Análisis de Sistemas. Modelo de Casos de Uso.

El modelado de caso de uso es otra forma de ingeniería de requerimientos. A continuación se presenta el modelado de caso de uso según la propuesta de Arlow y Neustadt (2006), Capítulos 4 y 5, donde se sigue el proceso unificado de desarrollo.

UML no especifica ninguna estructura formal para la especificación de caso de uso. Esto es problemático, ya que diferentes modeladores adoptan estándares diferentes. Para ayudar con esto, Arlow y Neustadt (2006) han adoptado plantillas de documentación que emplearon a modo de ejemplo.

Modelado de caso de uso

El modelado de caso de uso es una forma de ingeniería de requerimientos. En unidades previas vimos cómo crear un modelo de requerimiento que englobaba requerimientos funcionales y no funcionales en lo que podríamos denominar la forma “tradicional”. El modelado de case de uso es una forma diferente y complementaria de crear y documentar requerimientos. El modelado de cases de uso funciona de la siguiente manera:

- Encontrar el límite de un sistema candidato.
- Encontrar los actores.
- Encontrar los casos de uso:
    - Especificar el caso de uso.
    - Identificarlos flujos alternativos clave.
- Repetir hasta que los casos de uso, actores y límite del sistema son estables.

Generalmente, empieza con una estimación inicial de donde se encuentra el límite del sistema para ayudarle a definir el ámbito de aplicación de la actividad de modelado. Las acciones se realizan luego iterativamente y a menudo en paralelo.

El resultado de estas actividades es el modelo de caso de uso. Existen cuatro componentes de este modelo.

- **Límite del sistema**: Un cuadro dibujado alrededor de los casos de uso para indicar el borde o límite del sistema que se modela. Esto se conoce como el sujeto en UML 2.
- **Actores**: Roles desempeñados por personas o elementos que utilizan el sistema.
- **Casos de uso**: Lo que los actores pueden hacer con el sistema.
- **Relaciones**: Relaciones significativas entre los actores y casos de uso.

El modelo de caso de uso proporciona una fuente principal para identificar objetos y clases. Por tal motivo algunos autores lo consideran la entrada principal para el modelado de clases.

Encontrar actores y casos de uso

En la Figura 1 se ilustra la actividad encontrar actores y casos de uso definida como parte del workflow de requerimientos (proceso Unificado de desarrollo). En la figura se indica que la actividad es llevada a cabo por el Analista de sistema, la misma permite definir el Modelo de caso de uso y el glosario del proyecto. Además, en la figura se definen las siguientes entradas para la realización de dicha actividad:

- Modelo de negocio (o modelo de dominio): Puede tener o no un modelo de negocio disponible que esté relacionado con el sistema que esté modelando. Si lo tiene, ésta es una excelente fuente de requisitos.
- Modelo de requisitos: Describimos la creación de este modelo unidades previas. Estos requisitos proporcionan entrada de utilidad para el proceso de modelado de caso de uso. En particular, los requisitos funcionales sugerirán casos de uso y adores. Los requisitos no funcionales sugerirán algo que debería mantener en mente cuando construye el modelo de caso de uso.
- Lista principal: Este es un conjunto de requisitos candidatos que pueden adoptar la forma de un documento de visión o similar.

En el trabajo original de Jacobson, el modelo de requisitos se reemplazó por requisitos adicionales. Este documento constaba de requisitos (normalmente no funcionales) que no estaban relacionados con ningún caso de uso en particular.

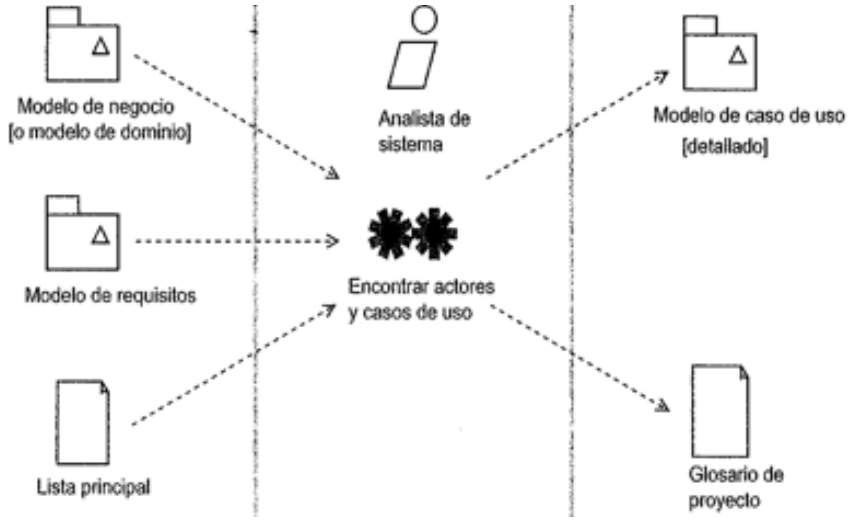

Figura 1. Actividad Encontrar actores y casos de uso

El documento de requisitos adicionales fue inicialmente algo general para requisitos no funcionales que atraviesa los casos de uso.

El sujeto (límite del sistema)

Lo primero que necesita hacer cuando piense en crear un sistema es decidir donde se encuentran los limites del sistema. Es decir, necesita decidir qué es parte de su sistema (dentro de los límites del sistema) y qué es externo a su sistema (fuera de los límites del sistema). Esto suena obvio, pero en muchos proyectos han surgido problemas serios por unos límites del sistema dudosos. La posición del límite del sistema tiene un impacto enorme sobre los requisitos funcionales (y a veces no funcionales) y ya ha visto que los requisitos incompletos y mal especificados pueden ser la razón principal de que los proyectos fracasen. En UML 2, los límites del sistema se conocen como el sujeto, y éste es el término que utilizaremos de ahora en adelante.

El sujeto está definido por quién o qué utiliza el sistema (los actores) y qué beneficios específicos ofrece el sistema a esos actores (los casos de uso).

El sujeto se dibuja como un cuadro (un rectángulo), etiquetado con el nombre del sistema, con los actores dibujados fuera del límite y los casos de uso dentro. Empezará el modelado de los casos de uso solamente con una idea provisional de dónde se encuentra realmente el sujeto. A medida que encuentre los actores y caso de uso, el sujeto estará cada vez más definido.

¿Qué son actores?

Un actor especifica un rol que cierta entidad externa adopta cuando interactúa con su sistema directamente. Puede representar un rol de usuario, o un rol desempeñado por otro sistema o hardware que toca el límite de su sistema.

En UML 2, los actores también pueden representar otros sujetos, proporcionándole una forma de vincular diferentes modelos de caso de uso.

Para entender los actores, es importante entender el concepto de roles. Piense en un rol como un sombrero que un elemento lleva en un contexto particular. Los elementos pueden tener muchos roles simultáneamente y con el tiempo. Esto significa que un rol puede desempeñarse por muchos elementos diferentes simultáneamente y con el tiempo.

Por ejemplo, si hemos identificado el actor Cliente para nuestro sistema, la persona real Beatriz, María, Nicolás y muchas otras pueden desempeñar ese rol. Esas personas pueden desempeñar también otros roles. Por ejemplo, María podría también administrar el sistema (el actor AdministradorSistema) al igual que utilizarlo como un Cliente. El error más fundamental que los principiantes realizan en el modelado de casos de uso es confundir un rol que desempeña algo en el contexto del sistema con el propio elemento. Siempre pregúntese "¿qué rol desempeña el elemento con respecto al sistema?". De esta forma, puede encontrar comportamiento común entre muchos elementos diferentes y de esta forma simplificar su modelo de caso de uso.

Los actores están representados en UML como se muestra en la Figura 2. En la Figura se ilustra a los actores Cliente y AdministradorSistema con el icono de actor de figura de hombre. Además, es posible ilustrar los actores con un icono de clase (un rectángulo) estereotipado <<actor>>. Ambas formas de notación de actor son válidas, pero muchos modeladores prefieren utilizar la figura de hombre para representar roles que desempeñarán personas, y forma de icono de clase (rectángulo) para representar roles que desempeñarán otros sistemas. En la parte izquierda de la figura se describen las distintas formas de definir actores en PlantUML.

```
@startuml
:Cliente:
actor AdministradorSistema
@enduml
```

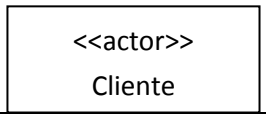

Figura 2. Representación de actor

Es importante darse cuenta que los actores son siempre externos al sistema. Por ejemplo, si utiliza un sistema de comercio electrónico como una librería online para comprar un libro, entonces usted es externo a ese sistema. Sin embargo, es interesante indicar que, aunque los

propios actores son siempre externos al sistema, los sistemas a menudo mantienen alguna representación interna de uno o más actores. Por ejemplo, la librería online mantendrá un objeto de detalles de cliente para la mayoría de clientes que contiene su nombre, dirección y otra información. Esta es una representación interna del sistema del actor externo Cliente. Ahora, es importante ser muy claro sobre esta diferencia, el actor Cliente es externo al sistema, pero el sistema podría mantener una clase DetallesCliente que es una representación interna de personas que desempeñan el rol del actor Cliente.

Identificar actores

Para identificar los actores, necesita considerar quién y qué utiliza el sistema y qué roles desempeñan en sus iteraciones con el sistema. Puede llegar a los que desempeñan personas y elementos en relación a un sistema al considerar casos de personas y elementos específicos y luego generalizar. Formular las siguientes preguntas le ayudará a identificar actores.

- ¿Quién y qué utiliza el sistema?
- ¿Qué roles desempeñan en la iteración?
- ¿Quién instala el sistema?
- ¿Quién o qué inicia y cierra el sistema?
- ¿Quién mantiene el sistema?
- ¿Qué otros sistemas interactúan con este sistema?
- ¿Quién o qué consigue y proporciona información al sistema?
- ¿Sucede algo en un momento dado?

