C.F.G.S. DESARROLLO DE APLICACIONES INFORMÁTICAS

MÓDULO: Entornos de Desarrollo

Unidad 4

Elaboración de diagramas de comportamiento

ÍNDICE DE CONTENIDOS

OBJETIVOS

El objetivo general de esta unidad es conseguir que el alumno conozca los diagramas que componen UML, para tener una visión clara sobre todos ellos. El orden en que estos diagramas se abordan esta decidido para que el alumno vaya desde una idea más general a una idea más profunda y concreta.

1.- Diagrama de Casos de Uso.

Los diagramas de casos de uso se agrupan dentro de la clasificación de diagramas de comportamiento de UML ya que **realizan un estudio de que hace el sistema desde el punto de vista del usuario.**

Pero **¿qué son los casos de uso?**

Los **casos de uso** no son más que el estudio de los usos que se le va a dar a algo. Enfocado al mundo del software, **serían un estudio de los requisitos que queremos que nuestra aplicación cumpla, entendiendo por requisitos las funciones que va a tener nuestra aplicación una vez desarrollada.**

Para entender esto un poco mejor imaginemos que nos vamos a comprar un automóvil. Bien, lo más lógico es que cuando lo hagamos pensemos en **funciones** que necesitamos que tenga ese automóvil, de manera que nuestra compra satisfaga nuestras necesidades. De esta forma nos preguntaremos cosas como las siguientes:

- ¿Lo usaremos para 2 ó para más personas?
- ¿Llevar a los niños al colegio?
- ¿Ir a trabajar?
- ¿Ir de excursión?
- ¿Qué pensamos cargar en el maletero?
- Cargar materiales para la obra?
- ¿Lo usaremos en viajes largos o cortos?
- ¿Necesitamos que sea rápido?.
- ¿Necesitamos que sea potente?.
- ¿Queremos gasolina o diesel?
- Etc..

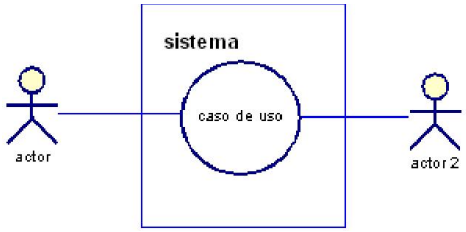
Con esto realizamos un estudio de las funciones que queremos que nuestro automóvil tenga, desechando aquéllas que no utilizaremos o bien que no nos parezcan interesantes. Esto sería realizar un estudio de casos de uso del automóvil que vamos a adquirir. Siguiendo dicho estudio la compra de nuestro futuro automóvil debería ser la más acertada.

Como se ha podido ver, los **casos de uso no se enfocan hacia cómo se van a hacer** realidad nuestras necesidades sino **hacia cuáles son, es decir,** no se interesan por la implementación (cómo se hace dicha función) sino **en las funciones que se necesitan**.
Puesto que los casos de uso **se utilizan para capturar los requisitos, los diagramas de caso de uso serán empleados durante la fase de análisis del sistema.**

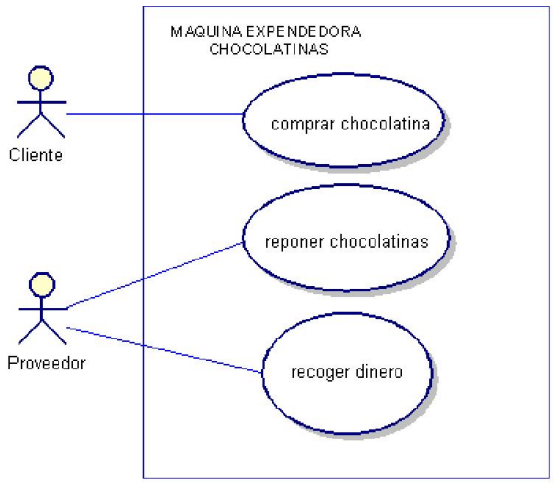
En los diagramas de casos de uso encontraremos:

- **Actores,** se representan mediante un muñeco y son los que realizan o demandan la tarea (el uso).
- **Casos de uso y** representan las funciones (usos).
- **Relaciones de dependencia, generalización y asociación entre ellos**. Se representan con líneas que conectan los casos de uso o actores.

Generalmente los actores permanecen fuera del sistema y los casos de uso están contenidos en él. Para representar cuáles son los límites del sistema se utiliza un rectángulo.



Imaginemos que queremos **implementar una máquina expendedora de chocolatinas**. En este sistema encontramos gente que quiere comprar una chocolatina, quien recarga la máquina de chocolatinas y quien retira el dinero de la máquina. Una solución de este ejemplo sería el siguiente modelo de casos de uso.



Este ejemplo se irá desarrollando para su total compresión en puntos sucesivos.

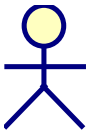
1.1.- Actores.

En el diagrama de casos de uso, los usuarios o elementos que se relacionan o usan el sistema se llaman **actores** y se clasifican en:

- **Actores Principales**: Que serían las personas que usan el sistema.

- **Actores Secundarios**: Serían las personas que mantienen o administran el sistema.
- **Actores no humanos**: Dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados. Por ejemplo el reloj del sistema podría modelarse como un actor que, a cierta hora, lanzaría el caso de uso "realizar backup".
- **Otros sistemas**: Sistemas con los que el sistema interactúa.

Los actores se representan por la figura:



Respecto de los actores debemos tener en cuenta lo siguiente:

- **Una misma persona física puede interpretar varios papeles como actores distintos.** En el ejemplo que estamos desarrollando, una misma persona se encarga de reponer la máquina de chocolatinas y de recoger el dinero. Bien, si distinguiésemos entre "Proveedor" y "Recaudador" estaríamos diferenciando dos actores que podrían ser por ejemplo el dueño de una tienda (la misma persona física).

- **El nombre del actor describe el papel desempeñado.** Así a la persona que recarga la máquina de chocolatinas se le da el nombre de "Proveedor" y a la que compra chocolatinas de "Cliente".

Los actores se irán determinando conforme se vaya entrevistando a los diferentes usuarios que usarán el sistema. Sobre el **ejemplo** anteriormente introducido, el de la máquina expendedora de chocolatinas por un lado tendremos los clientes que serán los usuarios que buscan una chocolatina y por otro tendremos los que reponen las chocolatinas en la máquina expendedora y retiran el dinero. En este caso nos encontramos con dos actores, el **"Cliente"** y el **"Proveedor**". También se podría haber dado el caso de que hubiese una persona para reponer y otra para retirar el dinero de forma que en vez de dos actores estaríamos hablando de tres, **"Cliente"**, **"Proveedor"** y **"Recaudador"**.

1.2.- Caso de Uso.

Los casos de uso son operaciones o tareas específicas que se realizan tras un estímulo de un agente externo, que puede ser un actor e incluso otro caso de uso. Los casos de uso los vamos a representar como elipses en cuyo interior indicaremos la tarea o escenario que representa.

Cada caso de uso es una colección de escenarios y cada escenario es una secuencia de pasos.

Como se podrá observar más adelante en esta unidad, en el punto llamado "**construcción de casos de uso**", esta secuencia de pasos no aparece en el diagrama, sino que para cada caso de uso se creará un documento independiente con la información que debe incluir.

