2. DIAGRAMAS DE CASOS DE USO

2.1. INTRODUCCIÓN

Los casos de uso fueron normalizados por Ivar Jacobson. Describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de un usuario. También permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.

Los casos de uso vienen a llenar una laguna de los primeros métodos de objetos, como OMT-2 y Booch-94, que no proponían técnicas para la determinación de las necesidades. En este sentido, los casos de uso asociados a las técnicas de objetos permiten una aproximación completa para todo el ciclo de vida, desde el cuaderno de carga hasta la implementación.

2.2. DIAGRAMAS DE CASOS DE USO

Los diagramas de casos de uso se utilizan durante la fase de análisis de un proyecto para identificar y dividir la funcionalidad del sistema. Normalmente contienen: casos de uso, actores y relaciones entre ellos: de asociación, de dependencia y/o de generalización.

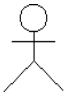
2.2.1. Casos de uso

Los casos de uso describen el comportamiento del sistema cuando uno de los actores envía un estímulo concreto. Este comportamiento se puede explicar de forma gráfica y/o textual, describiendo la naturaleza del estímulo que dispara el caso de uso: cuáles son las entradas y las salidas a otros actores y los comportamientos que convierten las entradas en las salidas. El texto del caso de uso también puede describir aquellos errores que pueden ocurrir durante el curso del comportamiento especificado, y qué solución va a tomar el sistema.

Los casos de uso se representan gráficamente por medio de una burbuja en forma de elipse y denotan los requisitos funcionales del sistema. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo. Va acompañado de un nombre significativo.

2.2.2. Actores

Los actores interpretan papeles que pueden ser representados por los usuarios del sistema. Dichos usuarios pueden ser humanos, otros ordenadores o incluso otros sistemas de software. El único criterio es que tienen que ser externos a la parte del sistema que se ha dividido en casos de uso. Deben suministrar un estímulo a dicha parte y generar salidas a partir del mismo.

Los actores se representan mediante un  , acompañado de un nombre significativo, si es necesario.

2.2.3. Relaciones en un diagrama de casos de uso

Por defecto, en UML se supone que las relaciones son bidireccionales, excepto cuando están presentes las puntas de flecha para restringirlas. Entre los elementos de un diagrama de casos de uso se pueden presentar tres tipos de relaciones:

- Relación de asociación

Relación entre un actor y un caso de uso. Una asociación entre un actor y un caso de uso indica que el actor y el caso de uso se comunican entre sí, enviándose y recibiendo mensajes. Se trata de una relación bidireccional, que se representa por una línea continua pero sin flechas en el origen y en el destino. Sin embargo, hay circunstancias en las que sí es necesario limitar la comunicación pues sólo se utiliza en una sola dirección, en cuyo caso el sentido de la misma se indica mediante la punta de una flecha.

- Relación de dependencia

Relación entre casos de uso. Se representa por una línea discontinua dirigida entre ellos (del elemento dependiente al independiente). Hay dos tipos:

- Incluye *«includes»* o Utiliza *«uses»*. Significa que una instancia del caso de uso fuente comprende también el comportamiento descrito por el caso de uso destino. Es decir, denota la inclusión del comportamiento de un escenario en otro.
- Extiende *«extends»*. Significa que el caso de uso fuente extiende el comportamiento del caso de uso destino. Es decir, denota cuando un caso de uso es una especialización de otro.

- Relación de generalización

Relación entre casos de uso y raras veces entre actores. La generalización entre casos de uso es igual que la generalización entre clases. Aquí significa que el caso de uso hijo hereda el comportamiento y el significado del caso de uso padre; el hijo puede añadir o incluso invalidar el comportamiento de su padre. Se representa por una línea continua dirigida entre ellos (del hijo al padre) con la punta de una flecha en forma triangular.

