# Casos de Uso

## Metodología de trabajo para Identificación de Casos de Uso

Rumbaugh, Booch y Jacobson en su libro "*El proceso unificado de desarrollo de software*" (Rumbaugh, Jacobson, & Booch, El Proceso Unificado de Desarrollo de Software, 1ra ed. 2000) plantean una metodología de trabajo para realizar en forma correcta el MCU que consiste básicamente en encontrar los actores que van a interactuar con el sistema, encontrar los casos de uso para cada actor y describirlos primero brevemente y luego en forma detallada para completar el modelo. Identificar actores y casos de uso es una actividad muy importante ya que ayuda a delimitar el sistema y a definir su alcance.
Si bien esta actividad es responsabilidad del analista de sistemas, se requiere un grado de participación importante del cliente, ya que este último es quien proporciona los requerimientos y la información acerca de los distintos roles y requerimientos/funcionalidades del sistema que se transformaran en casos de uso.

El desarrollo del MCU es iterativo, el analista de sistemas junto a su equipo pueden crear un modelo preliminar que luego se validará con el cliente quien podría sugerir modificaciones o agregados de funcionalidades nuevas, de esta manera se va refinando el modelo hasta llegar a la versión definitiva que será aprobada por el cliente y sirve como contrato donde se especifican todas las funcionalidades que el sistema debe cumplir.

Permite representar distintos escenarios que piden presentarse al desarrollarse una funcionalidad completa del sistema a modelar.

Un escenario es una secuencia de acciones/pasos que permite describir la funcionalidad del caso de uso.

> Escenario básico (Flujo Normal -> todo va bien)
> Escenarios alternativos (Flujos Alternativos -> gestión de errores y/o excepciones)

Aplicar esta metodología consiste en:

1) **Identificar actores**: antes de avanzar con los casos de uso se debe tratar de identificar a todos los actores que van a interactuar con el sistema. El analista de sistemas junto con el cliente tiene la tarea de identificar los usuarios y organizarlos según el rol que van a representar al interactuar con el sistema, esta tarea permite definir una lista de actores candidatos. En un primer momento es común encontrar varios actores que cumplan con el mismo rol, en estos casos habrá que revisar la lista de actores candidatos para finalmente

encontrar el conjunto de actores adecuados y eVentualmente definir generalizaciones en los casos necesarios. El paso siguiente será asignarle a cada actor un nombre que defina claramente el rol que está representando y describir brevemente para que utilizará el sistema, indicando necesidades y responsabilidades.

Como resultado de este paso se obtiene un conjunto de actores actualizado, cada uno con su descripción, que pueden utilizarse como punto de partida para encontrar los casos de uso.

2) **Identificar los casos de uso para cada actor**: el analista de sistemas irá revisando los actores y proponiendo los casos de uso para cada uno, estos casos de uso propuestos en primera instancia son casos de uso candidatos y no todos llegarán a ser casos de uso por sí mismos. Recordemos que un caso de uso debe representar una funcionalidad completa del sistema y dejar un resultado de valor para el actor, revisar minuciosamente los casos de uso candidatos es una tarea importante ya que este análisis permite identificar las funcionalidades completas que representan casos de uso en sí mismos y las que son sólo parte de otro caso de uso. Una vez encontrados los casos de uso para cada actor se le debe asignar un nombre a cada uno, el nombre de un caso de uso siempre comienza con un verbo en infinitivo y debe reflejar cual es el objetivo de la interacción entre el actor y el sistema visto desde el punto de vista del actor, por ejemplo, un caso de uso llamado *Registrar Venta* refleja claramente que el objetivo del *Empleado de Ventas* (actor) al interactuar con el sistema es que la Venta quede registrada en el mismo.

   Como resultado de este paso, se obtiene un primer diagrama de casos de uso sujeto a evaluaciones y restructuraciones.