En términos de modelar actores, recuerde los siguientes puntos:

- Los actores siempre son externos al sistema; están por lo tanto fuera de su control.
- Los actores interactúan directamente con el sistema; así es como ayudan a definir el sujeto.
- Los actores representan roles que personas y elementos desempeñan en relación al sistema, no personas específicas o elementos específicos.
- Una persona o elemento puede desempeñar muchos roles en relación al sistema simultáneamente o con el tiempo. Por ejemplo, si escribiera al igual que impartiera cursos de formación, desde la perspectiva de un sistema de planificación de cursos, desempeñaría dos roles: Formador y AutorCurso.
- Todo actor necesita un nombre breve que tenga sentido desde la perspectiva del negocio.
- Todo actor debe tener una breve descripción (una o dos líneas) que describa qué es este actor desde una perspectiva de negocio.
- Como las clases, los actores pueden tener compartimentos que muestran atributos del actor y eventos que el actor puede recibir. Normalmente, estos compartimentos no se utilizan demasiado y apenas se muestran en diagramas de caso de uso. Por lo tanto, no los consideraremos.

Tiempo como un actor

Cuando necesita modelar elementos que suceden a su sistema en un punto específico en el tiempo pero que no parecen estar activados por ningún actor, puede introducir un actor denominado Tiempo según se ilustra en la Figura 3. Un ejemplo de esto sería un backup automático del sistema que se ejecuta todas las noches.

@startuml :Tiempo: @enduml	

Figura 3. Representación de actor Tiempo

¿Qué son casos de uso?

The UML Reference Manual (Rumbaugh y colab.,) define un caso de uso como una especificación de secuencias de acciones, incluidas secuencias variantes y secuencias de error que un sistema, subsistema o clase puede realizar al interactuar con actores externos".

Un caso de uso es algo que el actor quiere que el sistema haga. Es un "caso de uso" del sistema por un actor específico:

- Los casos de uso se inician siempre por un actor.
- Los casos de uso se escriben siempre desde el punto de vista de los actores.

Normalmente pensamos en los casos de uso en el nivel del sistema, pero como indica la definición, también podemos aplicar casos de uso para describir "casos de uso" de un subsistema (parte de un sistema) o incluso una clase individual. Los casos de uso pueden ser también muy eficaces en el modelado de procesos de negocio, aunque no abordamos ese aspecto en el curso.

El icono UML para los casos de uso es un óvalo, como se muestra en la Figura 4. El nombre del caso de uso se puede escribir dentro o debajo del óvalo. En la parte izquierda de la Figura se incluye la notación en PlantUML. Es posible escribir el nombre del caso de uso entre paréntesis o mediante la palabra reservada usecase, el nombre entre paréntesis y luego mediante as definir un nombre, el cual podrá ser usado luego en el modelo.

@startuml (CursarPedido) usecase (EstadoDePedido) as UCEdP @enduml	

Figura 4. Representación de casos de uso

Identificar casos de uso

La mejor forma de identificar casos de uso es empezar con la lista de actores y luego considerar cómo cada actor va a utilizar el sistema. Utilizando esta estrategia puede obtener una lista de casos de uso candidatos. Cada caso de uso debe tener asignado un nombre descriptivo, breve, que es una frase verbal; después de todo, el caso de uso es hacer algo. A medida que identifique casos de uso, también puede encontrar algunos nuevos actores. Esto está bien. Algunas veces tiene que considerar la funcionalidad del sistema muy detenidamente antes de encontrar todos los actores o todos los actores correctos.

El modelado de caso de uso es iterativo y continúa vía un proceso de mejora de pasos. Empieza con un nombre para un caso de uso y completa los detalles más adelante. Estos detalles constan de una breve descripción inicial que se convierte en una especificación completa. Aquí tiene una lista de preguntas que puede formular cuando trate de identificar casos de uso:

- ¿Qué funciones querrá un actor específico del sistema?
- ¿El sistema almacena y recupera información? Si es así, ¿qué actores activan este comportamiento?
- ¿Qué sucede cuando el sistema cambia de estado (por ejemplo, iniciar o detener el sistema)? ¿Se notifica a algún actor?
- ¿Afecta algún evento externo al sistema? ¿Qué notifica el sistema sobre estos eventos?
- ¿Interactúa el sistema con algún sistema externo? ¿Genera el sistema algún informe?

El diagrama de casos de uso

En el diagrama de caso de uso representa el sujeto del modelo de caso de uso por un cuadro (rectángulo) etiquetado con el nombre del sujeto. Este cuadro es el sujeto y, como se acaba de mencionar anteriormente, representa el límite del sistema modelado por los casos de uso. Muestra actores fuera del sujeto (externos al sistema) y casos de uso que constituyen el comportamiento del sistema dentro del sujeto, interno al sistema. Esto se ilustra en la Figura 5. El diagrama está formado por el sujeto Sistema_pedido_por_correo. Los casos de uso definidos son CursarPedido, CancelarPedido, ComprobarEstado, SolicitarCatálogo, y EnviarProducto. Los actores son Cliente, EmpresaTransporte, y Transportista. En primer lugar en la Figura se incluye la notación en PlantUML.

```
@startuml
rectangle Sistema_pedido_por_correo {
   (CursarPedido)
   (CancelarPedido)
   (ComprobarEstado)
   (SolicitarCatálogo)
   (EnviarProducto)
}
:Cliente:
:EmpresaTransporte:
:Transportista:
Cliente -- (CursarPedido)
Cliente -- (CancelarPedido)
Cliente -- (ComprobarEstado)
Cliente -- (SolicitarCatálogo)
EmpresaTransporte -- (EnviarProducto)
Transportista -- (EnviarProducto)
@enduml
```

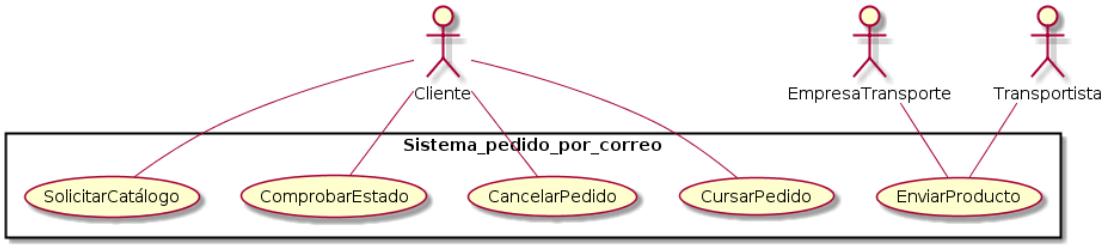

Figura 5. Diagrama de casos de uso

La relación entre un actor y un caso de uso se muestra por una línea continua. En PlantUML se la define mediante una serie de guiones (Cliente -- (CursarPedido)). La asociación entre el actor y el caso de uso indica que el actor y el caso de uso se comunican de alguna forma.

El glosario del proyecto

Todo ámbito de negocio tiene su propio lenguaje único y la finalidad principal de la ingeniería de requisitos y análisis es entender y capturar ese lenguaje. El glosario proporciona un diccionario de términos claves de negocio y definiciones. Debería ser entendible por todo el mundo en el proyecto, incluidos todos los grupos de decisión.

Además de definir los términos clave, el glosario del proyecto debe resolver sinónimos y homónimos.

- Los sinónimos son palabras diferentes que significan lo mismo. Como analista orientado a objetos debe elegir una de estas palabras (la que se utilice más ampliamente) y ajustarse a ella. Las otras variantes se deberían excluir de sus modelos. La razón es que si permite el uso de sinónimos, puede acabar con dos clases que hacen más o menos lo mismo pero tienen nombres diferentes. Igualmente, si permite el uso de todos los sinónimos, puede estar seguro de que la semántica actual de los términos divergirá gradualmente con el tiempo.
- Los homónimos ocurren cuando la misma palabra significa cosas diferentes a diferentes personas. Esto siempre da lugar a problemas de comunicación ya que las diferentes partes hablan diferentes idiomas cuando creen que hablan el mismo idioma. Nuevamente, la forma de resolver esto es elegir un significado para el término y quizás introducir nuevos términos para los otros homónimos.

En el glosario del proyecto, debería grabar el término preferido y listar cualquier sinónimo bajo la definición. Esto puede implicar animar a sus grupos de decisión del negocio a que se acostumbren a diferente terminología. A menudo, es una tarea difícil conseguir que los grupos de decisión cambien su uso del idioma y con mucha persistencia se puede conseguir. UML no establece ningún estándar para un glosario de proyecto. Es una buena práctica mantenerlo lo más sencillo y conciso posible. Utilice un formato como el de un diccionario con una lista ordenada alfabéticamente de palabras y definiciones. Un documento basado en texto puede ser suficiente, pero los proyectos más grandes pueden necesitar un glosario online basado en HTML o XML o incluso una base de datos sencilla. Recuerde que cuanto más accesible y fácil de utilizar sea el glosario, más positivo será el impacto que tendrá sobre el proyecto. En la Tabla 1 se incluye parte de un glosario de proyecto de ejemplo. Como parte del estilo, siempre escribimos "Ninguno" si no existen sinónimos u homónimos, en lugar de dejar los espacios en blanco u omitirlos. Esto muestra que hemos considerado el término.

Una consideración con el glosario del proyecto es que los términos y definiciones en el glosario también se utilizarán en el modelo UML. Tiene que asegurarse que los dos documentos se mantienen sincronizados. Desafortunadamente, la mayor parte de las herramientas de modelado de UML no proporcionan ningún soporte para esto y normalmente es una actividad manual.

Tabla 1. Glosario de proyecto para la plataforma de comercio electrónico de Clear View Training.

Término	Definición
Catálogo.	Listado de todos los productos que en estos momentos se ofrecen a la venta. Sinónimos: Ninguno Homónimos: Ninguno
Pagar.	Una analogía electrónica de pagar en un supermercado en el mundo real. Un lugar donde los clientes pueden pagar los artículos en su carrito de la compra. Sinónimos: Ninguno Homónimos: Ninguno
Clear View Training.	Una empresa especializada en venta de libros y CD. Sinónimos: CVT Homónimos: Ninguno
Tarjeta de crédito.	Una tarjeta como Visa o Mastercard que se puede utilizar para pagar productos. Sinónimos: Tarjeta Homónimos: Ninguno
Cliente.	Una parte que compra productos o servicios en Clear View Training. Sinónimos: Ninguno Homónimos: Ninguno

Detallar un caso de uso

Habiendo creado un diagrama de caso de uso e identificado los actores y casos de uso claves, necesita empezar a especificar cada caso de uso. En la Figura 6 se ilustra la actividad Detallar un caso de uso. La actividad requiere como entrada el modelo de caso de uso en su versión preliminar, el modelo de requisitos y el glosario de proyecto. El resultado de esta actividad es un caso de uso más detallado. Esto consta de al menos el nombre del caso de uso y una especificación de caso de uso. Es importante que observe que en este punto no sigue una secuencia exacta y puede elegir especificar algunos o todos los casos de uso a medida que los encuentre.

