El Diagrama de Clases en UML

El diagrama de clases es uno de los más utilizados en el desarrollo de aplicaciones software, especialmente dentro del paradigma de la programación orientada a objetos. Su objetivo principal es representar la estructura estática de un sistema, mostrando las clases que lo componen, sus atributos, sus operaciones y las relaciones que existen entre ellas. Cada clase representa una categoría o grupo de cosas que comparten propiedades y comportamientos similares. Por ejemplo, una clase puede corresponder a un grupo de vehículos, motores o personas, donde cada instancia de esa clase tendrá atributos que la caracterizan y métodos que definen su comportamiento.

En la notación UML, una clase se representa mediante un rectángulo dividido en tres compartimentos: el superior contiene el nombre de la clase, centrado y en negrita, con la primera letra en mayúsculas; el segundo contiene los atributos, justificados a la izquierda, y el tercero, los métodos o funciones, también justificados a la izquierda. Además, los atributos o métodos que son estáticos, es decir, pertenecen a la clase y no a cada instancia individual, se subrayan. Una clase en la que todos sus miembros son estáticos se conoce como clase utilidad, como ocurre con la clase Math en muchos lenguajes de programación. Este tipo de clases no se instancian y suelen marcarse como “leaf”, indicando que no pueden tener subclases.

UML también permite especificar la visibilidad de los atributos y métodos mediante símbolos: el signo más indica visibilidad pública, el menos indica privada, la almohadilla representa visibilidad protegida y una tilde indica visibilidad de paquete. Por otro lado, las clases abstractas, que no pueden instanciarse directamente y se utilizan como base para otras clases, se etiquetan con letra cursiva y negrita.

Además de las clases estándar, UML contempla estereotipos no oficiales que, aunque no forman parte del estándar formal, son ampliamente utilizados en herramientas como IBM Rational Software Architecture. Estos estereotipos incluyen clases de tipo Boundary, Control y Entity, y se corresponden lógicamente con los componentes del patrón Modelo-Vista-Controlador (MVC). Se emplean sobre todo en la fase de análisis orientado a objetos, no tanto en el diseño detallado, y ayudan a abstraer la implementación concreta de las clases, centrándose más en su función conceptual dentro del sistema. La clase Boundary representa una interfaz de entrada y salida del sistema, y suele emplearse para modelar la interacción entre actores y sistema. La clase Control modela la lógica de negocio o la coordinación del comportamiento, y la clase Entity representa datos o información relevante, ya sea persistente o no, que el sistema gestiona. Estas clases pueden representarse mediante símbolos gráficos o mediante estereotipos escritos como <Boundary>, <Control> o <Entity>.

El diagrama de clases también incluye las interfaces. Una interfaz es un clasificador que declara un conjunto de atributos y métodos públicos, estáticos y finales, pero que no contiene implementación. Cualquier clase que implemente una interfaz debe definir los métodos declarados en ella. Aunque no pueden instanciarse directamente, sí es posible declarar objetos de tipo interfaz, siempre y cuando se utilice el constructor de una clase que implemente dicha interfaz. Las interfaces se representan como clases normales con la etiqueta <interface>, y opcionalmente con un símbolo gráfico en forma de círculo con una T tumbada a la izquierda.

Una clase puede implementar varias interfaces, y una misma interfaz puede ser implementada por distintas clases.

Dentro del diagrama de clases existen distintos tipos de relaciones entre las clases. La asociación es una relación estructural que indica que dos clases conocen la existencia una de la otra y pueden interactuar. Esta relación puede ser simple o incluir propiedades como nombres de rol o cardinalidades. La cardinalidad especifica cuántas instancias de una clase pueden estar relacionadas con instancias de otra. Por ejemplo, una relación uno a muchos se indica con "1..*", mientras que una relación opcional se indica con "0..1". En algunos casos, la asociación puede transformarse en una clase si requiere almacenar atributos propios, como por ejemplo en el caso de representar funciones teatrales que incluyen fecha y lugar.

Otra relación importante es la agregación, que representa una relación contenedor-contenido de tipo débil. En este caso, el objeto contenido forma parte del contenedor, pero puede existir de forma independiente. Por ejemplo, una agenda puede contener múltiples contactos, pero estos pueden seguir existiendo aunque la agenda desaparezca. La agregación se representa mediante un rombo blanco y permite que el objeto contenedor y el contenido tengan ciclos de vida independientes. En contraste, la composición representa una relación todo-parte de tipo fuerte. Aquí, el objeto parte no puede existir sin el objeto todo. Cuando el objeto todo se destruye, también lo hacen sus partes. Un ejemplo sería una silla y sus patas: si se destruye la silla, sus patas también dejan de existir. Esta relación se representa con un rombo negro.

La herencia o generalización-especialización es otra relación fundamental en UML. Una subclase hereda los atributos y métodos de una superclase y puede añadir nuevos comportamientos o redefinir los heredados, lo que se conoce como polimorfismo. Esta relación se representa mediante una flecha vacía que apunta desde la subclase hacia la superclase.

La dependencia es una relación más débil y representa el hecho de que una clase utiliza o depende de otra de alguna forma. Se representa mediante una flecha discontinua, y sugiere que un cambio en la clase de la que se depende podría afectar al funcionamiento de la clase que la utiliza. Por último, la realización es una relación que se establece entre una clase concreta y una interfaz. La interfaz especifica el comportamiento, pero no su implementación, mientras que la clase que realiza dicha interfaz proporciona la implementación concreta de ese comportamiento. Esta relación se representa con una flecha discontinua terminada en triángulo vacío.

En conjunto, el diagrama de clases es una herramienta poderosa que permite representar de forma clara y estructurada los elementos fundamentales de un sistema software, facilitando la comunicación entre analistas, diseñadores y desarrolladores, y sirviendo como base para la implementación, el mantenimiento y la evolución del sistema.