3) **Identificar nuevos casos de uso a partir de los existentes**: en este paso se debe verificar si el sistema está completo o completarlo buscando otras funcionalidades a partir de:

   a. *Variaciones significativas de los casos de uso existentes*: se pueden encontrar nuevos casos de uso a partir de las variantes de un caso de uso existente. Por ejemplo, si existieran muchas diferencias entre registrar la Venta a un cliente minorista y registrar la Venta a un cliente mayorista, entonces, del caso de uso *Registrar Venta* se podrían obtener los casos de uso *Registrar Venta a cliente minorista* y *Registrar Venta a cliente mayorista*, siempre y cuando las variaciones del caso de uso lo justifiquen.

   b. *Casos de uso opuestos*: se podrían encontrar nuevas funcionalidades para completar el sistema buscando casos de uso opuestos a los existentes, por ejemplo, si existe el caso de uso *Registrar Venta*, debería existir también el caso de uso *Anular Venta*.

c. *Casos de uso que preceden a casos de uso existentes*: siempre hay que preguntarse qué sucede antes del caso de uso que se está analizando. Por ejemplo, si se está registrando una Venta a un cliente, el cliente ya tendría que estar registrado como tal en el sistema, por lo tanto, debería existir otro caso de uso llamado *Registrar Cliente*.

d. *Casos de uso que suceden a los casos de uso existentes*: en este caso hay que preguntarse qué ocurre después del caso de uso que se está analizando. Por ejemplo, ¿qué ocurre después de *Registrar Venta*?, ésta es una buena forma de asegurarse que no se están dejando requerimientos sin identificar y permite descubrir partes de alguna nueva funcionalidad y nuevos casos de uso.

Luego de analizar todos estos casos, se obtiene como resultado un segundo diagrama de casos de uso más completo donde se agregan los nuevos casos de uso encontrados.

4) **Describir los casos de uso con una Descripción General**: una vez identificados todos los casos de uso hay que empezar a documentarlos haciendo la descripción con trazo grueso. Recordemos que ésta debe ser una breve descripción, realizada desde la perspectiva del actor, que especifique lo que hace el caso de uso indicando quien es el actor involucrado, la funcionalidad que se va a llevar a cabo y cuál es el resultado de valor que deja ese caso de uso para el actor.

   Como resultado de este paso se obtiene el MCU compuesto por el diagrama de casos de uso completo y la descripción general de cada caso de uso.

   Este MCU representa los requerimientos del cliente “convertidos” en casos de uso que especifican las funcionalidades que el sistema debe cumplir junto a sus descripciones y a los actores que van a interactuar con las mismas.

   Llegado este punto, el cliente deberá ver el MCU para determinar si los casos de uso capturan todos sus requerimientos, si la descripción de los mismos es correcta, completa y comprensible y si identifica algún caso de uso que no proporcione ningún resultado de valor, en cuyo caso habría que reconsiderar la inclusión del mismo en el modelo.

   Es muy importante que el analista de sistemas junto con su equipo revisen bien éste modelo antes de presentárselo al cliente ya que el mismo servirá como contrato donde se especifica todo lo que el sistema “garantiza que cumplirá” si el cliente decide aceptar y se avanza con el proyecto.

5) **Priorizar los casos de uso:** en este paso se deben definir prioridades y seleccionar los casos de uso de la primera iteración. Una vez documentados los casos de uso con su Descripción General, es conveniente definir las prioridades de los mismos, para determinar cuáles serán tenidos en cuenta en las primeras

iteraciones del proceso de desarrollo y cuáles se pueden dejar para más adelante. Sabiendo que los casos de uso especifican los requisitos funcionales del sistema basados en los requerimientos del cliente, para priorizarlos se debe tener en cuenta la siguiente categorización:

a. Requerimientos *imprescindibles o de prioridad alta*, son aquellos requerimientos que si no se implementan el sistema no funcionaría y no tendría sentido.

b. Requerimientos *importantes o de prioridad media*, son aquellos requerimientos muy deseables pero no imprescindibles y se pueden negociar con el cliente.

c. Los requisitos *deseables o de prioridad baja*, son aquellos requerimientos posibles de implementar pero que podrían eliminarse. Solo se implementan si hay tiempo disponible.

