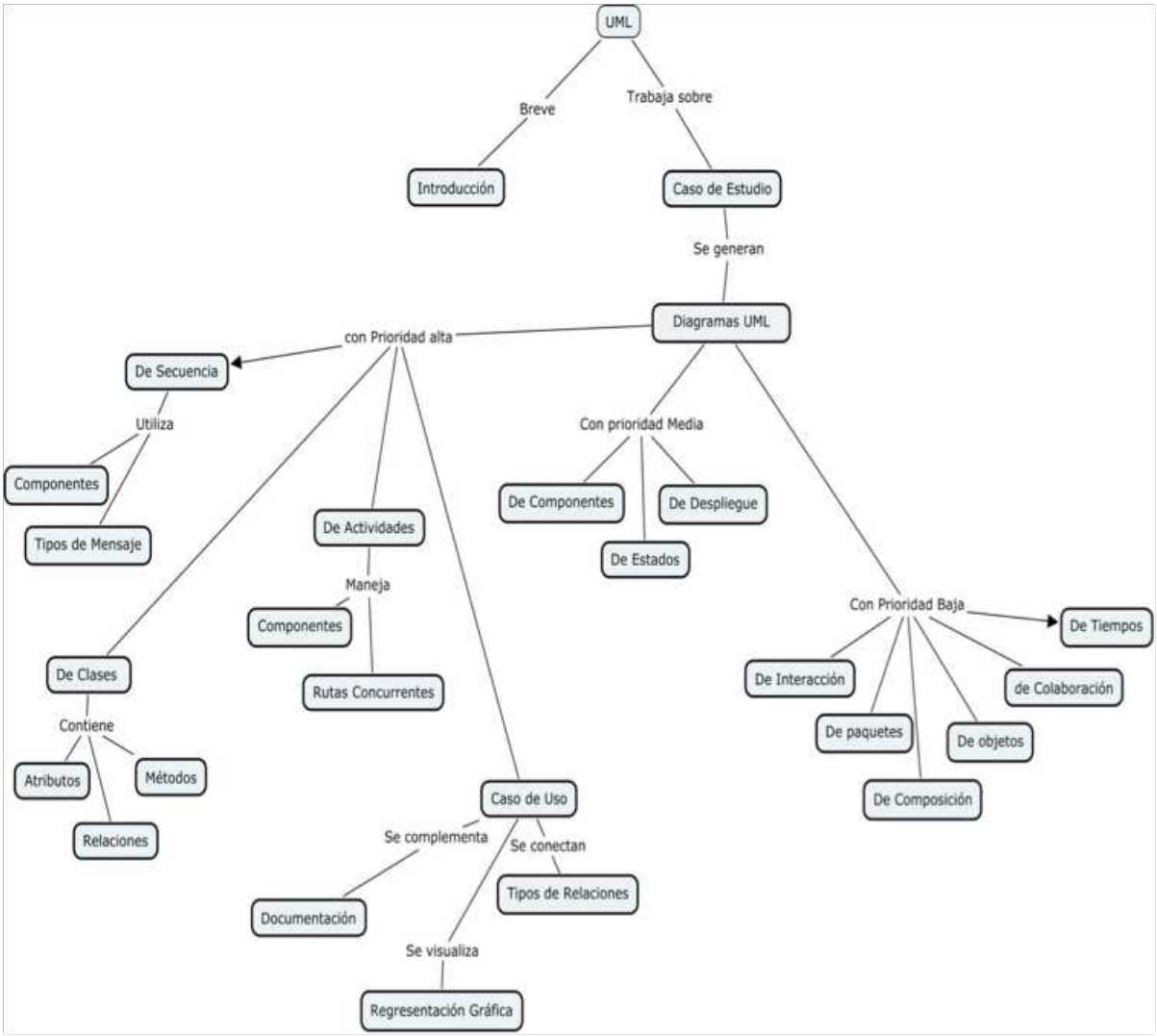
INTRODUCCIÓN AL LENGUAJE DE MODELADO UNIFICADO (UML)

Mapa conceptual
Introducción al Lenguaje de Modelado Unificado (UML)



INTRODUCCIÓN AL LENGUAJE DE MODELADO UNIFICADO (UML)

INTRODUCCIÓN

Imaginemos si tuviéramos que crear una obra de teatro con un Alemán, un Chino, un Árabe y un Ruso y cada uno solo hablara en su lenguaje nativo sin traductor alguno?. Sería un poco complicado no?, ahora, pensemos que somos los lectores de la obra de teatro y no dominamos bien esos lenguajes, seguramente sería muy difícil entender dicha obra. Pues cuando surgió la Programación Orientada a Objetos, se crearon diferentes tipos de modelos, símbolos y lenguajes para representar los sistemas que se desarrollaban. Sin lugar a dudas esto originó una anarquía en los desarrolladores de software en su momento; buscando solucionar este problema nace el UML (Unified Modeling Language – Lenguaje de Modelado Unificado).

UML es un conjunto de herramientas que permite modelar (analizar y diseñar) sistemas orientados a objetos. Así mismo proporciona de alguna manera muy particular a los programadores, desarrolladores, analistas y diseñadores de aplicaciones informáticas, las reglas técnicas que permiten representar de forma gráfica el comportamiento y las estructuras que forman parte de un determinado sistema.

"El UML (Lenguaje Unificado de Modelado) es una de las herramientas más emocionantes en el mundo actual del desarrollo de sistemas. Esto se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas".

(Joseph Schmuller. Aprendiendo UML en 24 horas pag.24)

1. INTRODUCCIÓN A UML

El lenguaje de modelado (UML) permite a través de sus elementos gráficos representar flujos de trabajo diversos para proyectos de tecnología informática, hardware, electrónica, robótica, hidráulica, procesos industriales, empresariales y gerenciales de cualquier nivel en una organización sin importar su tamaño o naturaleza.

UML es un lenguaje para visualizar los elementos de un gran sistema software, facilitando:

- La comunicación entre los participantes (incluidas herramientas) en el desarrollo.
- La comprensión de las soluciones (notación gráfica).
- El mantenimiento de las soluciones conceptuales a lo largo del tiempo (documentación).

Algunos componentes particulares de UML son:

Partes (elementos):
Son todos los objetos, cosas, personas, animales, sistemas, subsistemas que pueden relacionarse.

Acciones (relaciones):
A través de las acciones las partes se pueden relacionar. Estas pueden ser (correr, vender, cantar, comer, bailar o acciones abstractas como querer, sentir) hacen que los sistemas tengan funcionalidad, que adquieran vida.

Diagramas de modelado (Diseños):
Reflejan gráficamente el comportamiento, relaciones entre los elementos intervinientes dentro de un sistema.

Todo sistema puede ser representado gráficamente mediante modelos empleando UML el cual soporta sus diseños usando diagramas que facilitan el entendimiento, trazabilidad y comprensión entre sus componentes brindando claridad y calidad en la información presentada a los diversos actores involucrados en la distribución de información que requiere ser procesada por un sistema o flujo de trabajo propio de una organización empresarial.

TIPOS DE DIAGRAMA UML

A continuación se presenta un resumen elaborado por Ambler Scott. 2003 sobre los diferentes diagramas UML:

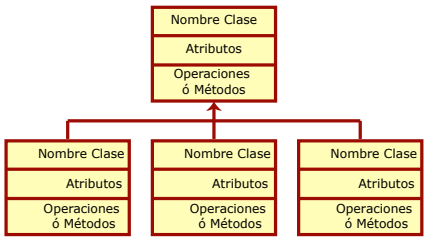
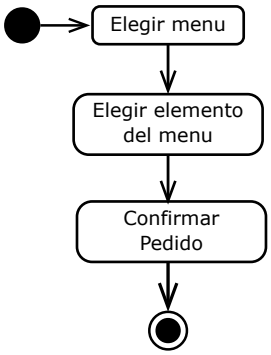
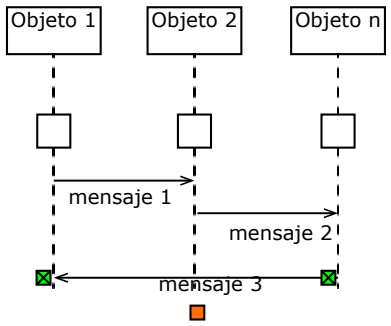
Diagrama	Descripción	Prioridad
Diagrama de Clases 	Muestra una colección de elementos de modelado declarativo (estáticos), tales como clases, tipos y sus contenidos y relaciones.	**Alta**
Diagrama de Actividades 	Representa los procesos de negocios de alto nivel, incluidos el flujo de datos. También puede utilizarse para modelar lógica compleja y/o paralela dentro de un sistema.	**Alta**
Diagrama de Secuencias 	Un diagrama que representa una interacción, poniendo el foco en la secuencia de los mensajes que se intercambian, junto con sus correspondientes ocurrencias de eventos en las Líneas de Vida.	**Alta**