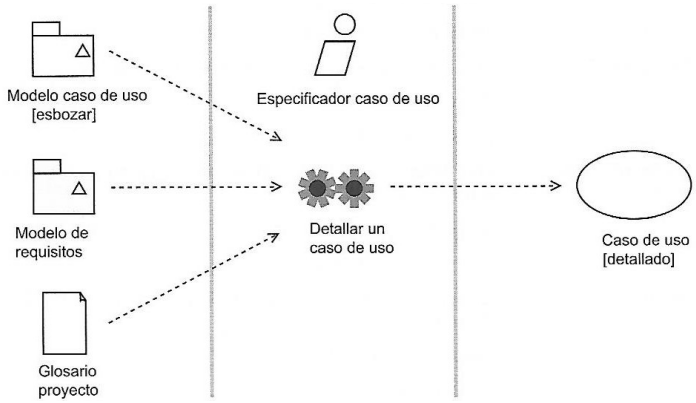

Figura 6. Actividad Detallar un caso de uso

Especificación de caso de uso

No existe estándar UML para una especificación de caso de uso, la plantilla mostrada en la Figura 7 es la que emplea Arlow y Neustadt (2006). Sin embargo, es importante que su organización decida un estándar para las especificaciones de caso de uso que se utilice coherentemente dentro de los proyectos. Existen múltiples formatos diferentes, niveles de detalles e incluso interpretaciones de lo que es, o no es un caso de uso. Un estándar sencillo y eficaz para las especificaciones de caso de uso puede ayudar a asegurar que su proyecto tiene éxito con el análisis del caso de uso. La planilla propuesta por Arlow y Neustadt (2006) contiene la siguiente información:

- Nombre del caso de uso.
- ID del caso de uso.
- Breve descripción. Un párrafo que indica el objetivo del caso de uso.
- Actores implicados en el caso de uso.
- Precondiciones. Condiciones que deben ser verdaderas antes de que el caso de uso pueda ejecutarse; son restricciones en el estado del sistema.
- Flujo principal. Los pasos en el caso de uso.
- Poscondiciones. Condiciones que deben ser ciertas al final del caso de uso.
- Flujos alternativos. Una lista de alternativas al flujo principal.

El caso de uso en la Figura 7 es sobre el pago del IVA, una forma de impuesto sobre las ventas en muchos países. En este ejemplo, la autoridad correspondiente siempre consigue su impuesto de una forma u otra, y por lo tanto lo establecemos como una poscondición del caso de uso. Una buena forma de escribir un caso de uso es utilizar lenguaje estructurado.

nombre caso de uso	Caso de uso: PagarImpuestoVentas
identificador caso de uso	ID: 1
descripción breve	Breve descripción: Pagar impuesto Ventas a Autoridad al final del trimestre.
actores implicados en el caso de uso	Actores principales: Tiempo.
el estado del sistema antes de que el caso de uso pueda empezar	Precondiciones: 1. Es el final del trimestre.
los pasos del caso de uso	Flujo principal: 1. El caso de uso empieza cuando es el final del trimestre. 2. El sistema determina la cantidad de Impuesto Ventas debido a la Autoridad. 3. El sistema envía un pago electrónico a la Autoridad.
el estado del sistema cuando el caso de uso ha terminado	Poscondiciones: 1. La Autoridad recibe la cantidad correcta de Impuesto Ventas.
flujos alternativos	Flujo alternativos: Ninguno.

Figura 7. Plantilla de especificación de caso de uso.

Nombre del caso de uso

No existe estándar UML para nombrar casos de uso. Siempre nombramos casos de uso poniendo en mayúscula la primera letra de cada palabra con la primera letra de todas en mayúscula. Las palabras del nombre del caso de uso se unen y cada palabra empieza con una letra en mayúscula. Los casos de uso describen el comportamiento del sistema, por lo que el nombre del caso de uso debería ser siempre un verbo o una frase verbal como Pagarlmpuesto. Siempre debería tratar de elegir un nombre que sea breve, pero descriptivo. Un lector de su modelo de caso de uso debería hacerse una idea clara de la función o proceso de negocio que el caso de uso está modelando solamente por el nombre del caso de uso. Los nombres de los casos de uso deben ser únicos en el modelo.

ID de caso de uso

Aunque los nombres del caso de uso deben ser únicos, dentro de su modelo de caso de uso, es posible que cambien con el tiempo. Es posible que desee añadir otro identificador inmutable que identifique de forma única un caso de uso particular dentro de su proyecto, el ID de caso de uso. Es común utilizar un número.

Cuando se trabaja con flujos alternativos, puede elegir utilizar un sistema jerárquico de numeración de modo que el flujo alternativo se pueda vincular con el flujo principal. Por ejemplo, si un caso de uso está numerado X, entonces sus flujos alternativos están numerados X. 1, X.2,..., X.n.

Breve descripción

Esto debería ser un solo párrafo que resuma el objetivo del caso de uso. Trate de captar la esencia del caso de uso, el beneficio de negocio que proporciona a sus actores.

Actores

Desde el punto de vista de un caso de uso específico, existen dos tipos de actores:

- Actores principales: Estos actores activan el caso de uso.
- Actores secundarios: Estos actores interactúan con el caso de uso después de haberse activado.

Cada caso de uso siempre se activa por un solo actor. Sin embargo, el mismo caso de uso puede activarse por diferentes actores en diferentes momentos en el tiempo. Cada actor que puede activar el caso de uso es un actor principal. El resto de actores son actores secundarios.

Precondiciones y poscondiciones

Las precondiciones y poscondiciones son restricciones.

- Las precondiciones restringen el estado del sistema antes de que el caso de uso pueda empezar. Piense en ellas como guardianes que impiden que un actor active el caso de uso hasta que se cumplan todas sus condiciones.
- Las poscondiciones restringen el estado del sistema después de que el caso de uso se ha ejecutado.

Otra forma de examinar esto es que las precondiciones especifican lo que debe ser cierto antes de que el caso de uso se pueda activar, y las poscondiciones especifican qué será verdadero después de que el caso de uso se haya ejecutado. Las precondiciones y poscondiciones le ayudan a diseñar sistemas que funcionan correctamente.

Las precondiciones y poscondiciones siempre deberían ser declaraciones sencillas sobre el estado del sistema que evaluará como verdadero y falso; esto se conoce como condiciones booleanas.

Si su caso de uso no tiene ninguna precondición o poscondición, entonces se debería escribir "Ninguno" en el apartado apropiado de la especificación del caso de uso. Esto demuestra que ha considerado el asunto, mientras que dejar el apartado en blanco es ambiguo.

Flujo principal

Los pasos en un caso de uso se listan en un flujo de eventos. Puede pensar en un caso de uso como el delta de un río con muchos canales. Cada caso de uso tiene un flujo principal que es el canal principal hacia el delta. Los otros canales más pequeños en el delta son flujos alternativos. Estos flujos alternativos pueden capturar errores e interrumpir el flujo principal. El flujo principal a veces se conoce como el escenario principal, y los flujos alternativos como escenarios secundarios.

El flujo principal lista los pasos en un caso de uso que capturan el "mundo perfecto", situación donde todo sucede según lo esperado y deseado, y no existen errores, desviaciones o interrupciones.

Puede modelar las desviaciones al flujo principal de dos formas, que se tratarán en breve.

1. Desviaciones simples: Crea ramificaciones en el flujo principal.
2. Desviaciones complejas: Escribe flujos alternativos.

El flujo principal siempre empieza por el actor principal haciendo algo para activar el caso de uso. Una buena forma de iniciar un flujo de eventos es como sigue:

1. El caso de uso empieza cuando un <actor><función>.

Recuerde que el tiempo puede ser un actor, por lo tanto el caso de uso puede también empezar con una expresión de tiempo en lugar del actor, como muestra la Figura 7.

El flujo de eventos consta de una secuencia de pasos breves que son declarativos, están numerados y ordenados en el tiempo. Cada paso en el flujo de caso de uso debería estar en la forma de

<número> El <algo> <alguna acción>.

El flujo de caso de uso de los eventos también se puede capturar como prosa. Sin embargo, no lo recomendamos ya que es demasiado impreciso. Aquí tiene un ejemplo de un par de pasos en un caso de uso Cursar Pedido.

1. El caso de uso empieza cuando el cliente selecciona “cursar pedido”.
2. El cliente escribe su nombre y dirección en el formulario.

Estos son casos correctos. En ambos casos tenemos una sencilla declaración declarativa de algo realizando alguna acción. Un ejemplo de un paso de caso de uso mal formado sería como el siguiente:

2. Se incorporan los detalles del cliente.

De hecho, cualquier paso escrito en pasiva está mal formado. Este caso en particular contiene tres omisiones importantes:

- ¿Quién incorpora los detalles del cliente?
- ¿En qué se incorporan los detalles del cliente?
- ¿Qué son específicamente "detalles del cliente"?

Es importante que reconozca y evite omisiones cuando escriba flujos de caso de uso. Aunque es posible que lo pueda saber por el contexto o averiguar lo que significa, éste no es el caso. El caso es que el caso de uso debería ser una declaración precisa de una pieza de la funcionalidad del sistema.

Cuando encuentra vaguedad, omisiones o generalizaciones durante el proceso de análisis, es de utilidad hacerse las siguientes preguntas:

- ¿Quién específicamente...?
- ¿Qué específicamente... ?
- ¿Cuándo específicamente... ?
- ¿Dónde específicamente...?

Ramificación dentro de un flujo

La especificación UML no especifica ninguna forma de mostrar ramificaciones dentro de un flujo. Utilizamos un modismo para mostrar la ramificación de una forma sencilla sin tener que escribir un flujo alternativo aparte. Utilizamos la palabra clave Si para indicar una ramificación.

Merece la pena saber que algunos modeladores de caso de uso pueden desaprobar la ramificación dentro de casos de uso. Argumentan que siempre que hay alguna ramificación, se debería escribir un nuevo flujo alternativo. En realidad, este argumento merece la pena tenerlo en cuenta; sin embargo, tomaremos el enfoque más pragmático de que una pequeña cantidad de ramificación en un flujo es deseable porque reduce el número total de flujos alternativos y conduce a una representación más compacta de los requisitos.