Una vez categorizados los casos de uso, se seleccionan los que se van a desarrollar en la primera iteración, en general, se incluyen los de prioridad alta, se discuten los de prioridad media y se descartan los de prioridad baja cuyo costo sea alto.

Como resultado de este paso se obtienen los casos de uso priorizados para comenzar a planificar cuáles se van a desarrollar en cada iteración del proceso de desarrollo.

6) **Especificar Detalladamente los casos de uso:** una vez seleccionados los casos de uso que se van a implementar en la primera iteración, se realiza la descripción detallada de cada caso de uso describiendo el paso a paso de interacción entre el actor y el sistema, siempre realizada desde *el punto de vista del actor*, y se divide en el flujo normal, que muestra el curso normal de los acontecimientos, el caso en “que todo sale bien” y el flujo alternativo que muestra las desviaciones del curso normal por posibles errores o excepciones. Es muy importante describir todas las alternativas o de lo contrario no se estaría especificando por completo el caso de uso, estas desviaciones del curso normal, errores o excepciones pueden ocurrir por varias razones, por ejemplo, el actor puede elegir entre diferentes caminos en el caso de uso, sólo uno de esos caminos (el que habitualmente se va a seguir) se debe describir en el curso normal y los demás se describen como alternativas. Otro ejemplo de errores o excepciones se puede dar cuando el sistema detecta una entrada errónea por parte del actor, lo cual también se debe contemplar como una alternativa al curso normal.
En este paso se crea un prototipo de interfaz del usuario para cada caso de uso a describir que ilustre cómo el actor puede ejecutarlo y brinde una idea de como se verá el sistema. Teniendo una interfaz tentativa donde se visualicen los datos involucrados en la ejecución del caso de uso resulta más sencillo escribir

la descripción detallada de la interacción indicando las acciones del actor y las respuestas del sistema.

Como resultado de este paso se obtiene la descripción detallada de los casos de uso junto a sus prototipos de interfaz.

7) **Identificar relaciones entre casos de uso:** luego de describir los casos de uso con trazo fino, revisando estas descripciones, se pueden detectar secuencias de pasos que se repiten *siempre* en distintos casos de uso. Para evitar este tipo de redundancia, si no existiera, se debe crear un nuevo caso de uso que describa esta secuencia de pasos repetitiva y relacionarlo con los casos de uso que incluyen dicha secuencia a través de una relación *include*, de esta manera cada vez que se ejecute un caso de uso A que incluya al caso de uso B, también se ejecutará B. Ahora bien, si se encuentra una secuencia de pasos que se repiten en distintos casos de uso pero solo en algunas oportunidades, dicha secuencia sería *opcional* o alternativa dentro de la ejecución de un caso de uso dado, y en caso de ejecutarse, estaría extendiendo o añadiendo funcionalidad al caso de uso en cuestión. En estos casos, si no existiera, se crea un nuevo caso de uso que describa la secuencia opcional y se lo relaciona con los casos de uso extendidos a través de una relación *extend*. De esta manera un caso de uso A que extiende a un caso de uso B se puede ejecutar o no cada vez que se ejecute B, la llamada a la ejecución del caso de uso A se hará en algún punto del flujo alternativo dentro de la descripción detallada del caso de uso B, una vez finalizado el caso de uso A se continua con la secuencia descripta dentro del trazo fino del caso de uso B.
   Así se puede ver claramente la diferencia entre las relaciones extend e include, a diferencia del caso anterior, si un caso de uso A incluye al caso de uso B, la llamada a la ejecución del caso de uso B se hará en algún punto dentro del flujo normal de la descripción detallada del caso de uso A ya que B se ejecutará siempre que se ejecute A.

   Como resultado de este paso se obtiene el diagrama de casos de uso completo incluyendo las relaciones entre casos de uso.

