***Facultad de Ciencia y Tecnología*** 

***Cátedra: Sistemas de Información II***

***Guía: Casos de Uso***

**INTRODUCCION**

***¿Qué es UML?***

Son las siglas de Lenguaje de Modelado Unificado (Unified Modeling Language). UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de sistemas con una componente software significativa. Como ejemplos de artefactos de un sistema tenemos el código fuente, el diseño, los requisitos, la arquitectura y los prototipos, entre otros.

UML es un lenguaje de modelado orientado a objetos. Se debe recalcar que UML no es una metodología, aunque proporciona técnicas que pueden ser usadas en conjunto o parcialmente en metodologías, fundamentalmente aquellas destinadas al desarrollo orientados a objetos, como el Proceso Unificado de Rational o Métrica V.3 (aunque Métrica V.3 es una metodología mixta que también permite el desarrollo estructurado).

Al ser UML un lenguaje, está compuesto de una sintaxis (reglas que indican cómo ensamblar los componentes de su vocabulario para crear expresiones) y una semántica (reglas que indican el significado de las expresiones.

Existe 9 tipos de diagramas, algunos son estáticos y otros dinámicos

* Diagrama de casos de uso
* Diagrama de clases
* Diagrama de objetos
* Diagrama de estados
* Diagrama de secuencias
* Diagrama de actividades
* Diagrama de colaboraciones
* Diagrama de componentes
* Diagrama de distribución

***¿Qué son los Casos de Uso?***

Los casos de uso son una técnica para especificar el comportamiento de un sistema:

*“Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios.”*

Todo sistema de software ofrece a su entorno –aquellos que lo usan– una serie de servicios. Un caso de uso es una forma de expresar cómo alguien o algo externo a un sistema lo usa. Cuando decimos “alguien o algo” hacemos referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware y software.

Por ejemplo, un sistema de ventas, si pretende tener éxito, debe ofrecer un servicio para ingresar un nuevo pedido de un cliente. Cuando un usuario accede a este servicio, podemos decir que está “ejecutando” el caso de uso *ingresando pedido*.

***Los Casos de Uso y UML***

UML (Lenguaje Unificado de Modelado) va en camino de convertirse en un estándar para modelado de sistemas de software de amplia difusión. A pesar de ser considerada una técnica de Análisis Orientado a Objetos, es importante destacar que los casos de uso poco tienen que ver con entender a un sistema como un conjunto de objetos que interactúan, que es la premisa básica del análisis orientado a objetos “*clásico*”. En este sentido, el éxito de los casos de uso no hace más que dar la razón al análisis estructurado, que propone que la mejor forma de empezar a entender un sistema es a partir de los servicios o funciones que ofrece a su entorno, independientemente de los objetos que interactúan dentro del sistema para proveerlos.

De lo dicho anteriormente podemos concluir que los casos de uso son independientes del método de diseño que se utilice, y por lo tanto del método de programación. Luego de documentar los requerimientos de un sistema con casos de uso, se puede diseñar un sistema “estructurado” (manteniendo una separación entre datos y funciones), o un sistema Orientado a Objetos, sin que la técnica sea de mayor o menor utilidad en alguno de los dos casos. Esto da más flexibilidad al método, y probablemente contribuya a su éxito.

**Definiciones Básicas**

***Actores***

Un actor es una agrupación uniforme de personas, sistemas o máquinas que interactúan con el sistema que estamos construyendo de la misma forma. Por ejemplo, para una empresa que recibe pedidos en forma telefónica, todos los operadores que reciban pedidos y los ingresen en un sistema de ventas, si pueden hacer las mismas cosas con el sistema, son considerados un único actor: *Empleado de Ventas*.

Los actores son externos al sistema que vamos a desarrollar. Por lo tanto, al identificar actores estamos empezando a delimitar el sistema, y a definir su alcance. Definir el alcance del sistema debe ser el primer objetivo de todo analista, ya que un proyecto sin alcance definido nunca podrá alcanzar sus objetivos.

