediante un óvalo o elipse, y su descripción se suele hacer rellenando una o más tablas como la de la imagen (obtenida de la herramienta Visual Paradigm).

Super Use Case			
Author	usuario		
Date	26-ago-2011 13:56:56		
Brief Description			
Preconditions			
Post-conditions			
Flow of Events		Actor Input	System Response
	1		

"Tras comprobar todos los artículos el pedido queda en el almacén a la espera de ser recogido." ¿Dónde incluirías esta afirmación sobre un caso de uso en un contrato?

- ○ En el flujo de eventos normal. -----
- ○ En el flujo de eventos alternativo.
- ○ En las precondiciones.
- ◉ **En las postcondiciones.**

2.3 Relaciones

Caso práctico

—De acuerdo, y ahora ¿Cómo asociamos a los actores con los casos de uso que pueden realizar?

Los diagramas de casos de uso son grafos no conexos en los que los nodos son actores y casos de uso, y las aristas son las relaciones que existen entre ellos. Representan qué actores realizan las tareas descritas en los casos de uso, en concreto qué actores inician un caso de uso. Pero además existen otros tipos de relaciones que se utilizan para especificar relaciones más complejas, como uso o herencia entre casos de uso o actores.

Existen diferentes tipos de relaciones entre elementos:

- ✓ **Asociación**: representa la relación entre el actor que lo inicia y el caso de uso.

- ✓ **Inclusión**: se utiliza cuando queremos dividir una tarea de mayor envergadura en otras más sencillas, que son utilizadas por la primera. Representa una relación de uso, y son muy útiles cuando es necesario reutilizar tareas.
- ✓ **Extensión**: se utiliza para representar relaciones entre un caso de uso que requiere la ejecución de otro en determinadas circunstancias.
- ✓ **Generalización**: se utiliza para representar relaciones de herencia entre casos de uso o actores.

A continuación las vemos con un poco más de detalle.

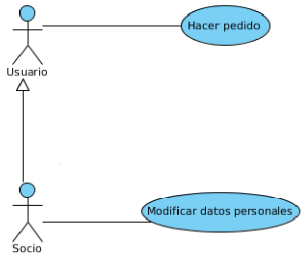
2.3.1 Interacción o asociación.

Hay una asociación entre un actor y un caso de uso si el actor interactúa con el sistema para llevar a cabo el caso de uso o para iniciarlo.

Una asociación se representa mediante un linea continua que une un actor con un caso de uso. Por ejemplo, un usuario de un sistema de venta por Internet puede hacer un pedido, lo que se representa del siguiente modo:

2.3.2 Generalización

Es posible que, igual que con los diagramas de clases, existan casos de uso que tengan comportamientos semejantes a otros que los modifican o completan de alguna manera. El caso base se define de forma abstracta y los hijos heredan sus características añadiendo sus propios pasos o modificando alguno. Normalmente la herencia se utiliza menos en diagramas de casos de uso que en diagramas de clases.



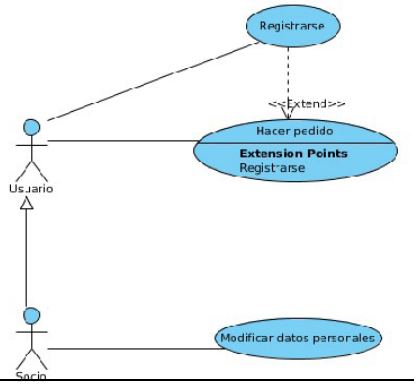
Por ejemplo, el usuario del sistema de venta por Internet puede a su vez darse de alta en la página web para que tengan sus datos registrados a la hora de hacer el pedido, en este caso el usuario es la generalización del socio.

Ambos actores pueden hacer un pedido, pero solo el socio puede modificar sus datos en el sistema.

2.3.3 Extensión.

Se utiliza una relación entre dos casos de uso de tipo "**extends**" cuando se desea especificar que el comportamiento de un caso de uso es diferente dependiendo de ciertas circunstancias.

La principal función de esta relación es simplificar el flujo de casos de uso complejos. Se utiliza cuando existe una parte del caso de uso que se ejecuta sólo en determinadas ocasiones, pero no es imprescindible para su completa ejecución. Cuando un caso de uso extendido se ejecuta, se indica en la especificación del caso de uso como un **punto de extensión**. Los puntos de extensión se pueden mostrar en el diagrama de casos de uso.



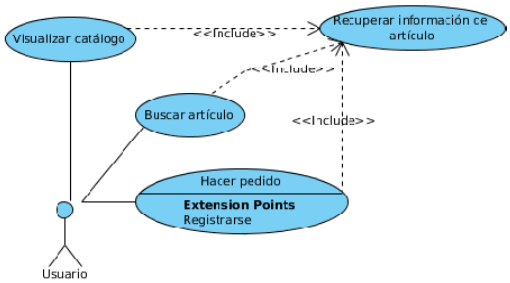
Por ejemplo, cuando un usuario hace un pedido si no es socio se le ofrece la posibilidad de darse de alta en el sistema en ese momento, pero puede realizar el pedido aunque no lo sea.

2.3.4 Inclusión.

Se incluye una relación entre dos casos de uso de tipo "**include**" cuando la ejecución del caso de uso incluido se da en la rutina normal del caso que lo incluye.

Esta relación es muy útil cuando se desea especificar algún comportamiento común en dos o más casos de uso, aunque es frecuente cometer el error de utilizar esta técnica para hacer subdivisión de funciones, por lo que se debe tener mucho cuidado cuando se utilice.

Por ejemplo, a la hora de hacer un pedido se debe buscar la información de los artículos para obtener el precio, es un proceso que necesariamente forma parte del caso de uso, sin embargo también forma parte de otros, como son el que visualiza el catálogo de productos y la búsqueda de un artículo concreto, y dado que tiene entidad por sí solo se separa del resto de casos de uso y se incluye en los otros tres.



Las ventajas de esta asociación son:

- ✓ Las descripciones de los casos de uso son más cortas y se entienden mejor.
- ✓ La identificación de funcionalidad común puede ayudar a descubrir el posible uso de componentes ya existentes en la implementación.

Las desventajas son:

- ✓ La inclusión de estas relaciones hace que los diagramas sean más difíciles de leer, sobre todo para los clientes.

Cuando usamos relaciones de inclusión o extensión no podemos olvidar que los casos de uso extendidos o incluidos deben cumplir con las características propias de un caso de uso, es decir, deben representar un flujo de actividad completo desde el punto de vista de lo que un actor espera que el sistema haga por él, así como no utilizar estas herramientas sólo para descomponer un caso de uso de envergadura en otros más pequeños, piedra angular del diseño estructurado y no del orientado a objetos.

Suponer el siguiente sistema que modela el caso de uso Servir pedido en el que el Empleado de almacén revisa si hay suficientes artículos para hacer el pedido y si todo es correcto, el pedido se embala y se envía:
¿Qué tipo de relación emplearías en el modelo del dibujo?

- ☐ Asociación
- ☐ Generalización
- ☐ Extends
- ◉ **Include**

Así, la consulta de existencias debe realizarse necesariamente y, además, tiene entidad suficiente como para ser un caso de uso por si mismo, que puede usarse en otros casos.

2.4 Elaboración de casos de uso.

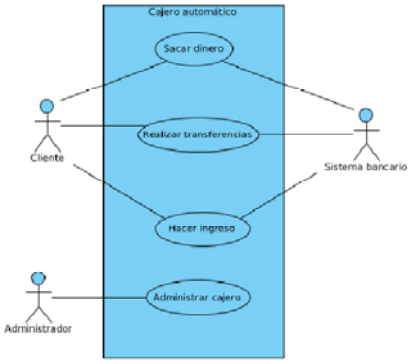
CASO PRÁCTICO

Después de analizar todos los elementos que formen un diagrama de casos de uso el equipo de Ada está preparado para hacer frente a su primer diagrama de casos de uso.

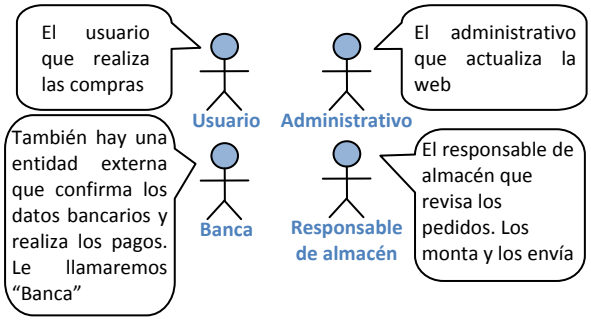
Sea cual sea el tipo de diagrama que estemos creando, cuando lo hacemos realizamos un proceso de abstracción por el cual representamos elementos de la realidad esquemáticamente, y en el diagrama de casos de uso pasa igual, necesitamos abstraer la realidad en un dibujo, en el que representamos qué cosas pueden hacerse en nuestro sistema y quien las va a hacer. Necesitamos diagramas que incluyan suficiente información para que el equipo de desarrollo tome las decisiones más adecuadas en la fase de análisis y diseño para una construcción de software que cumpla con los requerimientos, así como que sean útiles en la fase de implementación en un lenguaje orientado a objetos.

