

## DESARROLLO DE SOFTWARE 3<sup>ER</sup> SEMESTRE

### PROGRAMA DE LA ASIGNATURA: ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS

#### UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML (LENGUAJE UNIFICADO DE MODELADO)

CLAVE:

INGENIERÍA: TSU:

15142313    16142313

CIUDAD DE MÉXICO, NOVIEMBRE DEL 2016

UNIVERSIDAD ABIERTA Y A DISTANCIA DE MÉXICO





## UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML. (LENGUAJE UNIFICADO DE MODELADO)

### ÍNDICE

UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML (LENGUAJE UNIFICADO DE MODELADO) .....	3
PRESENTACIÓN DE LA UNIDAD .....	3
PROPÓSITOS DE LA UNIDAD .....	4
COMPETENCIA ESPECÍFICA .....	4
4.1. REPRESENTACIÓN DE OBJETOS Y CLASES CON UML .....	4
4.1.1. REPRESENTACIÓN DE CLASES CON UML .....	5
4.1.2. REPRESENTACIÓN DE OBJETOS CON UML .....	6
4.2. DIAGRAMAS Y DOCUMENTACIÓN PARA EL DISEÑO DEL SOFTWARE CON UML .....	6
4.2.1. CASOS DE USO .....	8
4.2.2. ESCENARIOS DEL CASO DE USO .....	13
4.2.3. DIAGRAMAS DE ACTIVIDADES .....	15
4.2.4. DIAGRAMA SECUENCIAL .....	18
4.2.5. DIAGRAMA DE CLASE .....	19
4.2.6. DIAGRAMA DE GRÁFICO DE ESTADO .....	22
CIERRE DE LA UNIDAD .....	25
PARA SABER MÁS .....	25
FUENTES DE CONSULTA .....	26



## UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML. (LENGUAJE UNIFICADO DE MODELADO)

### UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML (LENGUAJE UNIFICADO DE MODELADO)

#### PRESENTACIÓN DE LA UNIDAD

En esta unidad comprenderás que el lenguaje unificado de modelado (UML) brinda un sistema de trabajo con objetos, basado en el análisis y diseño, a través de un conjunto de herramientas, con una consistencia del lenguaje para especificar un sistema de software. Esta especificación se logra mediante la representación de las mejores prácticas en la tecnología de la industria de objetos. Este conjunto de herramientas permite modelar, es decir, analizar y diseñar un sistema orientado a objetos. En la unidad 3 de este curso, se vio que el UML es un sucesor de los lenguajes de modelado de objetos, derivado de las metodologías Booch, OMT y OOSE.

El UML que surge de estas tres metodologías mencionadas integra a la comunidad orientada a objetos con conceptos de modelado básico y permite desviaciones de las herramientas que componen este lenguaje, las cuales son muchas, tales como diagramas de clase, secuenciales, de actividades, casos de uso, escenarios de caso de uso y gráficos de estados. Lo anterior consiste en definir la semántica del programa en cuestión a lo que se va a diseñar y de esta forma se maneja el Lenguaje de Especificación de Objetos, llamado UML.

De esta manera, al elaborar un programa con orientación a objetos se deberá determinar cómo se va a diseñar el problema; una vez establecido esto, se distinguen los tipos de diagramas que se van a usar para hacer el diseño del



## UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML. (LENGUAJE UNIFICADO DE MODELADO)

software y ejemplificar a través de lenguaje UML, los objetos y las clases, para finalmente hacer los diagramas de los mismos.

### PROPÓSITOS DE LA UNIDAD

Al término de esta unidad lograrás:

- Realizar un análisis y un diseño que den forma precisa y detallada a las especificaciones de clases y objetos, evitando el costo de una mala planeación en el comienzo.
- Comprender mejor la relación entre los negocios y los requerimientos del sistema a diseñar

### COMPETENCIA ESPECÍFICA

- Aplicar los tipos de diagramas para estructurar un sistema orientado a objetos mediante el método de modelado UML.

#### 4.1. REPRESENTACIÓN DE OBJETOS Y CLASES CON UML

El diseño de esta representación se realiza a través de objetos y clases. Por lo general cuando se diseña una clase no es necesario mostrar todos los atributos, ni todas sus operaciones, basta con mostrar los subconjuntos más relevantes para organizarlos.



Un objeto, como recordarás, es la unidad mínima en la representación de algo real y una clase, es un objeto con atributos y métodos o comportamientos. La representación en UML no es más que una manera visual en la que se muestra la información que se requiere para diseñar un sistema y así mostrar de manera muy clara la forma en que el usuario interactúa con el éste.

### 4.1.1. REPRESENTACIÓN DE CLASES CON UML

**Clases:** una clase es representada por medio de un marco subdividido en tres niveles: En el primer nivel se muestra el nombre de la clase, el siguiente presenta los atributos y en el último nivel se incluyen las operaciones. Una clase puede representarse de forma esquemática con los atributos y operaciones suprimidos; queda entonces tan sólo un rectángulo con el nombre de la clase. En la siguiente figura se ve cómo una misma clase puede representarse a distinto nivel de detalle, según interese, y según la fase en la que se esté.



