Entornos de Desarrollo
Tema 5. UML (I)

1º CFGS: Desarrollo de Aplicaciones Web
IES Severo Ochoa - Elche
2011/2012

Entornos de Desarrollo

Tema 5. UML (I)

por: Javier Martín Juan

Revisión: dee9dd525a0f

Última actualización: 27 de noviembre de 2011

Reconocimientos:

- Francisco Aldarias Raya - Plantillas LaTeX

Índice

1. Objetivos

- Tomar conciencia de la importancia de los procesos y actividades relacionadas con la ingeniería del software en proyectos de todo tipo.
- Describir las fases del ciclo de vida del software en distintas metodologías.
- Realizar diagramas de casos de uso en notación UML.
- Realizar diagramas de secuencia en notación UML.

2. Introducción a la ingeniería del software

Lectura previa: Capítulo 1 del libro "Ingeniería del Software - UOC". Poned atención al apartado 1.4 ("Conceptos"), y especialmente a los conceptos de:

- Análisis de requisitos
- Estimación de costes
- Diseño
- Documentación
- Pruebas de calidad
- Seguridad

También es importante el apartado 1.5.1 ("metodología eXtreme Programming"), pero os podéis saltar el apartado 1.5.2 ("Métrica v3").

3. Introducción a UML

El UML (Lenguaje Unificado de Modelado) es una familia de diagramas gráficos que ayudan a describir y diseñar sistemas de software, particularmente aquellos construidos con lenguajes orientados a objetos como Java o C++.

UML es un estándar relativamente abierto, controlado por el OMG (Object Management Group), un consorcio abierto de empresas. Nació en 1997 como una unificación de los diversos sistemas de modelado gráfico que existían hasta el momento.

Consta de los siguientes diagramas:

- Diagramas de estructura
  - ⋆ Diagrama de clases (UML 1)
  - ⋆ Diagrama de componentes (UML 1)
  - ⋆ Diagrama de composición de estructuras (UML 2)
  - ⋆ Diagrama de despliegue (UML 1)
  - ⋆ Diagrama de objetos (UML 1 no oficialmente)
  - ⋆ Diagrama de paquetes (UML 1 no oficialmente)
- Diagramas de comportamiento
  - ⋆ Diagrama de actividad (UML 1)
  - ⋆ Diagrama de casos de uso (UML 1)
  - ⋆ Diagrama de estados (UML 1)
  - ⋆ Diagramas de interacción
    - ○ Diagrama de secuencia (UML 1)
    - ○ Diagrama de comunicación (colaboración en UML 1)
    - ○ Diagrama de interacción (UML 2)
    - ○ Diagrama de temporización (UML 2)

La medida en que se utiliza UML depende del proyecto y las partes implicadas. Podemos usar desde unos pocos gráficos para comunicar la información más importante del sistema hasta completos diagramas de clases que nos permiten, mediante una herramienta CASE, generar esqueletos de código basado en el diseño.

En cualquier caso, siempre debe considerarse UML como una herramienta para facilitar la documentación y comunicación, y no entorpecerla. Generalmente los requisitos de un sistema cambian continuamente, así que no conviene perder el tiempo en documentar exhaustivamente cada detalle del mismo porque los diagramas se quedarán obsoletos en poco tiempo. Tampoco vale la pena elaborar multitud de diagramas en equipos pequeños si nadie los va a leer. Lo normal es que los diagramas UML surjan en reuniones de análisis, diseño o implementación y se desechen en seguida. Si algún diagrama aparece con mucha frecuencia, entonces es cuando debemos plantearnos capturarlo y añadirlo a la documentación.

4. Herramientas de diseño de diagramas

Existe una gran cantidad de herramientas de modelado UML, tanto privativas como libres y gratuitas. Nosotros utilizaremos ArgoUML, StarUML y Dia.

Otra herramienta de referencia es IBM Rational Modeler, creada por uno de los fundadores de UML.

ArgoUML

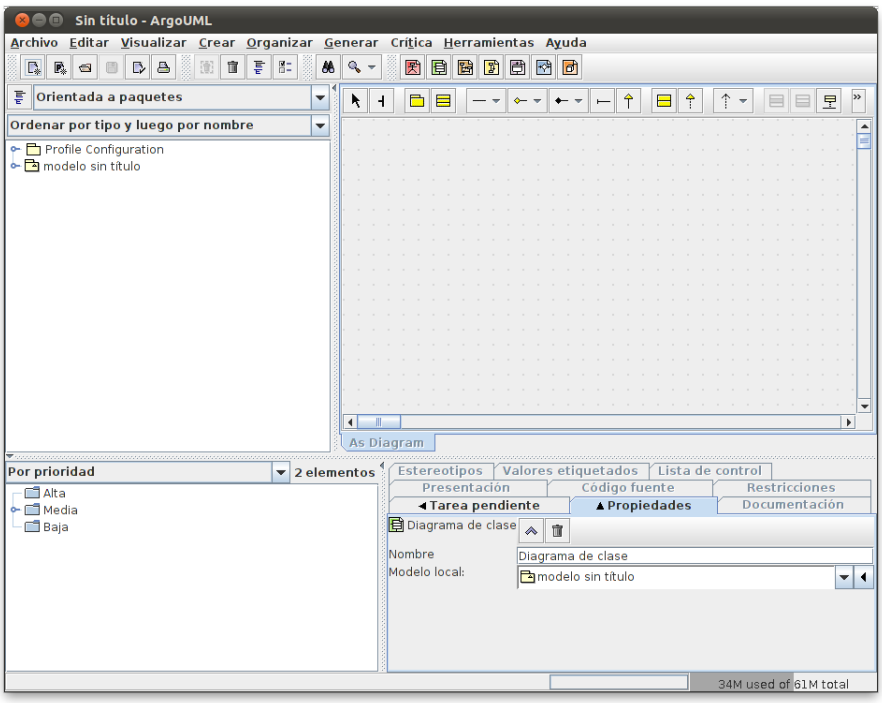
Para instalar ArgoUML vamos a `http://argouml.tigris.org` y descargamos la versión para nuestro sistema operativo.

Para ejecutar ArgoUML basta con descomprimirlo *(botón derecho → Extraer Aquí)*. Después, movemos la carpeta resultante (p.ej. *argouml-0.32.2*) al escritorio para tenerlo más a mano.

ArgoUML está escrito en Java, así que necesitamos el JRE para poder ejecutarlo. Dentro de la carpeta *argouml-0.32.2* hay 2 ejecutables:

- *argouml.sh* - Soporta UML 1.4
- *argouml2.sh* - Soporta UML 2 (experimental)

Para los ejercicios nos bastará con la versión 1.4 de UML, así que cargaremos el *script argouml.sh* haciendo doble clic en él, seguido de *Ejecutar*.



Dia

Para instalar Dia en Ubuntu:

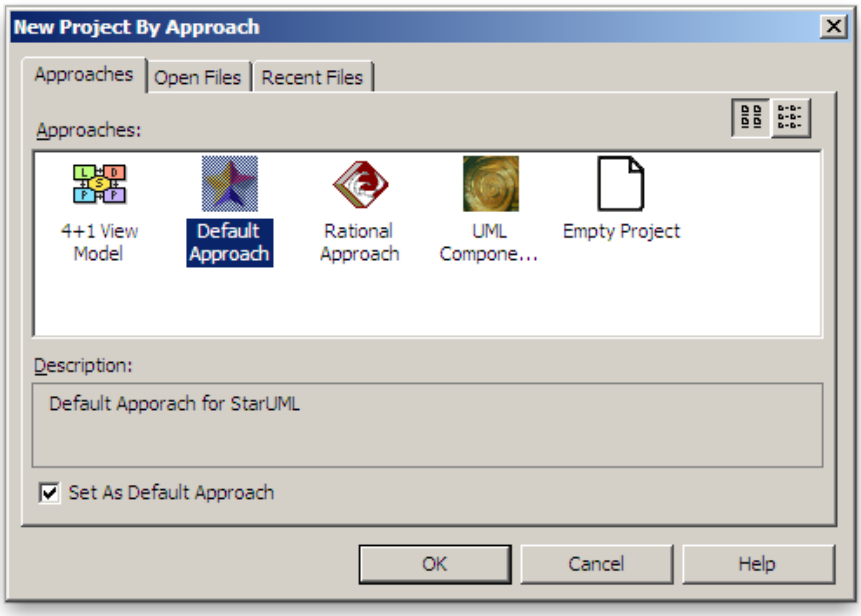
```
sudo apt-get update
sudo apt-get install dia
```

StarUML

StarUML está sólo para Windows, aunque es libre y gratuita. Se puede descargar de:

`http://staruml.sourceforge.net/en/`

Al arrancar, nos preguntará qué estilo de edición queremos. Lo dejamos en *Default approach*:



5. Los diagramas de casos de uso

Los diagramas de casos de uso son un tipo de diagrama de comportamiento que sirve para describir lo que debe hacer un sistema desde el punto de vista de quien lo va a utilizar.

Un modelo de casos de uso se construye mediante un proceso iterativo durante las reuniones entre los desarrolladores del sistema y los clientes (y/o los usuarios finales) conduciendo a una especificación de requisitos sobre la que todos coinciden.

Un caso de uso captura algunas de las acciones y comportamientos del sistema y cómo los actores interactúan con él.

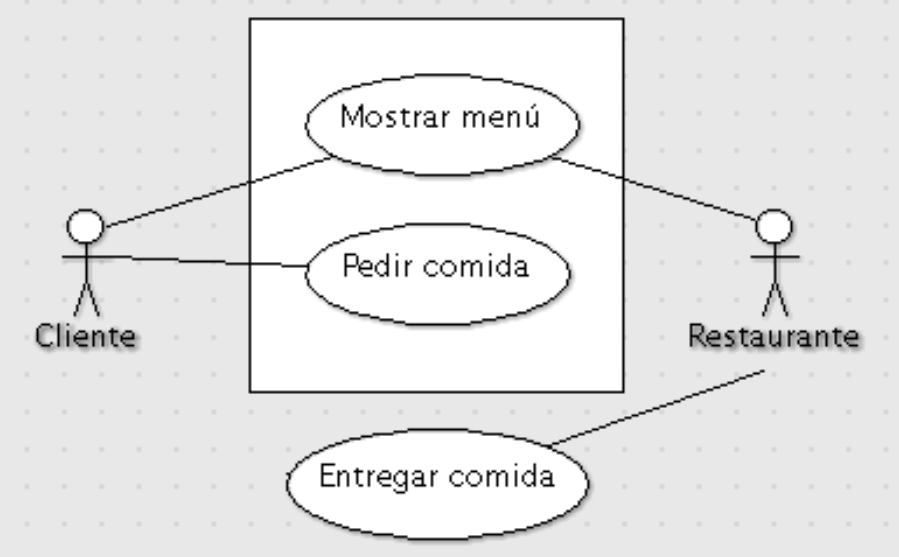
Elementos del diagrama

- *Actor*: Los actores son personas o procesos automáticos que necesitan interactuar con el sistema. Se deben identificar sus papeles en el sistema. En el diagrama, se representan del siguiente modo:

- *Caso de uso*: Los casos de uso se representan mediante elipses y corresponden a acciones generales del sistema. Una forma de reconocerlos es que suelen ser verbos en la descripción del caso de uso.

- *Asociación*: La interacción entre los actores y los casos de uso del sistema se representan por una línea recta que une a ambos.

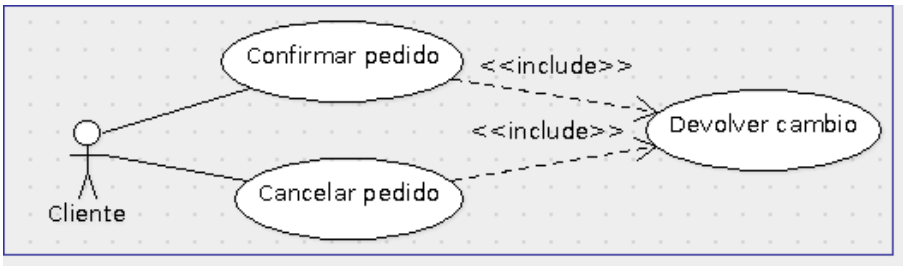
- *Sistema*: El sistema es el software que vamos a desarrollar. Puede ser un pequeño componente cuyos actores son otros componentes, o puede ser una aplicación completa. Se representa como una caja rectangular. Dentro de ella se incluyen los casos de uso soportados por el sistema.



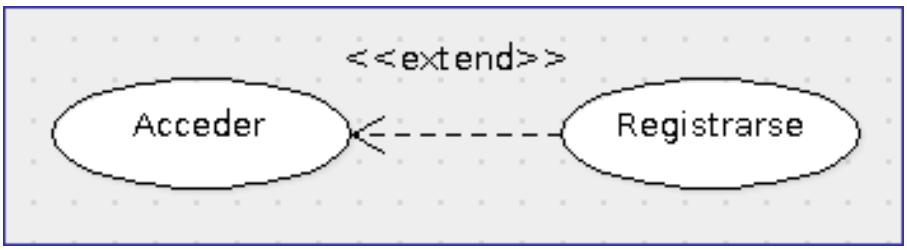
- *Inclusión*: se utiliza cuando el comportamiento de un caso de uso se incluye dentro del comportamiento de otro. Se representa con una flecha de trazo discontinuo desde el caso que incluye hasta el caso incluido, con la etiqueta *«incluye»* o *«usa»*

  Los casos de uso incluidos se pueden compartir, así evitamos repetirlos.

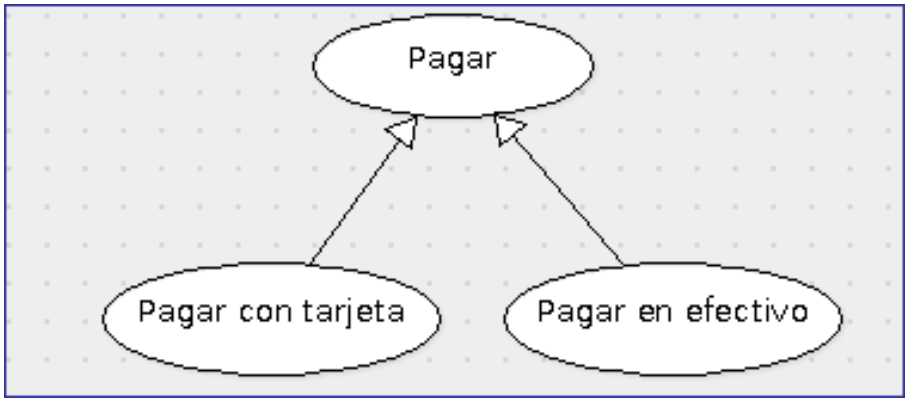
  También se pueden utilizar para estructurar el diagrama en varios niveles de detalle, pero no conviene abusar de ellos (ver recomendaciones más abajo).



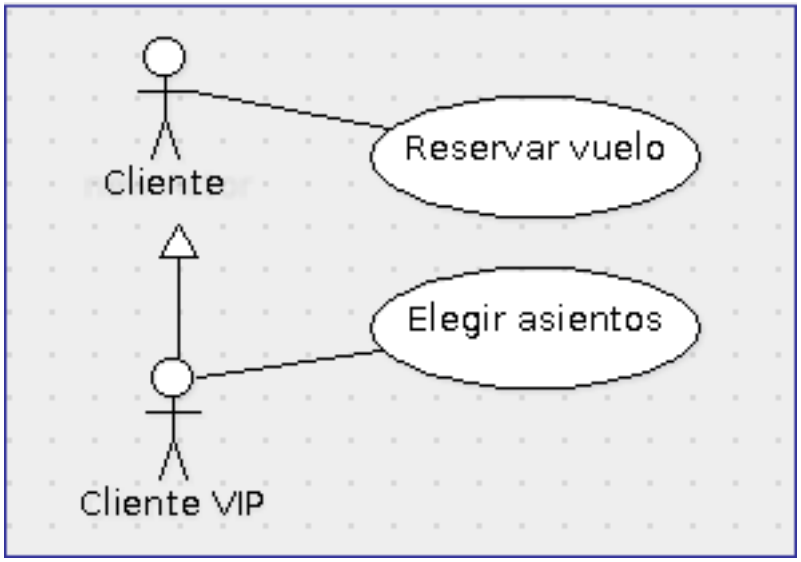
- *Extensión*: se utiliza cuando un caso hace lo mismo que otro, pero además aporta un comportamiento adicional en determinadas circunstancias o cuando se cumple cierta condición. Se representa con una flecha de trazo discontinuo que apunta al caso que queremos extender.



- *Generalización*: se utiliza para expresar que un caso de uso especializado es una forma particular de conseguir los objetivos de otro caso de uso más general. Se representa como una flecha continua que apunta al caso más general.



También se puede utilizar con actores:



¿Cómo distinguir las relaciones entre casos?

A veces no está del todo claro cuándo debemos usar la extensión, generalización o inclusión. En caso de duda, podemos guiarnos por las siguientes reglas:

- La *inclusión* equivale a "copiar y pegar" un caso de uso dentro de otro. Si imaginamos los casos de uso como líneas de un programa, sería equivalente a incluir el código de un caso dentro del otro.
- La *extensión* se utiliza cuando un caso añade funcionalidad a otro dependiendo de alguna condición. Si imaginamos los casos de uso como líneas de un programa, la relación "B extiende a A" equivaldría a introducir una condición *if* en algún punto de A, de forma que si se cumple la condición, se ejecuta B antes de proseguir con A.
- La *generalización* sirve para indicar que varios casos tienen el mismo comportamiento pero lo llevan a cabo de distinto modo. Equivale a la *herencia* en programación orientada a objetos, donde las clases hijas pueden rescribir parte del código de la clase padre.

Recomendaciones

Algunas recomendaciones a la hora de elaborar casos de uso:

- Los casos de uso describen las interacciones más importantes con el sistema, no su funcionamiento interno.
- Conviene mantener los casos de uso lo más simples posibles. Todos los implicados en el proyecto deben ser capaces de entender el diagrama de casos de uso. Si se complica demasiado, pierde su utilidad.
- Hay que tener en cuenta que el diagrama de casos de uso se elabora durante la fase de especificación de requisitos y que éstos son cambiantes por naturaleza. Por ello no conviene ahondar demasiado en detalles ni descomponer cada caso en sub-casos, ya que si éstos cambian habremos malgastado el tiempo.
- En cualquier caso, es preferible acompañar al diagrama de una buena descripción narrativa del caso de uso.

Herramientas

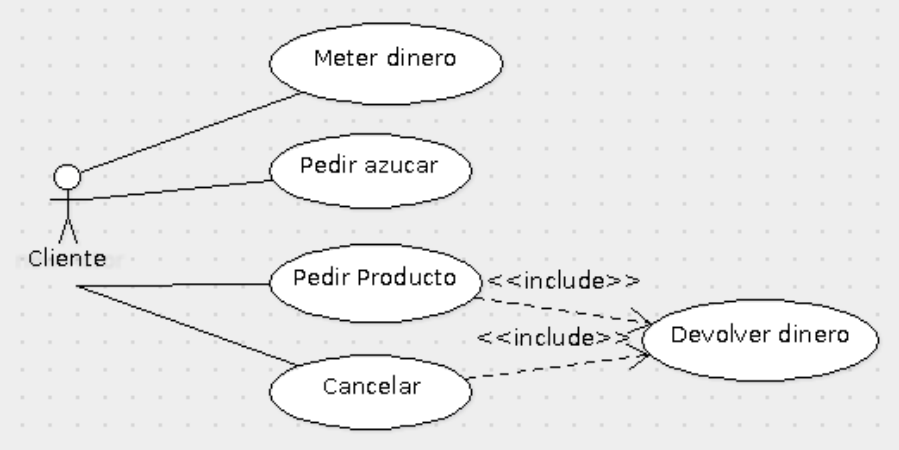
Para realizar diagramas con ArgoUML debemos ir al menú *Crear* → *Diagrama de casos de uso*. Después debemos de ponerle nombre al diagrama y sobre la zona de dibujo podemos empezar a añadir componentes.

Ejemplo: La máquina de café

Supongamos que se requiere desarrollar el control de una máquina de entrega de café automática.

La máquina debe permitir a una persona introducir dinero, escoger uno de los productos de acuerdo a su precio, escoger un nivel de azucar y entregar el producto y las vueltas.

El usuario puede en cualquier momento antes de escoger el azúcar cancelar la operación, mediante un botón existente para este objetivo.



Ejemplo: Tienda en Internet

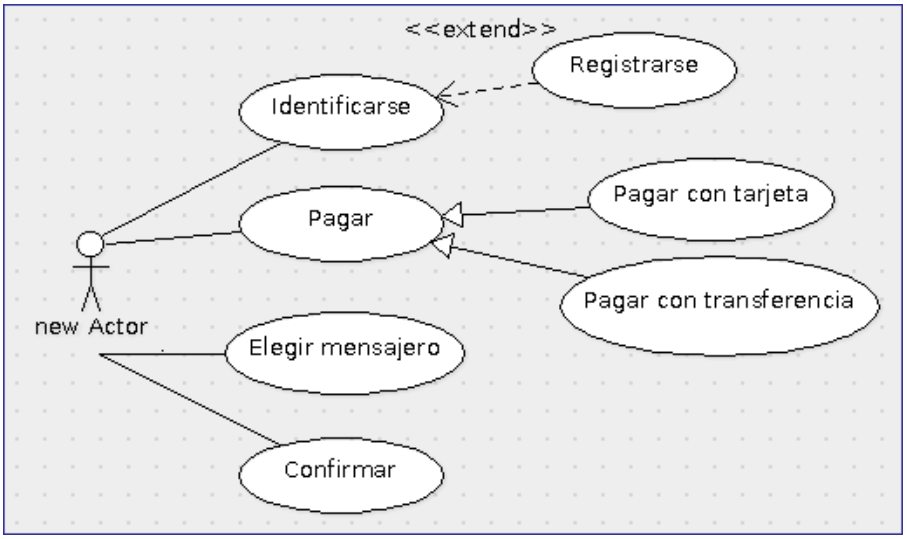
Queremos modelar el sistema de pago en una tienda web.

El cliente debe identificarse mediante su dirección de correo. Si es un nuevo cliente se le debe registrar en el sistema previamente, pidiéndole los datos personales.

Una vez identificado al cliente, éste podrá elegir el medio de pago: por transferencia bancaria o con tarjeta de crédito. Según el medio de pago se le solicitarán unos datos u otros.

El cliente también deberá elegir el método de envío.

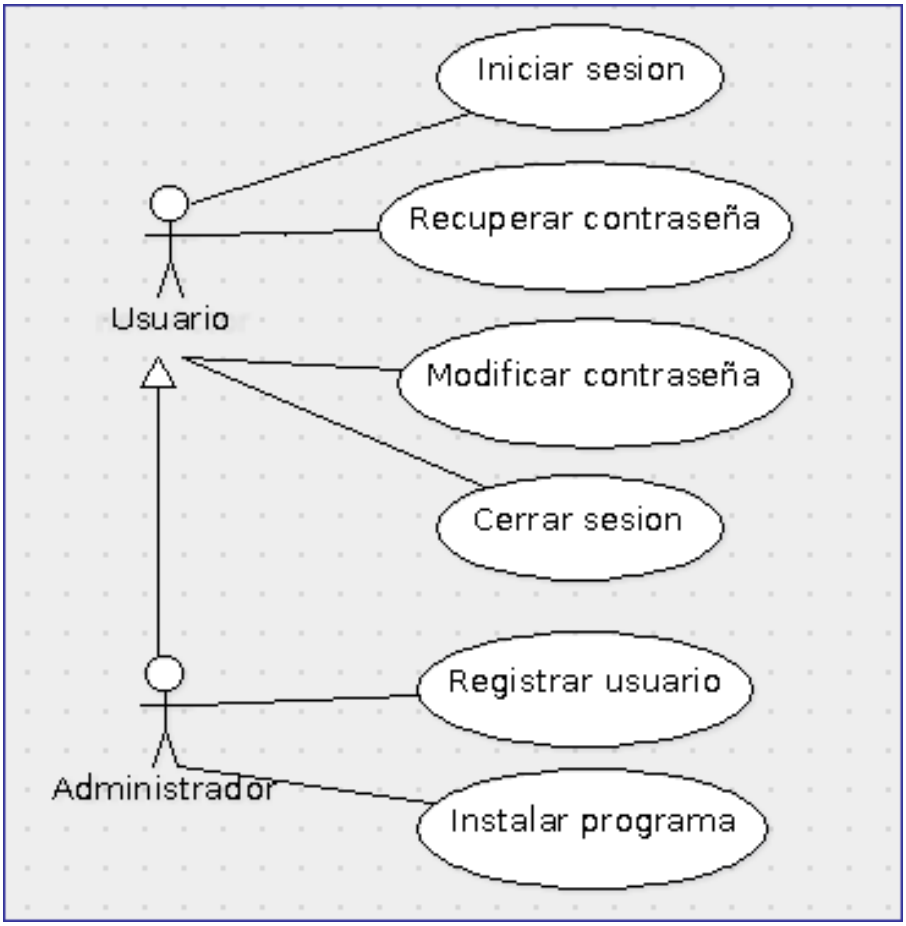
Finalmente se le mostrarán todos los datos del pedido para pedirle que confirme.



Ejemplo: Usuarios y administradores

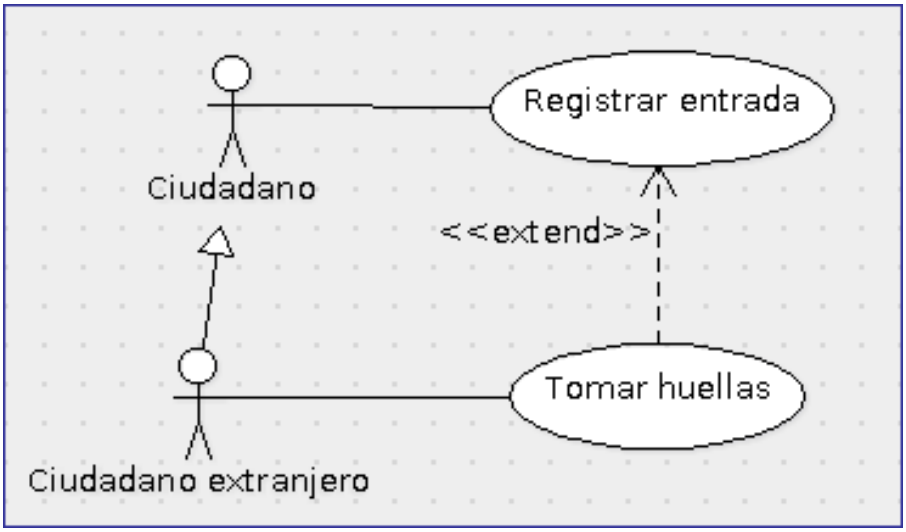
Queremos modelar un sistema en el que hay usuarios. Los usuarios pueden iniciar sesión, modificar su contraseña, recuperar su contraseña y cerrar sesión.

Los administradores tienen los mismos permisos que los usuarios, pero además, pueden registrar usuarios e instalar programas.



Ejemplo: Puesto fronterizo

En la frontera de un país se registran todos los ciudadanos que entran. Además, en caso de que el ciudadano sea extranjero, se le toma la huella dactilar.



6. Diagramas de secuencia

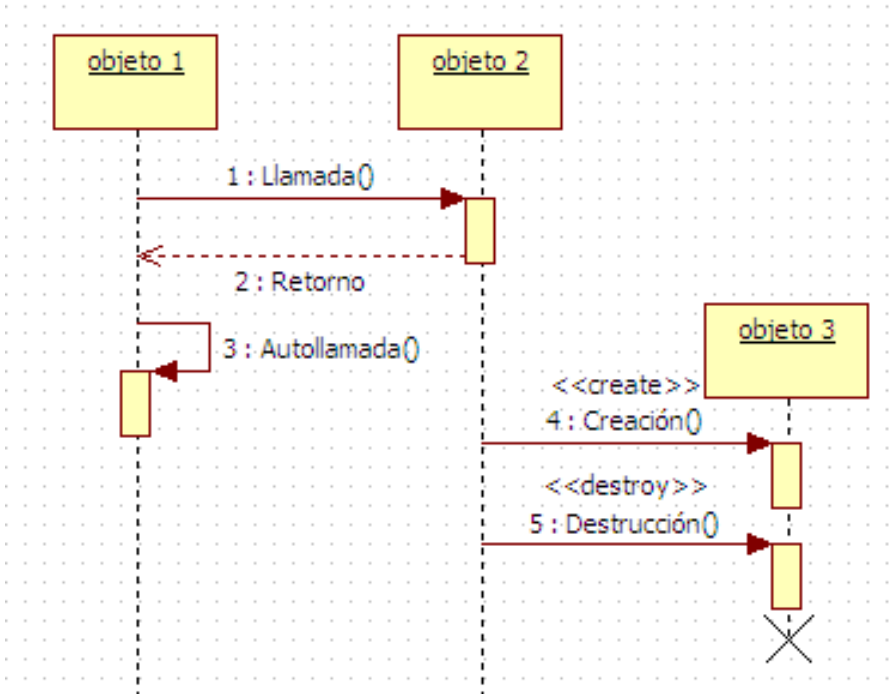
Los diagramas de secuencia son diagramas de comportamiento que muestran muestran la visión temporal de la secuencia de un caso de uso.

Frente a los diagramas de caso de uso, un diagrama de secuencia destaca la ordenación temporal de los mensajes. Es un gráfico bidimensional donde el eje vertical representa el tiempo y el eje horizontal muestra los objetos de la colaboración.

Elementos del diagrama de secuencia

- *Objetos*: son elementos que participan en el diagrama. Y son instancias de una clase. Se representan por un rectángulo con el nombre del objeto escrito del siguiente modo: "Objeto : Clase".
- *Línea de vida*: indica la existencia de un objeto (desde que se crea hasta que se destruye) mediante una línea discontinua. El fin de la vida se expresa como un aspa.
- *Activación*: indica cuándo el objeto está realizando una tarea concreta. Equivale al tiempo durante el cual se está ejecutando el método o función. Se expresa como un rectángulo a lo largo de la línea de vida.
- *Mensaje*: la comunicación entre objetos y activaciones. Los mensajes pueden ser síncronos, asíncronos, de creación, de destrucción, de llamada o de retorno. En este tema únicamente nos preocuparemos de los mensajes de llamada y retorno. Para más información, consultar la bibliografía, en especial el Manual de Referencia de UML 2.

En la siguiente figura se pueden ver los elementos más habituales en un diagrama de secuencia:



Recomendaciones

- En general, un diagrama de secuencia sólo representa secuencias de mensajes y no intervalos de tiempo precisos. Si lo que queremos es representar tiempos, deberíamos pensar en usar un diagrama de temporización.
- Hay que **sintetizar**. Es un error intentar incluir todos los métodos de todos los participantes en el diagrama. Tampoco conviene llenar el diagrama de objetos y de mensajes. Cuanto más complicado sea el diagrama, menos probabilidad hay de que alguien lo lea.

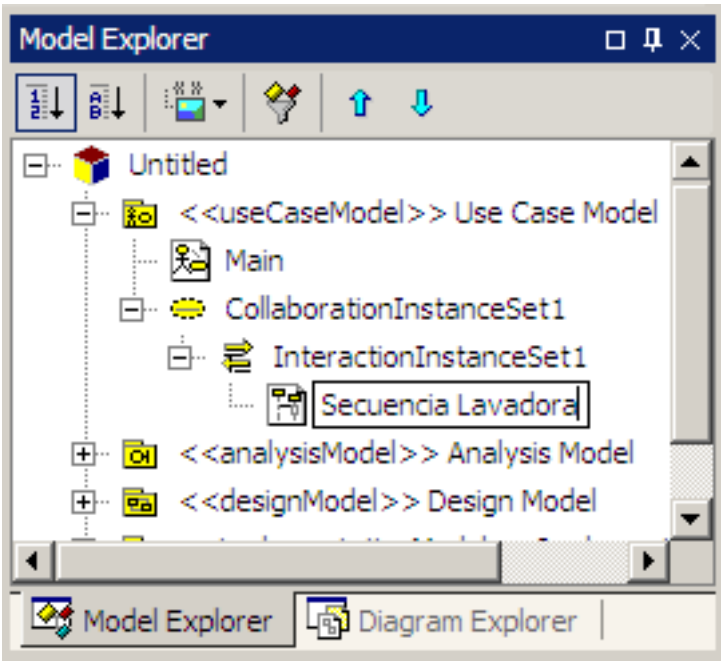
Herramientas

Para crear el diagrama con StarUML:

1.- En la parte derecha, seleccionamos la pestaña *Model Explorer*.

2.- Pinchamos con el botón derecho en *«useCaseModel»* → *Add Diagram* → *Sequence Diagram*

3.- Le damos un nombre al diagrama, por ejemplo: *Secuencia Lavadora*

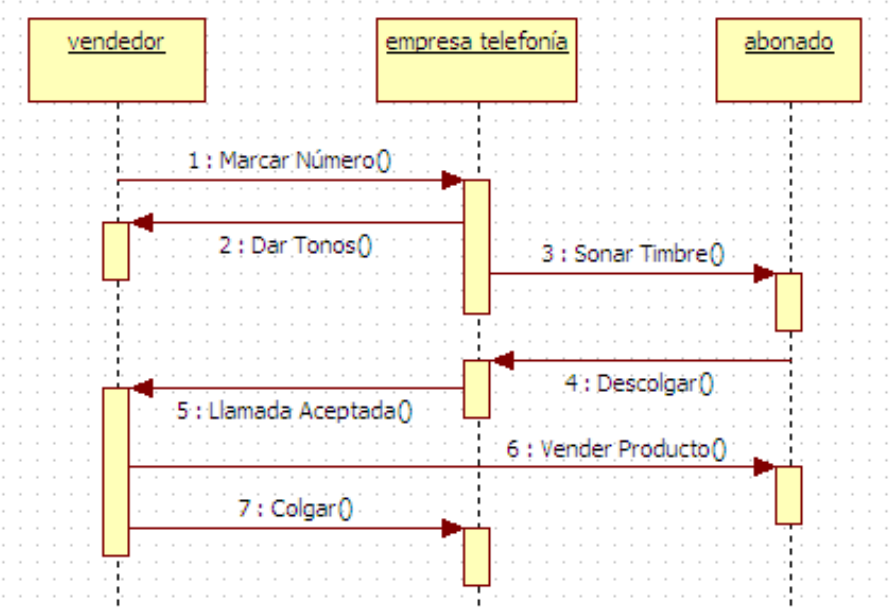


4.- A la izquierda, en *Toolbox*, seleccionamos *Sequence*.

5.- Ya podemos empezar a poner elementos en el diagrama.

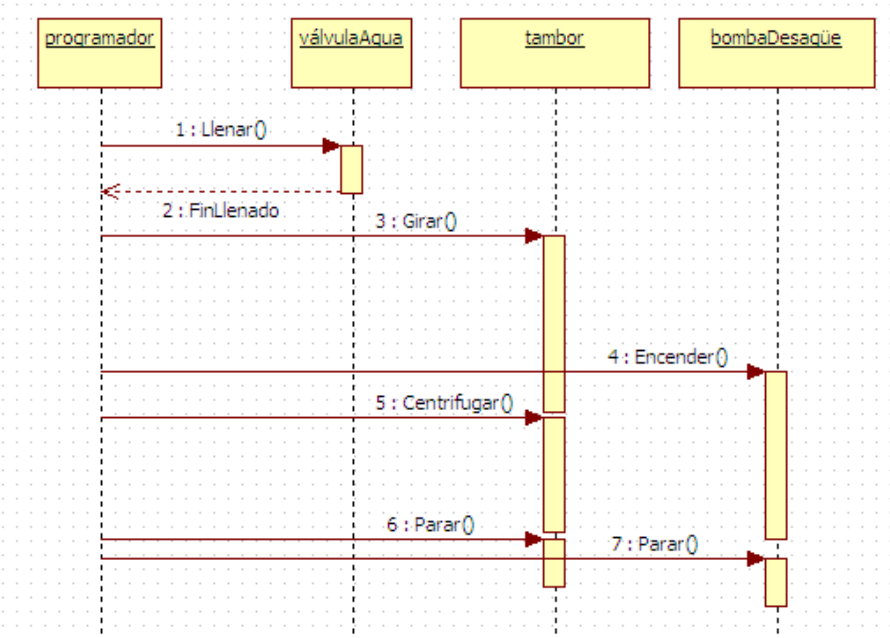
Ejemplo: llamada telefónica

El siguiente ejemplo modela un sistema de propaganda telefónica. Consiste en una máquina que va llamando a números de teléfono. Cuando el abonado descuelga, una voz robotizada lee la propaganda y, después, cuelga.



Ejemplo: lavadora

El siguiente diagrama de secuencia modela el comportamiento de una lavadora.



7. Ejercicios propuestos

Diagramas de casos de uso

Realiza el diagrama de casos de uso en los siguientes supuestos:

Ejercicio 5.1

Una aplicación de móvil permite hacer llamadas de teléfono, videoconferencia y escribir mensajes cortos. El programa también cuenta con una agenda de contactos. Cuando el usuario quiere hacer una de las tres operaciones, primero se le muestra una agenda para que elija el contacto.

Ejercicio 5.2

Un cliente de una entidad bancaria puede realizar las siguientes operaciones on-line: transferencias y operaciones en bolsa. En caso de transferencias de más de 9.000 euros, se notificará a un agente para que éste verifique la operación.

Ejercicio 5.3

Un usuario de un curso on-line debe de poder registrarse a un curso, ver los cursos en los que está matriculado, realizar la configuración personalizada de su entorno, consultar uno de los cursos en que está matriculado, publicar en el foro, descargar material de la web y entregar resultados, que unas veces pueden ser subir archivos y otras rellenar en una página web de la propia plataforma.

Diagramas de secuencia

Modelar mediante diagramas de secuencia los siguientes supuestos:

Ejercicio 5.4

Un usuario entra a una página por Internet tecleando la web en el navegador. El navegador hace la petición al servidor. El servidor accede a la base de datos, y finalmente la base de datos accede al disco. El disco devuelve a la base de datos los ficheros solicitados. Ésta, a su vez, devuelve al servidor las tablas con los datos. El servidor entrega al navegador código HTML, y finalmente el navegador muestra la página al usuario.

Ejercicio 5.5

Representa un diagrama que muestre la gestión de compras de entradas a través de un cajero automático. En el sistema el comprador introduce la tarjeta en el cajero, introduce la fecha para el espectáculo y el sistema le devuelve los asientos disponibles. Una vez seleccionados los asientos el sistema recoge la petición para pasarla al sistema de control de entradas y éste a su vez a la entidad de crédito.

Ejercicio 5.6

Sistema de pedidos por Internet para un restaurante: el proceso comienza con el cliente consultando el menú a través de la web. Después elige los platos e introduce sus datos de pago.

A continuación, el sistema contacta con el banco para procesar el pago. Una vez verificado, pasa la comanda a la cocina. La cocina informará de cuánto tardará el pedido en función del número de comandas acumuladas.

Finalmente se le confirma al cliente el pedido y se le informa de cuándo podrá pasar a recogerlo.

8. Bibliografía y documentación adicional

- Ingeniería del Software - UOC

  `http://ocw.uoc.edu/informatica-tecnologia-y-multimedia/ingenieria-del-software-en-ento materiales/`

- UML Reference Manual, 2nd Edition

  Booch, Jacobson, Rumbaugh, Ed. Addison-Wesley

- UML for Java Programmers

  Robert C. Martin, Ed. Prentice Hall

- Learning UML 2.0

  Miles, Hamilton, Ed. O'Reilly

- UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition)

  Martin Fowler, Ed. Addison-Wesley

- MSDN (Microsoft): UML Use Case Diagrams: Guidelines

  `http://msdn.microsoft.com/en-us/library/dd409432.aspx`

- Aprendiendo UML en 24 horas

  Joseph Schmuller, Ed. Prentice Hall

- UML 2.0 - Pocket Reference

  Dan Pilone, Ed. O'Reilly

- Análisis y Diseño Estructurado y Orientado a Objetos de Sistemas Informáticos

  Amescua et al, Ed. McGraw-Hill

- Diseño Orientado a Objetos con UML

  Raúl Alarcón, Grupo Eidos

- Plantillas ReadySET para documentación de proyectos

  `http://readyset.tigris.org/index.html`