Es importante tener clara la diferencia entre usuario y actor. Un actor es una clase de rol, mientras que un usuario es una persona que, cuando usa el sistema, asume un rol. De esta forma, un usuario puede acceder al sistema como distintos actores. La forma más simple de entender esto es pensar en perfiles de usuario de un sistema operativo. Una misma persona puede acceder al sistema con distintos perfiles, que le permiten hacer cosas distintas. Los perfiles son en este caso equivalentes a los actores.

Otro sistema que interactúa con el que estamos construyendo también es un actor. Por ejemplo, si nuestro sistema deberá generar asientos contables para ser procesados por el sistema de contabilidad, este último sistema será un actor, que usa los servicios de nuestro sistema.

También puede ocurrir que el actor sea una máquina, en el caso en que el software controle sus movimientos, o sea operado por una máquina. Por ejemplo, si estamos construyendo un sistema para mover el brazo de un robot, el hardware del robot será un actor, asumiendo que dentro de nuestro sistema están las rutinas de bajo nivel que controlan al hardware.

Los actores se representan con dibujos simplificados de personas, llamados en inglés “stick man” (hombres de palo).

Si bien en UML los actores siempre se representan con “hombres de palo”, a veces resulta útil representar a otros sistemas con alguna representación más clara.

UML, pueden usarse para indicar el flujo de información entre el sistema y el actor. Si la flecha apunta desde el actor hacia el sistema, esto indica que el actor está ingresando información en el sistema. Si la flecha apunta desde el sistema hacia el actor, el sistema está generando información para el actor.

Identificar a los actores es el primer paso para usar la técnica de casos de uso. Por ejemplo, en el sistema de pedidos nombrado anteriormente, sin conocer prácticamente ningún detalle sobre cómo funcionará, podemos decir que:

1) El grupo de usuarios que ingrese pedidos al sistema será un actor.

2) El grupo de usuarios que haga otras operaciones con los pedidos, como por ejemplo autorizarlos, cancelarlos y modificar sus plazos de entrega, será un actor.

3) Todo grupo de usuarios que reciba ciertos informes del sistema, como por ejemplo estadísticas de ventas, será un actor.

Es común que los distintos actores coincidan con distintas áreas de la empresa en la que se implementará el sistema, o con jerarquías dentro de la organización (empleado, supervisor y gerente son distintos actores, si realizan tareas distintas).

Todos los actores participan de los casos de uso. Ahora bien, es lógico que existan intersecciones entre lo que hacen los distintos actores. Por ejemplo, un supervisor puede autorizar pedidos, pero también puede ingresarlos. Veremos más adelante cómo, definiendo actores abstractos, podemos especificar este comportamiento común para evitar redundancia.

Como mencionamos anteriormente, *un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios*. Un caso de uso es iniciado por un actor. A partir de ese momento, ese actor, junto con otros actores, intercambian datos o control con el sistema, participando de ese caso de uso.

El nombre de un caso de uso se expresa con un verbo en gerundio, seguido generalmente por el principal objeto o entidad del sistema que es afectado por el caso. Gráficamente, los casos de uso se representan con un óvalo, con el nombre del caso en su interior.

Es importante notar que el nombre del caso siempre está expresado desde el punto de vista del actor y no desde el punto de vista del sistema. Por eso el segundo caso de uso se llama *Recibiendo información de pedidos* y no *Generando información de pedidos*.

Los casos de uso tienen las siguientes características:

1) Están expresados desde el punto de vista del actor.

2) Se documentan con texto informal.

3) Describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él, aunque el énfasis está puesto en la interacción.

4) Son iniciados por un único actor.

5) Están acotados al uso de una determinada funcionalidad –claramente diferenciada– del sistema.

El último punto es tal vez el más difícil de definir. Uno podría, después de todo, decir que todo sistema tiene un único caso de uso *Usando el Sistema*. Sin embargo, la especificación resultante sería de poco utilidad para entenderlo; sería como implementar un gran sistema escribiendo un único programa.

La pregunta importante es: ¿Qué es una “funcionalidad claramente diferenciada”? Por ejemplo, ¿ingresar pedidos es un caso de uso y autorizarlos es otro? ¿Cancelar los pedidos, es otro caso de uso, o es parte del caso de uso referido al ingreso de pedidos? Si bien se pueden encontrar argumentos válidos para cualquiera de las dos alternativas, en principio la respuesta a todas estas preguntas es que son todos casos de uso distintos.