Representación de clases con diferentes tipos de detalles



### 4.1.2. REPRESENTACIÓN DE OBJETOS CON UML

**Objetos:** el objeto es representado de forma similar que una clase. En la parte superior del marco aparece el nombre del objeto junto con el nombre de la clase subrayado. El siguiente ejemplo muestra la sintaxis: nombre\_del\_objeto: nombre\_de\_la\_clase.

Puede representarse un objeto sin un nombre específico, entonces sólo aparece el nombre de la clase.



Representación de objeto sin nombre

### 4.2. DIAGRAMAS Y DOCUMENTACIÓN PARA EL DISEÑO DEL SOFTWARE CON UML

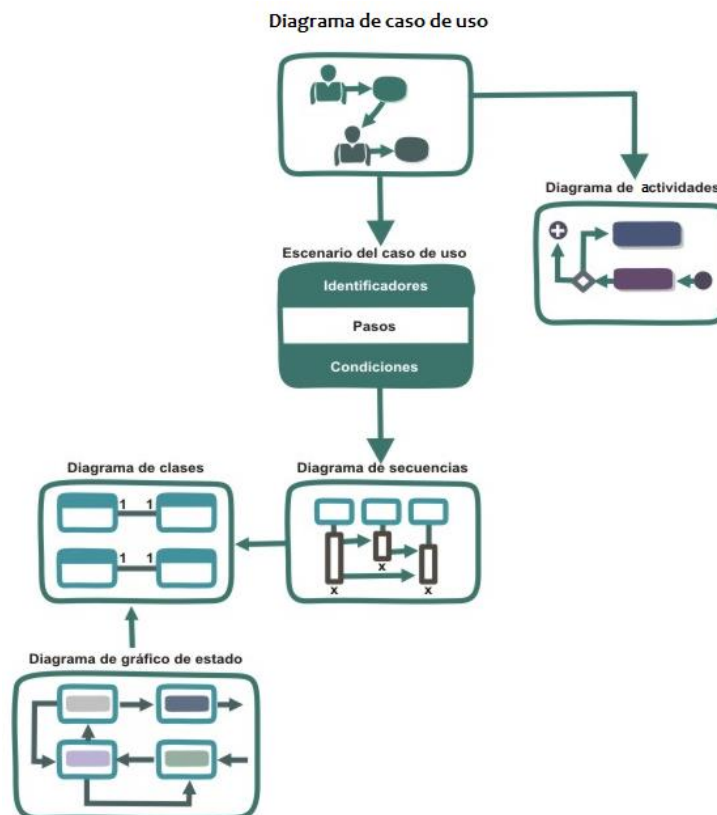
Los diagramas se utilizan para representar diferentes perspectivas de un sistema, de forma que un diagrama es una proyección del propio sistema. El diseño de modelado UML proporciona un amplio conjunto de diagramas que normalmente se usan en pequeños subconjuntos para poder representar las



vistas principales de la arquitectura de un sistema. Los seis diagramas según Kendall y Kendall de UML que más se utilizan son:

1. **Diagrama de caso de uso:** describe cómo se usa el sistema; es ideal para comenzar.
2. **Escenario de caso de uso (aunque técnicamente no es un diagrama):** es una descripción verbal de las excepciones.
3. **Diagrama de actividades:** muestra el recorrido de las actividades.
4. **Diagramas de secuencias:** muestran el orden de actividades y las relaciones de las clases.
5. **Diagramas de clases:** muestran las clases y las relaciones.
6. **Diagramas de gráfico de estado:** muestra las transiciones de estado. Cada clase podría crear un diagrama de gráfico de estado.

La siguiente figura muestra cómo se relacionan (Kendall y Kendall, 2005):



Conjunto de diagramas UML. Tomada de Kendall y Kendall (2005)



Cada uno de los diagramas mencionados se explicará en la imagen anterior de forma detallada en los siguientes temas.

### 4.2.1. CASOS DE USO

Un *diagrama de casos de uso* muestra la relación entre los actores (usuarios del sistema) y el sistema. De esta forma se representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción con la parte externa.

En el caso de uso se describen las secuencias de acciones recíprocas entre dos o más objetos que se producen entre el actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica.

El caso de uso expresa una unidad coherente de funcionalidad, y se representa en el *diagrama de casos de uso* mediante una elipse con el nombre del caso de uso en su interior. Como se verá más adelante, el nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Los casos de uso son la parte realmente útil del documento, ya que con esta descripción tan detallada a través de imágenes y figuras queda asentado de una manera muy clara la forma de interactuar entre el sistema y el usuario.

En los casos de uso, la palabra “uso” se utiliza como sustantivo en lugar de verbo. Un modelo de caso de uso muestra lo que hace un sistema sin describir cómo lo hace, muestra como si estuviera un usuario fuera del sistema,





mostrando los requerimientos; se puede hacer usando los objetos y sus interacciones para derivar comportamiento del objeto, atributos y relaciones.