2.2.4. Ejemplos

1) Consideremos un punto de un sistema de venta. Uno de los actores es el cliente y otro es el dependiente. Uno de los casos de uso de este sistema es:

Caso de uso 1: El dependiente comprueba un artículo

1. El cliente coloca el artículo en el mostrador.
2. El dependiente pasa el lector de códigos de barras por el código CPU (Código de Producto Universal) que figura en el artículo.
3. El sistema busca el código CPU en la base de datos proporcionando la descripción y el precio del artículo.
4. El sistema emite un pitido audible.
5. El sistema añade el precio y el tipo del artículo a la factura actual.
6. El sistema añade al subtotal el IVA correspondiente a dicho artículo.

Caso de error 1: Código CPU no leíble.

Si después del paso 2 el código CPU no es válido o no es posible leerlo de forma apropiada, emitir un sonido grave audible.

Caso de error 2: Artículo no está en la base de datos.

Si después del paso 3 el código CPU del artículo no se encuentra en la base de datos, pulsar el botón de "entrada manual" en el terminal e introducir (el dependiente) el código CPU, el precio y el código del IVA de dicho artículo. En la descripción teclear "Artículo desconocido". Ir al paso 4.

Un punto de un sistema de venta tiene muchos más casos de uso que éste. En la Figura 2.1 viene representado este caso de uso según la forma gráfica de UML.

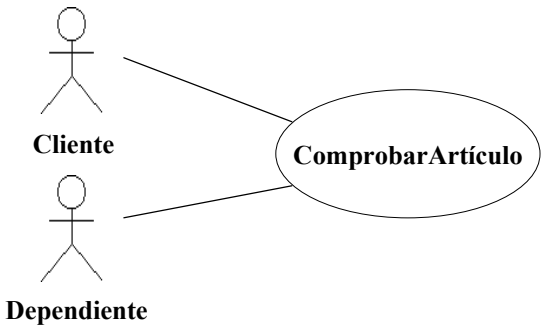


Figura 2.1

A primera vista parece una notación simple, pero hay mucho más. Los iconos correspondientes a los actores y a los casos de uso se pueden reunir en diagramas que delimitan el sistema complejo. Tales diagramas muestran todos los casos de uso de un sistema rodeado por un rectángulo. Fuera del rectángulo están todos los actores del sistema y ellos se conectan a sus casos de uso mediante líneas. El rectángulo representa el límite del sistema, por ejemplo, muestra todos los casos de uso que pertenecen a un sistema en concreto. Todo lo que está dentro del rectángulo forma parte del sistema, mientras que lo de fuera es externo al mismo. En la Figura 2.2 vienen representados los actores y los casos de uso de un sistema limitado.

Además de mostrar el límite del sistema, unos pocos actores y unos pocos casos de uso, este diagrama muestra algunas relaciones entre los casos de uso. Estas relaciones son *«utiliza»* y *«extiende»*.

