

Programación Orientada a Objetos

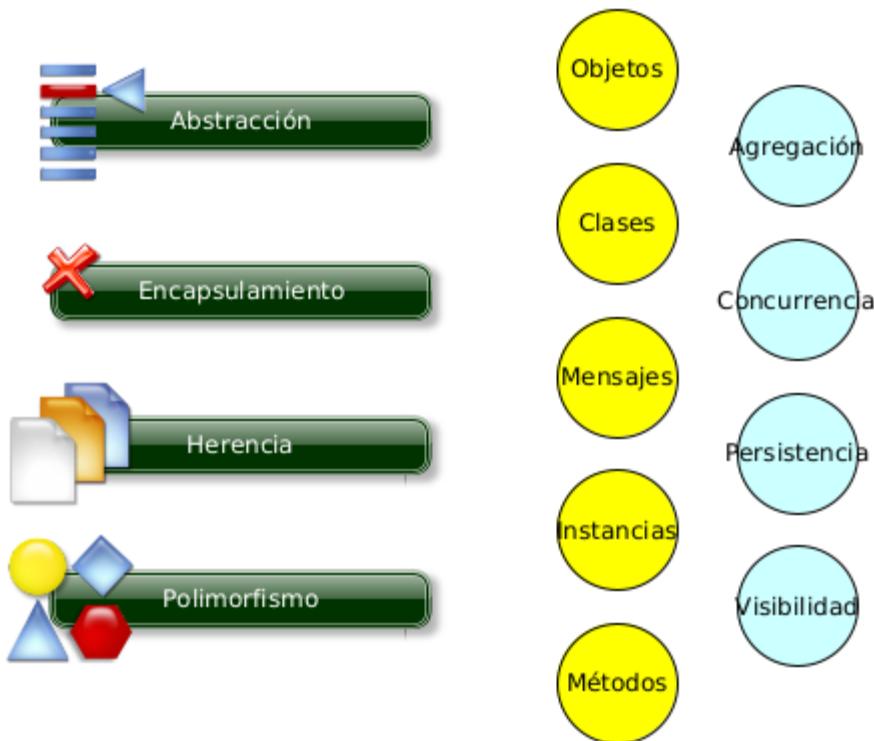
Programación Orientada a Objetos

Contenidos

- › Conceptos orientados a objetos
- › Reutilización del código y relaciones entre clases
- › Principios de diseño
- › Patrones de diseño
- › Introducción a UML

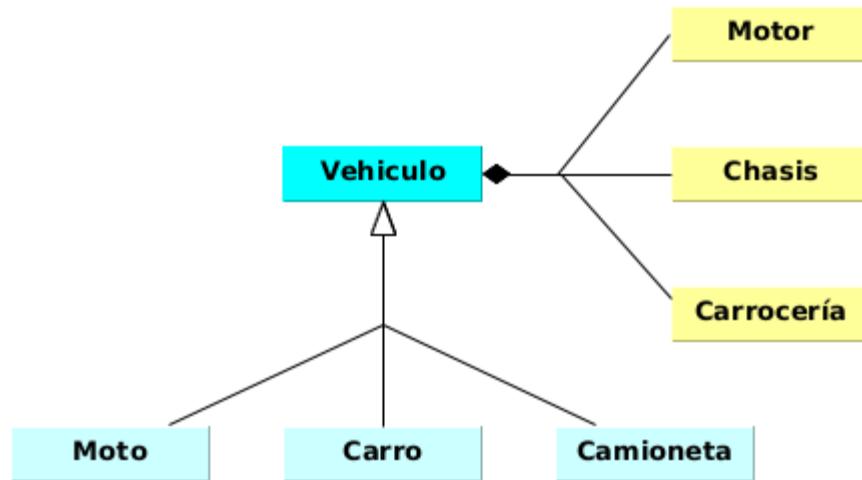
Conceptos orientados a objetos

Un sistema se califica como Orientado-a-Objetos cuando reúne las características de: **abstracción, encapsulación, herencia y polimorfismo**; y los conceptos básicos que las forman: **objetos, mensajes, clases, instancias y métodos**.

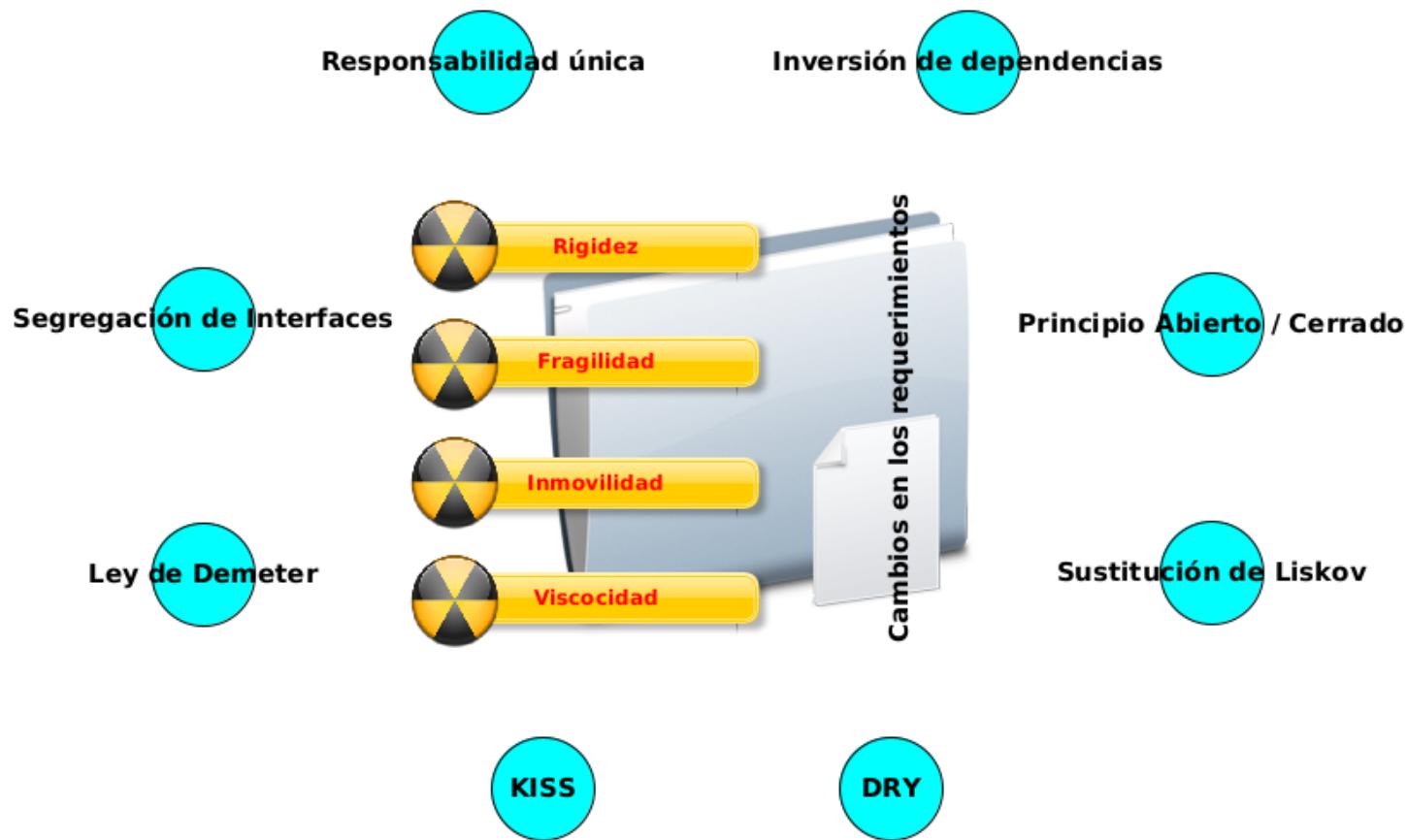


Reutilización del código y relaciones entre clases

Reutilización de código no se refiere a copiar (ctrl+c) y pegar (ctrl+v)



Principios de diseño



Patrones de Diseño

Un patrón de diseño contiene la solución probada a un problema en un determinado contexto.

Tal es la importancia que cobran los patrones de diseño, como técnica de reutilización de software y resolución rápida de problemas habituales, que este tipo de conocimiento es recopilado y publicado en diferentes manuales y catálogos.

Patrones de Diseño

Tipos de patrones

Patrones de creación

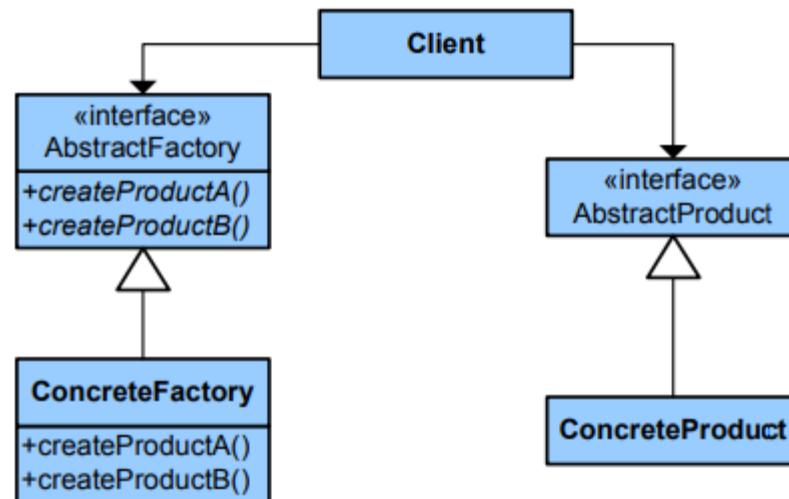
Los patrones de creación o creacionales abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas.

Patrones de Diseño

Tipos de patrones

Patrones de creación

- **Abstract Factory:** Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

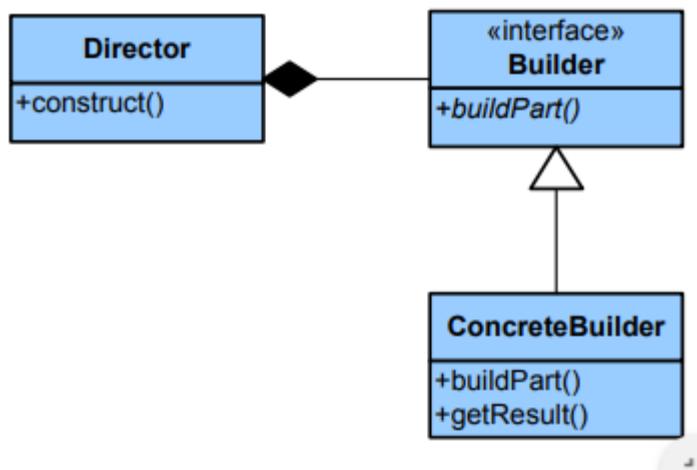


Patrones de Diseño

Tipos de patrones

Patrones de creación

- **Builder:** Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

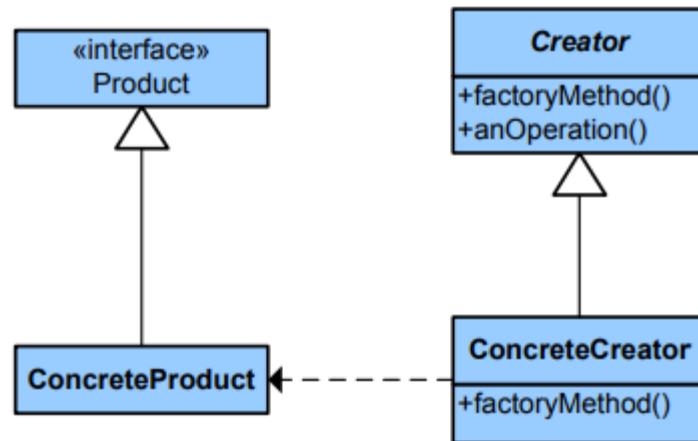


Patrones de Diseño

Tipos de patrones

Patrones de creación

- **Factory Method:** Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.

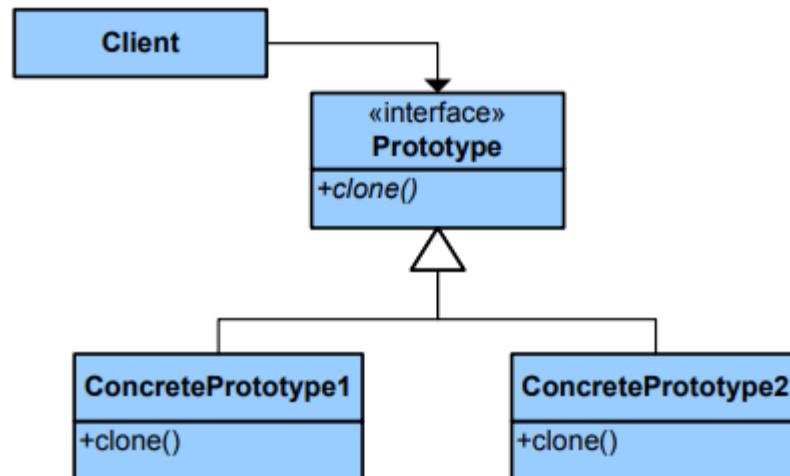


Patrones de Diseño

Tipos de patrones

Patrones de creación

- **Prototype:** Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.

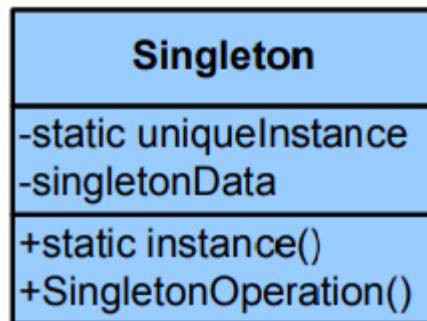


Patrones de Diseño

Tipos de patrones

Patrones de creación

- **Singleton:** Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.



Patrones de Diseño

Tipos de patrones

Patrones estructurales

Tratan de conseguir que cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y, éstas están determinadas por las interfaces que soportan los objetos.

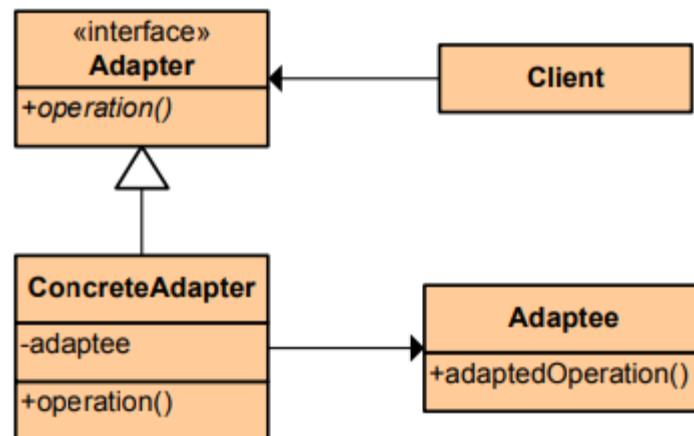
Estudian como se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos.

Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Adapter:** Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

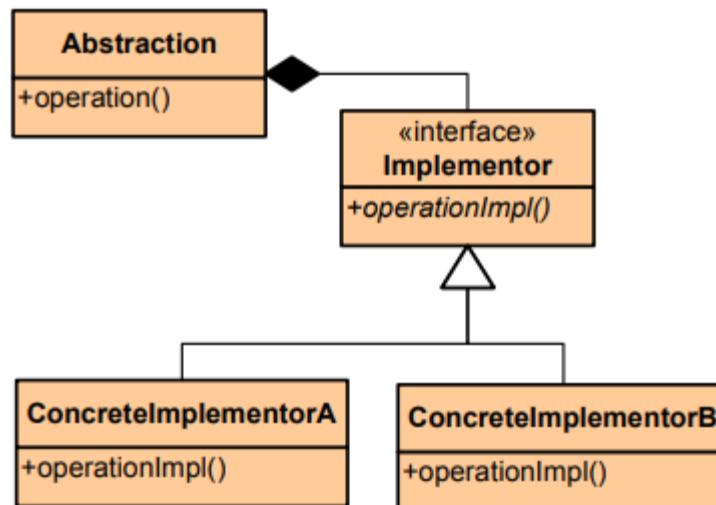


Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Bridge:** Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

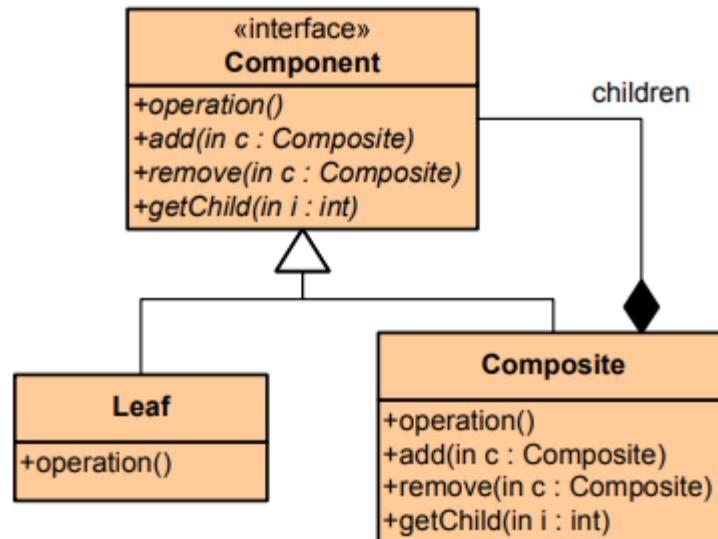


Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Composite:** Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

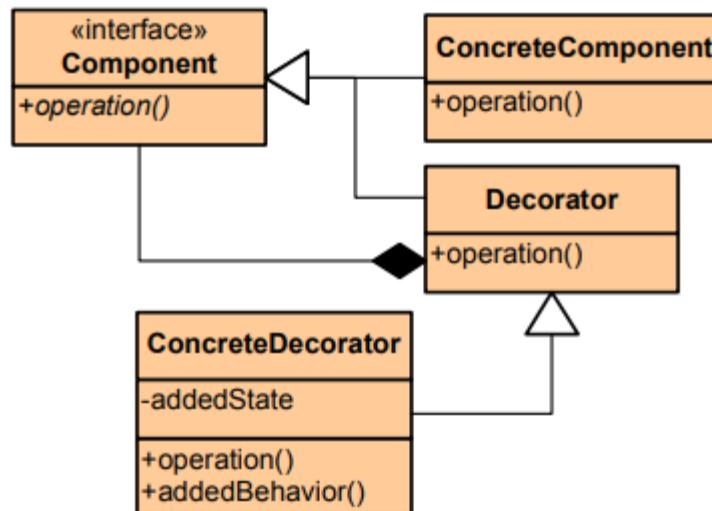


Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Decorator:** Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

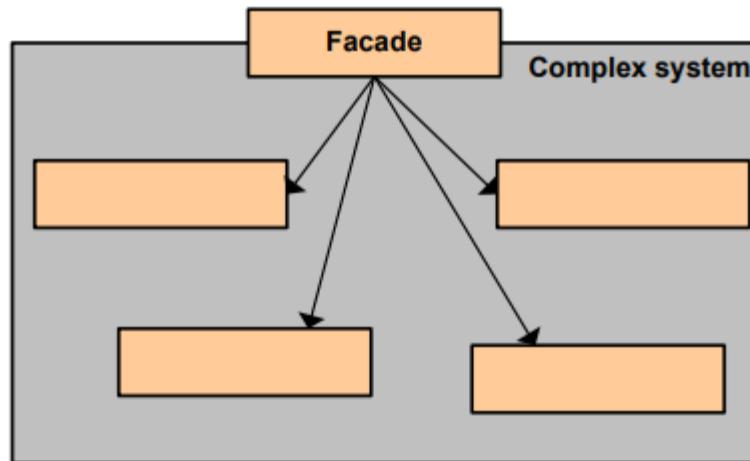


Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Facade:** Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

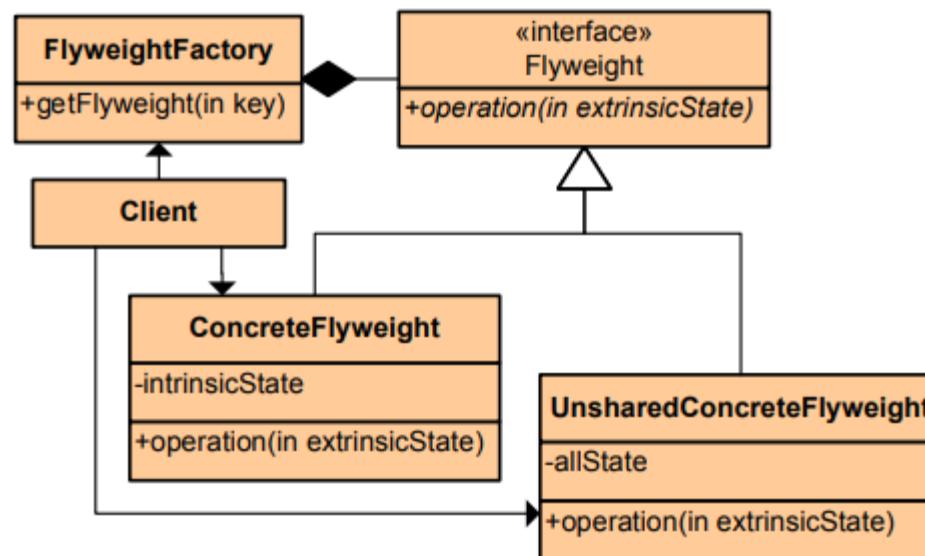


Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Flyweight:** Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.

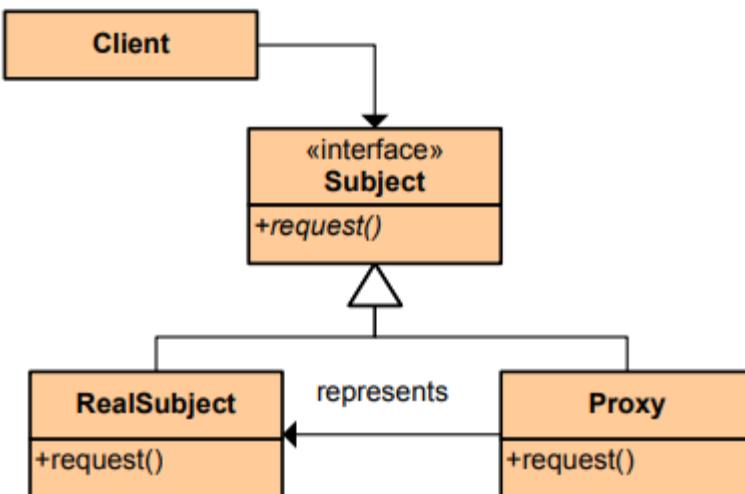


Patrones de Diseño

Tipos de patrones

Patrones estructurales

- **Proxy:** Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.



Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

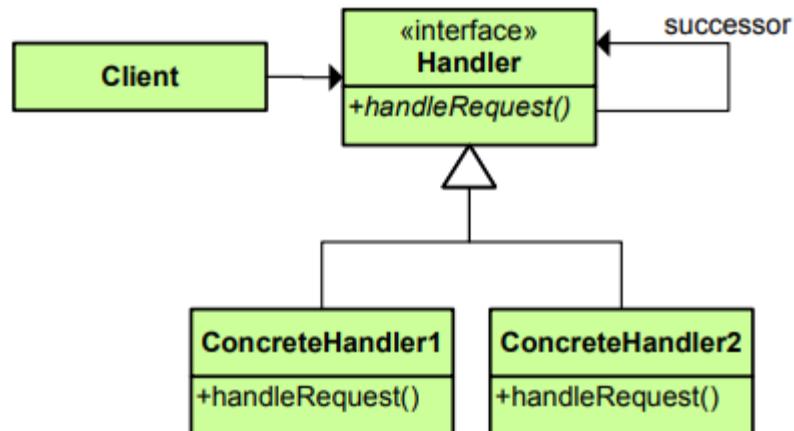
Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal.

Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Chain of Responsibility:** Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.

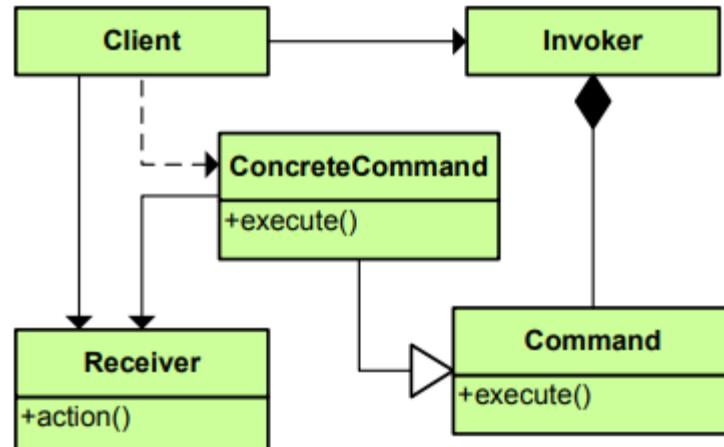


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Command:** Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.

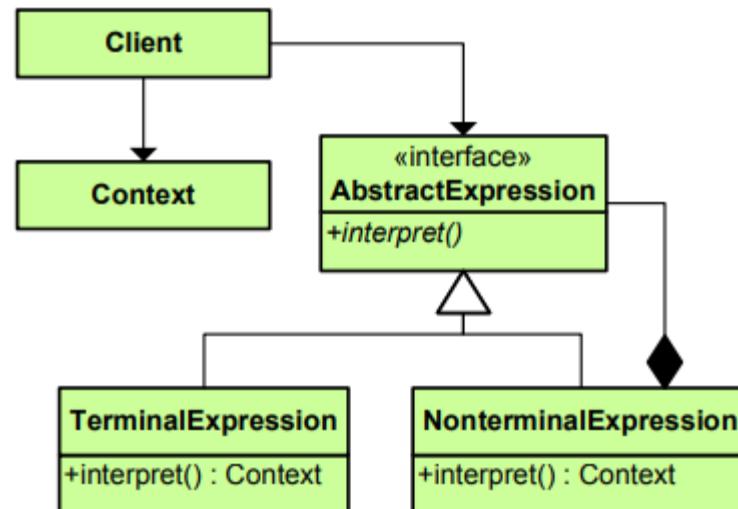


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Interpretar:** Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.

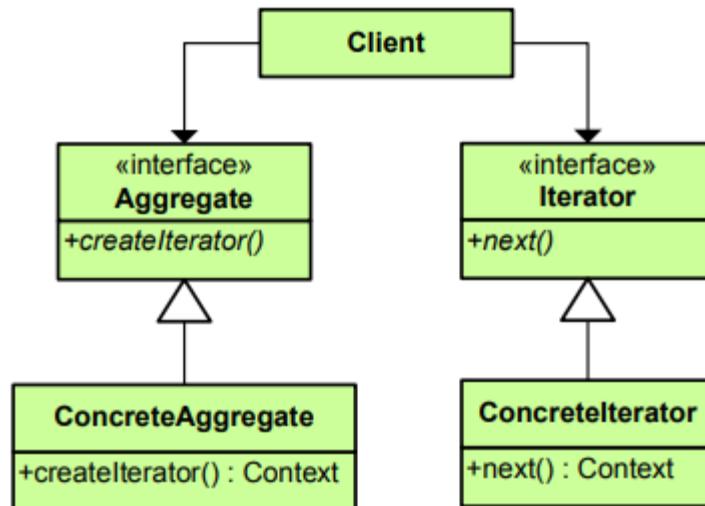


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Iterator:** Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

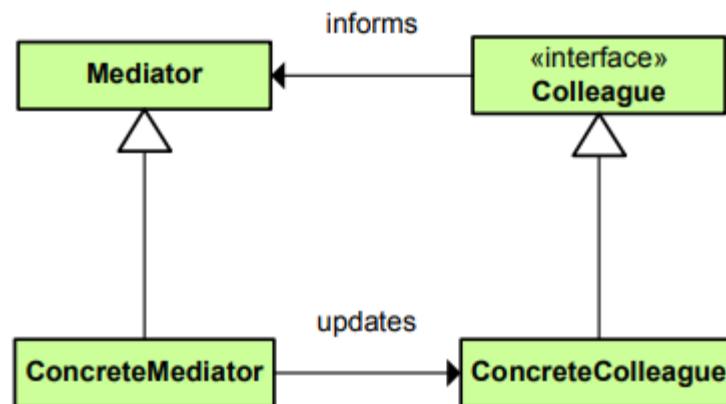


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Mediator:** Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

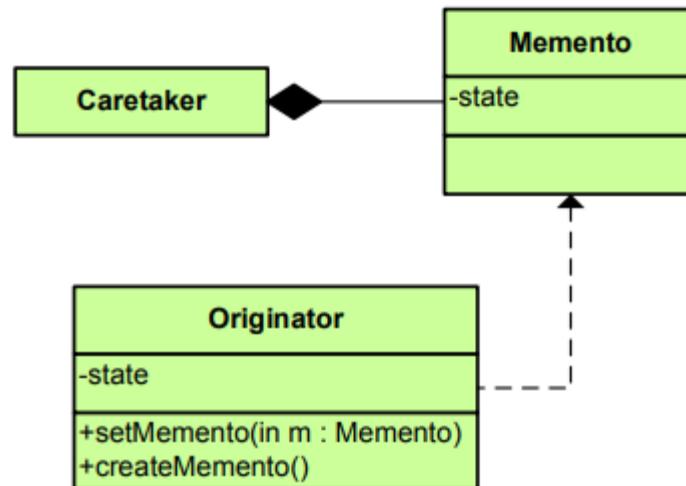


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Memento:** Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.

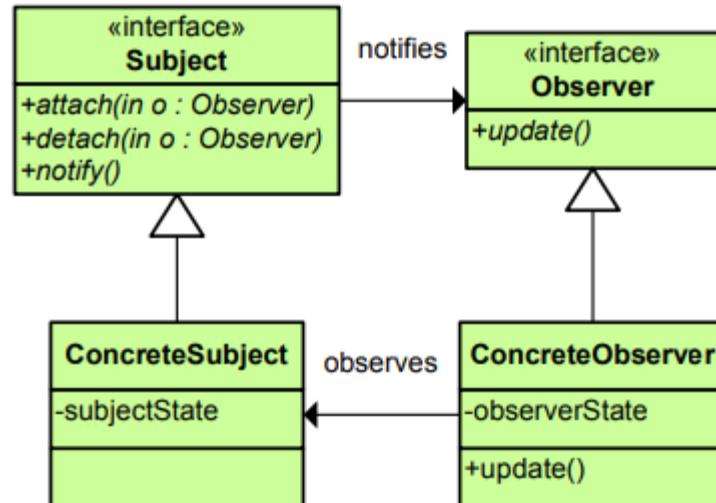


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Observer:** Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

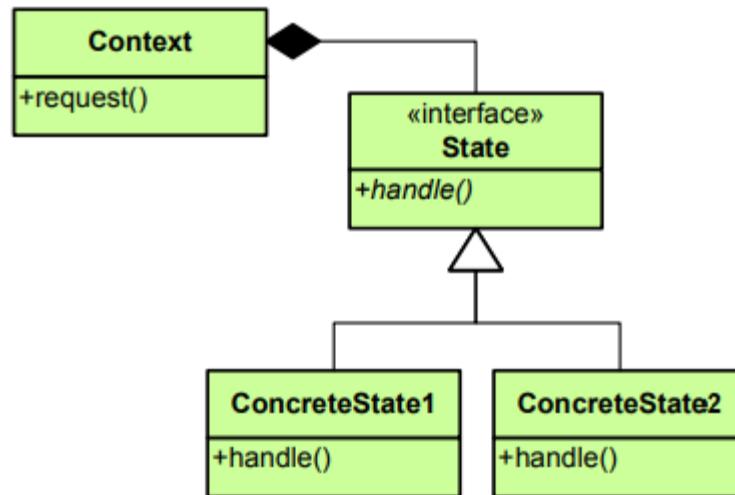


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **State:** Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.

