**UD6: ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS. DIAGRAMAS DE COMPORTAMIENTO**

**1. Introducción**

**2. Diagramas de casos de uso.**

**3. Diagramas de secuencia.**

1. **Introducción**

Recordar que UML (UnifiedModelingLanguage o Lenguaje Unificado de Modelado) es un conjunto de herramientas que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

UML permite a los desarrolladores visualizar el producto de su trabajo en esquemas o diagramas estandarizados denominados modelos (representación gráfica o esquemática de una realidad; sirve para organizar y comunicar de forma clara los elementos que involucran el desarrollo en un todo) que representan el sistema desde diferentes perspectivas.

Además UML puede conectarse a lenguajes de programación mediante ingeniería directa (transformación de un modelo en código a través de su traducción a un determinado lenguaje de programación) e inversa (transformación del código en un modelo a través de su traducción desde un determinado lenguaje de programación).

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

Tipos:

En los diagramas de comportamiento cabe destacar que al no ser diagramas estructurales, sino conceptuales de manejo, flujo y secuencia de un programa, es posible que dependiendo de la profundidad y extensión de un diagrama no aparezcan algunas de las entidades definidas en el diagrama de clases.

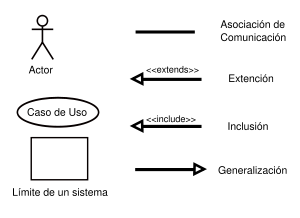
En esta unidad vamos a centrarnos en los principales diagramas de comportamiento, que son los de casos de uso y de interacción de secuencia.

1. **Diagramas de casos de uso.**

Un diagrama de casos de uso nos ayuda a determinar QUÉ puede hacer cada tipo diferente de usuario con el sistema, en una forma que los no versados en el mundo de la informática o, más concretamente el desarrollo de software, pueda entender.

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar.

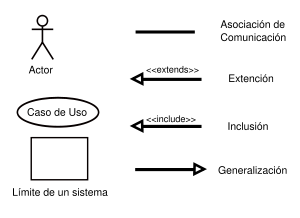
Un diagrama de casos de uso es una visualización gráfica de los requisitos funcionales del sistema. Está formado por casos de uso (se representan como elipses) y los actores, que interactúan con ellos (se representan como monigotes). Su principal función es iniciar y dirigir el proceso de creación del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



Los diagramas de casos de uso se crean en la primera etapa de desarrollo del software, y se enmarcan en el proceso de análisis, para definir de forma detallada la funcionalidad que se espera que cumpla el software, y que se pueda comunicar fácilmente con el usuario. Se incluyen en el Documento de Especificación de Requisitos.

Veamos los diferentes elementos que forman un diagrama de casos de uso:

1. *Actores*

Actor: representa un tipo de rol que juegan los usuarios del sistema (hardware, software, personas). Es cualquier usuario del sistema, entendido como cualquier cosa externa que interactúa con el sistema. No tiene por qué ser un humano, puede ser otro sistema informático o unidades administrativas de una empresa.

Un usuario del sistema puede representar diferentes roles según la operación que esté ejecutando, cada uno de estos roles representará un actor diferente, es decir, un actor representa un rol que alguien puede estar jugando, no un individuo particular. Pueden especializarse usando la relación de generalización (se ve más adelante)

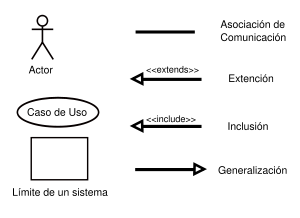
Tipos de actores:

• *Primarios*: interaccionan con el sistema para explotar su funcionalidad. Trabajan directa y frecuentemente con el software, con sus principales funciones.

• *Secundarios*: soporte del sistema (supervisan y mantienen) para que los primarios puedan trabajar. Son necesarios para alcanzar algún objetivo.

• Inici*a*dores: no interactúan con el sistema, pero desencadenan el trabajo de otro actor.

1. *Casosdeuso*

Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto.

El conjunto de casos de uso forman el “comportamiento requerido” de un sistema.