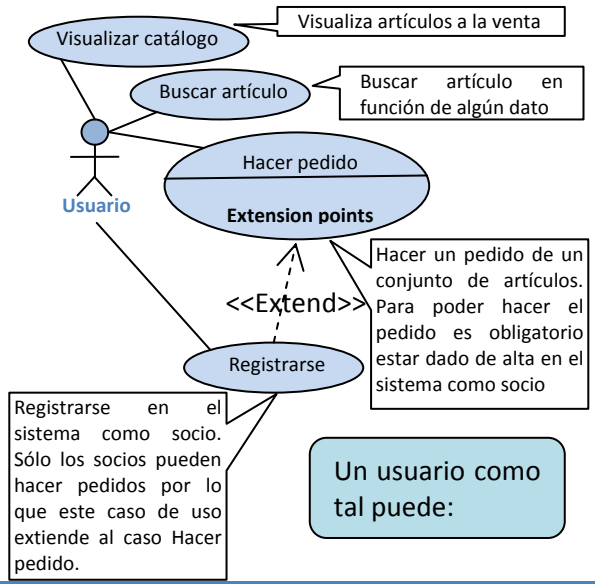
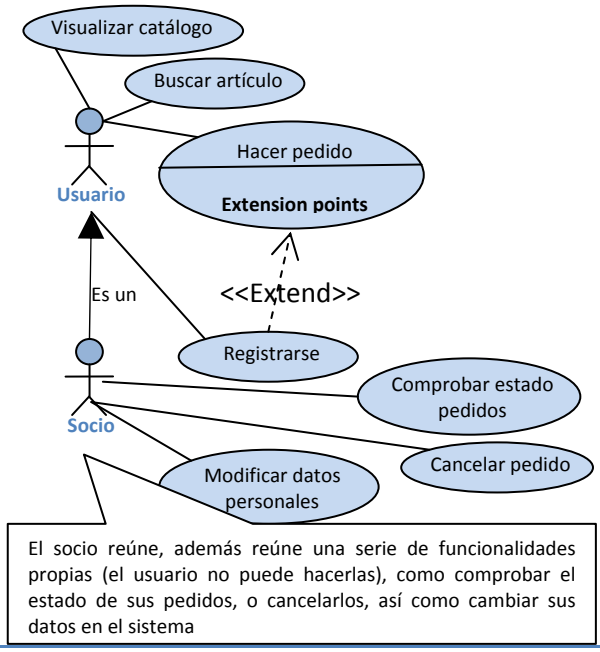
Partiremos de una descripción lo más detallada posible del problema a resolver y trataremos de detectar quien interactúa con el sistema, para obtener los actores diagrama de casos de uso, a continuación buscaremos qué tareas realizan estos actores para determinar los casos de uso más genéricos. El siguiente paso es refinar el diagrama analizando los casos de uso más generales para detectar casos relacionados por inclusión (se detectan fácilmente cuando aparecen en dos o más casos de uso generales), extensión y generalización. Al diagrama generado se le denomina diagrama frontera.

Se conoce como **diagrama frontera** al diagrama de casos de uso que incluye todos los casos de uso genéricos del sistema, que podrán ser desglosados después en nuevos diagramas de casos de uso que los describan si es necesario. Se especifica enmarcando los casos de uso en un recuadro, que deja a los actores fuera.

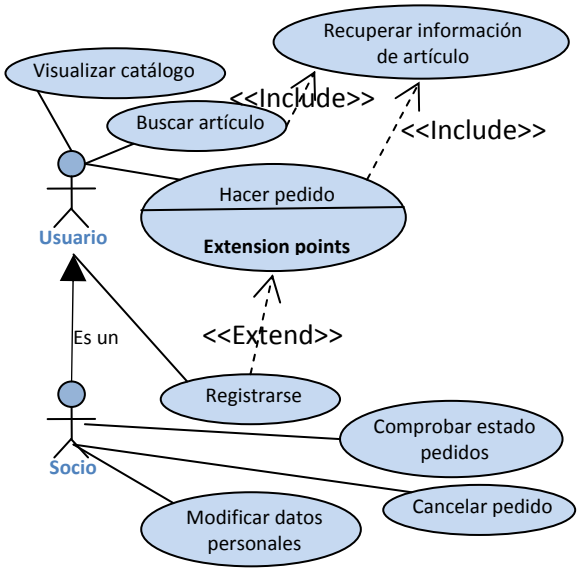


Creación de casos de uso

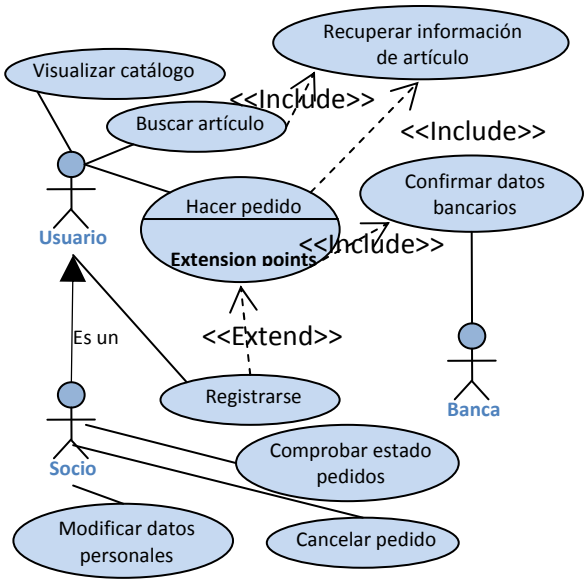
Primeros pasos	Antes de elaborar el diagrama tienes que leer con detenimiento el documento con la especificación del problema a resolver y asegurarte de que entiendes la idea central del problema, crear una tienda virtual en la que se puedan realizar pedidos de los productos a la venta (zapatos). El proceso se centra en el pedido, desde poner a disposición del cliente los artículos en venta, pasando por la selección de artículos a pedir, la cumplimentación de toda la información necesaria para el pedido, pago, confección del pedido, envío y reajuste del stock en almacén, todo ello, a través de la web
Identificar actores	En nuestro sistema ¿Quién interviene? 
Identificar funcionalidades	Para facilitar la creación del diagrama vamos a ir sacando funcionalidades para cada usuario. Debemos recordar que un caso de uso representa una

	interacción de un actor con el sistema, que está relacionado con los requisitos funcionales de la aplicación final y que, en definitiva representa tareas que llevará a cabo el sistema.
Funcionalidad del usuario	Cuando una persona se conecta al sistema lo primero que podrá hacer será **visualizar** el catálogo de la temporada. También puede hacer un **pedido** con uno o varios artículos del catálogo, para ello visualizará los artículos de forma que pueda seleccionar algunos de ellos e indicar la cantidad que quiere comprar. También puede hacer **búsquedas** por datos concretos de artículos. Cualquier persona que acceda al sistema puede **darse de alta** para ser socio. Así mismo, si es socio, podrá **comprobar el estado** de sus pedidos y **cancelarlos**.
	
Al hacerse socio la identidad del usuario con respecto del sistema cambia, por lo que surge un nuevo tipo de actor que hereda de usuario. Un socio es un usuario que se ha registrado en el sistema. Los socios almacenan cierta información, como sus datos personales y bancarios, email, etc.	

Al revisar un poco la funcionalidad de los casos de uso del usuario podemos comprobar que en los casos **Buscar articulo** y **Hacer pedido** es necesario buscar en el sistema y recuperar la información de un artículo del que tenemos algún dato, en el primer caso para obtener todos los datos del artículo buscado y en el segundo para recuperar el precio de los artículos que se añaden al pedido, por lo que extraemos el caso de uso "**Recuperar información de artículo**" que se incluye en los otros dos.



Cuando se realiza el pedido es obligatorio hacer una comprobación de los datos bancarios del cliente, que dependen de una entidad externa, por lo que se añade otro caso de uso para esta función, "Comprobar datos bancarios" que trae de la mano la inclusión del actor Banca.

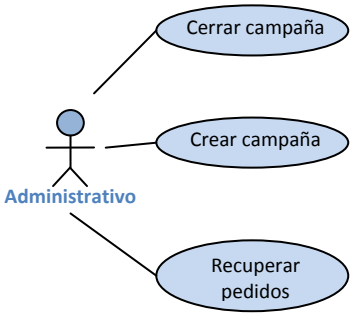


Funcionalidad del administrador web

El objetivo del administrativo web es gestionar los contenidos de la web, en concreto de las diferentes campañas, ya que cada temporada se debe cerrar la campaña antigua, retirando los artículos de la temporada anterior y abrir la temporada nueva, añadiendo sus artículos. Para que se pueda cerrar una temporada es necesario que en el almacén se hayan gestionado todos sus pedidos, por lo que es obligatorio comprobarlo, antes de cerrar.

Se incluye el caso de uso **Recuperar pedidos** en **Crear campaña** por dos motivos:

1. Es una función que puede llevarse a cabo de forma independiente, tanto por el administrador de la web como desde el almacén, y
2. Es de comprobación obligatoria antes de cerrar la temporada.



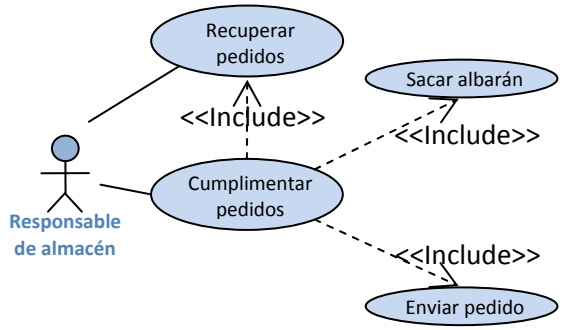
Funcionalidad del responsable de almacén

Es el encargado de leer los pedidos de los usuarios y cumplimentarlos. Ésta será su única función, si bien, es una

función complicada, ya que implica realizar una serie de tareas:

- ✓ Seleccionar el pedido más antiguo
- ✓ Buscar los artículos a servir
- ✓ Empaquetarlos junto con un albarán para el socio
- ✓ Colocarlos en su ruta de envío

El primer paso es **recuperar la lista de pedidos** sin procesar. Esta tarea recupera el pedido más antiguo para ser procesado.
Sacar albarán produce un listado en papel con la información del pedido para el socio.
Enviar pedido es colocar el pedido en la ruta de envío más apropiada para su destino.



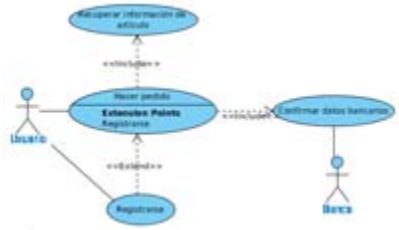
Especificación del problema a modelar.

Descripción del problema: "El tacón de oro".