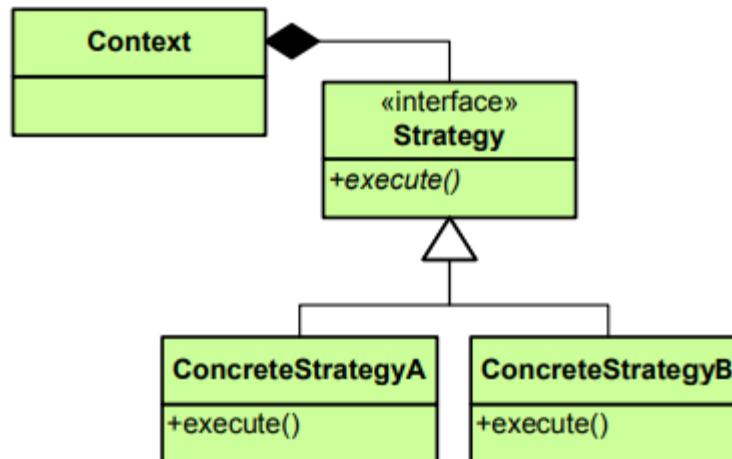


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Strategy:** Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

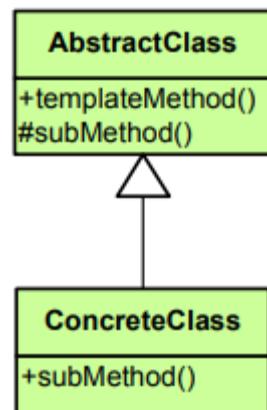


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Template Method:** Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

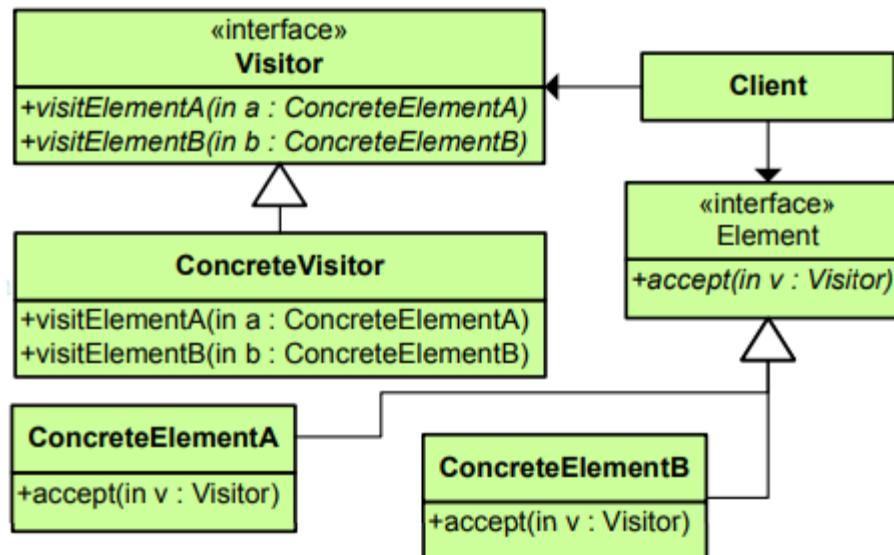


Patrones de Diseño

Tipos de patrones

Patrones de comportamiento

- **Visitor:** Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.



Introducción a UML

El Lenguaje de Modelado Unificado (UML) es un lenguaje de diseño para modelar software mediante diferentes tipos de diagramas.

Es un lenguaje de modelado visual que permite mediante representaciones, para los proyectos de software:

- **Visualizar:** permite mediante representaciones gráficas construir modelos que representan el dominio del problema y la solución.
- **Especificar:** permite especificar mediante el modelo estático la estructura y mediante el modelo dinámico el comportamiento.
- **Construir:** permite mediante el uso de herramientas CASE la generación de código en base a los modelos planteados.
- **Documentar:** permite gestionar desde los requerimientos, todos los conjuntos de diagramas, códigos, manuales, etc.

Introducción a UML

Los principales diagramas son:

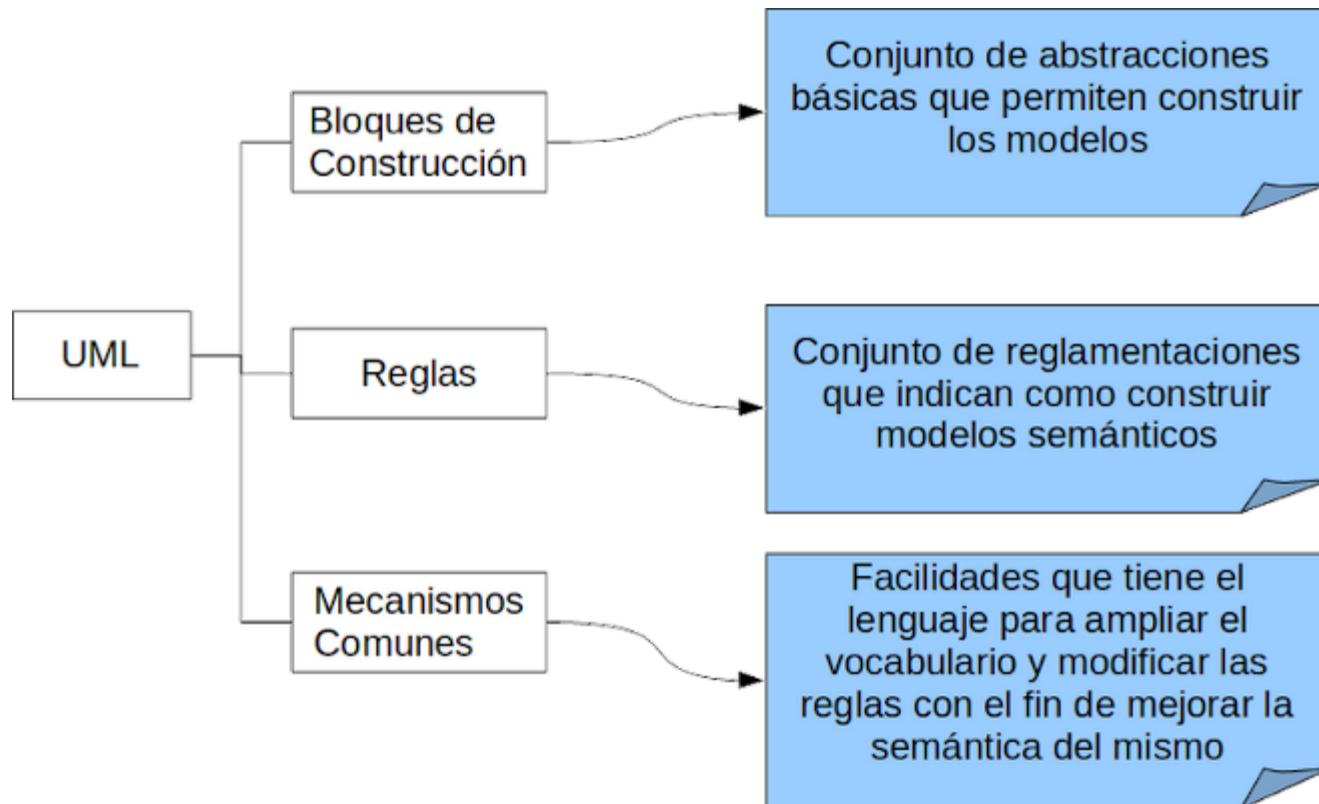
- **Diagrama de clases:** proporciona elementos para modelar las clases (con sus atributos y métodos) y las relaciones y asociaciones entre ellas.
- **Diagrama de implementación:** permite mostrar la distribución física de un sistema software en las plataformas de hardware y entornos de ejecución.
- **Diagramas de casos de uso:** representan la interacción del usuario con las diferentes funciones del sistema.
- **Diagramas de secuencia:** muestra la comunicación dinámica entre los diferentes objetos para llevar a cabo una tarea, describiendo el orden de ejecución.

Introducción a UML

- **Diagramas de comunicación:** representan como los diagramas de secuencia la comunicación entre los objetos para realizar una tarea, enfatizando en las relaciones entre las clases y objetos.
- **Diagramas de actividad:** muestran el comportamiento dinámico del sistema mediante el flujo de control de las acciones que debe realizar.
- **Diagramas de estado:** modelan los estados por los que pasa un objeto y las acciones que desencadenan la transición de estados.

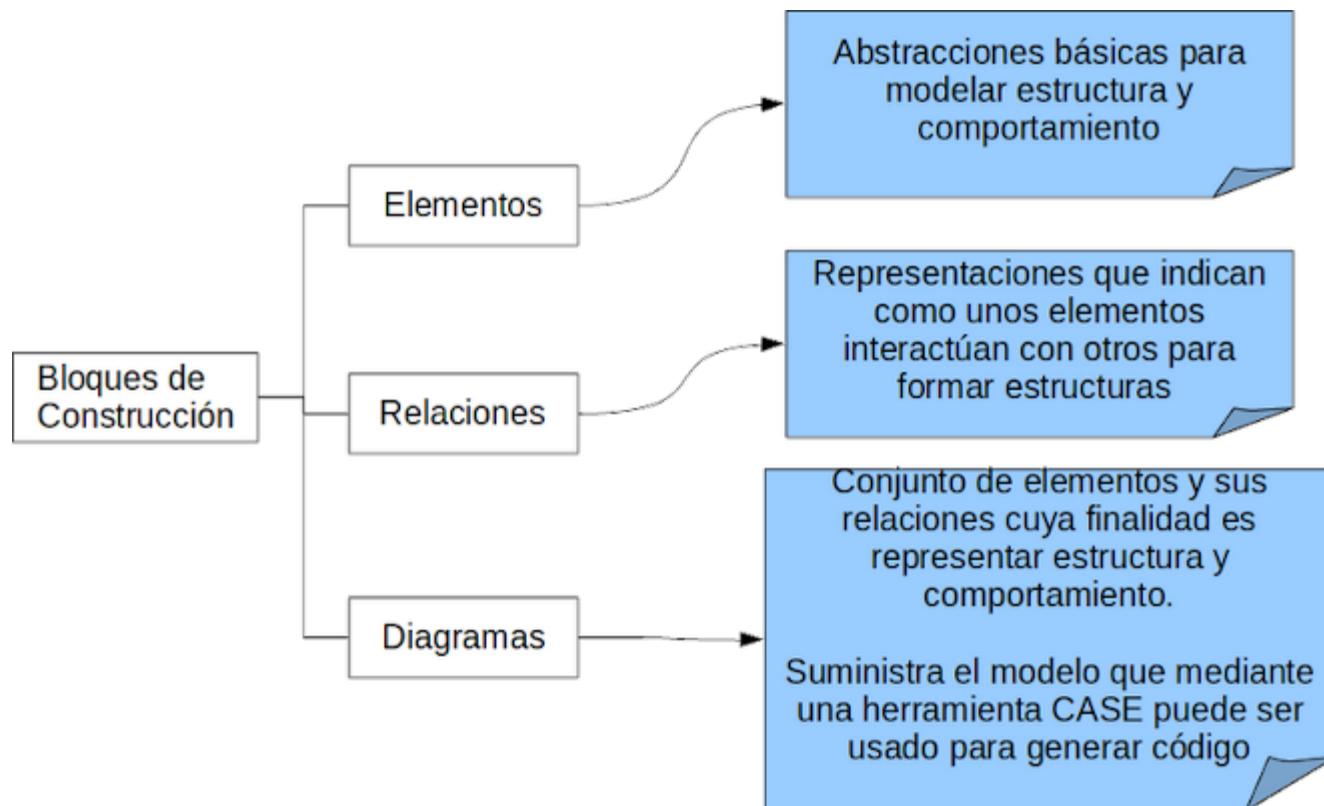
Introducción a UML

Modelo conceptual:



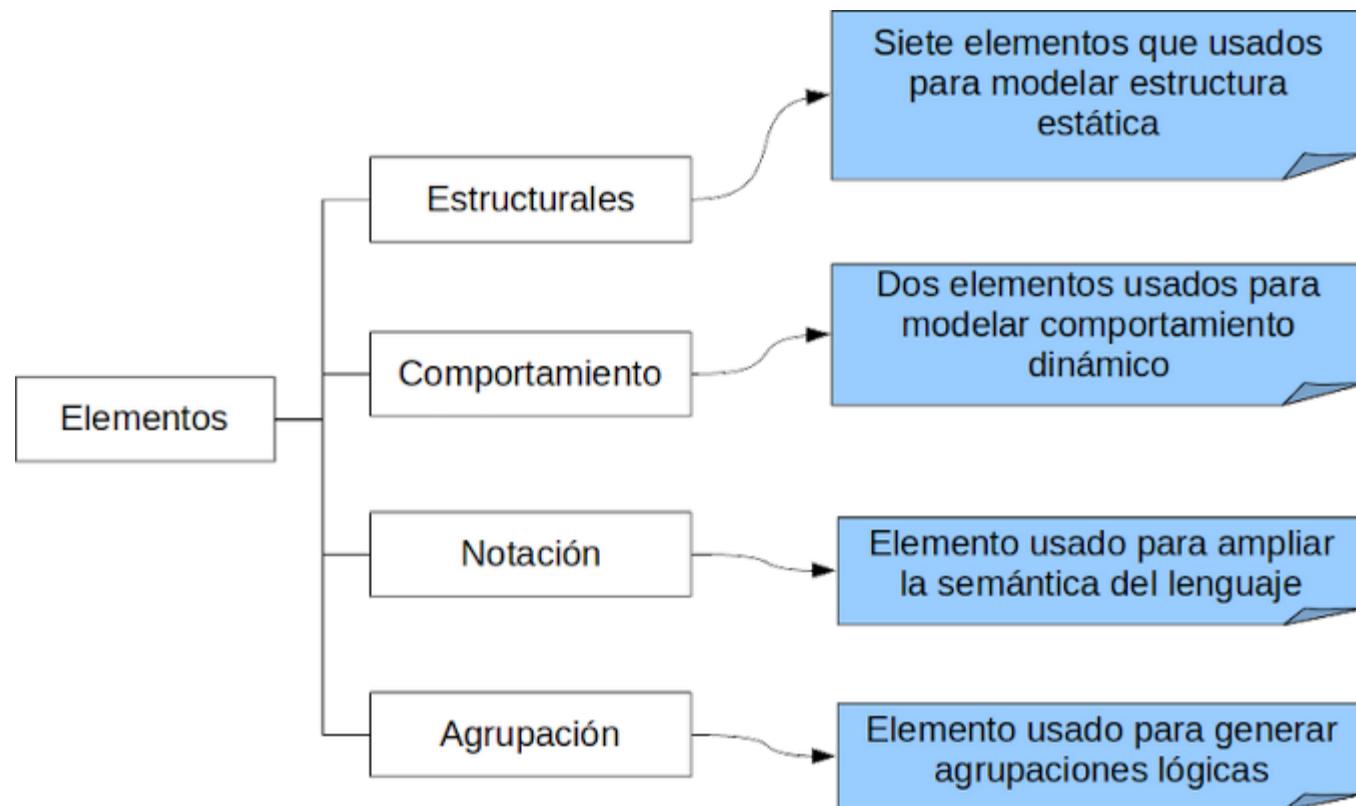
Introducción a UML

Modelo conceptual:



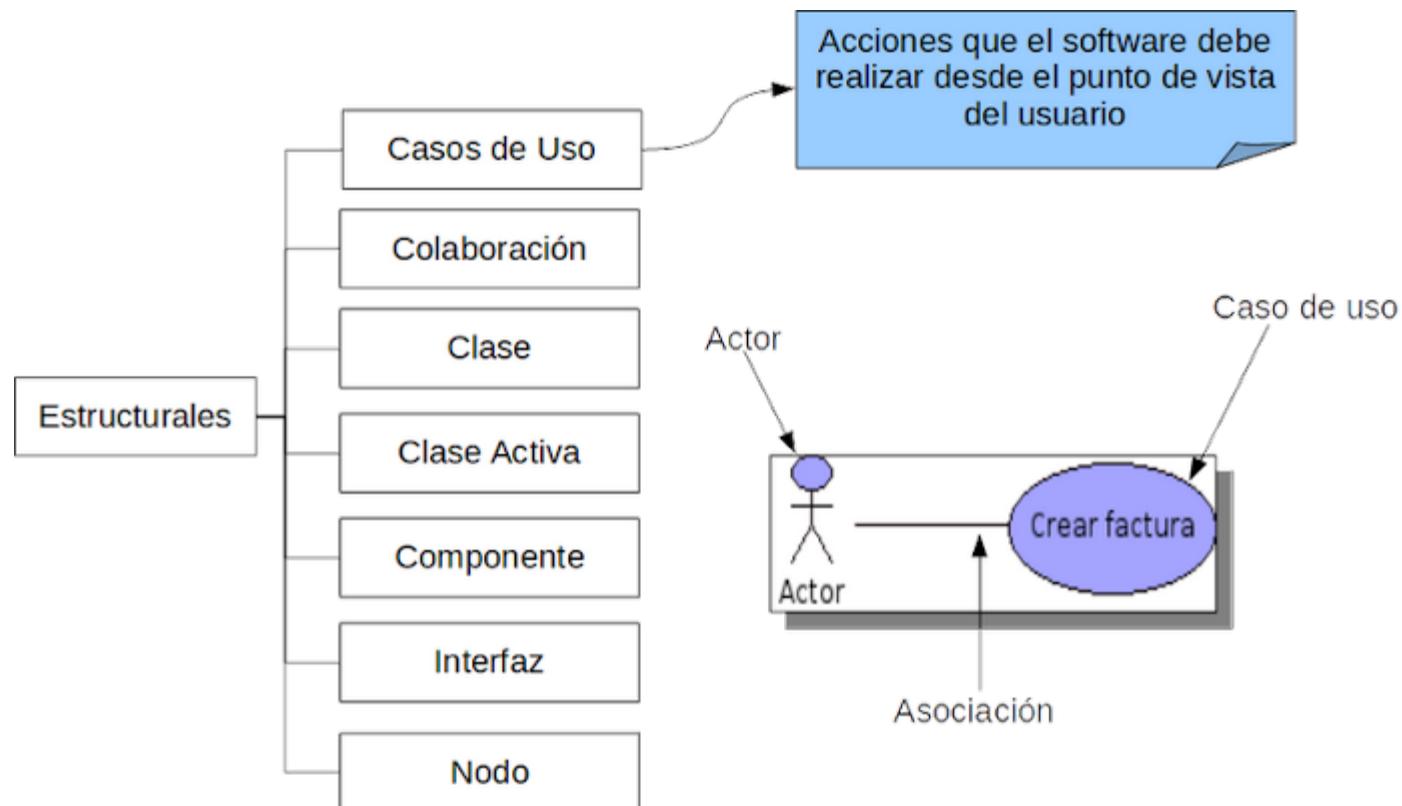
Introducción a UML

Modelo conceptual:



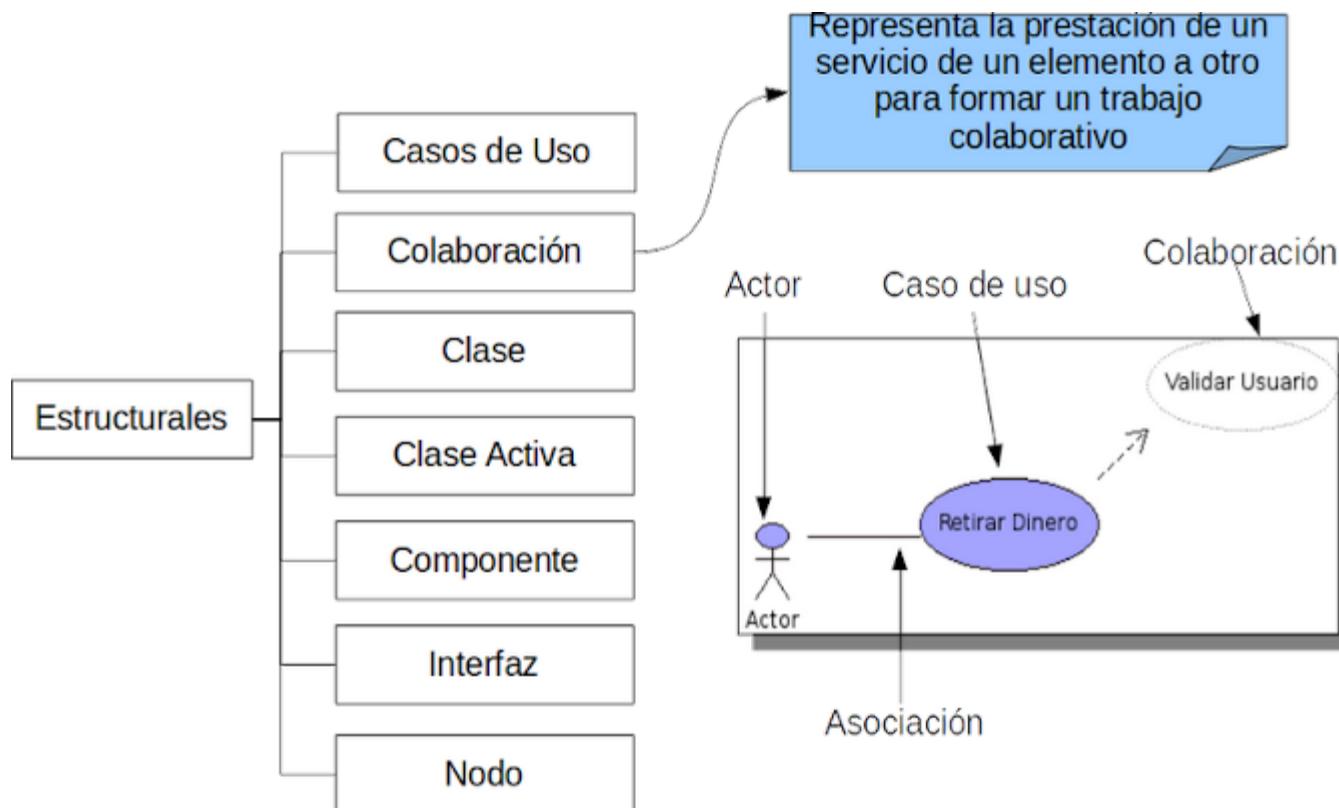
Introducción a UML

Modelo conceptual:



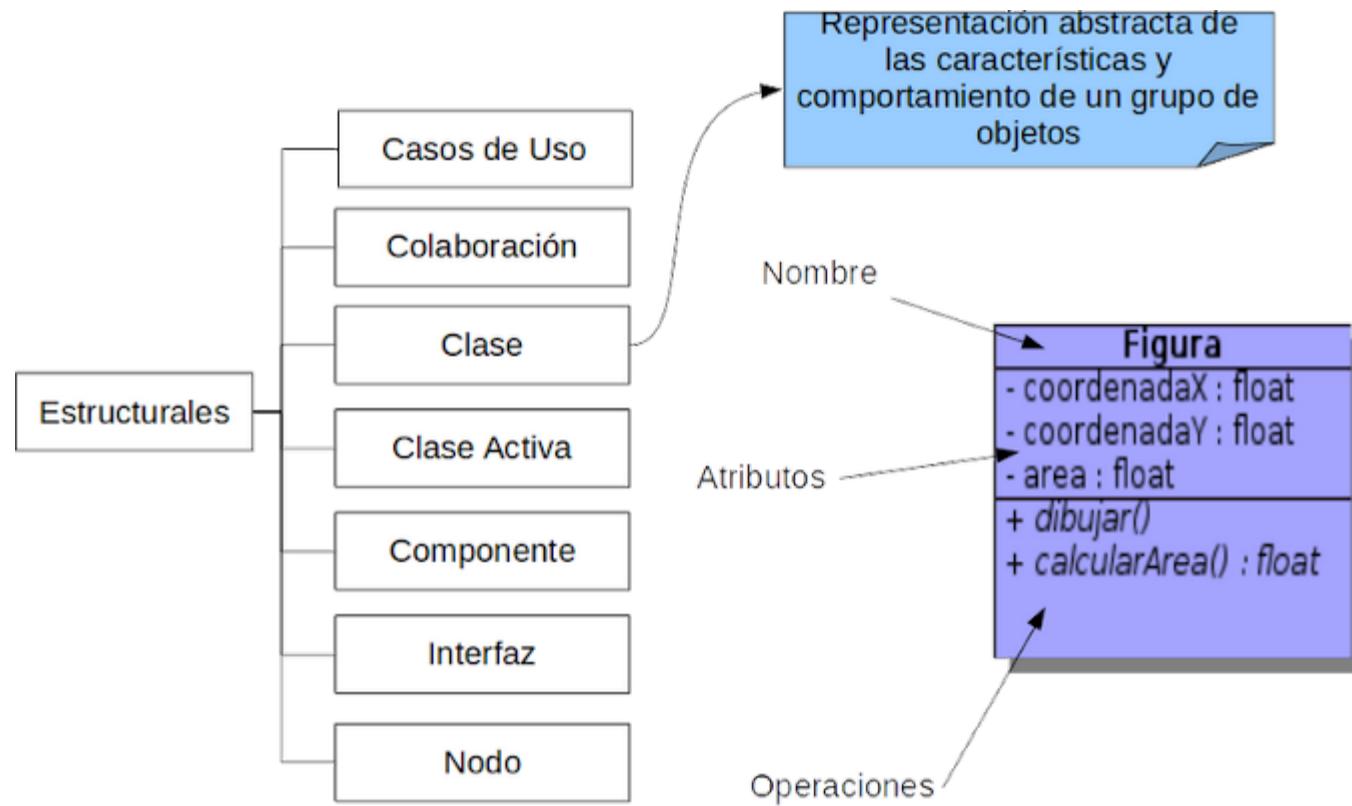
Introducción a UML

Modelo conceptual:



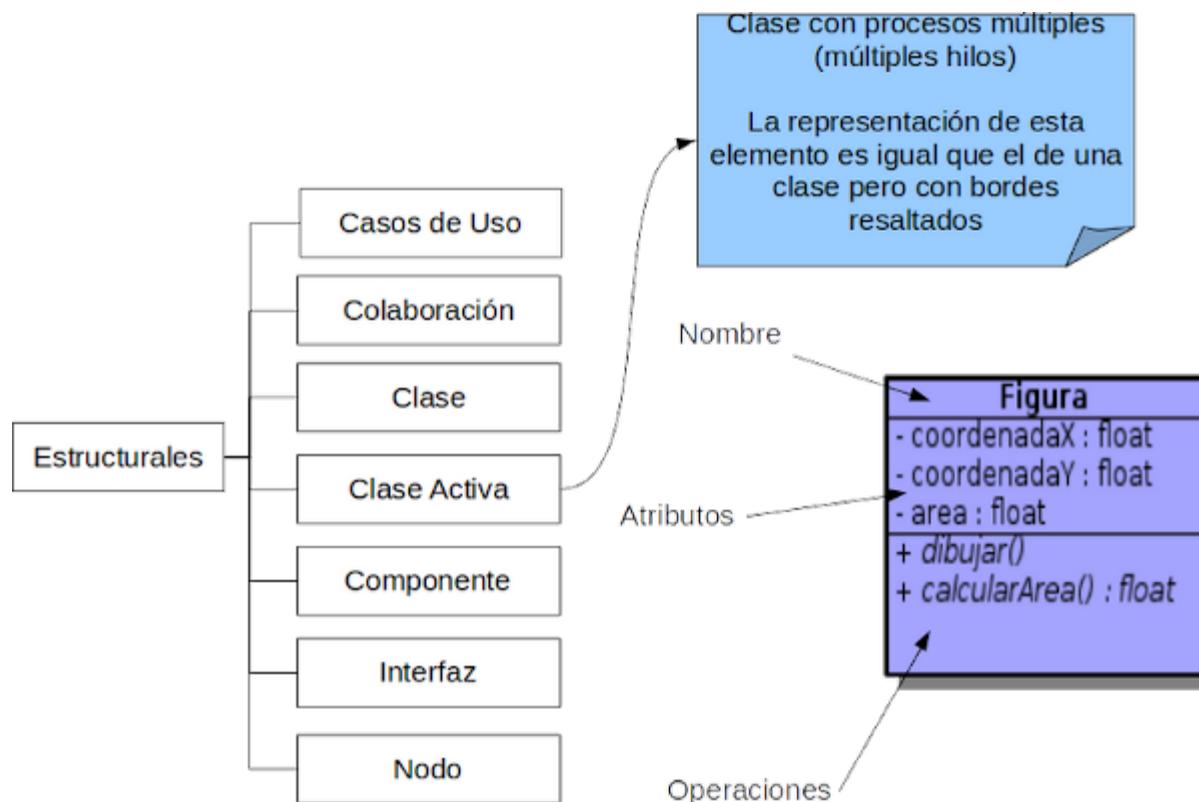
Introducción a UML

Modelo conceptual:



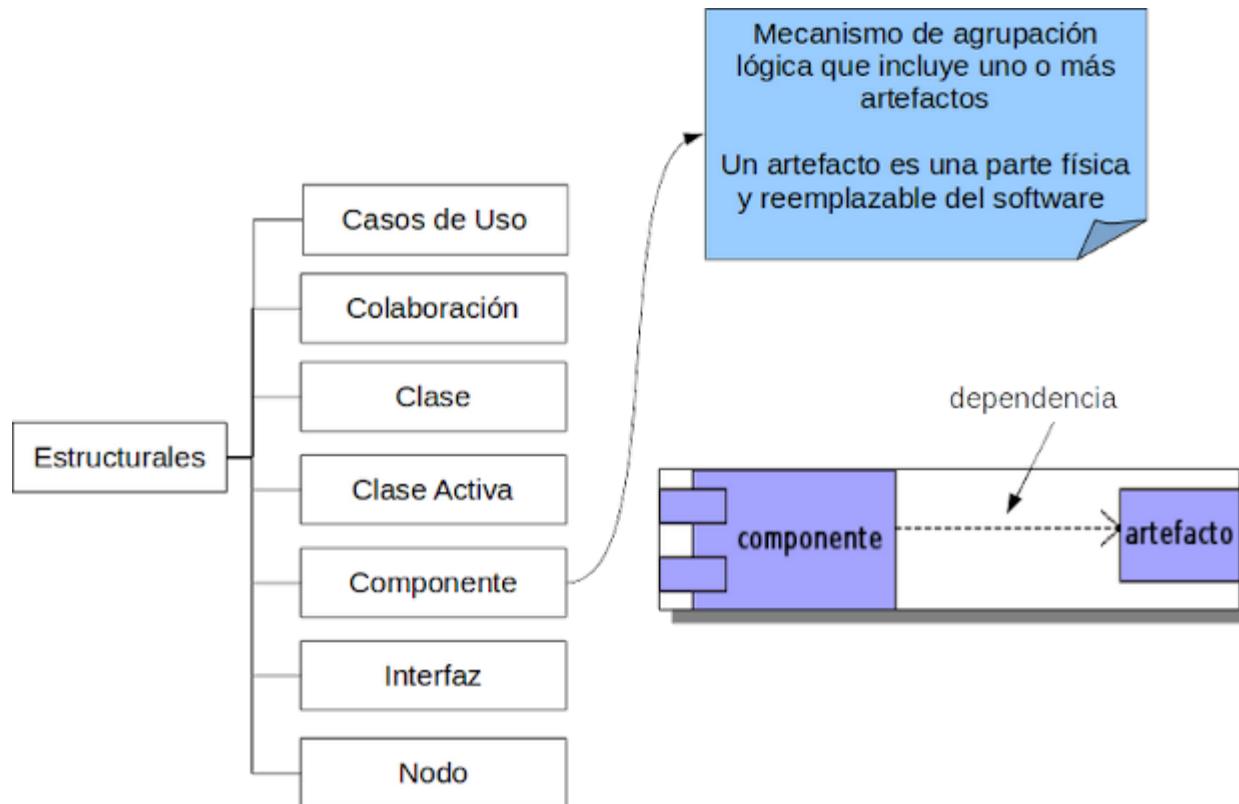
Introducción a UML

Modelo conceptual:



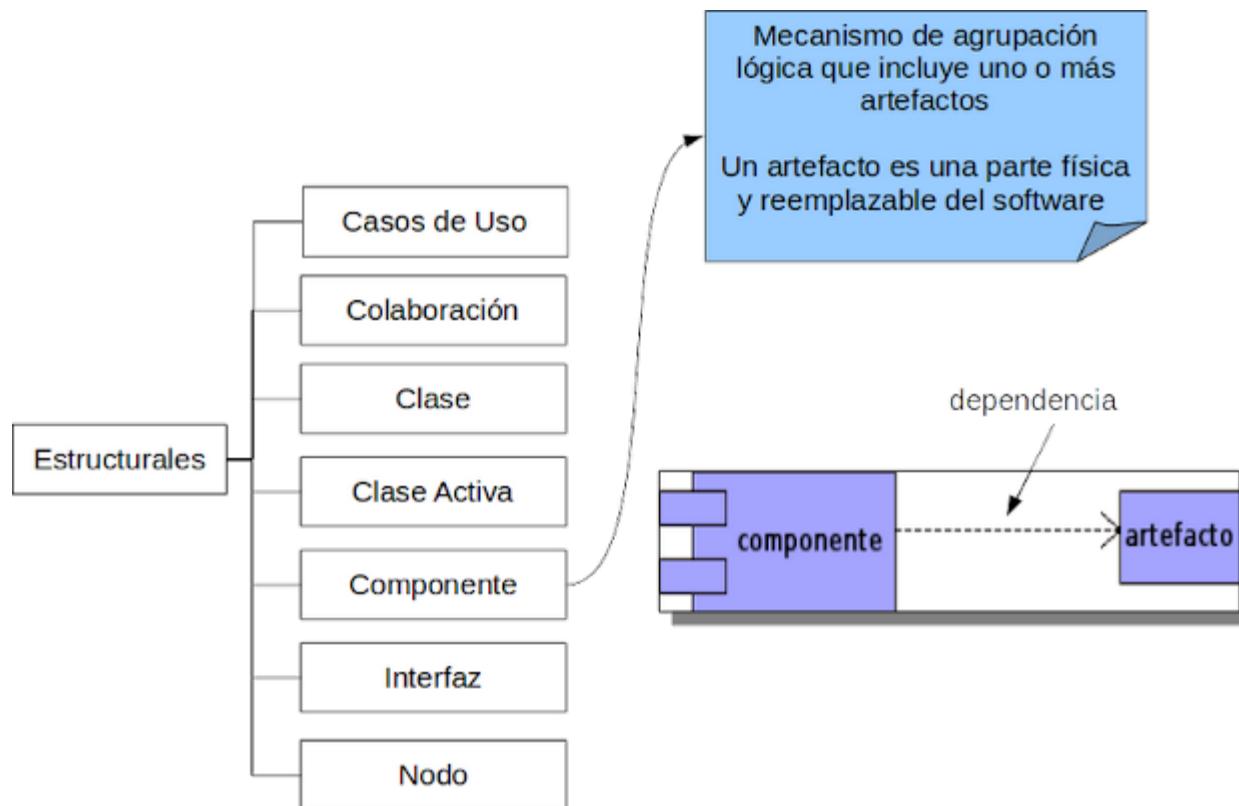
Introducción a UML

Modelo conceptual:



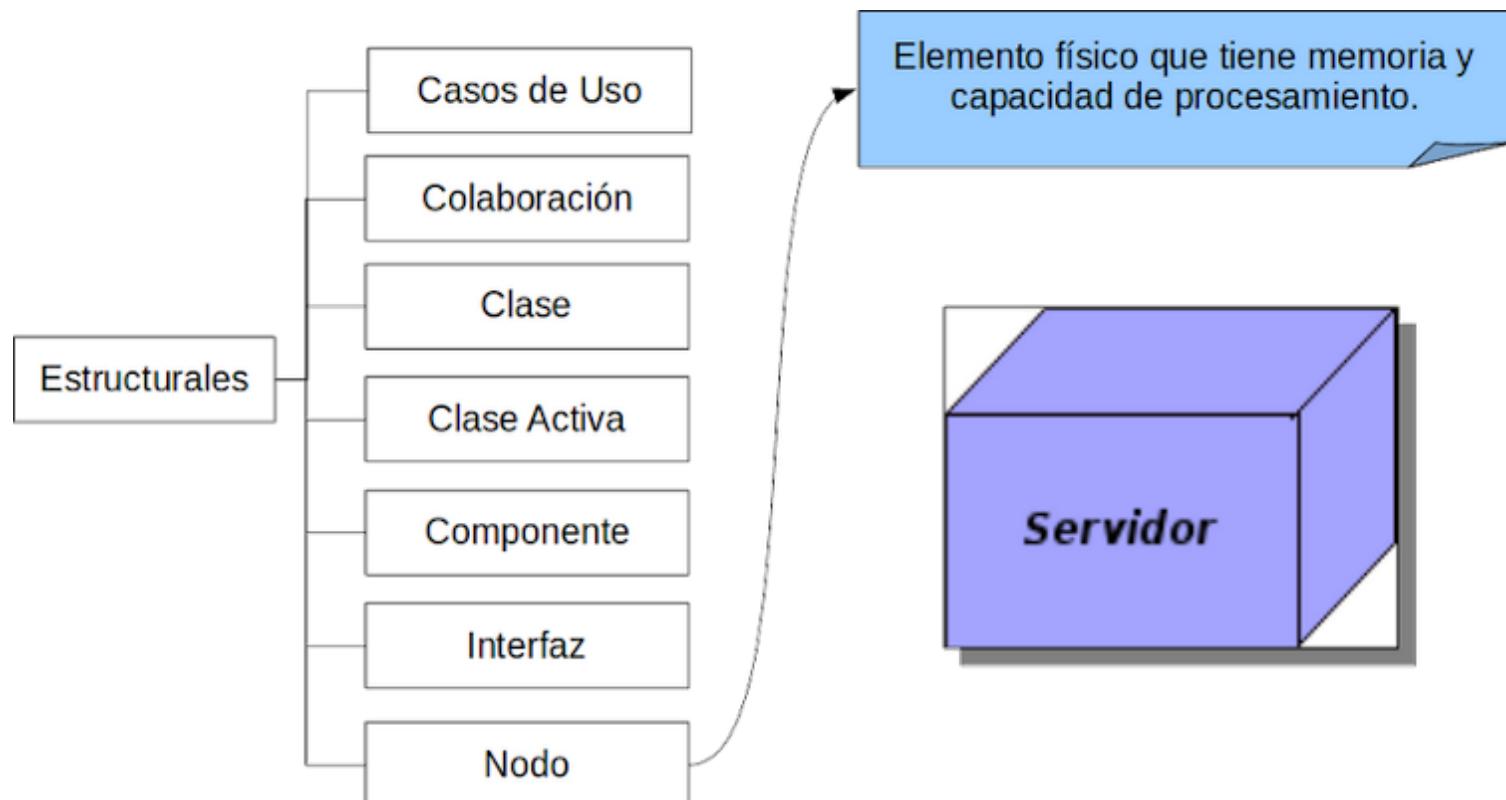
Introducción a UML

Modelo conceptual:



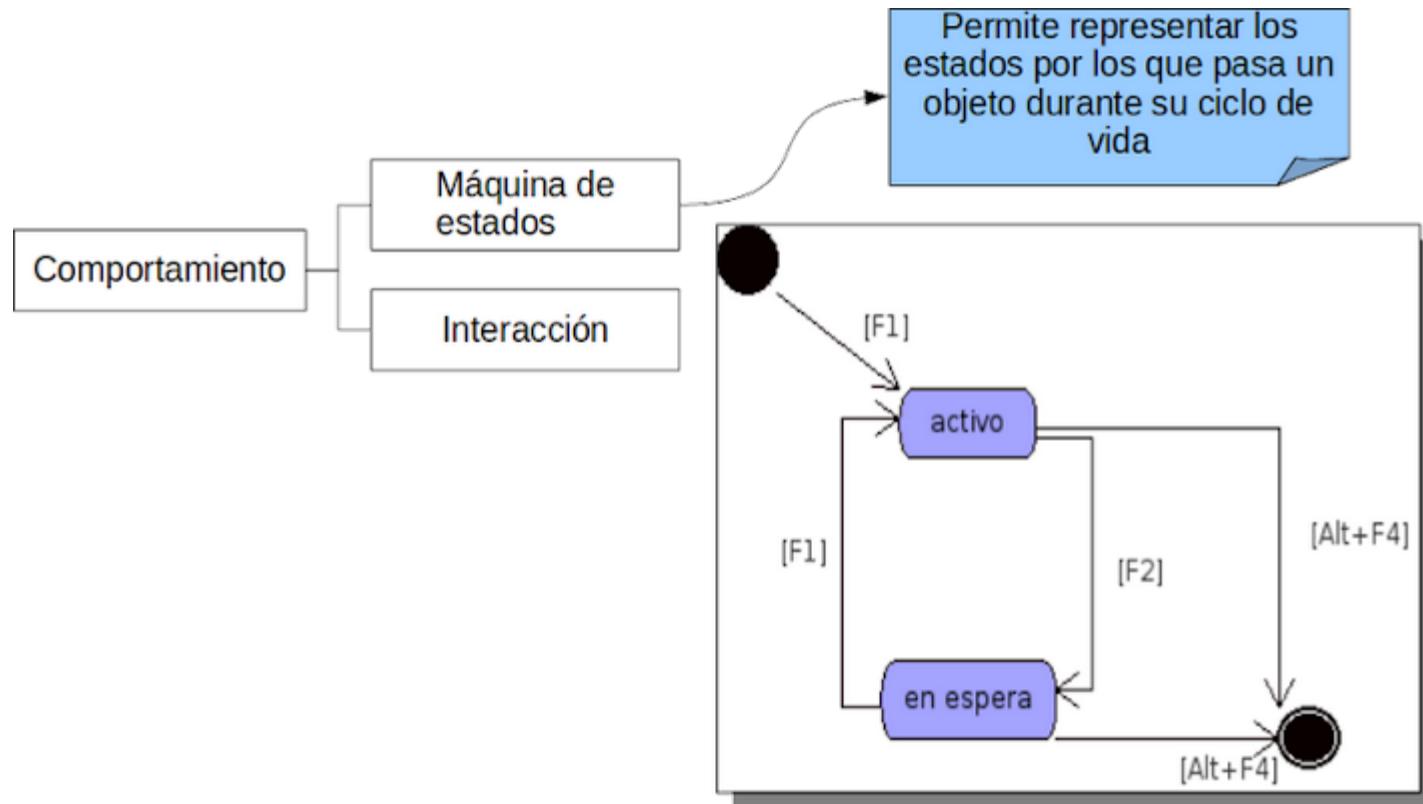
Introducción a UML

Modelo conceptual:



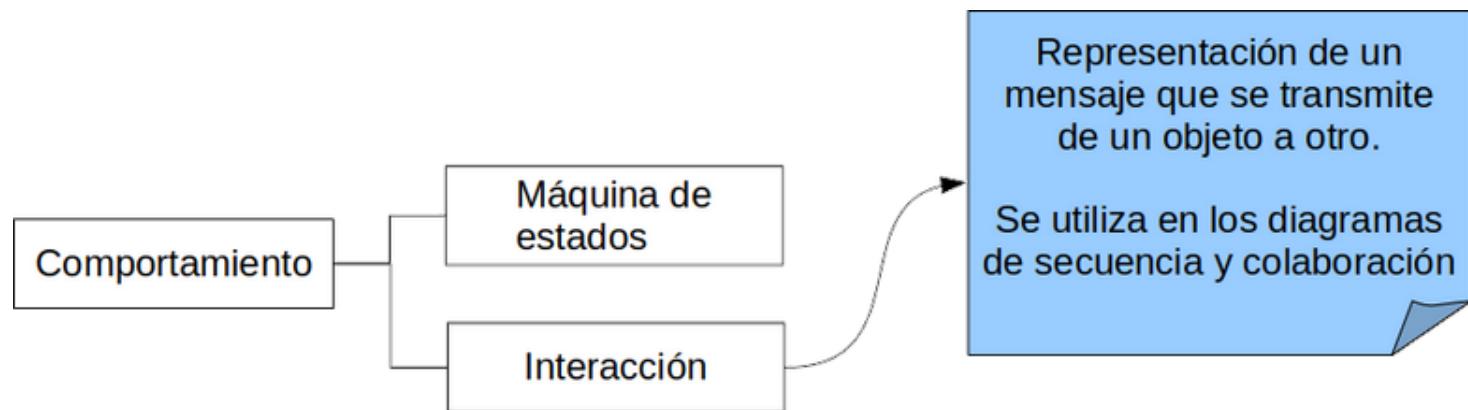
Introducción a UML

Modelo conceptual:



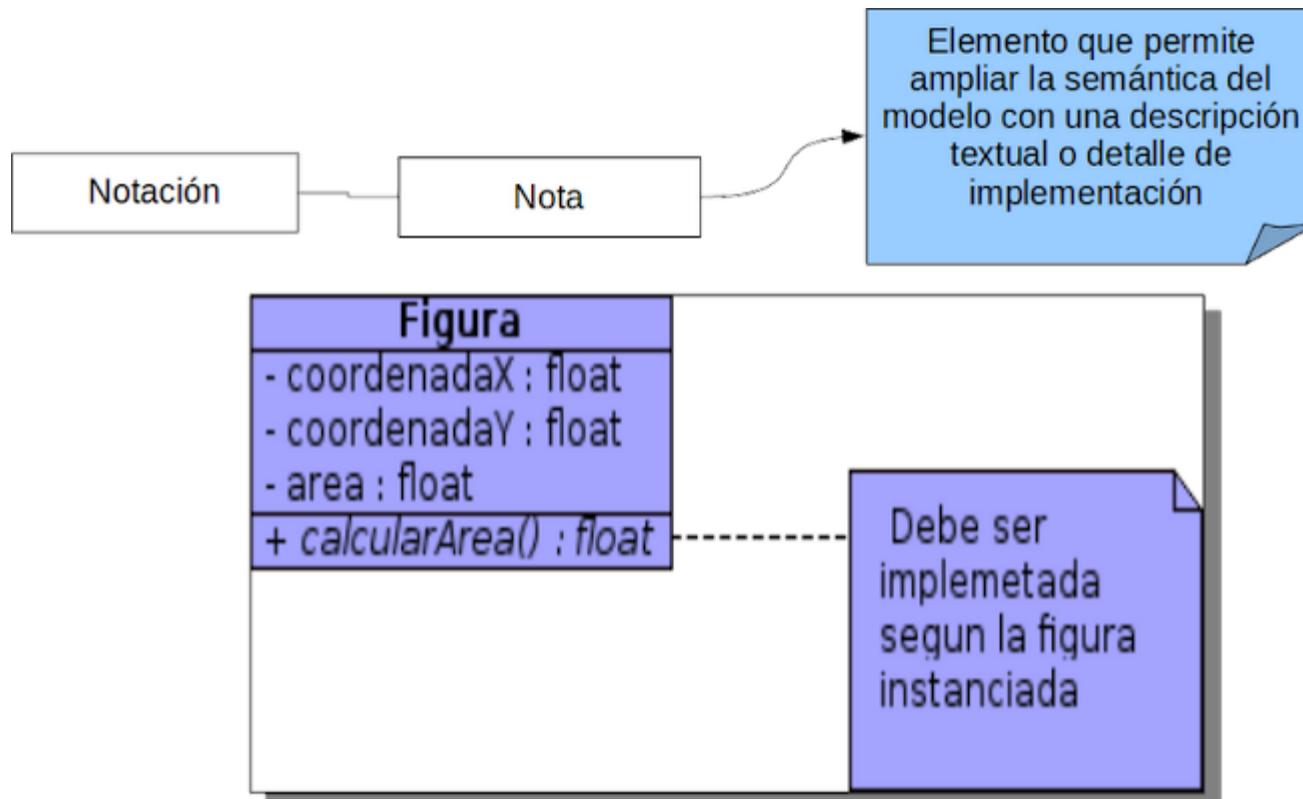
Introducción a UML

Modelo conceptual:



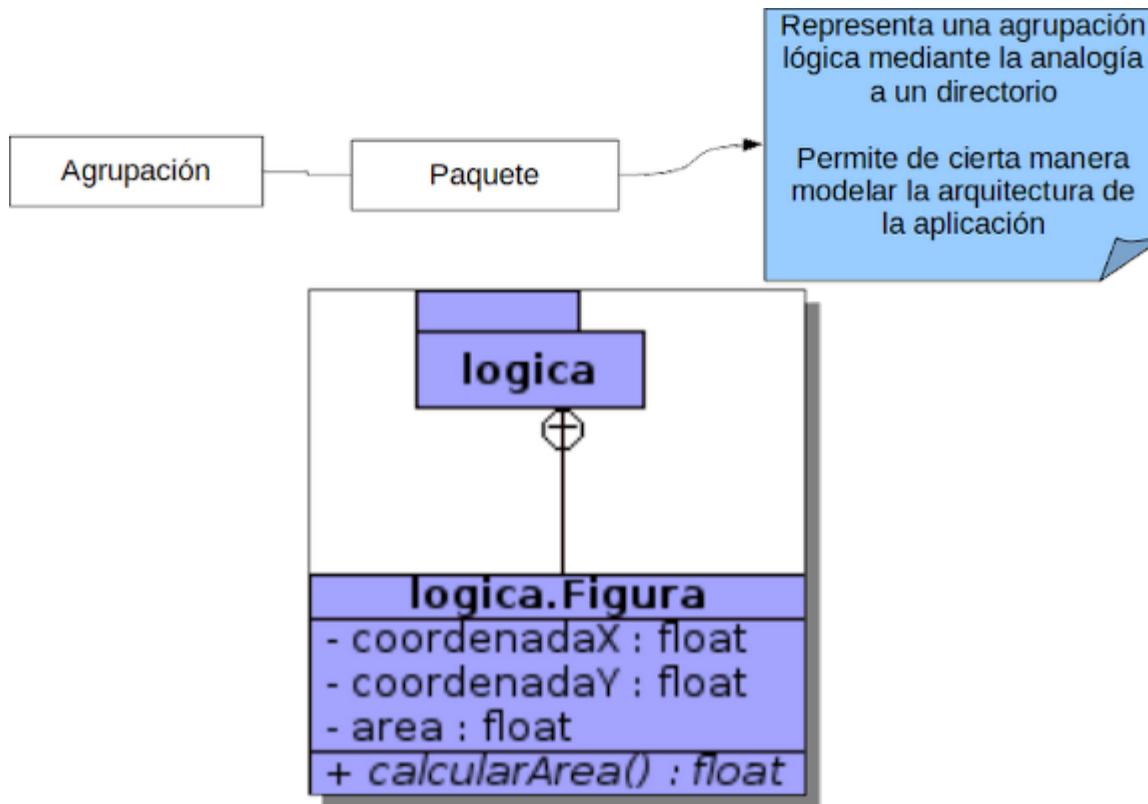
Introducción a UML

Modelo conceptual:



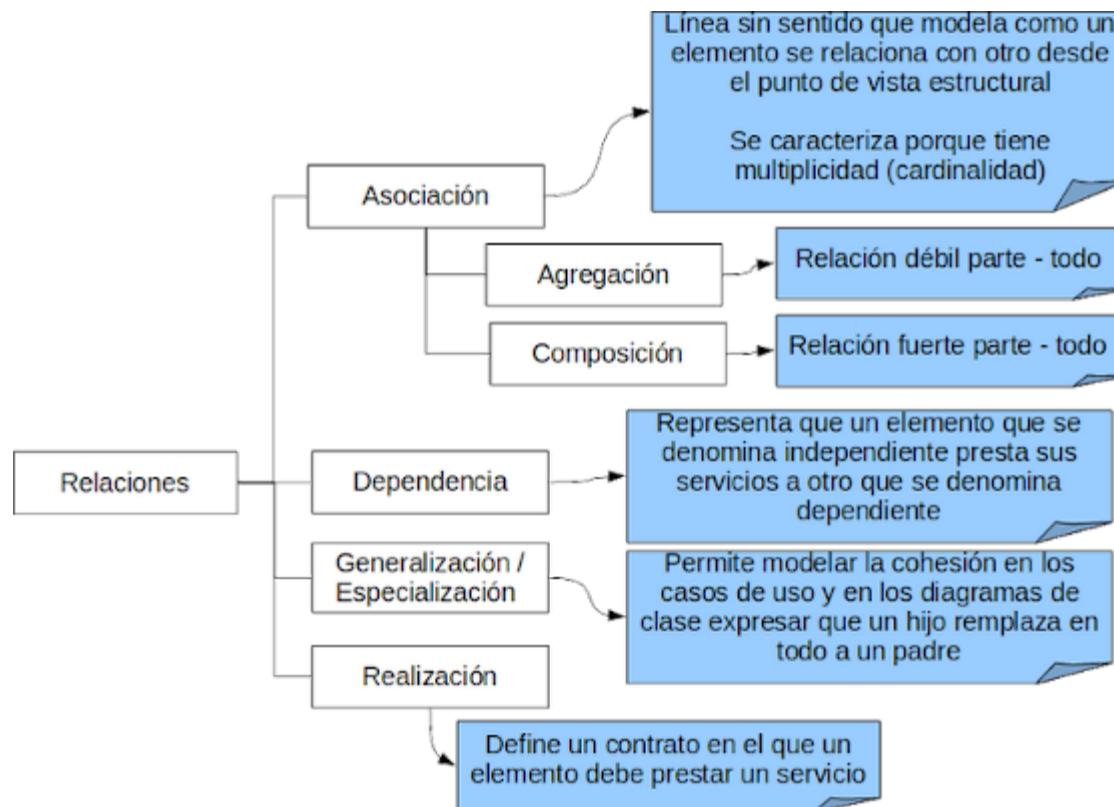
Introducción a UML

Modelo conceptual:



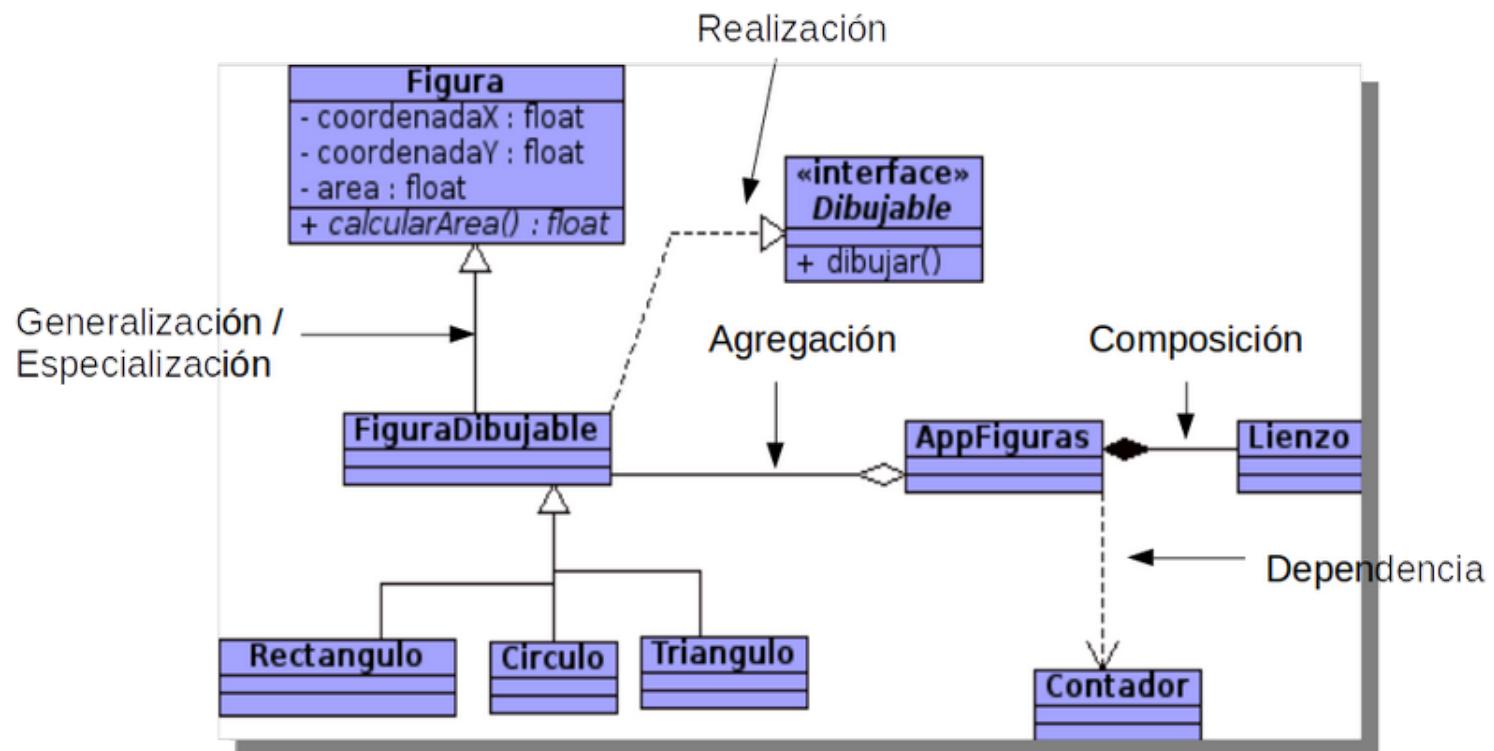
Introducción a UML

Modelo conceptual:



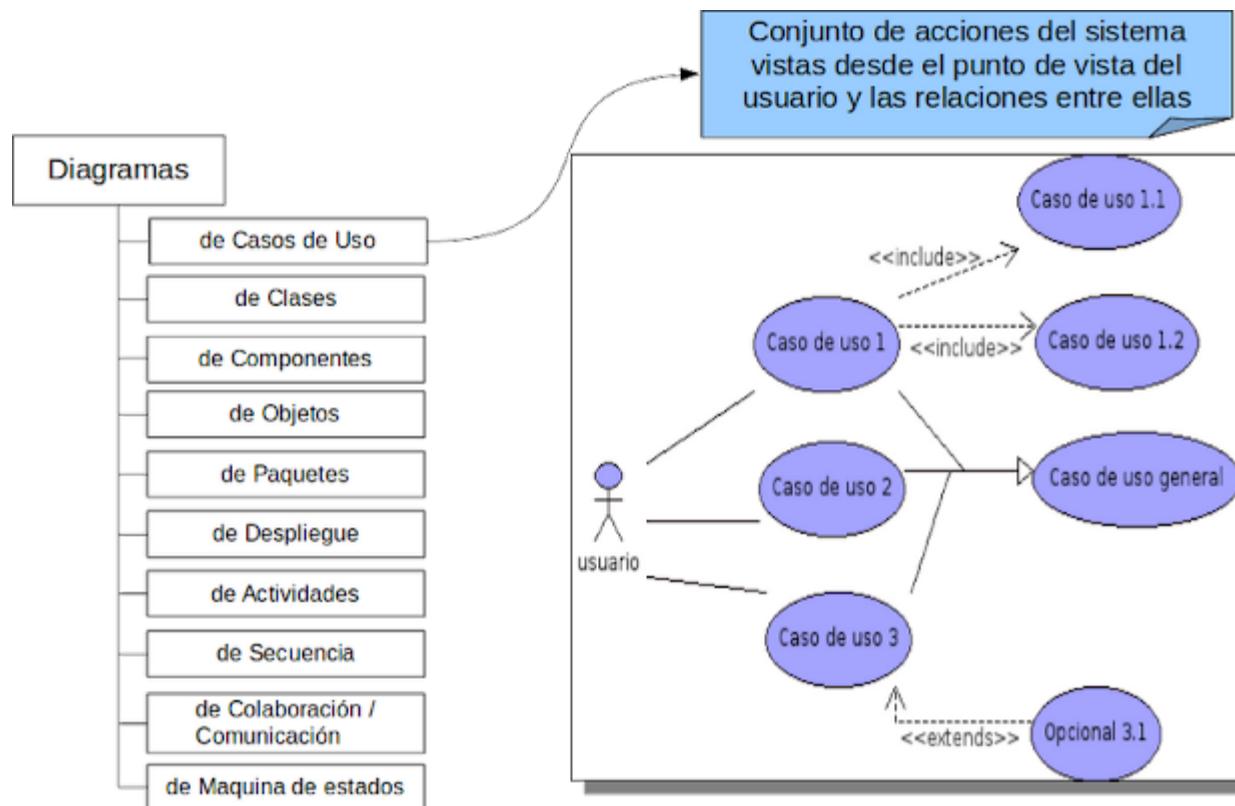
Introducción a UML

Modelo conceptual:



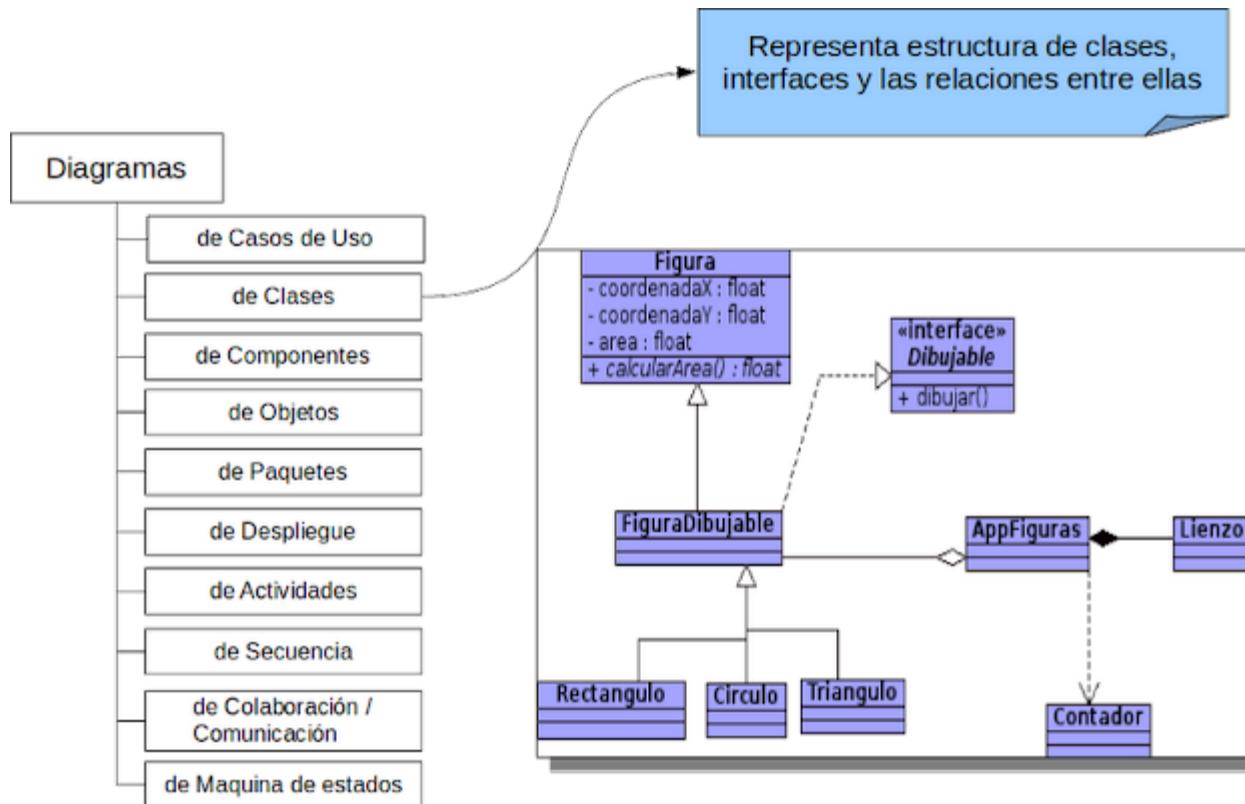
Introducción a UML

Modelo conceptual:



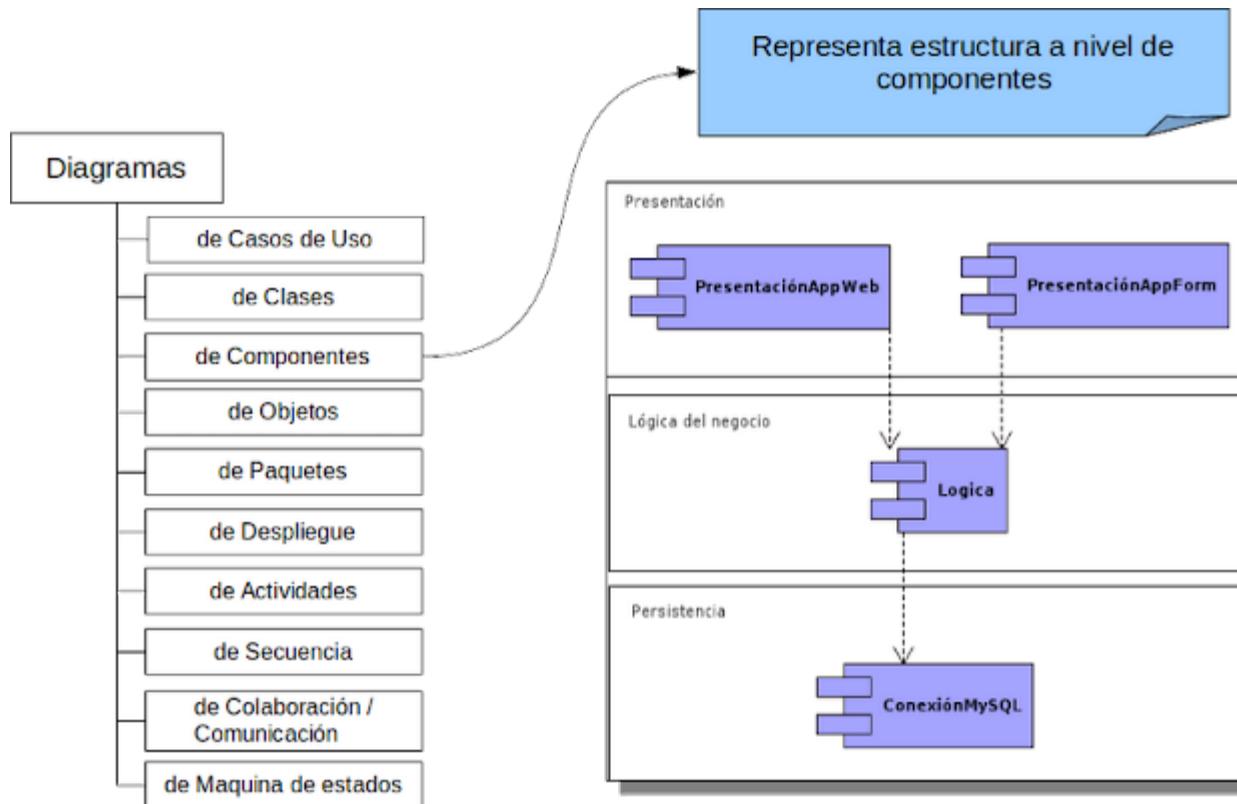
Introducción a UML

Modelo conceptual:



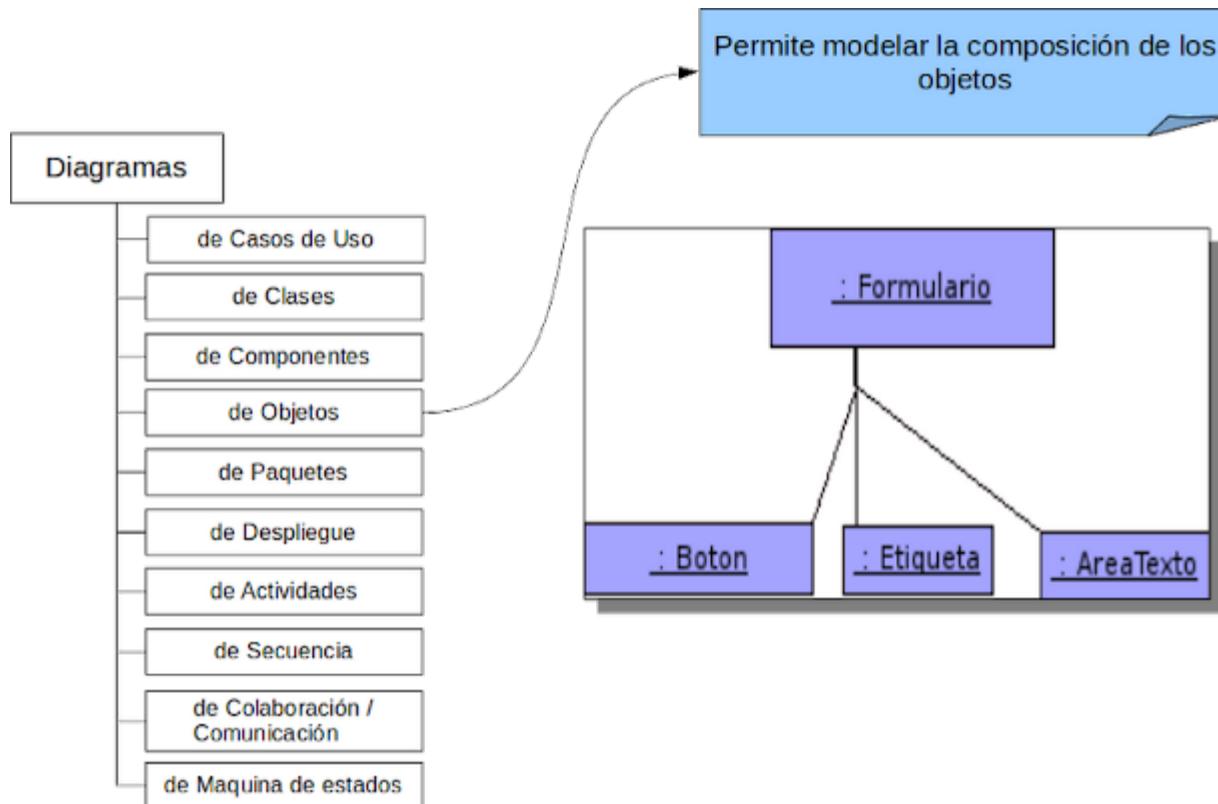
Introducción a UML

Modelo conceptual:



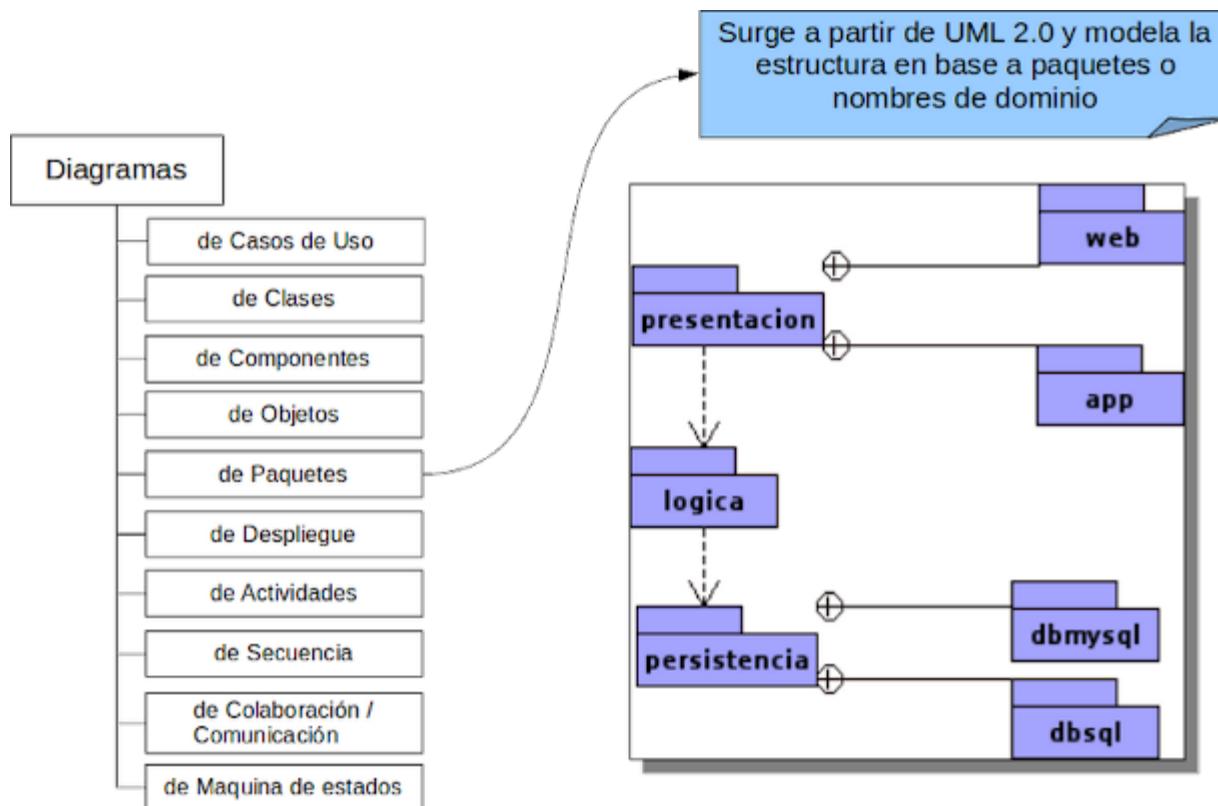
Introducción a UML

Modelo conceptual:



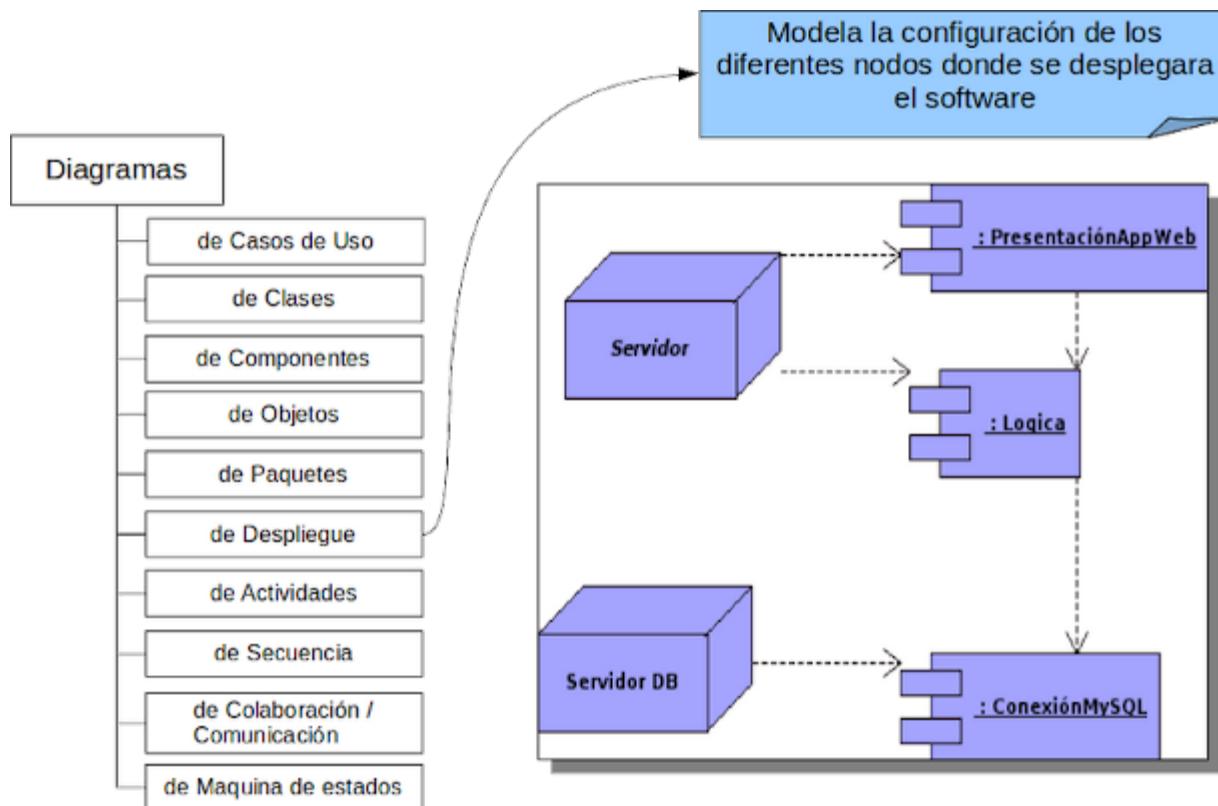
Introducción a UML

Modelo conceptual:



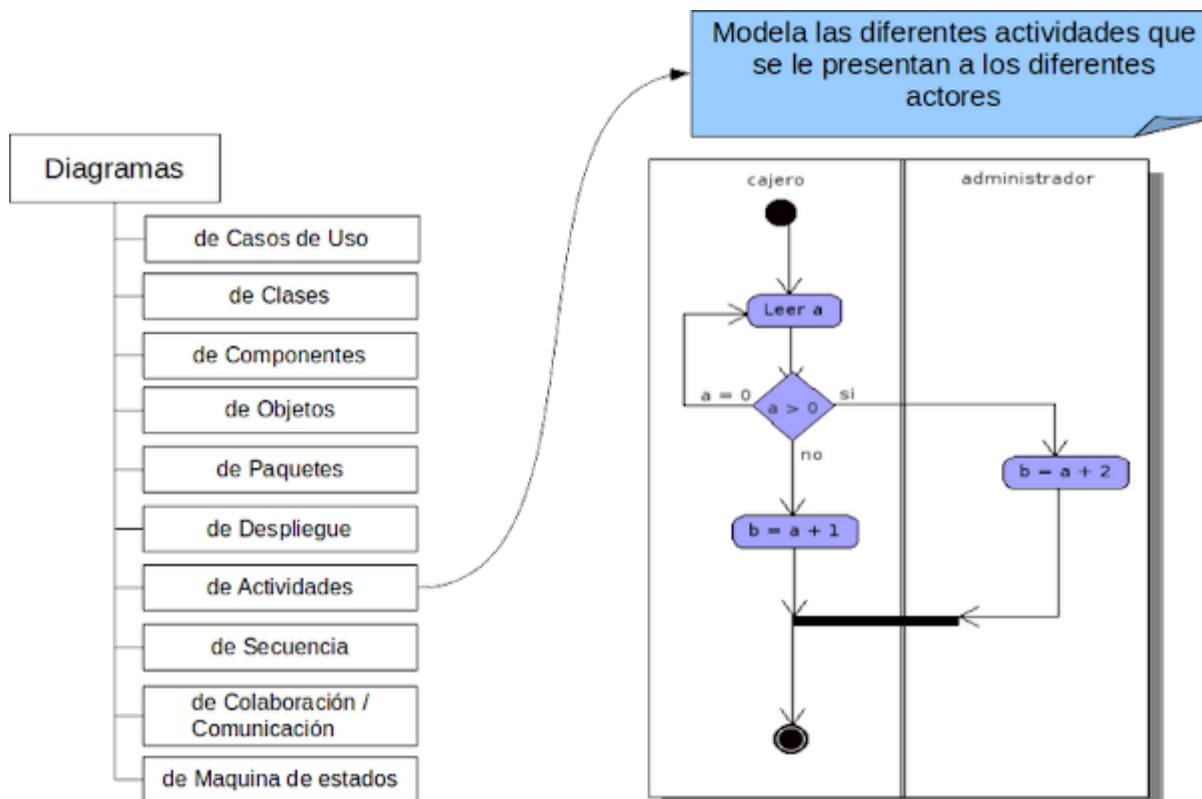
Introducción a UML

Modelo conceptual:



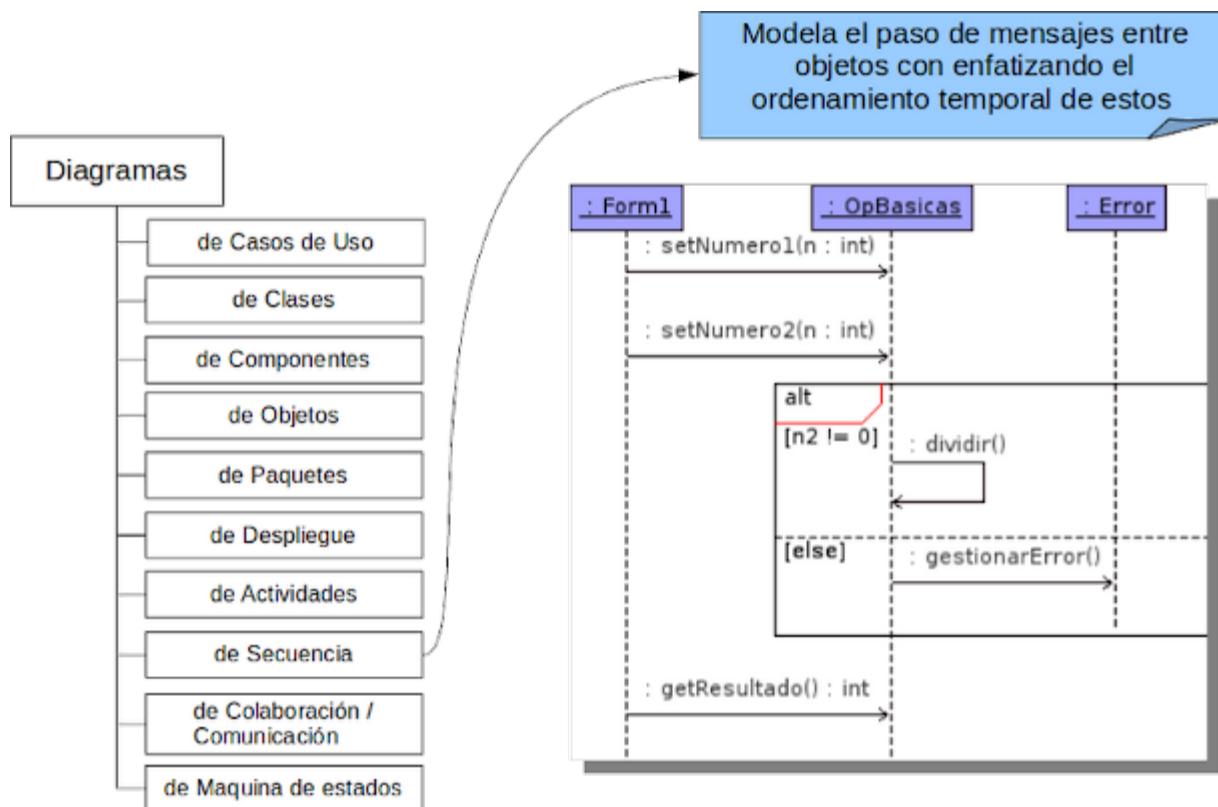
Introducción a UML

Modelo conceptual:



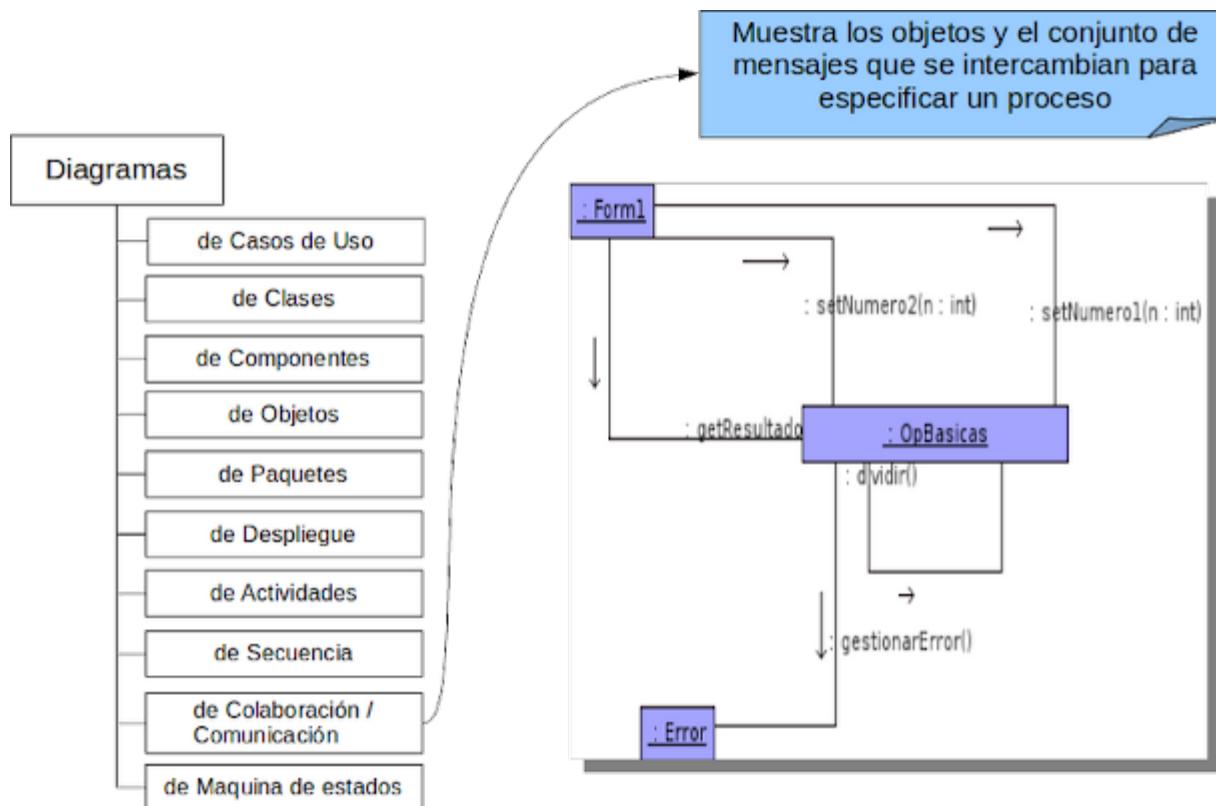
Introducción a UML

Modelo conceptual:



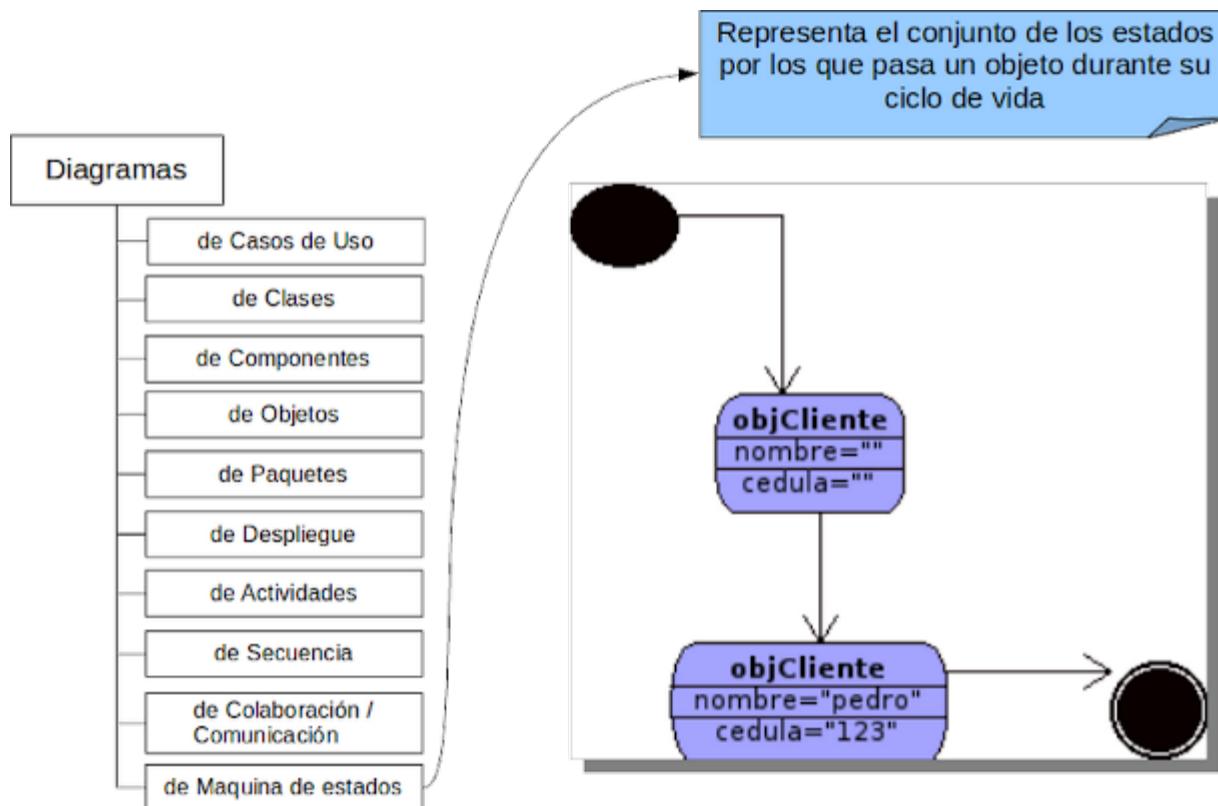
Introducción a UML

Modelo conceptual:



Introducción a UML

Modelo conceptual:



Introducción a UML

Ejemplo:

Cronómetro:

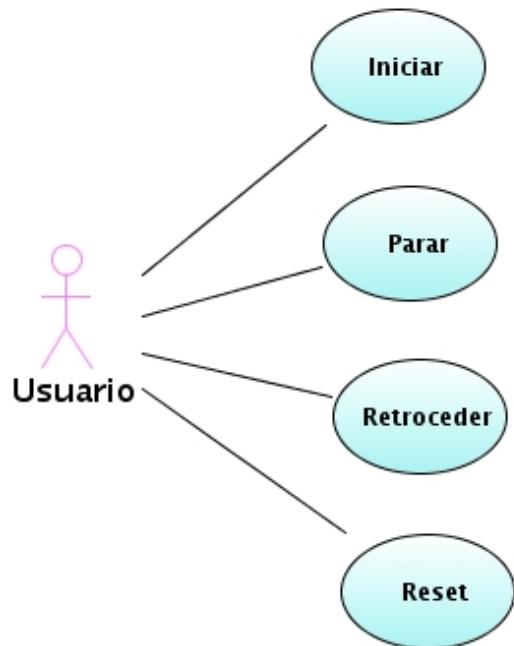
El caso práctico elegido para la elaboración de los diagramas de UML es un aplicativo de escritorio que permita al usuario manejar un cronómetro con la implementación normal de las funciones básicas de un elemento como este: avanzar, parar, retroceder, reset.



Introducción a UML

Ejemplo:

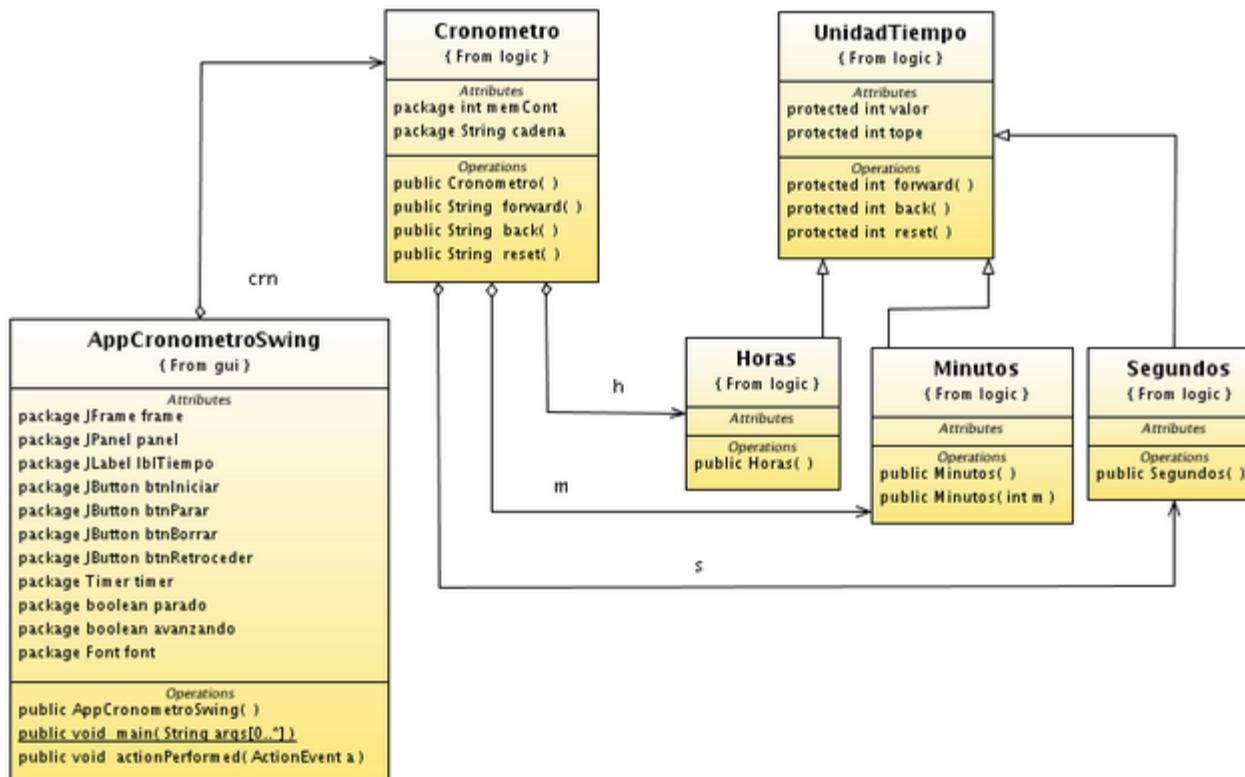
Diagrama de casos de uso:



Introducción a UML

Ejemplo:

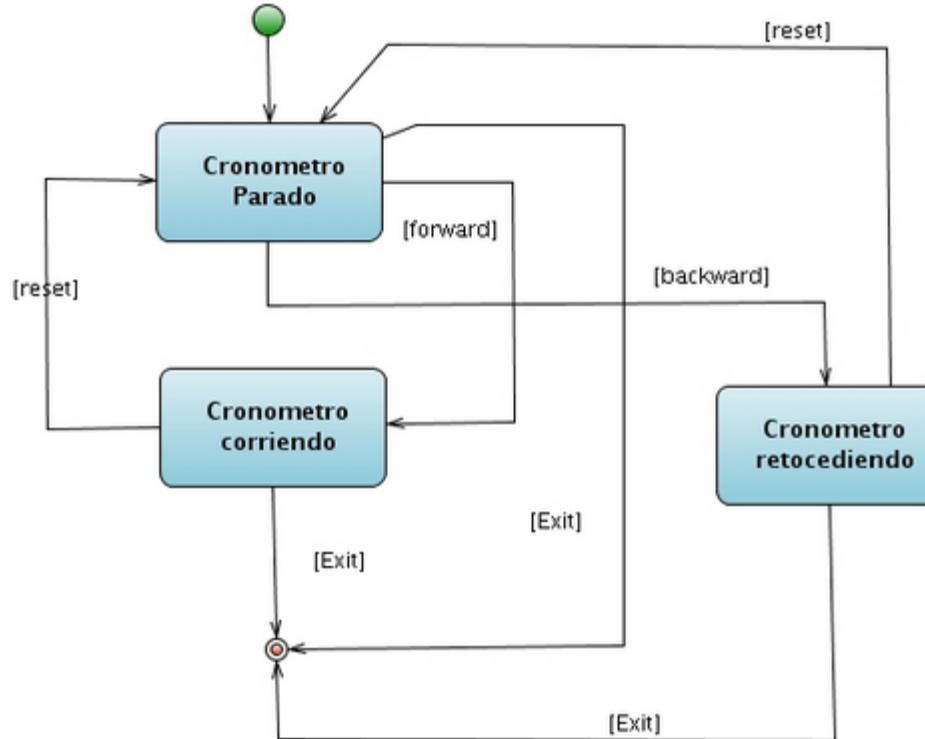
Diagrama de clases:



Introducción a UML

Ejemplo:

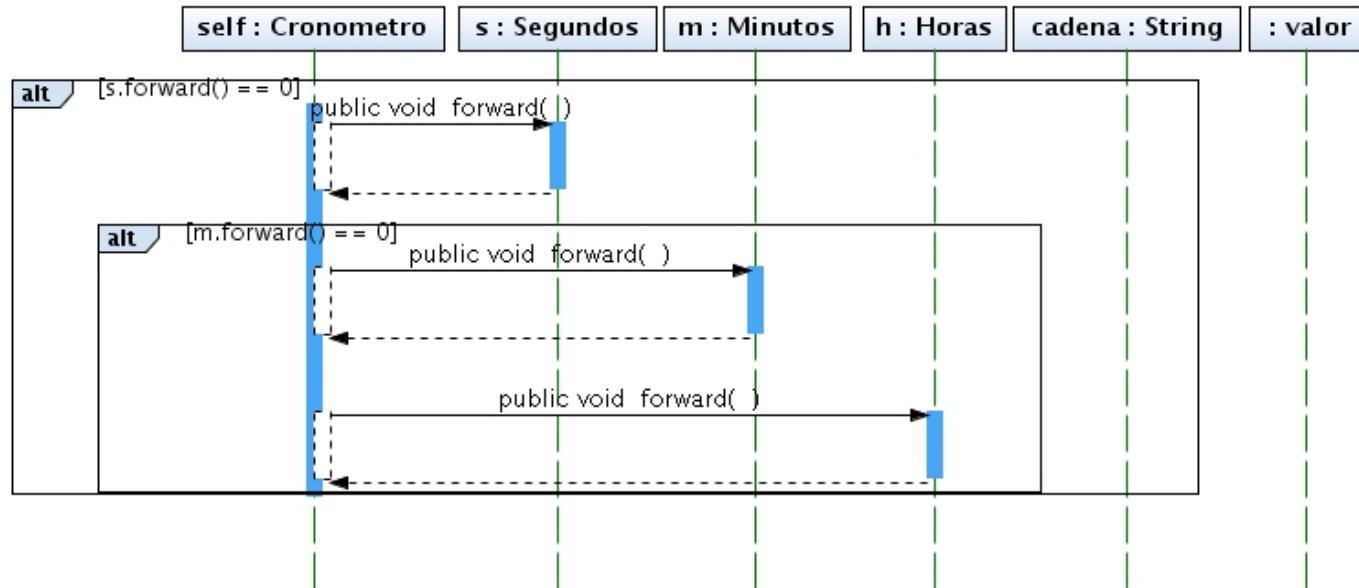
Diagrama de estados:



Introducción a UML

Ejemplo:

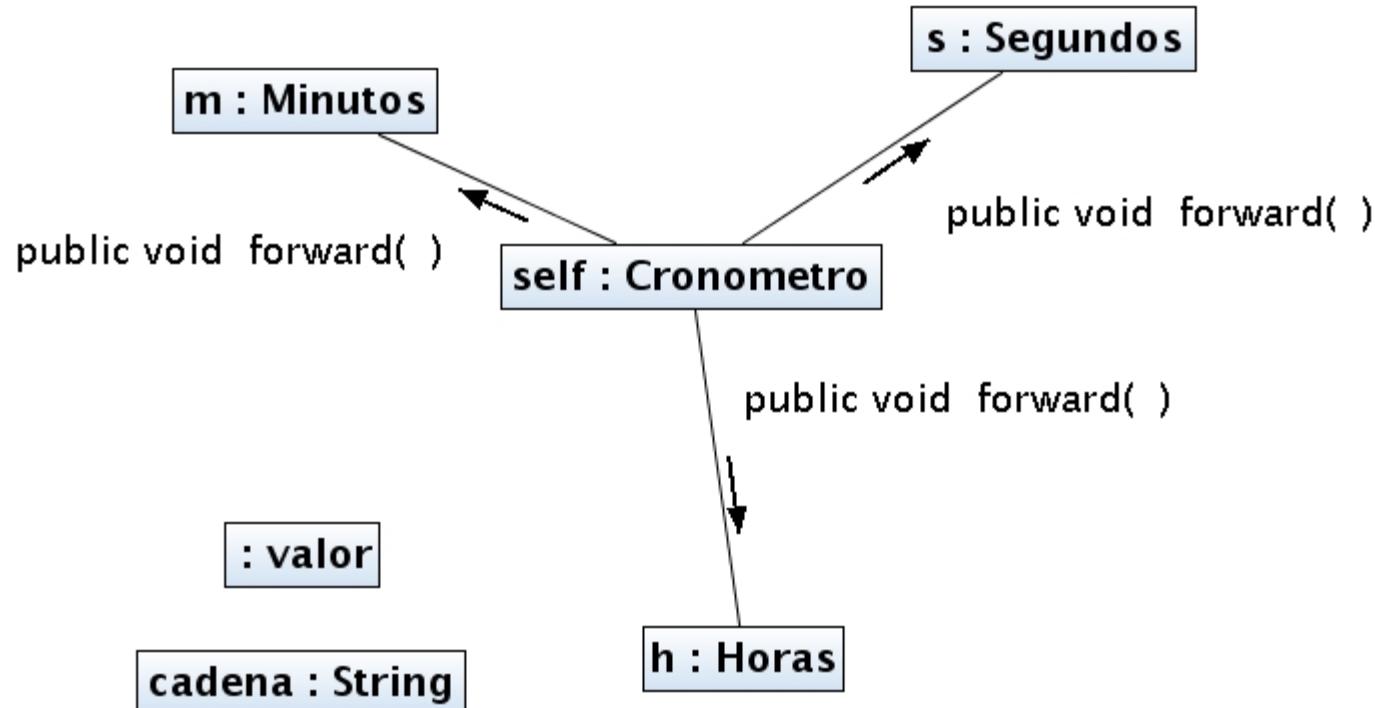
Diagrama de secuencia:



Introducción a UML

Ejemplo:

Diagrama de colaboración:



Introducción a UML

Ejemplo:

Diagrama de paquetes:

