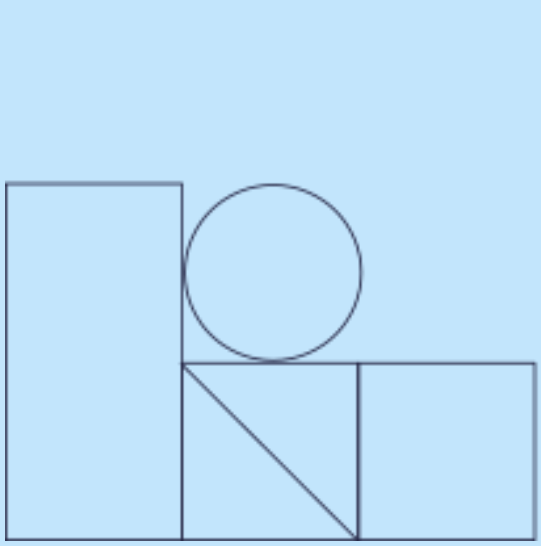
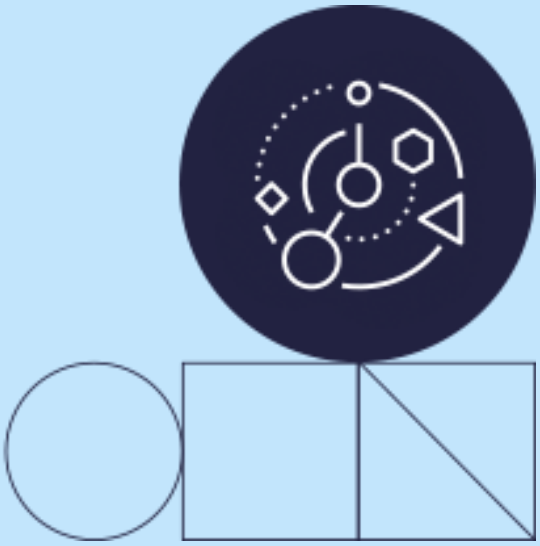
IBM SkillsBuild

IBM

POO con Python

¿Qué es UML y cómo se utiliza para la POO?





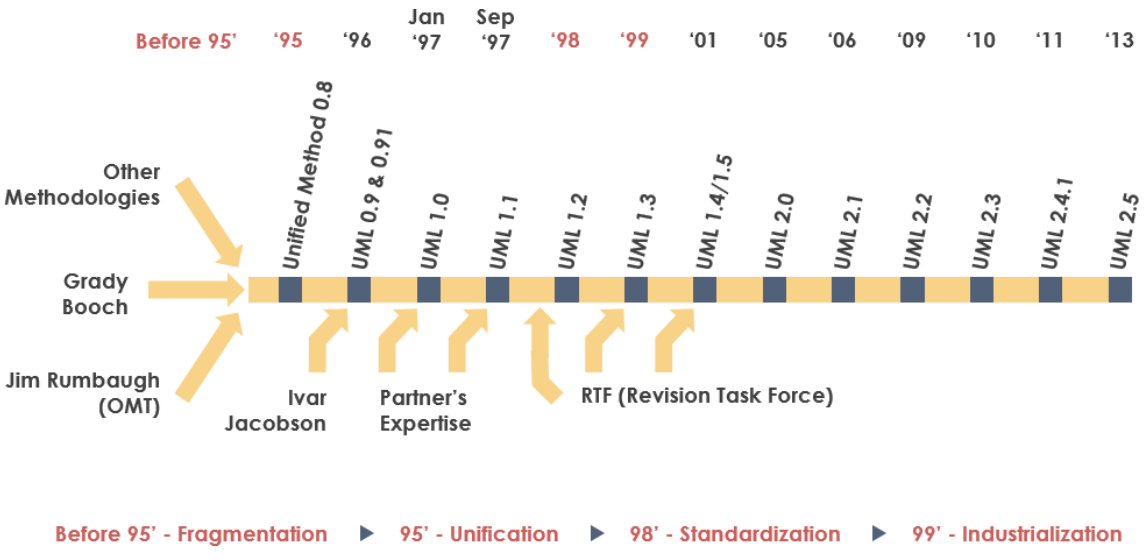
Índice

Introducción

El lenguaje de modelado unificado (UML) apareció por primera vez en la década de 1990 como un esfuerzo por seleccionar los mejores elementos de los muchos sistemas de modelado propuestos en ese momento y combinarlos en una sola notación coherente. Desde entonces, se ha convertido en el estándar de la industria para el modelado y diseño de software, así como para el modelado de otros procesos en el mundo científico y empresarial.

El UML es una herramienta para especificar sistemas de software. Tipos de diagramas estandarizados que nos ayudan a describir y mapear visualmente el diseño y la estructura de un sistema de software. Usando UML es posible modelar casi cualquier tipo de aplicación, tanto específica como independientemente de una plataforma de destino. Si bien UML está naturalmente orientado hacia la programación orientada a objetos, es igual de fácil modelar lenguajes de procedimiento como C, Visual Basic, Fortran, etc.

El uso de UML como herramienta para definir la estructura de un sistema es una forma muy útil de administrar sistemas grandes y complejos. Tener una estructura claramente visible facilita la introducción de nuevas personas en un proyecto existente.



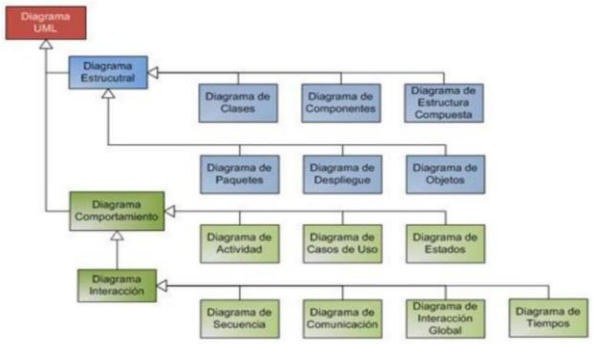
¿Por qué necesitamos UML?

- Proporcionan una representación visual del problema o de todo el sistema. De este modo, incluso alguien que no sea desarrollador puede entender cómo funciona el sistema.
- Los diagramas UML también desempeñan un papel fundamental en la resolución de problemas. Podemos representar un problema del mundo real y encontrar gradualmente la forma de resolverlo.
- Como estos diagramas son sencillos de comprender, no es necesario tener conocimientos previos para saber cómo funciona una entidad o cómo fluye un proceso.
- Al utilizar un diagrama UML holístico, todo el equipo de desarrollo de software puede colaborar y trabajar en conjunto.
- Tienen muchas aplicaciones en la programación orientada a objetos, desarrollo web, desarrollo de prototipos, análisis de negocios, entre otros.

Tipos de diagramas UML

UML no se define en un único tipo o principio de diagrama. Por el contrario, es el enfoque unificado para proporcionar diferentes tipos de representaciones visuales. Hay diferentes tipos de diagramas UML con los que podemos trabajar en función de nuestras necesidades.

Los tipos de diagramas UML pueden distinguirse como estructurales o de comportamiento. Cada una de estas clasificaciones puede tener sus tipos.



A continuación, se hará una breve descripción de los 14 tipos de diagramas que engloba UML.