Los usuarios del sistema navegan por la web para ver los artículos, zapatos, bolsos y complementos que se venden en la tienda. De los artículos nos interesa su nombre, descripción, material, color, precio y stock. De los zapatos nos interesa su número y el tipo. De los bolsos nos interesa su tipo (bandolera, mochila, fiesta). De los complementos (cinturones y guantes) su talla.
Los artículos se organizan por campañas para cada temporada (primavera/verano y otoño/invierno) de cada año.
Los artículos son de fabricación propia, pero, opcionalmente, pueden venderse artículos de otras firmas. De las firmas nos interesa saber su nombre, CIF y domicilio fiscal. La venta de artículos de firma se realiza a través de proveedores, de forma que un proveedor puede llevar varios artículos de diferentes firmas, y una firma puede ser suministrada por más de un proveedor. Los artículos pertenecen a una firma solamente. De los proveedores debemos conocer su nombre, CIF, y domicilio fiscal.
Los usuarios pueden registrarse en el sitio web para hacerse socios. Cuando un usuario se hace socio debe proporcionar los siguiente datos: nombre completo, correo electrónico y dirección.
Los socios pueden hacer pedidos de los artículos. Un pedido está formado por un conjunto de detalles de pedido que son parejas formadas por artículo y la cantidad. De los pedidos interesa saber la fecha en la que se realizó y cuanto debe pagar el socio en total. El pago se hace a través tarjeta bancaria, cuando se va a pagar una entidad bancaria comprueba la validez de la tarjeta. De la tarjeta interesa conocer el número.
Las campañas son gestionadas por el administrativo de la tienda que se encargará de dar de baja la campaña anterior y dar de alta la nueva siempre que no haya ningún pedido pendiente de cumplimentar.
Existe un empleado de almacén que revisa los pedidos a diario y los cumplimenta. Esto consiste en recopilar los artículos que aparecen en el pedido y empaquetarlos. Cuando el paquete está listo se pasa al almacén a la espera de ser repartido. Del reparto se encarga una empresa de transportes que tiene varias rutas preestablecidas. Según el destino del paquete (la dirección del socio) se asigna a una u otra ruta. De la empresa de transportes se debe conocer su nombre, CIF y domicilio fiscal. Las rutas tienen un área de influencia que determina los destinos, y unos días de reparto asignados. Se debe conocer la fecha en la que se reparte el pedido. Si se produce alguna incidencia durante el reparto de algún pedido se almacena la fecha en la que se ha producido y una descripción.
Los socios pueden visualizar sus pedidos y cancelarlos siempre y cuando no hayan sido cumplimentados por el empleado de almacén. Así mismo puede modificar sus datos personales.

Documentación del caso de uso "Hacer pedido".

Como se indicaba en los contenidos del apartado 2.2, lo más importante en la elaboración de un diagrama de casos de uso, no es el diagrama en sí, sino la documentación de los casos de uso que es lo que permitirá desarrollar otros diagramas que ayuden en la codificación del sistema, y la elaboración de los casos de prueba de caja negra.
A modo de ejemplo vamos a desarrollar la documentación del caso de uso **Hacer Pedido**, ya que, por su complejidad abarca todos los apartados que hemos visto. El ejemplo se hará con la herramienta Visual Paradigm for UML, aunque tu puedes usar la herramienta que consideres más oportuna. Recordamos el aspecto del caso de uso:



Los datos que debemos incluir para elaborar la documentación del caso de uso, el contrato, eran:

- ✓ **Nombre:** nombre del caso de uso.
- ✓ **Actores**: aquellos que interactúan con el sistema a través del caso de uso.
- ✓ **Propósito**: breve descripción de lo que se espera que haga.
- ✓ **Precondiciones**: aquellas que deben cumplirse para que pueda llevarse a cabo el caso de uso.
- ✓ **Flujo normal**: flujo normal de eventos que deben cumplirse para ejecutar el caso de uso exitosamente.
- ✓ **Flujo alternativo**: flujo de eventos que se llevan a cabo cuando se producen casos inesperados o poco frecuentes. No se deben incluir aquí errores como escribir un tipo de dato incorrecto o la omisión de un parámetro necesario.
- ✓ **Postcondiciones**: las que se cumplen una vez que se ha realizado el caso de uso.

Para incluir el nombre, actores, propósito, precondiciones y postcondiciones abrimos la especificación del caso de uso. Esto da lugar a la aparición de una ventana con un conjunto de pestañas que podemos rellenar:

- ✓ En la pestaña "**General**" rellenamos el nombre "**Hacer pedido**", y tenemos un espacio para escribir una breve descripción del caso de uso, por ejemplo:

  "*El cliente visualiza los productos que están a la venta, que se pueden seleccionar para añadirlos al pedido. Puede añadir tantos artículos como desee, cada artículo añadido modifica el total a pagar según su precio y la cantidad seleccionada.*
  Cuando el cliente ha rellenado todos los productos que quiere comprar debe formalizar el pedido.
  En caso de que el cliente no sea socio de la empresa antes de formalizar la compra se le indica que puede hacerse socio, si el cliente acepta se abre el formulario de alta, en caso contrario se cancela el pedido.
  En caso de que se produzca algún problema con los datos bancarios se ofrecerá la posibilidad se volver a introducirlos.
  Al finalizar un pedido se añade al sistema con el estado pendiente."
- ✓ En la pestaña "**Valores etiquetados**" encontramos un conjunto de campos predefinidos, entre los que se encuentran el autor, precondiciones y postcondiciones, que podemos rellenar de la siguiente manera:

  Autor*: usuario.*
  Precondiciones*: Existe una campaña abierta con productos de la temporada actual a la venta.*
  Postcondiciones*: Se ha añadido un pedido con un conjunto de productos para servir con el estado "pendiente" que deberá ser revisado.*

También se pueden incluir otros datos como la complejidad, el estado de realización del caso de uso la complejidad que no hemos visto en esta unidad.

Para incluir el resto de los datos en el caso de uso hacemos clic en la opción "**Open Use Case Details...**" del menú contextual, lo que da lugar a la aparición de una ventana con una serie de pestañas. En ellas se recupera la información de la especificación que hemos rellenado antes, además, podemos rellenar el flujo de eventos del caso de uso, en condiciones normales usaríamos la pestaña "Flow of events", sin embargo como esta opción solo está disponible en la versión profesional, utilizaremos la pestaña "**Descripción**", que está disponible en la versión community. Para activarla pulsamos el botón "**Create/Open Description**".

Podemos añadir varias descripciones de diferentes tipos, pulsando el botón "**Nuevo**". Para añadir filas al flujo de eventos pinchamos en el botón. En principio añadimos la descripción principal, luego añadiremos otras alternativas.

Esta es la descripción principal, en ella se describe el flujo normal de eventos que se producen cuando se ejecuta el caso de uso sin ningún problema.

Flujo de eventos normal para el caso de uso Hacer Pedido.

Use Case	Hacer pedido		
Author	usuario		
Date	26-ago-2011 13:56:56		
Brief Description	EL usuario selecciona un conjunto de artículos, junto con la cantidad de los mismos, para crear el pedido. Cuando se formaliza se comprueba que el usuario sea socio. A continuación se comprueban los datos bancarios, se realiza el cobro y se crea el pedido.		
Preconditions	Existe un catálogo de productos disponibles para pedir. El usuario está registrado. Los datos bancarios son correctos.		
Post-conditions	Se crea un pedido con los datos del usuario que lo realiza y los artículos solicitados.		
Flow of Events		**Actor Input**	**System Response**
	1	Inicia el pedido.	
	2		Se crea un pedido en estado "en construcción".
	3	Selecciona un artículo.	
	4	Selecciona una cantidad.	
	5		Recupera la información del artículo para obtener el precio y modifica el precio total del pedido.
	6	El proceso se repite hasta completar la lista de	

		artículos.	
	7	Se acepta el pedido.	
	8		Se comprueba si el usuario es socio, si no lo es se le muestra un aviso para que se registre en el sitio.
	9		Se comprueban los datos bancarios con una entidad externa.
	10		Se genera: calcula el total, sumando los gastos de envío.
	11		Se realiza el pago a través de la entidad externa.
	12		Se almacena la información del pedido con el estado "pendiente".

Añadimos un par de descripciones alternativas para indicar que hacer cuando el usuario no es socio y cuando los datos bancarios no son correctos:

Flujo alternativo para el caso de uso Hacer Pedido cuando el usuario no está registrado.			
Author	usuario		
Date	26-ago-2011 17:56:35		
Brief Description	Cuando el usuario hace un pedido si no está registrado se abre la opción de registro para que se dé de alta.		
Preconditions	El usuario no está registrado. Existe un catálogo de artículos para hacer pedido. Los datos bancarios son correctos.		
Post-conditions	El usuario queda registrado. Se crea un pedido con los datos del usuario que lo realiza y los artículos solicitados.		
Flow of Events		**Actor Input**	**System Response**
	1	Inicia el pedido.	
	2		Se crea un pedido en estado "en construcción".
	3	Selecciona un artículo.	
	4	Selecciona la cantidad.	
	5		Recupera la información del artículo para obtener su precio y modifica el precio total a

			pagar por el pedido.
	6		Añade la información al pedido en creación.
	7	EL proceso se repite hasta completar la lista de artículos del pedido.	
	8	Acepta el pedido.	
	9		Se comprueba si el usuario es socio.
	10		Se invoca el registro de usuario.
	11		Se comprueban los datos bancarios con una entidad externa.
	12		Se genera: calcula el total, sumando los gastos de envío.
	13		Se realiza el pago a través de la entidad externa.
	14		El pedido queda almacenado con el estado "pendiente".

Flujo alternativo para hacer pedido cuando los datos bancarios no son correctos.			
Author	usuario		
Date	26-ago-2011 18:14:35		
Brief Description	Una vez que se han seleccionado los artículos y se han introducido los datos del socio, al hacer la comprobación de los datos bancarios con la entidad externa se produce algún error, se da la posibilidad al usuario de modificar los datos o de cancelar el pedido.		
Preconditions	Existe un catálogo de productos disponibles para pedir. El usuario está registrado. Los datos bancarios no son correctos.		
Post-conditions	Se crea un pedido con los datos del usuario que lo realiza y los artículos solicitados.		
Flow of Events		**Actor Input**	**System Response**
	1	Inicia el pedido.	
	2		Se crea un pedido en estado "en

			construcción".
	3	Selecciona un artículo.	
	4	Selecciona una cantidad.	
	5		Recupera la información del artículo para obtener el precio y modifica el precio total del pedido.
	6	El proceso se repite hasta completar la lista de artículos.	
	7	Se acepta el pedido.	
	8	Acepta el pedido.	
	9		Se comprueban los datos bancarios con una entidad externa, fallando la comprobación.
	10		Se solicitan los datos de nuevo.
	11	Introduce los datos de nuevo.	
	12		Se repite el proceso hasta que se acepten los datos bancarios o se cancele la operación.
	13		Se genera: calcula el total, sumando los gastos de envío.
	14		Se realiza el pago a través de la entidad externa.
	15		Se almacena la información del pedido con el estado "pendiente".