Como ya se mencionó los *actores* dentro del modelado UML son aquellos que interactúan con el sistema a desarrollar. Las *precondiciones* son los hechos que se han de cumplir para que el flujo de evento se pueda llevar a cabo. El flujo de eventos corresponde a la ejecución normal y exitosa del caso de uso. Los *flujos alternativos* son los que permiten indicar qué es lo que hace el sistema en los casos menos frecuentes e inesperados. Por último, las *poscondiciones* son los hechos que se han de cumplir si el flujo de eventos normal se ha ejecutado correctamente.

El diagrama de caso de uso está formado por:

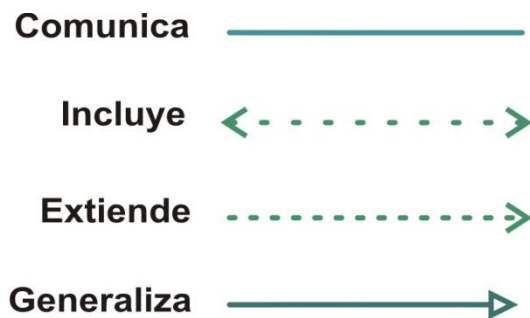
- El actor
- Símbolos de caso de uso
- Líneas de conexión

Los *actores* se refieren en una situación de usuario del sistema; por ejemplo, un actor podría ser el cliente. Los actores dan o reciben los datos del sistema. Los actores secundarios ayudan a mantener el sistema en ejecución o proporcionan ayuda, como las personas que operan el centro de atención telefónica, los empleados(as) de ventanilla, etc.

Un caso de uso ilustra a los desarrolladores un panorama de lo que quieren los usuarios. No tiene detalles técnicos o de implementación. Los casos de uso documentan un solo evento; se nombran con un verbo y un sustantivo. Las



relaciones son el comportamiento y se usan para conectar a un actor, de las cuales existen cuatro tipos básicos:



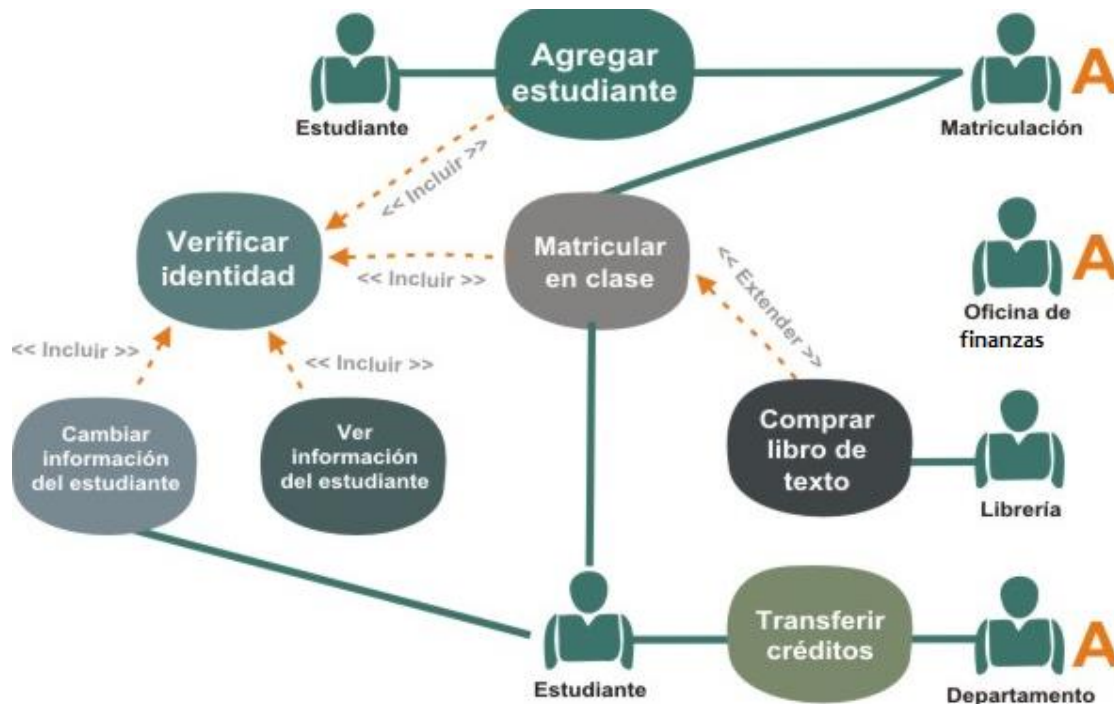
Por el tipo de flechas, se muestran los cuatro tipos: comunica, incluye, extiende y generaliza.

Al hacer el *diagrama del caso de uso* hay que pedir todo lo que los usuarios quieren que el sistema haga para ellos; es decir, mostrar los servicios que se deben proporcionar.

En las fases iniciales ésta podría ser una lista de requerimientos que se extiende en las últimas fases del análisis. Usa los siguientes lineamientos para saber cuáles son estos requerimientos:

- Revisar las especificaciones para establecer los actores.
- Identificar los eventos y hacer los casos de uso.
- Revisar el caso de uso para determinar el flujo de los datos.

A continuación se mostrará una figura que muestra un caso de uso de una matriculación de un estudiante. Agregar un estudiante incluye verificar la identidad de éste.