El objetivo del diagrama de casos de uso será la descripción de lo que realiza cada caso de uso, que es lo que va a ayudar al equipo de desarrollo a crear el sistema a posteriori.

La descripción de los casos de uso se realiza mediante una descripción textual de lo que se vaya a realizar. Esta descripción se llama contrato (o especificación) del caso de uso.

No hay un formato básico de contrato de caso de uso, cada organización de desarrollo de software establecerá una plantilla de documento del contrato, pero tendrá al menos los siguientes elementos:

* *Nombre*: nombre del caso de uso (expresiones verbales que describen algún comportamiento del sistema).
* *Actores*: aquellos que interactúan con el sistema a través del caso de uso.
* *Propósito*: breve descripción de lo que se espera que haga. Reflejando, posiblemente, uno o varios requisitos funcionales del sistema o una parte de algún requisito
* *Precondiciones*: aquellas que deben cumplirse para que pueda llevarse a cabo el caso de uso.
* *Flujonormal*: flujo normal de eventos que deben cumplirse para ejecutar el caso de uso exitosamente, desde el punto de vista del actor que participa y del sistema.
* *Flujoalternativo*: flujo de eventos que se llevan a cabo cuando se producen casos inesperados o poco frecuentes. No se deben incluir aquí errores como escribir un tipo de dato incorrecto o la omisión de un parámetro necesario.
* *Postcondiciones*: las que se cumplen una vez que se ha realizado el caso de uso.

El Flujo se puede describir de forma textual, lo suficientemente claro para que alguien ajeno al sistema lo entienda fácilmente.

A medida que se avanza en el proceso de desarrollo se hace uso de diagramas de interacción o actividad. Cuando se describe un flujo de eventos se debe incluir:

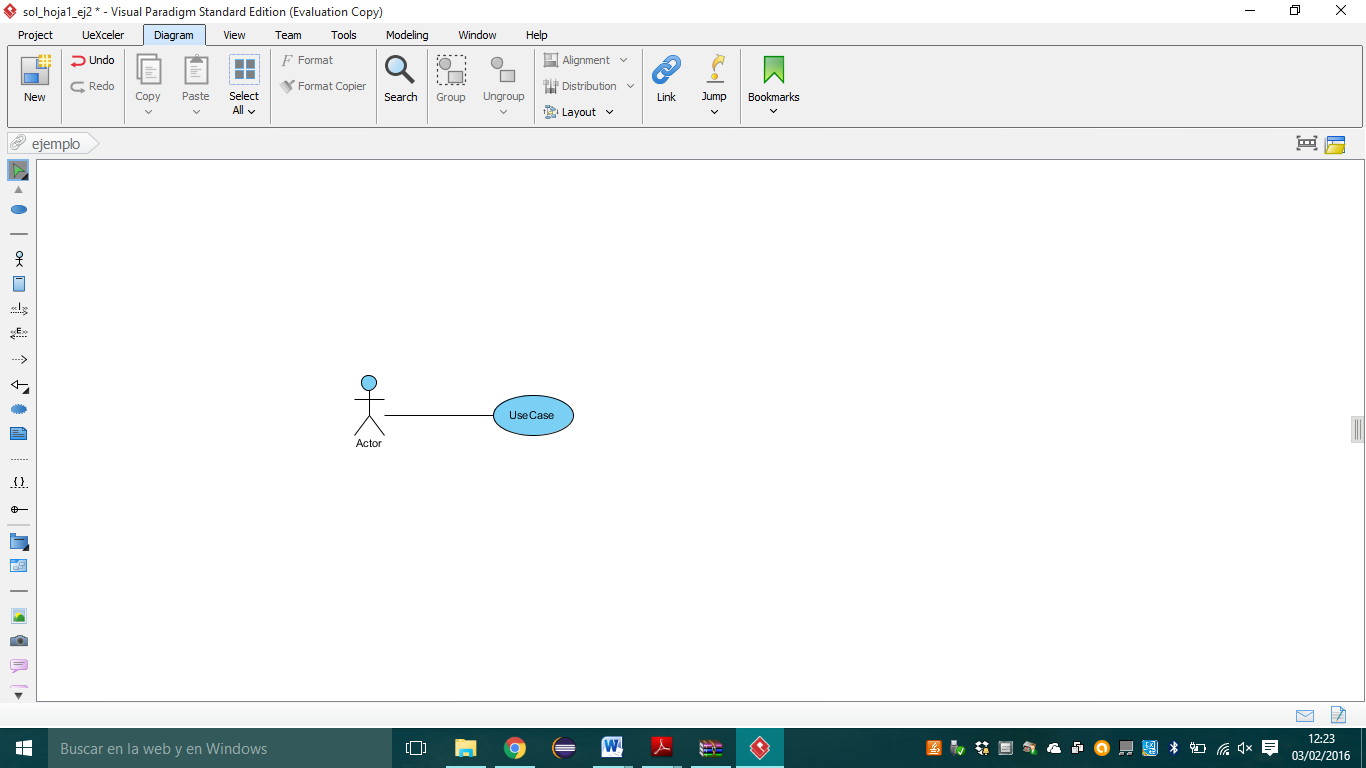
* Cómo y cuándo empieza y acaba el caso de uso.
* Cuándo interactúa con los actores y qué objetos se intercambian.
* El flujo normal y los flujos alternativos del comportamiento.