El resto de casos se uso se documentan de forma similar hasta completar la descripción formal de la funcionalidad del sistema.

2.5 Escenarios

Caso práctico

Ada continua la investigación, junto con el equipo de BK programación, que una vez ha creado su primer diagrama de casos de uso, se da cuenta de que realmente es una herramienta muy útil a la hora de definir la funcionalidad de un sistema. Continuando con la investigación descubren una ventaja adicional, utilizando los flujos de eventos, pueden describir interacciones concretas de los actores con el sistema, estas interacciones son los escenarios.

Un caso de uso debe especificar un comportamiento deseado, pero no imponer cómo se llevará a cabo ese comportamiento, es decir, debe decir QUÉ pero no CÓMO. Esto se realiza utilizando escenarios que son casos particulares de un caso de uso.

Un **escenario** es una ejecución particular de un caso de uso que se describe como una secuencia de eventos. Un caso de uso es una generalización de un escenario.

Por ejemplo, para el caso de uso hacer pedido podemos establecer diferentes escenarios:

Un posible escenario podría ser:

Realizar pedido de unos zapatos y unas botas.

1. El usuario inicia el pedido.
2. Se crea el pedido en estado “en construcción”.
3. Se selecciona un par de zapatos “Lucía” de piel negros, del número 39.
4. Se selecciona la cantidad 1.
5. Se recupera la información de los zapatos y se modifica la cantidad a pagar sumándole 45 €.
6. Se selecciona un par de botas “Aymara” de ante marrón del número 40.
7. Se selecciona la cantidad 1.
8. Se recupera la información de las botas y se modifica la cantidad a pagar sumándole 135€.
9. El usuario acepta el pedido.
10. Se comprueba que el usuario es, efectivamente socio.
11. Se comprueban los datos bancarios, que son correctos.
12. Se calcula el total a pagar añadiendo los gastos de envío.
13. Se realiza el pago a través de una entidad externa.
14. Se genera un pedido para el usuario con los dos pares de zapatos que ha comprado, con el estado “pendiente”.

Los escenarios pueden y deben posteriormente documentarse mediante diagramas de secuencia.

3. Diagramas de secuencia

Caso práctico
María se ha dado cuenta de que los casos de uso permiten, de una manera sencilla, añadir información sobre qué hace el sistema, sin embargo por completa que sea la descripción de la secuencia de eventos no permite incluir información útil, como los objetos que intervienen en las tareas, y como se comunican.
—Tendríamos que buscar la forma de representar como circula la información, que objetos participan en los casos de uso, qué mensajes envían, y en que momento, esto nos ayudaría mucho a completar después el diagrama de clases.
Como siempre, Ada tiene una solución.
—Tendremos que investigar los diagramas de secuencia.

Los diagramas de secuencia se utilizan para formalizar la descripción de un escenario o conjunto de ellos representando que mensajes fluyen en el sistema así como quien los envía y quien los recibe.

Los objetos y actores que forman parte del escenario se representan mediante rectángulos distribuidos horizontalmente en la zona superior del diagrama, a los que se asocia una linea temporal vertical (una por cada actor) de las que salen, en orden, los diferentes mensajes que se pasan entre ellos. Con esto el equipo de desarrollo puede hacerse una idea de las diferentes operaciones que deben ocurrir al ejecutarse una determinada tarea y el orden en que deben realizarse.

Los diagramas de secuencia completan a los diagramas de casos de uso, ya que permiten al equipo de desarrollo hacerse una idea de qué objetos participan en el caso de uso y como interaccionan a lo largo del tiempo.

3.1 Representación de objetos, línea de vida y paso de mensajes.

CASO PRÁCTICO
Ada, orienta a su equipo:
—Bien, ¿qué nos haría falta para poder representar la interacción de los objetos que participan en el caso de uso a lo largo del tiempo?
—Alguna manera de representar los objetos, el paso del tiempo y el paso de mensajes ¿no?

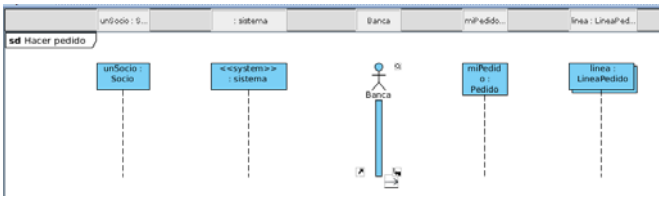
Representación de objetos y linea de vida.

En un diagrama de secuencia se dibujan las entidades que participan dentro de rectángulos que se distribuyen horizontalmente. De cada rectángulo o entidad sale una línea de puntos que representa el paso del tiempo, se les denomina línea de vida y representan que el objeto existe.

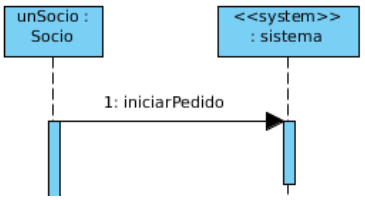
Para formar el nombre de una linea de vida de un objeto se usa el nombre del objeto, que es opcional, seguido del símbolo dos puntos y a continuación el nombre de la clase a la que pertenece.

Una linea de vida puede estar encabezada por otro tipo de instancias como el sistema o un actor que aparecerán con su propio nombre. Usaremos el sistema para representar solicitudes al mismo, como por ejemplo pulsar un botón para abrir una ventana o una llamada a una subrutina.

Cuando tenemos un objeto que puede tener varias instancias, aparece como un rectángulo sobre otro, como en es el caso de las lineas del pedido que pueden ser varias.



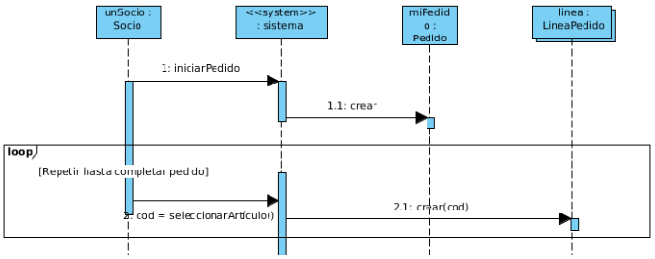
Invocación de métodos.



Los mensajes, que significan la invocación de métodos, se representan como flechas horizontales que van de una linea de vida a otra, indicando con la flecha la dirección del mensaje, los extremos de cada mensaje se conectan con una línea de vida que conecta a las entidades al principio del diagrama. Los mensajes se dibujan desde el objeto que envía el mensaje al que lo recibe, pudiendo ser ambos el mismo objeto y su orden viene determinado por su posición vertical, un mensaje que se dibuja debajo de otro indica que se envía después, por lo que no se hace necesario un número de secuencia.

Iteraciones y condicionales.

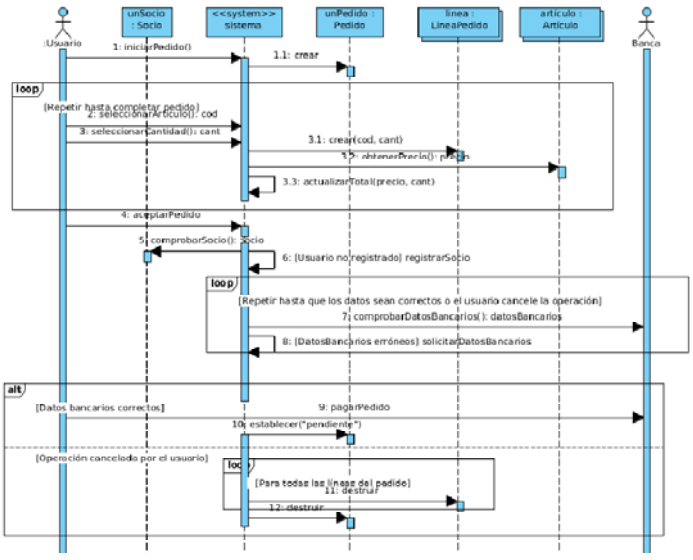
Además de presentar acciones sencillas que se ejecutan de manera secuencial también se pueden representar algunas situaciones más complejas como bucles usando marcos, normalmente se nombra el marco con el tipo de bucle a ejecutar y la condición de parada. También se pueden representar flujos de mensajes condicionales en función de un valor determinado.



Por último destacar que se puede completar el diagrama añadiendo etiquetas y notas en el margen izquierdo que aclare la operación que se está realizando.

3.2 Elaboración de un diagrama de secuencia.

Vamos a generar el diagrama de secuencia para el caso de uso **Hacer pedido**. En el se establece la secuencia de operaciones que se llevarán a cabo entre los diferentes objetos que intervienen en el caso de uso.



Este es el diagrama ya terminado, en el se han incluido todas las entidades (actores, objetos y sistema) que participan en el diagrama, y se han descrito todas las operaciones, incluidos los casos especiales, como es el registro de usuarios o la gestión de los datos bancarios. También incluye el modelado de acciones en bucle, como es la selección de artículos y de acciones regidas por condición, como es la posibilidad de cancelar el pedido si hay problemas con la tarjeta de crédito.

Creación del diagrama de colaboración

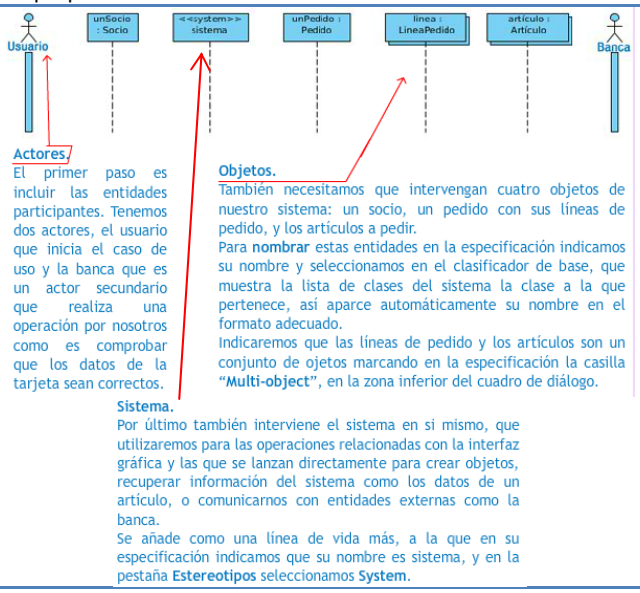
Crear el diagrama de secuencia

- ✓ El diagrama de secuencia que vamos a tratar es el del caso de uso Hacer pedido, que hemos descrito en el apartado anterior. Si no recuerdas bien cual era la descripción del caso te invito a que la repases en el punto 2.4 de los

contenidos de la unidad.

