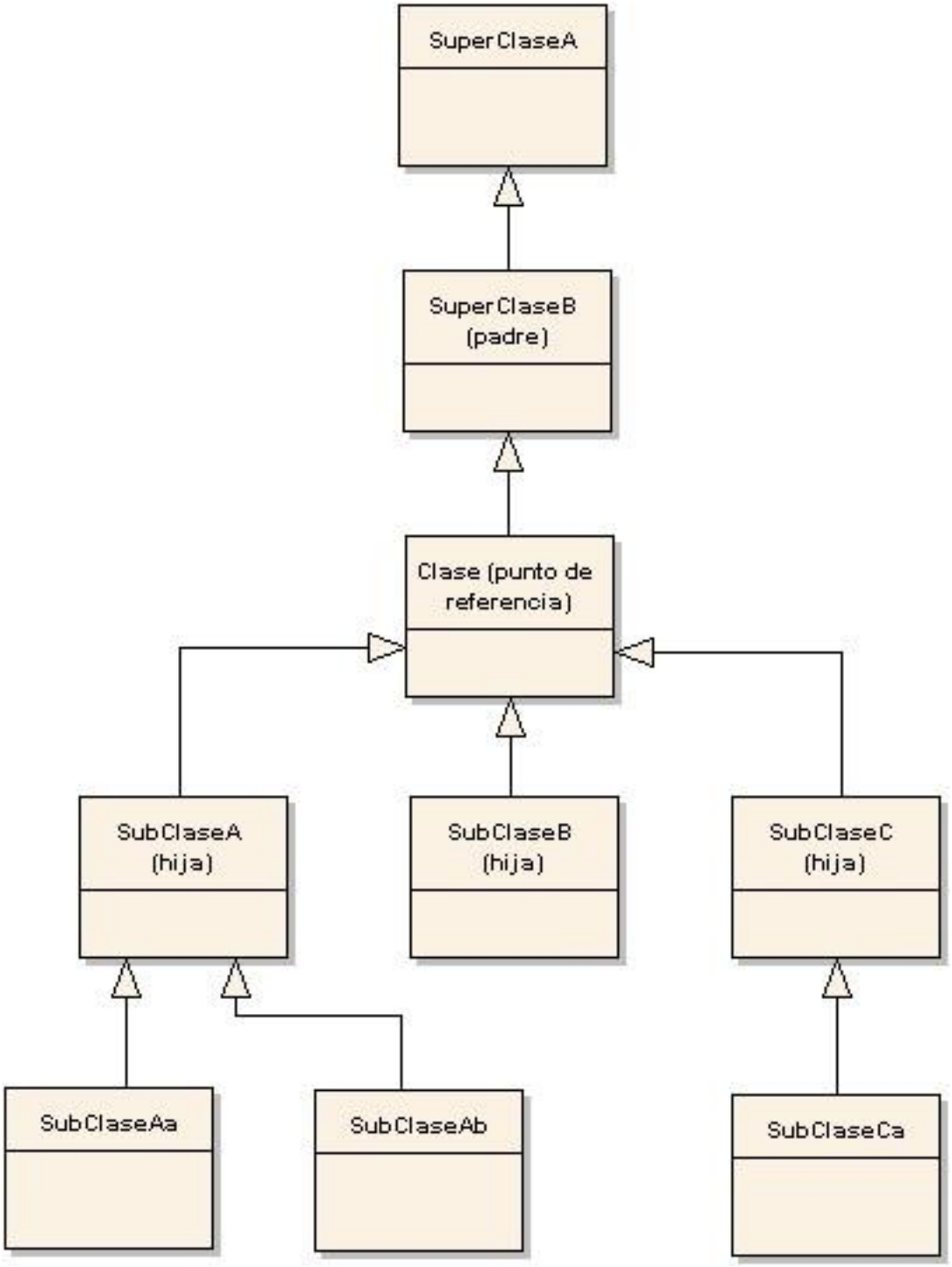
Mecanismo de Herencia

El mecanismo de herencia es otro de los conceptos clave de la POO, y uno de sus puntos fuertes. Este mecanismo nos permite que una clase "herede de otra clase" o "extienda otra clase" recibiendo o heredando todos los atributos y operaciones de su clase "padre".