Ejemplo de caso de uso de matriculación de estudiante. Tomada de Kendall y Kendall (2005)

El caso de uso *Comprar libro de texto* extiende el caso de uso *Matricular en clase* y podría ser parte de un sistema para matricular a los estudiantes de un curso en línea.

Pareciera como si el caso de uso *Cambiar información del estudiante* fuera una característica menor del sistema y no se debiera incluir en el diagrama de caso de uso, pero debido a que esta información cambia con frecuencia, la administración tiene un interés sutil en permitir a los estudiantes cambiar su propia información personal. El hecho de que los administradores juzguen esto como importante no sólo justifica, sino que exige que el estudiante pueda cambiar su información.



A los estudiantes no se les permitiría cambiar su promedio de calificaciones, cuotas a pagar y otra información. Este caso de uso también incluye el caso de uso *Verificar identidad*, y en esta situación, significa que el estudiante tiene que introducir una clave de usuario y contraseña antes de acceder al sistema. Ver información del estudiante permite a los estudiantes visualizar su información personal, así como también los cursos y calificaciones. Los diagramas de caso de uso son un buen punto de partida, pero se necesita una descripción más completa de ellos para su documentación.

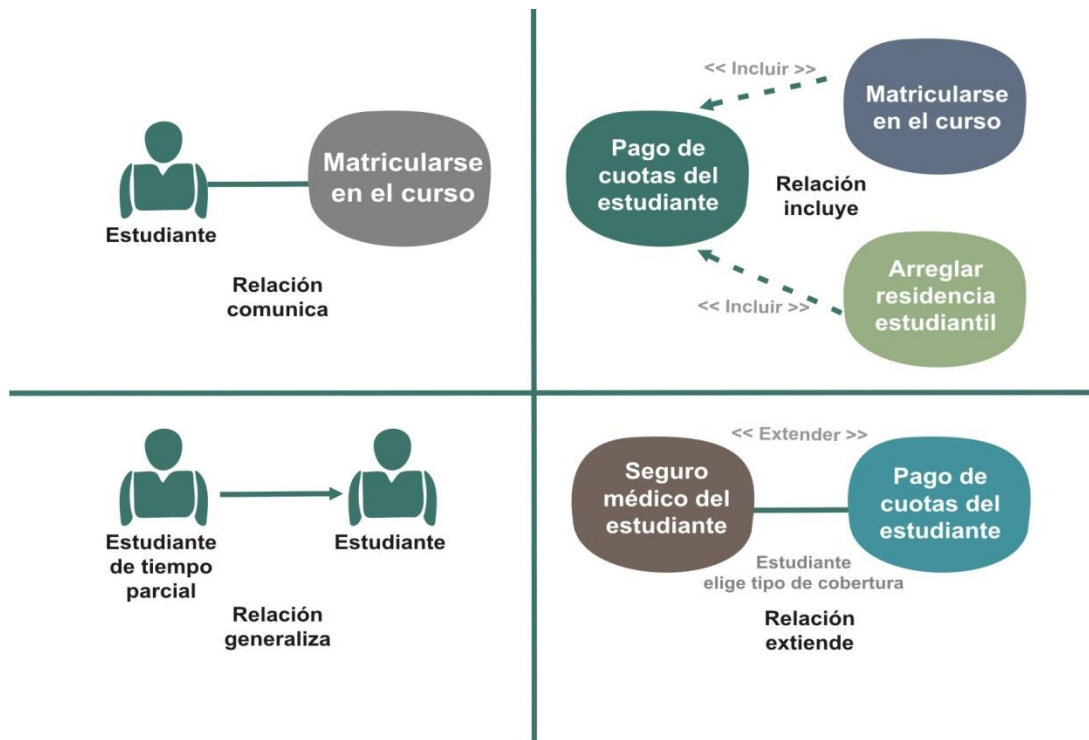
En la siguiente imagen se muestran ejemplos de los cuatro tipos de relaciones que se usan para conectar al actor con el sistema.

El primero de ellos es una *relación-comunica* en la que el estudiante debe comunicar que se matricula en el curso.

El segundo es sobre una *relación-incluye*, en la cual matricularse en un curso y arreglar su residencia estudiantil integra el pago de cuotas por parte del estudiante, el tercero de ellos es una *relación-generaliza*, ya que el proceso que haga un estudiante normal es el mismo que el proceso de un tiempo parcial, y por último la *relación-extiende* se hace presente al momento de que el estudiante paga las cuotas se extiende su seguro médico.



## UNIDAD 4. DISEÑO ORIENTADO A OBJETOS CON UML. (LENGUAJE UNIFICADO DE MODELADO)



Cuatro tipos de relaciones. Tomada de Kendall y Kendall (2005)

### 4.2.2. ESCENARIOS DEL CASO DE USO

Cada caso de uso tiene una descripción como escenario del caso de uso de la imagen anterior, la cual representa un caso en el que hay flujo de eventos y rutas para el comportamiento.