Palabra clave Si

Utilice la palabra clave Si para indicar una ramificación en un flujo. El ejemplo en la Figura 8 muestra un flujo de eventos bien estructurado con dos ramificaciones.

Toda ramificación va prefijada con la palabra clave Si y empieza con una sencilla expresión booleana como “Si el Cliente escribe una nueva cantidad” lo que es verdadero o falso. El texto bajo la declaración Si es lo que sucederá si la expresión booleana es verdadera. Puede indicar claramente el cuerpo de la declaración Si por medio del uso de la sangría y la numeración sin necesidad de introducir alguna otra sintaxis de cierre de declaración.

La ramificación puede reducir el número de poscondiciones de caso de uso. Esto es porque los pasos dentro de una ramificación pueden o no ocurrir, dependiendo de las circunstancias. No pueden generar poscondiciones, que son hechos que “deben” ser verdaderos y no hechos que “pueden” ser verdaderos.

Repetición dentro de un flujo

Algunas veces tiene que repetir una acción varias veces dentro de un flujo de eventos. Esto no ocurre demasiado a menudo en el modelado de casos de uso, pero cuando lo hace, es de utilidad contar con una estrategia.

La especificación UML no especifica ninguna forma de mostrar repetición dentro de un flujo, por lo que incorporamos sencillas palabras clave Para y Mientras.

Puede modelar la repetición al utilizar la palabra clave Para. El formato es el siguiente:

n. Para (expresión de iteración)

n.1. Hacer algo.

n.2. Hacer algo más.

n.3. …

n+1.

Caso de uso: GestionarCarro
ID: 2
Breve descripción: El Cliente cambia la cantidad de un artículo en el carro.
Actores principales: Cliente
Actores secundarios: Ninguno
Precondiciones: 1. Los contenidos del carro son visibles.
Flujo principal: 1. El caso de uso empieza cuando el Cliente selecciona un artículo en el carro. 2. Si el Cliente selecciona "eliminar artículo" 2.1 El sistema elimina el artículo del carro. 3. Si el Cliente escribe una nueva cantidad 3.1 El sistema actualiza la cantidad del artículo en el carro.
Postcondiciones: Ninguna.
Flujos alternativos: Ninguno.

Figura 8. Especificación del caso de uso GestionarCarro

La expresión de iteración se evalúa como cierta un número positivo de iteraciones. Cada línea parte del cuerpo que sigue al para se repite el número de iteraciones especificado en la expresión de la iteración. En la Figura 9 se incluye un ejemplo.

Utilice la palabra clave Mientras para modelar una secuencia de acciones en el flujo de eventos que se realiza mientras cierta condición booleana es verdadera. El formato es el siguiente:

n. Mientras (condición booleana)

n. 1. Hacer algo.

n. 2. Hacer algo más.

n.3. …

n+1.

Igual que Para, la palabra clave Mientras se utiliza poco a menudo. Un ejemplo se muestra en la Figura 10. La secuencia de líneas detrás de la declaración Mientras se repite hasta que la condición booleana especificada en la cláusula Mientras se hace falsa.

Caso de uso: EncontrarProducto
Breve descripción: El sistema encuentra productos basándose en los criterios de búsqueda del Cliente y se los muestra.
Actores principales: Cliente.
Actores secundarios: Ninguno.
Precondiciones: Ninguna.
Flujo principal: 1. El caso de uso empieza cuando el Cliente selecciona "encontrar producto" 2. El sistema pregunta al Cliente los criterios de búsqueda. 3. El Cliente escribe los criterios solicitados. 4. El sistema busca productos que coincidan con criterios del Cliente. 5. Si el sistema encuentra algún producto que coincide entonces 5.1. Para cada producto encontrado 5.1.1. El sistema muestra una imagen del producto. 5.1.2. El sistema muestra un resumen de los detalles del producto. 5.1.3. El sistema muestra el precio del producto. 6. Sino 6.1. El sistema le dice al Cliente que no se han encontrado productos.
Postcondiciones: Ninguna.
Flujos alternativos: Ninguno.

Figura 9. Ejemplo de especificación empleando Para

Modelar flujos alternativos

Todo caso de uso tiene un flujo principal y puede tener muchos flujos alternativos. Estos son rutas de acceso alternativas a través del caso de uso que capturan errores, ramificaciones e interrupciones en el flujo principal. Como acaba de ver, la especificación del caso de uso contiene el flujo principal y una lista de los nombres de los flujos alternativos.

El punto clave sobre los flujos alternativos es que frecuentemente no regresan al flujo principal. Esto es porque los flujos alternativos a menudo tratan con errores y excepciones al flujo principal y tienden a tener diferentes poscondiciones. Puede ver los flujos alternativos ilustrados en la Figura 11.

Caso de uso: MostrarDetallesEmpresa
ID: 4
Breve descripción: El sistema muestra los detalles de la empresa al Cliente.
Actores principales: Cliente.
Actores secundarios: Ninguno.
Precondiciones: Ninguna.
Flujo principal: 1. El caso de uso empieza cuando Cliente selecciona "mostrar detalles empresa" 2. El sistema muestra una página Web con los detalles de la empresa. 3. Mientras el Cliente examina los detalles de la empresa 3.1. El sistema hace sonar música de fondo. 3.2. El sistema muestra ofertas especiales en un banner.
Postcondiciones: 1. El sistema ha mostrado los detalles de la empresa. 2. El sistema ha hecho sonar música de fondo. 3. El sistema ha mostrado ofertas especiales.
Flujos alternativos: Ninguno.

Figura 10. Ejemplo de especificación empleando Mientras

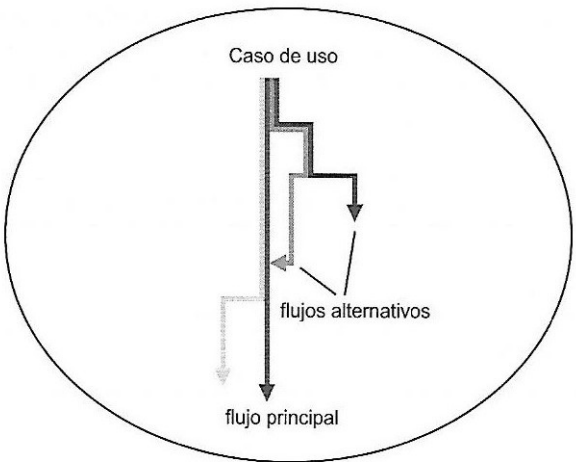

Figura 11. Flujos alternativos.

Puede documentar los flujos alternativos de forma separada o anexarlos al final del caso de uso. Nosotros preferimos documentarlos aparte. Como ejemplo de cómo puede modelar un caso de uso con flujos alternativos, considere la Figura 12.

Caso de uso: CrearNuevaCuentaCliente
ID: 5
Breve descripción: El sistema crea una nueva cuenta para el Cliente.
Actores principales: Cliente.
Actores secundarios: Ninguno.
Precondiciones: Ninguna.
Flujo principal: 1. El caso de uso empieza cuando el Cliente selecciona "crear nueva cuenta cliente". 2. Mientras que los detalles del Cliente no son válidos 2.1. El sistema pide al Cliente que escriba sus detalles de correo electrónico, contraseña, y contraseña de nuevo para confirmación. 2.2. El sistema valida los detalles del Cliente. 3. El sistema crea una nueva cuenta para el Cliente.
Postcondiciones: 1. Se ha creado una nueva cuenta para el Cliente.
Flujos alternativos: DirecciónCorreoInválida. ContraseñaInválida. Cancelar.

Figura 12. Especificación de caso de uso con flujos alternativos.

Puede ver que este caso de uso tiene definido tres flujos alternativos: DirecciónCorreoInválida, ContraseñaInválida, y Cancelar. En la Figura 13 se documenta el flujo alternativo DirecciónCorreoInválida.

Observe que podemos realizar varios cambios en la plantilla del caso de uso para acomodar los flujos alternativos.

- Nombre: Utilizamos la siguiente convención de nombre para flujos alternativos:
  Flujo alternativo: CrearNuevaCuentaC1iente:DirecciónCorreoInválida
- Esto indica que es un flujo alternativo denominado DirecciónCorreoInválida para el caso de uso CrearNuevaCuentaC1iente.
- ID: Observe cómo hemos utilizado un sistema de numeración jerárquico para relacionar el flujo alternativo con el caso de uso principal.
- Actores: Lista los actores utilizados por el flujo alternativo.

- Precondiciones y poscondiciones: Los flujos alternativos pueden tener su propio conjunto de precondiciones y poscondiciones que son diferentes de los del caso de uso. Si el flujo alternativo regresa al flujo principal, entonces sus poscondiciones se añaden eficazmente a las del flujo principal.
- Flujo alternativo: Los pasos en flujo alternativo.

Un flujo alternativo no debería tener flujos alternativos: De lo contrario, se hacen demasiado complejos muy rápidamente.

Flujo alternativo: CrearNuevaCuentaCliente: DirecciónCorreoInválida
ID: 5.1
Breve descripción: El sistema informa al Cliente que ha facilitado una dirección de correo inválida.
Actores principales: Cliente.
Actores secundarios: Ninguno.
Precondiciones: 1. El Cliente ha facilitado una dirección de correo inválida.
Flujo alternativo: 1. El flujo alternativo empieza después del paso 2.2 del flujo principal. 2. El sistema informa al Cliente que ha facilitado una dirección de correo inválida.
Postcondiciones: Ninguna.

Figura 13. Especificación de un flujo alternativo.

Los flujos alternativos se pueden activar de tres formas diferentes:

1. El flujo alternativo se puede activar en lugar del flujo principal.
2. El flujo alternativo se puede activar después de un paso en particular en el flujo principal.
3. El flujo alternativo se puede activar en cualquier momento durante el flujo principal.

Cuando un flujo alternativo se ejecuta en lugar del flujo principal, se activa por el actor principal y reemplaza a todo el caso de uso.

Cuando el flujo alternativo se activa después de un paso en particular en el flujo principal, debería empezar de la siguiente forma:

1. El flujo alternativo empieza detrás del paso X del flujo principal.

Esta es una forma de ramificación, pero es diferente de la ramificación que hemos tratado anteriormente porque es una desviación importante del flujo principal y podría no regresar a él.