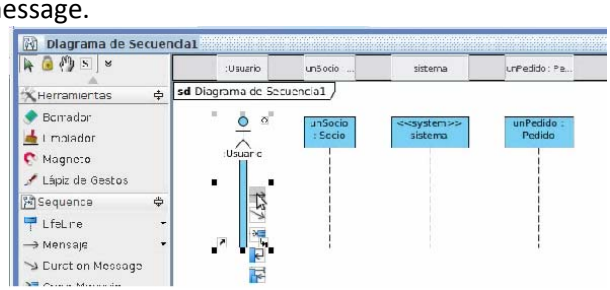
✓ Este es un diagrama muy completo que incluye bastantes elementos de este diagrama, como bucles o condicionantes, veamos como incluirlos con la herramienta Visual Paradigm, como siempre, debes saber que puedes investigar y utilizar otras herramientas que sirvan para este mismo propósito.

Incluir entidades

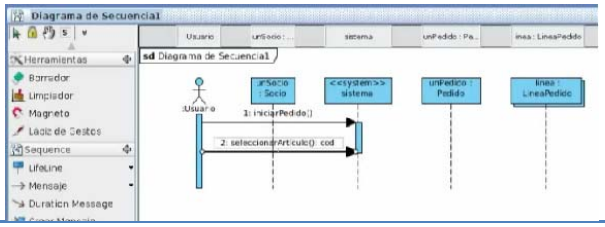


Mensajes

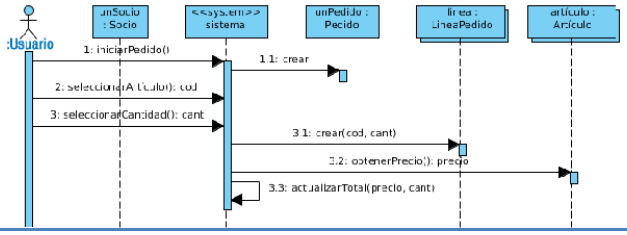
Los mensajes se añaden pinchando sobre la línea de vida que envía el mensaje, y seleccionando la flecha, entonces se abre una ventana en la que elegir la línea destino, ten en cuenta que un objeto puede enviarse un mensaje a si mismo seleccionando la oprición Self-message.



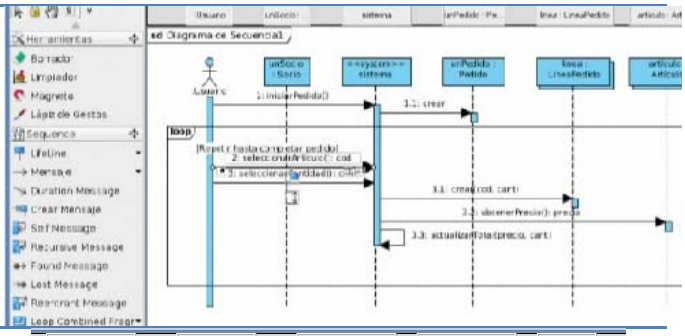
Los mensajes pueden devolver un valor, que podemos escribir en la especificación del mismo en la opción return value.



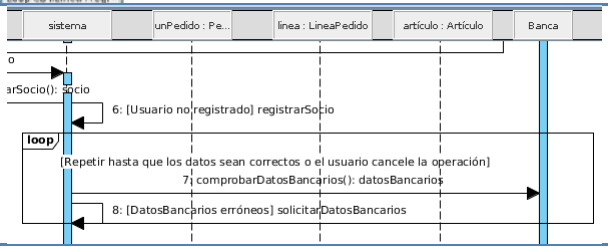
En esta imagen vemos los mensajes correspondientes al apartado de añadir los datos del pedido. Se selecciona el artículo y la cantidad, se crea una nueva línea de pedido y se recuperan los datos del artículo para obtener su precio y actualizar el total. Puesto que este proceso lo podemos repetir en más de una ocasión lo meteremos en un bucle.



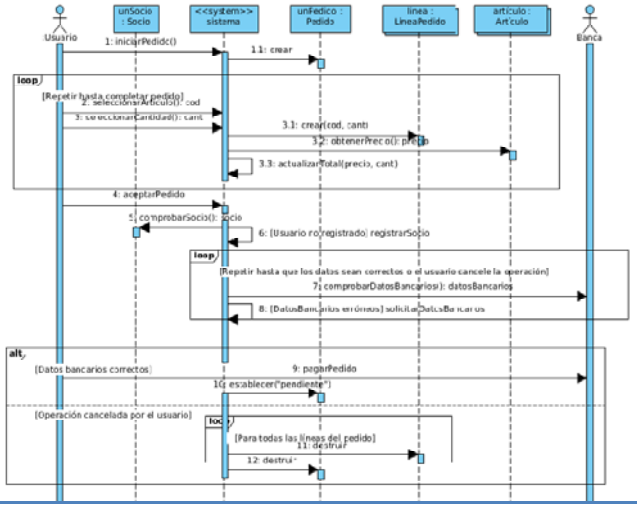
Para añadir un bucle seleccionamos la opción Loop Combined Fragment.



Añadiremos condiciones de guarda cuando queramos indicar que un mensaje se enviará sólo si se cumple cierta condición, por ejemplo, sólo registraremos a un usuario si no es socio ya.



También se pueden incluir condiciones más elaboradas que implique una bifurcación en el flujo de eventos, como es el caso de la comprobación de la tarjeta, si todo marcha bien se finalizará la creación del pedido, si no, el usuario puede cancelar la operación sin guardar nada.



¿Cual de estos elementos no forma parte de un diagrama de secuencia?

- ☐ Actor
- ☐ Objeto
- ☐ Bucle
- ☉ **Evento**

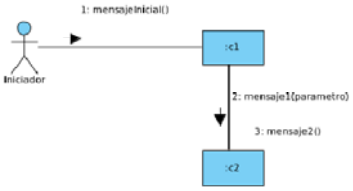
4. Diagramas de colaboración.

CASO PRÁCTICO.
—El diagrama de secuencia ha aportado información muy valiosa sobre la circulación de mensajes en los casos de uso, sin embargo estaría bien poder mostrar esta información de otra forma en la que se apreciase mejor el anidamiento de los mensajes, y el flujo de control entre objetos, ¿no creéis?

Los diagramas de colaboración son un complemento para los de secuencia cuyo objetivo es mostrar las interacciones entre los objetos del diagrama mediante el paso de mensajes entre ellos. Las interacciones entre los objetos se describen en forma de grafo en el que los nodos son objetos y las aristas son enlaces entre objetos a través de los cuales se pueden enviar mensajes entre ellos. Los objetos se conectan mediante enlaces y se comunican a través de los mensajes.

Como, a diferencia de los diagramas de secuencia, no se incluye una línea temporal los mensajes son numerados para determinar su orden el el tiempo.

En este diagrama de colaboración el actor Iniciador manda un mensaje al objeto c1 que inicia el escenario, a continuación el objeto c1 envía el mensaje mensaje1 que lleva un parámetro al objeto c2 y después el mensaje mensaje2, que no tiene parámetros de nuevo al objeto c2.



Los diagramas de colaboración y secuencia utilizan los mismos elementos pero distribuyéndolos de forma diferente, ¿crees que son semejantes?
Es cierto, ambos diagramas representan la misma información, representan entidades del sistema y los mensajes que circulan entre ellas, además en ambos casos es fácil detectar la secuencialidad bien por el uso de lineas de vida, bien por los números de secuencia. De hecho en algunas aplicaciones para el desarrollo de estos diagramas es posible crear un diagrama a partir del otro.

4.1 Representación de objetos.

CASO PRÁCTICO.
—De acuerdo, mientras investigamos los diagramas de colaboración vamos a ver con un poco más de detalle qué significa la notación que se asigna a los objetos, ¿que diferencia hay entre usar los dos puntos o no hacerlo? ¿Podemos usar el nombre de una clase, solamente, o es obligatorio indicar el nombre del objeto?

Un objeto puede ser cualquier instancia de las clases que hay definidas en el sistema, aunque también pueden incluirse objetos como la interfaz del sistema, o el propio sistema, si esto nos ayuda a modelar las operaciones que se van a llevar a cabo.

Los objetos se representan mediante rectángulos en los que aparece uno de estos nombres.

- ✓ **NombreClase:** directamente se puede utilizar el nombre de la clase a la que pertenece el objeto que participa en la interacción. Pero esta representación hace referencia a la clase, el resto son objetos.

- ✓ **NombreObjeto** : se puede usar el nombre concreto del objeto que participa en la interacción, normalmente aparece subrayado.

- ✓ **:nombreClase** : cuando se coloca el símbolo : delante del nombre de la clase quiere decir que hace referencia a un objeto genérico de esa clase.

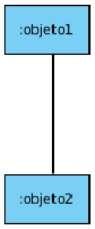
- ✓ **NombreObjeto:nombreClase** : hace referencia al objeto concreto que se nombre añadiendo la clase a la que pertenece.

4.2 Paso de mensajes.

CASO PRÁCTICO.

—Y cuando enviamos un mensaje ¿cómo se representa exactamente?, ¿podemos incluir de alguna forma parámetros en los mensajes o valores devueltos? ¿Y si necesitamos indicar que el mensaje se enviará sólo si se cumple una determinada condición? ¿o que se envía dentro de un bucle?

Para que sea posible el paso de mensajes es necesario que exista una asociación entre los objetos. En la imagen es posible el paso de mensajes entre el objeto objeto1 y objeto2, además de quedar garantizada la navegación y visibilidad entre ambos.



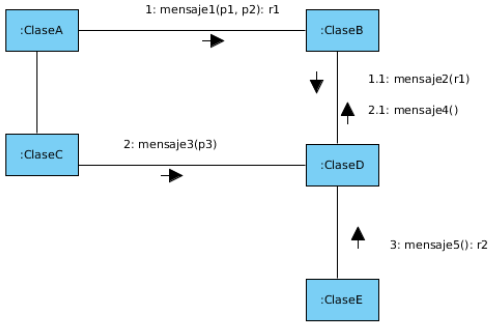
Un mensaje es la especificación de una comunicación entre objetos que transmite información y desencadena una acción en el objeto destinatario. La sintaxis de un mensaje es la siguiente:

```
[secuencia][*] [Condición de guarda]{valorDevuelto} : mensaje (argumentos)
```

donde:

- ✓ **Secuencia**: representa el nivel de anidamiento del envío del mensaje dentro de la interacción. Los mensajes se numeran para indicar el orden en el que se envían, y si es necesario se puede indicar anidamiento incluyendo subrangos.
- ✓ *: indica que el mensaje es iterativo.
- ✓ **Condición de guarda**: debe cumplirse para que el mensaje pueda ser enviado.
- ✓ **ValorDevuelto**: lista de valores devueltos por el mensaje. Estos valores se pueden utilizar como parámetros de otros mensajes. Los corchetes indican que es opcional.
- ✓ **Mensaje**: nombre del mensaje.
- ✓ **Argumentos**: parámetros que se pasan al mensaje.

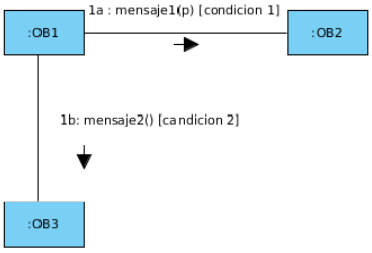
Como se ve en el ejemplo se puede usar la misma asociación para enviar varios mensajes. Vemos que hay dos mensajes anidados, el 1.1 y el 2.1, se ha usado el nombre de los mensajes para indicar el orden real en el que se envían.



Los mensajes 1, 2 y 3 tiene parámetros y los mensajes 1 y 5 devuelve un resultado.

Se contempla la bifurcación en la secuencia añadiendo una condición en la sintaxis del mensaje:

```
[Secuencia][*][CondiciónGuarda]{valorDevuelto} : mensaje (argumentos)
```



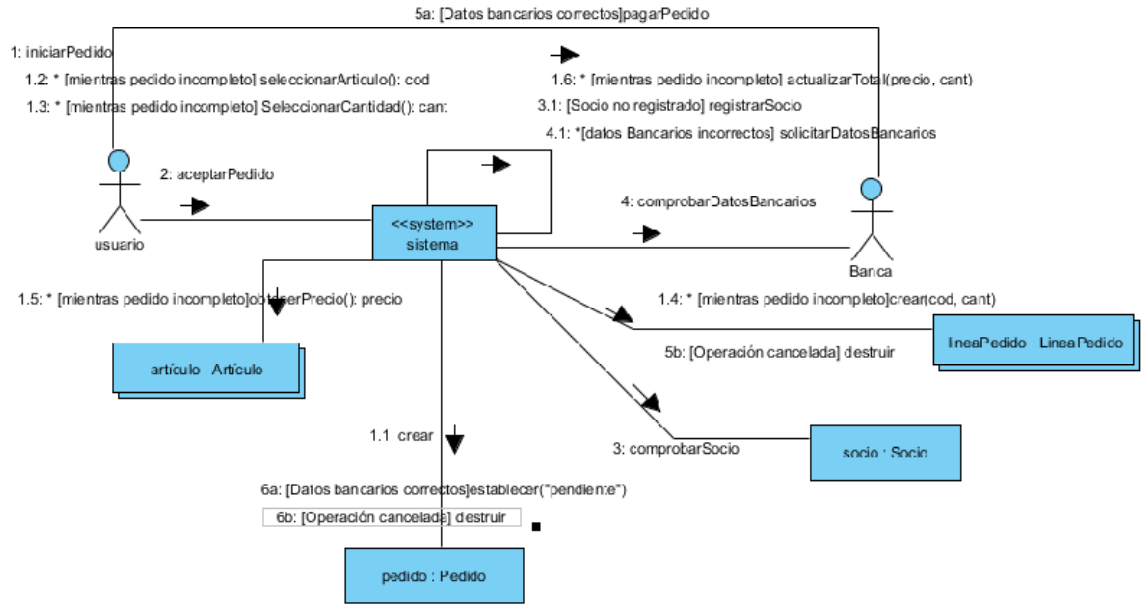
Cuando tenemos una condición se repite el número de secuencia y se añaden las condiciones necesarias, como vemos en la imagen según la condición se enviará el mensaje 1 o el 2, pero no ambos, por lo que coinciden en número de secuencia.

La iteración se representa mediante un * al lado del número de secuencia, pudiendo indicarse ente corchetes la condición de parada del bucle.

Nota: VP-UML modifica el orden en el que aparecen los datos pero no su notación.

4.3 Elaboración de un diagrama de colaboración.

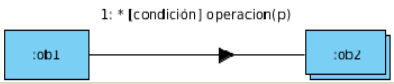
Este es el diagrama de colaboración del caso de uso **Hacer pedido**, se ha creado siguiendo el diagrama de secuencia, por lo que no te debe ser muy difícil seguirlo, de hecho algunas aplicaciones para la creación de estos diagramas permiten la obtención de uno a partir de otro. Debes tener en cuenta que la aplicación modifica un poco la signatura de los mensajes, el valor devuelto se representa al final precedido de dos puntos.



Los aspectos más destacados son los siguientes:

- ✓ Las actividades que se repiten o pueden repetirse se marcan con un asterisco y su condición.
- ✓ Las condiciones de guarda se escriben en el mismo nombre del mensaje.
- ✓ El flujo alternativo de eventos según si el usuario cancela el pedido o no, obliga a modificar los números de secuencia de los mensajes 5 y 6, pasando a tener los mensajes 5a y 6a y 5b y 6b, según la condición. Puedes modificar el número se secuencia de los mensajes abriendo la especificación del diagrama, y seleccionando la pestaña Mensajes, donde puedes editar los números de secuencia haciendo doble clic sobre ellos.
- ✓ Al objeto "sistema" se le ha asignado el estereotipo system.

Indica qué afirmación no es correcta para el siguiente diagrama:



- ☐ El objeto ob2 es multiobjeto
- ☐ Se envía un mensaje del objeto 1 al objeto 2
- ◉ **El mensaje operación(pp) se ejecutará siempre**
- ☐ La operación se puede ejecutar varias veces

5. Diagramas de actividad.

CASO PRÁCTICO.
Por el momento el equipo de BK no ha tenido problema en seguir lo que Ada les cuenta sobre los diagramas UML. Antonio, que está verdaderamente interesado en el tema hace a Ada la siguiente pregunta:
—¿Que pasaría si quisiera representar sólo las acciones que tienen lugar, prescindiendo de quien las genera, solo el flujo de la actividad del sistema, qué pasa primero, qué ocurre después y qué cosas pueden hacerse al mismo tiempo?
—Pasaría que tendrías que hacer un diagrama de actividad.

El Diagrama de Actividad es una especialización del Diagrama de Estado, organizado respecto de las acciones, que se compone de una serie de actividades y representa cómo se pasa de unas a otras. Las actividades se enlazan por transiciones automáticas, es decir, cuando una actividad termina se desencadena el paso a la siguiente actividad.

Se utilizan fundamentalmente para modelar el flujo de control entre actividades en el que se puede distinguir cuales ocurren secuencialmente a lo largo del tiempo y cuales se pueden llevar a cabo concurrentemente.

Permite visualizar la dinámica del sistema desde otro punto de vista que complementa al resto de diagramas. En concreto define la lógica de control:

- ✓ **En el modelado de los procesos del negocio**: Permiten especificar y evaluar el flujo de trabajo de los procesos de negocio.
- ✓ **En el análisis de un caso de uso**: Permiten comprender qué acciones deben ocurrir y cuáles son las dependencias de comportamiento.
- ✓ **En la comprensión del flujo de trabajo, a través de varios casos de uso**: Permiten representar con claridad las relaciones de flujo de trabajo (workflow) entre varios casos de uso.
- ✓ Aclaran cuando se trata de expresar **aplicaciones multihilos**.

Un diagrama de actividades es un grafo conexo en el que los nodos son estados, que pueden ser de actividad o de acción y los arcos son transiciones entre estados. El grafo parte de un nodo inicial que representa el estado inicial y termina en un nodo final.

¿Por qué decimos que el diagrama de actividades visualiza el comportamiento desde otro punto de vista del resto de diagramas?
En los anteriores diagramas, tratábamos la interacción entre objetos y entidades, cual es el flujo de mensajes, en qué orden y bajo qué condiciones se envían, en los diagramas de actividades se incluye el factor de la concurrencia, y trata sobre todo de expresar el flujo de las actividades, su elemento fundamental, y como se pasa de unas a otras. Además también representa como se influye en los objetos.

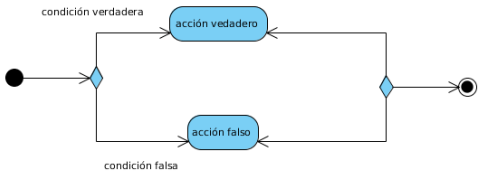
5.1 Elementos del diagrama de actividad.

CASO PRÁCTICO.
—Vale estoy preparado, ¿qué necesito para tener un diagrama de actividad?
Normalmente los diagramas de actividades contienen:

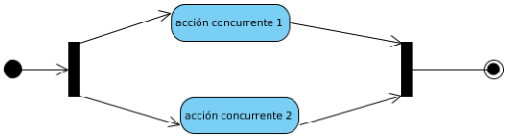
- ✓ **Estados** de actividad y estados de acción.
  - ➔ **Estado de actividad**: Elemento compuesto cuyo flujo de control se compone de otros estados de actividad y de acción.

- **Estado de acción**: Estado que representa la ejecución de una acción atómica, que no se puede descomponer ni interrumpir, normalmente la invocación de una operación. Generalmente se considera que su ejecución conlleva un tiempo insignificante.
- Pueden definirse también otro tipo de estados:
  - **Inicial**.
  - **Final**.