No hay un formato establecido, pero se deben considerar tres puntos importantes:

1. Identificadores e iniciadores de caso de uso
2. Pasos desempeñados
3. Condiciones, suposiciones y preguntas



A continuación se presenta un ejemplo de Gracia (2003), que podría ser el caso de uso (*use case*) de escribir un mensaje en un foro:

<b>Nombre:</b>	Crear mensaje foro
<b>Autor:</b>	Juan Pérez
<b>Fecha:</b>	28/03/2011
<b>Descripción:</b>	Permite crear un mensaje en el foro de discusión.
<b>Actores:</b>	Usuario de Internet firmado.
<b>Precondiciones:</b>	El usuario debe haberse firmado en el sistema.
<b>Flujo normal:</b>	<ol style="list-style-type: none"><li>1. El actor pulsa sobre el botón para crear un nuevo mensaje.</li><li>2. El sistema muestra una caja de texto para introducir el título del mensaje y una zona de mayor tamaño para introducir el cuerpo del mensaje.</li><li>3. El actor introduce el título del mensaje y el cuerpo del mismo.</li><li>4. El sistema comprueba la validez de los datos y los almacena.</li></ol>
<b>Flujo alternativo:</b>	<ol style="list-style-type: none"><li>1. El sistema comprueba la validez de los datos, si los datos no son correctos, se avisa al actor de ello permitiéndole que los corrija.</li></ol>
<b>Poscondiciones:</b>	El mensaje ha sido almacenado en el sistema.

Ejemplo de caso de uso. Tomado de Gracia (2003)



Un escenario del caso de uso es una instancia expresada en el mismo caso.



### 4.2.3. DIAGRAMAS DE ACTIVIDADES

Como su nombre lo dice, estos diagramas están centrados en mostrar las actividades y la secuencia en que deben realizarse, mostrando la forma de trabajar desde el inicio hasta el final del sistema que se está desarrollando. Este tipo de diagramas cubre la parte dinámica del sistema y son muy útiles para especificar detalladamente el funcionamiento del flujo entre los objetos.

Como ya se dijo, un diagrama de actividades al final de cuentas es para ver la secuencia de las actividades, las cuales son una acción. Dicha acción no es más que la ejecución de un proceso o parte del sistema que puede tener como consecuencia cambiar un valor en el mismo o solicitar o devolver un dato. Estas acciones pueden estar enlazadas, hablando en términos de programación orientada a objetos. Las acciones involucran manejo de





información, creación o destrucción de objetos nuevos, o simplemente devolver un valor. Gráficamente, un *diagrama de actividades* es una colección de nodos y arcos que contiene:

- **Estados de actividad:** la representación de este estado está basada en un rectángulo con las puntas redondeadas, donde el interior se representa una actividad. El estado de actividad puede descomponerse en más subactividades representadas a través de otros diagramas de actividades; estos estados pueden ser interrumpidos y pueden tardar cierto tiempo en concluir. Adicionalmente se pueden encontrar elementos como acciones de entrada y acciones de salida, tal como se muestra en el ejemplo siguiente:

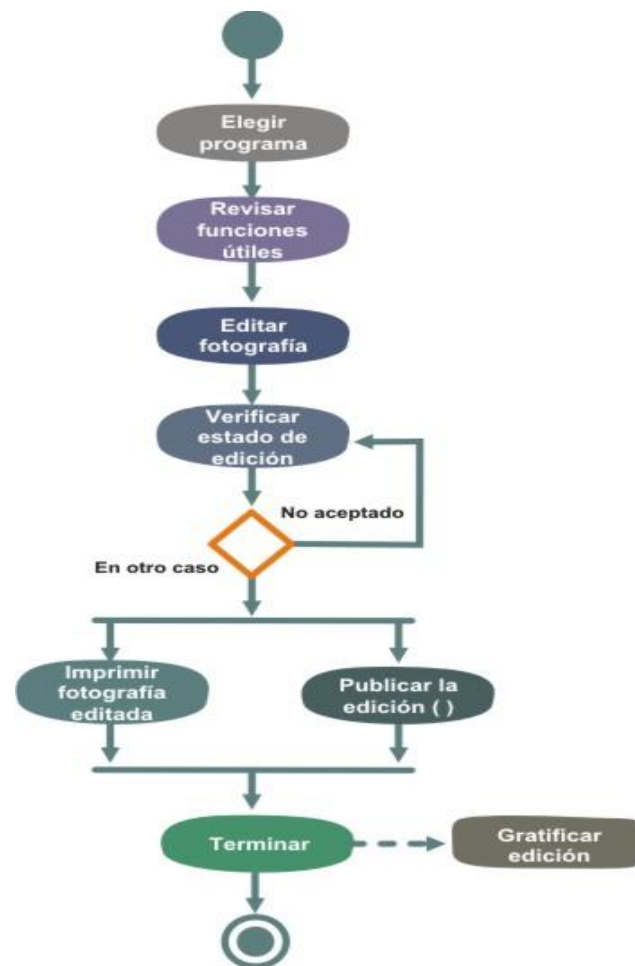
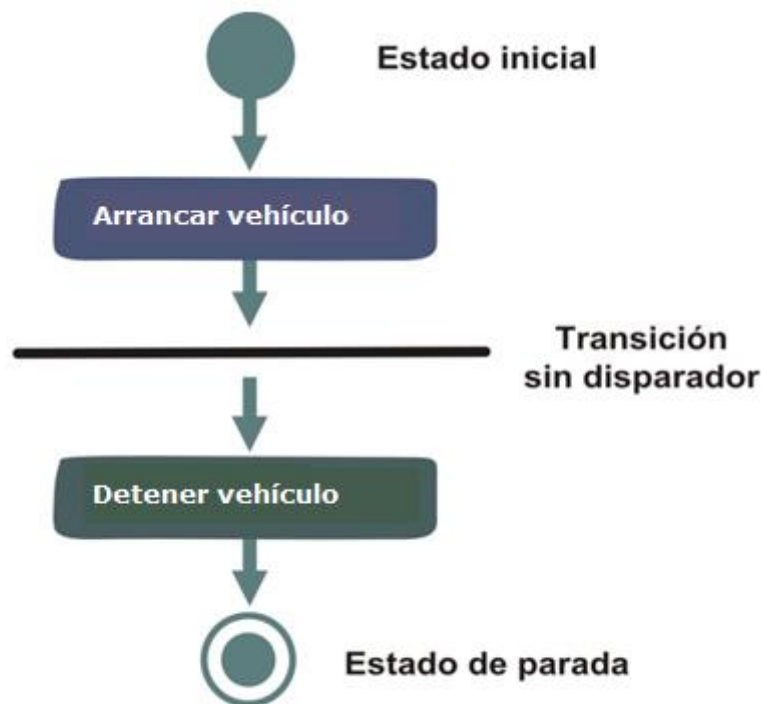


