¿Qué es UML?

UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

Con UML nos debemos olvidar del protagonismo excesivo que se le da al diagrama de clases, este representa una parte importante del sistema, pero solo representa una vista estática, es decir muestra al sistema parado. Sabemos su estructura, pero no sabemos que les sucede a sus diferentes partes cuando el sistema empieza a funcionar. UML introduce nuevos diagramas que representa una visión dinámica del sistema. Es decir, gracias al diseño de la parte dinámica del sistema podemos darnos cuenta en la fase de diseño de problemas de la estructura al propagar errores o de las partes que necesitan ser sincronizadas, así como del estado de cada una de las instancias en cada momento. El diagrama de clases continúa siendo muy importante, pero se debe tener en cuenta que su representación es limitada, y que ayuda a diseñar un sistema robusto con partes reutilizables, pero no a solucionar problemas de propagación de mensajes ni de sincronización o recuperación ante estados de error. En resumen, un sistema debe estar bien diseñado, pero también debe funcionar bien.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

UML es ahora un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.

UML permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir mediante la restricción estamos forzando el comportamiento que debe tener el objeto al que se le aplica.

Diagrama de casos de uso:

Describen un uso del sistema y cómo este interactúa con el usuario. Lo realmente útil de los casos de uso es el documento que describe el caso de uso, en este documento se explica la forma de interactuar entre el sistema y el usuario. Este podría ser el caso de uso de escribir un mensaje en un foro.

Simbología:

Casos de uso: Representado por una elipse, cada caso de uso contiene un nombre, que indique su funcionalidad. Los casos de uso pueden tener relaciones con otros casos de uso. Sus relaciones son:

Inclusión: Representado por una flecha con línea discontinua y la palabra <<include>>. A include B quiere decir que A incluye el caso de uso B.

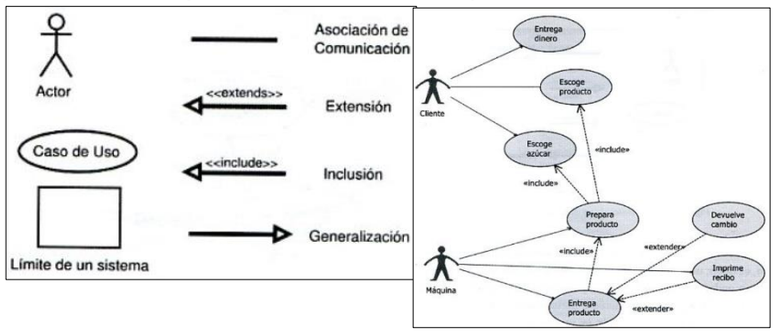
Extensión: Representado por una flecha con línea discontinua y la palabra <<extends>>. A extends B indica que A se usa en B.

Generalizacion: Representado por una flecha con línea continua. Es la típica relación de herencia A generaliza B quiere decir que B es un caso particular de A.

Actores: se representan por un muñeco. Sus relaciones son:

Comunicación: Comunica un actor con un caso de uso, o con otro actor.

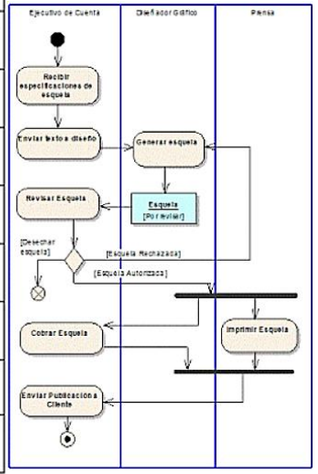
Límite del sistema: Representado por un cuadro, identifica las diferentes partes del sistema y contiene los casos de uso que la forman.



Diagramas de actividad:

Los diagramas de actividades son particularmente útiles para la descripción del comportamiento que tiene una gran cantidad de proceso paralelo. Permite seleccionar el orden en que se harán las cosas. Indica las reglas esenciales de secuenciación que tengo que seguir. Ésta es la diferencia clave entre un diagrama de actividades y un diagrama de flujo. Los diagramas de flujo se limitan normalmente a procesos secuenciales; los diagramas de actividades pueden manejar procesos paralelos. Los diagramas de actividades también son útiles para los programas concurrentes, ya que se pueden plantear gráficamente cuáles son los hilos y cuándo necesitan sincronizarse Los diagramas de actividad describen el comportamiento detallado, mientras que los casos de uso describen el comportamiento global.

SÍMBOLO	SIGNIFICADO
●	**Nodo inicial**. Muestra el punto de partida de las acciones.
Nombre	**Acción**. El nombre suele ser un verbo y representa la actividad.
◇	**Decisión**. Llega una línea y salen varias o unión (al revés)
→	**Flujo o transión.** Muestra el orden de ejecución.
▬	**Concurrencia**. Inicio o final de varias acciones concurrentes.
◉	**Nodo final**. Final de todas las acciones del diagrama.
Actor	**Carriles**. Cada actividad está en el carril del actor que la ejecuta.
nombre	**Objetos.** Representa los objetos afectados por un flujo.



Diagramas de secuencia:

El diagrama de secuencia forma parte del modelado dinámico del sistema. Se modelan las llamadas entre clases desde un punto concreto del sistema. Es útil para observar la vida de los objetos en sistema, identificar llamadas a realizar o posibles errores del modelado estático, que imposibiliten el flujo de información o de llamadas entre los componentes del sistema.

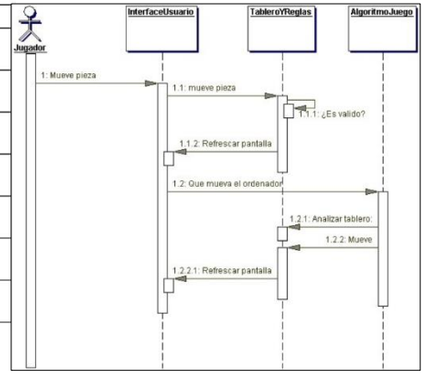
En el diagrama de secuencia se muestra el orden de las llamadas en el sistema. Se utiliza un diagrama para cada llamada a representar. Es imposible representar en un solo diagrama de secuencia todas las secuencias posibles del sistema, por ello se escoge un punto de partida. El diagrama se forma con los objetos que forman parte de la secuencia, estos se sitúan en la parte superior de la pantalla, normalmente en la izquierda se sitúa al que inicia la acción. De estos objetos sale una línea que indica su vida en el sistema. Esta línea simple se convierte en una línea gruesa cuando representa que el objeto tiene el foco del sistema, es decir cuando él está activo.

En un diagrama de secuencia se indicarán los módulos o clases que forman parte del programa y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. Se realizan diagramas de secuencia para definir acciones que se pueden realizar en la aplicación en cuestión. Así, en el caso de una aplicación para jugar al ajedrez, se podrían realizar diagramas de secuencia para “jugar una partida” o bien para acciones más específicas como “mover pieza”.

El detalle que se muestre en el diagrama de secuencia debe estar en consonancia con lo que se intenta mostrar o bien con la fase de desarrollo en la que esté el proyecto, no es lo mismo un diagrama de secuencia que muestre la acción de “mover pieza” a otro que sea “mover caballo”, o bien no es lo mismo un diagrama de secuencia “mover pieza” que verifique ciertos parámetros antes de mover como la viabilidad del movimiento con respecto a una estrategia marcada a una diagrama que no muestre este nivel de detalle por estar en una fase inicial de diseño del sistema.

El detalle del diagrama depende de la fase en la que estemos, lo que pretendamos contar con el diagrama y a quién. En una primera fase de diseño podemos poner clases grandes y ficticias, que representen un paquete/librería o, si nuestro programa está compuesto por varios ejecutables corriendo a la vez, incluso clases que representen un ejecutable. Si estamos en una fase avanzada, estamos diseñando el programa y queremos dejar bien atados los detalles entre dos programadores, que cada uno va a programar una de las clases o módulos que participan, entonces debemos posiblemente ir al nivel de clase real de codificación y método, con parámetros y todo, de forma que los programadores tengan claro que métodos van a implementar, qué deben llamar de la clase o módulo del otro, etc. Incluso si es un diagrama para presentar al cliente, podemos hacer un diagrama de secuencia en el que sólo salga el actor "jugador" y una única clase "juego ajedrez" que representa nuestro programa completo, de forma que el cliente vea qué datos y en qué orden los tiene que meter en el programa y vea qué salidas y resultados le va a dar el programa. El siguiente puede ser un diagrama de secuencia del ejemplo del ajedrez a un nivel de diseño muy preliminar.

SÍMBOLO	SIGNIFICADO
●	Punto de inicio del proceso
▭	Actividad
◇	Condicional
→	Flujo de secuencia
▬	Bifurcación o entrada
◉	Punto final del proceso
Cliente	Swinlanes ("Calles")

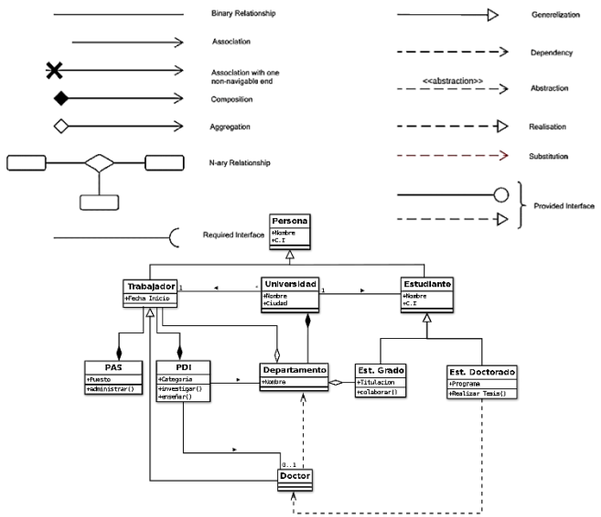


Diagramas de clases:

Forma parte de la vista estática del sistema. Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento.
Un diagrama de clases está compuesto por los siguientes elementos:
Clase: Nombre de la clase, atributos, métodos y visibilidad.
Relaciones: Herencia, Composición, Agregación, Asociación y Uso.
Diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.
Miembros: UML proporciona mecanismos para representar los miembros de la clase, como atributos y métodos, así como información adicional sobre ellos.
Visibilidad**:** Para especificar la visibilidad de un miembro de la clase (es decir, cualquier atributo o método), se coloca uno de los siguientes signos delante de ese miembro:

+	Público
-	Privado
#	Protegido
/	Derivado (se puede combinar con otro)
~	Paquete

Ámbitos: UML especifica dos tipos de ámbitos para los miembros: instancias y clasificadores y estos últimos se representan con nombres subrayados.
Los miembros clasificadores se denotan comúnmente como “estáticos” en muchos lenguajes de programación. Su ámbito es la propia clase.
Los valores de los atributos son los mismos en todas las instancias
La invocación de métodos no afecta al estado de las instancias
Los miembros instancias tienen como ámbito una instancia específica.
Los valores de los atributos pueden variar entre instancias
La invocación de métodos puede afectar al estado de las instancias (es decir, cambiar el valor de sus atributos)
Para indicar que un miembro posee un ámbito de clasificador, hay que subrayar su nombre. De lo contrario, se asume por defecto que tendrá ámbito de instancia.
Diagramas: El diagrama de clases puede tener como ejemplo: una clase que sería un objeto o persona misma en la cual se especifica cada acción y especificación.
Propiedades de objetos que tienen propiedades y/u operaciones que contienen un contexto y un dominio, los primeros dos ejemplos son clases de datos y el tercero clase de lógica de negocio, dependiendo de quién diseñe el sistema se pueden unir los datos con las operaciones.
El diagrama de clases incluye mucha más información como la relación entre un objeto y otro, la herencia de propiedades de otro objeto, conjuntos de operaciones/propiedades que son implementadas para una interfaz gráfica.
Presenta las clases del sistema con sus relaciones estructurales y de herencia.
El diagrama de clases es la base para elaborar una arquitectura MVC o MVP.



Diagramas de estados:

Una máquina de estados es cualquier dispositivo que almacena el estado de un objeto en un momento dado y puede cambiar el estado o causar otras acciones según la entrada que reciba. Estados se refiere a las diferentes combinaciones de información que un objeto puede mantener, no la forma en que el objeto se comporta. Para comprender los diferentes estados de un objeto, podrías visualizar todos los estados posibles y mostrar cómo un objeto llega a cada estado, y puedes hacerlo con un diagrama de estados UML.
Cada diagrama de estados generalmente empieza con un círculo oscuro que indica el estado inicial y termina con un círculo de contorno blanco que denota el estado final. Sin embargo, a pesar de tener puntos de inicio y finalización definidos, los diagramas de estado no necesariamente son la mejor herramienta para plasmar un desarrollo general de eventos. En lugar de ello, ilustran tipos específicos de comportamiento —en particular, cambios de un estado a otro.
Los diagramas de estado representan principalmente estados y transiciones. Los estados se representan con rectángulos de esquinas redondeadas que se etiquetan con el nombre del estado. Las transiciones se marcan con flechas que fluyen de un estado a otro, mostrando cómo cambian los estados. A continuación, podrás ver estos dos elementos en acción en un diagrama básico para la vida estudiantil. Nuestra herramienta de diagramas UML puede ayudarte a diseñar cualquier diagrama personalizado de máquina de estados.

Símbolo	Nombre
State1	Estado: Identifica un periodo de tiempo (no instantáneo) en el cual el objeto está esperando una operación.
●	Estado inicial: Representa el inicio del diagrama
▬	Fork: desprende dos estados que suceden al mismo tiempo, resultado de un único estado
▬	Join: Une dos estados que suceden al mismo tiempo, tiene como resultado un único estado.
◉	Final: Representa el final del diagrama
→	Transición: Representa el cambio de un estado a otro y las condiciones por las que pasa al siguiente estado.