Lamentablemente, si en la programación los criterios para dividir la funcionalidad en programas suelen ser difusos, los criterios para dividir la funcionalidad de un sistema en casos de uso son aún más difusos, y por esto se hace importante usar el sentido común en estas decisiones.

En principio podríamos decir que la regla general es: *una función del sistema es un caso de uso si se debe indicar explícitamente al sistema que uno quiere acceder a esa función*. Por ejemplo, si uno quiere dar de alta un pedido, accederá a la funcionalidad de alta de pedidos del sistema. Sin embargo, si uno quiere dar de alta un campo del pedido, no debe indicar al sistema que quiere acceder a esa función. Dar de alta un campo de un pedido es una función que forma parte de un caso de uso mayor: dar de alta un pedido.

Esta regla, si bien puede ser útil, no debe seguirse al pie de la letra, ya que se puede prestar a confusiones. Por ejemplo, supongamos que uno quiere especificar un sistema en el cual los usuarios pueden ver un pedido, y tienen disponibles funciones para ver el siguiente pedido, el anterior, el último y el primero. El actor debe indicar al sistema que quiere acceder a cada una de esas funciones, y según nuestra regla serían todas ellas casos de uso distintos. Sin embargo, en esta situación es mucho más práctico definir un único caso de uso *navegando pedidos*, que especificarlos todos como casos de uso distintos.

Cuando pensamos en el grado de detalle de la división de los casos de uso también resulta útil imaginar que uno está escribiendo el manual del usuario del sistema. A nadie se le ocurriría escribir un manual de usuario con un solo capítulo en el que se describe toda su funcionalidad. De la misma forma, no se debe escribir una especificación con un solo caso de uso. A pesar de saber que uno se quiere mantener lejos de este extremo, la cantidad de capítulos del manual es variable dependiendo de la persona que lo escriba.

***Descripción de los Casos de Uso***

Los casos de uso se documentan con texto informal. En general, se usa una lista numerada de los pasos que sigue el actor para interactuar con el sistema. A continuación se muestra una parte simplificada de la descripción del caso de uso “Ingresando Pedido”.

Una limitación de los casos de uso es que no hay una sintaxis clara para indicar, dentro de la descripción del caso, las decisiones e iteraciones. De esta forma, es común que en las descripciones de los casos se deba recurrir a frases como “Se repite el paso X hasta que ocurre C”, o “Si ocurre C se pasa al paso X”. En estas situaciones lo importante no es la forma en la que se expresan las condiciones e iteraciones, sino hacerlo de una forma consistente. Si la descripción del caso fuera muy compleja, es conveniente usar notaciones gráficas, por ejemplo los diagramas de actividad.

***2.3. Alternativas***

Durante la ejecución de un caso de uso, suelen aparecer errores o excepciones. Por ejemplo, mientras se ingresa un pedido, el cliente puede solicitar un producto que está discontinuado. El sistema deberá en este caso informar esta situación al empleado que ingresa el pedido. Esas desviaciones del curso normal del caso de uso se llaman alternativas. Las alternativas tienen las siguientes características:

1) Representan un error o excepción en el curso normal del caso de uso.

2) No tienen sentido por sí mismas, fuera del contexto del caso de uso en el que ocurren.

Si bien en la bibliografía las alternativas se documentan al final del caso de uso, la experiencia demuestra que resulta útil documentar los casos en tablas, mostrando el curso principal en la primera columna, y las alternativas en una segunda columna, como lo muestra el siguiente ejemplo:

...

De esta forma, es mucho más simple ver en qué parte del caso de uso puede ocurrir la excepción, y se mantiene la ventaja de poder leer de corrido el curso normal.

**Modularización de Casos de Uso**

Las próximas secciones muestran cómo se puede organizar una especificación que utiliza casos de uso para evitar redundancia (duplicación innecesaria de información) y facilitar su comprensión.

***Relaciones de Extensión***

Muchas veces, la funcionalidad de un caso de uso incluye un conjunto de pasos que ocurren sólo en algunas oportunidades. Supongamos que estamos especificando un sistema en el cual los clientes pueden ingresar pedidos interactivamente, y que dentro de la funcionalidad del ingreso de pedidos el usuario puede solicitar al sistema que le haga una presentación sobre los nuevos productos disponibles, sus características y sus precios.