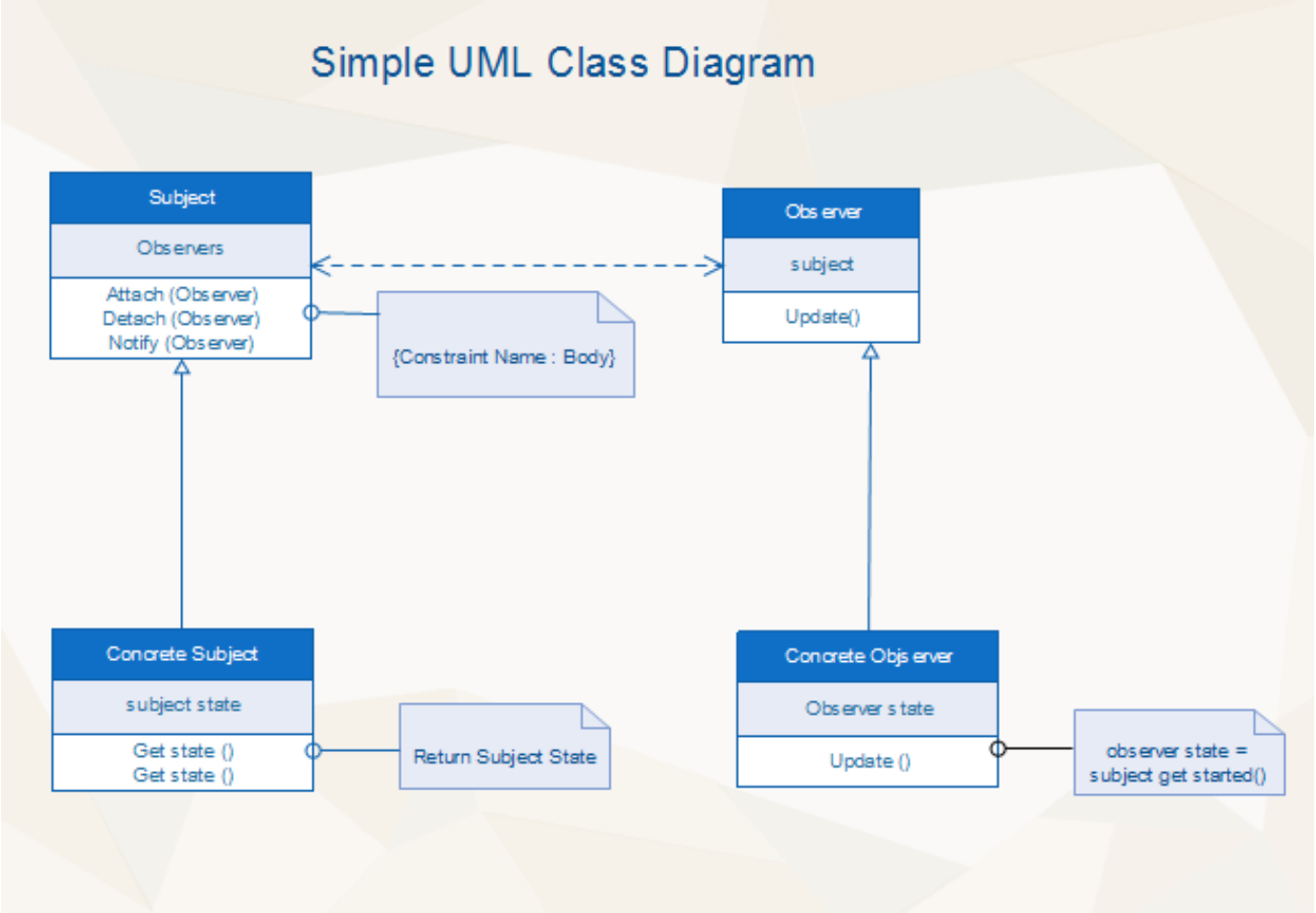
Diagramas estructurales UML

Como su nombre lo indica, representan la estructura general de un sistema: sus objetos, atributos y relaciones entre las distintas entidades. Es decir, nos preocupamos más por lo que es el sistema y no por cómo se comporta.

- **Diagrama de clases**: Describe los diferentes tipos de objetos en un sistema y las relaciones existentes entre ellos. Dentro de las clases muestra las propiedades y operaciones, así como las restricciones de las conexiones entre objetos.
- **Diagrama de objetos**: Fotos de los objetos de un sistema en un momento del tiempo.
- **Diagrama de paquetes**: Muestra la estructura y dependencia entre paquetes, los cuales permiten agrupar elementos (no solamente clases) para la descripción de grandes sistemas.
- **Diagrama de despliegue**: Muestra la relación entre componentes o subsistemas software y el hardware donde se despliega o instala.
- **Diagrama de estructura compuesta**: Descompone jerárquicamente una clase mostrando su estructura interna.
- **Diagrama de componentes**: Muestra la jerarquía y relaciones entre componentes de un sistema software.

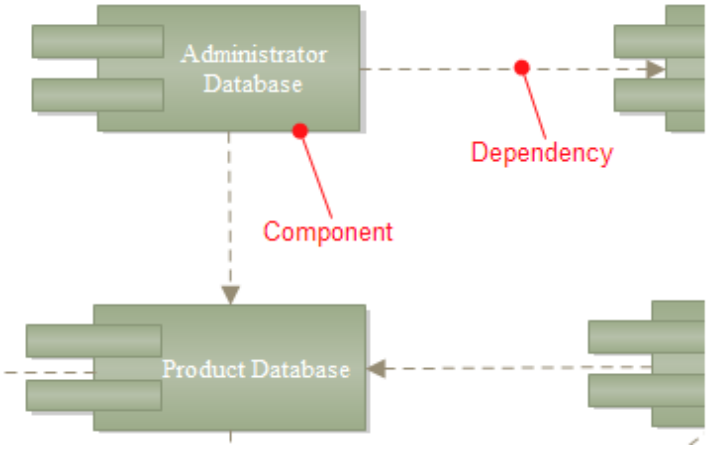
1. Diagrama de clases

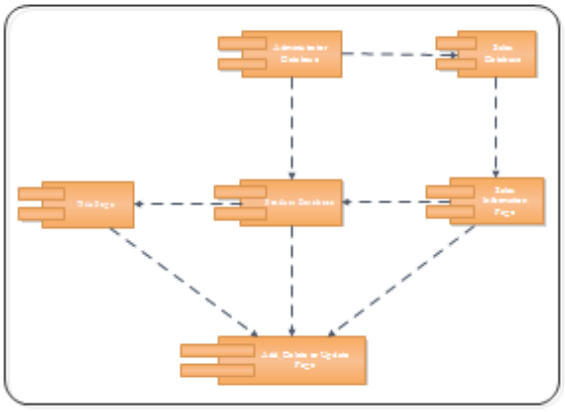
Este diagrama UML básico representa la bifurcación de un sistema en clases individuales. Utilizamos estos tipos de diagramas UML para proporcionar la representación estática del programa. Una clase tiene tres elementos principales: su nombre, sus atributos y su comportamiento.