1. *Relaciones*

Las relaciones representan qué actores realizan las tareas descritas en los casos de uso, en concreto qué actores inician un caso de uso.

Tipos:

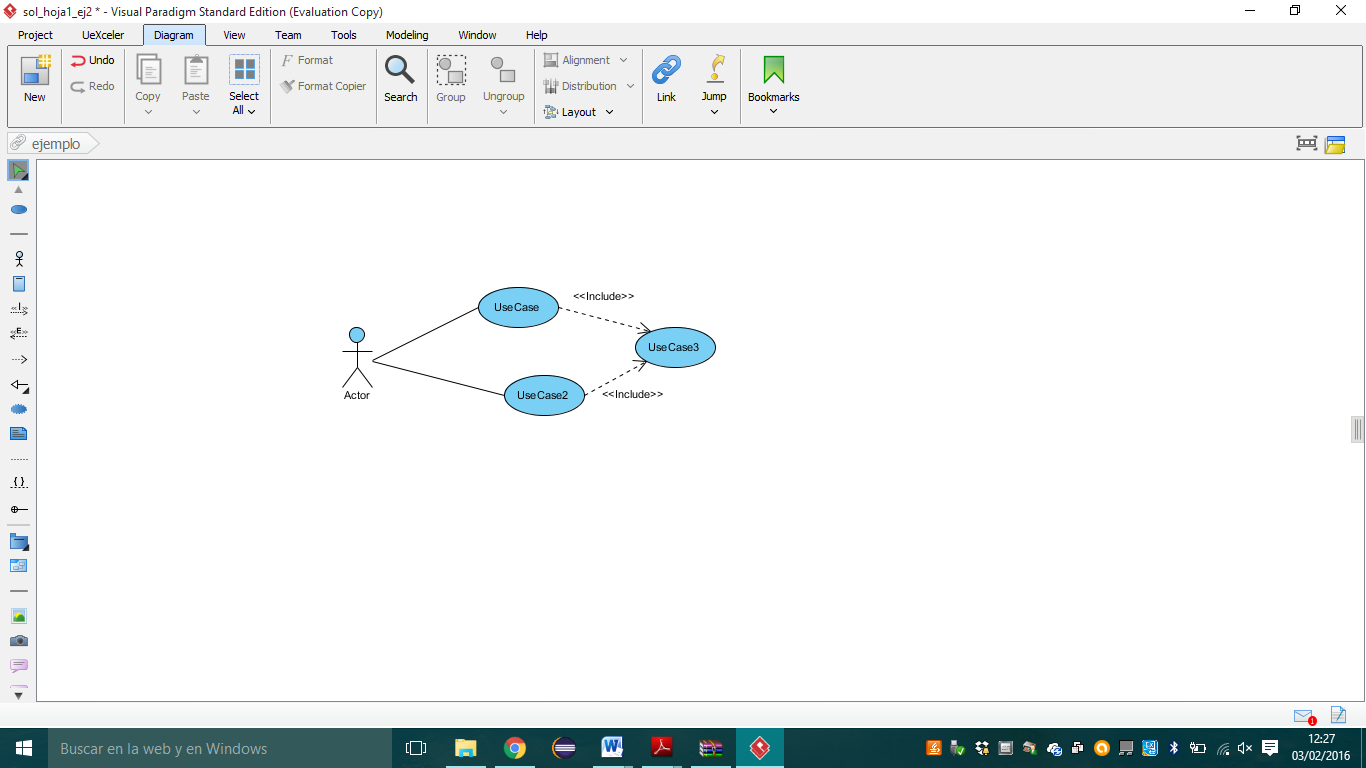
* *Asociación*: relación/comunicación entre el actor y el CU. Decimos que hay una asociación entre un actor y un caso de uso si el actor interactúa con el sistema para llevar a cabo el caso de uso o para iniciarlo. Puede haber 1 ó 2 direcciones.