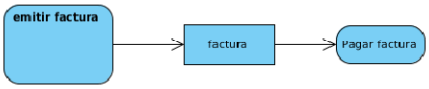
- **Transiciones**: Relación entre dos estados que indica que un objeto en el primer estado realizará ciertas acciones y pasará al segundo estado cuando ocurra un evento específico y satisfaga ciertas condiciones. Se representa mediante una línea dirigida del estado inicial al siguiente. Podemos encontrar diferentes tipos de transacciones:
  - **Secuencial o sin disparadores**: Al completar la acción del estado origen se ejecuta la acción de salida y, sin ningún retraso, el control sigue por la transición y pasa al siguiente estado.
  - **Bifurcación(Descision node)**: Especifica caminos alternativos, elegidos según el valor de alguna expresión booleana. Las condiciones de salida no deben solaparse y deben cubrir todas las posibilidades (puede utilizarse la palabra clave else). Pueden utilizarse para lograr el efecto de las iteraciones.



  - **Fusión (Merge node)**: Redirigen varios flujos de entrada en un único flujo de salida. No tiene tiempo de espera ni sincronización.
  - **División (Fork node)**: Permiten expresar la sincronización o ejecución paralela de actividades. Las actividades invocadas después de una división son concurrentes.
  - **Unión (Join node):** Por definición, en la unión los flujos entrantes se sincronizan, es decir, cada uno espera hasta que todos los flujos de entrada han alcanzado la unión.



- **Objetos**: Manifestación concreta de una abstracción o instancia de una clase. Cuando interviene un objeto no se utilizan los flujos de eventos habituales sino flujos de objetos (se representan con una flecha de igual manera) que permiten mostrar los objetos que participan dentro del flujo de control asociado a un diagrama de actividades. Junto a ello se puede indicar cómo cambian los valores de sus atributos, su estado o sus roles.



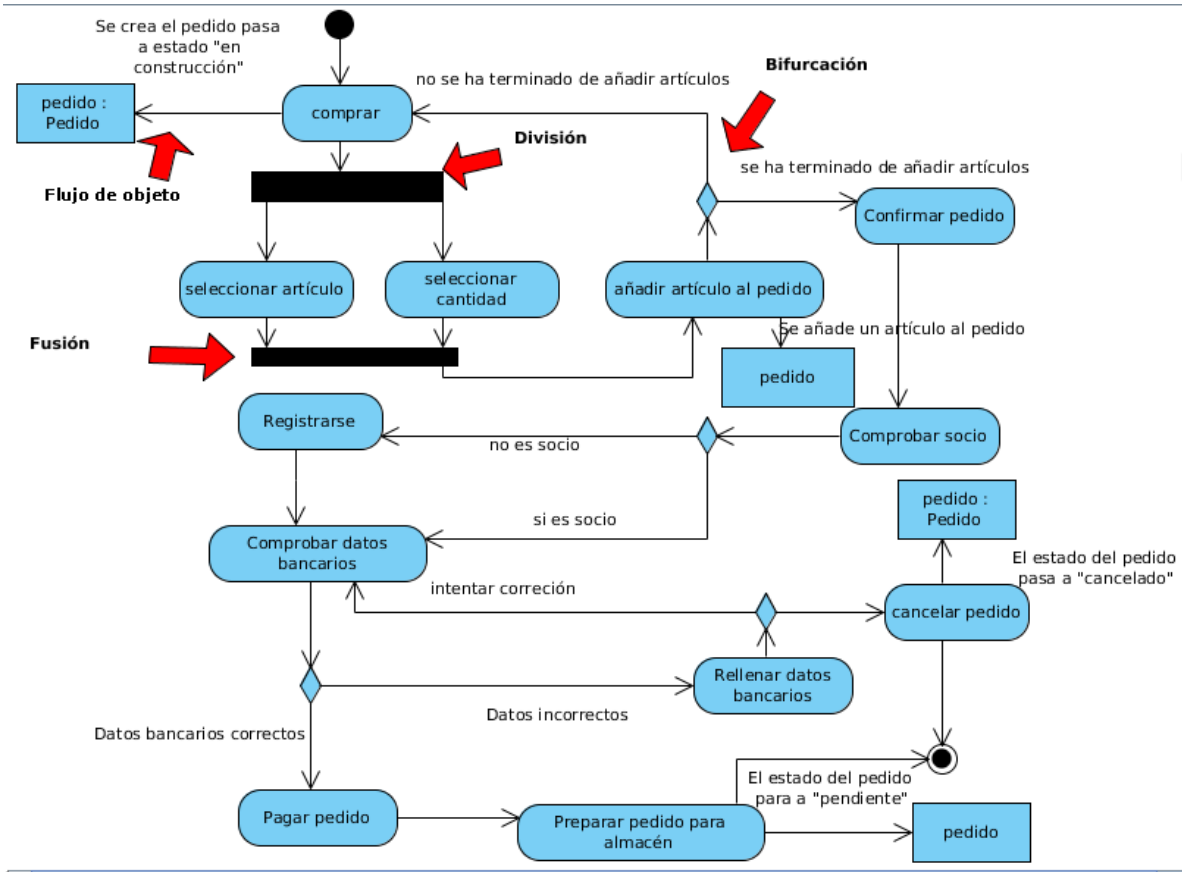
Se utilizan carriles o calles para ver **QUIENES** son los responsables de realizar las distintas actividades, es decir, especifican qué parte de la organización es responsable de una actividad.

- Cada calle tiene un nombre único dentro del diagrama.
- Puede ser implementada por una o varias clases.
- Las actividades de cada calle se consideran independientes y se ejecutan concurrentemente a las de otras calles.

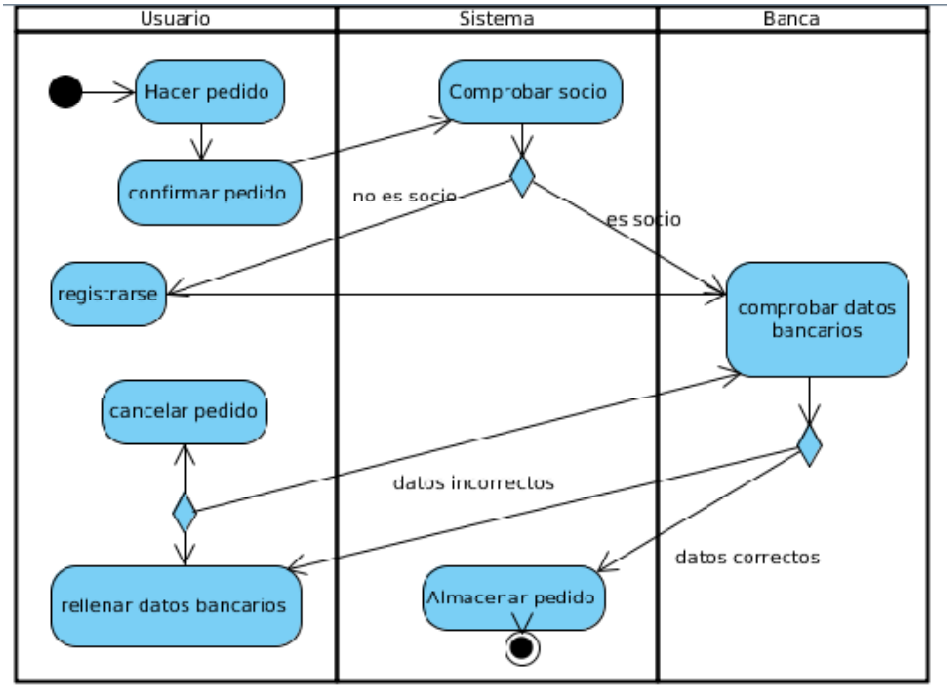
5.2 Elaboración de un diagrama de actividad.

El siguiente diagrama de actividad representa el caso de uso hacer pedido, en el aparecen los elementos que hemos visto en las secciones anteriores.

- En las bifurcaciones se ha añadido la condición que indica si se pasa a una acción o a otra.
- Las acciones Seleccionar artículo y Seleccionar cantidad se han considerado concurrentes.



En este otro diagrama se simplifican las acciones a realizar y se eliminan los objetos para facilitar la inclusión de calles que indican quien realiza cada acción:



PARA AÑADIR LAS CALLES EN VISUAL PARADIGM SE UTILIZA LA HERRAMIENTA DEL PANEL "VERTICAL SWIMLANE"

Los diagramas de actividades, a diferencia del resto, permiten incluir la concurrencia en la representación del diagrama.

- [x] Verdadero
- [ ] Falso

6. Diagramas de estados.

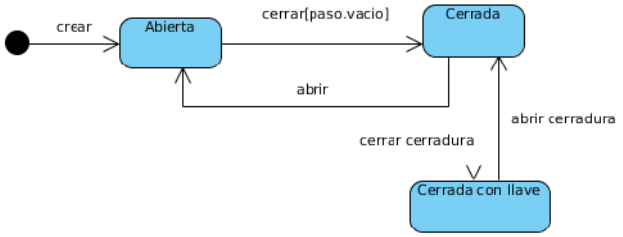
CASO PRÁCTICO.
Ada espera que su equipo continúe con tan buen ánimo para estudiar un tipo de diagrama más, que completará las diferentes visiones de la dinámica de un sistema que proporciona UML. Son los diagramas de estados, que les permitirán analizar cómo va cambiando el estado de los objetos que tienen una situación variable a lo largo del tiempo.

Basado en los statecharts de David Harel. Representan máquinas de estados (autómatas de estados finitos *(Modelo matemático que realiza cómputos sobre una entrada para producir una salida. Se representa mediante un grafo cerrado y dirigido cuyos vértices son estados y sus aristas son transiciones, que van etiquetadas con un símbolo que representa el evento que desencadena la transición. Parte de un estado inicial y termina en uno o varios estados finales.)*) para modelar el comportamiento dinámico basado en la respuesta a determinados eventos de aquellos objetos que requieran su especificación, normalmente por su comportamiento significativo en tiempo real y su participación en varios casos de uso. El resto de objetos se dice que tienen un único estado.

