UF3:Introducción al diseño orientado a objetos

Olimpia Olguín Tinoco

Innovación en Formación Profesional

Grupo Planeta

Programación Orientada a objetos

- Es un método de programación basado en las clases y su jerarquía, así como en las relaciones existentes entre los objetos de proyecto:
  - **Clase:** una definición de la realidad podría hacerse mediante grupos de objetos con conductas y características diferentes, sin embargo estas características comunes suelen agruparse para formar un tipo de objeto(clase).
  - **Atributo:** es una característica aplicada a un elemento de la clase.
  - **Método:** como elemento de la clase, el método contiene un procedimiento que fijará la conducta manifiesta por los objetos pertenecientes a dicha clase.
  - **Visibilidad:** La visibilidad de una propiedad o un método puede mencionarse como la posibilidad que tengan de ser accedidos. Palabra clave, como *Public, Private o Protected*, de los códigos definen si puede acceder a cualquiera o no.

- Un **objeto de software** almacena su estado en variable, y manifiesta su conducta a través de los métodos que operan con los estados de los objetos y canalizan la comunicación entre otros objetos.

Programación Orientada a objetos

- **Instanciación**. Es la creación de una *instancia real*, esto es, de *un objeto a partir de una clase*. Se instancia la clase con la creación de un objeto concreto de dicha clase. Este objeto, a diferencia de la clase, es ejecutable por el ordenador y le hemos dado un nombre y colocado en algún lugar concreto de nuestro proyecto.
- **Relaciones**:
  - **Herencia.** La herencia define las relaciones entre las clases en POO: podemos tener una clase base(superclases) de las que pueden derivar otro tipo de clases (subclases).
  - **Composición.** Es una forma de asociación en la que un objeto forma parte integra del otro, por ejemplo un departamento de una Universidad.
  - **Agregación.** Hay objetos contenidos y objetos que lo contienen, pero la relación es parametrizada y su existencia, o inexistencia , no les afecta.

UML

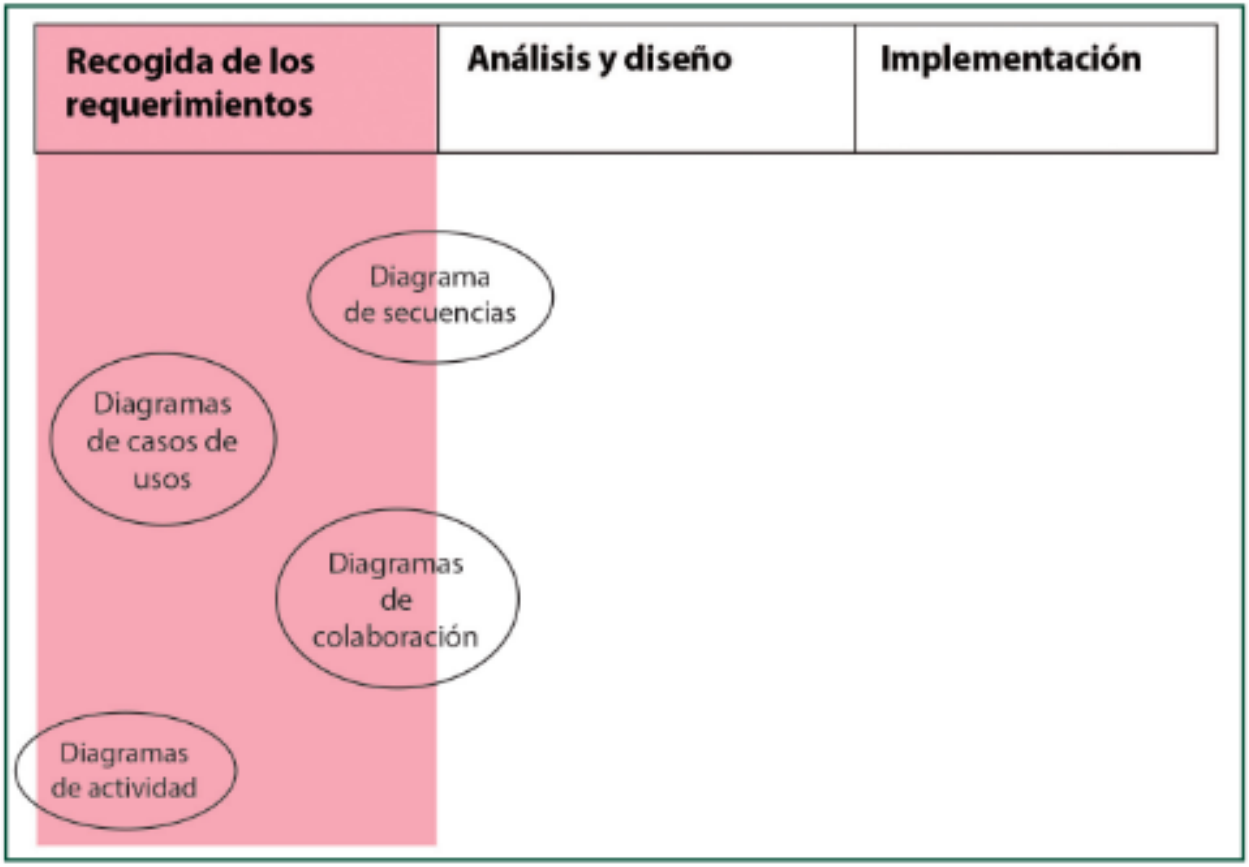
- **UML** son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel *internacional* por numerosos *organismos* y empresas para crear *esquemas, diagramas y documentación* relativa a los desarrollos de software (programas informáticos).

- En realidad el término lenguaje quizás no es el más apropiado, ya que *no es un lenguaje propiamente dicho*, sino una serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas relativos al software.

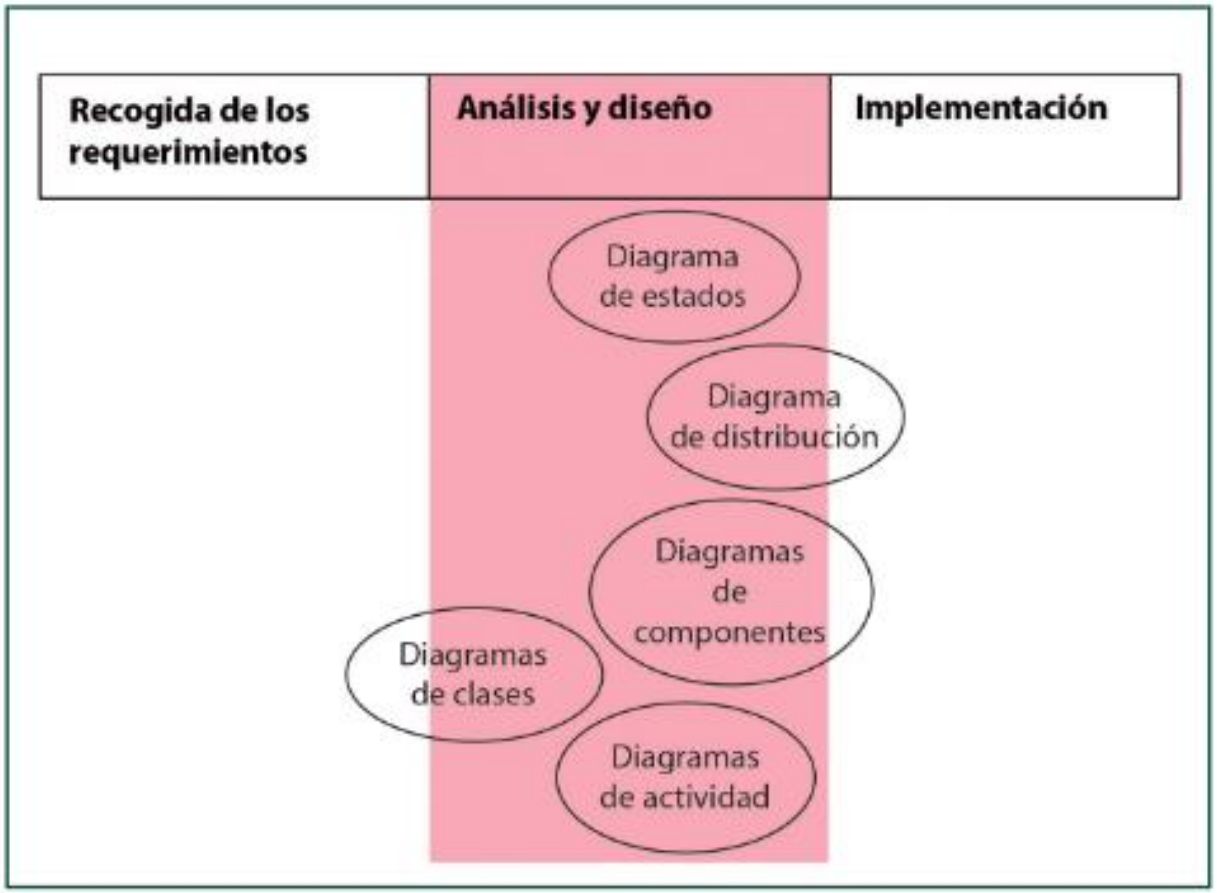
- UML es una *herramienta* propia de personas que tienen conocimientos relativamente avanzados de programación y es frecuentemente usada por *analistas funcionales* (aquellos que definen qué debe hacer un programa sin entrar a escribir el código) y analistas-programadores (aquellos que dado un problema, lo estudian y escriben el código informático para resolverlo en un lenguaje

- El lenguaje UML ofrece un total de 14 diagramas que representan el comportamiento del software. Todos estos se pueden clasificar en :
  - **Diagramas Estructurale**s: Presentan modelos **estáticos** del objeto, como clases, paquetes o componentes, aunque también pueden ser utilizados como modelos en tiempo de ejecución.
  - **Diagramas de Comportamiento:** aquellos que nos muestran la forma que tienen de reaccionar ante los estímulos externos. Son diagramas **dinámicos**, basados en su devenir en tiempo de ejecución

Diagramas en las fase de Requerimientos



Diagramas en la fase de Análisis y diseño



Diagramas estáticos

- **Diagrama de paquetes**, que representa esencialmente las relaciones de diferentes tipos entre los contenidos de diferentes paquetes de un modelo.
- **Diagrama de clases**, Definen la estructura de un sistema. Describen las clases como conjuntos de objetos que comparten atributos, operaciones, métodos, etc.
- **Diagrama de objetos**, que representa instancias de clasificadores definidos en un diagrama de clases previo y relaciones entre ellas.
  Diagrama de estructuras compuestas, que describe casos en que, o bien las instancias de un clasificador tienen como partes instancias otros, o bien en el comportamiento ejecutante de un clasificador participan instancias otros.
- **Diagrama de componentes**, que es un diagrama de clases y a la vez de estructuras compuestas simplificado y más adecuada para determinadas tecnologías de programación. Describen la organización de los componentes de software.
- **Diagrama de despliegue**, que describe la configuración en tiempo de ejecución de un software especificado, normalmente, por un diagrama de componentes.
- **Diagrama de perfil**, permite adaptar o personalizar el modelo con construcciones que son específicas de un dominio en particular, de una determinada plataforma, o de un método de desarrollo de software…

Diagramas dinámicos

- **Diagrama de casos de uso**, que considera los comportamientos de un sistema principalmente desde el punto de vista de las interacciones que tiene con el mundo exterior. Muestra como un elemento cliente (llamado actor) interactúa con el sistema.
- **Diagrama de estados,** describen la conducta dinámica de un sistema como respuesta a estímulos externos.
- **Diagrama de actividades**, se utiliza para mostrar la traza de operaciones que se realiza en un proceso del sistema cuestionado.
- **Diagrama de secuencias**, los diagramas de secuencia ofrecen una visión de las interacciones de los objetos, ordenadas según el momento en el que ocurren.
- **Diagrama de comunicaciones**, que se basa directamente en una estructura compuesta.
  Diagrama de visión general de la interacción, que da una visión resumida del comportamiento emergente.
  Diagrama temporal, que se basa en un diagrama de estados previo o más de uno y a la vez pone énfasis en los cambios de estado a lo largo del tiempo.

Diagrama de clases

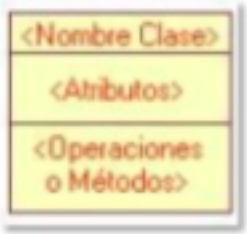
- Un diagrama de clases trae vinculados algunos conceptos que ayudarán a entender la creación y el funcionamiento en su totalidad. Estos conceptos son:
  - Clase, atributo y método (operaciones).
  - Visibilidad.
  - Objeto.
  - Relaciones. Herencia, composición y agregación.
  - Clase asociativa.
  - Interfaces.

Diagrama de clases: Clase

- **Clase**

Es la unidad básica que encapsula toda la información de un Objeto (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:



En donde:

- **Superior**: Contiene el nombre de la Clase
- **Intermedio**: Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser private, protected o public).
- **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).

Diagrama de clases: Elementos

- **Atributos:** son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Ejemplo: el objeto es una puerta, sus propiedades o atributos serían: la marca, tamaño, color y peso.
- Tipos de atributos:
  - **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 utilizar).
  - **protected** (#, ): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

Diagrama de clases: Relaciones

- **Cardinalidad de relaciones**: indica el grado y nivel de dependencia de las clases, se anotan en cada extremo de la relación y éstas pueden ser:
- * = Cero, uno ó n.
- 0,1 = Cero o uno.
- 1..* = Uno o más.
- 1 = Exactamente uno (también podría ser otro número).
- 1..5 = Entre uno y cinco.

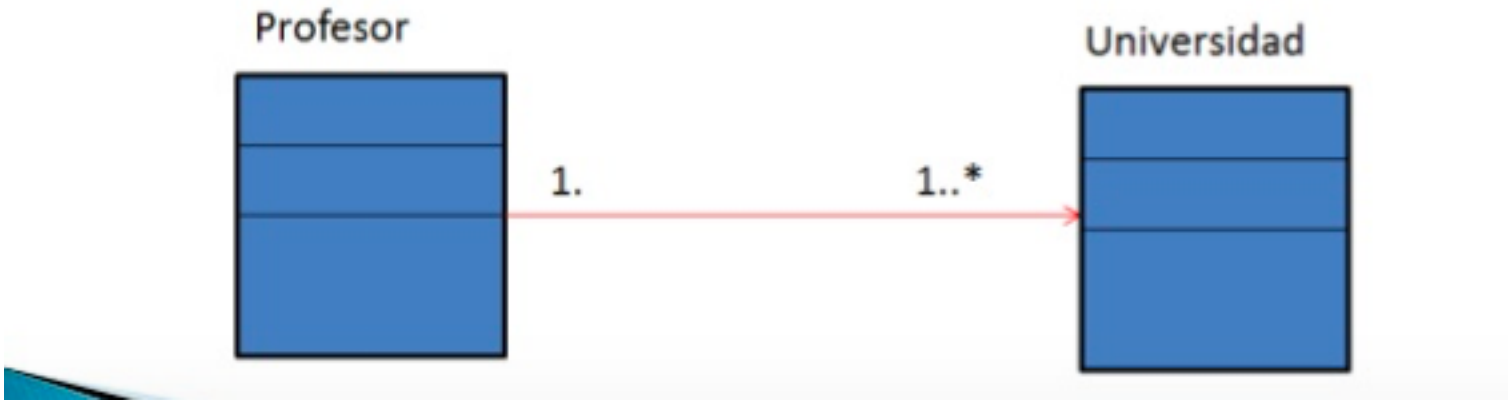
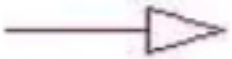


Diagrama de clases: herencia

- **Herencia (Especialización/Generalización):** 

Indica que una subclase hereda los métodos y atributos especificados por una Super Clase (también llamada clase padre), por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected).

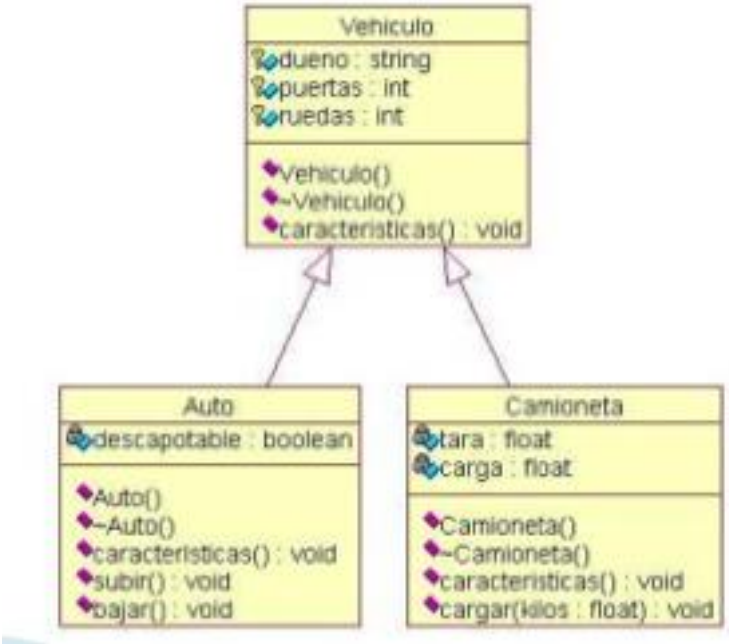


Diagrama de casos de uso

- El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).
- Un diagrama de casos de uso consta de los siguientes elementos:
  - Actor.
  - Casos de Uso.
  - Relaciones de Uso, Herencia y Comunicación.

Diagrama de casos de uso: Actor

- Una definición previa, es que un **Actor** es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.
- Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de Vendedor con respecto al sistema puede ser realizado por un Vendedor o bien por el Jefe de Local.

Diagrama de casos de uso: Caso de uso

- Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.
-

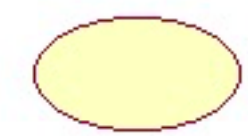
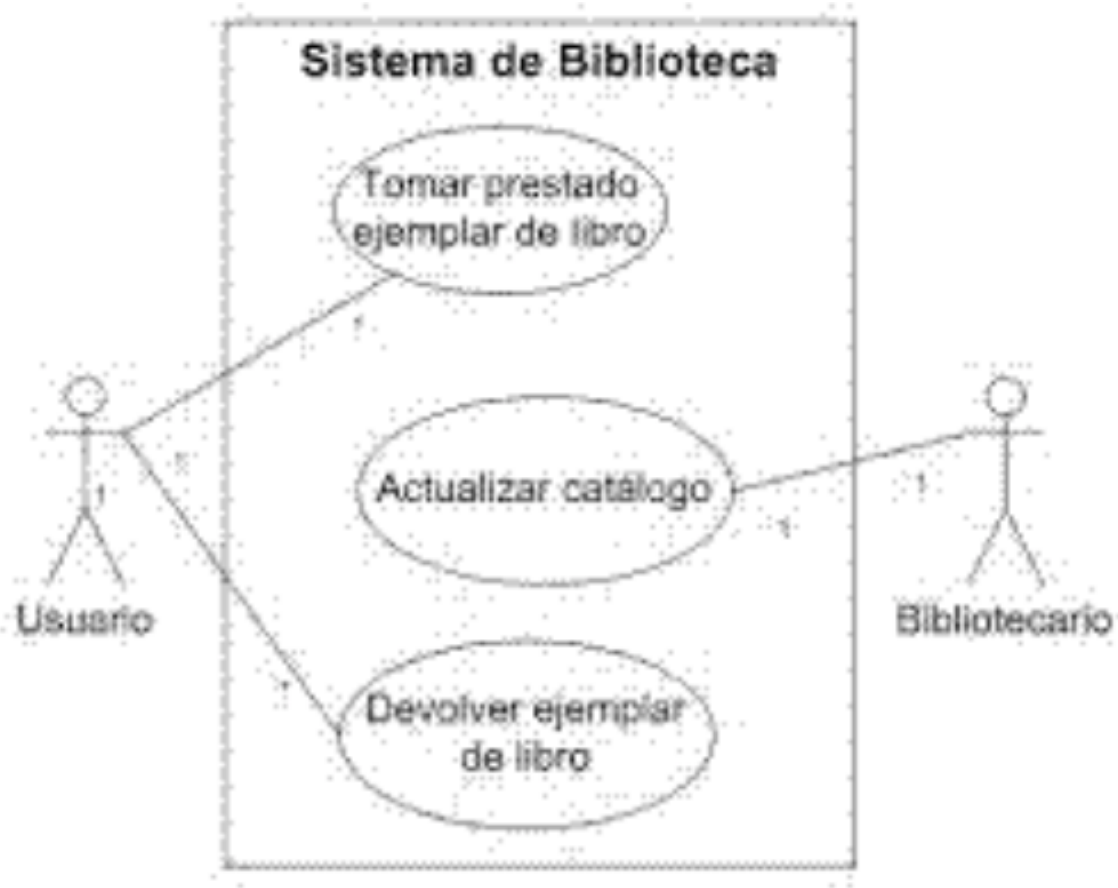


Diagrama de casos de uso: Relaciones

- Asociación

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

Ejemplo caso de uso



Herramientas de diseño de diagramas

Herramientas de asistencia gráfica para diagramas:

- **Microsoft Visio**. Programa comercial considerado de propósito general, aunque posee capacidad para los diagramas UML.
- **JDeveloper.** Plug-in no comercial para el entorno de desarrollo NetBeans. Diseño de casos de uso, clases, actividad y diagramas de secuencia.
- **Visual Paradigm.** SDE for NetBeans. Plug-in comercial. Facilidad de migración de diseños ya realizados. Capacidad de trabajo en grupo
- **Bouml.** No comercial. Compatible con C++, Java, PHP, entre otros. Sigue UML 2.0 Standard.
- **Enterprise Architect**. Comercial. Análisis UML y herramienta de diseño. Trazabilidad desde los requerimientos hasta la publicación. Permite el trabajo en equipo.
- **MDT-UML2Tools.** Integración en entorno de desarrollo Eclipse. Permite diagramas de estructura, comportamiento e interacción
- **SharpDevelop.** Entorno de desarrollo no comercial para los lenguajes de programación: C#, Visual, Basic .NET, F#, Python, Ruby, Boo y C++.

Diagrama de secuencia

Un **diagrama de secuencias** es un tipo de diagrama de interacción que muestra el progreso de los objetos representado por unas líneas, llamadas **líneas de vida**, que se despliegan hacia abajo en el espacio reservado para nuestro modelaje.

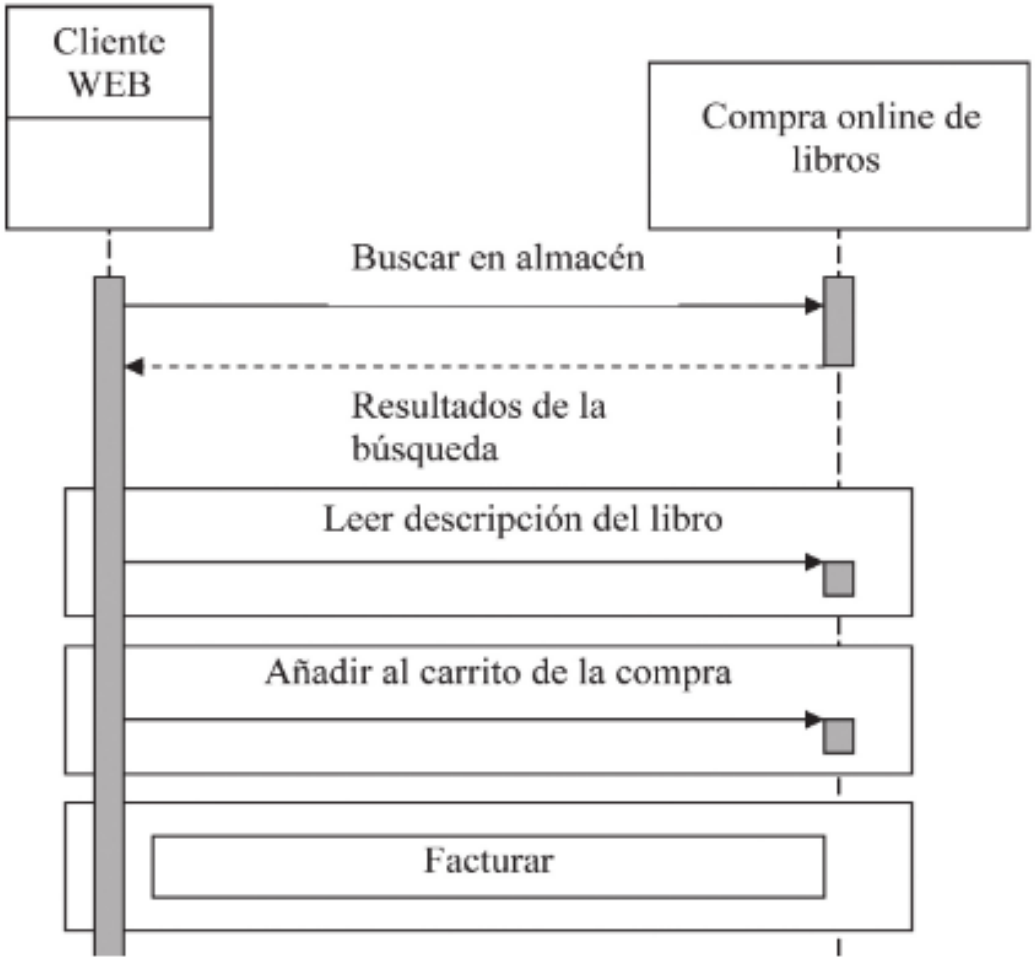
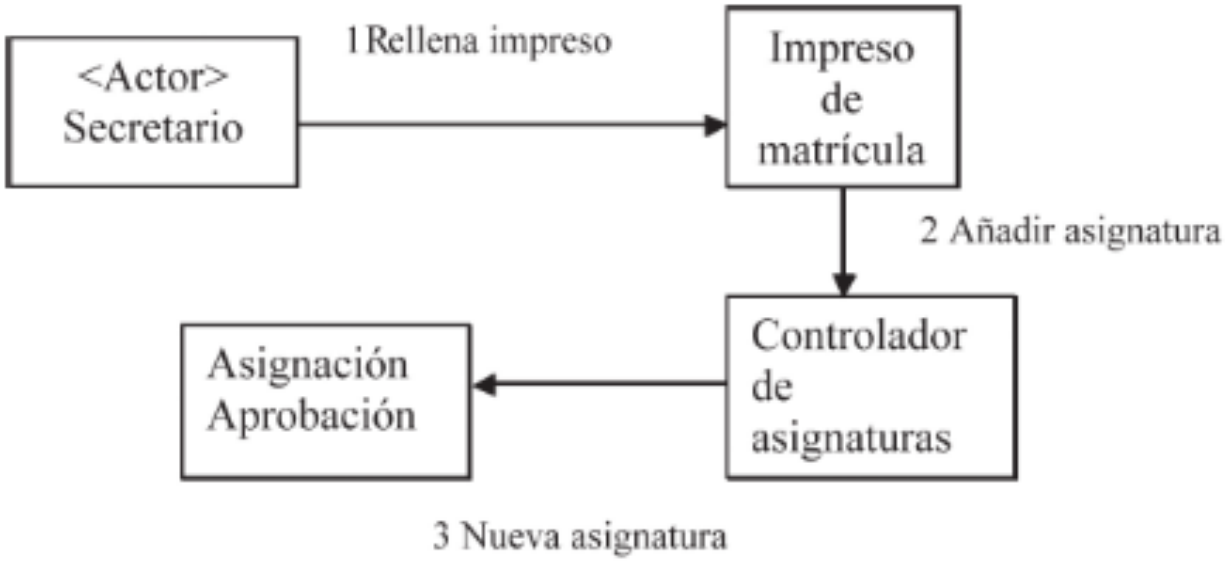


Diagrama de Colaboración(comunicación)



Los diagramas de colaboración, también conocidos como diagramas de comunicación. Describen las interacciones entre los objetos mediante la secuencia de los mensajes y los roles que adquieren cada uno de ellos. Los diagramas de colaboración son una combinación de los diagramas de clase, los de secuencia y los casos de uso

Diagrama de actividades

El diagrama **de actividades se utiliza para mostrar una secuencia de actividades.** Se suele indicar el inicio y el final de la actividad, detallando todos los caminos que bifurquen mientras dura su realización

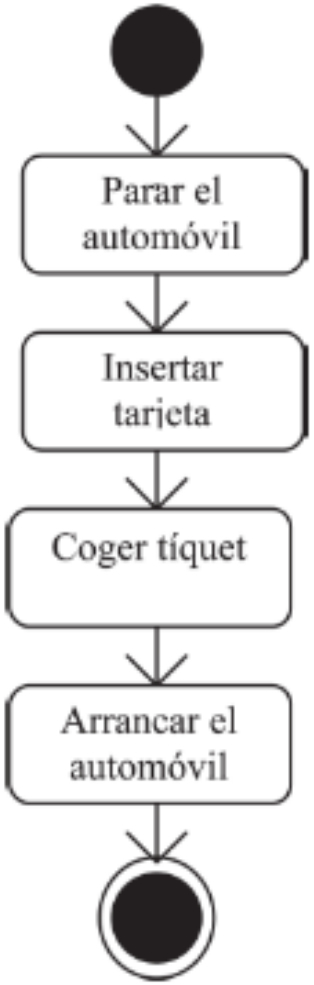
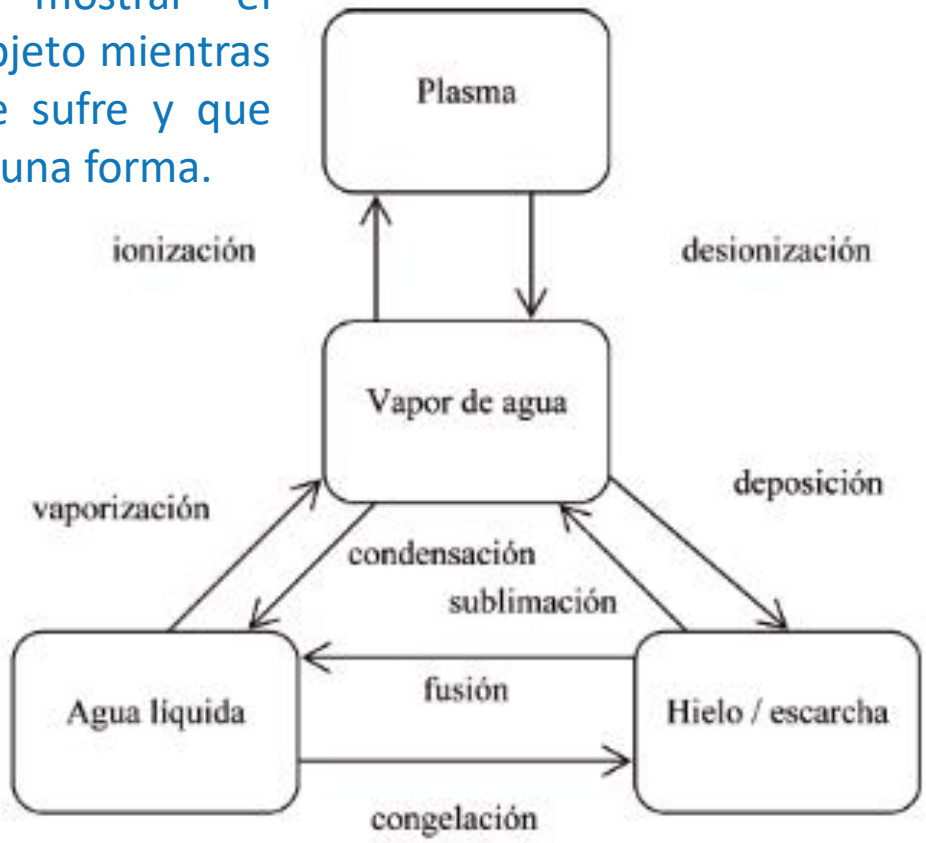


Diagrama de estado

Un diagrama de estado (diagrama de máquina de estado) se encarga de mostrar el comportamiento que tiene un objeto mientras atraviesa todos los eventos que sufre y que pueden hacerle reaccionar de alguna forma.



Innovación
en Formación
Profesional

La mejor forma de aprender es "haciendo".
Educación y empresas forman un binomio inseparable