Diagrama de actividades para modelar una actividad. Tomada de Alarcón (2005)





- **Estados de acción:** al igual que el estado de actividad, el estado de acción está basado en un rectángulo con las puntas redondeadas, donde en el interior se representa una acción.
- **Transiciones:** las transiciones muestran el paso de un estado a otro, bien sea estado de actividad o de acción. Esta transición se produce como resultado de la finalización del estado del que parte el arco dirigido que marca la transición. Como todo flujo de control debe empezar y terminar en algún momento, se puede indicar esto utilizando dos disparadores de inicio y final.

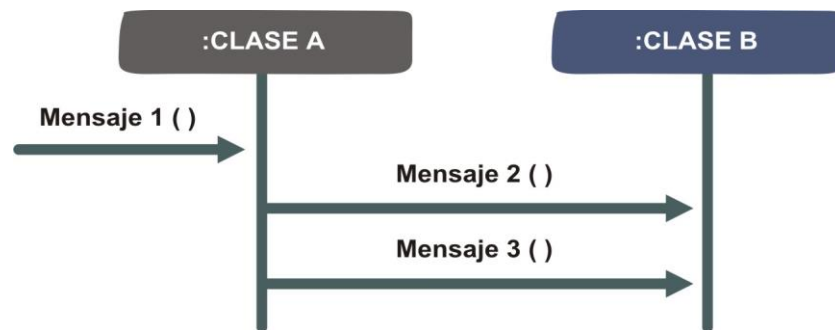


Transición



### 4.2.4. DIAGRAMA SECUENCIAL

El *diagrama de actividades para modelar una actividad* representa de manera visual las actividades que va a realizar; éstos son conocidos como diagramas de iteración y en este tipo de diagramas se especifica el orden en que se va a realizar, para lo cual se identifican los objetos que involucran al sistema y sus relaciones entre ellos; es decir, muestran y resaltan el orden de los mensajes mientras que los diagramas de colaboración muestran la estructura de los objetos. Sin embargo, de manera fácil, puede haber un cambio de diagrama de secuencia en diagrama de colaboración y viceversa.



Representación de clases en diagramas de secuencia

Un diagrama de secuencia en el modelado UML muestra una interacción ordenada según la secuencia de cada evento. Muestra los objetos participantes en la interacción y los mensajes que intercambian, ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo,



descripciones de acciones, etc.) en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

### 4.2.5. DIAGRAMA DE CLASE

Los diagramas de clase son utilizados para mostrar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido. Un diagrama de clases está compuesto por los siguientes elementos: clase, atributos, métodos y visibilidad, y relaciones: herencia, composición, agregación, asociación y uso.

A continuación se describen los elementos que forman el diagrama de clases:

**Clase:** es la unidad básica que incluye toda la información de un objeto, y un objeto es una instancia de una clase. A través de ella podemos modelar el entorno en estudio: una casa, un auto, una cuenta corriente, etc.

En UML una clase es representada por un rectángulo que posee tres divisiones:



Representación de una clase



**Nivel superior:** contiene el nombre de la clase a utilizar.

**Nivel intermedio:** contiene los atributos (o variables de instancia) que caracterizan a la clase (pueden ser *private*, *protected* o *public*).

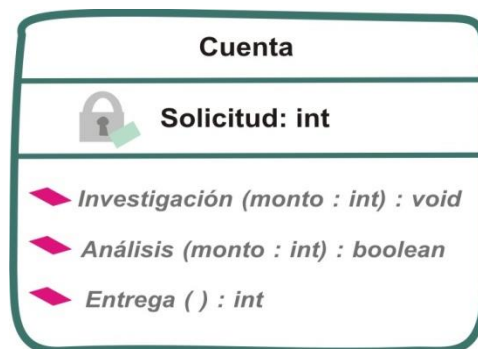
**Nivel inferior:** contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: *private*, *protected* o *public*).