En este caso, tengo una excepción dentro del caso de uso *Ingresando Pedido*. La excepción consiste en interrumpir el caso de uso y pasar a ejecutar el caso de uso *Revisando Presentación de Nuevos Productos*. En este caso decimos que el caso de uso *Revisando Presentación de Nuevos Productos* **extiende** el caso de uso *Ingresando pedido* y se representa por una línea de trazos desde el caso que ‘extiende a’ al caso que es

Las extensiones tienen las siguientes características:

1) Representan una parte de la funcionalidad del caso que no siempre ocurre.

2) Son un caso de uso en sí mismas.

3) No necesariamente provienen de un error o excepción. En su libro, Jacobson ejemplifica los casos de uso con ir a cenar a un restaurant. Para él, tomar café después de cenar es un ejemplo de una extensión.

La pregunta que surge claramente es ¿cuál es la diferencia entre una alternativa y una extensión? La respuesta puede derivarse de las características de cada uno:

* Una extensión es un caso de uso en sí mismo, mientras que una alternativa no.
* Una alternativa es un error o excepción, mientras que una extensión puede no serlo.

De todas formas, en la práctica aparecen dudas con respecto a la conveniencia de considerar algo optativo en un caso como una alternativa o una extensión, sobre todo porque no queda claro si algo puede ser visto como un caso de uso en sí mismo o no. Como regla aproximada en este caso podemos pensar que si algo opcional debe ser expresado con más de un paso, seguramente es una extensión y no una alternativa.

***Relaciones de Uso***

Es común que la misma funcionalidad del sistema sea accedida a partir de varios casos de uso. Por ejemplo, la funcionalidad de buscar un producto puede ser accedida desde el ingreso de pedidos, desde las consultas de productos, o desde los reportes de ventas por producto. ¿Cómo hago para no repetir el texto de esta funcionalidad en todos los casos de uso que la acceden? La respuesta es simple: sacando esta funcionalidad a un nuevo caso de uso, que es usado por los casos de los cuales fue sacada. Este tipo de relaciones se llama *relaciones de uso* y se representa por una línea punteada desde el caso que ‘usa a’ al caso que es ‘usado’.

Decimos, por ejemplo, que el caso de uso O*bteniendo reporte de ventas por producto* usa al caso de uso.

Este concepto no es novedoso, es simplemente el concepto de la subrutina o subprograma usado en un nivel más alto de abstracción.

Las características de las relaciones de uso son:

1) Aparecen como funcionalidad común, luego de haber especificado varios casos de uso.

2) Los casos usados son casos de uso en sí mismos.

3) El caso es usado *siempre* que el caso que lo usa es ejecutado. Esto marca la diferencia con las extensiones, que son opcionales.

La definición de las relaciones de uso y extensión deja una zona sin definir:

¿Qué pasa con la funcionalidad que es común a varios casos de uso, pero al mismo tiempo es opcional? Por ejemplo, pensemos en la impresión de un comprobante, algo que el usuario de un sistema puede o no hacer en distintos casos de uso. Si uno se guía por la funcionalidad común a varios casos, piensa que el caso de uso *imprimiendo comprobante* es **usado** por otros casos, pero si se guía por la opcionalidad, piensa que **extiende** a otros casos. Como esto no queda claro a partir de la bibliografía, *creemos conveniente que este tipo de situaciones se especifiquen como extensiones*, ya que de esta forma podemos remarcar gráficamente la opcionalidad de la relación.

***Actores y Casos de Uso Abstractos***

Al modularizar la especificación, identificando relaciones de uso y extensión, puede pasar que extraigamos casos de uso que son accedidos por varios actores. Por ejemplo, el caso de uso *buscando datos de producto* es accedido por muchos actores (el empleado de ventas que ingresa un pedido, el gerente que quiere obtener estadísticas por producto, el supervisor que quiere consultar la información de algún producto, etc.). Ahora bien, como el caso de uso nunca se ejecuta fuera del contexto de otro caso de uso, decimos que es un caso de uso *abstracto*. Lo llamamos abstracto porque no es implementable por sí mismo: sólo tiene sentido como parte de otros casos.