Cuando un flujo alternativo se puede activar en cualquier momento durante el flujo principal, debería empezar de la siguiente forma:

1. El flujo alternativo empieza en cualquier momento.

Utilice este tipo de flujo alternativo para modelar algo que puede ocurrir en cualquier momento en el flujo principal antes del último paso. Por ejemplo, en el caso de uso CrearNuevaCuentaCliente, el Cliente puede elegir Cancelar la creación de la cuenta en cualquier momento. Puede documentar Cancelar como se muestra en la Figura 14.

Flujo alternativo: CrearNuevaCuentaCliente:Cancelar
ID: 5.2
Breve descripción: El Cliente cancela el proceso de creación de cuenta.
Actores principales: Cliente.
Actores secundarios: Ninguno.
Precondiciones: Ninguna.
Flujo alternativo: 1. El flujo alternativo empieza en cualquier momento. 2. El Cliente cancela la creación de cuenta.
Postcondiciones: 1. No se ha creado una nueva cuenta para el Cliente.

Figura 14. Especificación de un flujo alternativo que se puede iniciar en cualquier lugar del flujo principal.

Si quisiera que el flujo alternativo regresara al flujo principal, podría expresarlo de la siguiente manera:

N. El flujo alternativo regresa al paso M del flujo principal.

En este ejemplo, el flujo alternativo ejecuta su último paso N, y luego la ejecución del flujo principal continúa en el paso M.

Cómo encontrar flujos alternativos

Puede identificar flujos alternativos al inspeccionar el flujo principal. En cada caso del flujo principal busque:

- Alternativas posibles al flujo principal.
- Errores que puedan surgir en el flujo principal.
- Interrupciones que puedan ocurrir en un punto particular en el flujo principal.
- Interrupciones que puedan ocurrir en cualquier punto en el flujo principal.

Cada una de éstas es una fuente posible de un flujo alternativo.

Como hemos dicho, existe exactamente un flujo principal por caso de uso. Sin embargo, existen muchos flujos alternativos. La pregunta es ¿cuántos?. Debería tratar de limitar el número de flujos alternativos al mínimo necesario. Existen dos estrategias para esto.

- Seleccione los flujos alternativos más importantes y documente estos.
- Donde existen grupos de flujos alternativos que son todos muy similares, documente un miembro del grupo como ejemplo y (si fuera necesario) añada notas a esto explicando cómo difieren de éste.

Retomando la analogía del delta del río, además del canal principal, puede haber muchas ramificaciones y flujos alternativos en el delta. No puede permitirse mapear todos ellos, por lo que elige solamente los principales. Igualmente, muchas de estas ramificaciones fluyen en casi la misma dirección con sólo pequeñas diferencias. Por lo tanto, puede mapear un canal en detalle y proporcionar notas explicando cómo los otros canales más pequeños se desvían de éste.

El principio básico en el modelado de caso de uso es mantener la cantidad de información capturada en el mínimo necesario. Esto significa que muchos flujos alternativos nunca se pueden especificar del todo; una descripción de una línea añadida al caso de uso puede ser suficiente detalle para permitir entender el funcionamiento del sistema. Este es un punto importante. Es fácil verse inundado por flujos alternativos, y hemos visto más de una actividad de modelado de caso de uso fracasar por esto.

Recuerde que captura casos de uso para entender el comportamiento deseado del sistema y no por el hecho de crear un modelo completo de caso de uso. Por lo tanto, deja de modelar casos de uso cuando cree que ha conseguido ese entendimiento. También, puesto que el proceso unificado de desarrollo (RUP) es un ciclo de vida iterativo, siempre puede regresar a un caso de uso y realizar más trabajo si hay algún aspecto del comportamiento del sistema que decide que no entiende.

Generalización de actor

En el ejemplo en la Figura 15, puede ver muchas similitudes entre los dos actores, Cliente y AgenteVentas, en la forma que interactúan con el Sistema ventas (aquí, el AgenteVentas puede gestionar una venta en nombre del Cliente).

```
@startuml
rectangle Sistema_ventas {
   (ListarProductos)
   (ComprarProductos)
   (AceptarPago)
   (CalcularComisión)
}
:Cliente:
:AgenteVentas:
Cliente -- (ListarProductos)
Cliente -- (ComprarProductos)
Cliente -- (AceptarPago)
AgenteVentas -- (ListarProductos)
AgenteVentas -- (ComprarProductos)
AgenteVentas -- (AceptarPago)
AgenteVentas -- (CalcularComisión)
@enduml
```

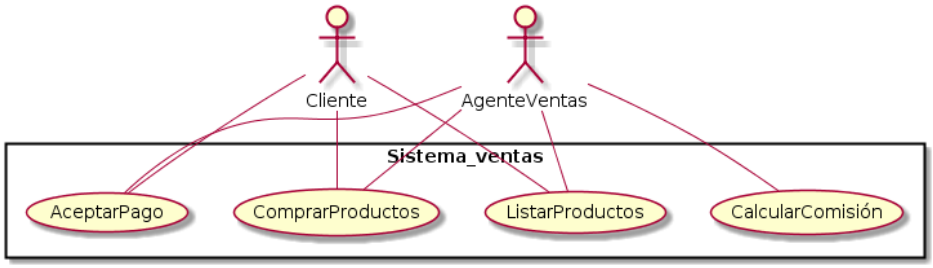

Figura 15.

Ambos actores activan los casos de uso Listar Productos, Comprar Productos y Aceptar Pago. De hecho, la única diferencia entre los dos actores es que AgenteVentas también activa el caso de uso CalcularComisión. Aparte del hecho de que esta semejanza en comportamiento proporciona muchas líneas cruzadas en el diagrama, parece indicar que existe cierto comportamiento común de actor que se podría resolver en un actor más generalizado.

Puede resolver este comportamiento común al utilizar la generalización de actor como se muestra en la Figura 16. Cree un actor abstracto denominado Comprador que interactúa con los casos de uso Listar Productos, Comprar Productos y Aceptar Pago.

```
@startuml
rectangle Sistema_ventas {
   (ListarProductos)
   (ComprarProductos)
   (AceptarPago)
   (CalcularComisión)
}
:Comprador:
:Cliente:  --|> :Comprador:
:AgenteVentas: --|> :Comprador:
Comprador -- (ListarProductos)
Comprador -- (ComprarProductos)
Comprador -- (AceptarPago)
AgenteVentas -- (CalcularComisión)
@enduml
```

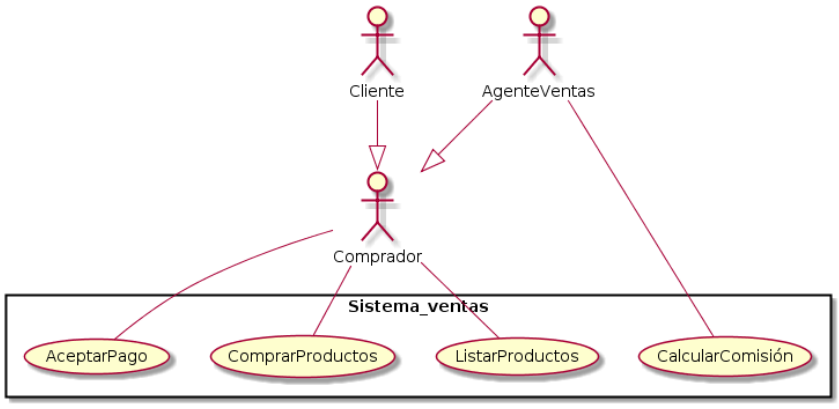

Figura 16. Generalización de actores.

Cliente y AgenteVentas se conocen como actores concretos porque personas reales (u otros sistemas) podrían desempeñar esos roles. Sin embargo, Comprador es un actor abstracto ya que es una abstracción introducida simplemente para capturar el comportamiento común de los dos actores concretos, que ambos pueden actuar en un rol de compra.

Cliente y AgenteVentas heredan todos los roles y relaciones de los casos de uso de su padre abstracto. Por lo tanto, interpretando la Figura 16, Cliente y AgenteVentas tienen interacciones con los casos de uso Listar Productos, Comprar Productos y Aceptar Pago que heredan de su padre, Comprador. Además, AgenteVentas tiene una interacción con el caso de uso CalcularComisión que no se hereda, es específica del actor AgenteVentas. Puede ver que un uso juicioso de actores abstractos puede simplificar los diagramas de caso de uso. También simplifica la semántica de su modelo de caso de uso porque puede tratar elementos diferentes de la misma forma.

Merece la pena destacar que el actor padre en la generalización de actor no siempre tiene que ser abstracto, puede ser un rol concreto que una persona o sistema puede desempeñar. Sin embargo, un buen estilo dicta que los actores padre son normalmente abstractos para mantener la semántica de generalización sencilla.

Lo que hemos visto es que si dos actores se comunican con el mismo conjunto de casos de uso de la misma forma, podemos expresarlo como una generalización a otro actor (posiblemente abstracto). Los actores descendentes heredan los roles y relaciones a casos de uso albergados por el actor padre. Puede sustituir un actor descendente en cualquier lugar que se espere el ascendiente. Este es el principio de sustitución que es una prueba importante para un uso correcto de generalización con cualquier clasificador.

En este ejemplo, es razonable que pueda sustituir un AgenteVentas o Cliente en cualquier lugar que se espere un Comprador (interactuando por ejemplo con los casos de uso Listar Productos, Comprar Productos y Aceptar Pago), por lo que la generalización de actor es la estrategia correcta.

Generalización de caso de uso

La generalización de caso de uso se utiliza cuando tiene uno o más casos de uso que son realmente especificaciones concretas de un caso más general. Igual que la generalización de actor, solamente debería utilizarlo cuando simplifique su modelo de caso de uso. En la generalización de caso de uso, los casos de uso hijo representan formas más específicas del padre. Los hijos pueden:

- Heredar características de su caso de uso padre.
- Añadir nuevas características.
- Anular (cambiar) características heredadas.

El caso de uso hijo hereda automáticamente todas las características de su padre. Sin embargo, no todos los tipos de característica de caso de uso se pueden anular. Las restricciones se resumen en la Tabla 2.

Por lo tanto, ¿cómo se documenta la generalización de caso de uso en especificaciones de caso de uso?