En el siguiente diagrama podemos ver cómo está compuesto un típico árbol de herencia simple:

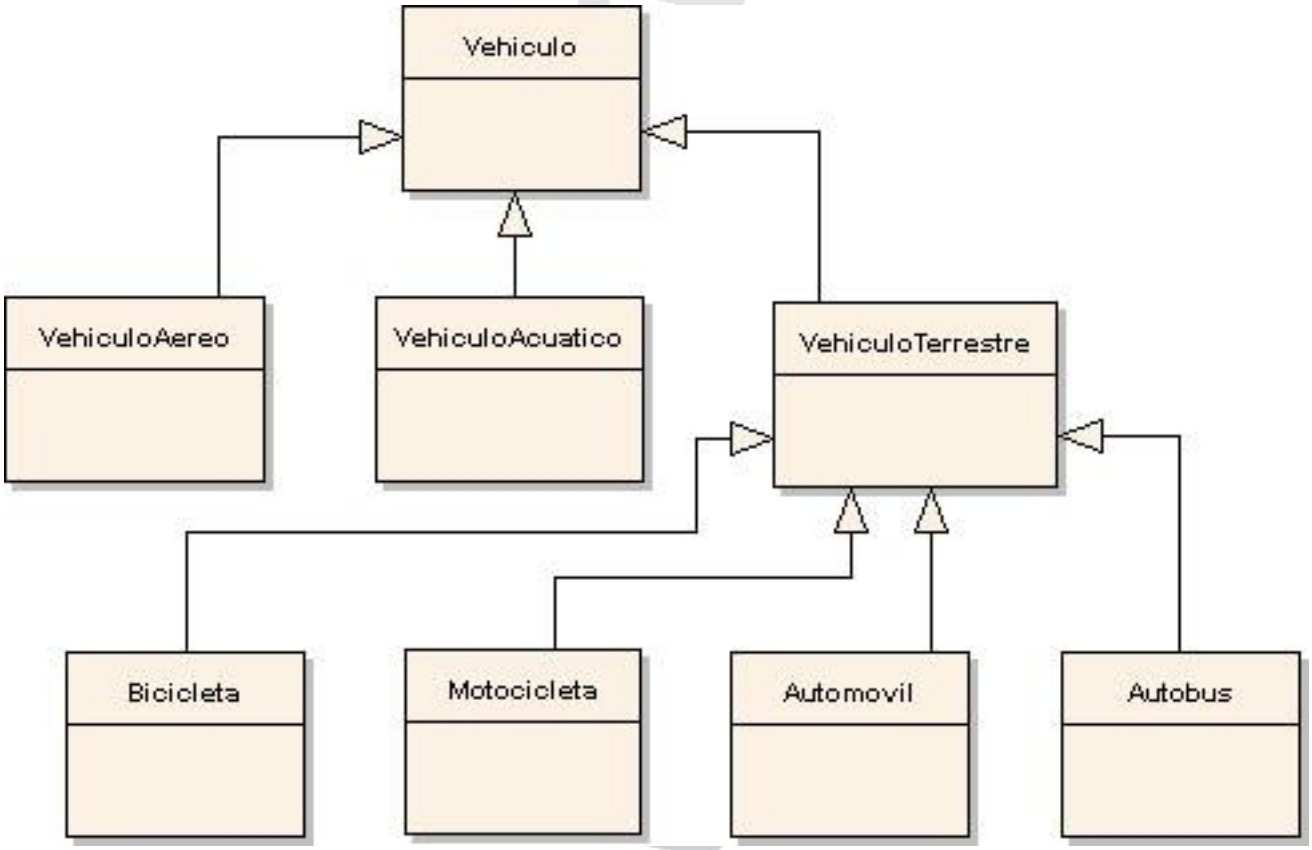


En el diagrama anterior vemos como una clase puede tener tanto “superclases” como “subclases”. Esto significa que una clase hereda de una “superclase” o clase padre, y es heredada por “subclases” o clases hijas. La flecha indica “quien hereda a quien”, es decir, el origen de la flecha es la clase hija y el destino de la flecha es la clase padre.

Se considera clases hijas a las clases que heredan directamente de una clase y subclases tanto a las clases hijas como al resto de clases que tengan menor jerarquía en el árbol de herencia; se considera clase padre a la clase de la que se hereda directamente.

Cuando una clase hereda de otra, todas las operaciones y atributos pasan a estar disponibles en la clase hija, es decir que partimos de todos estos elementos e incluso podemos agregar nuevos o modificar estos ya heredados. Con las clases heredadas ampliamos la descripción de un concepto de la vida real, lo volvemos más específico. Se mencionará algo más sobre esto más adelante.

Para comprender mejor la herencia y la naturaleza de las clases heredadas, es necesario entender que una clase hija es más específica conceptualmente que la clase padre, a la vez que la clase padre es más general conceptualmente que la clase hija. Esto lo podemos visualizar mejor de la siguiente manera:

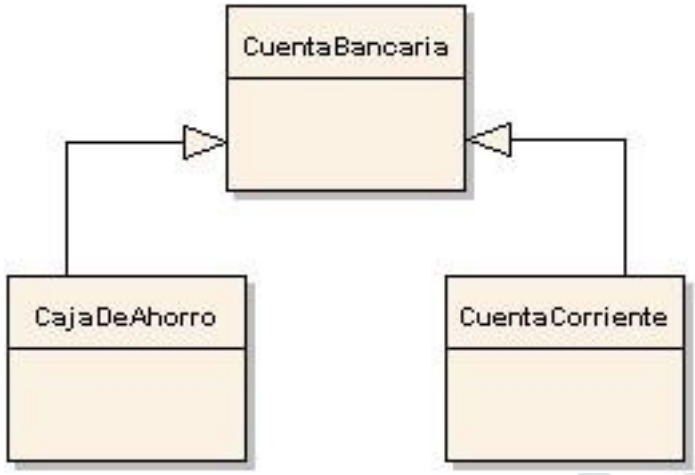


En este diagrama podemos ver a la clase padre “Vehículo” y su árbol de herencia. Por comodidad no hemos colocado las clases hijas de las clases “VehículoAereo” y “VehículoAcuatico”. ¿Qué tendrían en común las clases “Avion” y “Motocicleta”? Que ambas clases tienen como superclase a la clase vehículo, es decir, ambas SON “Vehículo”.

Es importante destacar que una clase ES UN cualquiera de sus superclases, es decir: “Motocicleta” ES UN “VehículoTerrestre” y ES UN “Vehículo”,

Con cada subclase, se amplía la cantidad de elementos de dicha clase con respecto a sus superclases, volviéndola más específica. Así, por ejemplo, la clase vehículo hace referencia a cualquier tipo de transporte, en cambio la clase “VehículoTerrestre” hace referencia a cualquier tipo de transporte que se desplace en un medio sólido, y la clase “Automóvil” hace referencia a un tipo de transporte específico. Podríamos ir más allá en el árbol de herencia y crear clases para cada modelo de automóvil y hacer que extiendan a la clase “Automóvil”.

Otro árbol de herencia podría ser el siguiente:



En este caso, la clase padre es “CuentaBancaria” y las clases hijas son “CajaDeAhorro” y “CuentaCorriente”, donde “CajaDeAhorro” ES UNA “CuentaBancaria” y “CuentaCorriente” ES UNA “CuentaBancaria”

Resumiendo, el mecanismo de herencia es la capacidad que tienen las clases de recibir el comportamiento (operaciones) y estado (atributos) de otras clases.

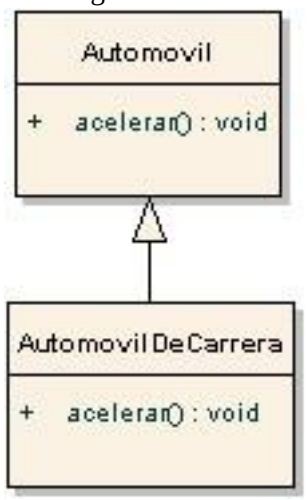
Polimorfismo

Otro mecanismo presente en el POO es el polimorfismo. En el POO encontramos dos tipos de polimorfismo, uno de ellos ya lo vimos y se conoce comúnmente como "Sobrecarga de Operaciones". El otro, el que veremos ahora, tiene que ver con la capacidad que tienen las clases de redefinir el comportamiento de una operación heredada de la clase padre.

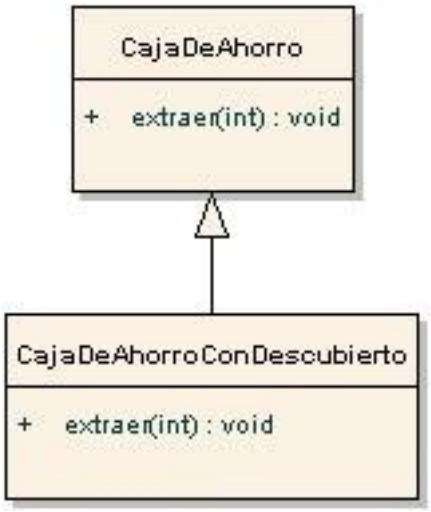
Esto significa que, en una clase padre conceptualmente más general, al tener un método al cual le dimos cierto comportamiento básico, que puede no ser exactamente lo que debe suceder al invocar este método desde un objeto correspondiente a una clase hija, debemos sobrescribir esa operación en la clase hija con la funcionalidad o comportamiento apropiado. Si la clase hija no redefiniera ese comportamiento, el comportamiento sería el mismo de la clase padre (tras invocar a dicho método).

Por lo cual, tendríamos una clase padre con la operación “hacerAlgo” que hace “x” cosa, y una clase hija con la operación “hacerAlgo” que hace “y” cosa., es decir que la misma operación tiene comportamientos distintos según la clase

En el siguiente caso:



La clase “AutomovilDeCarrera” debe reemplazar el comportamiento de la operación “acelerar()” heredado de la clase “Automóvil”, ya que el tipo de aceleración es completamente distinta. Lo mismo sucede con las siguientes clases:



En este caso, las operaciones “extraer()” de ambas clases deben tener comportamientos distintos debido a que el límite de dinero a extraer es distinto en cada clase. La clase “CajaDeAhorro” no tiene la posibilidad de extraer en descubierto, sin embargo la clase “CajaDeAhorroConDescubierto” permite extraer dinero aun cuando ya no hay mas saldo en la caja. Es decir que ambas clases permiten extraer dinero pero de formas distintas.

En resumen, el polimorfismo es el mecanismo que permite que un objeto sea tratado como alguna de sus superclases conservando los miembros de su propia clase.

Atributos de Clase

También llamados atributos estáticos, son atributos que pertenecen a la clase y no a un objeto o instancia de clase. Esto significa que son atributos compartidos por todos los objetos.

Por ejemplo en una clase “CajaDeAhorroConDescubierto” podríamos declarar un atributo estático llamado “descubierto” que contenga el valor en pesos de descubierto que puede brindar esa caja de ahorro a los clientes, es decir, no necesitamos que cada instancia de “CajaDeAhorroConDescubierto” almacene un valor independiente, sino tan solo uno que sirve para todas las instancias. Este es el tipo de usos que se le da a los atributos estáticos.

En el diagrama anterior el atributo “descubierto” aparece subrayado, esto es lo que nos indica que es un atributo estático. Lo mismo sucedería para una clase “Automovil” con un atributo estático “velocidadMaxima”:

En resumen, los atributos de clase (o estáticos) son atributos que existen en la clase y no en cada uno de los objetos, motivo por el cual son compartidos por todos los objetos.

Operaciones de Clase

También llamadas operaciones estáticas, son operaciones que corresponden a la clase y no a las instancias de dicha clase, es decir a los objetos.

La explicación es análoga a la presentada en el tema anterior de atributos de clase.

En los siguientes ejemplos se aplican operaciones estáticas:

Se debe observar que en ambas clases se dispone de un atributo estático y dos operaciones estáticas, y en este caso ambas operaciones estáticas se utilizan modificar los atributos estáticos.

En resumen, las operaciones de clase (o estáticas) son operaciones que existen en la clase y no en cada uno de los objetos, motivo por el cual son compartidos por todos los objetos.

INTRODUCCIÓN A UML

Qué es UML

UML es un lenguaje gráfico que permite modelar, visualizar y documentar sistemas. Está compuesto por distintos diagramas que permiten ir representando las distintas vistas de un sistema, cada diagrama tiene un objetivo bien definido.

UML significa Unified Modeling Language o Lenguaje Unificado de Modelado, y está basa en tres principios fundamentales:

Es un Lenguaje: está formado por elementos y reglas bien definidas, que poseen su propia sintaxis y semántica

Está Unificado: unifica los distintos criterios utilizados antes de su creación, es decir que toma las mejores propuestas de herramientas previas para presentar una propuesta sumamente abarcativa e integradora

Permite Modelar: está basado en la construcción de modelos que permite representar abstracciones de la realidad.

UML está estrechamente ligado con el paradigma de objetos, lo que permite construir sistemas de información de una forma mucho más intuitiva, integrada y sencilla con el proceso de desarrollo.

UML no es una metodología que presenta los pasos a seguir para realizar un desarrollo, sino que es un lenguaje gráfico de modelado.

Como nace UML

La historia cuenta que UML da sus primeros pasos con la unión de los “tres amigos”: Booch, Rumbaugh y Jacobson.

En los años 80 cada uno utilizaba un lenguaje propietario aunque como denominador común tenían como objetivo el desarrollo de sistemas, con previa modelización. A partir de los años 90 comienzan a intercambiar ideas para intentar unificar criterios.

Booch es el fundador de Rational Software Corp, y recluta en el año 1995 a Rumbaugh y Jacobson para comenzar a determinar una especificación genérica, sencilla y abarcativa.

Así es como las grandes empresas de tecnologías de información – entre ellas Rational - deciden formar un consorcio para la construcción de un Lenguaje Unificado de Modelado: UML. La primer versión de UML, la 1.0, sale a la luz en el año 1997, y a partir de 1998 un organización llamada OMG (Object Management Group) se encargó de generar nuevas revisiones. En el año 1998 UML se establece como Standard de facto en la industria del Software.

Dónde se utiliza

UML se utiliza dentro del marco de IT, aunque puede utilizarse en proyectos que no son de tecnología de la información, como ser el modelado de un motor o de una turbina.

En el campo de IT, se utiliza tanto para sistemas monolíticos como para sistemas distribuidos, abarca desde proyectos pequeños hasta grandes proyectos. Permite realizar la integración del software, donde representa el correcto enlace de los roles para lograr el éxito de la construcción del sistema. En proyectos de software, es utilizado desde la gestación hasta la instalación y el testing.

Si bien para utilizar UML es posible realizar los distintos diagramas con papel y lápiz, es conveniente contar con alguna herramienta del tipo IDE que facilite su construcción, corrección e integración entre diagramas.

INTRODUCCIÓN A LOS DIAGRAMAS DE UML

Introducción

UML está organizado en una serie de diagramas que tienen objetivos bien definidos, con una sintaxis y semántica determinada, que intentan representar / modelar distintas vistas de un sistema.

Diagrama de Clases

El Diagrama de Clases tiene como objetivo describir las clases del dominio y sus relaciones. Permite modelar la estructura del sistema desde un punto de vista estático, modelando las clases desde distintos enfoques de acuerdo a la etapa del proyecto.

Está compuesto por clases, relaciones entre clases y opcionalmente los paquetes que agrupan a las clases.

Diagrama de Objetos

El Diagrama de Objetos tiene como objetivo describir los objetos del dominio y sus relaciones. Permite representar al sistema en un momento determinado del tiempo, es proporcional a obtener una fotografía o snapshot del sistema en un momento determinado.

Está compuesto por objetos y relaciones de enlace. También es posible pensarlo como una instancia de un Diagrama de Clases.

Diagrama de Casos de Uso

El Diagrama de Casos de Uso tiene como objetivo describir las acciones del sistema desde el punto de vista del usuario. Representa las formas que tiene un usuario de utilizar un sistema, y se puede utilizar como un "contrato" entre cliente y proveedor de software para determinar la funcionalidad del sistema, es decir los requisitos funcionales.

