UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERIA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SYSTEMAS

DESARROLLO DE SOFTWARE



**LENGUAJE UNIFICADO DE MODELADO(UML).**

**ELEMENTOS Y DIAGRAMAS.**

Presentado por:

Christian Yánez C.I. 18.099.810

Aldo Olivares C.I. 19.854.994

Dixon Marrufo C.I. 20.358.837

Cesar Roldan C.I. 21.231.177

**Barcelona, Mayo de 2015**

Índice

1. Elementos del UML 3

2. Elementos de Agrupación 3

2.1. Paquete 4

3. Elementos de Notación 4

4. Diagrama de Estructura estática 4

4.1. Clase 5

4.2. Asociaciones 5

a) Asociación binaria 5

b) Sentido 6

c) Multiplicidad 6

d) Roles 7

e) Composición 7

4.3. Generalización 7

4.4. Clase paramétrica 8

4.5. Paquete 8

4.6. Dependencia 9

4.7. Agregación 10

a) Agregación por contenido físico o por valor 10

b) Agregación conceptual o por referencia 10

4.8. Elementos Derivados 11

5. Diagrama de caso de uso 11

5.1. Actor 12

5.2. Relaciones en un diagrama de caso de uso 12

5.3. Caso de uso 13

5.4. Límite de Sistema: 13

6. Diagrama de secuencia 13

6.1. Línea de vida de un objeto 13

6.2. Activación 14

6.3. Mensaje 14

7. Diagrama de Colaboración 15

7.1. Objeto 15

7.2. Enlaces 15

7.3. Flujo de mensajes 15

7.4. Objeto compuesto 16

8. Diagrama de Estados 16

8.1. Estado 17

8.2. Eventos 17

8.3. Envío de mensajes 18

8.4. Transición simple 18

8.5. Transición interna 19

8.6. Diagrama de Actividades 19

8.7. Estado de acción 20

8.8. Transiciones 20

8.9. Decisiones 21

9. Diagrama de Implementación 21

10. Diagrama de Componentes 21

11. Diagrama de Ejecución 22

11.1. Nodos 22

11.2. Componentes 22

.

1. Elementos del UML

Hay 4 tipos de elementos en UML, elementos estructurales, elementos de comportamiento, elementos de agrupación y elementos de anotación:

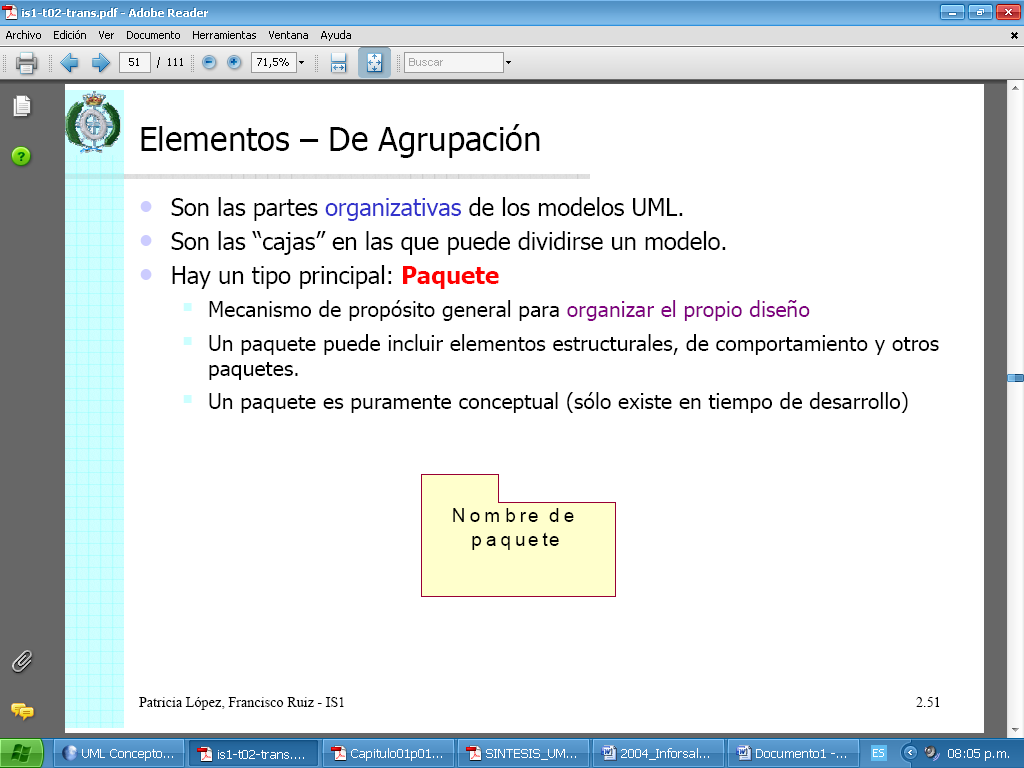
1. Elementos de Agrupación

Los elementos de agrupación son las partes organizativas de los modelos UML. Estos son las cajas en las que se puede descomponer un modelo. Hay un tipo principal de elementos de agrupación.

* 1. Paquete

Un paquete es un mecanismo de propósito general para organizar el propio diseño. Un paquete puede incluir elementos estructurales, de comportamiento y otros paquetes.

Un paquete es puramente conceptual, sólo existe en la etapa de desarrollo. Gráficamente, un paquete se visualiza como una carpeta, normalmente incluye su nombre, y su contenido. Hay variaciones, tal como Frameworks, modelos y subsistemas (tipos de paquetes).



1. Elementos de Notación

A continuación se verán una serie de elementos notaciones pero sin entrar en detalle en la esencia de lo que es el concepto de cada uno de ellos.

También visualizaremos como representar gráficamente los conceptos principales de la orientación a objetos.

A continuación listaremos los siguientes elementos que serán tratados:

* Nota.

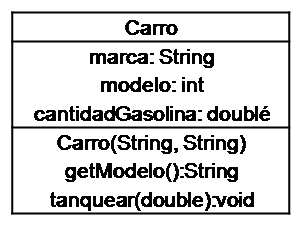
1. Diagrama de Estructura estática

Un diagrama de estructura estática muestra el conjunto de clases y objetos importantes que hacen parte de un sistema, junto con las relaciones existentes entre estas clases y objetos. Muestra de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con las demás en el modelo.

* 1. Clase

Una clase se representan como un rectángulo dividido en tres partes (tres compartimentos).

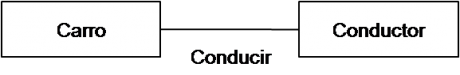
* En el compartimento superior, centrado, va el nombre de la clase.
* El segundo compartimento contiene los atributos, en la forma: visibilidad nombreAtributo: tipo
* El tercer compartimento contiene los métodos, en la forma: visibilidad nombreMétodo(parámetros): tipoRetorno.
* Los parámetros van separados por comas y sólo se especifica el tipo.
* Los constructores no tienen tipo de retorno.
* La visibilidad, de atributos y métodos, puede ser:  – privado, + público, # protegido, ~ de paquete



Ejemplo de clase

* 1. Asociaciones

Una asociación (relación entre dos clases) se representa como una línea continua entre dos Clases, y puede tener el nombre de la relación sobre esta línea. Ejemplo:

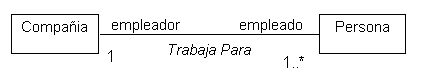
[](http://3.bp.blogspot.com/-fiUZI3Kazyk/TyNOLf0RuxI/AAAAAAAAAwg/nGFTGwyqqyU/s1600/ejemploasociacionconducir.png)

Ejemplo de asociaciones

En el ejemplo anterior cada Carro tiene una relación con Conductor, pero también cada Conductor tiene relación con Carro. Esta es una relación en ambos sentidos.

1. Asociación binaria

Se identifica como una línea sólida que une dos clases. Representa una relación de algún tipo entre las dos clases, no muy fuerte ( es decir, no se exige dependencia existencial ni encapsulamiento).



Un ejemplo es la relación entre una compañía y sus empleados

En este caso la relación recibe el nombre genérico *Trabaja Para*, la compañía tiene uno o más instancias de la clase Persona denominadas *empleado*y cada empleado conoce su *empleador*(en este caso único).

1. Sentido

Para mostrar que la relación sólo tiene un sentido se muestra con una flecha que indica el sentido de la relación. Ejemplo:

[http://3.bp.blogspot.com/-41X-hGO6xZo/TyNOLM3sc0I/AAAAAAAAAwU/Dc6Dta2lGKA/s1600/ejemploasociacion2.png](http://3.bp.blogspot.com/-41X-hGO6xZo/TyNOLM3sc0I/AAAAAAAAAwU/Dc6Dta2lGKA/s1600/ejemploasociacion2.png)

En este ejemplo un Pasajero conoce el Carro(o carros) con el cual viaja, pero el Carro no tiene ninguna relación con los Pasajeros.