2. Diagrama de componentes

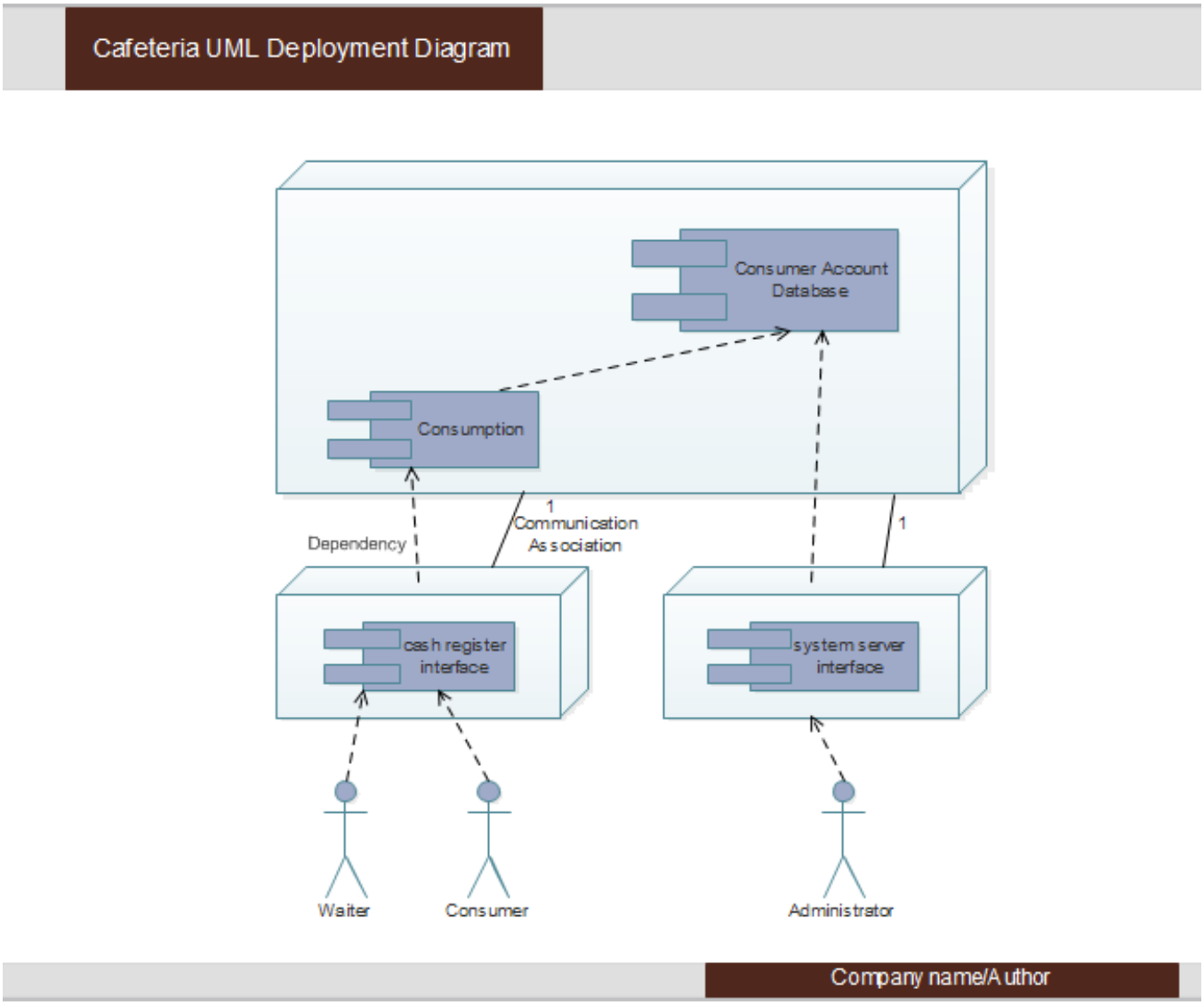
Los diagramas de componentes se crean sobre todo cuando un sistema es complejo y consta de demasiadas clases. Por lo tanto, desglosamos todo el sistema en diferentes componentes y mostramos cómo dependen unos de otros. Estos tipos de diagramas UML sólo constan de dos cosas, los componentes y las dependencias.





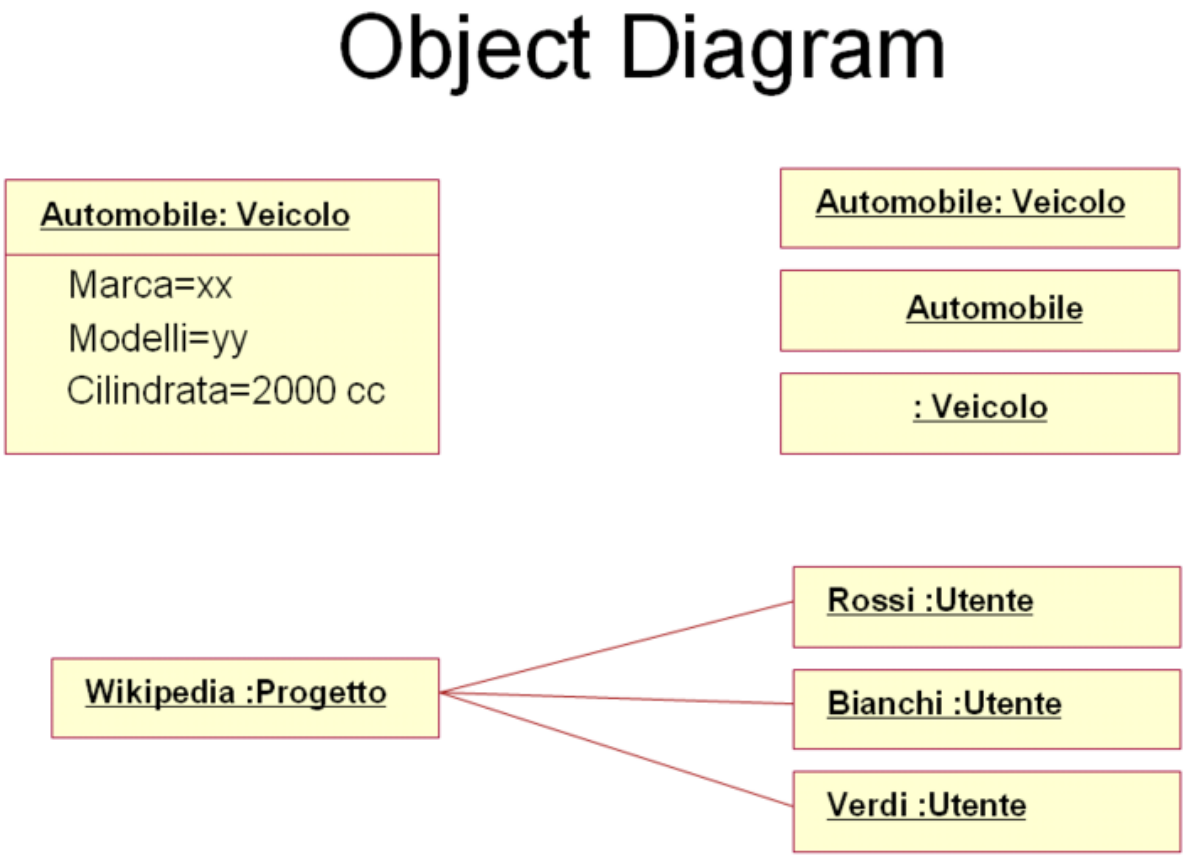
3. Diagrama de despliegue

Esto proporciona la representación real del hardware del sistema (con la inclusión del componente de software en él). Este diagrama UML sigue un enfoque más real sobre cómo se desplegaría el sistema. Consta de dos unidades principales: nodos y artefactos. Los nodos representan el servidor de la base de datos o la unidad de hardware, mientras que los artefactos representan los clientes o los esquemas.



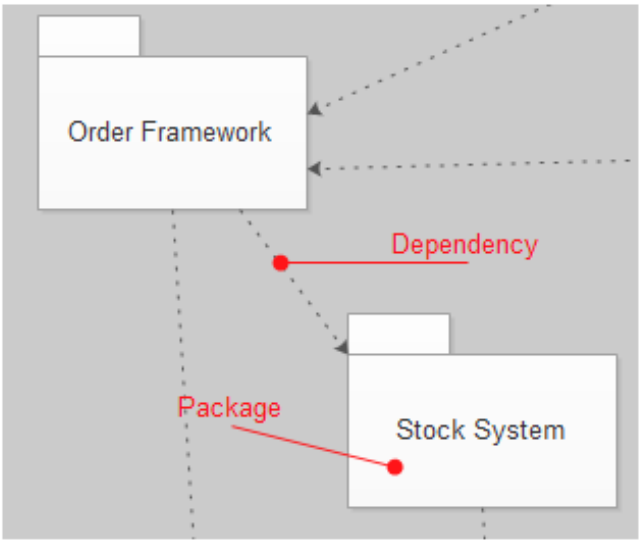
4. Diagrama de objetos

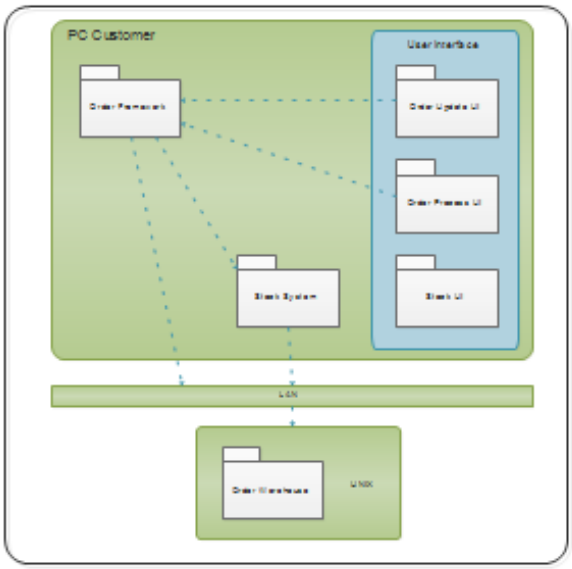
Considera esto como un desglose de un diagrama de clases (ya que una clase es una colección de diferentes objetos). Sin embargo, estos tipos de diagramas UML se basan principalmente en entidades del mundo real. Presenta diferentes objetos con sus enlaces. Cada entidad tiene un nombre de objeto y su conjunto de atributos.