Diagrama	Descripción	Prioridad
Diagrama de Casos de Uso	Un diagrama que muestra las relaciones entre los actores y el sujeto (sistema), y los casos de uso.	**Media**
Diagrama de Componentes	Representa los componentes que conforman una aplicación, sistema o empresa. Los componentes, sus relaciones, interacciones y sus interfaces públicas.	**Media**
Diagrama de Máquinas de Estado	Un diagrama de Máquina de Estados ilustra cómo un elemento, muchas veces una clase, se puede mover entre estados que clasifican su comportamiento, de acuerdo con disparadores de transiciones, guardias de restricciones y otros aspectos de lso diagramas de Máquinas de Estados, que representan y explican el movimiento y el comportamiento.	**Media**
Diagrama de Despliegue Físico	Un diagrama de despliegue físico muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos y la construcción interna puede ser representada por nodos o artefactos embebidos. Como los artefactos se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue.	**Media**
Diagrama de Objetos	Un diagrama que presenta los objetos y sus relaciones en un punto en el tiempo. Un diagrama de objetos se puede considerar como un caso especial de un diagrama de clases o un diagrama de comunicaciones	**Baja**

Diagrama	Descripción	Prioridad
Diagrama de Estructura de Composición	Representa la estructura interna de un clasificador (tal como una clase, un componente o un caso de uso), incluyendo los puntos de interacción de clasificador con otras partes del sistema.	**Baja**
Diagrama de Paquetes	Un diagrama que representa cómo se organizan los elementos de modelado en paquetes y las dependencias entre ellos, incluyendo importaciones y extensiones de paquetes.	**Baja**
Diagrama de Comunicaciones (anteriormente: Diagrama de Colaboraciones)	Los diagramas de Revisión de la Interacción enfocan la revisión del flujo de control, donde los nodos son Interacciones u Ocurrencias de Interacciones. Las líneas de vida los mensajes no aparecen en este nivel de revisión.	**Baja**
Diagrama de Tiempos	El propósito primario del diagrama de tiempos es mostrar los cambios en el estado o la condición de una línea de vida (representando una Instancia de un Clasificador o un Rol de un clasificador) a lo largo del tiempo lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o estímulos aceptados. Los eventos que se reciben se anotan, a medida que muestran cuándo se desea mostrar el evento que causa el cambio en la condición o en el estado.	**Baja**

Para una mejor interpretación de los diferentes diagramas, se debe tomar como referencia la situación que se presenta a continuación correspondiente a la gestión de información en un centro médico, los ejemplos y elementos de diagrama construidos se basan en él.

El propietario de un centro médico de su ciudad, requiere con urgencia la construcción de un sistema de información que le permita administrar los datos básicos de sus pacientes, tratamientos, citas y gestión de reportes de tal manera que al ejemplificar el sistema se pueda fijar claramente el límite de este.

2. DIAGRAMAS INICIALES

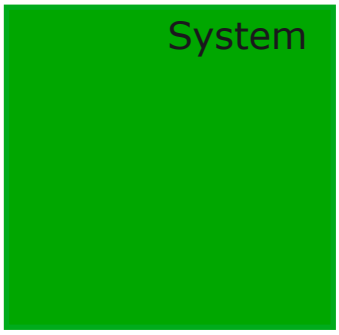
A continuación se relacionan los diagramas iniciales, utilizados en la fase de análisis de un sistema de información.

2.1 Casos de Uso

Al momento de desarrollar un proyecto se debe pensar en cuáles serán las principales funcionalidades que el software debe permitir llevar a cabo y quiénes serán los que podrán ejecutar dichas funcionalidades. La identificación de estos elementos se puede visualizar de manera efectiva a través de la elaboración de diagramas de Casos de Uso. Estos diagramas que son elaborados durante las etapas iniciales de un proyecto se convierten en un referente para cada una de las etapas siguientes del desarrollo del proyecto.

2.1.1. Componentes.

Sistema: Se representa mediante un rectángulo. Este se emplea para delimitar gráficamente lo que se encuentra por dentro del sistema (los casos de uso) y lo que está por fuera del sistema (los actores).



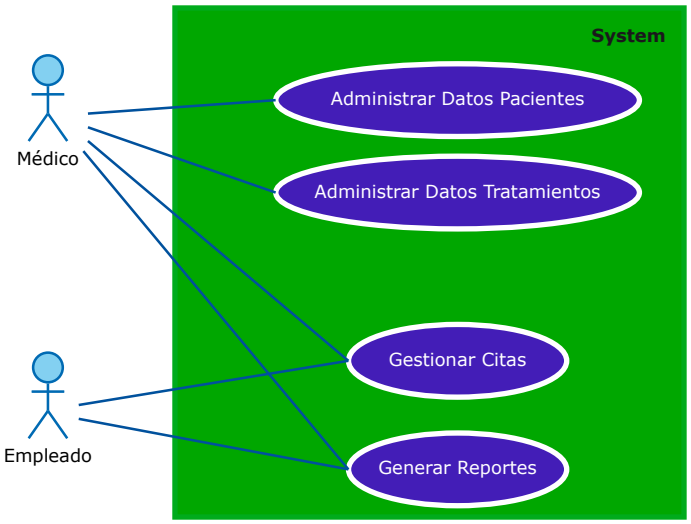
Actor: Se representa mediante un "hombre de palo". Este se emplea para indicar el tipo de usuario del sistema que podrá ejecutar alguna función (caso de uso) en el mismo.

Caso de Uso: Se representa mediante un óvalo e indica una función que el sistema debe proveer.

Para ejemplificar un proceso se puede emplear un verbo conjugado en infinitivo y que represente la función a realizar (Administrar, Gestionar, Registrar, entre otros).

- Administrar datos pacientes.
- Administrar datos tratamientos.
- Gestionar citas.
- Generar Reportes.

2.1.2. Representación gráfica.



Identificación de Casos de Uso: En el ejemplo anterior se observan los casos de uso identificados en el sistema, es decir, las funcionalidades que el sistema va a proveer (Administrar datos pacientes, Administrar datos tratamientos, Gestionar citas, Generar reportes)

Identificación de Actores: Los actores son los usuarios que podrán ejecutar los casos de uso, en el ejemplo anterior, se identificaron dos actores (Médico y Empleado).

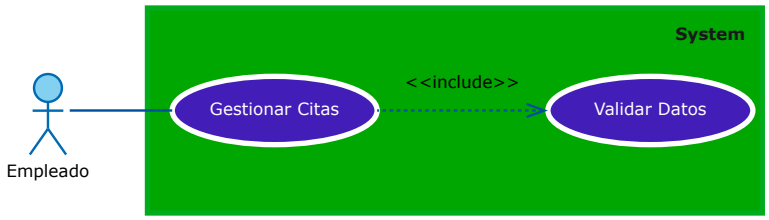
Las líneas que van del actor al caso de uso se denominan Asociación y sirven para determinar cuáles Casos de uso lleva a cabo un determinado Actor.

2.1.3. Tipos de relaciones.

Asociación: relación que se da entre los actores y el caso de uso. Esta relación indica que el actor lleva a cabo el caso de uso.

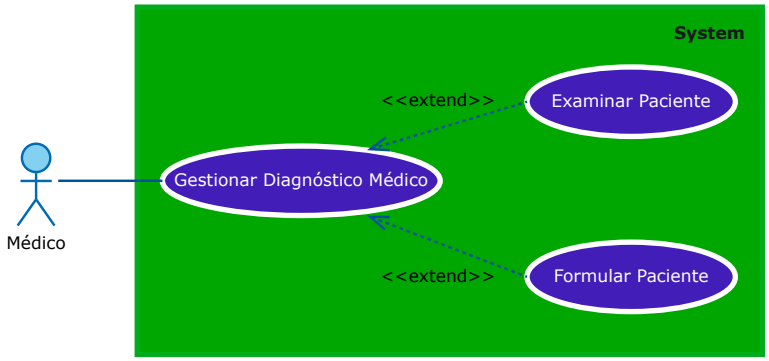
El ejemplo anterior indica que el Médico dentro del sistema Sismédico puede llevar a cabo la funcionalidad de Administrar datos Pacientes.

Incluido ó <<uses>> ó <<include>>: relación que se da entre casos de uso donde uno termina incluyendo al otro (describe el comportamiento de uno en otro).



El ejemplo anterior indica que para Gestionar una cita se debe validar datos, es decir, el caso de uso Gestionar cita "usa" al caso de uso validar datos.

Extendido ó <<extend>>: relación que se da entre casos de uso donde se evidencia que un caso de uso es la generalización o especialización de otro.



El ejemplo anterior indica que al gestionar un diagnóstico médico, se puede examinar un paciente o formular un paciente, es decir, los casos de uso formular a paciente y examinar a paciente se extienden del caso de uso gestionar diagnóstico médico.

2.1.4. Documentación.

La técnica de casos de uso requiere además de construir el diagrama de Casos de Uso, la descripción de los mismos. Esta descripción permite detallar el flujo de eventos que se da entre el Sistema y el Actor para llevar a cabo el Caso de Uso. A continuación se presenta el formato diligenciado de acuerdo con el ejemplo del centro médico.