1. Multiplicidad

Es una restricción que se pone a una asociación, que limita el número de instancias de una clase que pueden tener esa asociación con una instancia de la otra clase.

[http://3.bp.blogspot.com/-DrrJUVYz7BU/TyNOJozrTeI/AAAAAAAAAvg/On53-yx69WM/s1600/ejemplomultiplicidad.png](http://3.bp.blogspot.com/-DrrJUVYz7BU/TyNOJozrTeI/AAAAAAAAAvg/On53-yx69WM/s1600/ejemplomultiplicidad.png)

En este caso las relaciones son:

- Un chofer tiene relación con cero o más autobuses.

- Un autobús tiene relación con uno o dos choferes.

- Una terminal de pasajero tiene relación con cero o muchos autobuses.

- Un autobús tiene relación con un terminal de pasajero.

1. Roles

Para indicar el papel que juega una clase en una asociación se puede especificar un nombre de rol.  Se representa en el extremo de la asociación junto a la clase que desempeña dicho rol.

ejemploroles

Ejemplo de roles.

1. Composición

Es una asociación fuerte, que implica tres cosas:

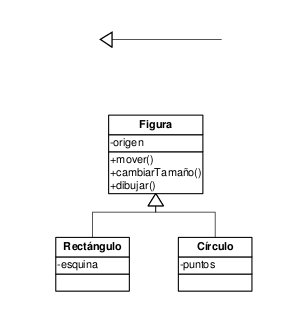
1. Dependencia existencial. El elemento dependiente desaparece al destruirse el que lo contiene y, si es de cardinalidad 1, es creado al mismo tiempo.
2. Hay una pertenencia fuerte. Se puede decir que el objeto contenido es parte constitutiva y vital del que lo contiene
3. Los objetos contenidos no son compartidos, esto es, no hacen parte del estado de otro objeto.

Se denota dibujando un rombo relleno del lado de la clase que contiene a la otra en la relación.

Existe también una relación de composición menos fuerte (no se exige dependencia existencial, por ejemplo) que es denotada por una un rombo sin rellenar en uno de los extremos.

* 1. Generalización

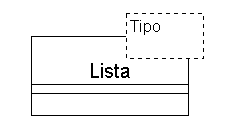
La relación de generalización denota una relación de herencia entre clases. Se representa dibujando un triángulo sin rellenar en el lado de la superclase. La subclase hereda todos los atributos y mensajes descritos en la superclase.



* 1. Clase paramétrica

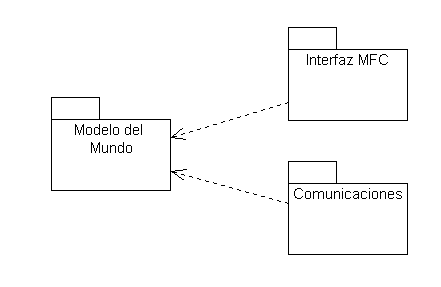
Una clase paramétrica representa el concepto de clase genérica en los conceptos básicos OO o de témplate en C++. Se dibuja como una clase acompañada de un rectángulo en la esquina superior derecha, con los parámetros del caso.

Por ejemplo, la clase Lista que utiliza un parámetro formal *Tipo*se vería de la siguiente manera



* 1. Paquete

Un paquete es una forma de agrupar clases (u otros elementos en otro tipo de diagramas) en modelos grandes. Pueden tener asociaciones de dependencia o de generalización entre ellos. Un ejemplo puede ser el siguiente

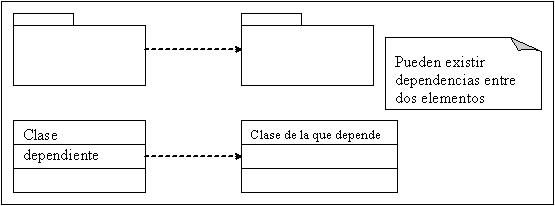


Ejemplo de paquetes

En este caso existen tres paquetes (que se muestran vacios en este caso, con su contenido encapsulado), con dos de ellos dependiendo del Modelo del Mundo.

* 1. Dependencia

Denota una relación semántica entre dos elementos (clases o paquetes, por el momento) del modelo. Indica que cambiar el elemento independiente puede requerir cambios en los dependientes. Se muestra como una línea punteada direccional, indicando el sentido de la dependencia. Puede tener por medio de estereotipos una explicación del tipo de dependencia presentada.



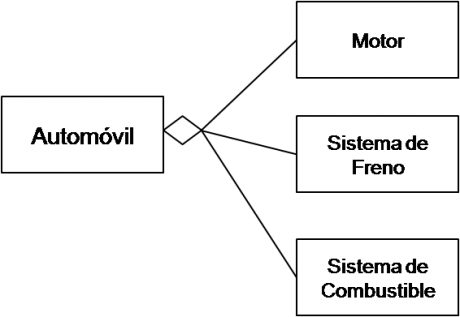
En el ejemplo anterior pueden verse dos relaciones de dependencia hacia el paquete Modelo del Mundo.

* 1. Agregación

La agregación representa el objeto compuesto.  Durante el desarrollo de una aplicación se nos presentara la necesidad de crear objetos complejos que no encajan con los tipos de datos básicos que proveen los lenguajes: tipo caracteres, enteros, reales, entre otros. El símbolo de agregación es un diamante colocado en el extremo en el que está la clase que representa el todo.  Podemos trabajar con dos tipos de agregación: Agregación por Valor y Agregación por Referencia.

1. Agregación por contenido físico o por valor

El contenedor contiene el objeto en sí. Cuando creamos un objeto contenedor, se crean también automáticamente los contenidos.  Ejemplo:

[](http://2.bp.blogspot.com/-7qR8Lkr2r34/TyNOK8F-SNI/AAAAAAAAAwQ/-78ymkfO3G4/s1600/ejemploagregacionvalor.png)

1. Agregación conceptual o por referencia

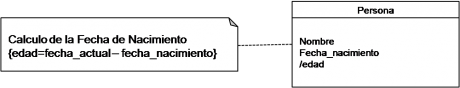
Se tienen punteros a objetos. No hay un acoplamiento fuerte. Los objetos se crean y se destruyen dinámicamente.

[](http://3.bp.blogspot.com/-jQ3eGWooiMU/TyNOKllO7sI/AAAAAAAAAwA/TijvtF8fCbA/s1600/ejemploagregacionreferencia.png)

En ambos casos la agregación se destaca por un rombo transparente.

* 1. Elementos Derivados

 En algunas ocasiones nos encontramos con la posibilidad de calcular un valor de un elemento a partir de otros elementos presentes en el modelo.  Se incluye para dar mayor claridad al diseño.  Se representa con una barra / precediendo al nombre del elemento derivado.

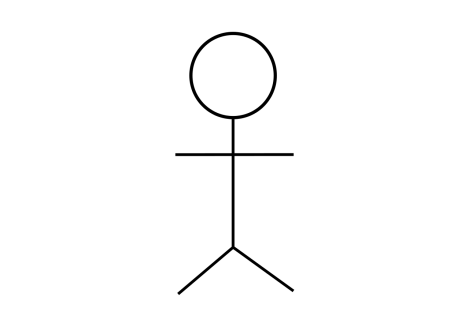
[](http://1.bp.blogspot.com/-BZJHi1k8vU8/TyNOL05EheI/AAAAAAAAAwk/Al2IR1Aj6BA/s1600/ejemploderivado.png)

1. Diagrama de caso de uso

Es la representación gráfica de cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.  El caso de uso es la representación gráfica de cada proceso desde la perspectiva usuario, lo que se espera del sistema en cuanto a funcionalidad.

* 1. Actor

Representa un rol que es llevado a cabo por una persona, otro sistema, sensores, etc. El actor es representado por una figura en forma de persona.



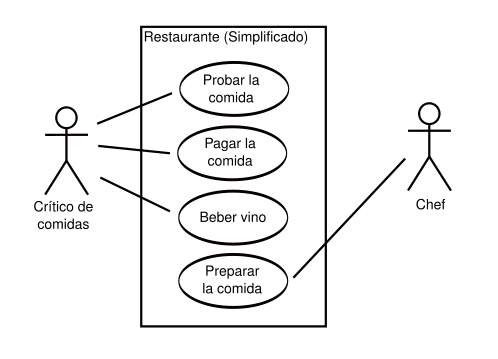
* 1. Relaciones en un diagrama de caso de uso

Este elemento representa la relación que existe entre un Caso-Uso y un Actor, dicho elemento es representado por una línea recta que se extiende de la figura del actor hacia el ovalo del caso-uso.

Entre dos casos de uso puede haber relaciones:

1. Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.
2. Usa: Cuando un caso de uso utiliza a otro.

Se representa como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta “extiende” o “usa” según sea el tipo de relación.



* 1. Caso de uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Se representa mediante un ovalo.

* 1. Límite de Sistema:

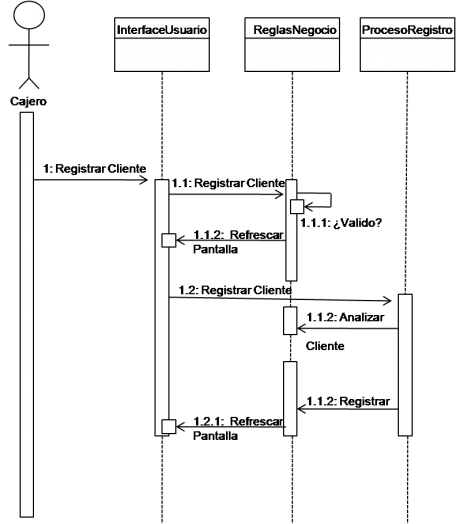
Empleado para delimitar los límites del sistema.  Se representa por un rectángulo con color de fondo distintivo.

1. Diagrama de secuencia

Muestra una interacción ordenada según la secuencia temporal de eventos.  El diagrama de secuencia muestra la interacción entre los objetos que tiene lugar a través del intercambio de mensajes, en particular, muestra los objetos participantes en la interacción y los mensajes que se intercambian ordenadas según la secuencia de tiempo.

* 1. Línea de vida de un objeto

Un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal que denotan la ejecución de métodos. El rectángulo de encabezado contiene el nombre del objeto y el de su clase, en un formato *nombreObjeto: nombreClase*



* 1. Activación

Muestra el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto.

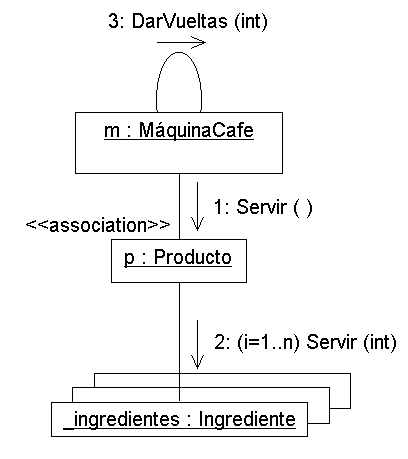
* 1. Mensaje

El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta.

Los diagramas de clases y los objetos representan información estática, no obstante en un sistema funcional los objetos interactúan entre sí, y tales interacciones suceden con el tiempo, el diagrama de secuencias UML muestra la mecánica de interacción con base en tiempos.

1. [Diagrama de Colaboración](http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/colaboracion01.html)

Un diagrama de colaboración es una forma de representar interacción entre objetos, alterna al diagrama de secuencia. A diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación (cuáles objetos son atributos, cuáles temporales, etc. ) y ciclos en la ejecución. Se toma como ejemplo el caso de uso PedirProducto ya descrito como diagrama de secuencia.



* 1. Objeto

Un objeto se representa con un rectángulo, que contiene el nombre y la clase del objeto en un formato *nombreObjeto: nombreClase*.

* 1. Enlaces

Un enlace es una instancia de una asociación en un diagrama de clases. Se representa como una línea continua que une a dos objetos. Esta acompañada por un número que indica el orden dentro de la interacción y por un estereotipo que indica que tipo de objeto recibe el mensaje. Pueden darse varios niveles de subíndices para indicar anidamiento de operaciones. Los estereotipos indican si el objeto que recibe el mensaje es un atributo (asociación y se asume por defecto), un parámetro de un mensaje anterior, si es un objeto local o global.

* 1. Flujo de mensajes

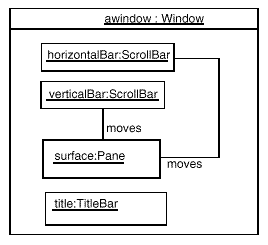
Expresa el envío de un mensaje. Se representa mediante una flecha dirigida cercana a un enlace.

Marcadores de creación y destrucción de objetos

Puede mostrarse en la gráfica cuáles objetos son creados y destruidos, agregando una restricción con la palabra *new*o *delete*, respectivamente, cercana al rectángulo del objeto

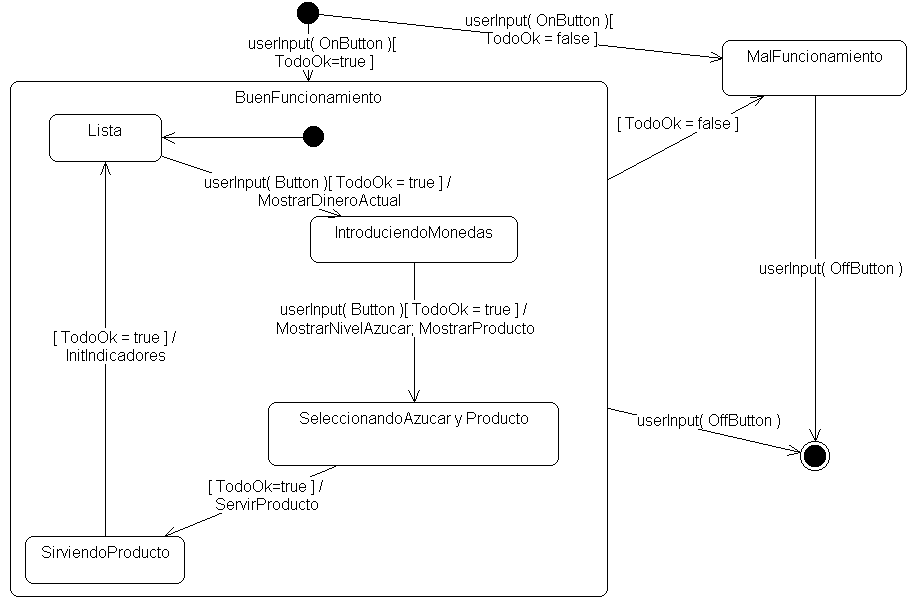
* 1. Objeto compuesto

Es una representación alternativa de un objeto y sus atributos. En esta representación se muestran los objetos contenidos dentro del rectángulo que representa al objeto que los contiene. Un ejemplo es el siguiente objeto ventana



1. Diagrama de Estados

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro.Un ejemplo en el caso de la cafetera son los estados posibles para la clase MaquinaCafe:



* 1. Estado

Identifica un periodo de tiempo del objeto (no instantáneo) en el cual el objeto esta esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos. Se representa mediante un rectángulo con los bordes redondeados, que puede tener tres compartimientos: uno para el nombre, otro para el valor característico de los atributos del objeto en ese estado y otro para las acciones que se realizan al entrar, salir o estar en un estado (entry, exit o do, respectivamente). En el caso del ejemplo anterior, se tienen cuatro estados (EnFuncionamiento, SinCambio, SinIngredientes, MalFuncionamiento) , en los cuales se desarrollan ciertas acciones al entrar; por ejemplo, al entrar al estado SinIngredientes se debe realizar la accion "Indicador SinIngredientes en On".

Se marcan también los estados iniciales y finales mediante los símbolos Estado inicial y Ejemplo de estado final, respectivamente.

* 1. Eventos

Es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una de varias cosas:

\*Condición que toma el valor de verdadero o falso

\*Recepción de una señal de otro objeto en el modelo

\*Recepción de un mensaje

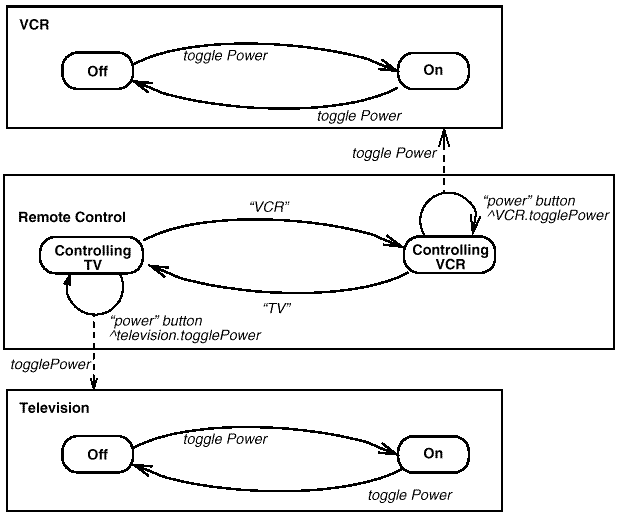
\*Paso de cierto período de tiempo, después de entrar al estado o de cierta hora y fecha particular

El nombre de un evento tiene alcance dentro del paquete en el cual está definido, no es local a la clase que lo nombre.