5. Diagrama de paquetes

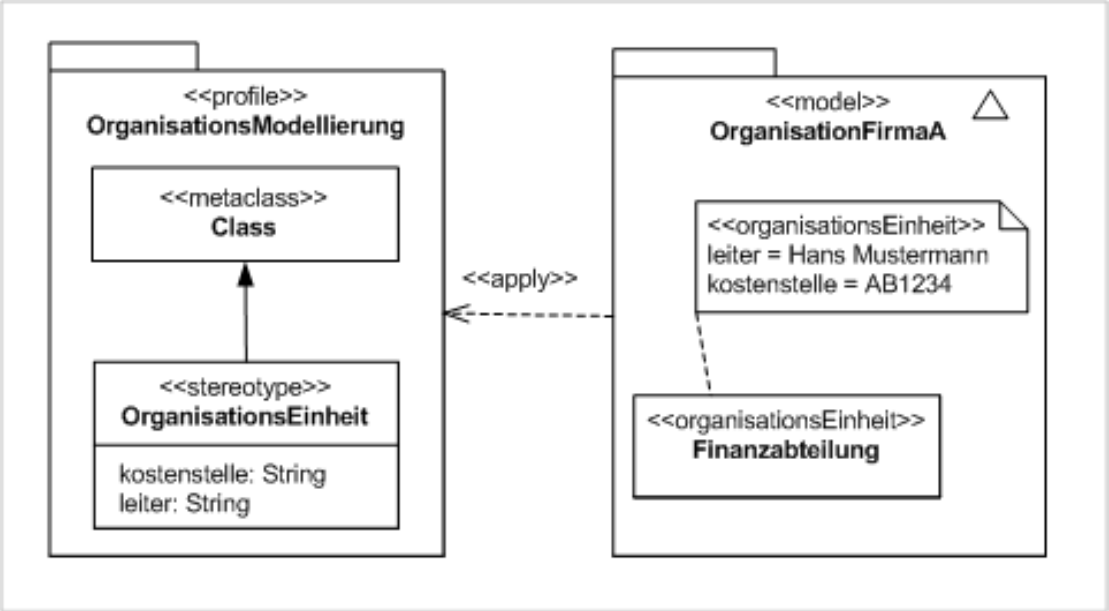
Esto proporciona una representación de un sistema con sus módulos y subsistemas de más alto nivel. En un diagrama de paquetes, podemos incluir varias estructuras de diagrama de despliegue. Se puede desglosar fácilmente todo el sistema y vincular sus componentes. Un paquete está representado por un icono de carpeta con un nombre. Un paquete puede tener diferentes paquetes para representar un enfoque descendente.





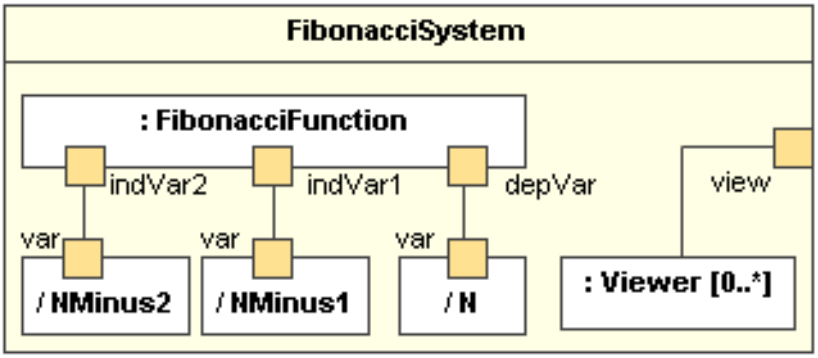
6. Diagrama de perfil

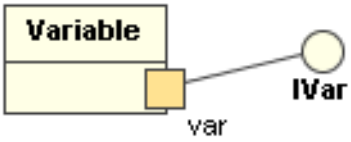
Se trata de un nuevo tipo de diagrama UML que se introdujo en UML 2. Aunque estos tipos de diagramas UML no son tan populares, pueden utilizarse para representar la meta estructura del sistema. Es decir, cuáles son los perfiles principales, meta clases, y demás. Puede haber conexiones dentro de los elementos de un perfil y entre diferentes perfiles.



7. Diagrama de la estructura compuesta

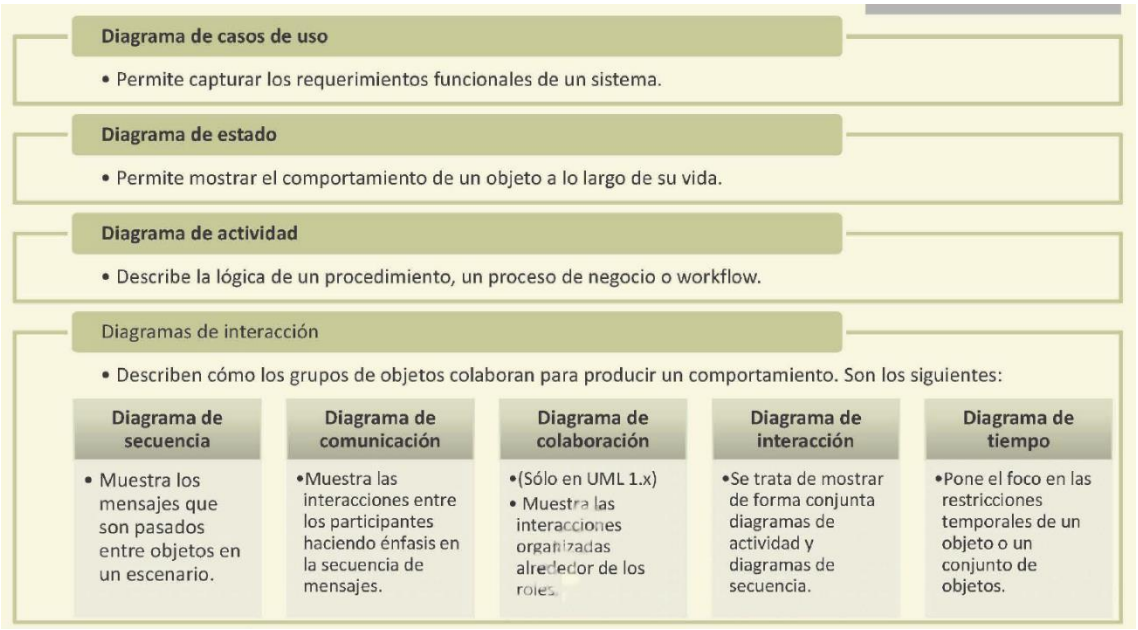
Estos tipos de diagramas UML se utilizan para representar la estructura interna de una clase. Con ellos podemos saber cómo se relacionan las diferentes entidades de una clase entre sí y cómo se asocia la propia clase con la entidad o sistema exterior. Cada elemento tiene un "rol" asignado aquí.