Casos de Uso	Administrar datos pacientes	
Descripción	El comportamiento del sistema deberá describir el paso a paso del caso de uso cuando el personal encargado de gestionar datos del paciente inicie el ingreso de estos.	
Precondición	El paciente no se encuentra ingresado al sistema y tiene la documentación necesaria para poder ser ingresado al sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El personal médico ingresa al sistema para registrar el nuevo paciente.
	2	El sistema carga formulario para registro de datos del paciente así: identificación, nombre(s), apellido(s), dirección, teléfono, estrato, tipo de RH, sexo, acudiente.
	3	El personal médico ingresa los datos suministrados por el paciente y ejecuta la acción en el sistema.
	4	El sistema almacena los datos suministrados por el personal médico, imprime el carnet del Sistema General de Seguridad Social en Salud, el sistema comunica al personal médico que el proceso ha terminado de manera exitosa.
	5	El personal médico genera reporte del sistema de registro al nuevo paciente, mediante la expedición del carnet o constancia de inscripción al sistema de Seguridad Social en Salud.
Post Condición	El paciente se encuentra registrado en el Sistema General de Seguridad Social en Salud, su historial clínico es nuevo.	
Excepciones	**Paso**	**Acción**
	3	Si el sistema detecta la duplicación de un paciente registrado con la identificación que se registra, procede a informar al personal médico, estos deben modificar y/o actualizar la información que sea necesaria y continuar el caso de uso.
	3	Si el personal médico cancela el registro del paciente se termina el caso de uso.

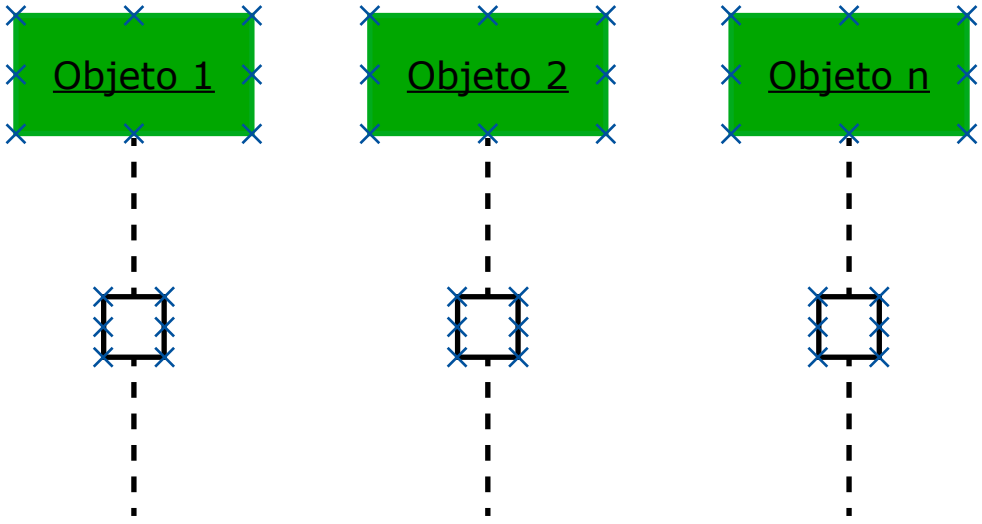
2.2. DIAGRAMA DE SECUENCIA

Los diagramas de secuencia se utilizan para describir el funcionamiento interno del sistema desde el punto de vista de la implementación y son parte fundamental de la vista de interacción de los diagramas UML. Estos son empleados para definir el comportamiento interno de un Caso de Uso y los mensajes que se representan en el mismo sirven de base para definir los métodos de las clases en un Diagrama de Clases.

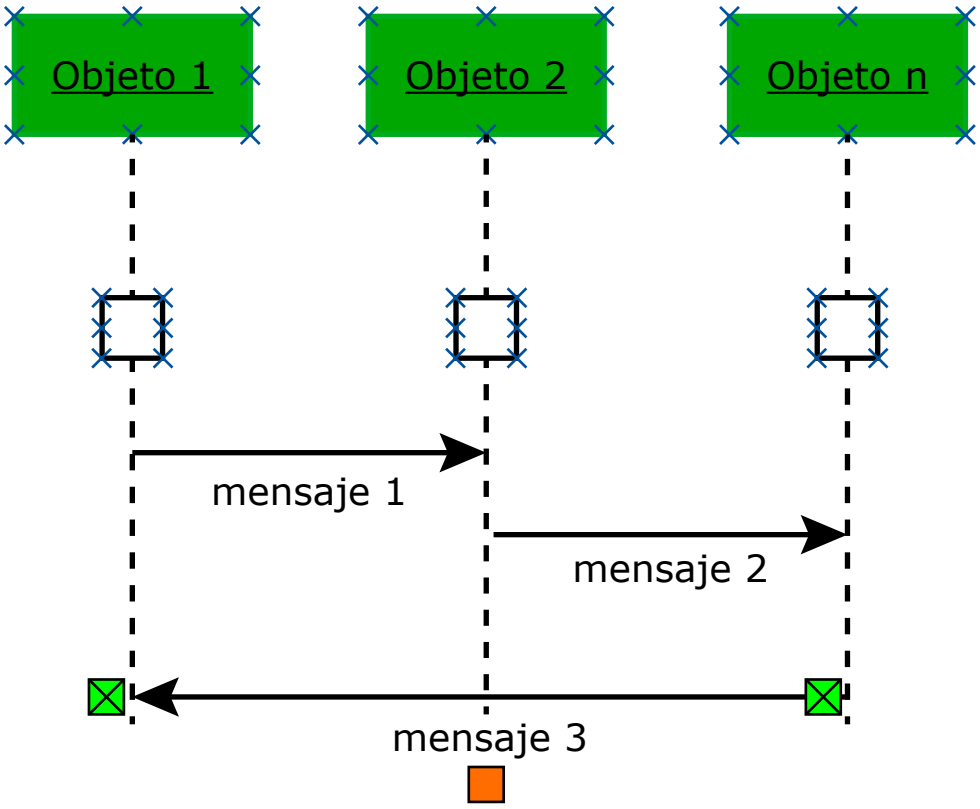
Según James Rumbaugh et al. (Pearson Eduation 2002)
Los objetos obran recíprocamente para implementar comportamiento. Esta interacción se puede describir de dos maneras complementarias, una de ellas se centra en los objetos individuales y la otra en una colección de objetos cooperantes.

2.2.1 Representación Gráfica

Un diagrama de secuencia representa una interacción como un gráfico bidimensional. La dimensión vertical es el eje de tiempo, que avanza hacia abajo de la página. La dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración. Cada rol de clasificador se representa mediante una columna vertical-línea de vida. Durante el tiempo que existe un objeto, el rol se muestra por una línea discontinua. Durante el tiempo que dura una activación de un procedimiento en el objeto, la línea de vida se dibuja como una línea doble.



Se muestra un mensaje como una flecha desde la línea de vida de un objeto a la del otro. Las flechas se organizan en el diagrama en orden cronológico hacia abajo.



Mientras que el diagrama de casos de uso permite el modelado de una vista del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

2.2.2. Tipos de mensajes.

Existen dos tipos de mensajes: **sincrónicos y asincrónicos.**

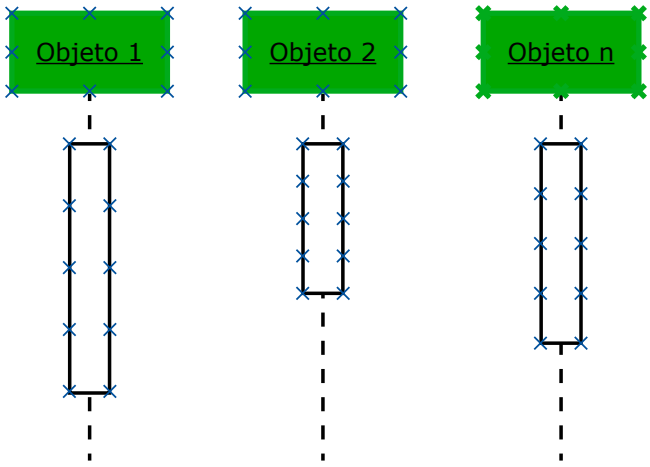
Los mensajes sincrónicos se corresponden con llamadas a métodos del objeto que recibe el mensaje. El objeto que envía el mensaje queda bloqueado hasta que termina la llamada. Este tipo de mensajes se representan con flechas con la cabeza llena.

Los mensajes asincrónicos terminan inmediatamente, y crean un nuevo hilo de ejecución dentro de la secuencia. Se representan con flechas con la cabeza abierta. También se representa la respuesta a un mensaje con una flecha discontinua.

Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria. Durante el análisis inicial, el modelador típicamente coloca el nombre 'business' de un mensaje en la línea del mensaje. Más tarde, durante el diseño, el nombre 'business' es reemplazado con el nombre del método que está siendo llamado por un objeto en el otro. El método llamado, o invocado, pertenece a la definición de la clase instanciada por el objeto en la recepción final del mensaje.