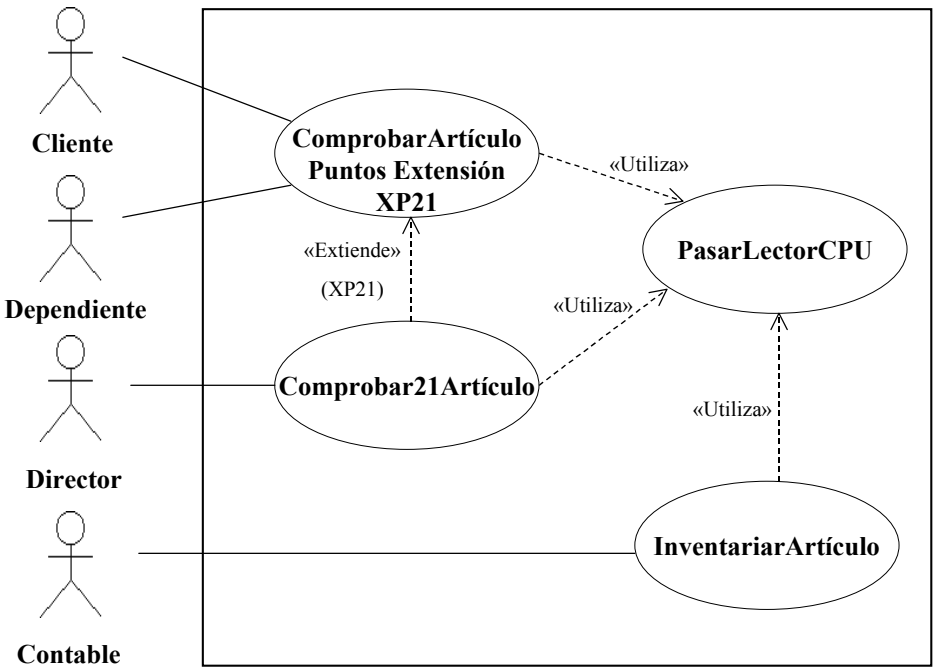


Figura 2.2

La relación *«utiliza»* aparece dos veces en la Figura 2.2, una desde ComprobarArtículo hasta PasarLectorCPU y la otra desde InventariarArtículo hasta PasarLectorCPU. El caso de uso PasarLectorCPU describe el comportamiento del actor y del sistema cuando el actor desliza el lector de CPU por el código de barras de un producto.

Si nos fijamos en nuestro primer caso de uso: **El dependiente comprueba un artículo**, el comportamiento del caso de uso PasarLectorCPU aparece dentro de la descripción ComprobarArtículo. Asimismo, el contable también utiliza el lector de códigos de barras cuando necesita hacer inventario de un artículo. Por lo tanto, este mismo comportamiento forma parte del caso de uso InventariarArtículo. Mejor que escribir la descripción para este comportamiento dos veces, podemos emplear la relación *«utiliza»* para mostrar que pertenece a ambos casos de uso.

Teniendo en cuenta esto podemos cambiar la descripción del caso de uso ComprobarArtículo:

Caso de uso 1: El dependiente comprueba un artículo

1. El cliente coloca el artículo en el mostrador.
2. *«utiliza»* PasarLectorCPU.
3. El sistema busca el código CPU en la base de datos proporcionando la descripción y el precio del artículo.
4. El sistema emite un pitido audible.
5. El sistema añade el precio y el tipo del artículo a la factura actual.
6. El sistema añade al subtotal el IVA correspondiente a dicho artículo.

Entonces, la relación *«utiliza»* es mucho más que una llamada a una función o una subrutina. El caso de uso empleado de esta forma se llama caso de uso abstracto, puesto que no puede existir por si solo sino que tiene que ser utilizado por otros casos de uso.

La otra relación importante es la relación *«extiende»* entre ComprobarArtículo y Comprobar21Artículo. En muchas tiendas norteamericanas, a los dependientes menores de 21 años no les está permitido dispensar bebidas alcohólicas. Cuando un empleado de esta edad ve en el mostrador un artículo que contiene alcohol, dice "21" a través del sistema de megafonía. Enseguida se acerca un director y pasa el lector de CPU por el código de barras del licor. Esto representa un cambio en el caso de uso ComprobarArtículo que se puede resolver de dos formas.

Primero, podríamos añadir sentencias "si" al caso de uso ComprobarArtículo de la siguiente forma:

Caso de uso 1: El dependiente comprueba un artículo

1. El cliente coloca el artículo en el mostrador.
2. Si el artículo es una bebida alcohólica:
   2.1. Avisar "21" por el sistema de megafonía.
   2.2. Esperar al director.
   2.3. El director *«utiliza»* PasarLectorCPU.
   2.4. Ir a paso 4.
3. *«utiliza»* PasarLectorCPU.
4. El sistema busca el código CPU en la base de datos proporcionando la descripción y el precio del artículo.
5. El sistema emite un pitido audible.
6. El sistema añade el precio y el tipo del artículo a la factura actual.
7. El sistema añade al subtotal el IVA correspondiente a dicho artículo.

Esta solución tiene una gran desventaja teniendo en cuenta el Open Closed Principle. Este principio dice que, en un software bien diseñado, un cambio en los requisitos debería implicar el hecho de añadir código nuevo y no modificar el código viejo. Las mismas reglas son aplicables a las especificaciones funcionales de los casos de uso. Es decir, cuando los requisitos cambian debemos añadir nuevos casos de uso, no modificar los ya existentes.

Segundo, mejor que añadir la sentencia "si" al caso de uso, podemos utilizar la relación *«extiende»*. Esta relación nos permite especificar un nuevo caso de uso que contiene comandos para invalidar y modificar el caso de uso extendido. Entonces el caso de uso Comprobar21Artículo de la Figura 2.2 anula y extiende el caso de uso ComprobarArtículo. El texto para el caso de uso Comprobar21Artículo podría aparecer:

Caso de uso 2: Comprobar "21" artículo

1. Sustituir Paso 2 de ComprobarArtículo por:
   1.1. Avisar "21" por el sistema de megafonía.
   1.2. Esperar al director.
   1.3. El director *«utiliza»* PasarLectorCPU.

Con esto se logra nuestro objetivo de añadir nuevas características al modelo de casos de uso sin necesidad de cambiar los casos de uso existentes.

Si nos fijamos en el caso de uso Comprobar21Artículo menciona el caso de uso ComprobarArtículo directamente. ¿Qué ocurre si queremos extender otros casos de uso similares? Todos los nuevos casos de uso que se extiendan deberían ser idénticos. Esta situación se puede solventar añadiendo puntos de extensión a los casos de uso extendidos. Los puntos de extensión son nombres simbólicos simples que identifican posiciones en el caso de uso extendido (*use case extended*) y que los nuevos casos de uso a extender (*use case extending*) pueden mencionar en su texto. Entonces, nuestros dos casos de uso quedarían así:

Caso de uso 1: El dependiente comprueba un artículo

1. El cliente coloca el artículo en el mostrador.
2. *XP21*: El dependiente *«utiliza»* PasarLectorCPU.
3. El sistema busca el código CPU en la base de datos proporcionando la descripción y el precio del artículo.
4. El sistema emite un pitido audible.
5. El sistema añade el precio y el tipo del artículo a la factura actual.
6. El sistema añade al subtotal el IVA correspondiente a dicho artículo.

Caso de uso 2: Comprobar "21" artículo

2. Sustituir XP21 del caso de uso extendido por:
   2.1. Avisar "21" por el sistema de megafonía.
   2.2. Esperar al director.
   2.3. El director *«utiliza»* PasarLectorCPU.

2) Para estudiar con más detalle las relaciones de dependencia y de generalización, vamos a considerar los siguientes casos de uso de un sistema de un banco, que aparecen dibujados en la Figura 2.3.

Una relación *«utiliza»* se usa para evitar describir el mismo flujo de eventos varias veces, incluyendo el comportamiento común en un caso de uso de su propiedad. Se puede decir que es esencialmente un ejemplo de delegación.

Una relación *«utiliza»* se traduce como una dependencia, estereotipada como *«utiliza»*. Para especificar la localización en un flujo de eventos en el que un caso de uso origen incluye el comportamiento de otro, simplemente se escribe *«utiliza»* seguido del nombre del caso de uso que se quiere incluir. Por ejemplo, el caso de uso SeguirPistaOrden tiene el siguiente flujo:

1. El oficinista obtiene y verifica el número de la orden.
2. *«utiliza»* ValidarUsuario.
3. Por todas las cuestiones relacionadas con la petición (orden), pregunta su estado y luego informa al usuario.