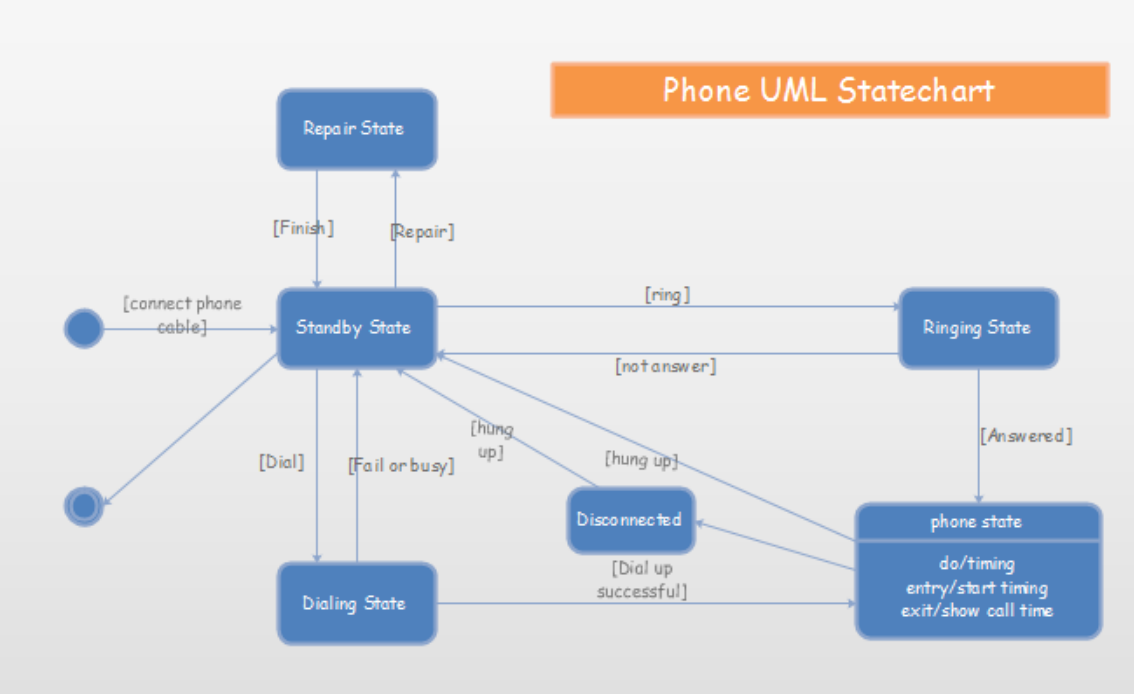
Diagramas UML de comportamiento

Estos diagramas definen el comportamiento del sistema y la funcionalidad general de cada unidad. Además, algunos de estos diagramas representan también el flujo de información y control en el sistema.



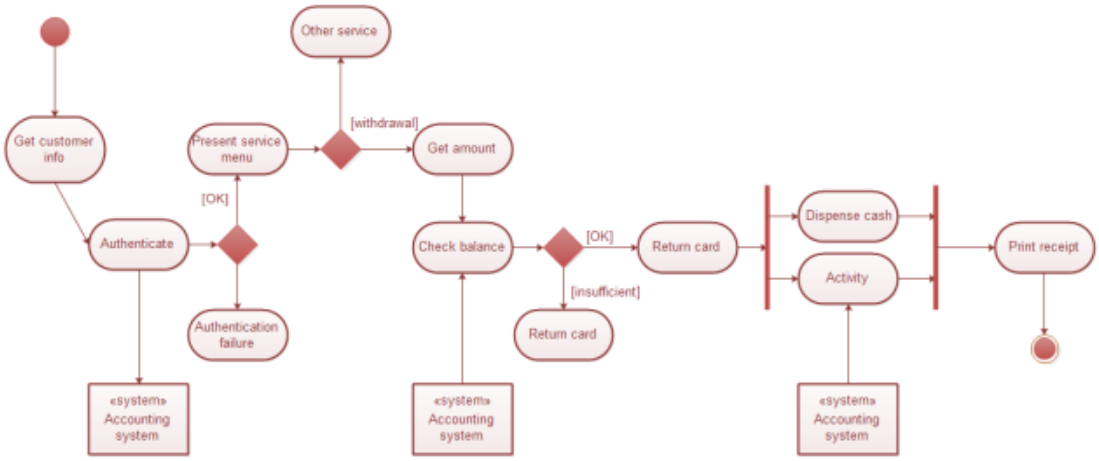
8. Diagrama de máquina de estado

Se trata de un diagrama dinámico que describe los diferentes estados del sistema. Presenta todo tipo de acciones y cómo respondería una clase a ellas. En consecuencia, se establece un flujo centrado en el estado de las entidades centrales.



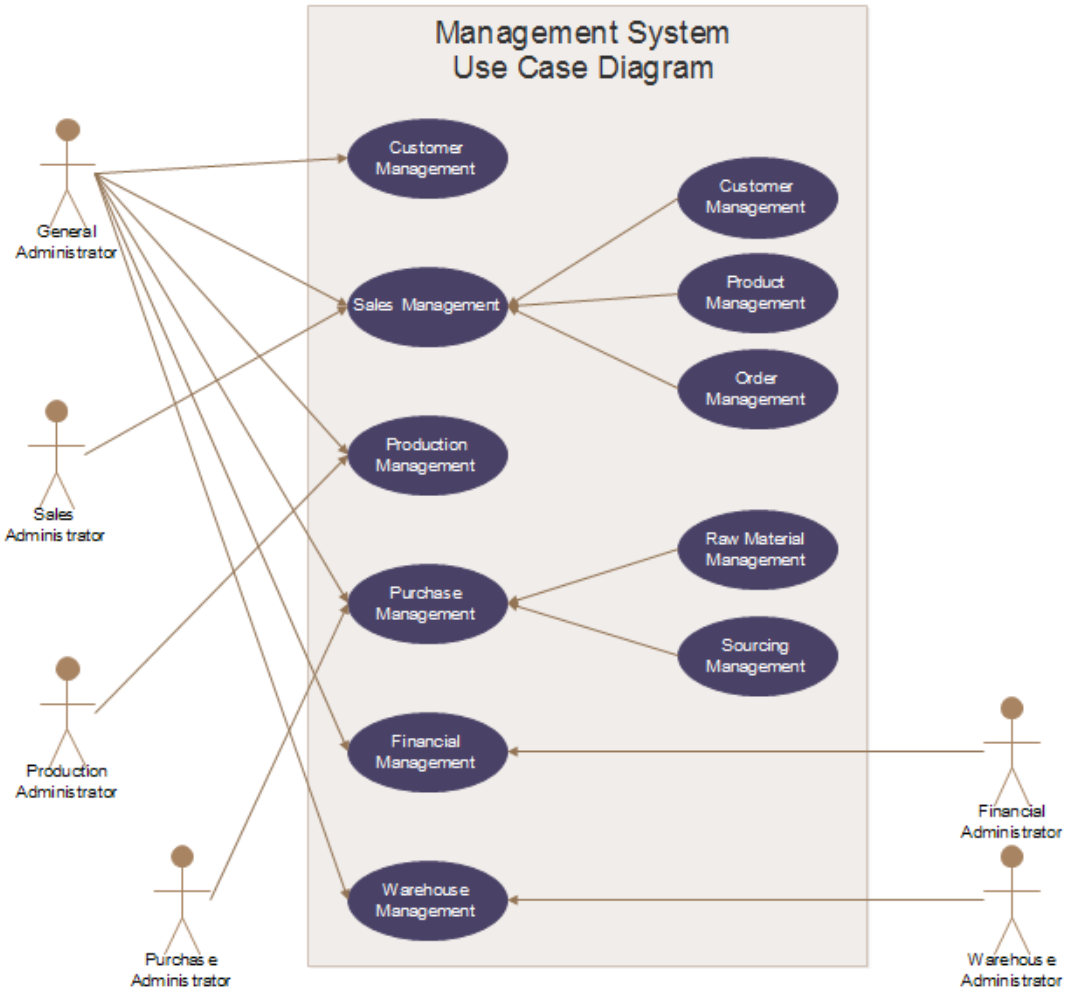
9. Diagrama de actividad

Estos tipos de diagramas UML se utilizan ampliamente, ya que representan el flujo global del sistema con respecto a numerosas actividades. Toma una instancia del sistema y registra cómo reacciona. Estos diagramas se centran más en la implementación del sistema en el mundo real que en su estructura.