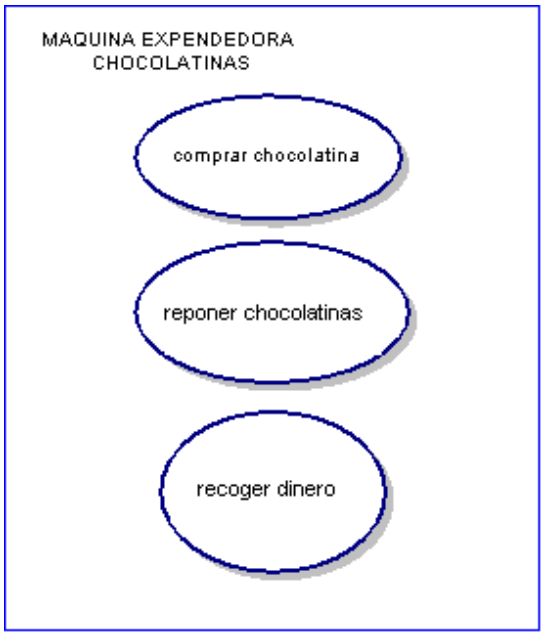
El nombre de los casos de uso debe ser significativo, alusivo a la tarea que realiza. Así si la función que desempeña un caso, por ejemplo, es la de recoger dinero le daremos el nombre de "recoger dinero".

Los casos de uso describen tareas que tienen sentido completo para el usuario, como por ejemplo, "comprar chocolatina".

Los casos de uso serán los escenarios que describirán las personas que usan el sistema (Actores), así que dependiendo de la persona que use el sistema aparecerá uno u otro escenario. Por esto es importante entrevistarse con todas las personas que van a manejar el sistema.

En el **ejemplo** que estamos desarrollando se puede deducir que el objetivo principal de nuestra máquina expendedora de chocolatinas será "**comprar una chocolatina**". Este caso de uso será el que se deduzca desde el punto de vista del **cliente**. Desde el punto de vista del **proveedor** deduciremos los casos de uso "**reponer chocolatinas**" y "**recoger dinero**" ya que serán las funciones que realice este actor en el sistema.

Se dice entonces que estamos ante tres escenarios distintos, los de "**Comprar chocolatina**", "**Recoger dinero**" y "**Reponer chocolatinas**".

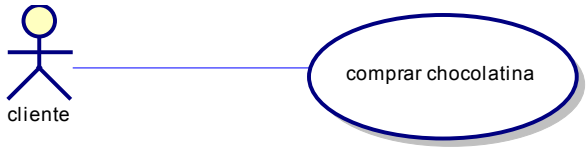


1.3.- Relaciones.

Recordarás que anteriormente dijimos que las **relaciones** en los diagramas de casos de uso están representadas mediante líneas que conectan a actores y casos de uso o a casos de uso entre ellos.

Dependiendo del tipo de relación que se trate la representación gráfica de la línea que une será de una u otra forma como veremos a continuación.

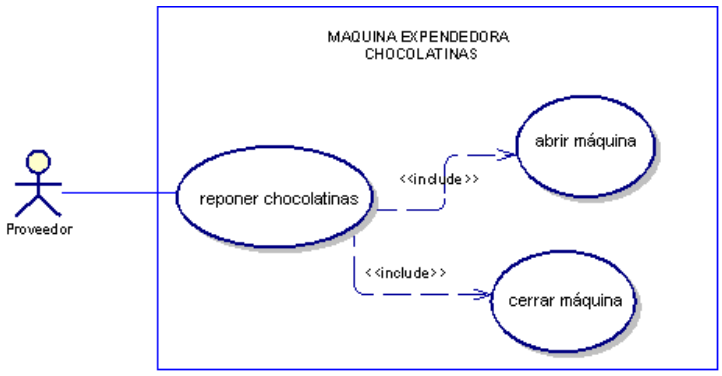
- **Asociación**: Con este tipo de relación se indica la invocación por parte de **un Actor a un caso de uso**. La relación de asociación la representamos **mediante una línea sólida que va del actor al caso de uso.**



- **Inclusión:** Este tipo de relación se utiliza para indicar que **un caso de uso incluye los pasos que se dan dentro de otro caso de uso,** es decir, **la ejecución de un caso de uso involucra que se ejecute también otro caso de uso**.

  Este tipo de relación aporta la ventaja de que los pasos que serían comunes a varios casos de uso se pueden concentrar en otro caso de uso y éste ser utilizado por los demás.

  Basándonos en nuestro ejemplo, tanto a la hora de "Reponer Chocolatinas" como de "Recoger dinero" será necesario "Abrir la máquina de chocolatinas" y "Cerrar la máquina de chocolatinas". Para no estar repitiendo estos pasos que son comunes, crearemos los casos de uso "abrir la máquina de chocolatinas" y "Cerrar la máquina de chocolatinas" y los incluiremos desde "Reponer chocolatinas" y "Recoger dinero".



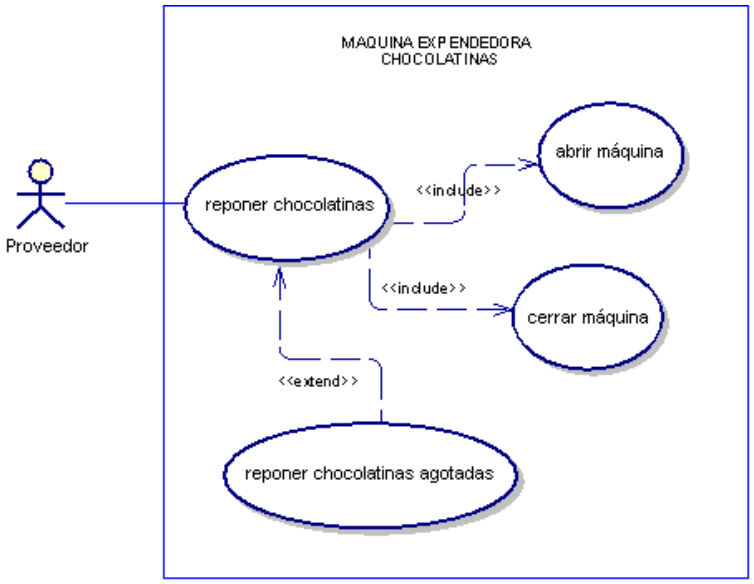
  La relación de inclusión se representa mediante **una flecha discontinua con una punta de flecha que apunta a la clase dependiente**.

- **Extensión**: Con la relación de extensión indicamos **un conjunto de operaciones que se pueden realizar o no dependiendo de unas condiciones**. Basándonos en nuestro ejemplo, a la hora de reponer la máquina de chocolatinas podemos hacerlo según las ventas que se hayan hecho, es decir, repondremos sólo aquellas chocolatinas que se hayan agotado (se realizará si se cumple esto). De esta forma podemos crear otro caso de uso que sea "Reponer chocolatinas Agotadas" el cual lo relacionaremos con el caso de uso base mediante una relación de extensión. Se dice que el nuevo caso de uso **(Reponer chocolatinas agotadas)** extiende al primero ya que incluye sus mismos pasos además de otros nuevos.

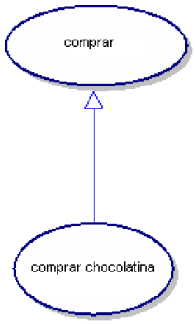
La relación de inclusión se representa mediante **una flecha discontinua con una punta de flecha que apunta al caso de uso incluido**. Sobre la flecha se pondrá el texto “extiende” o “extend”.