En relación con el diagrama de estados se cumple que:

- ✓ Un objeto está en un estado concreto en un cierto momento, que viene determinado, parcialmente, por los valores de sus atributos.
- ✓ La transición de un estado a otro es momentánea y se produce cuando ocurre un determinado evento.
- ✓ Una máquina de estados procesa un evento cada vez y termina con todas las consecuencias del evento antes de procesar otro. Si ocurren dos eventos simultáneamente se procesan como si se hubieran producido en cualquier orden, sin pérdida de generalidad.

Un diagrama de máquina de estados expresa el comportamiento de un objeto como una progresión a través de una serie de estados, provocada por eventos y las acciones relacionadas que pueden ocurrir. Por ejemplo, aquí tenemos el diagrama de estados de una puerta.



Analiza el diagrama de estados de la puerta, según está dibujado, ¿Se puede abrir una puerta que está cerrada con llave directamente?

○ Verdadero ◉ **Falso**

6.1 Estados y eventos.

CASO PRÁCTICO.
Ada indica a su equipo que para entender bien la dinámica de un diagrama de estados deben comenzar por analizar sus componentes fundamentales: estados y eventos.

Un estado es una situación en la vida de un objeto en la que satisface cierta condición, realiza alguna actividad o espera algún evento.

Elementos de un estado

- ✓ Nombre
- ✓ Acciones entrada/salida.
- ✓ Actividad a realizar.
- ✓ Subestados, cuando el estado es complejo y necesita de un diagrama que lo defina.
- ✓ Eventos diferidos.

Existen dos tipos de estado especiales, estado inicial y estado final.

- ✓ **Estado inicial**: es un pseudoestado que indica el punto de partida por defecto para una transición cuyo destino es el límite de un estado compuesto. El estado inicial del estado de nivel más alto representa la creación de una nueva instancia de la clase.
- ✓ **Estado final**: Estado especial dentro de un estado compuesto que, cuando está activo, indica que la ejecución del estado compuesto ha terminado y que una transición de finalización que sale del estado compuesto está activada.

Un evento es un acontecimiento que ocupa un lugar en el tiempo y espacio que funciona como un estímulo que dispara una transición en una máquina de estados. Existen eventos externos y eventos internos según el agente que los produzca.

Tipos de eventos:

- ✓ **Señales (excepciones)**: la recepción de una señal, que es una entidad a la que se ha dado nombre explícitamente (clase estereotipada), prevista para la comunicación explícita - y asíncrona- entre objetos. Es enviada por un objeto a otro objeto o conjunto de objetos. Las señales con nombre que puede recibir un objeto se modelan designándolas en un compartimento extra de la clase de ese objeto. Normalmente una señal es manejada por la máquina de estados del objeto receptor y puede disparar una transición en la máquina de estados.
- ✓ **Llamadas**: la recepción de una petición para invocar una operación. Normalmente un evento de llamada es modelado como una operación del objeto receptor, manejado por un método del receptor y se implementa como una acción o transición de la máquina de estados.
- ✓ **Paso de tiempo**: representa el paso del tiempo (ocurrencia de un tiempo absoluto respecto de un reloj real o virtual o el paso de una cantidad de tiempo dada desde que un objeto entra en un estado). Palabra clave `after`: `after (2 segundos); after 1 ms desde la salida de devInactivo`.
- ✓ **Cambio de estado**: evento que representa un cambio en el estado o el cumplimiento de una condición. Palabra clave `when`, seguida de una expresión booleana, que puede ser de tiempo o de otra clase: `when (hora = 11:30); when (altitud < 1000)`.

6.2 Transiciones.

CASO PRÁCTICO.

—De acuerdo, los estados son situaciones específicas en las que se puede encontrar un objeto, y los eventos pueden hacer que un objeto cambie de estado, y, ¿cómo representamos eso?

Una transición de un estado A a un estado B, se produce cuando se origina el evento asociado y se satisface cierta condición especificada, en cuyo caso se ejecuta la acción de salida de A, la acción de entrada a B y la acción asociada a la transición.

La signatura de una transición es como sigue:

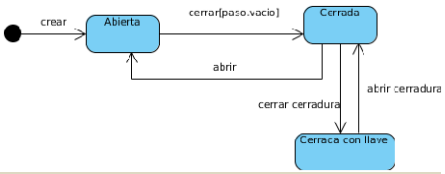
```
Evento(argumentos) [condición] / accion
```

donde todos los elementos son opcionales.

Elementos de una transición:

- ✓ **Estados origen y destino**: la transición se disparará si, estando en el estado origen se produce el evento de disparo y se cumple la condición de guarda (si la hay), pasando a ser activo el estado final.
- ✓ **Evento de disparo**: cuando se produce un evento, afecta a todas las transiciones que lo contienen en su etiqueta. Todas las apariciones de un evento en la misma máquina de estados deben tener la misma signatura. Los tipos de evento los hemos visto en el punto anterior.
- ✓ **Condición de guarda**: expresión booleana. Si es falsa, la transición no se dispara, y si no hay otra transición etiquetada con el mismo evento que pueda dispararse, éste se pierde.
- ✓ **Acción**: computación atómica ejecutable. Puede incluir llamadas a operaciones del objeto que incluye la máquina de estados (o sobre otros visibles), creación o destrucción de objetos o el envío de una señal a otro objeto.

Recordemos el diagrama de estado de la puerta: ¿Qué significa la signatura de la transición "cerrar [paso.vacio]"?



- ☐ Que cuando cerremos la puerta el paso quedará vacío.
- ◉ **Que para cerrar la puerta el paso debe estar vacío.**
- ☐ Que cuando se está cerrando la puerta se vacía el paso.

Los corchetes en un diagrama UML significan una condición que se debe cumplir. No se podrá cerrar la puerta hasta que el paso no esté vacío.

6.3 Creación de un diagrama de estados

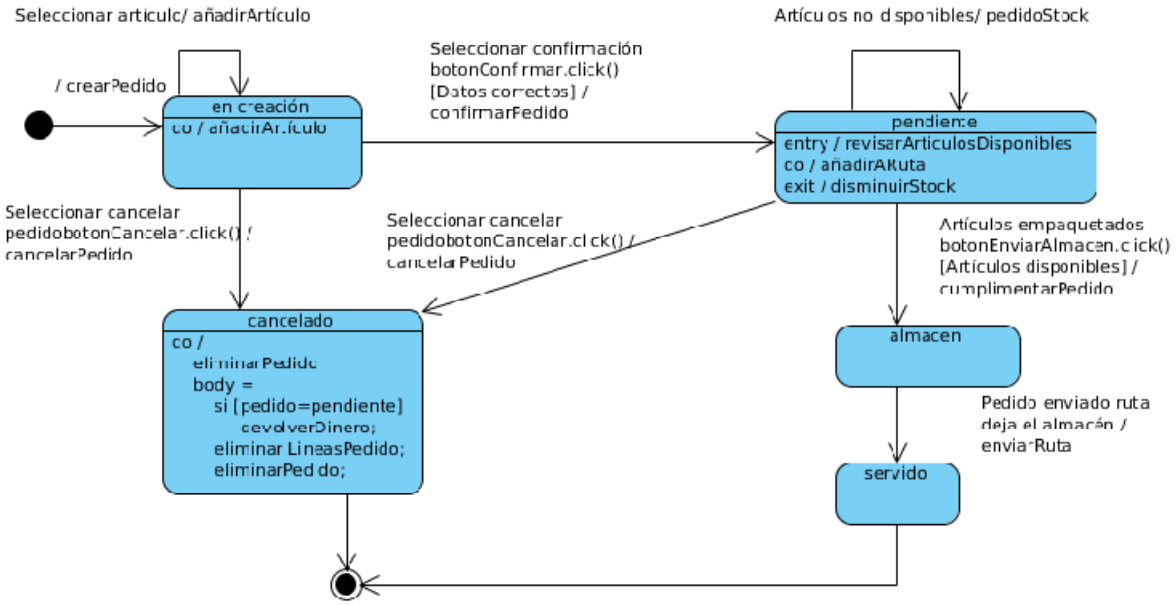
Para ejemplificar la creación de un diagrama de estados vamos a ver el que se asocia al objeto pedido, que cumple con las condiciones que hemos visto al principio, tiene un comportamiento significativo en tiempo real, ya que su situación tanto física, como el sistema, va evolucionando conforme pasa el tiempo, y participa en varios casos de uso (como Hacer pedido y Cumplimentar pedido).

Los diferentes estados en los que puede estar un pedido son:

- ✓ **En creación**: es cuando se están seleccionando los productos que formará el pedido.
- ✓ **Pendiente**: está en este estado desde que se confirma el pedido hasta que se selecciona para preparar su envío.
- ✓ **En almacén**: está en este estado cuando es elaborado el paquete y se ha asignado a una ruta, hasta que se envía a través de la ruta que le corresponde.
- ✓ **Servido**: Cuando el pedido es enviado. En este caso se envía una señal física desde el almacén cuando el transporte abandona el almacén.
- ✓ **Cancelado**: puede llegarse a esta situación por dos motivos, o bien se cancela mientras se está haciendo por problemas con la tarjeta de crédito, o bien porque, una vez pendiente de su gestión el usuario decide cancelarlo, la diferencia fundamental entre ambos es que en el segundo caso hay que devolver el importe pagado por el pedido al socio que lo ha comprado.

Las transiciones entre estados se producen por llamadas a procedimientos en todos los casos, no intervienen cambios de estado o el tiempo, ni señales.

El diagrama quedaría de la siguiente manera:



En las **transiciones** se ha incluido el nombre de la transición, el evento que la dispara (normalmente hacer clic en algún botón de la interfaz), si existe condición de guarda se pone entre corchetes y la acción a realizar para llegar al siguiente estado junto al símbolo /. En todos los casos el evento de disparo es de tipo llamada (incluye la llamada a una función o pulsar un botón de la interfaz), salvo en el caso de pedido enviado que se controla por una señal que se envía cuando el transporte abandona el almacén.

A los **estados** se les ha añadido la acción a realizar, apartado do/ y en algunos casos la acción de entrada, por ejemplo en el caso del estado pendiente, se debe revisar que los artículos a enviar tengan disponibilidad y la de salida, en el ejemplo disminuir el stock.

Nota: para incluir las condiciones de guarda en el diagrama debes rellenar el apartado “Guard” de la especificación, si necesitas añadir alguna acción puedes hacerlo rellenando el apartado “Effect”. Los eventos de disparo.