De la misma forma, el actor que participa de este caso de uso, que reúne características comunes a todos los actores de los casos de uso que lo usan, es un actor abstracto. En nuestro ejemplo, si bien el nombre suena poco elegante, podemos decir que tenemos un actor abstracto “Buscador de Datos de Producto”. Los actores abstractos, entonces, son necesarios para no “dejar sin actores” a los casos de uso abstractos.

La duda ahora es cómo relacionar este actor abstracto con los actores concretos: los que sí existen en la realidad y ejecutan casos de uso concretos, como *ingresando pedido* y *obteniendo estadísticas de ventas*.

Para esto podemos usar el concepto de herencia, uno de los conceptos básicos de la orientación a objetos. Como todos los actores concretos también ejecutan el caso *buscando datos de producto*, a través de la relación de uso, podemos decir que los actores concretos *heredan* al actor abstracto.

La relación de herencia no necesariamente implica la existencia de un caso abstracto. Puede ocurrir que un actor ejecute todos los casos que ejecuta otro actor, y algunos más. En nuestro sistema, el supervisor de ventas puede hacer todo lo que hace el empleado de ventas, pero además puede autorizar pedidos. En este caso, podemos decir que el Supervisor de Ventas *hereda* al Empleado de Ventas, aunque el Empleado de Ventas no sea un actor abstracto. De esta forma, toda la funcionalidad que está habilitada para el Empleado de Ventas también lo está para el Supervisor.

Usando la herencia o especialización en el análisis de requerimientos evitamos redundancia y simplificamos la especificación y los gráficos de casos de uso.

**Ejercitación**

**1)** A partir de las siguientes descripciones, identificar a el/los actores y detallar el curso normal y alternativas del caso de uso (completar la descripción).

a. “... el usuario introduce la tarjeta en el cajero, el sistema le solicita la clave, si es correcta le solicita la operación a realizar (si el número no es correcto, le pide reintento), el usuario introduce la opción retirar dinero...”

b. “... el usuario solicita un libro, el sistema verifica la condición del socio (si es moroso o ya retiro más de tres libros). El sistema controla si el libro está disponible para préstamo...”

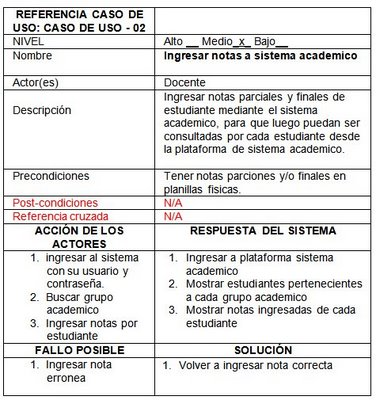
**2)** Haga el diagrama UML de casos de uso que modele este sistema

La empresa CelularSoft ha enviado recientemente un grupo de investigación a  
Japón, y ha querido replicar acá en Argentina una tendencia que está dando muy  
buenos resultados en ese país asiático.

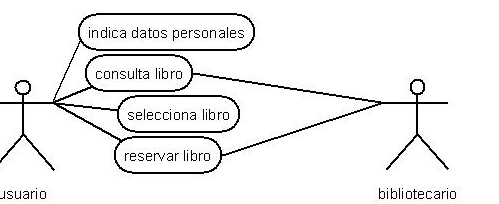
Se trata de un sistema de pagos utilizando teléfonos celulares. Hoy día muchos de los teléfonos celulares cuentan con antenas Bluetooth y/o WiFi. Los japoneses pueden utilizar estos celulares para realizar múltiples pagos (similar a lo que actualmente se hace con tarjetas de crédito o débito). Por ejemplo, al acercar el teléfono a una máquina de bebidas o alimentos y seleccionar un producto, la máquina le envía un mensaje al celular para cargarle el cobro. Cuando el usuario acepta el pago en su clular, la máquina le entrega el producto comprado, y el cobro le llegará l usuario en su cuenta telefónica.

También se puede utilizar este sistema para comprar productos en tiendas, supermercados, y en todos los locales que quieran implementar este tipo de cobros/pagos.