En resumen, al aplicar esta metodología de trabajo se obtiene el MCU que captura los requisitos del sistema y que está compuesto por el diagrama de casos de uso, la descripción general y la descripción detallada de cada caso de uso. Utilizando el proceso unificado de desarrollo de software como marco de trabajo, el MCU obtenido será el punto de partida para completar las etapas de análisis, diseño, implementación y prueba ya que los casos de uso van a dirigir el trabajo a lo largo de estas etapas iteración por iteración.

## Relaciones entre Casos de Uso

Existen dos tipos de relaciones entre casos de uso:

- Relaciones de extensión (extend)
- Relaciones de inclusión (include)

Estas relaciones se podrán detectar una vez hecha la descripción detallada de todos los casos de uso.

**Relaciones de extensión**: en una relación de extensión, un caso de uso que extiende a otro le agrega funcionalidad al caso de uso extendido y se ejecutará de manera *opcional* cada vez que se ejecute éste. La funcionalidad descripta en el caso de uso *"que extiende"* representa una serie de pasos opcionales o una excepción dentro de la funcionalidad descripta en el caso de uso *"extendido"* que se ejecutará sólo en algunas oportunidades. Por ejemplo, supongamos que estamos describiendo el caso de uso *Registrar Venta*, puede ocurrir que tengamos que vender a un cliente nuevo que no esté registrado en el sistema, en este caso aparece una excepción, la excepción consiste en interrumpir la ejecución de este caso de uso y pasar a la ejecución del caso de uso *Registrar cliente*, de esta manera podemos decir que el caso de uso *Registrar cliente* que se ejecuta de manera opcional ***extiende*** o agrega funcionalidad al caso de uso *Registrar Venta*.

Las relaciones de extensión entre casos de uso se representan en el diagrama con una relación de dependencia desde el caso de uso "*que extiende a"* hacia el caso de uso *"extendido"* acompañada por el estereotipo UML <<extends>>.

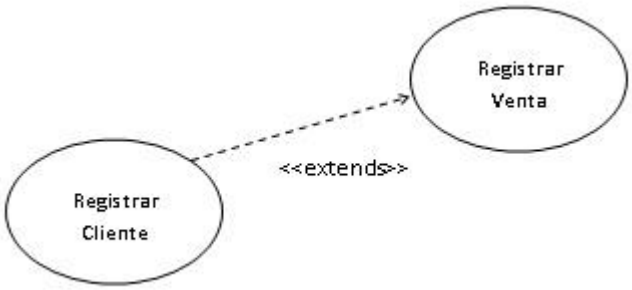



**Figura 1: Relación de extensión**

Además, las relaciones de extensión tienen las siguientes características:

- Representan una parte de la funcionalidad de un caso de uso que *no siempre* ocurre.
- La funcionalidad *"que extiende"* al caso de uso también es un caso de uso en sí mismo.
- Si bien es lo más común, no necesariamente provienen de un error o una excepción.

Diferencias entre una alternativa y una relación de extensión:

- En una relación de extensión, la funcionalidad que extiende a un caso de uso es un caso de uso en sí mismo mientras que una alternativa no lo es.
- Una alternativa es un error o excepción, mientras que una extensión puede no serlo.

**Relaciones de inclusión**: luego de realizar la descripción detallada de los casos de uso, puede ocurrir que se detecte alguna serie de pasos que se repite en dos o más casos de uso. Si esta serie de pasos representa una funcionalidad completa, a partir de ella se crea un nuevo caso de uso que estará *"incluido"* en cada uno de los casos de uso que utilicen dicha funcionalidad y que se ejecutará *siempre* que éstos se ejecuten. Por ejemplo, la funcionalidad *Buscar producto* puede ser accedida por los casos de uso *Registrar pedido*, *Registrar Venta*, etc., se puede entonces, crear el caso de uso *Buscar producto* y relacionarlo a través de una relación de inclusión con los casos de uso que lo utilizan. De esta manera, *Buscar producto*, al ser el caso de uso ***incluido*** se ejecutará siempre que se ejecuten los casos de uso que lo incluyen.