Un ejemplo sobre el diagrama de clase:

Un crédito que posee como característica lo siguiente:

- Solicitud
- Puede realizar las operaciones de:
  - Investigación de crédito (investigación)
  - Análisis de crédito (análisis)
  - Y entrega de crédito (entrega)

El diseño asociado es:





Representación de la clase cuenta (adaptación de un ejemplo)


La clasificación de los atributos y métodos que a continuación se presentan fue tomada de Thompson (2004). Los *atributos* conocidos como las




características de una clase pueden ser de tres tipos: públicos privados o protegidos.


**Públicos** (): indican que el atributo será visible y accesible de todos lados.


**Privados** (): indican que el atributo sólo será accesible desde dentro de la clase (sólo sus propios métodos pueden entrar).

**Protegidos** (): indican que el atributo no será accesible desde fuera de la clase, pero sí se podrá entrar en él por métodos de la clase además de las subclases que se deriven.

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

**Públicos** (): 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 pueden entrar).

**Protegido** (): indica que el método no será accesible desde fuera de la clase, pero sí se podrá entrar en él por métodos de la clase, además por los de las subclases que se deriven.



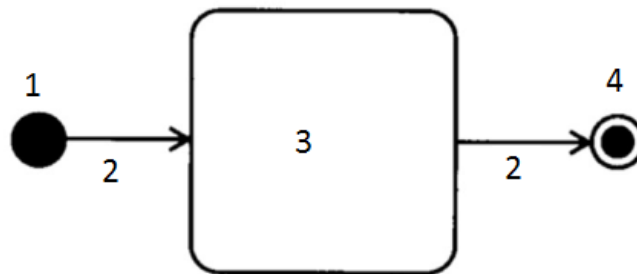
### 4.2.6. DIAGRAMA DE GRÁFICO DE ESTADO

En el diagrama de estados, se representan todos los estados posibles en los que puede entrar un objeto y la manera en que cambia entre un estado y otro; es decir, muestran todo el ciclo de vida del objeto.

La manera en que se representa, es que los objetos son nodos y sus cambios o transiciones se representan con arcos y se les coloca una etiqueta con los cambios que suceden.

#### Elementos de un diagrama de estado

A continuación se exponen los elementos de un diagrama de estado (Sparx System, 2007).



Partes de un diagrama de estado (Figueroa, s/f)

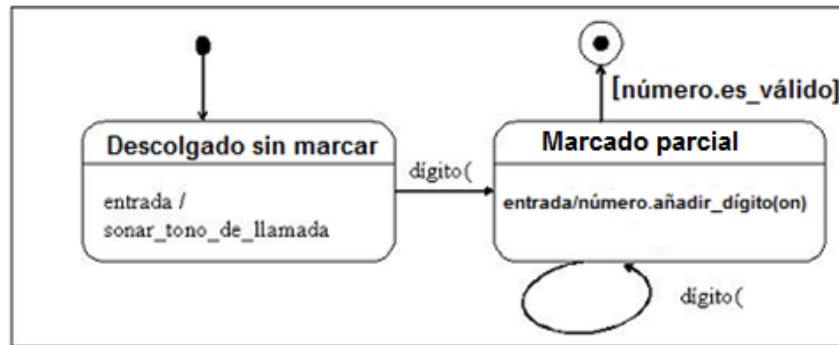
- El **círculo negro relleno** denota el punto inicial de la secuencia del diagrama de estados.
- El símbolo de **fecha** denota una transición.
- El **rectángulo con bordes** redondeados denota un estado.
- El **círculo** que a su vez contiene un **círculo relleno** denota punto final.



- **“Estado:** Identifica un periodo del objeto (no instantáneo) en el cual el objeto está esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos.
- **Eventos:** Es una ocurrencia que puede causar la transición de un estado a otro de un objeto.
- **Envío de mensajes:** Además de mostrar la transición de estados por medio de eventos, puede representar el momento en el cual se envían mensajes a otros objetos.
- **Subestados:** Un estado puede descomponerse en subestados, con transiciones entre ellos y conexiones al nivel superior. Las conexiones se ven a nivel inferior como estados de inicio o fin, los cuales se suponen conectados a las entradas y salidas del nivel inmediatamente superior.
- **Acciones:** Es posible especificar la solicitud de un servicio a otro objeto como consecuencia de la transición. Se puede especificar el ejecutar una acción como consecuencia de entrar, salir, estar en un estado, o por la ocurrencia de un evento.”

Un estado puede componerse de uno o dos niveles: en el primero se coloca el nombre del estado y si se tiene el segundo nivel, se ponen las acciones de ese estado, y estas acciones pueden ser de entrada, de transición o de salida.