**3)** Dada la siguiente especificación realice el gráfico del caso de uso.



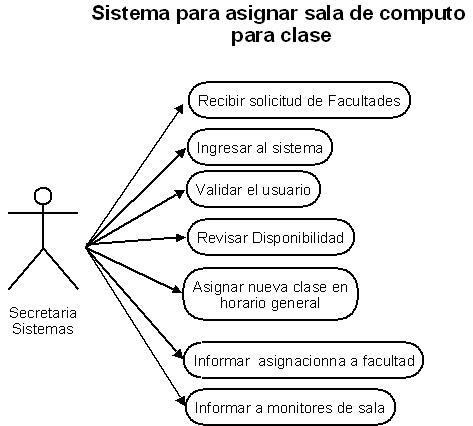
**4)** Dado el siguiente grafico realice la especificación del caso de uso



**5)** Determine si las siguiente Afirmaciones son Verdaderas o Falsas.

|  |  |  |
| --- | --- | --- |
| **Afirmaciones** | **V** | **F** |
| Los actores de un sistema representan, en particular, personas (mas precisamente roles que interpretan personas), dispositivos u otros sistemas, y en general, cualquier cosa que interactúa con dicho sistema. |  |  |
| Los casos de uso, sus especificaciones y el diagrama de casos de uso de un sistema permiten acordar, entre el equipo de desarrollo y el cliente, los límites y los requisitos funcionales de dicho sistema. |  |  |
| La especificación de un caso de uso describe cómo se implementa el comportamiento requerido para el sistema en dicho caso de uso. |  |  |
| El diagrama de casos de uso de un sistema puede organizarse por medio de relaciones que se pueden dar entre los diferentes casos de uso. Estas relaciones son las de: generalización/especialización, inclusión, y extensión. |  |  |
| Debería utilizarse una relación de extensión, entre casos de uso, cuando es necesario factorizar el comportamiento común a varios casos de uso en otro caso de uso. |  |  |
| Un caso de uso incluido en otros, es un caso de uso que es “usado” por esos otros casos de uso. El caso de uso “usado” se “activa” toda vez que el caso de uso que lo usa se “activa”. |  |  |
| Según la definición de Casos de Usos un actor es parte del Sistema |  |  |
| Un actor no puede vincularse a más de un Caso de Uso |  |  |
| Dos actores pueden vincularse a través de una relación de especialización o herencia. |  |  |
| Un caso de Uso puede aparecer con diferentes nombres en diferentes diagramas de Casos de Usos. |  |  |
| El mismo Caso de Uso puede poseer descripciones complementarias en diferentes diagramas de Casos de Uso. |  |  |

**SOLUCION 3**

****

|  |  |  |
| --- | --- | --- |
| **Afirmaciones** | V | F |
| Los actores de un sistema representan, en particular, personas (mas precisamente roles que interpretan personas), dispositivos u otros sistemas, y en general, cualquier cosa que interactúa con dicho sistema. | X |  |
| Los casos de uso, sus especificaciones y el diagrama de casos de uso de un sistema permiten acordar, entre el equipo de desarrollo y el cliente, los límites y los requisitos funcionales de dicho sistema. | X |  |
| La especificación de un caso de uso describe cómo se implementa el comportamiento requerido para el sistema en dicho caso de uso. | X |  |
| El diagrama de casos de uso de un sistema puede organizarse por medio de relaciones que se pueden dar entre los diferentes casos de uso. Estas relaciones son las de: generalización/especialización, inclusión (USO), y extensión. | X |  |
| Debería utilizarse una relación de extensión, entre casos de uso, cuando es necesario factorizar el comportamiento común a varios casos de uso en otro caso de uso. | X |  |
| Un caso de uso incluido en otros, es un caso de uso que es “usado” por esos otros casos de uso. El caso de uso “usado” se “activa” toda vez que el caso de uso que lo usa se “activa”. | X |  |
| Según la definición de Casos de Usos un actor es parte del Sistema |  | X |
| Un actor no puede vincularse a más de un Caso de Uso |  |  |
| Dos actores pueden vincularse a través de una relación de especialización o herencia. | X |  |
| Un caso de Uso puede aparecer con diferentes nombres en diferentes diagramas de Casos de Usos. |  | X |
| El mismo Caso de Uso puede poseer descripciones complementarias en diferentes diagramas de Casos de Uso. | X |  |