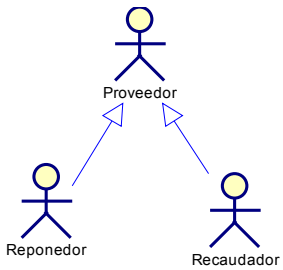
La extensión se realiza en un punto concreto dentro del caso de uso base. **A este punto se le conoce como punto de extensión**. En nuestro ejemplo si seguimos la secuencia de pasos tendremos:

- Queremos reponer las chocolatinas,
- Abriremos la máquina y
- Una vez abierta es cuando se realiza la extensión “Reponer chocolatinas agotadas”.
- El punto de extensión aquí lo podríamos llamar “Reponer compartimentos”.
- El nombre del punto de extensión también se refleja en la relación de extensión.



- **Generalización (Herencia):** Al igual que ocurría con las clases en que unas podían heredar de otras, podemos encontrar **la relación de herencia en los casos de uso**.
  Esto se puede dar entre casos de uso y entre Actores. Cuando ocurre una relación de herencia un caso de uso hereda de otro sus operaciones, a las que añade las suyas propias.





La relación de generalización se representa **mediante una línea continua y una punta de flecha. La punta de flecha indica el caso de uso del que hereda.**

1.4.- Construcción de casos de uso.

Una vez que hemos estudiado los elementos que forman parte de los diagramas de casos de uso nos queda quizás el punto más importante, ¿cómo construimos este **diagrama**? Bien vamos a intentar dar respuesta a dicha pregunta. A la hora de construir un diagrama de casos de uso debemos tener en cuenta los siguientes aspectos:

- **Un caso de uso debe ser siempre entendible, claro y conciso**.
- **Normalmente hay pocos actores asociados a cada caso de uso**.
- Para **determinar cuáles son los casos de uso** podemos realizarnos las siguientes preguntas:
  - ¿Qué tareas son las que desempeña un actor?.
  - ¿Qué información crea, guarda, modifica, destruye o lee el actor?.
  - ¿Debe el actor notificar al sistema cambios externos?.
  - ¿Debe el sistema informar al actor de los cambios internos?.

Una vez que hemos determinado los casos de uso que aparecen en nuestro sistema debemos recoger para cada uno de ellos la siguiente información.

- **El Inicio**: Cuándo y qué actor lo produce.
- **El fin**: Cuándo se termina el caso de uso y qué valor devuelve.
- **La interacción actor-caso de uso**: Es decir qué mensajes intercambian ambos.
- **Objetivo del caso de uso**: ¿Qué lleva a cabo o qué intenta?
- **Cronología y origen de las interacciones**.
- **Repeticiones de comportamiento**: ¿Qué operaciones son repetidas?
- **Situaciones opcionales**: ¿Qué ejecuciones alternativas se presentan en el caso de uso?

Esta información se puede representar en forma de plantilla. A modo de ejemplo de plantilla podemos considerar la siguiente:

Características: Caso de uso _____		
Objetivo		
Pre-Condiciones		
Post-Condiciones		
Actores Principales		
Otros Actores		
Curso Normal (Escenario normal)		
Paso	**Actor**	**Descripción**
Cursos Alternativos (Extensiones del Curso Normal)		

Paso	variante	Descripción
Otras características		
Importancia		Sin importancia/muy importante
Urgencia		Puede esperar/hay urgencia/inmediatamente
comentarios		

2.- Diagrama de Secuencia.

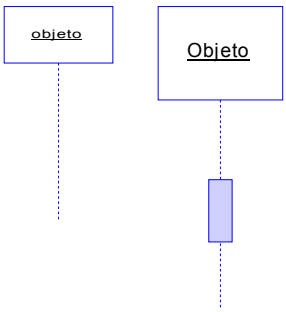
Una vez que conocemos los **diagramas de casos** de uso que nos ayudan a determinar qué interacciones vamos a realizar con nuestro sistema nos planteamos la siguiente pregunta ¿Cómo suceden dichas **interacciones** a lo largo del tiempo?.
Los **diagramas de secuencia** nos van ayudar a obtener tal información.

Los diagramas de secuencia **muestran el orden cronológico en el que ocurren las interacciones entre objetos**. Además, estos diagramas aportan detalles sobre la implementación del sistema ya que incluyen los objetos y clases que usa el sistema así como los mensajes que se pasan entre objetos.
Los diagramas de secuencia constan de los siguientes elementos:

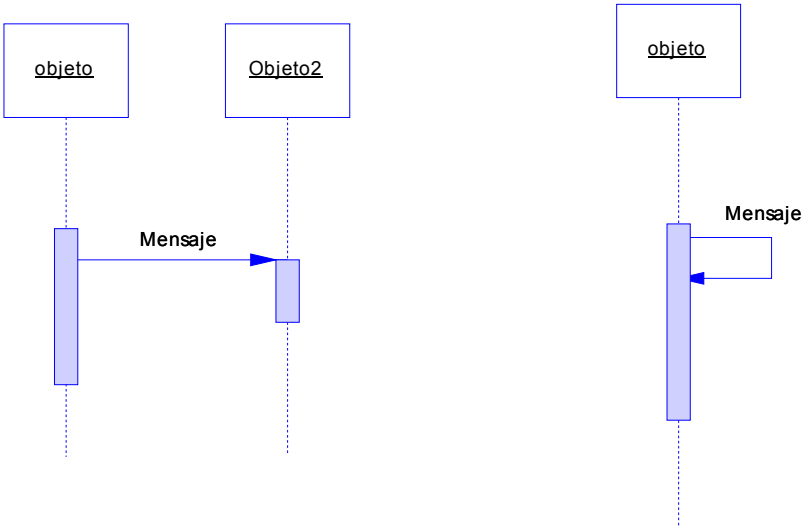
- **Objetos.**
- **Mensajes.**
- **El tiempo**.

Veamos con un poco más de detalle cada uno de ellos.

- **Objetos**: Los objetos son representados en los diagramas de secuencia mediante **un rectángulo** en cuyo centro aparece subrayado su nombre.
  Asociado a cada objeto aparece una **línea discontinua vertical** que representa el tiempo de vida del objeto.
  Sobre la línea vertical que representa el tiempo de vida del objeto representaremos mediante un **rectángulo**, la activación o ejecución de métodos del objeto. El tamaño del rectángulo representará la duración de ejecución del método en cuestión.

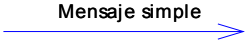


- **Mensajes**: los mensajes se pueden dar **entre objetos o entre un mismo objeto.** Los mensajes pueden ir sin parámetros o con ellos. En la primera figura se puede observar cómo se pasa un mensaje de un objeto a otro. El mensaje ocurre como consecuencia de la ejecución de un método del objeto 1 y tiene como resultado una llamada a otro método en el objeto 2.

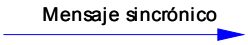


Los mensajes pueden ser de tres tipos**:**

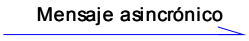
  - **Simples:** se da el control de un objeto a otro.



  - **Sincrónicos:** el objeto que envía el mensaje esperará la respuesta del mensaje antes de continuar con su trabajo.



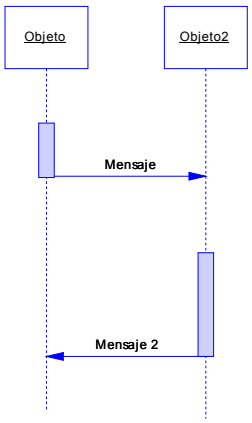
- **Asincrónicos:** son aquellos en los que el objeto no esperará la respuesta del mensaje para seguir su ejecución.



Normalmente un mensaje desemboca en la llamada de un método del objeto.

- **Tiempo:** En el diagrama de secuencia el tiempo **se representa en dirección vertical**. De esta forma un mensaje situado más cerca de la parte superior ocurrirá antes que otro situado cerca de la parte inferior.
  Podemos decir que **los diagramas de secuencia tienen dos dimensiones:**

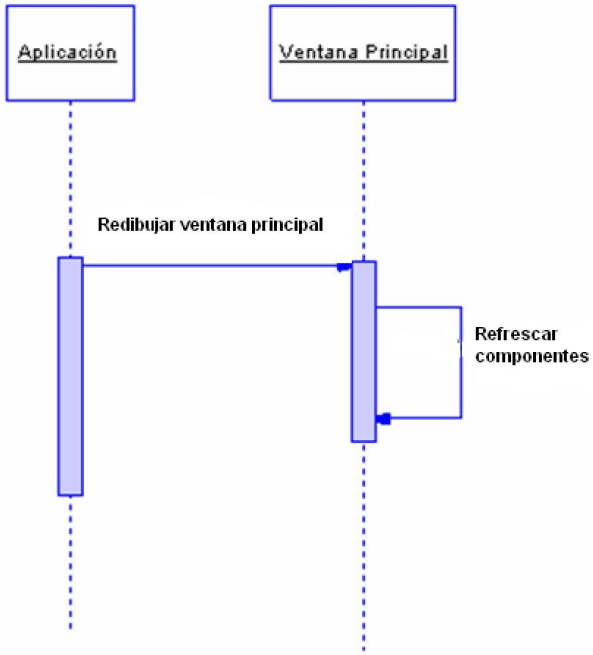
  - **la dimensión horizontal** en la que se muestra la disposición de los objetos y
  - **la dimensión vertical** donde se muestra el paso del tiempo.



Para entender un poco mejor los diagramas de secuencia veamos un **ejemplo**. Consideremos que cuando una aplicación se restaura se visualizará su ventana principal.
Si suponemos dos objetos:

- aplicación.
- ventana principal.

Se podría crear el siguiente diagrama de secuencia.



En este diagrama se observa como el objeto Aplicación pasa un mensaje al objeto Ventana principal para que se vuelva a pintar. Éste recibirá el mensaje y repintará todos sus componentes asociados.

3.- Diagrama de Comunicación.

El diagrama de comunicación al igual que ocurre con el diagrama de secuencia muestra cómo los objetos se comunican entre sí. **La diferencia** que existe entre ambos diagramas es que **los diagramas de secuencia atienden a la secuencia en la que ocurre la colaboración entre los objetos mientras que los diagramas de comunicación lo hacen atendiendo al contexto y organización de los objetos**.

Debido a que los diagramas de secuencia y comunicación son **equivalentes** exceptuando las particularidades propias de cada diagrama se puede pasar de un diagrama a otro con una mínima pérdida de información respecto al diagrama se secuencia, obteniendo en dicha transformación los elementos comunes a ambos. Será luego labor del diseñador completar el diagrama obtenido con los elementos propios del diagrama resultante. Más adelante veremos cómo es posible esto gracias a que los diagramas de comunicación van a tener numerados los mensajes, y esta numeración hace referencia a cómo suceden los mensajes en el tiempo, es decir, su secuencia.

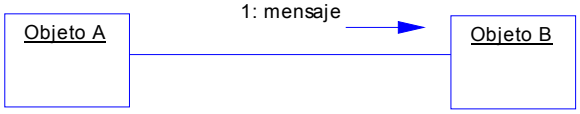
Veamos a continuación las particularidades de este tipo de diagramas que son las que vamos a tener en cuenta a la hora de construirlos.

A la hora de **construir** un diagrama de comunicación debemos indicar que éste se trata de una extensión de un diagrama de objetos. ¿Qué quiere decir esto?.

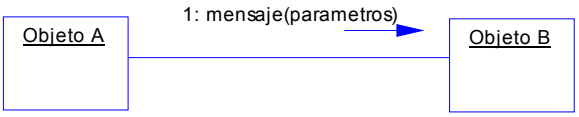
Pues que su **representación gráfica** será equivalente al de objetos pero añadiendo una mayor información. Por ejemplo, el diagrama de comunicación además de representar las relaciones entre objetos, indicará los mensajes que se intercambian éstos.

Ahora bien, **¿cómo representamos estos mensajes**?

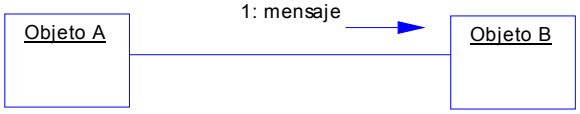
- Para representar un mensaje **se dibujará una flecha cerca de la línea de asociación entre objetos. Esta flecha apuntará al objeto que recibe el mensaje**.



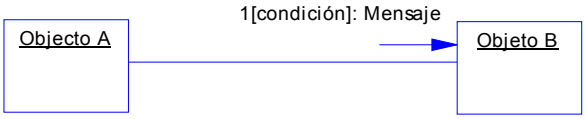
- **El mensaje terminará con un paréntesis donde incluiremos, si existen, los parámetros con los que éste funcionará**.



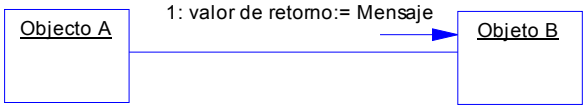
- **Los mensajes muestran un número en su parte izquierda separado del mensaje por dos puntos**. Esto es la secuencia en la que ocurren dichos mensajes, es decir, el uno sucederá en el tiempo antes que el número dos y así sucesivamente. **Debido a esto es posible pasar de un diagrama de comunicación a uno de secuencia de forma inmediata.**



- **Podemos indicar condiciones en los mensajes**. Si el mensaje que indicamos está sujeto a alguna condición podemos indicarlo incluyendo la condición dentro de corchetes entre el número que indica el orden de secuencia del mensaje y los dos puntos.



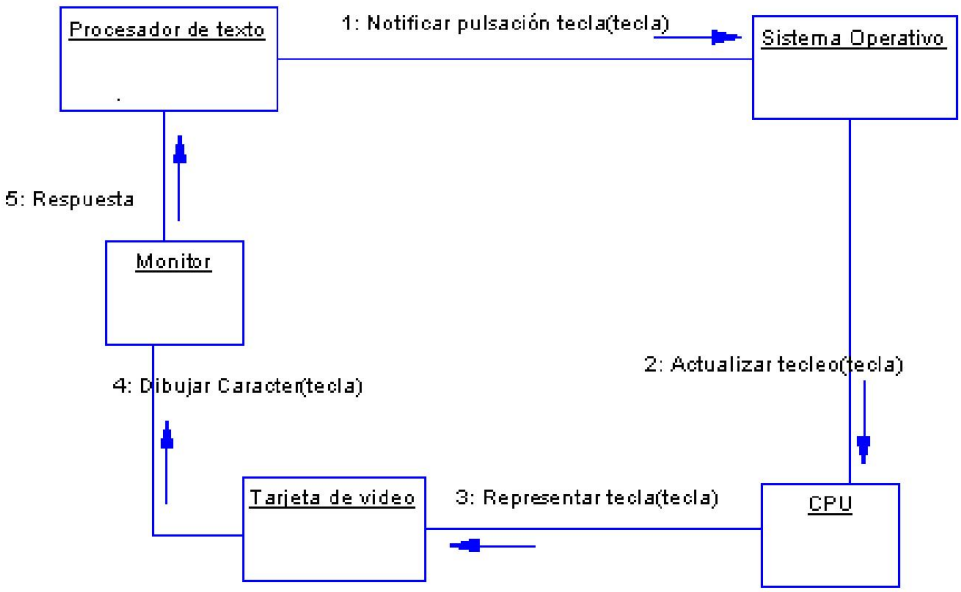
- **Podemos indicar si el mensaje retorna algún valor**. Si el mensaje retorna un valor lo indicaremos delante de los dos puntos que van junto al número que indica la secuencia con la que ocurre dicho mensaje.



Imaginemos el proceso que se sigue en un entorno gráfico como puede ser un procesador de textos desde el que se pulsa una tecla y se imprime por pantalla. Para ello vamos a detallar la información que necesitamos para construir el diagrama.

- **Los objetos que intervienen en el caso práctico serían**: Procesador de Textos, Sistema Operativo, CPU, Tarjeta Gráfica y Monitor.
- **Los mensajes que se producen serían**: El procesador de textos notifica al sistema operativo que se ha pulsado una tecla, el sistema operativo se lo indica a la CPU, la CPU se lo indica a la tarjeta de vídeo, ésta envía el mensaje al monitor, por último el monitor representa el carácter en pantalla.
- **Además del mensaje que los objetos se van pasando,** éste va acompañado de un parámetro "tecla" que contiene el carácter a representar.

Bien, con la información que tenemos **el diagrama resultante sería el siguiente**:



En el diagrama observamos los objetos, los mensajes y dentro de éstos el orden en que suceden y en algunos casos el paso de un parámetro, ya que es el valor que se utiliza para representar finalmente el carácter en pantalla.

4.- Diagrama de Estado.

Mientras los diagramas de interacción y comunicación ofrecen información (modelan) sobre grupos de objetos de un sistema, **los diagramas de estado los usaremos para modelar el comportamiento dinámico de un objeto en particular, o de una clase de objetos**.

Por tanto este diagrama nos proporcionará información detallada sobre algo más particular como es un **objeto**, mostrando los diferentes **estados** que éste irá tomando como consecuencia de los acontecimientos que lo involucren a lo largo del tiempo. Además, indicarán las respuestas y acciones que el objeto realizará como consecuencia de dichos acontecimientos.

Como se puede deducir de lo comentado, este tipo de diagramas aportará una mayor información sobre objetos determinados, es decir, si nos damos cuenta con el conjunto de diagramas UML que estamos estudiando, obtendremos como resultado el diseño del sistema al detalle, que es de lo que se trata.

Pero, **¿qué entendemos por acontecimientos o sucesos?**

Aunque es seguro que ya habrás encontrado la respuesta a esta pregunta, te expondremos algunos ejemplos:

Un acontecimiento o suceso de la vida real puede ser pulsar un interruptor eléctrico que conlleva a la **consecuencia** (acción) de que se encienda o apague la luz según la situación actual (con lo que obtenemos un cambio de estado), otro puede ser, girar la llave de contacto de un coche obteniendo que éste se pone en marcha, etc. Todas estas cosas son las que entendemos por acontecimientos o sucesos y como se ha expuesto y es lógico, a cada acontecimiento o suceso le acompaña un cambio de estado.

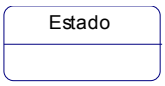
Bien, los diagramas de estado van a reflejar esos cambios de estado sobre los objetos, indicando los sucesos y acciones que ocurren en cada uno de esos cambios de estado.

Para construir un diagrama de estado debemos primero saber cómo vamos a representar todo esto. Para ello vamos a ver los diferentes elementos que componen dichos diagramas así como los diferentes casos que podemos indicar.

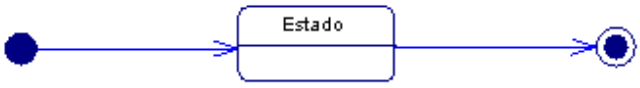
4.1.- Elementos que integran un diagrama de estados.

Vamos a ver la simbología propia de este diagrama con la que construiremos el mismo y plasmaremos la información que queremos.

- Un estado se representa mediante **un rectángulo de vértices redondeados**.



- Para llegar a un estado determinado se deberá pasar por una transición. Ésta la representaremos mediante **una línea continua terminada en flecha**. El sentido de la flecha nos indicará hacia donde conduce dicha transición. También encontraremos transiciones internas del estado como veremos más adelante.



- En la figura anterior**,** además de la flecha que representa la transición observamos dos círculos, **el primero de ellos con relleno sólido, india el inicio de una transición mientras que el segundo, no totalmente relleno, sino con un punto central, indica el final de una transición. Esto círculos representan el estado inicial y final de un objeto.**

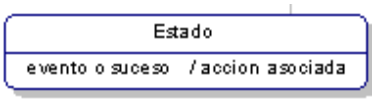
Pero solamente con esto nuestro diagrama no representará gran cosa. Vamos a entrar ahora en detalle para cada uno de los elementos expuestos en este punto.

4.2.- Sucesos y Acciones del estado.

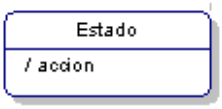
Vamos a ver con más detalle en este punto los sucesos y las acciones que lleva asociadas la consecución de un nuevo estado.
El estado podemos dividirlo en dos partes:

- **La primera es obligatoria y será donde indiquemos el nombre del estado.**
- **La segunda es opcional y en ella indicaremos los eventos y acciones o si se producen.**

Para representar en un estado los sucesos y acciones, lo haremos **separándolos mediante una línea diagonal**, donde situaremos en su lado izquierdo el suceso o evento y en su lado derecho la acción que desencadena.



En los estados podemos encontrar **sucesos asociados a acciones y acciones solas.**

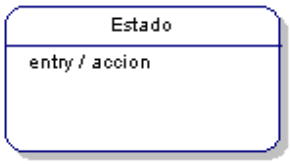


En la mayoría de las ocasiones las acciones irán agrupadas dentro de tres tipos de sucesos:

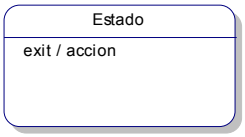
- **De entrada.**
- **Internos del estado.**
- **De salida**.

Aunque también dependiendo de la herramienta que utilicemos para modelar este tipo de diagramas, es posible definirnos nuestros propios eventos y sucesos**. Esto nos va a indicar cuándo se realizan las acciones en el estado**. Pero para entenderlo mejor veámoslo más detenidamente.

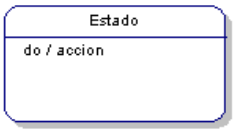
- **De entrada** (suceden cuando el sistema entra en el estado), son las acciones que suceden cuando se entra en un estado debido a una transición. Para indicar que una acción se produce cuando el sistema entra en el estado, ésta se asocia al evento “entry” o “entrada”.



- **De salida** (suceden cuando el sistema sale del estado), son las acciones que suceden cuando se sale del estado por medio de una transición. Para indicar esto las acciones se asocian al suceso o evento “exit” o “salida”.



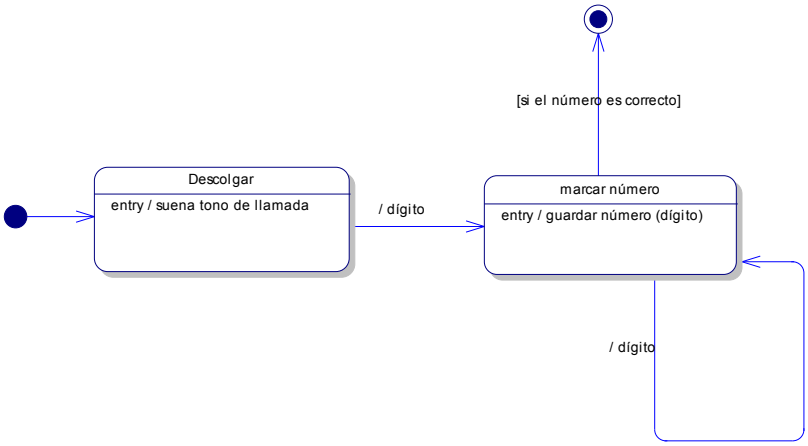
- **Internas, denominadas como “hacer”** (que suceden cuando el sistema se encuentra en el estado), son las acciones o sucesos que ocurren cuando se está dentro del estado y se recibe un determinado evento. Este tipo de acciones no genera ninguna transición. Aunque este tipo de eventos engloba todos aquellos que no son de entrada ni de salida, ya hemos comentado que podemos personalizar nuestros eventos y no utilizar siempre éste, aunque esto será decisión del profesional.



Veamos ahora un ejemplo de lo comentado: Imaginemos que queremos realizar una llamada telefónica a un amigo y queremos representar los estados por los que pasa el teléfono para ello:

Primer estado, “**descolgamos el teléfono”**. Al hacer esto escuchamos el tono de llamada. Aquí podemos ver cómo cuando se entra en el estado “descolgar”, es decir, se descuelga el teléfono inmediatamente se escucha el tono de llamada. Por esto “suena el tono de llamada” esta asociado a un evento de entrada.

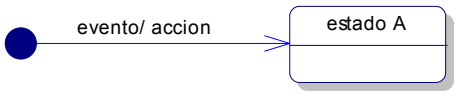
Una vez que tenemos el teléfono descolgado lo siguiente que realizamos, ¿qué sería? Exactamente, marcar el número telefónico de nuestro amigo. Esto nos va a conducir a **un nuevo estado del teléfono "marcando números",** ya que el teléfono cambia su comportamiento. Veamos este nuevo estado.



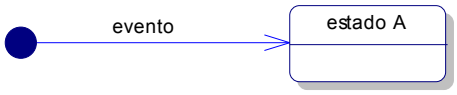
Ahora se producirá en este nuevo estado otra acción asociada a un suceso de entrada, que será ir guardando los números que vamos tecleando. Por último, cuando el número está completo y es correcto se llegará al estado final de la llamada.

4.3.- Sucesos y acciones en las transiciones.

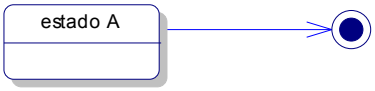
Al igual que en los estados, en las transiciones (que son las que provocan los nuevos estados) **se pueden indicar sucesos** (algo que ocurre) **y acciones** (algo que se realiza). Para indicar esto, los sucesos y acciones los escribiremos **cerca de la línea de transición separados por una diagonal**, de esta forma identificamos cuál es el suceso y la acción asociada. **El suceso se sitúa a la izquierda de la diagonal y la acción a la derecha.**



Como se puede observar en la figura anterior, la transición que lleva al estado A, está asociada a un evento al que le corresponde una acción. Pero también puede ocurrir que una transición se desencadene por un evento solamente, sin que se produzca ninguna acción asociada. Esto se refleja de la siguiente forma:



Otras veces encontraremos que una **transición se origina debido a que un estado terminará una acción** (en lugar de darse por un suceso). A este tipo de transiciones se les llama como **"transición no desencadenada"**.



Llegado a este punto ya tenemos que tener claro que:

- En una transición, **un suceso o evento puede desencadenar o no una acción**. Si ésta ocurre se separarán por medio de una diagonal para identificarlos.
- Existen transiciones que **son producidas por la finalización de una acción del estado**. A estas transiciones se les denominan como "**transición no desencadenada**".

5.- Diagrama de Actividad.

Los diagramas de actividad surgieron como una extensión de los diagramas de estados. Mientras los diagramas de estados se centran en representar los estados por los que pasa un objeto y las actividades o acciones que los desatan, **el diagrama de actividad se centrará en desarrollar las acciones o actividades de los objetos**.
Como se puede apreciar este tipo de diagramas profundiza todavía más, dando información más precisa sobre los objetos, en este caso sobre sus operaciones.

5.1.- Elementos del diagrama de Actividad.

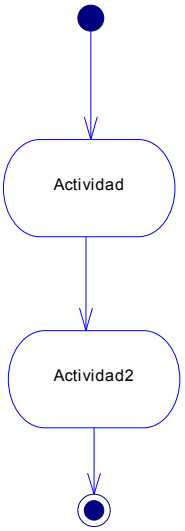
Los **elementos** que vamos a utilizar en un diagrama de actividad son los siguientes:

- Al igual que pasaba con el diagrama de estados, en los diagramas de actividad vamos a tener un **punto de origen y un punto final del diagrama**. Estos se representan mediante los mismos símbolos que en el anterior.

- **Las actividades las vamos a representar mediante un rectángulo con vértices ovalados**. Dentro de cada actividad se realizarán sus operaciones y una vez que éstas terminen se pasa a la siguiente actividad.

- **Para pasar de una actividad a otra se indica mediante un flujo de control**. Ésta se representa mediante una línea acabada en punta.

En la figura se puede observar un diagrama completo de actividades que integra todos estos elementos.

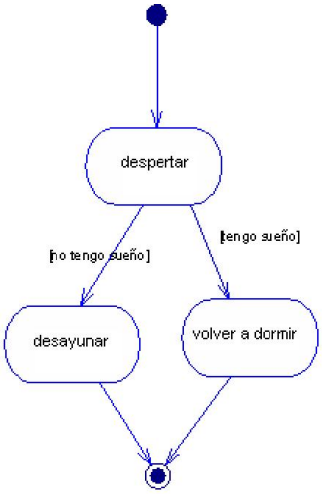


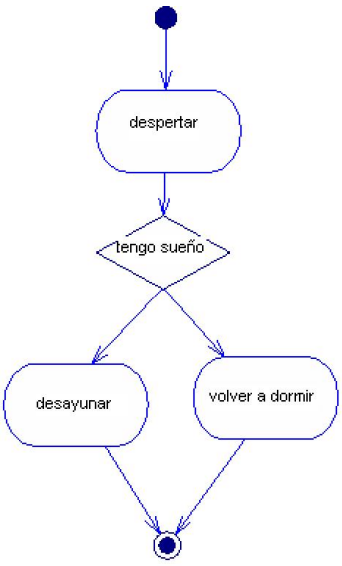
5.2.- Indicando Decisiones.

Muchas veces el término de una actividad nos puede llevar, dependiendo de alguna condición, a **dos actividades diferentes**, es decir, se nos plantea una decisión.

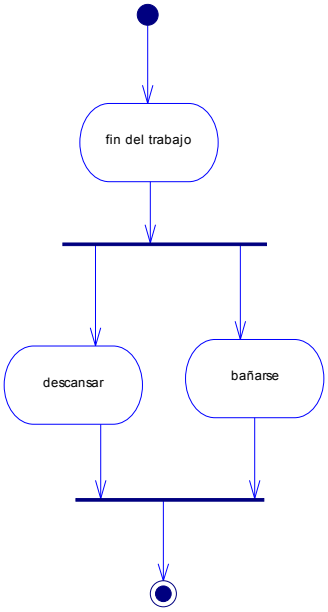
Para representar una decisión, lo podemos hacer de dos formas:

- **Plantear todas las rutas posibles que parten de una actividad**.
- **Utilizando un rombo de decisión.**





5.3.- Rutas concurrentes.

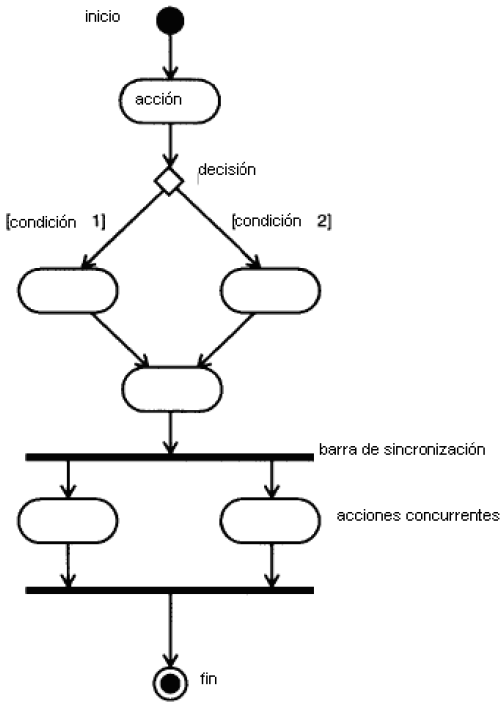


En un diagrama de actividades también pueden existir barras de sincronización, a las que se encuentran asociadas varios caminos salientes**, es decir flujos de control**. Cada transición saliente se dirige a una actividad, realizándose dichas actividades en paralelo. Esto quiere decir que el orden en que se realicen dichas actividades es irrelevante, siendo válido cualquier orden entre ellas. **De esta manera representamos concurrencia en el sistema**.

Dado que el diagrama de actividades permite expresar el orden en que se realizan las cosas, se puede utilizar para el modelado de organizaciones (*business modeling*) y el de programas concurrentes.

Como la mayoría de las técnicas de modelado de comportamiento, los diagramas de actividades tienen sus puntos fuertes y sus puntos débiles, de forma que es necesario utilizarlos en combinación con otras técnicas**. Su principal aportación al modelado del comportamiento es que soportan el comportamiento paralelo, lo que resulta adecuado para el modelado de flujo de trabajo y programación** **multihilo** (multi-thread). Por contra, su principal desventaja es que no muestran de una forma clara los enlaces existentes entre las acciones y los objetos, siendo mucho más apropiado para ello los diagramas de interacción.

En la figura se muestran los elementos que podemos encontrar en un diagrama de actividad.



6.- Diagrama de componentes.

Lo que distingue a **un diagrama de componentes** de otros tipos de diagramas es **su contenido.** Normalmente contienen **componentes, interfaces y relaciones entre ellos**. Un componente software es una parte física de un sistema, y se encuentra en el ordenador, no en la mente del analista.

Ahora bien, ¿**Qué podemos considerar como componentes**?

Una parte física y reemplazable del sistema que realiza cierta funcionalidad y la expone a través de un conjunto de interfaces bien definidos. Este conjunto de interfaces representa el contrato del componente, de tal manera que la implementación interna puede variar pero manteniendo el contrato inalterable.

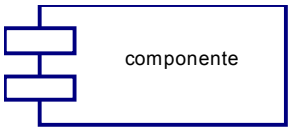
Los componentes los podemos relacionar con las clases, de forma que un componente estará constituido por una o varias clases que colaboran para realizar determinada funcionalidad. Esta funcionalidad se ofrece al exterior a través de los interfaces del componente.

Pero, ¿**por qué es necesario modelar componentes?**

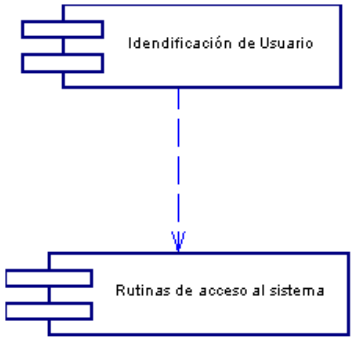
- Para que los clientes puedan ver el sistema finalizado.
- Al modelar los componentes de un sistema los desarrolladores contarán con una estructura con la que trabajar.
- El modelo de componentes ayudará a las personas que tengan que generar la documentación del sistema.

Uno de los aspectos más importantes de los componentes **es la posibilidad que tienen de volver a ser utilizados**. De esta manera, al reutilizar un componente obtendremos grandes beneficios, ya que se ahorra en tiempo, recursos humanos y materiales, que al fin y al cabo redunda en dinero.

Un componente se representa mediante el siguiente objeto, dentro del cual se situará su nombre.



Los componentes pueden aparecer conectados indicando comunicación entre ellos. Como indicamos al comienzo del apartado, **un componente es la representación física de una clase**. Los componentes tienen **una interfaz.** Esta interfaz es lo que el componente muestra, ocultando los detalles que le interesan. A la hora de trabajar con el componente lo haremos con su interfaz.Normalmente los diagramas de componentes se utilizan para modelar código fuente, versiones ejecutables, bases de datos físicas, entre otros:



- **Código fuente**: En el modelado de código fuente se suelen utilizar para representar las dependencias entre los ficheros de código fuente, o para modelar las diferentes versiones de estos ficheros.
- **Código ejecutable**: Se utiliza para modelar la distribución de una nueva versión a los usuarios. Para tal propósito se identifican el conjunto de componentes ejecutables que intervienen, se utilizan estereotipos para los diferentes tipos de componentes (ejecutables, bibliotecas, tablas, archivos, documentos, etc.), se consideran las relaciones entre dichos componentes que la mayoría de las veces incluirán interfaces que son exportadas (realizadas) por ciertos componentes e importadas (utilizadas) por otros.
- **Bases de datos físicas**: UML permite el modelado de bases de datos físicas así como de los esquemas lógicos de bases de datos.

PARA SABER MÁS.

En este enlace podrás encontrar más información sobre los diagramas de componentes UML

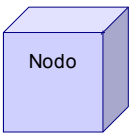
Diagrama de componentes UML

http://es.wikipedia.org/wiki/Diagrama_de_componentes

7.- Diagrama de despliegue.

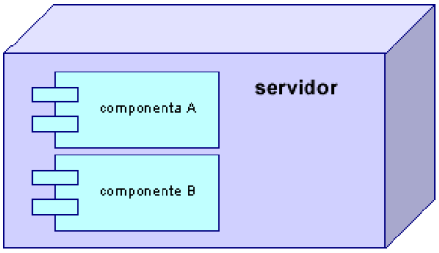
En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Es decir, se sitúa el software en el hardware que lo contiene**.** Muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software**.**

Cada elemento hardware se representa con una figura que simboliza un **cubo** y que recibe el nombre de **nodo**.



En el nodo se ejecutan los componentes (que es precisamente el software como se ha visto en puntos anteriores). Para ello el nodo requiere ser **objeto físico en tiempo de ejecución que representa un recurso computacional**, generalmente con memoria y capacidad de procesamiento. Por ejemplo un ordenador.