La especificación UML permanece en silencio en este punto pero existen varias técnicas estándar. Preferimos utilizar un lenguaje sencillo de etiquetas para resaltar las cinco posibilidades en un caso de uso hijo. Existen dos reglas para aplicar la técnica:

- Todo número del paso en el hijo está postfijado por el número de paso equivalente en el padre, si hay uno. Por ejemplo, I . (2 .) . Algún paso.
- Si el paso en el hijo anula un caso padre, se postfija por "o" y luego el número del paso en el padre. Por ejemplo, 6 . (06 .) Otro paso.

Tabla 2.

Característica de caso de uso	Heredar	Añadir	Anular
Relación	Si	Si	No
Punto de extensión	Si	Si	No
Precondición	Si	Si	No
Poscondición	Si	Si	Si
Paso en flujo principal	Si	Si	Si
Flujo alternativo	Si	Si	Si

La Tabla 3 resume la sintaxis para las cinco opciones en casos de uso hijo.

Tabla 3.

La característica se …	Ejemplo de etiqueta
Hereda sin cambio	3.(3.) El cliente incorpora los criterios requeridos.
Hereda y renumera.	6 . 2 (6 . 1) El sistema dice al Cliente que no se pudieron encontrar productos coincidentes.
Hereda y anula.	1 . (01 .) El Cliente selecciona "buscar libro".
Hereda, anula y renumera.	5 . 2 (05 . 1) El sistema muestra una página que indica los detalles de un máximo de cinco libros.
Añade.	6 . 3 El sistema vuelve a mostrar la página de búsqueda "buscar libro".

La Figura 17 muestra un extracto del diagrama de caso de uso de un sistema de ventas. Tenemos el caso de uso padre BuscarProducto y luego dos especializaciones de esto, BuscarLibro y BuscarCD.

El caso de uso hijo BuscarLibro es mucho más concreto que el caso de uso BuscarProducto. Especializa el padre más abstracto para tratar con un tipo específico de producto, libros.

Si el caso de uso padre no tiene flujo de eventos o un flujo de eventos que está incompleto, es un caso de uso abstracto. Los casos de uso abstractos son bastante comunes porque puede utilizarlos para capturar comportamiento en los niveles más altos de abstracción. Puesto que los casos de uso abstracto tienen un flujo incompleto de eventos, pueden no ser ejecutados por el sistema. En lugar de un flujo de eventos, los casos de uso abstractos pueden tener un resumen de texto plano del comportamiento de alto nivel que sus hijos esperarán implementar. Puede situar esto en la breve descripción del caso de uso.

```
@startuml
rectangle Sistema_ventas {
   (BuscarProducto)
   (BuscarLibro) --|> (BuscarProducto)
   (BuscarCD)  --|> (BuscarProducto)
}
:Cliente:
Cliente -- (BuscarProducto)
@enduml
```

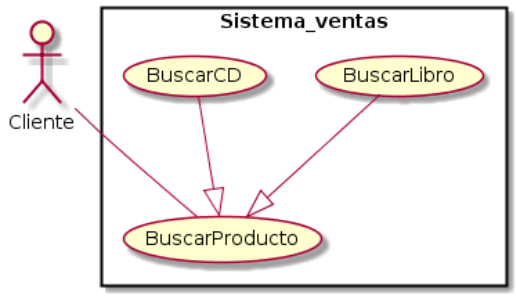

Figura 17. Generalización de casos de uso

Es difícil mostrar características heredadas en casos de uso hijo. Tiene que utilizar algún tipo de lenguaje de etiqueta o convención tipográfica que los grupos de decisión normalmente encuentran confuso. Puesto que los casos de uso son herramientas de comunicación con los grupos de decisión, ésta es una desventaja seria. Otra desventaja es que tiene que mantener manualmente coherencia entre los padres e hijos cuando uno de ellos cambia. Esta es una tarea laboriosa y propensa a errores.

Un enfoque a este problema es restringir el caso de uso padre para que no tenga ningún flujo principal, sino solamente una breve descripción de su semántica. En ese caso, no tiene que preocuparse por la herencia o borrado. Este enfoque hace que la generalización de caso de uso sea sencilla y es una forma muy eficaz de mostrar que uno o más casos de uso son simplemente variantes específicas de un caso de uso más general. El caso de uso más general le permite pensar en el sistema en la forma abstracta y puede incluir oportunidades para perfeccionar el sistema de software.

Relación <<include>>

Escribir casos de uso puede ser muy repetitivo algunas veces. Suponga que está escribiendo un sistema de personal (véase la Figura 18). Casi cualquier cosa que le pida al sistema que haga implicará primero localizar los detalles de un empleado específico. Si tuviera que escribir esta secuencia de eventos cada vez que necesitara los detalles del empleado, sus casos de uso se harían bastante repetitivos. La relación <<include>> entre los casos de uso le permite incluir el comportamiento de un caso de uso en el flujo de otro caso de uso.

Denominamos al caso de uso que incluye como el caso de uso base, y al caso de uso incluido como caso de uso de inclusión. El caso de uso de inclusión proporciona el comportamiento a su caso de uso base.

Debe especificar el punto exacto en el caso de uso base cuando necesita que se incluya el comportamiento del caso de uso de inclusión. La sintaxis para <<include>> es como una

llamada de función, y de hecho tiene semántica similar. La semántica de <<include>> es sencilla (véase la Figura 19). El caso de uso base se ejecuta hasta que se alcanza el punto de inclusión, luego la ejecución se pasa al caso de uso de inclusión. Cuando termina el caso de uso de inclusión, el control regresa de nuevo al caso de uso base.

```
@startuml
rectangle Sistema_de_personal {
   (CambiarDetallesEmpleado) ..> (BuscarDetallesEmpleado) :<<include>>
   (VerDetallesEmpleado) ..> (BuscarDetallesEmpleado) :<<include>>
   (EliminarDetallesEmpleado) ..> (BuscarDetallesEmpleado) :<<include>>
   (BuscarDetallesEmpleado)
}
:Director:
Director -- (CambiarDetallesEmpleado)
Director -- (VerDetallesEmpleado)
Director -- (EliminarDetallesEmpleado)
@enduml
```

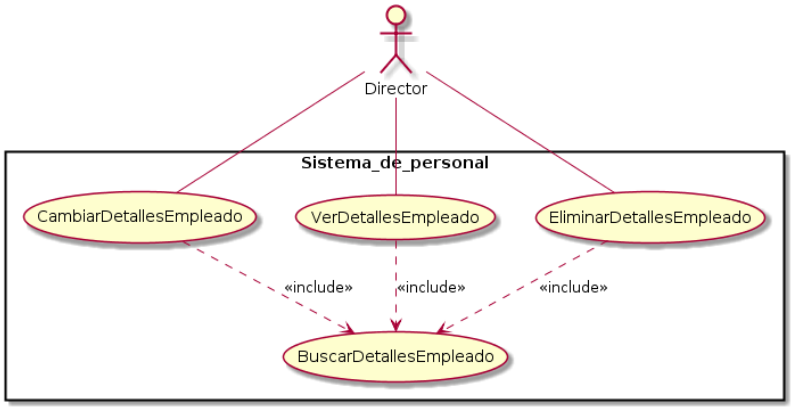

Figura 18. Relación include

El caso de uso no está completo sin todos sus casos de uso de inclusión. Los casos de uso de inclusión forman partes integrales del caso de uso base. Sin embargo, los casos de uso de inclusión pueden o no estar completos. Si un caso de uso de inclusión no está completo, simplemente contiene un flujo parcial de eventos que solamente tendrán sentido cuando se incluya en una base apropiada.

A menudo hacemos referencia a esto como un fragmento de comportamiento. En este caso, decimos que el caso de uso de inclusión no se puede instanciar, es decir, no se puede activar directamente por los actores, solamente se puede ejecutar cuando esté incluido en una base apropiada. Sin embargo, si los casos de uso de inclusión están completos, actúan como casos de uso normales y se pueden instanciar. Es bastante razonable activarlos por actores. Puede ver el caso de uso de inclusión, BuscarDetallesEmpleado, en la Figura 19. Está incompleto y no se puede instanciar.

Caso de uso: CambiarDetallesEmpleado
ID: 1
Breve descripción: El Director cambia detalles del empleado.
Actores principales: Director
Actores secundarios: Ninguno.
Precondiciones: 1 . El Director se conecta al sistema.
Flujo principal: 1. include(BuscarDetallesEmpleado). 2. El sistema muestra los detalles del empleado. 3. El Director cambia los detalles del empleado.
Postcondiciones: 1. Se han cambiado los detalles del empleado.
Flujos alternativos: Ninguno.

Caso de uso: VerDetallesEmpleado
ID: 2
Breve descripción: El Director ve los detalles del empleado.
Actores principales: Director
Actores secundarios: Ninguno.
Precondiciones: 1. El Director se conecta al sistema,
Flujo principal: 1. Include(BuscarDetallesEmpleado). 2. El sistema muestra los detalles del empleado
Postcondiciones: 1. El sistema ha mostrado los detalles del empleado.
Flujos alternativos: Ninguno.

Caso de uso: EliminarDetallesEmpleado
ID: 3
Breve descripción: El Director elimina los detalles del empleado.
Actores principales: Director
Actores secundarios: Ninguno.
Precondiciones: 1. El Director se conecta al sistema.
Flujo principal: 1. Include(BuscarDetallesEmpleado). 2. El sistema muestra los detalles del empleado. 3. El Director elimina los detalles del empleado.
Postcondiciones: 1. Los detalles del empleado se han eliminado.
Flujos alternativos: Ninguno.

Caso de uso: BuscarDetallesEmpleado
ID: 4
Breve descripción: El Director busca los detalles del empleado.
Actores principales: Director.
Actores secundarios: Ninguno.
Precondiciones: 1. El Director se conecta al sistema.
Flujo principal: 1. El Director escribe el ID del empleado. 2. El sistema encuentra los detalles del empleado.
Postcondiciones: 1. El sistema ha encontrado los detalles del empleado.
Flujos alternativos: Ninguno.

Figura 19. Especificación de una inclusión de caso de uso.