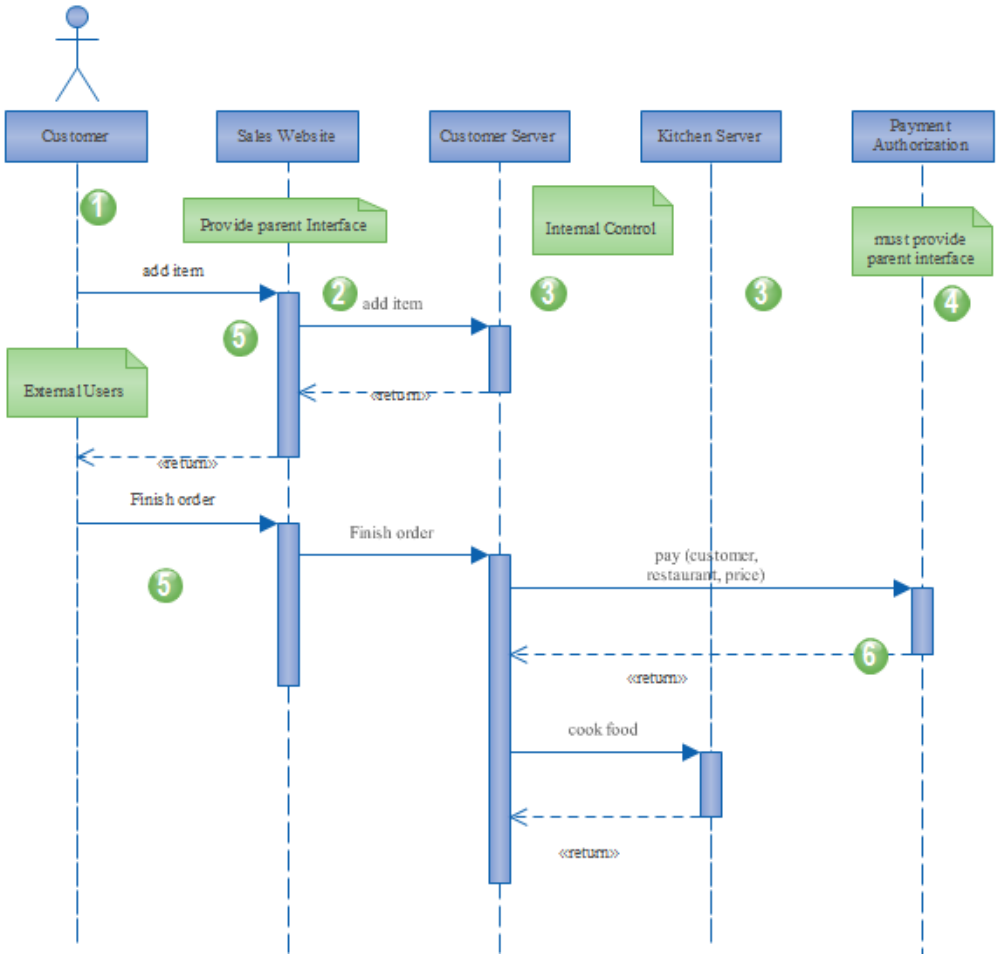
10. Diagrama de casos de uso

Este es uno de los tipos más populares de diagramas UML que representan cómo el usuario final interactuaría con el sistema. Presenta diferentes actores clave (usuarios) y su interacción con componentes específicos del sistema. Aparte del desarrollo de software, también desempeña un papel fundamental en su despliegue.



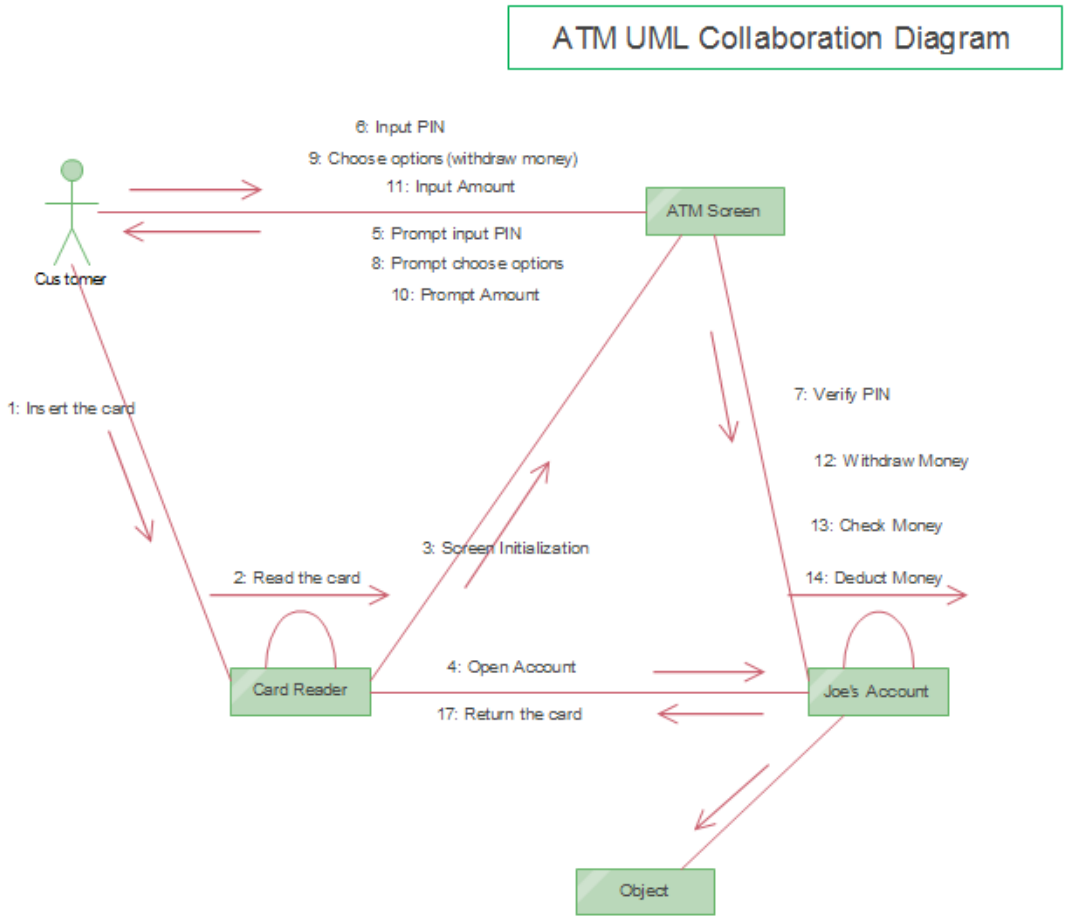
11. Diagrama secuencial

Como su nombre lo indica, describe el flujo de un proceso entre varios componentes de manera secuencial. Se utiliza sobre todo en el desarrollo de la arquitectura y en la implementación del sistema en el mundo real. Presenta diferentes componentes y describe cómo el flujo pasa de un componente a otro.