Las relaciones de inclusión se representan en el diagrama con una relación de dependencia desde el caso de uso *"que incluye"* hacia el caso de uso *"incluido"* acompañada por el estereotipo UML <<include>>.

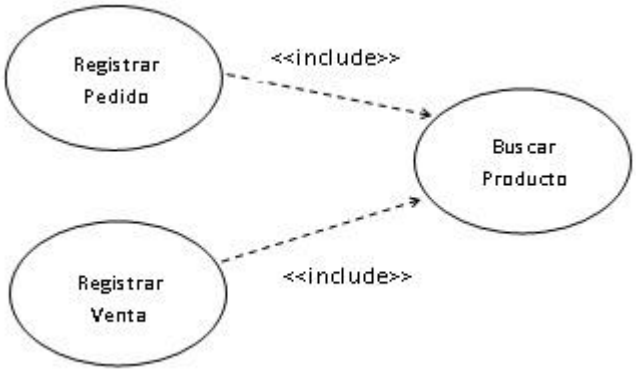



**Figura 2: Relación de inclusión**

Las características de las relaciones de inclusión son:

- Aparecen como una funcionalidad común luego de haber especificado varios casos de uso.
- Las funcionalidades incluidas son casos de uso en sí mismos.
- El caso de uso incluido se ejecuta *siempre* que el caso de uso que lo incluye se ejecute. Esta es la principal diferencia con las relaciones de extensión donde un caso de uso que extiende a otro se puede ejecutar, o no, de manera opcional.

**Aplicación Práctica del Modelo de Casos de Uso**

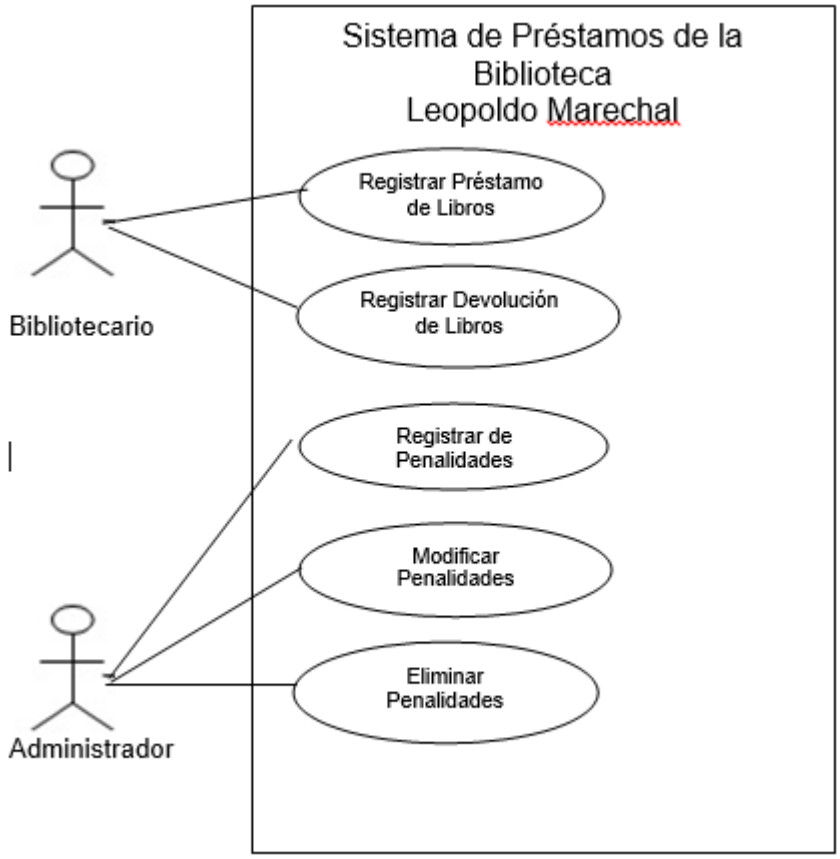
Retomando lo visto en la clase de Función Limite y Alcance, trabajaremos con el ejercicio de la Biblioteca Leopoldo Marechal de la UNLaM focalizándonos en los préstamos que se realizan a los alumnos de dicho Establecimiento.