* *Inclusión* (<<include>>): se utiliza cuando se quiere dividir una tarea de mayor envergadura en otras más sencillas, que son utilizadas por la primera. Representa una relación entre los casos de uso, y son muy útiles cuando es necesario reutilizar tareas, es decir, cuando se desea especificar algún comportamiento común repetido en dos o más casos de uso.Ventajas:
  + Las descripciones de los casos de uso son más cortas y se entienden mejor.
  + La identificación de funcionalidad común puede ayudar a descubrir el posible uso de componentes ya existentes en la implementación

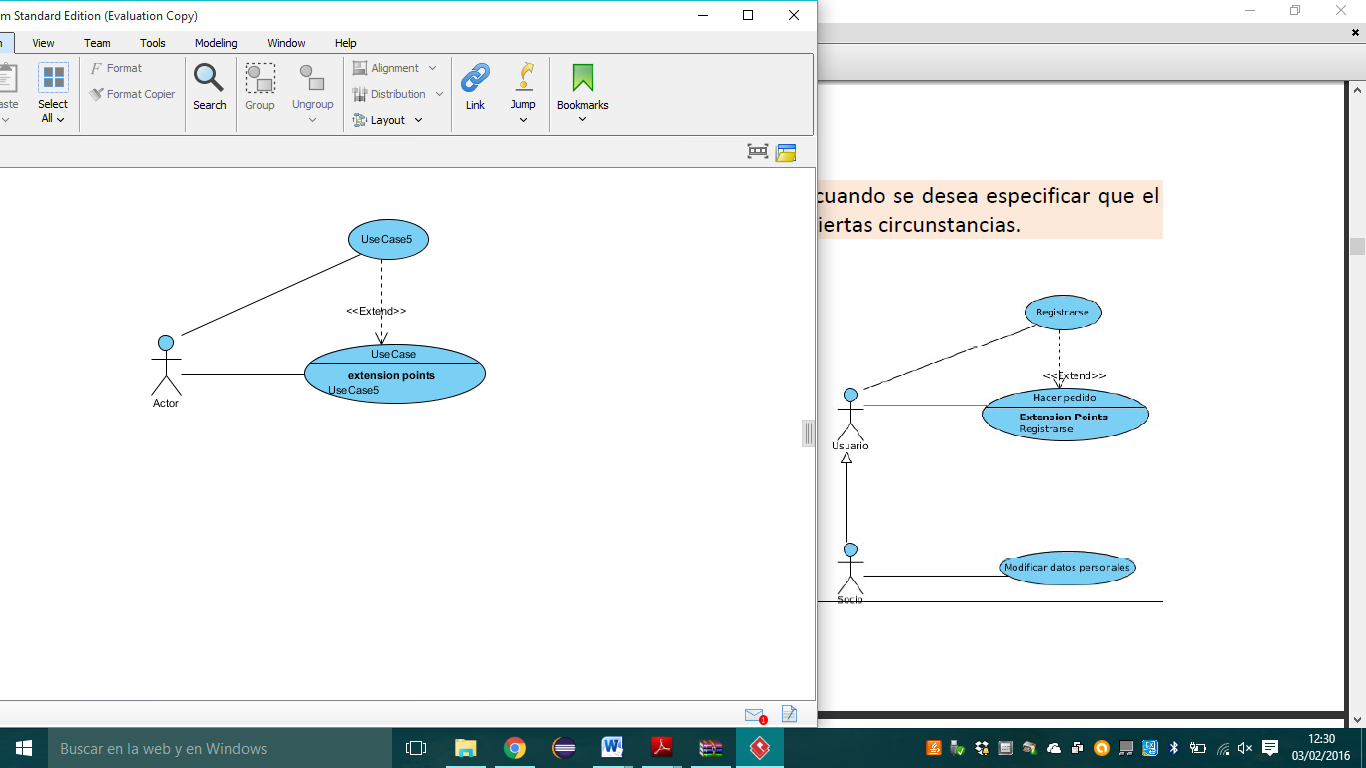
Desventajas:

* + La inclusión de estas relaciones hace que los diagramas de caso de uso sean más difíciles de leer.

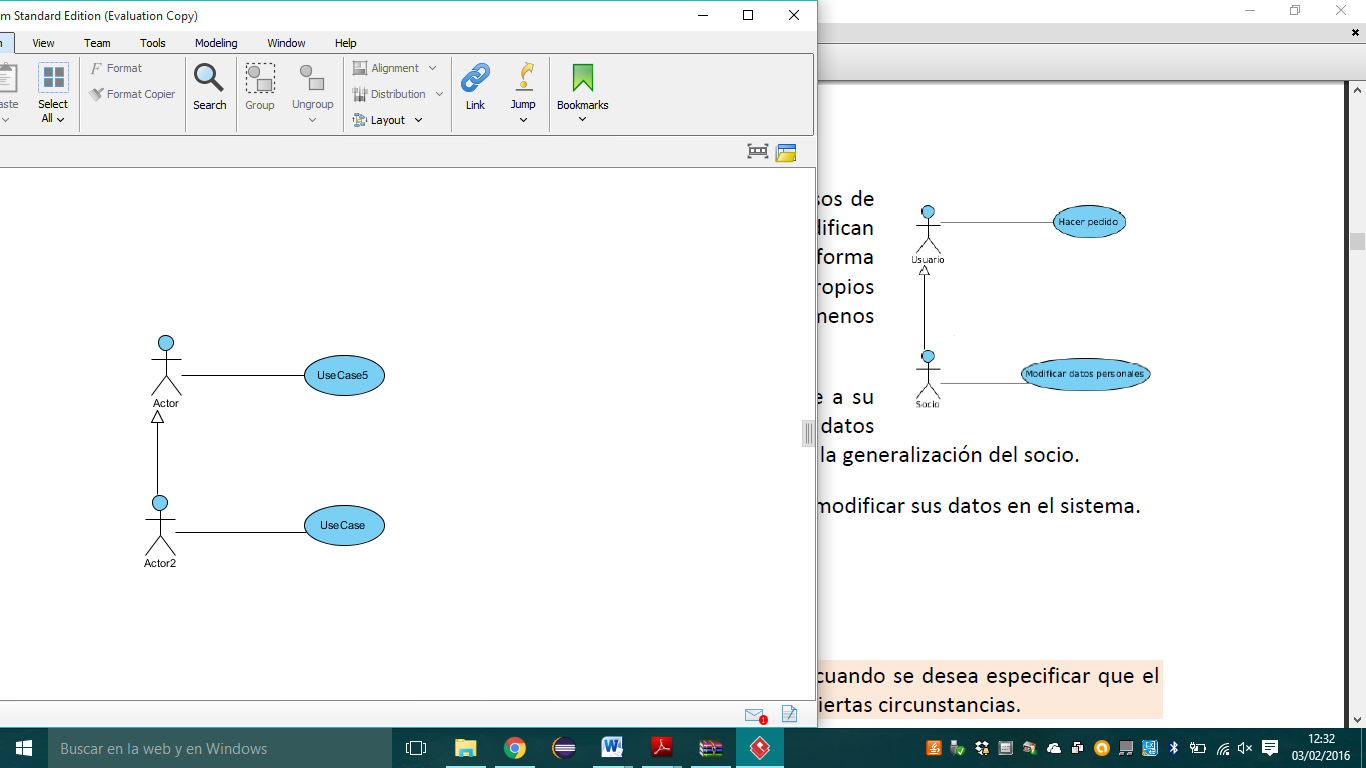


* *Extensión* (<<extends>>): Un caso extiende el comportamiento de otro caso (base). Se utiliza cuando existe una parte del caso de uso que se ejecuta sólo en determinadas ocasiones, pero no es imprescindible para su completa ejecución (es opcional).

Su principal función es simplificar el flujo de casos de uso complejos. Cuando un caso de uso extendido se ejecuta, se indica en la especificación del caso de usos como un punto de extensión.