En el caso del ejemplo anterior se encuentra nombrado en varias transiciones el evento **userInput**, que recibe como parámetro un **Button**, para indicar el botón que ha sido presionado por el usuario de la máquina de café.

* 1. Envío de mensajes

Además de mostrar y transición de estados por medio de eventos, puede representarse el momento en el cual se envían mensajes a otros objetos. Esto se realiza mediante una línea punteada dirigida al diagrama de estados del objeto receptor del mensaje. Si tomamos como ejemplo un control remoto que puede enviar órdenes de encender o apagar al televisor o a la videograbadora se puede obtener un diagrama de estados como el siguiente:



Los tres aparatos tienen diagramas de estados separados y algunas de las transiciones del control remoto causan el envío de mensajes (*togglePower*) a los otros aparatos.

* 1. Transición simple

Una transición simple es una relación entre dos estados que indica que un objeto en el primer estado puede entrar al segundo estado y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas. Se representa como una línea sólida entre dos estados, que puede venir acompañada de un texto con el siguiente formato:

event-signature [guard-condition] /action-expression send-clause

event-signature es la descripción del evento que da a lugar la transición, guard-condition son las condiciones adicionales al evento necesarias para que la transición ocurra, action-expression es un mensaje al objeto o a otro objeto que se ejecuta como resultado de la transición y el cambio de estado y send-clause son acciones adicionales que se ejcutan con el cambio de estado, por ejemplo, el envio de eventos a otros paquetes o clases.

En el caso del ejemplo inicial de esta hoja se tiene una transición entre los estados IntroduciendoMoneda y SeleccionadoAzucaryProducto que tiene una transición con el siguiente detalle:

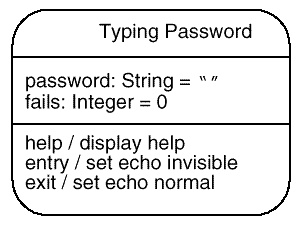
userInput( Button ) | [TodoOk=true} / MostrarNivelAzucar, MostrarProducto

El evento que dispara el cambio de estado es userInput( Button). Se requiere como condición adicional que no se haya detectado ninguna falla (TodoOk = true) y se ejecuta MostrarNivelAzucar y MostrarProducto, que deberían ser ejecutables por el objeto al cual pertenece el diagrama.

* 1. Transición interna

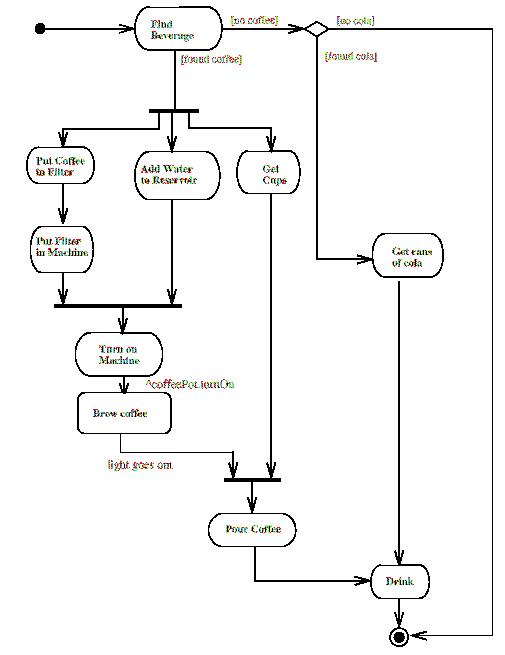
Es una transición que permanece en el mismo estado, en vez de involucrar dos estados distintos. Representa un evento que no causa cambio de estado. Se denota como una cadena adicional en el compartimiento de acciones del estado.

Supongamos el estado de una interfaz pidiendo password al usuario. En este caso puede tenerse una transición interna que muestre una ayuda al usuario. Esta transición se muestra en el siguiente diagrama con la cadena "help / display help " dentro del cuerpo del estado.



* 1. [Diagrama de Actividades](http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/actividades01.html)

Un diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la accion ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos. Se presenta a continuación un ejemplo de diagrama de actividades para un mensaje de un objeto. Generalmente modelan los pasos de un algoritmo.



* 1. Estado de acción

Representa un estado con acción interna, con por lo menos una transición que identifica la culminación de la acción (por medio de un evento implícito). No deben tener transiciones internas ni transiciones basadas en eventos (Si este es el caso, represéntelo en un diagrama de estados). Permite modelar un paso dentro del algoritmo. Se representan por un rectángulo con bordes redondeados. 

* 1. Transiciones

Las flechas entre estados representan transiciones con evento implícito. Pueden tener una condición en el caso de decisiones.

* 1. Decisiones

Se representa mediante una transición múltiple que sale de un estado, donde cada camino tiene un label distinto. Se representa mediante un diamante al cual llega la transición del estado inicial y del cual salen las múltiples transiciones de los estados finales. Un ejemplo se ve en la figura cuando *no hay cafe* y se toma una decisión entre *hay cola* o *no hay cola*.

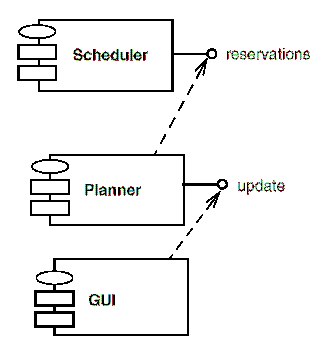
1. [Diagrama de Implementación](http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/implementacion01.html)

Un diagrama de implementación muestra la estructura del código (Diagrama de componentes) y la estructura del sistema en ejecución (Diagrama de ejecución).

1. Diagrama de Componentes

Un diagrama de componentes muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables. Los componentes software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución. Se consideran en este tipo de diagramas solo tipos de componentes. Instancias específicas se encuentran en el diagrama de ejecución.

Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (generalmente de compilación). Puede mostrar también contenencia de entre componentes software e interfaces soportadas. Un ejemplo es el siguiente:



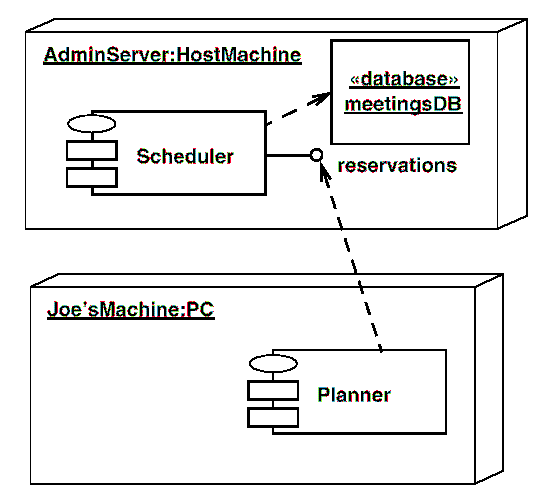
En este caso tenemos tres componentes, GUI dependiendo de la interfaz *update* provista por *Planner*, *Planner*dependiendo de la interfaz *reservations*provista por *Scheduler*.

1. Diagrama de Ejecución

Un diagrama de ejecución muestra la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software, procesos y objetos que se ejecutan en ellos. Instancias de los componentes software representan manifestaciones en tiempo de ejecución del código. Componentes que solo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes.

Un diagrama de ejecución es un grafo de nodos conectados por asociaciones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (un caso particular de un objeto). Las instancias de componentes software pueden estar unidos por relaciones de dependencia, posiblemente a interfaces.

Un ejemplo de diagrama de ejecución es el siguiente:



En este caso se tienen dos nodos, *AdminServer*y *Joe'sMachine*. *AdminServer* contiene la instancia del componente *Scheduler*y un objeto activo (proceso) Denominado *meetingsDB*. En*Joe'sMachine* se encuentra la instancia del componente software *Planner*, que depende de la interfaz *reservations*, definida por *Scheduler*.

* 1. Nodos

Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento. Pueden representarse instancias o tipos de nodos. Se representa como un cubo 3D en los diagramas de implementación.

* 1. Componentes

Un componente representa una unidad de código (fuente, binario o ejecutable) que permite mostrar las dependencias en tiempo de compilación y ejecución. Las instancias de componentes de software muestran unidades de software en tiempo de ejecución y generalmente ayudan a identificar sus dependencias y su localización en nodos.  Pueden mostrar también que interfaces implementan y qué objetos contienen. Su representación es un rectángulo atravesado por una elipse y dos rectángulos más pequeños.

Un ejemplo de componente que implementa dos interfaces es