Identificamos los siguientes actores:

- Bibliotecario
- Administrador

Identificamos los siguientes Casos de Uso:

- Registrar Préstamo de Libros
- Registrar Devolución de Libros
- Registrar de Penalidades
- Modificar de Penalidades
- Eliminar de Penalidades



Especificación de Casos de Uso

Para especificar los casos de uso, se utiliza una plantilla en la cual debe indicarse:

- **Nombre del caso de uso**: comienza con un verbo en infinitivo y está expresado desde el punto de vista del actor.
- **Actor principal**: es el que inicia el caso de uso y el que interactúa con el sistema.
- **Actor secundario**: participa una vez iniciado el caso de uso, los actores secundarios pueden ser varios o ninguno y no van a estar en el diagrama.
- **Descripción general**: es una breve descripción, a grandes rasgos, de lo que hace el caso de uso, donde se debe especificar quién es el actor involucrado, la funcionalidad que se va a llevar a cabo y cual es el beneficio que deja el caso de uso para el actor.
- **Precondición**: como precondición se debe indicar el nombre del/los casos de uso que tienen que haberse ejecutado antes del caso de uso que se está describiendo. Por ejemplo, si existe un caso de uso *Registrar Venta* de productos, la precondición debe ser *Registrar Productos*.
- **Postcondición**: como postcondición se indica cual es el resultado de interés o beneficio que deja el caso de uso para al actor.
- **Descripción detallada**: es la descripción de la interacción entre el actor y el sistema, se debe indicar paso a paso qué es lo que hace el actor y qué responde el sistema, sólo se indica la respuesta del sistema (lo que muestra), no lo que hace, por ejemplo, no se incluye en esta descripción si el sistema hace cálculos, realiza búsquedas, etc. También es común, y resulta muy intuitivo, acompañar esta descripción detallada con un prototipo de interfaz tentativa donde se puedan seguir los pasos de la interacción descripta.

  La descripción detallada de la interacción se divide en la descripción del *curso o flujo normal* y la descripción del *curso o flujo alternativo*:

  - **Flujo normal**: el flujo normal se describe en base a la interfaz y sigue el curso normal de los acontecimientos, es el caso en el que “todo sale bien”. Dentro del flujo normal puede ocurrir que el actor tenga que realizar varias interacciones generando una serie de pasos que se repiten, por ejemplo, seleccionar varios productos para una Venta. Estos casos se suelen expresar con frases como “para cada *x* seleccionado se repiten los pasos *y* a *z*” o “se repite el paso *x* hasta que ocurra *y*”, etc.
  - **Flujo alternativo**: en el flujo alternativo se describen las alternativas al flujo normal. Dentro de la interacción puede haber alternativas a las

acciones del actor y/o a las respuestas del sistema pero cabe destacar que, en general, las alternativas son a las respuestas del sistema, es decir, ante determinada acción del actor se espera que el sistema responda de una manera y lo hace de otra, en estos casos además de describir la alternativa hay que indicar a que punto del flujo normal se vuelve luego o si se finaliza el caso de uso. Por ejemplo, sabemos que durante la ejecución de un caso de uso pueden aparecer errores o excepciones que se tratarán como alternativas, si se está ejecutando el caso de uso *Registrar Préstamo de Libro* y el actor en forma errónea ingresa un código de libro inexistente, en este caso el sistema deberá informar de dicha situación y regresar al punto del flujo normal donde se permita al actor ingresar nuevamente el código del libro.

Las alternativas son entonces desviaciones del flujo normal y tienen las siguientes características:

- Representan un error o una excepción.
- No tienen sentido en sí mismas fuera del contexto del caso de uso en el que ocurren.