* *Generalización*: se utiliza para representar relaciones de especialización entre casos de uso o actores. El caso general (padre) se define y luego los hijos heredan sus características añadiendo sus propios pasos o modificando alguno



1. *Escenarios*

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

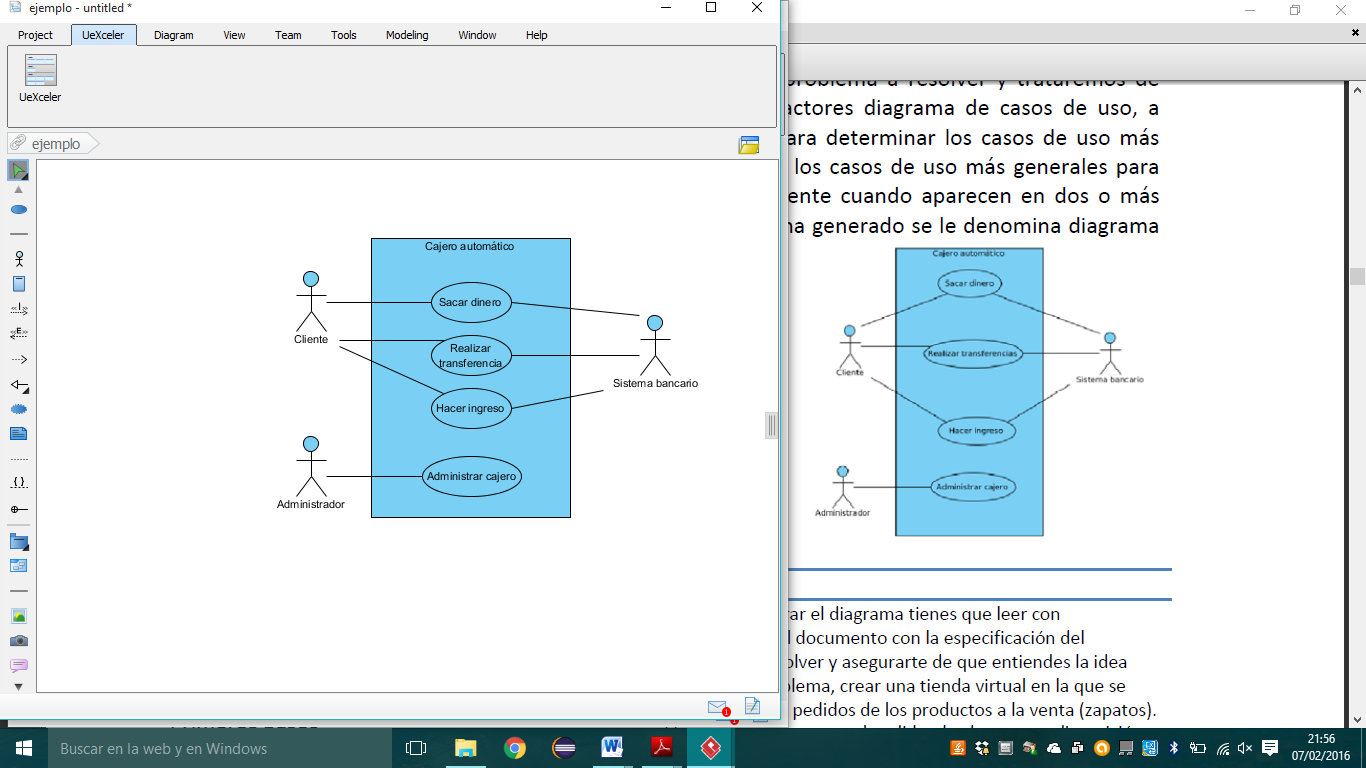
Así, un escenarioes una ejecución particular de un caso de uso que se describe como una secuencia de eventos. Un caso de uso es una generalización de un escenario.

Los escenarios deben documentarse mediante diagramas de secuencia.

1. *Diagramafrontera.*

El siguiente paso es refinar el diagrama analizando los casos de uso más generales para detectar casos relacionados por inclusión (se detectan fácilmente cuando aparecen en dos o más casos de uso generales), extensión y generalización. Al diagrama generado se le denomina diagrama frontera.

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



1. **Diagramas de secuencia.**

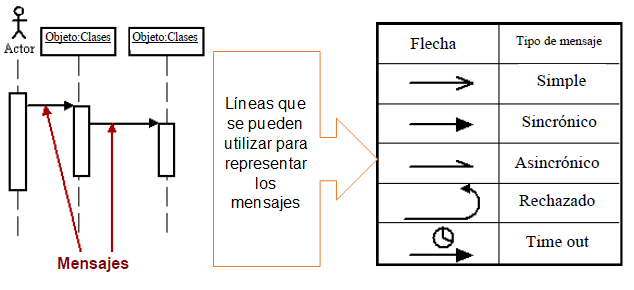
Los diagramas de secuencia se utilizan para formalizar la descripción de un escenario representando una ejecución real o instancia de un caso de uso. Contiene los mensajes que fluyen en el sistema así como quien los envía y quien los recibe a lo largo del tiempo.

Los objetos y actores que forman parte del escenario se representan mediante rectángulos distribuidos horizontalmente en la zona superior del diagrama. A cada uno de estos rectángulos se asocia una línea temporal vertical (línea de vida), de las que salen en orden los diferentes mensajes que se pasan entre ellos.

Algunos de los elementos son:

* *La Línea de Vida*: Representa la existencia de un objeto a lo largo de un período detiempo.
* El *Foco de Control* o Barra de activación (opcionales): representa el período de tiempo durante el cual un objeto ejecuta una acción.Se muestra como un rectángulo en la línea de vida.La parte superior se alinea con el comienzo de la acción.La parte inferior se alinea con su terminación. Puede tener un mensaje de retorno.
* Los *mensajes,* que significan la invocación de métodos, se representan como flechas horizontales que van de una línea de vida a otra, indicando con la flecha la dirección del mensaje. Los mensajes se dibujan desde el objeto que envía el mensaje al que lo recibe, pudiendo ser ambos el mismo objeto y su orden viene determinado por su posición vertical. Tipos de envío de mensajes:
  + simple: es la transferencia del control de un objeto a otro.
  + síncrono: es aquel en el que el objeto espera la respuesta antes de continuar con su trabajo.
  + asíncrono: es aquel en el que el objeto no espera la respuesta a ese mensaje antes de continuar con su trabajo.

Si hay un retorno de mensaje, se representa por una línea discontinua. Los mensajes pueden crear y destruir objetos.



* *Fragmentos:* Los fragmentos son aquellas secciones que cumplen determinada funcionalidad en un diagrama de secuencia y es necesario tener controlado a través de mecanismos que permiten agregar un grado de lógicas de procedimientos a los diagramas.Pueden representar algunas situaciones más complejas como bucles usando fragmentosdeinteracción**.**Normalmente se nombran con el tipo de bucle a ejecutar y la condición de parada. (condición que define la alternativa). Los llamados fragmentos combinados es una o más secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias específicas. Se puede utilizar fragmentos combinados para definir, entre otros, bucles (loop), bifurcaciones (alt) y procesamientos simultáneos (par).