Cada vez que hay una acción de entrada es porque fue provocada por un llamado al objeto. Las de salida pasan cuando ha sido ejecutada la acción que fue solicitada y las de transición suceden cuando es llamada una acción, pero ésta no causa el cambio hacia otro estado, por ejemplo:



**Donde:**

- El **estado de descolgado** sin marcar indica la entrada de tono de llamada, con la cual se otorga el permiso de ir a la siguiente transición.
- En la **transición entre estados** se introduce el dígito.
- En el estado **marcado parcial** se completa la marcación por medio de una transición interna en la cual se introducen los dígitos restantes hasta completar el número telefónico.
- Una vez **completada la transición** superior se llega al **punto final** que es la **llamada**.

A continuación se expone un segundo ejemplo de diagrama de estado de conexión a Internet.

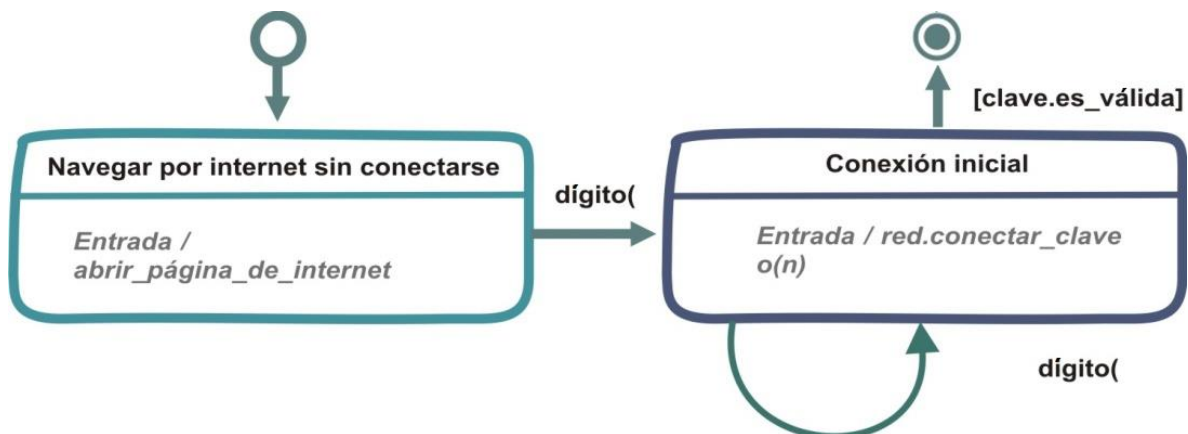


Gráfico de estado de conexión a Internet





En un diagrama de estados se pueden representar ciclos infinitos, pero si existe un estado inicial que comienza el ciclo y un estado final, es considerado de destrucción. En realidad estos estados iniciales y finales no se pueden considerar estados, pero se colocan para saber cuál es la transición inicial y cuál es la final.

### CIERRE DE LA UNIDAD

Has concluido la unidad 4 del curso. A lo largo de ésta se vieron conceptos de diseño orientado a objetos con UML, cómo se representan los objetos y las clases, además de cómo se hacen los diagramas para los casos de uso, escenarios del caso de uso, diagramas de actividades, diagramas secuenciales, de clase y el gráfico de estado.

Es aconsejable que revises nuevamente la unidad en caso de que los temas que se acaban de mencionar no te sean familiares, o no los recuerdes; de no ser éste tu caso, has concluido tu curso de Análisis y diseño orientado a objetos.

### PARA SABER MÁS

Para saber cómo insertar o crear un dibujo de Visio en otro programa, puedes consultar la siguiente página:

- ¿Cómo insertar en Word un diagrama en Visio?:  
<http://office.microsoft.com/es-mx/visio-help/utilizar-visio-con-otros-programas-de-2007-microsoft-office-system-HA010198865.aspx>



### FUENTES DE CONSULTA

- Alarcón, R. (2005). *Diseño orientado a objetos con UML*. Madrid: Grupo Eidos.
- Figueroa, P. (s. f.). *Conceptos básicos en un diagrama de estados*. Canadá: Universidad de Alberta. Recuperado de <http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/estados01.html>
- Fowler, M. y Scott, K. (1999). *UML. Gota a gota*. México: Addison Wesley.
- Gracia, J. (2003, 27 de septiembre). UML: Casos de uso. Use case. Desarrollo de Software Orientado a Objetos. *Ingeniero Software*. Recuperado de: <http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php>
- Kendall, K. y Kendall, J. (2005). *Análisis y diseño de sistemas*. México: Prentice Hall Iberoamericana.
- Kenneth, C. (2004). *Lenguajes de programación: principios y práctica*. México: Thomson.
- Larman, C. (2004). *Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design*. México: Prentice Hall.
- Loudon, K. C. (2004). *Lenguajes de programación: Principios y práctica*. México: Thompson.
- Quatrani, T. y James, R. (1997). *Visual modeling with rational rose and UML*. México: Addison Wesley.
- Sparx System (2007). *Diagrama de máquina de estados UML 2*. Mendoza, Argentina: Sparx Systems Argentina SOLUS S.A. Recuperado de [http://www.sparxsystems.com.ar/resources/tutorial/uml2\\_statediagram.html](http://www.sparxsystems.com.ar/resources/tutorial/uml2_statediagram.html)