Está compuesto por actores (agentes externos al sistemas, pueden ser usuarios u otros sistemas), casos de uso y distintos tipos de relaciones. Es posible construir diagramas con diferentes niveles de detalle.

Diagrama de Estados

El Diagrama de Estados tiene como objetivo describir los estados por los cuales puede pasar un objeto durante su ciclo de vida. Permite modelar tanto estas simples como compuestos y concurrentes.

Está compuesto por estados, pseudo-estados y transiciones entre estados.

Diagrama de Actividades

El Diagrama de Actividades tiene como objetivo describir las acciones que ocurren dentro de un proceso. Se utiliza principalmente para modelar flujo de trabajo o workflow, con lo cual visualiza las acciones de manera ordenada.

Está compuesto por acciones simples y concurrentes, y transiciones entre las acciones.

Diagrama de Comunicación

El Diagrama de Comunicación tiene como objetivo describir cómo colaboran o se comunican los distintos objetos entre sí para conseguir un objetivo. Se lo suele llamar también Diagrama de Colaboración. Es posible verlo como una extensión del Diagrama de Objetos, ya que es muy parecido pero tiene como valor agregado los mensajes que se envían entre los objetos.

Está compuesto por objetos, relaciones de enlace y relaciones del tipo llamadas, representando qué objeto se comunica con que otro.

Es semánticamente equivalente al Diagrama de Secuencia.

Diagrama de Secuencia

El Diagrama de Secuencia tiene como objetivo describir cómo colaboran los distintos objetos entre sí para conseguir un objetivo a lo largo del tiempo. Está directamente relacionado con el Diagrama de Comunicación ya que el objetivo es el mismo, pero tiene la particularidad de estar obligatoriamente ordenado en el tiempo.

Está compuesto por objetos y relaciones del tipo llamadas, representando qué objeto se comunica con que otro.

Es semánticamente equivalente al Diagrama de Comunicación.

Diagrama de Componentes

El Diagrama de Componentes tiene como objetivo describir la relación que existe entre los distintos componentes del sistema. Está directamente vinculado con el diseño del sistema, permitiendo modelar las relaciones e interfaces que existen entre los componentes. Está orientado a la implementación del sistema.

Está compuesto por componentes, interfaces y sus relaciones.

Diagrama de Despliegue

El Diagrama de Despliegue tiene como objetivo describir la arquitectura de un sistema. Es posible representar la arquitectura desde el punto de vista lógico, basándose en la organización del software, o desde una punto de vista físico, representando directamente cada unidad de hardware.

Está compuesto por nodos, componentes y sus relaciones.

CLASIFICACIÓN DE DIAGRAMAS

Categorías

Es posible clasificar a los diagramas según si tienen alguna relación con el tiempo o no. Existen dos posibles categorías, los diagramas estáticos y los diagramas dinámicos.

Diagramas Estáticos

Los Diagramas Estáticos son aquellos que no tienen ninguna relación con el tiempo. Generalmente representan a estructuras o a posibles acciones pero sin una relación directa con el tiempo.

Estos diagramas son:

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Casos de Uso
- Diagrama de Componentes
- Diagrama de Despliegue

Diagramas Dinámicos

Los Diagramas Dinámicos son aquellos que tienen relación con el tiempo. Están directamente vinculados con acciones que ocurren bajo cierta secuencia, es decir primero una acción, luego otra, y así sucesivamente.

Estos diagramas son:

- Diagrama de Actividades
- Diagramas de Interacción (es una subcategoría, que incluye a los Diagramas de Secuencia y Comunicación)
- Diagrama de Estado

Diagramas Estructurales

Los Diagramas Estructurales son aquellos que reflejan relaciones estáticas de una estructura.

Estos diagramas son:

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Despliegue

Diagramas de Comportamiento

Los Diagramas de Comportamiento son aquellos que reflejan características de comportamiento del sistema o modelan procesos de negocios.

Estos diagramas son:

- Diagrama de Casos de Uso
- Diagrama de Comunicación
- Diagrama de Secuencia
- Diagrama de Actividades
- Diagrama de Estados