Relación <<extend>>

<<extend>> proporciona un medio para insertar un nuevo comportamiento en un caso de uso existente (véase la Figura 20). El caso de uso base proporciona un conjunto de puntos de

extensión que son enganches donde se puede añadir nuevo comportamiento, el caso de uso de extensión proporciona un conjunto de segmentos de inserción que se pueden insertar en el caso de uso base en esos enganches. Como verá en breve, la relación <<extend>> se puede utilizar para especificar exactamente qué puntos de extensión en el caso de uso base se están extendiendo.

```
@startuml
rectangle Sistema_de_biblioteca {
   (DevolverLibro) <.. (PonerMulta) :<<extend>>
   (PrestarLibro)
   (BuscarLibro)
   (PonerMulta)
}
:Bibliotecario:
Bibliotecario -- (DevolverLibro)
Bibliotecario -- (PrestarLibro)
Bibliotecario -- (BuscarLibro)
@enduml
```

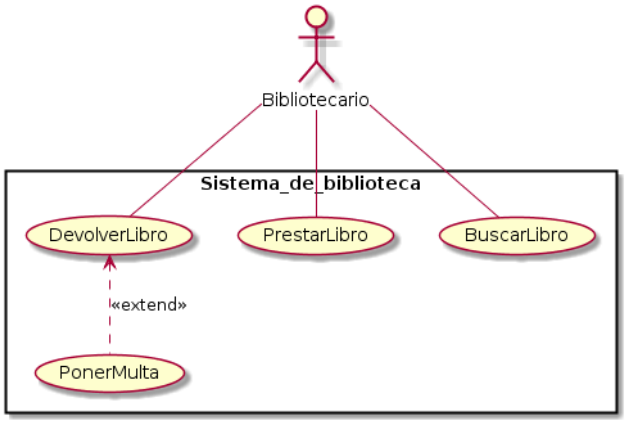

Figura 20. Caso de uso extendido.

Lo que es interesante de <<extend>> es que el caso uso base no sabe nada de los casos de uso de extensión; simplemente proporciona enganches para estos. De hecho, el caso de uso base está perfectamente completo sin sus extensiones. Esto es muy diferente de <<include>>, donde los casos de uso base estaban incompletos sin sus casos de uso de inclusión. Además, los puntos de extensión no están realmente insertados en el flujo de eventos del caso de uso base, sino que están añadidos a una capa en la parte superior del flujo de eventos.

Los puntos de extensión están indicados en el flujo de eventos del caso de uso base como se muestra en la Figura 21. También puede mostrar puntos de extensión en el diagrama de caso de uso al listarlos en un nuevo compartimiento en el icono de caso de uso base (esta funcionalidad no es soportada por Planttext / PlantUML).

Los puntos de extensión en el flujo principal no están numerados. En su lugar, aparecen entre los pasos numerados del flujo. De hecho, UML indica explícitamente que los puntos de extensión existen solamente en una capa sobre el flujo principal. Por lo tanto no son parte del flujo principal. Puede pensar en esta capa como un acetato sobre el flujo principal donde se graban los puntos de extensión. Lo interesante de esta idea de una capa es hacer que el caso

de uso base fluya completamente independiente de los puntos de extensión. Es decir, el flujo de caso de uso base no sabe dónde se extiende. Esto le permite utilizar <<extend>> para realizar extensiones arbitrarias a un flujo de caso de uso base. Cuando utiliza <<extend>>, el caso de uso base actúa como un marco de trabajo modular en el que conectar extensiones en puntos de extensión predefinidos. En el ejemplo en la Figura 21, puede ver que el caso de uso base DevolverLibro tiene un punto de extensión denominado libroRetrasado entre los pasos 3 y 4 en su flujo de eventos. Puede ver que <<extend>> proporciona una buena forma de tratar con casos excepcionales o casos en los que necesita un marco de trabajo flexible porque no puede predecir (o simplemente no conoce) todas las posibles extensiones por adelantado.

Caso de uso: DevolverLibro
ID: 1
Breve descripción: El Bibliotecario devuelve un libro prestado.
Actores principales: Bibliotecario
Actores secundarios: Ninguno.
Precondiciones: 1 . El Bibliotecario se conecta al sistema.
Flujo principal: 1 . El Bibliotecario escribe el número del prestatario. 2. El sistema muestra los detalles del prestatario incluida la lista de libros prestados. 3. El Biblioteca busca el libro a devolver en la lista de libros. punto de extensión: libroRetrasado 4. El Bibliotecario devuelve el libro.
Postcondiciones: 1. El libro se ha devuelto.
Flujos alternativos: Ninguno.

Figura 21: Especificación de punto de extensión en caso de uso.

El caso de uso de extensión

Los casos de uso de extensión son por lo general casos de uso no completos y por lo tanto no se pueden instanciar. Simplemente constan de uno o más fragmentos de comportamiento conocidos como segmentos de inserción (en el curso trataremos con un único segmento de inserción). La relación <<extend>> especifica el punto de extensión en el caso de uso base donde se insertará el segmento de inserción. Se aplican las siguientes reglas:

- La relación <<extend>> debe especificar uno o más puntos de extensión en el caso de uso base o se asume que la relación <<extend>> hace referencia a todos los puntos de extensión.
- El caso de uso de extensión debe tener el mismo número de segmentos de inserción ya que existen puntos de extensión especificados en la relación <<extend>>.

- Es legal para dos casos de uso de extensión <<extend>> el mismo caso de uso base en el mismo punto de extensión. Pero si sucede esto, el orden en el que se ejecutan las extensiones es indeterminado.

En el ejemplo en la Figura 22, existe un solo segmento de inserción en el caso de uso de extensión PonerMulta. El caso de uso de extensión puede tener precondiciones o poscondiciones. Las precondiciones deben estar completas; de lo contrario, el segmento no se ejecuta. Las poscondiciones restringen el estado del sistema después de que el segmento se ha ejecutado.

Los casos de uso de extensión pueden tener casos de uso de extensión o casos de uso de inclusión. Sin embargo, tendemos a evitar esto, ya que puede hacer que el modelo de caso de uso sea demasiado complejo.

Caso de uso de extensión: PonerMulta
ID: 10
Breve descripción: Segmento 1: El Bibliotecario graba e imprime una multa.
Actores principales: Bibliotecario.
Actores secundarios: Ninguno.
Segmento 1 precondiciones: 1. El libro devuelto está fuera de plazo.
Segmento 1 flujo: I. El Bibliotecario escribe los detalles de la multa en el sistema. 2. El sistema imprime la multa.
Segmento 1 postcondiciones: 1. La multa se ha grabado en el sistema. 2. El sistema ha impreso la multa.

Figura 22. Caso de uso extensión.

Extensiones condicionales

El ejemplo en la Figura 23 muestra un sistema de biblioteca algo más bueno en el que a los prestatarios se les da un aviso la primera vez que el libro se devuelve pasado el plazo y se les penaliza solamente después. Podemos modelar esto al añadir un nuevo caso de uso de extensión, EmitirAviso, y luego situar condiciones en las relaciones <<extend>>. Las condiciones sobre expresiones booleanas, y la inserción se realiza si, y sólo si, la expresión evalúa como verdadero.

Utilice relaciones de generalización, inclusión, y extensión cuando simplifiquen su modelo de caso de uso. Hemos encontrado que los mejores modelos de caso de uso son sencillos. Recuerde que el modelo de caso de uso es una declaración de requisitos y, como tal, debe estar accesible para los grupos de decisión así como para los modeladores.

Basándose en nuestra experiencia del modelado de caso de uso, en muchas empresas diferentes hemos encontrado que (Arlow y Neustadt, 2006):

- Por lo general, los grupos de decisión pueden entender fácilmente los actores y casos de uso con poca formación.
- Los grupos de decisión encuentran la generalización de actor más difícil de entender.
- Un gran uso de <<include>> puede hacer que los modelos de caso de uso sean más difíciles de entender; los grupos de decisión y los modeladores tienen que ver más de un caso de uso para hacerse una idea completa.
- Los grupos de decisión tienen mucha dificultad con <<extend>>; esto puede ser cierto incluso con explicaciones muy detalladas.
- Un sorprendente número de modeladores de objeto no entiende bien la semántica de <<extend>>.
- La generalización de caso de uso se debería evitar a menos que se utilicen casos de uso padre abstractos (en lugar de concretos), de lo contrario añade demasiada complejidad a los casos de uso hijos.

```
@startuml
rectangle Sistema_de_biblioteca {
   (DevolverLibro)
   note left
   puntos de extensión:
   libroRetrasado
end note
   (EmitirAviso)
   (PonerMulta)
   (DevolverLibro)<.. (PonerMulta) :<<extend>> \n condition: {primeraFalta} \n punto de extensión: libroRetrasado
   (DevolverLibro) <.. (EmitirAviso) :<<extend>> \n condition: {not primeraFalta} \n punto de extensión: libroRetrasado
}
:Bibliotecario:
Bibliotecario -- (DevolverLibro)
@enduml
```

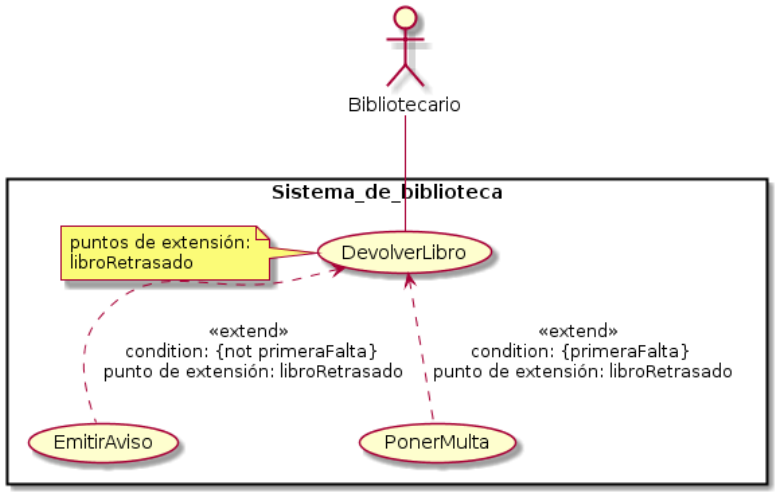

Figura 23. Caso de uso con extensiones condicionales

Sugerencias para escribir casos de uso

- Mantener los casos de uso breves y sencillos.

Nuestro eslogan para los casos de uso es "mantenlos breves, mantenlos sencillos". Incluye solamente el detalle suficiente para capturar los requisitos. Desafortunadamente, algunos proyectos tienen miedo de los que es breve y sencillo y están enamorados de grandes cantidades de documentación.

Una buena regla general a seguir es asegurarse de que el flujo principal de un caso de uso cabe en una sola hoja de papel. Más de éstas y el caso de uso es demasiado largo. De hecho, encontrará que muchos casos de uso ocupan menos de media página.

Empiece por simplificar el texto (recuerde utilizar frases declarativas cortas). Elimine cualquier detalle de diseño. Si sigue siendo demasiado largo, vuelva a analizar el problema. ¿Es posible que exista más de un caso de uso? ¿Puede resolver flujos alternativos?

- Centrarse en el qué, no en el cómo

Recuerde que está escribiendo casos de uso para solucionar lo que los actores necesitan que haga el sistema, no cómo el sistema debería hacerlo. El cómo viene después en el workflow de diseño. Confundir el qué con el cómo es un problema que vemos continuamente. El escritor del caso de uso inventa alguna solución cuando escribe el caso de uso. Por ejemplo, considere el fragmento de caso de uso que se proporciona a continuación.

4. El sistema pide al Cliente que confirme el pedido.

5. El Cliente pulsa el botón Aceptar.

En este paso, el escritor del caso de uso ha imaginado algún tipo de interfaz de usuario: un formulario con un botón Aceptar. Debido a esto, el caso de uso ha dejado de ser una declaración de requisitos, es un diseño primitivo. Una mejor forma de expresar el paso 5 es la siguiente:

5. El Cliente acepta el pedido.

Mantenga los detalles del diseño (que todavía no conoce) fuera del caso de uso.

- Evite descomposición funcional

Un error común en el análisis de caso de uso es crear un conjunto de casos de uso de "alto nivel" y luego desglosarlos en un conjunto de casos de uso de bajo nivel, y así sucesivamente hasta que acabe con casos de uso "primitivos" que están suficientemente detallados para implantarse. Este enfoque al diseño de software se conoce como descomposición funcional y es erróneo cuando se aplica al modelado de casos de uso.

Examinemos un ejemplo. En la Figura 24, el analista ha definido la operación de un sistema de biblioteca utilizando un sencillo caso de uso de alto nivel, GestionarBiblioteca, y luego lo ha descompuesto en casos de uso cada vez con más niveles de detalle, creando una descomposición funcional.

Muchos analistas no orientados a objetos encontrarán la Figura 24 bastante posible. Sin embargo, como modelo de caso de uso hay muchos errores en ella:

- En lugar de centrarse en capturar los requisitos, el modelo se centra en estructurar esos requisitos de forma artificial; existen muchas descomposiciones posibles.
- El modelo describe el sistema como un conjunto de funciones anidadas. Sin embargo, los sistemas orientados a objetos son conjuntos de objetos que cooperan enviando mensajes de un lado a otro. Aquí hay un mal emparejamiento conceptual.

- Solamente el nivel más bajo de casos de uso, AñadirLibro, EliminarLibro, AñadirPetición, El iminarPetición, PrestarLibro y DevolverLibro, tienen especificaciones interesantes. Los niveles más altos invocan a los más bajos y no añaden nada al modelo en términos de capturar requisitos.
- El modelo es difícil de entender para los grupos de decisión. Existen varios casos de uso abstractos (Gestionar Biblioteca, MantenerLibros, MantenerPeticiones y MantenerPréstamos) y numerosas relaciones <<include>> a los niveles más bajos.

El uso de un enfoque de descomposición funcional sugiere que un analista ha pensado en el sistema de forma errónea. A menudo es una indicación de que el analista está formado en técnicas de programación de procedimientos más tradicionales y no ha entendido el paradigma orientado a objetos. En este caso, siempre es una buena idea emplear a un modelador de casos de uso con experiencia para proporcionar consejos y revisiones.

```
@startuml
rectangle Sistema_de_biblioteca {
  (GestionarBiblioteca) ..> (MantenerLibros) :<<include>>
  (MantenerLibros) ..> (AñadirLibro) :<<include>>
  (MantenerLibros) ..> (EliminarLibro) :<<include>>
  (GestionarBiblioteca) ..> (MantenerPeticiones) :<<include>>
  (MantenerPeticiones) ..> (AñadirPetición) :<<include>>
  (MantenerPeticiones) ..> (EliminarPetición) :<<include>>
  (GestionarBiblioteca) ..> (MantenerPréstamos) :<<include>>
  (MantenerPréstamos) ..> (PrestarLibro) :<<include>>
  (MantenerPréstamos) ..> (DevolverLibro) :<<include>>
}
:Bibliotecario:
Bibliotecario -- (GestionarBiblioteca)
@enduml
```

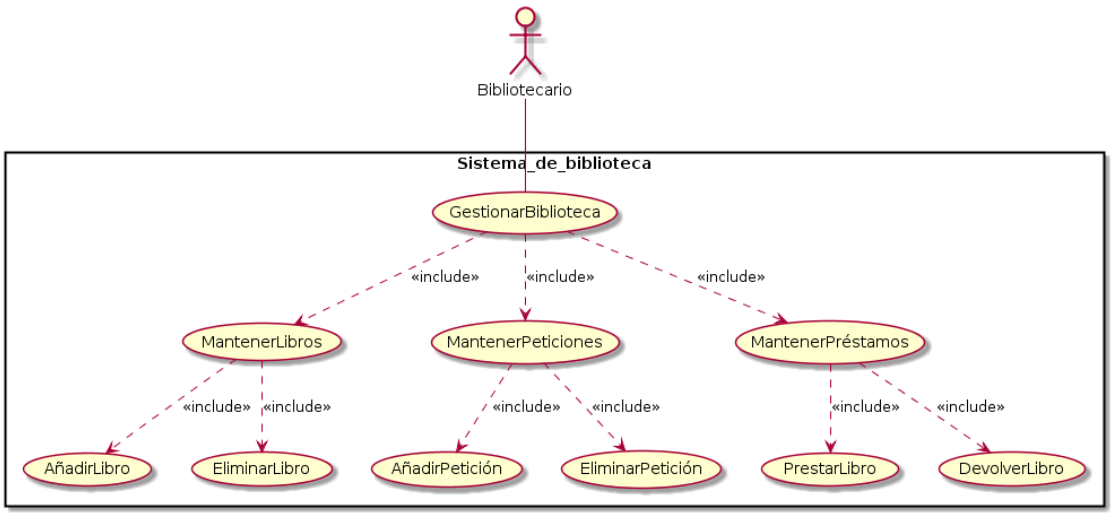

Figura 24. Ejemplo de lo que **NO** se debe hacer, descomposición funcional

No todos los ejemplos de descomposición funcional son tan obvios como el ejemplo en la Figura 24. A veces encontrará que partes del modelo de caso de uso son adecuadas, mientras otras partes se han descompuesto. Es una buena idea comprobar cualquier parte del modelo del caso de uso que tenga una jerarquía profunda para posible descomposición funcional.

Por último, deberíamos destacar que las jerarquías surgen de forma natural en el modelado de casos de uso. Sin embargo, estas jerarquías naturales no tienen nunca más de un nivel (o dos como mucho) y el modelo completo nunca parte de un caso de uso único.

Seguimiento de requisitos

Con un modelo de requisitos y un modelo de caso de uso, tiene dos "bases de datos" de requisitos funcionales. Es importante relacionar las dos para averiguar SI hay algo en su modelo de requisitos que no está tratado por el modelo de caso de uso y viceversa. Este es un aspecto del seguimiento de requisitos.

Seguir requisitos funcionales de casos de uso es complicado por el hecho de que existe una relación muchos a muchos entre requisitos funcionales individuales y casos de uso. Un caso de uso tratará muchos requisitos funcionales individuales y un requisito funcional puede estar presente en varios casos de uso diferentes.

Afortunadamente, tendrá soporte de herramientas de modelado para el seguimiento de requisitos y, de hecho, herramientas de ingeniería de requisitos como RequisitePro le permiten vincular requisitos individuales en su base de datos de requisitos con casos de uso específicos y viceversa. De hecho, UML proporciona un soporte bastante bueno para el seguimiento de requisitos. Utilizando valores etiquetados, puede asociar una lista de números ID de requisito con cada caso de uso. En la herramienta de requisitos, puede vincular uno o más identificadores de caso de uso con requisitos específicos.

Si no tiene ningún soporte de herramienta de modelado, tiene que hacer el trabajo manualmente. Un buen enfoque es crear una matriz de trazabilidad de requisitos. Esto es una sencilla cuadrícula con los números ID de los requisitos individuales hacia abajo en un eje y los nombres de caso de uso (y/o números ID) a lo largo del otro. Una cruz se sitúa en todas las celdas donde se cruzan un caso de uso y requisito. Las matrices de trazabilidad de requisitos se crean a menudo en hojas de cálculo. Un ejemplo se proporciona en la Tabla 4.

Tabla 4. Seguimiento de requisitos

		Caso de uso			
		CU1	CU2	CU3	CU4
Requisito	R1	x			
	R2	x	x		
	R3			x	
	R4				x
	R5	x			

Una matriz de trazabilidad de requisito es una herramienta de utilidad para comprobar coherencias. Si existe un requisito que no mapea con ningún caso de uso, entonces falta un caso de uso. Y al contrario, si hay un caso de uso que no mapea con ningún requisito, sabe que su conjunto de requisitos está incompleto.

¿Cuándo aplicar modelado de caso de uso?

Los casos de uso capturan requisitos funcionales y por lo tanto no son eficaces para sistemas dominados por requisitos no funcionales.

Los casos de uso son la mejor opción para la captura de requisitos cuando:

- El sistema está dominado por requisitos funcionales.
- El sistema tiene muchos tipos de usuarios para el que presenta diferente funcionalidad (existen muchos actores).
- El sistema tiene muchas interfaces (existen muchos actores).

Los casos de uso no serán una buena opción cuando:

- El sistema está dominado por requisitos no funcionales.
- El sistema tiene pocos usuarios.
- El sistema tiene pocas interfaces.

Ejemplos de sistemas donde los casos de uso pueden no ser apropiados son sistemas embebidos y sistemas que son algorítmicamente complejos pero con pocas interfaces. Para estos sistemas, sería mucho mejor que optara por técnicas de ingeniería de requisitos más convencionales. Es una cuestión de elegir la herramienta correcta para el trabajo entre manos.

Bibliografía

Arlow, J., I. Neustadt. UML 2.0. Anaya, 2006.