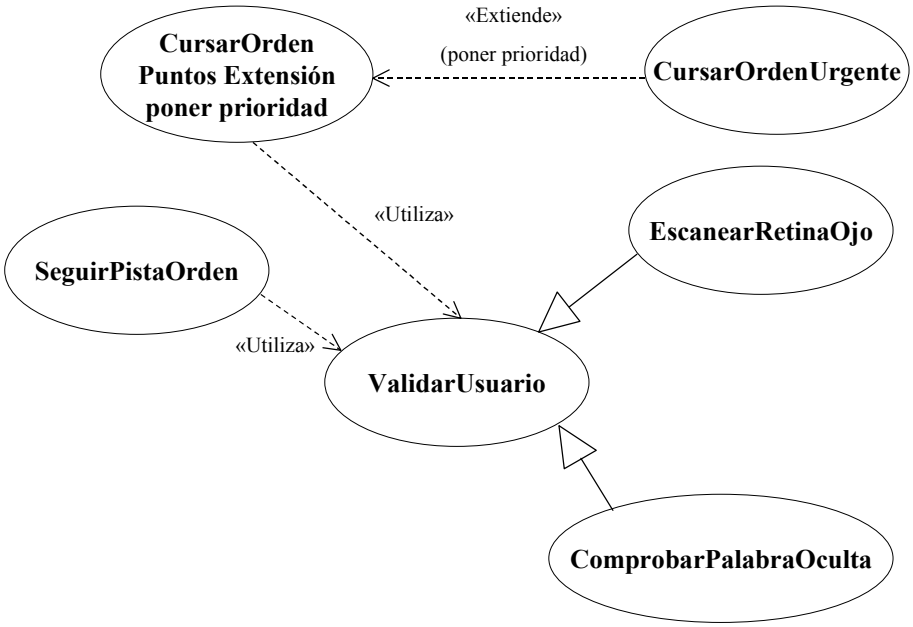


Figura 2.3

Una relación *«extiende»* se usa para modelar la parte de un caso de uso que el usuario puede ver como un comportamiento opcional del sistema. De esta forma, se separa el comportamiento opcional del obligatorio. También se puede utilizar para modelar un subflujo que es ejecutado bajo ciertas condiciones.

Una relación *«extiende»* se traduce como una dependencia, estereotipada como *«extiende»*. Para especificar la localización en un flujo de eventos en el que un caso de uso extiende su comportamiento a otro, se emplea lo que se denomina punto de extensión. Éste se inserta en el caso de uso extendido y simplemente se menciona en el caso de uso que se va a extender. Por ejemplo, el caso de uso extendido CursarOrden tiene el siguiente flujo:

1. *«utiliza»* ValidarUsuario.
2. Ir recogiendo las diferentes órdenes del usuario.
3. *poner prioridad*.
4. Procesar la orden.

En este ejemplo, *poner prioridad* es un punto de extensión. Un caso de uso puede tener más de un punto de extensión (que puede aparecer más de una vez). Bajo circunstancias normales, el caso de uso extendido se ejecutará sin tener en cuenta la prioridad de la orden. Si, en cambio, se trata de una instancia de una orden prioritaria, el flujo para este caso de uso se desarrollará como antes. Pero en el punto de extensión *ponerprioridad*, se ejecutará el comportamiento del nuevo caso de uso CursarOrdenUrgente, y luego el flujo seguirá su curso.

El caso de uso ValidarUsuario, que es responsable de verificar la identidad del usuario, tiene dos casos de uso (ComprobarPalabraOculta y EscanearRetinaOjo). Ambos casos de uso son hijos especializados de ValidarUsuario y añaden su propio comportamiento

(el primero verifica la palabra de paso textual, mientras que el segundo busca entre los modelos de retina de ojo el correspondiente al usuario). La generalización entre estos casos de uso está representado en la Figura 2.3 por una línea continua dirigida del hijo al padre con la punta de una flecha en forma triangular.