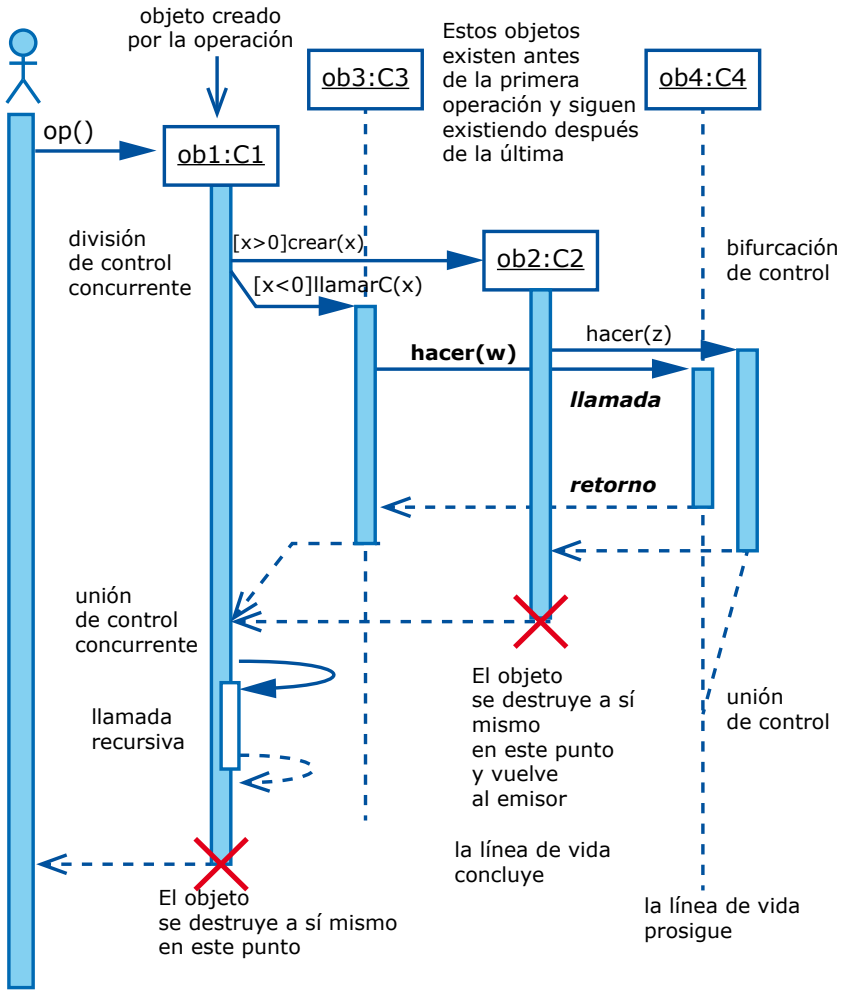
Activación

Una activación se representa en un diagrama de secuencia utilizando un rectángulo alto y delgado (una barra vertical hueca) cuya parte superior está alineada con el momento de su iniciación y cuya parte inferior lo está con el momento de su finalización. La operación a realizar se representa mediante una etiqueta de texto al lado del símbolo de activación o en el margen izquierdo, dependiendo del estilo.



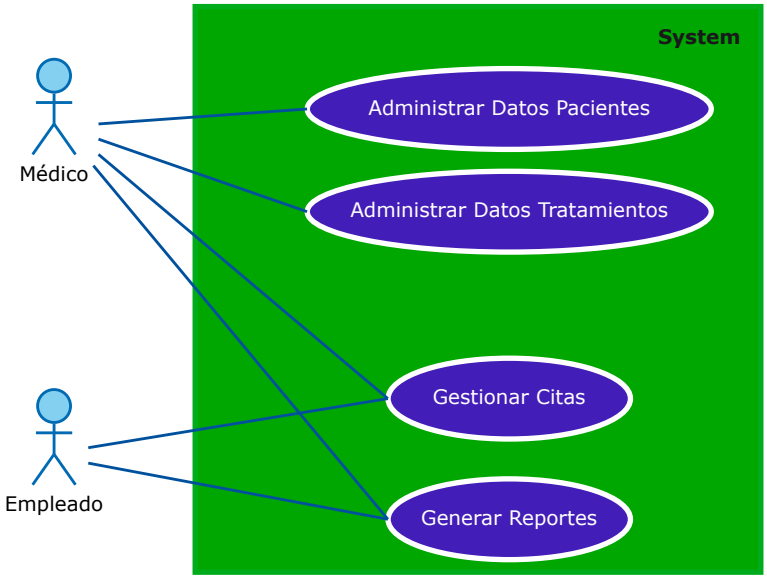
Si hay actividad concurrente entre varios objetos, la activación muestra la ejecución de un objeto concurrente. A menos que los objetos se comuniquen, las actividades concurrentes son independientes, y no es relevante reflejar sus tiempos relativos de ejecución.

Cuando se trata de código de procedimientos, una activación muestra o bien la duración durante la cual un procedimiento está activo en el objeto o bien el tiempo durante el que está activo un procedimiento subordinado llamado por el procedimiento original, posiblemente en algún otro objeto. En otras palabras, se muestran simultáneamente todas las activaciones de procedimientos anidados. Este conjunto de activaciones anidadas simultáneas forma el marco de pila de la computación en un computador convencional. En caso de una segunda llamada a un objeto con una activación existente, el segundo símbolo de activación se dibuja ligeramente a la derecha del primero, de forma que den la sensación de estar apilados.

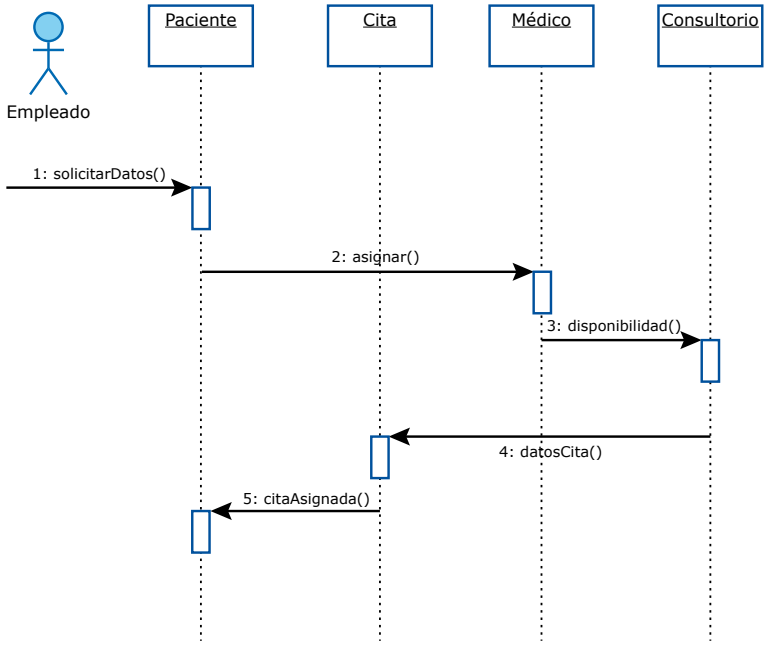


Las llamadas apiladas pueden tener cualquier nivel de anidamiento. Las llamadas pueden ser a la misma operación (llamada recursiva) o a diferentes operaciones del mismo objeto.

Retomando el ejemplo de caso de uso presentado anteriormente, su respectivo diagrama de Secuencia se presenta a continuación:



A continuación se presenta el diagrama de secuencia correspondiente con el caso de uso Gestionar citas



2.3. DIAGRAMA DE COMUNICACIÓN (Colaboración)

Lo primero que se debe aclarar es que hablar de un diagrama de Comunicación o diagrama de Colaboración es exactamente igual, la diferencia solo está en el nombre, pues en la versiones 1 de UML se denomina diagrama de Colaboración y a partir de la versión 2.0 se llama diagrama de Comunicación. Este diagrama forma parte de los diagramas de Interacción del UML, esto significa que tienen en cuenta el tiempo como un elemento importante dentro de la estructura del diagrama.

Un diagrama de Comunicación es muy similar al diagrama de Secuencia, de hecho la conversión de uno al otro es un proceso muy simple. La diferencia está en que el diagrama de Secuencia hace énfasis en el orden y secuencia en que se emiten mensajes entre los objetos para cumplir con un determinado proceso e incluye dentro de sus componentes la línea de vida para cada objeto, mientras que el diagrama de Comunicación hace énfasis en la relación entre los objetos para satisfacer una operación, generando una visión espacial del sistema en la ejecución de un proceso.

2.3.1. Representación Gráfica

Los componentes gráficos del diagrama de Comunicación son los mismos del diagrama de Secuencia excepto por el no uso de la línea de vida de los objetos y la necesidad de incluir un número en los mensajes que identifique el orden en que se deben ejecutar.

En este diagrama se puede aprovechar el espacio, ya que no existe la condición de utilizar el espacio de izquierda a derecha y de arriba abajo, la numeración de los mensajes determinará el orden en que estos se ejecutan.