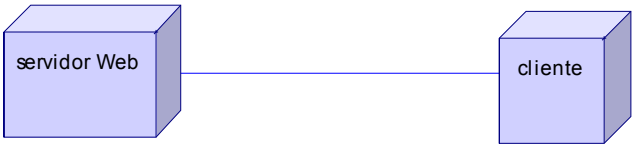
Imagina que el software que estamos desarrollando para una empresa se divide en dos componentes y estos se van a alojar en el ordenador que hace de servidor en la empresa que además será quien los ejecute. Bien, el resultado de esto sería el siguiente diagrama:



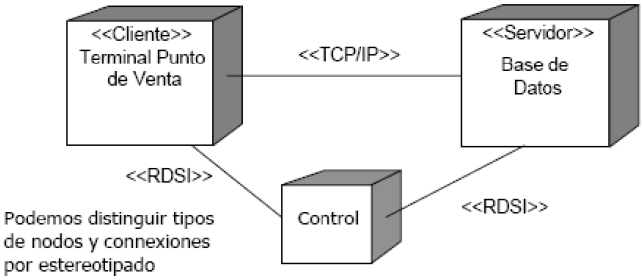
Con esto lo que indicamos es que nuestro software compuesto por el componente A y el componente B estará y será ejecutado en la máquina servidor de nuestro cliente. Además, los componentes que se sitúan en los nodos son componentes instanciados, es decir, representan el código que se está ejecutando o que está listo para ser ejecutado.

En un diagrama de despliegue podemos encontrar también **nodos unidos mediante una línea.** Esta línea recibe **el nombre de conexión de comunicación**.

De esta manera indicamos la existencia de conexión entre dos máquinas.



Otro ejemplo de conexiones entre nodos es el siguiente:



La mayoría de las veces el **modelado de la vista de despliegue** implica modelar la topología del hardware sobre el que se ejecuta el sistema. Aunque UML no es un lenguaje de especificación hardware de propósito general, se ha diseñado para modelar muchos de los aspectos hardware de un sistema a un

nivel suficiente para que un ingeniero software pueda especificar la plataforma sobre la que se ejecuta el software del sistema.

Algunos de los usos que se les da a los diagramas de despliegue son para modelar:

- **Sistemas empotrados**: Un sistema empotrado es una colección de hardware con una gran cantidad de software que interactúa con el mundo físico.

- **Sistemas cliente-servidor**: Los sistemas cliente-servidor son un extremo del espectro de los sistemas distribuidos y requieren tomar decisiones sobre la conectividad de red de los clientes a los servidores y sobre la distribución física de los componentes software del sistema a través de nodos.

- **Sistemas completamente distribuidos**: En el otro extremo encontramos aquellos sistemas que son ampliamente o totalmente distribuidos y que normalmente incluyen varios niveles de servidores. Tales sistemas contienen a menudo varias versiones de componentes software, alguno de los cuales pueden incluso migrar de un nodo a otro. El diseño de tales sistemas requiere tomar decisiones que permitan un cambio continuo de la topología del sistema.

PARA SABER MÁS.
En este enlace podrás encontrar más información sobre los diagramas de Despliegue UML. Toma nota de los cambios que este tipo de diagramas sufren en la versión 2.0 de UML.

Diagrama de Despliegue UML
http://es.wikipedia.org/wiki/Diagrama_de_despliegue

8.- Diagrama de Objetos.

Los diagramas de objetos modelan las instancias de elementos contenidos en los diagramas de clases. En los diagramas de objetos encontraremos dos elementos principales, los objetos y sus relaciones.
Hay que tener muy claro que **los diagramas de objetos muestran un conjunto de objetos y sus relaciones en un momento concreto**.

En UML, **los diagramas de clase se utilizan para visualizar los aspectos estáticos** del sistema y los **diagramas de interacción se utilizan para ver los aspectos dinámicos** del sistema, y además, como

ya hemos visto, estos últimos constan de instancias de los elementos del diagrama de clases y mensajes enviados entre ellos.

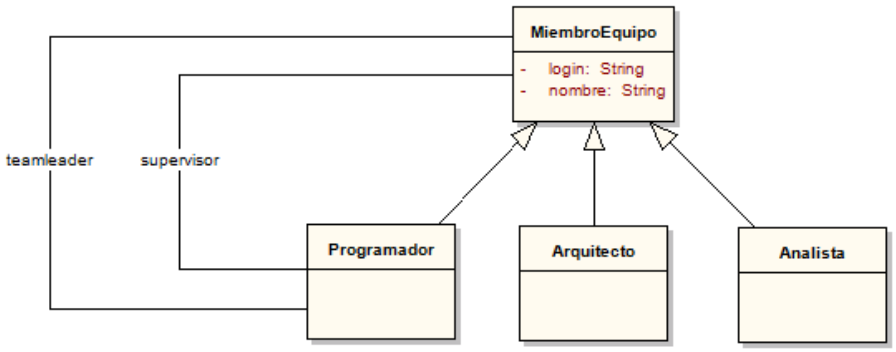
En un punto intermedio podemos situar **los diagramas de objetos, ya que contienen un conjunto de instancias de los elementos encontrados en el diagrama de clases, pero representando sólo la parte estática de una interacción**.

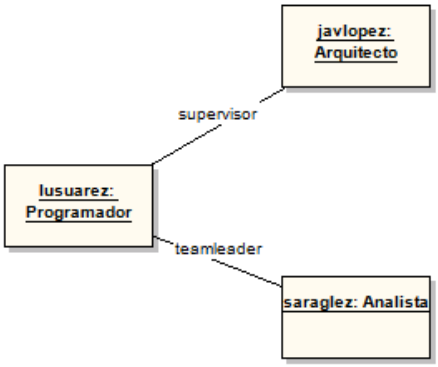
Por tanto, deducimos que este tipo de diagramas lo vamos a utilizar para modelar la **vista de diseño estática** o la vista de procesos estática de un sistema. Si recordamos esto también lo hacían los diagramas de clases, la diferencia es que el diagrama de objetos lo representará desde el punto de vista de instancias reales del sistema.

En general los diagramas de objetos se utilizan para modelar estructuras de objetos, lo que implica tomar una instantánea de los objetos de un sistema en un cierto momento. Un diagrama de objetos representa una escena estática dentro de la historia representada por un diagrama de interacción. Los diagramas de objetos se utilizan para visualizar, especificar, construir y documentar la existencia de ciertas instancias en el sistema, junto a las relaciones entre ellas.

Hay que tener presente a la hora de construir este tipo de diagramas, que pueden existir multitud de **instancias** de una clase y para un conjunto de clases con relaciones entre ellas. Esto nos lleva a la situación de que podemos tener multitud de situaciones distintas producidas por la misma clase. Por tanto un diagrama de objetos sólo mostrará conjuntos de objetos concretos.

Debido a que este diagrama se podrá generar automáticamente a partir del diagrama de clases, no se incluyen en este punto los elementos que lo integran (los mismos que para el diagrama de clases).





PARA SABER MÁS.

En este enlace encontrarás toda la información que necesites sobre UML. Por tanto será un buen enlace para ampliar tus conocimientos sobre cualquier diagrama visto, nuevas versiones del lenguaje UML como puede ser la versión 2, etc. El inconveniente que tiene es que está en inglés.

Todo sobre UML

http://www.uml.org/

GLOSARIO.

Base de datos física.

Se refiere a la implementación física de una base de datos, es decir, sería la base de datos creada en un ordenador.

Código ejecutable

Es el código que puede ser interpretado directamente por el ordenador.

Código fuente.

El código fuente es un conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos.

Diagramas de interacción.

Los diagramas de interacción engloban a dos diagramas: de secuencia y de comunicación.

Multihilo.

Un sistema multihilo es aquel en el que se ejecutan al menos dos procesos o hilos de forma simultánea.

Sistemas cliente-servidor.

Es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permita simplificarlas.

Sistemas distribuidos.

Es un sistema en que existen varias CPU conectadas entre sí, además las distintas CPU trabajan de manera conjunta. Los sistemas distribuidos necesitan un software distinto al de los sistemas centralizados.

Sistemas empotrados.

Son sistemas en los que encontramos una colección de hardware con una gran cantidad de software que interactúa con el mundo físico.