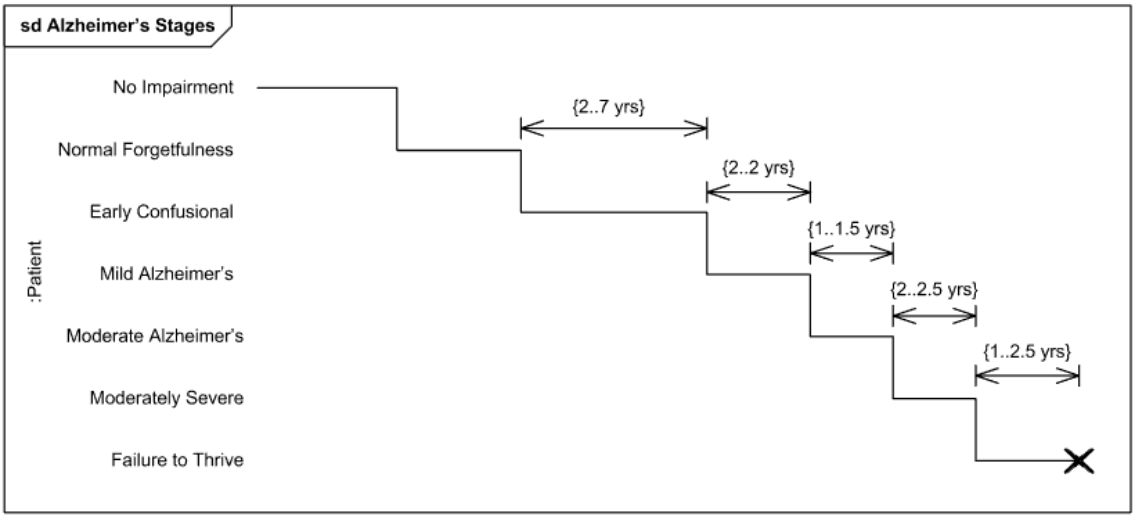
12. Diagrama de comunicación

También conocido como diagrama de colaboración, representa de forma sencilla cómo se comunican entre sí las distintas entidades de un sistema. Aparte de los objetos, también presenta flechas y puntos numéricos para mostrar la comunicación global en el sistema.



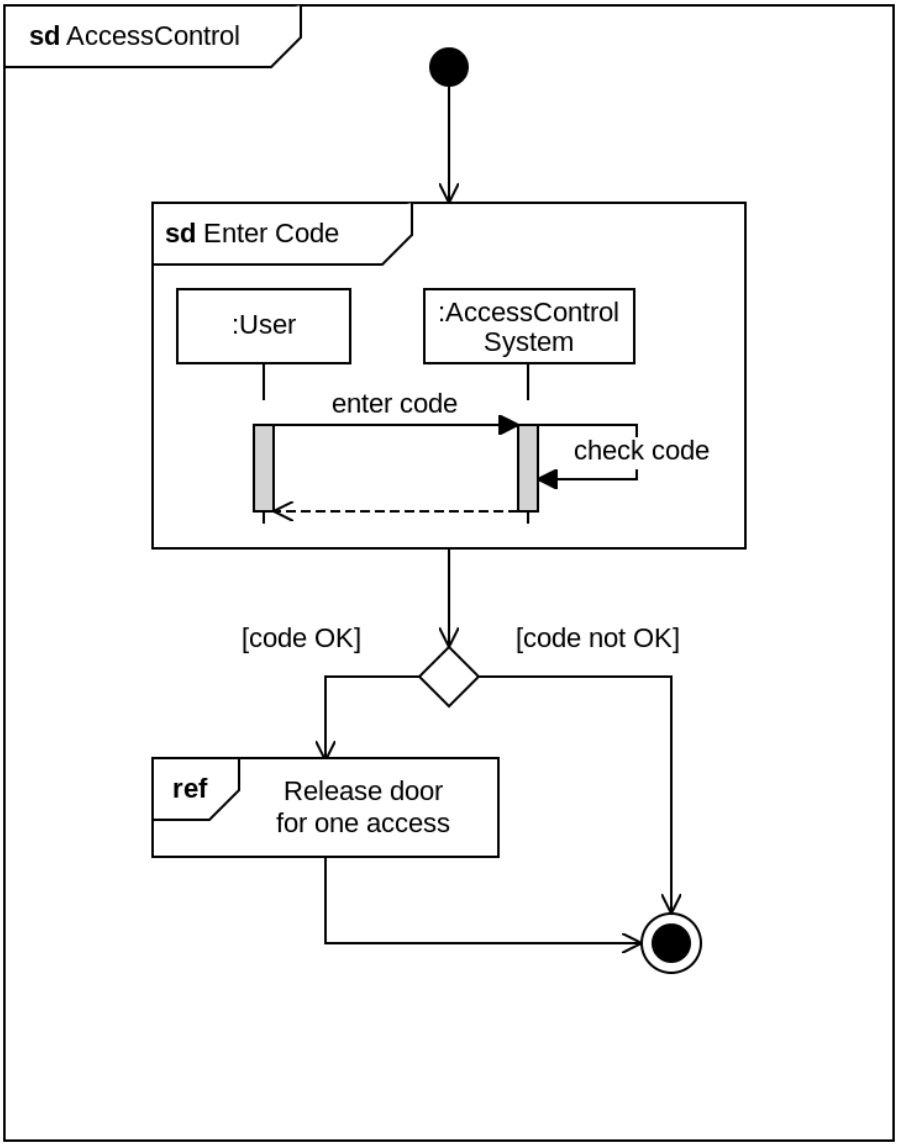
13. Diagrama de tiempos

Aunque son similares a los diagramas de secuencia, representan el comportamiento de cualquier objeto a lo largo de una duración específica. Estos tipos de diagramas UML se añadieron a UML 2 y se centran en el tiempo como restricción.



14. Diagrama general interactivo

Al principio, estos tipos de diagramas UML pueden parecer similares a los diagramas de actividad. Sin embargo, en lugar de actividades, muestran el flujo de varias secuencias interactivas. Una actividad está representada por un marco que puede tener diferentes unidades en su interior.



Diagramas de clases

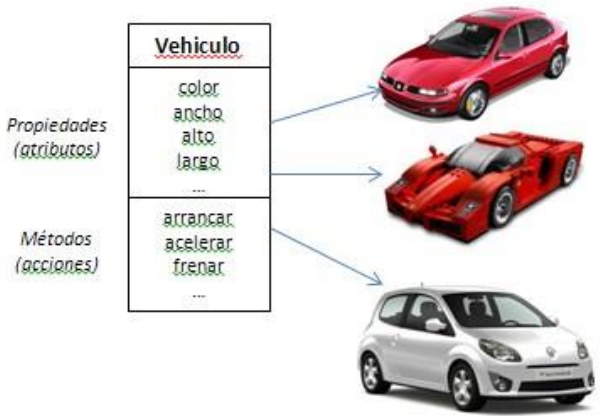
Una vez vistas las características generales de los diferentes tipos de diagramas UML, vamos a ver con más profundidad los diagramas de clases, ya que son los más usados para representar código escrito con POO y el tipo de diagramas que mayormente usaremos como programadores. De paso, aprovecharemos para repasar las principales características de los elementos del lenguaje Python.

El diagrama de clases muestra los componentes básicos de cualquier sistema orientado a objetos. Los diagramas de clase representan una vista estática del modelo, o parte del modelo, que describe qué atributos y comportamiento tiene en lugar de detallar los métodos para lograr las operaciones. Los diagramas de clases son más útiles para ilustrar las relaciones entre clases e interfaces. Las generalizaciones, agregaciones y asociaciones son valiosas para reflejar la herencia, la composición o el uso y las conexiones, respectivamente.

Los diagramas de clase muestran las distintas clases que componen un sistema y cómo se relacionan entre sí. Se dice que los diagramas de clase son «estáticos» porque muestran las clases, junto a sus métodos y atributos, así como las relaciones estáticas entre ellos: qué clases «conocen» a otras clases, o qué clases «son parte» de otra clase, aunque no muestran las llamadas a los métodos entre ellas.

Una clase define los atributos y los métodos de un conjunto de objetos. Todos los objetos de una clase (instancias de la clase) comparten el mismo comportamiento y poseen el mismo conjunto de atributos (cada objeto tiene su propio conjunto). A veces se usa el término «tipo» en lugar de «clase», pero es importante tener presente que no son lo mismo, ya que «tipo» es un término más general.

En UML, las clases se representan por rectángulos con el nombre de la clase, que pueden mostrar los atributos y las operaciones de la clase en dos «compartimentos» dentro del rectángulo.



Clases:

- + attr1: int
- +attr2: stirng
- + operation1(p : bool) : double
- # operation2 ()

Representación visual de una clase en UML.