Para ilustrar el uso de diagramas de Comunicación, tomaremos el ejemplo del diagrama de Secuencia que se ha venido trabajando como caso de estudio

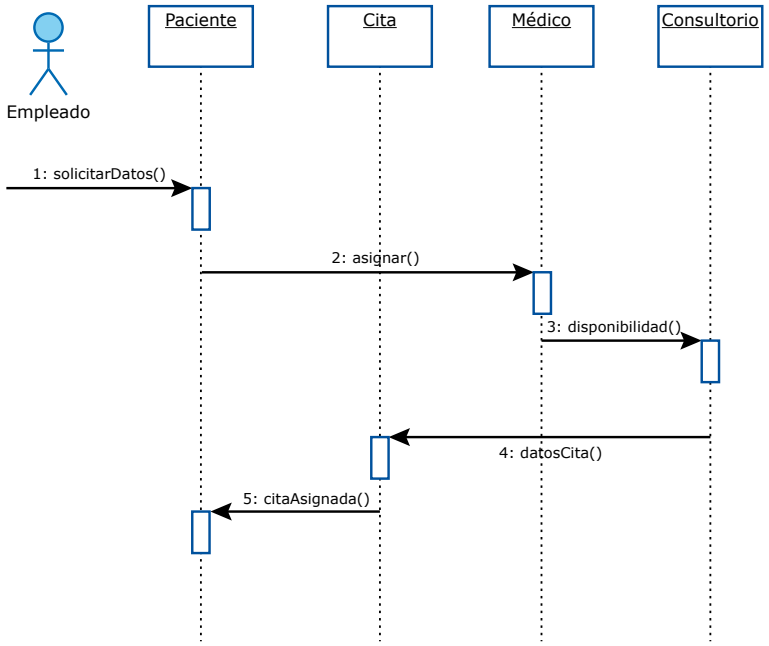
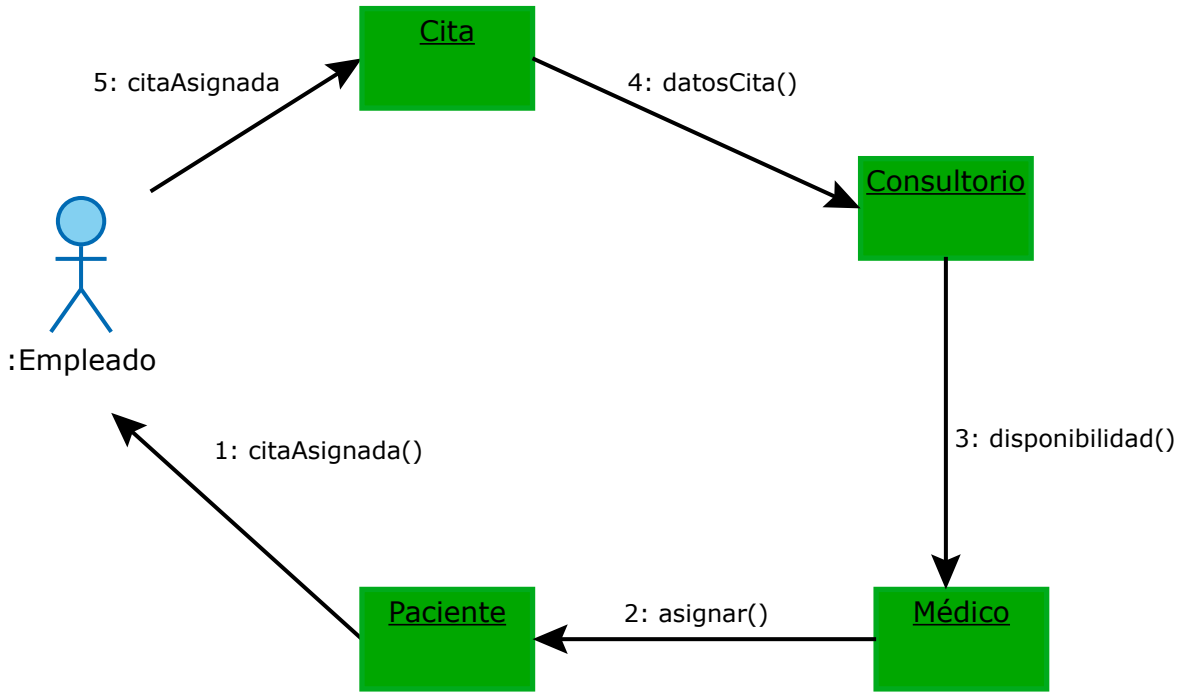


Diagrama de Secuencia asociado al caso de uso: Gestionar citas

El correspondiente diagrama de Comunicación sería:



2.4. DIAGRAMA DE ACTIVIDADES

Un diagrama de actividades es un diagrama complementario del UML que se emplea para definir la secuencia de pasos lógicos que se deben seguir para cumplir con una determinada función.

Este diagrama es similar al Diagrama de Flujo que se emplea para representar gráficamente un algoritmo, teniendo como principal diferencia que el diagrama de actividades permite la ejecución de actividades (pasos) en paralelo. Dentro del mundo de la orientación a objetos y el UML, un diagrama de actividades es usado comúnmente para representar los pasos lógicos requeridos para llevar a cabo un Caso de Uso.

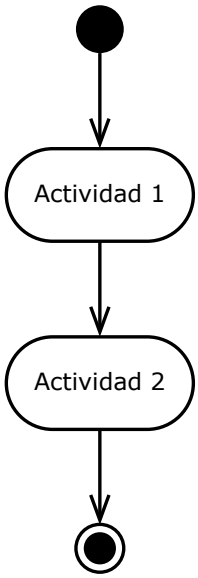
2.4.1. Componentes gráficos.

En el libro Aprendiendo UML en 24 Horas, su autor Joseph Schmuller detalla los componentes de un diagrama de Actividades mediante una serie de ejemplos que se presentan a continuación:

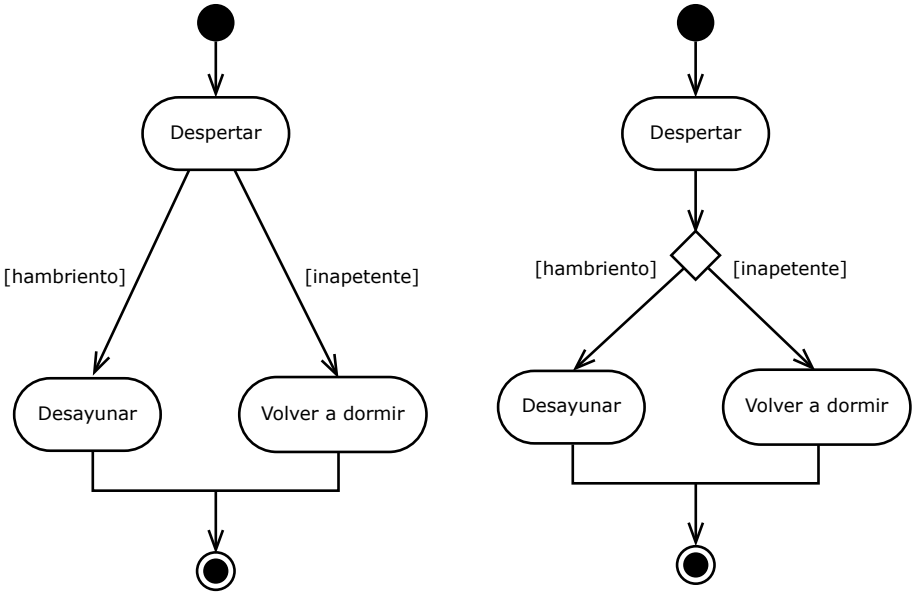
Inicio: Todo diagrama de Actividades debe poseer un único inicio, el cual se representa mediante un círculo relleno.

Fin: Indica que el proceso ha terminado. Se representa mediante un círculo que rodea al círculo relleno.

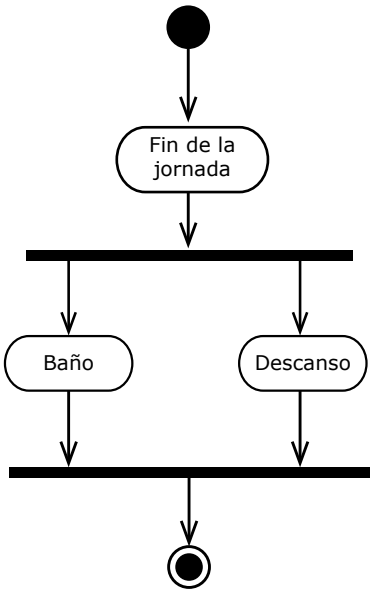
Actividades: Indica una acción requerida para llevar a cabo un proceso. Se representa mediante un rectángulo con sus vértices redondeados. Las flechas que van dirigidas de una actividad a otra se denominan transiciones.



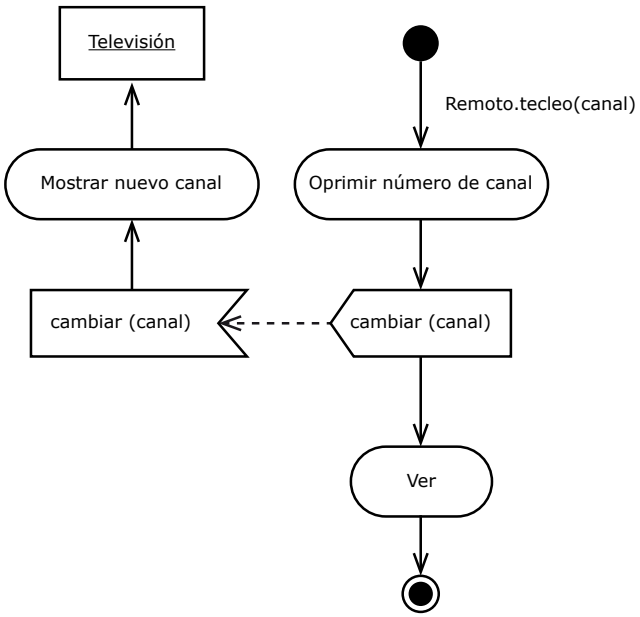
Decisión: Tal como sucede con los diagramas de flujo, en el diagrama de actividades se puede llegar a un punto donde de acuerdo con el cumplimiento o no de una condición se debe tomar por uno u otro camino. Existen dos formas de representar la decisión en un diagrama de Actividades, partiendo libremente de la actividad hacia alguno de los dos caminos o llegando a un rombo desde donde se parte hacia cualquiera de los dos caminos.



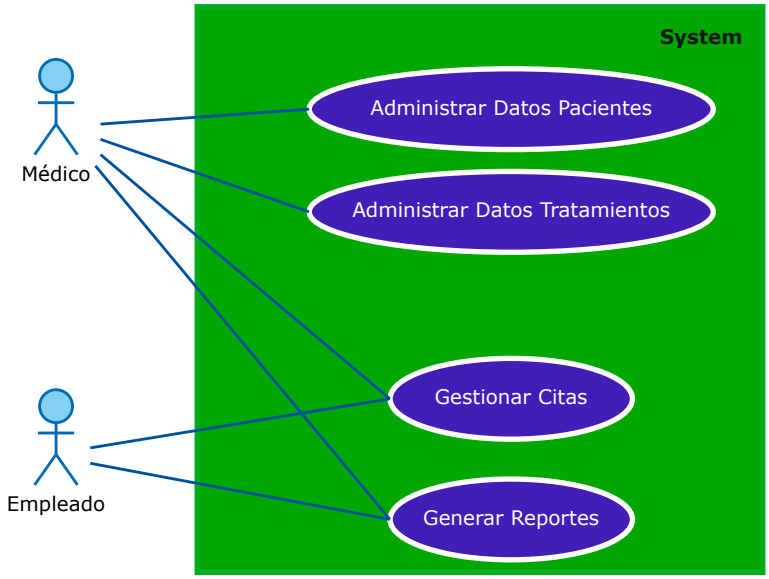
Rutas Concurrentes: Como una de las principales diferencias entre el diagrama de flujo y el de actividades es que este último permite la ejecución de actividades concurrentes, es decir, actividades que se pueden llevar a cabo en el mismo momento y después se unen a otra actividad. Tanto el inicio como el fin de una ruta concurrente se representa mediante una línea horizontal sólida.



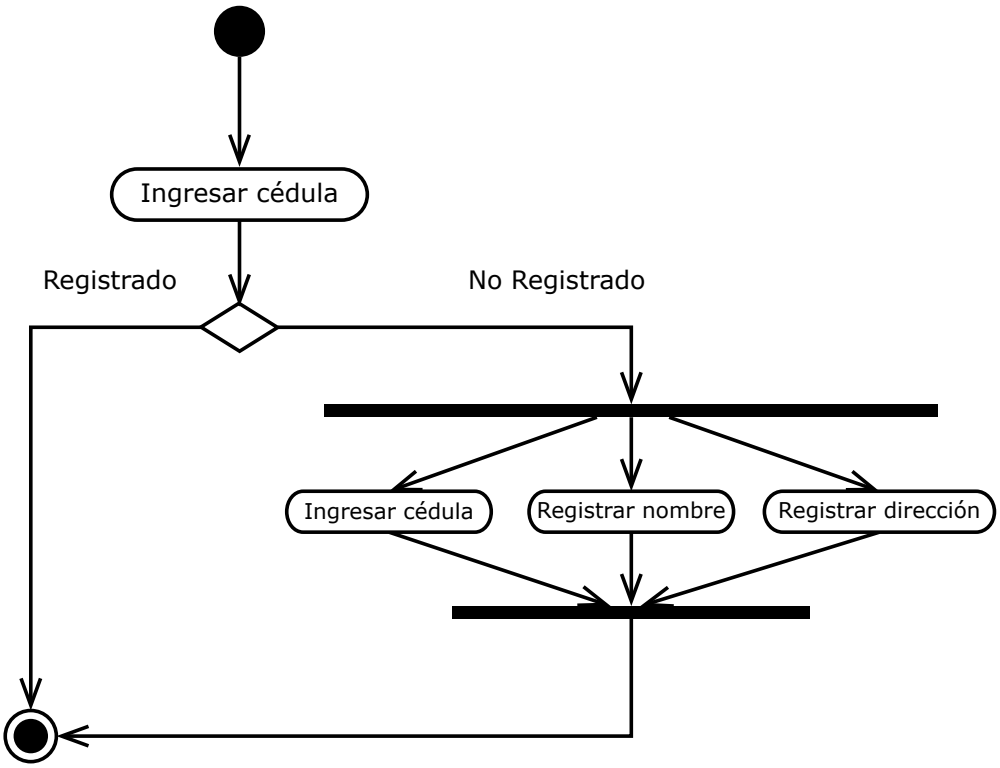
Indicaciones: Durante la secuencia de actividades es posible enviar una indicación para que al recibirla, se pueda ejecutar una actividad. Se representa mediante pentágonos que se acoplan generando la sensación de envío y recepción de la indicación.



Retomando el ejemplo empleado en la sección de Casos de Uso:



El diagrama de Actividades correspondiente con el caso de uso Administrar datos pacientes es:



2.5. DIAGRAMA DE CLASES

Para entender qué es un diagrama de clases, lo primero que se debe tener claro es qué es una clase, pues bien, una clase es un elemento importante dentro del contexto de un sistema, que puede tener información o datos valiosos y realizar acciones que sean necesarias dentro del funcionamiento del sistema.

Por ejemplo, en un software para un supermercado, seguramente los elementos más importantes sobre los cuales sea importante mantener información son los productos, los clientes, las ventas y los pedidos, en este caso se han encontrado las clases **PRODUCTO, CLIENTE, VENTA y PEDIDO**.

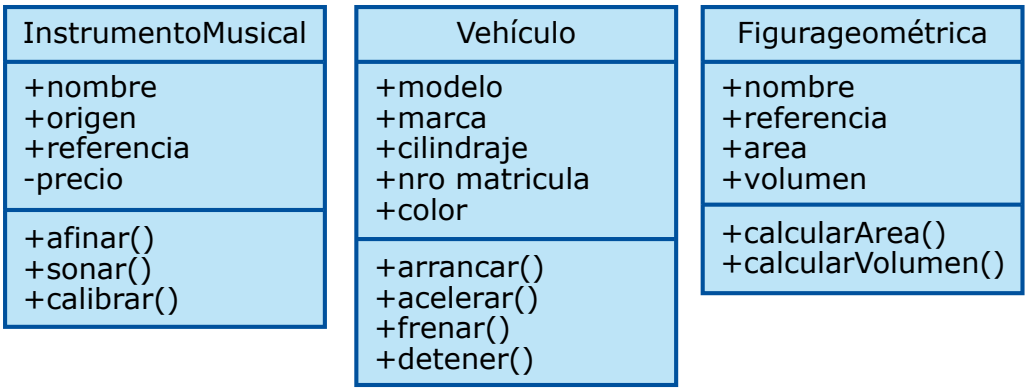
Estas clases a su vez tienen atributos (datos) y métodos (funciones), por ejemplo, la clase PRODUCTO tiene como uno de sus atributos el atributo precio y uno de sus métodos puede ser incrementarPrecio. De esta forma, a través de los atributos se puede acceder a la información de la clase y a través de los métodos se pueden ejecutar acciones sobre la clase. Estas clases se unen a otras clases a través de relaciones y así se conforma el diagrama de clases.

Los diagramas de clases describen la estructura estática que tiene un sistema, representado por el conjunto de clases que componen a este. Este tipo de diagrama UML es empleado en las fases de análisis y diseño del ciclo de vida de un proyecto de software.

2.5.1 Clases.

Clase es la unidad básica que agrupa una colección de objetos que poseen un tipo de comportamiento. Toda clases se compone de 3 elementos importantes así: Nombre de la clase, Atributos o propiedades también denominados miembros de la clase y los métodos (operaciones) o acciones propias de la clase. (Estas acciones se identifican con verbos en infinitivo).

Ejemplos de clases:



Atributos:
Los atributos o características de una Clase pueden ser de tres tipos, definen la visibilidad:

- **Public (+):** Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **Private(-):** Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden acceder).
- **Protected (#):** Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accedido por métodos de la clase además de las subclases que se deriven (ver herencia).
- **Sin modificador de acceso ( ):** Indica que el atributo será accesible desde cualquier clase que se encuentre en el mismo paquete de la clase que contiene al atributo sin modificador de acceso.

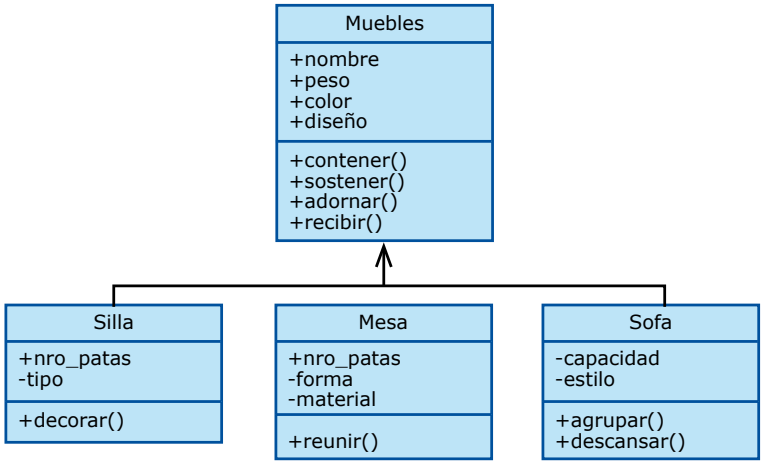
Métodos:
Los métodos u operaciones de una clase son la forma en cómo ésta interactúa con su entorno, éstos pueden tener las características:

- **Public (+):** Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **Private (-):** Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden acceder).
- **Protected (#):** Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accedido por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).
- **Sin modificador de acceso ( ):** Indica que el método será accesible desde cualquier clase que se encuentre en el mismo paquete de la clase que contiene al método sin modificador de acceso.

2.5.2 Relaciones.

Las relaciones entre las clases se dan por Herencia, Asociación, Agregación, Composición y Dependencia (uso).

- **Herencia (Especialización/Generalización)**



En el ejemplo anterior se ilustra que un Mueble puede ser una Silla, una Mesa o un Sofá.

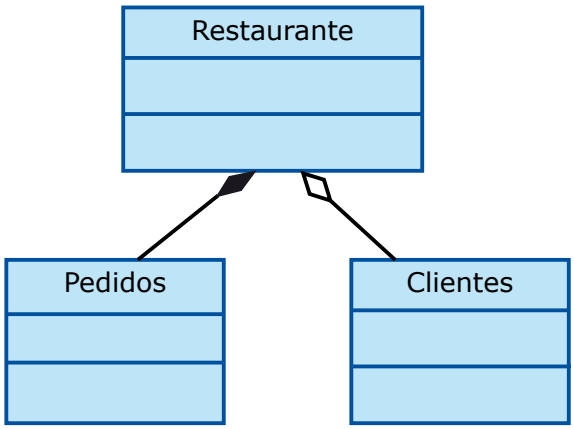
- **Agregación y composición:**

- **Composición: (Por Valor)**

Relación estática, donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada Composición (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").

- **Agregación: (Por Referencia)**

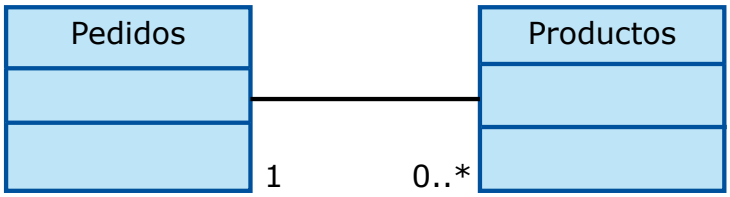
Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada Agregación (el objeto base utiliza al incluido para su funcionamiento).



El ejemplo anterior indica que el Restaurante tiene Pedidos y Clientes, sin embargo, los Pedidos requieren del Restaurante para poder existir (Composición), mientras que los Clientes no (Agregación).

- **Asociación:**

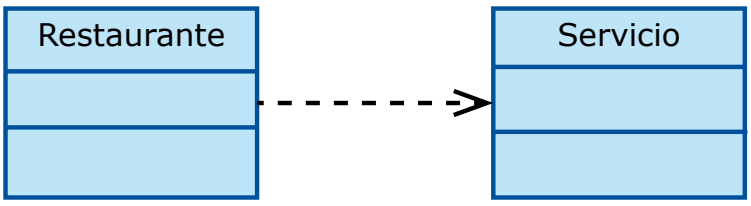
La relación entre clases, permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.



El ejemplo anterior indica que un Pedido está relacionado con cero o muchos Productos.

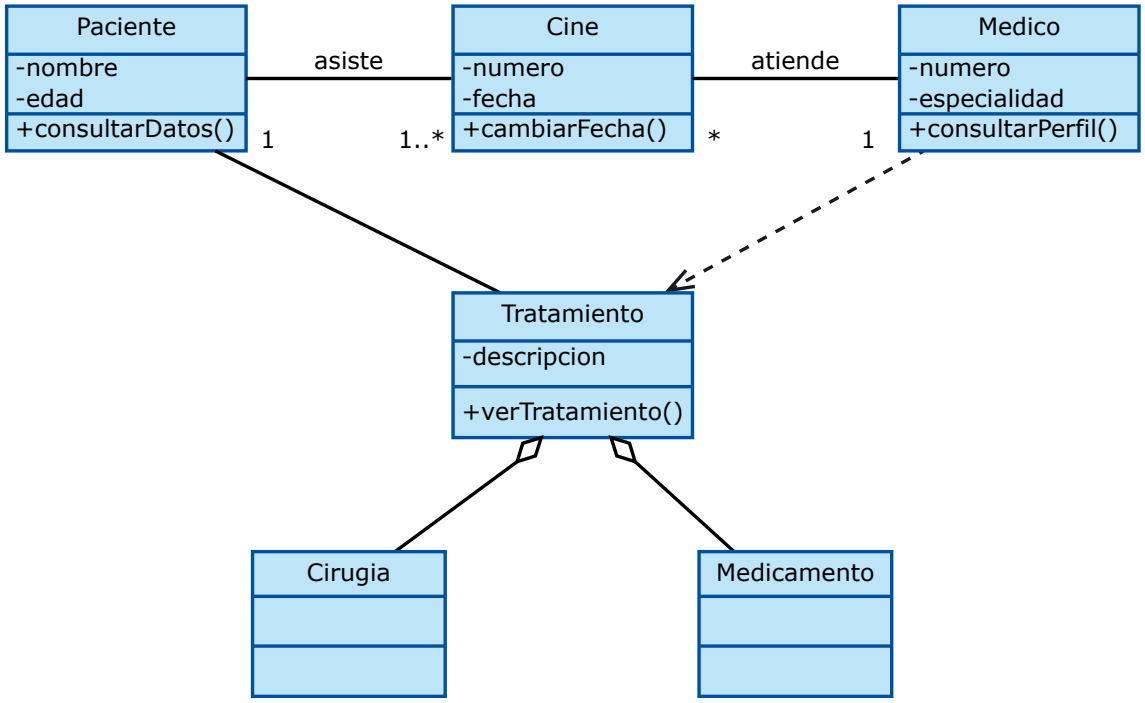
- Dependencia:

Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase).



El ejemplo anterior indica que la clase Restaurante instancia o crea objetos de la clase Servicio.

Retomando el ejemplo relacionado con la gestión citas en un centro médico, a continuación se presenta su respectivo diagrama de Clases:



2.6. DIAGRAMA DE OBJETOS

Un diagrama de Objetos es en esencia muy similar a un diagrama de clases, su propósito es modelar el sistema en un momento determinado a partir de las posibles instancias que se deriven de las clases.

Este diagrama se usa para hacer más entendible el sistema modelado ya que en lugar de presentar únicamente el nombre de la clase como algo general, se presenta el posible objeto derivado de la clase, dando mayor claridad y especificación al sistema.

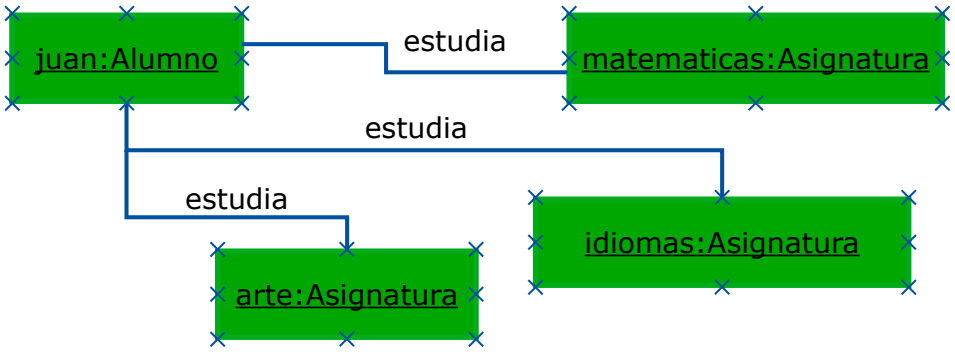
2.6.1 Representación gráfica.

En el diagrama de Objetos se emplean los mismos componentes que en el diagrama de Clases pero se omite la multiplicidad, la cual se puede observar de manera explícita cuando se incorpora una o varias instancias de una clase en el diagrama.

Los nombres de los objetos llevan la siguiente sintaxis:

nombreObjeto: NombreClase, de esta manera, si queremos presentar un objeto de la clase Asignatura, su nombre sería por ejemplo: matemáticas:Asignatura.

El siguiente ejemplo ilustra una asociación presente en un diagrama de Objetos, a través de la cual se visualiza que un alumno estudia diferentes asignaturas en una instanciación completa y específica.

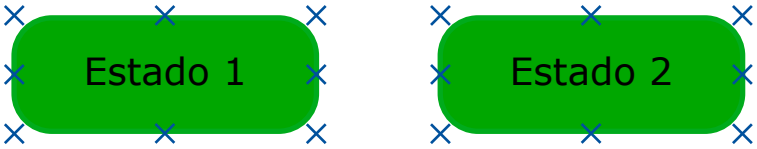


2.7. DIAGRAMA DE ESTADOS

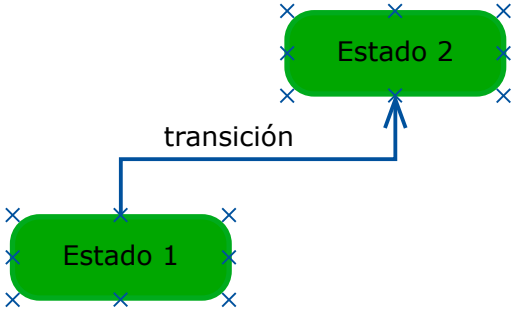
Un diagrama de transición de estados muestra el comportamiento dependiente del tiempo de un sistema de información. Representa los estados que puede tomar un componente o un sistema y muestra los eventos que implican el cambio de un estado a otro. Los dos elementos principales en estos diagramas son los estados y las posibles transiciones entre ellos. (James Rumbaugh et al. Pearson Education 2002)

2.7.1 Representación gráfica.

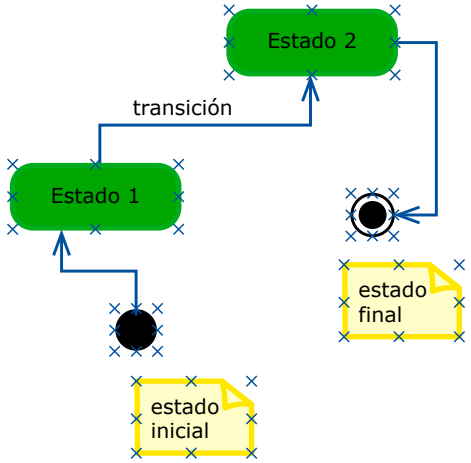
El estado de un componente o sistema representa algún comportamiento que es observable externamente y que perdura durante un periodo de tiempo finito. Viene dado por el valor de uno o varios atributos que lo caracterizan en un momento dado.



Una transición es un cambio de estado producido por un evento y refleja los posibles caminos para llegar a un estado final desde un estado inicial. Desde un estado pueden surgir varias transiciones en función del evento que desencadena el cambio de estado, teniendo en cuenta que las transiciones que provienen del mismo estado no pueden tener el mismo evento, salvo que exista alguna condición que se aplique al evento.



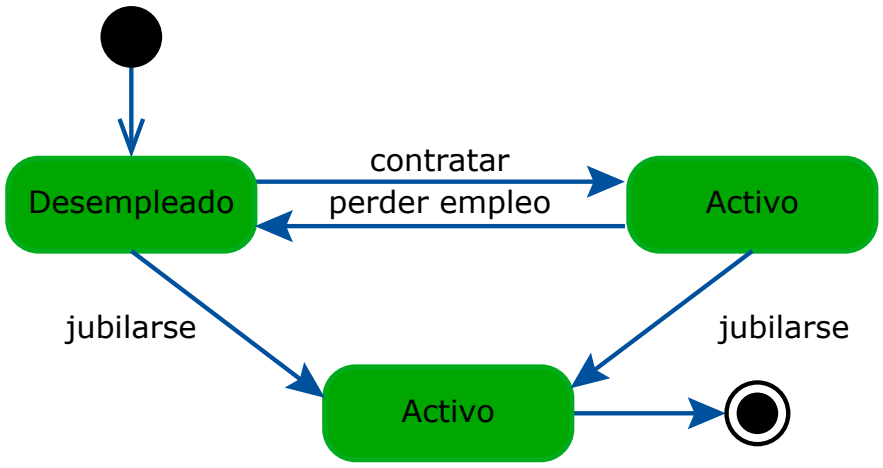
- Un sistema sólo puede tener un estado inicial, que se representa mediante una transición sin etiquetar al primer estado normal del diagrama.
- Pueden existir varias transiciones desde el estado inicial, pero deben tener asociadas condiciones, de manera que sólo una de ellas sea la responsable de iniciar el flujo.
- En ningún caso puede haber una transición dirigida al estado inicial.
- El estado final representa que un componente ha dejado de tener cualquier interacción o actividad.
- No se permiten transiciones que partan del estado final.
- Puede haber varios estados finales en un diagrama, ya que es posible concluir el ciclo de vida de un componente desde distintos estados y mediante diferentes eventos, pero dichos estados son mutuamente excluyentes, es decir, sólo uno de ellos puede ocurrir durante una ejecución del sistema.



Los diagramas de transición de estados comprenden además otros dos elementos que ayudan a clarificar el significado de los distintos estados por los que pasa un componente o sistema. Estos elementos se conocen como acciones y actividades. Una acción es una operación instantánea asociada a un evento, cuya duración se considera no significativa y que se puede ejecutar dentro de un estado, al entrar en un estado o al salir del mismo.

Una actividad es una operación asociada a un estado que se ejecuta durante un intervalo de tiempo hasta que se produce el cambio a otro estado. Para aquellos estados que tengan un comportamiento complejo, se puede utilizar un diagrama de transición de estados de más bajo nivel. Estos diagramas se pueden mostrar por separado o bien incluirse en el diagrama de más alto nivel, dentro del contorno del estado que representa. En cualquier caso su contenido formará un contexto independiente del resto con sus propios estados inicial y final.

Ejemplo diagrama de estados para el objeto empleado:



GLOSARIO

#(Protected): Indica que un atributo es protegido, no será accesible desde fuera de la clase.

-(Private): Indica que un atributo es privado, sólo los métodos de la clase pueden acceder al contenido de ese atributo.

+(Public): Indica que un atributo es público, su valor puede estar accesible desde el exterior de la clase.

Actividades: Acción requerida para llevar a cabo un proceso.

Actor: Se emplea para indicar el tipo de usuario del sistema que podrá ejecutar alguna función.

Asociación: Relación que se da entre elementos de los diagramas, por ejemplo entre los actores y el caso de uso o entre las clases.

Atributos: Características de una Clase, son datos específicos que interesa guardar de cada entidad

Cardinalidad: Indica la participación que tiene una entidad en la relación.

Caso de Uso: Indica una función que el sistema debe proveer.

Clases: Unidad básica que agrupa una colección de objetos.

Entidad: Elemento del sistema de los cuales interesa almacenar información.

Extend: Relación que se da entre casos de uso.

Métodos: Operaciones de una clase.

GLOSARIO

Relaciones: También se conoce como asociaciones, sirven para interconectar las entidades.

Transición: Es un cambio de estado producido por un evento.

UML: Unified Modeling Language, es un lenguaje estandarizado que se utiliza para visualizar los elementos de un sistema de software, compuesto por diagramas que representan elementos estáticos y dinámicos del sistema.

RECURSOS BIBLIOGRÁFICOS

- Andrew S. Tanenbaum and David J. Wetherall.(2003). Computer Networks (5th Edition). Prentice Hall.

- Craig Larman(2007), UML y patrones (2da. Edición). Editorial Prentice Hall.

- G. Booch, J. Rumbaugh, I. Jacobson (2007). El lenguaje unificado de modelado: manual de referencia (2da. Edición). Addison-Wesley.

- James Rumbaugh (2002). Object Oriented Modeling and Design. Pearson Education.

- Schmuller, Joseph. Aprendiendo UML en 24 horas. Prentice Hall

- Scott W. Ambler (2005). Introducción a los Objetos. Cambridge University Press.

- Gutierrez Cosio, Celia (2011). Casos prácticos de UML. Editorial Complutense. Disponible en: http://site.ebrary.com/lib/senavirtualsp/docDetail.action?docID=10536104&p00=uml

- Vélez Serrano, José Peña Abril, Alberto Gortazar Bellas, Patxi (2011). Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java. Editorial: Dykinson. Disponible en: http://site.ebrary.com/lib/senavirtualsp/docDetail.action?docID=10559590&p00=uml

- Kimmel, Paul (2010). Manual de UML. Editorial McGraw-Hill Professional Publishing. Disponible en: http://site.ebrary.com/lib/senavirtualsp/docDetail.action?docID=10433806&p00=uml

OBJETO DE APRENDIZAJE	Introducción al Lenguaje de Modelado Unificado UML
Desarrollador de contenido Experto temático	**Magda Milena García Gamboa** **Andrés Julián Valencia Osorio**
Asesor Pedagógico	**Claudia Milena Hernández** **Rafael Neftalí Lizcano Reyes**
Productor Multimedia	**Adriana Marcela Suárez Eljure** **Victor Hugo Tabares Carreño**
Programadores	**Daniel Eduardo Martínez Díaz**
Líder expertos temáticos	**Ana Yaqueline Chavarro Parra**
Líder línea de producción	**Santiago Lozada Garcés**