Atributos

En UML, los atributos se muestran como mínimo con su nombre, aunque también pueden mostrar su tipo, su valor inicial y otras propiedades. Los atributos también se pueden mostrar con su visibilidad:

+ representa atributos **públicos**

representa atributos **protegidos**

- representa atributos **privados**

Métodos

Las operaciones (métodos) también se muestran como mínimo con su nombre, aunque pueden mostrar también sus parámetros y los tipos que devuelven. Las operaciones pueden, al igual que los atributos, mostrar su visibilidad:

+ representa operaciones **públicas**

representa operaciones **protegidas**

- representa operaciones **privadas**

Asociaciones de clases

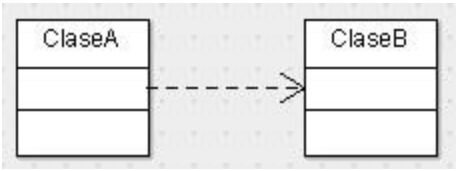
Las clases se pueden relacionar (estar asociadas) con otras de diferentes modos:

Dependencia

(...depende de...) (...restringido a...) (...usa...)

Es una relación de uso entre dos clases (una usa a la otra). Esta relación es la más básica entre clases y comparada con los demás tipos de relación, la más débil.

Se representa con una flecha discontinua que parte desde una clase y apunta a otra. El sentido de la flecha nos indica quien usa a quien.



Del diagrama anterior podemos observar que:

- La ClaseA usa a la ClaseB.
- La ClaseA depende de la ClaseB.
- Dada la dependencia, todo cambio en la ClaseB podrá afectar a la ClaseA.
- La ClaseA conoce la existencia de la ClaseB pero la ClaseB desconoce que existe la ClaseA.

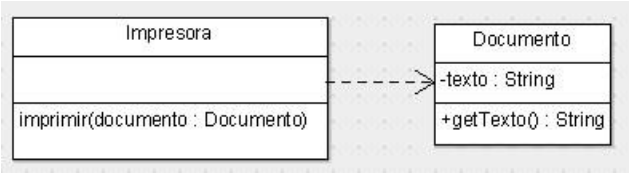
Ejemplo Práctico:

Tenemos una clase Impresora.

Tenemos una clase Documento con un atributo texto.

La clase Impresora se encarga de imprimir los Documentos.

Para esto generamos una relación de dependencia:

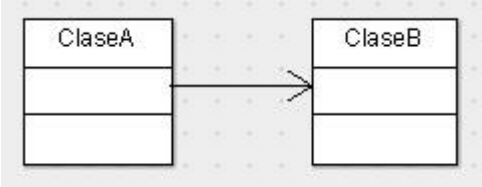


Asociación

(...conoce...) (...trabaja para...)

Es una relación de estructura entre clases, es decir, una entidad se construye a partir de otra u otras. Aunque este tipo de relación es más fuerte que la Dependencia es más débil que la Agregación, ya que el tiempo de vida de un objeto no depende de otro.

Se representa con una flecha continua que parte desde una clase y apunta a otra. El sentido de la flecha nos indica la clase que se compone (base de la flecha) y sus componentes (punta de la flecha).



Del diagrama anterior podemos observar que:

- La ClaseA depende de la ClaseB.
- La ClaseA está asociada a la ClaseB.
- La ClaseA conoce la existencia de la ClaseB pero la ClaseB desconoce que existe la ClaseA.
- Todo cambio en la ClaseB podrá afectar a la ClaseA.

Esto significa que la ClaseA tendrá como atributo un objeto o instancia de la ClaseB (su componente). La ClaseA podrá acceder a las funcionalidades o atributos de su componente usando sus métodos.

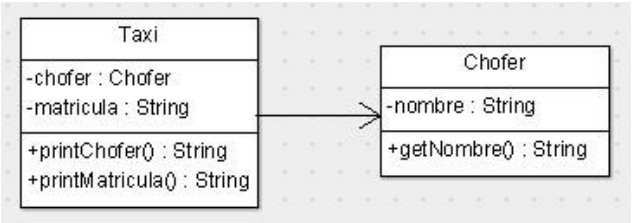
Ejemplo Práctico:

Tenemos una clase Taxi con un atributo matricula.

Tenemos una clase Chofer con un atributo nombre.

Cada Taxi necesita ser conducido por un Chofer.

Taxi necesita acceder a algunos de los atributos de su Chofer (por ejemplo, su nombre).

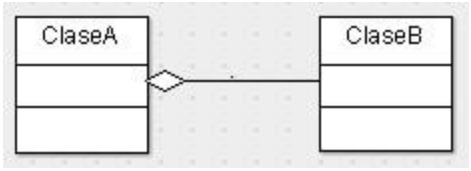


Agregación

(...es accesorio de...)

Es muy similar a la relación de Asociación solo varía en la multiplicidad ya que en lugar de ser una relación "uno a uno" es de "uno a muchos".

Se representa con una flecha que parte de una clase a otra en cuya base hay un rombo de color blanco.



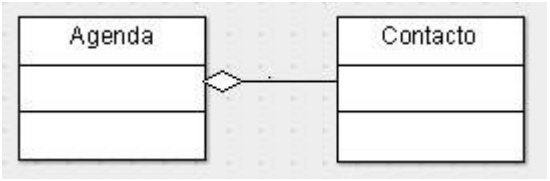
La ClaseA agrupa varios elementos del tipo ClaseB.

Ejemplo:

Tenemos una clase Agenda.

Tenemos una clase Contacto.

Una Agenda agrupa varios Contactos.

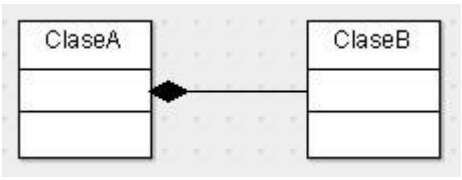


Composición

(...es parte de...)

Similar a la relación de Agregación solo que la Composición es una relación más fuerte. Aporta documentación conceptual ya que es una "relación de vida", es decir, el tiempo de vida de un objeto está condicionado por el tiempo de vida del objeto que lo incluye.

Se representa con una flecha que parte de una clase a otra en cuya base hay un rombo de color negro.



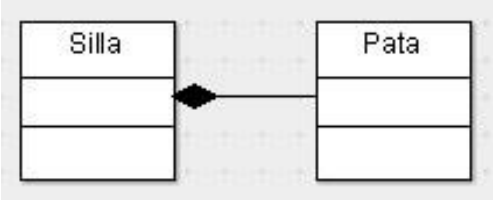
- La ClaseA agrupa varios elementos del tipo ClaseB.
- El tiempo de vida de los objetos de tipo ClaseB está condicionado por el tiempo de vida del objeto de tipo ClaseA.

Ejemplo:

Tenemos una clase Silla.

Un objeto Silla está a su vez compuesto por cuatro objetos del tipo Pata.

El tiempo de vida de los objetos Pata depende del tiempo de vida de Silla, ya que si no existe una Silla no pueden existir sus Patas.



Generalización

(...es un...)

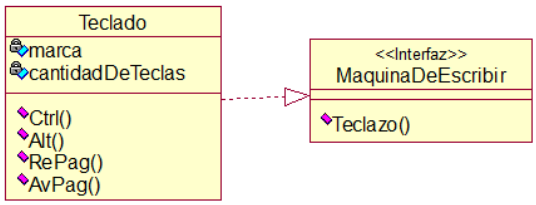
Es una relación de herencia. Se puede decir que es una relación “es un tipo de”.

En UML la herencia se representa mediante una flecha, cuya punta es un triángulo vacío. La flecha que representa a la herencia va orientada desde la subclase a la superclase.

Realización (Interfaces)

Una clase especifica una serie de condiciones que la otra tiene que cumplir. Es la representación típica de las interfaces. Una interfaz es un conjunto de operaciones que especifica cierto aspecto de la funcionalidad de una clase, y es un conjunto de operaciones que una clase presenta a otras. Y la relación entre una clase y una interfaz se conoce como realización.

La realización se indica con una línea discontinua y con una flecha al final sin rellenar o también puede representarse con un pequeño círculo que conecte con una línea a una clase.



Ejemplo de diagrama UML: