INTRODUCCIÓN A LA NOTACIÓN UML

Diagramas de clases

1 Introducción

Este documento proporciona una breve descripción de la notación UML utilizada en los diagramas UML de clases.

2 Clase

Una clase UML (figura 1) representa un concepto dentro del sistema que se está modelando. Es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. Una clase se representa por un rectángulo, con el borde externo continuo, con tres compartimentos separados mediante líneas horizontales. El compartimento superior tiene el nombre de la clase y otras propiedades generales (incluido su estereotipo); el compartimento intermedio contiene el listado de atributos; el compartimento inferior contiene una lista de operaciones. Los compartimentos de atributos y operaciones se pueden eliminar para simplificar el diagrama. La eliminación no indica que no existan atributos u operaciones.

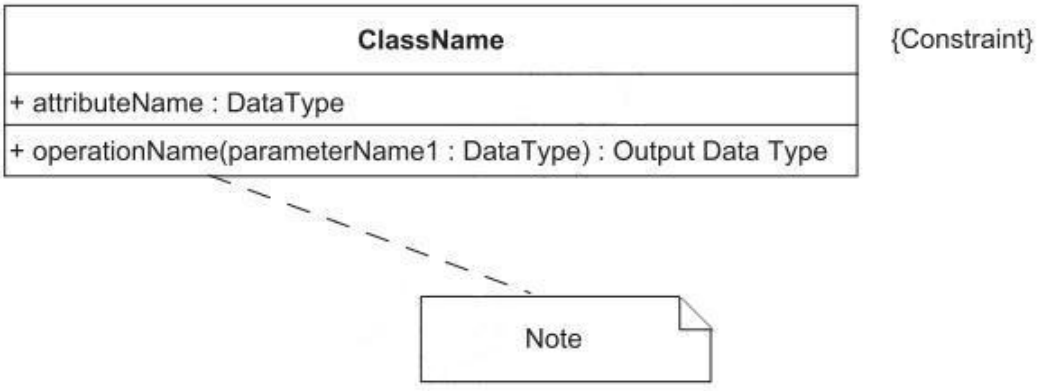

Figura 1 – Clase UML

La Especificación Técnica ISO/TS 19103 especifica que el nombre de una clase no debe contener espacios en blanco y que las palabras individuales contenidas en el nombre deben empezar con letra mayúscula.

3 Estereotipo

Los estereotipos extienden la semántica, pero no la estructura, de los tipos y clases pre-existentes. Los estereotipos a nivel de clase más utilizados son:

<<Type>> es un estereotipo de clase definido en la Norma ISO/IEC 19501. Type se usa para definir un dominio de objetos junto con operaciones aplicables a los objetos

sin definir su implementación física. También puede tener atributos y asociaciones que se definen únicamente con el propósito de especificar el comportamiento de las operaciones de los tipos y no representar ninguna implementación real del estado de los datos.

<<Interface>> es un estereotipo de clase definido en la Norma ISO/IEC 19501. Una Interface contiene un conjunto de operaciones que juntas definen un servicio ofrecido por la clase que realiza la interfaz. Una clase puede realizar varias interfaces y varias clases pueden realizar la misma interfaz. Las interfaces pueden no tener atributos, asociaciones y métodos. Una interfaz puede participar en una asociación siempre que la interfaz no pueda ver la asociación; esto es, una clase (otra diferente de una interfaz) puede tener una asociación con una interfaz que es navegable desde la clase pero no desde la interfaz.

<<DataType>> es un descriptor de un conjunto de valores que carecen de identidad (existencia independiente y la posibilidad de efectos colaterales). Data types incluyen tipos predefinidos de primitivas y tipos definibles por los usuarios. Un DataType es por consiguiente una clase con pocas operaciones o sin ninguna cuyo principal propósito es mantener el estado abstracto de otra clase para la transferencia, almacenamiento, codificación o almacenamiento permanente.

<<Enumeration>> es un tipo de dato cuyas instancias forman una lista de valores de nombres literales. Se declaran tanto los nombres de la enumeración como sus valores fijos. Una enumeración es una lista corta de posibles valores bien definidos dentro de una clase. Como ejemplo clásico tenemos los valores booleanos que sólo tienen 2 (o 3) posibles valores, VERDADERO, FALSO (y NULO). La mayoría de las enumeraciones se codificarán como un conjunto secuencial de enteros, a menos que se especifique otra cosa. En realidad, la codificación sólo se usa en los compiladores de lenguajes de programación.

<<CodeList>> se define en la Especificación Técnica ISO/TS 19103 como una enumeración flexible que utiliza cadenas de caracteres a través de un vínculo con una clave de tipo Diccionario y que devuelve valores de tipo cadena; por ejemplo, Diccionario (cadena, cadena). Las listas controladas son útiles para expresar listas extensas de posibles valores. Si se conocen completamente los elementos de la lista se debe utilizar una enumeración; si sólo se conocen los valores probables se debe usar una lista controlada. Las listas controladas mencionadas pueden codificarse conforme a una norma, tal como la Norma ISO 3166-1. Es más probable que las listas controladas expongan sus valores a los usuarios, y por lo tanto, son a menudo mnemotécnicas. Las diferentes implementaciones es probable que usen diferentes esquemas de codificación (con tablas de traducción hacia otros esquemas de codificación disponibles).

<<Union>> se define en la Norma ISO 19107 como un tipo que se compone de una y sólo una de las diversas alternativas (listadas como atributos miembro). Esto es parecido a las uniones discriminatorias de muchos lenguajes de programación. En algunos lenguajes de programación que utilizan punteros, éstos precisan un puntero “void” que pueda asignarse al tipo apropiado tal y como se determina por un atributo discriminador.

4 Atributo

Un atributo representa una característica común de los objetos de una clase. Un atributo se especifica mediante una cadena de texto que puede analizarse sistemáticamente para encontrar los elementos que describen las propiedades del atributo.

visibilidad nombre [multiplicidad]: tipo-expresión = valor-inicial

donde:

visibilidad puede ser pública (se indica con "+") o privada (se indica con "-").

nombre es una cadena de caracteres. La Especificación Técnica ISO/TS 19103 especifica que el nombre de un atributo no debe incluir espacios en blanco, debe comenzar con letra minúscula y las palabras individuales contenidas en el nombre, que sigan a la primera palabra, deben comenzar con letra mayúscula.

multiplicidad especifica el número de valores que una instancia de una clase puede tener para un atributo dado. La notación se explica en el capítulo B.10.

tipo-expresión identifica el tipo de dato del atributo.

valor-inicial especifica el valor por defecto del atributo.

5 Operaciones

Una operación representa un servicio que puede solicitar un objeto. Una operación se especifica mediante una cadena de texto que puede analizarse sistemáticamente para encontrar los elementos que describen las propiedades de la operación:

visibilidad nombre (parámetros): parámetros de salida

donde:

visibilidad puede ser pública (expresada mediante "+") o privada (expresada mediante "-").

nombre es una cadena de caracteres. La Especificación Técnica ISO/TS 19103 especifica que el nombre de una operación no debe incluir espacios en blanco, debe comenzar con letra minúscula y las palabras individuales, que siguen a la primera palabra del nombre, deben comenzar con mayúscula.

parámetros es una lista de parámetros, cada uno descrito por su nombre y tipo de dato. Estos se consideran como parámetros de entrada a menos que se especifique lo contrario.

parámetros de salida es una lista de valores devueltos, cada uno descrito por su tipo

de dato.

6 Restricción

Una restricción especifica una condición o limitación semántica. Aunque la Norma ISO/IEC 19501 define el Lenguaje de Restricción de Objetos *(Object Constraint Language)* para escribir restricciones, una restricción se puede escribir usando cualquier notación formal o el lenguaje natural. Una restricción se muestra como una cadena de texto entre llaves {}. Se coloca cerca del elemento al que se aplica. Si la notación para un elemento es una cadena de texto (como en el caso de un atributo), la cadena de restricción puede seguir a la cadena de texto del elemento entre llaves. Una restricción incluida como un elemento en una lista se aplica a todos los elementos subsiguientes de la lista, hasta el siguiente elemento de restricción o hasta el final de la lista.

7 Nota

Una nota contiene información textual. Se expresa mediante un rectángulo con la esquina superior derecha "doblada", unida a cero o más elementos del modelo mediante una línea discontinua. Las notas se pueden usar para incluir comentarios o restricciones.

8 Asociaciones

Una asociación (véase la figura 2) es una relación entre clases que definen conexiones entre sus instancias. Una asociación se representa con una línea continua que conecta los rectángulos de las clases. Una asociación puede tener un nombre, representado mediante una cadena de caracteres colocada cerca de la línea aunque no cercano a ninguno de los extremos. La Especificación Técnica ISO/TS 19103 define que el nombre de un atributo no debe incluir espacios en blanco y que las palabras individuales en el nombre deben comenzar con letra mayúscula. Los extremos de la asociación muestran información pertinente a la clase en ese extremo, incluyendo la multiplicidad y el nombre del rol.

Figura 2 – Asociaciones UML

9 Nombre del rol

Un nombre de rol identifica una asociación y especifica el comportamiento de la

clase en ese extremo con respecto a la clase del otro extremo de la asociación. En la figura B.2 “roleAlpha” describe el rol que tiene la clase Alpha con respecto a la clase Beta. Un nombre de rol se representa mediante una cadena de caracteres. La Especificación Técnica ISO/TS 19103 especifica que el nombre de rol no debe incluir espacios en blanco, debe empezar por letra minúscula y las palabras individuales contenidas en el nombre, que siguen a la primera palabra, deben comenzar por letra mayúscula.

10 Multiplicidad

La multiplicidad especifica el número de instancias de una clase que pueden estar asociadas con la clase del otro extremo de la asociación. Todos los valores que se muestran en la figura B.3 son válidos. Tienen los siguientes significados:

a) Cero o una instancia de Alpha pueden asociarse con una instancia de Beta.

b) Cero o más instancias de Beta pueden asociarse con una instancia de Alpha.

c) una y sólo una instancia de Gamma puede asociarse con una instancia de Delta.

d) Siendo n un número entero, n y sólo n instancias de Delta pueden asociarse con una instancia de Gamma.

e) Siendo n1 y n2 números enteros, con n2 > n1, el número de instancias de Epsilon que pueden asociarse con una instancia de Phi puede estar dentro del intervalo de n1 a n2.

f) Siendo n un número entero, n o más instancias de Phi pueden asociarse con una instancia de Epsilon.

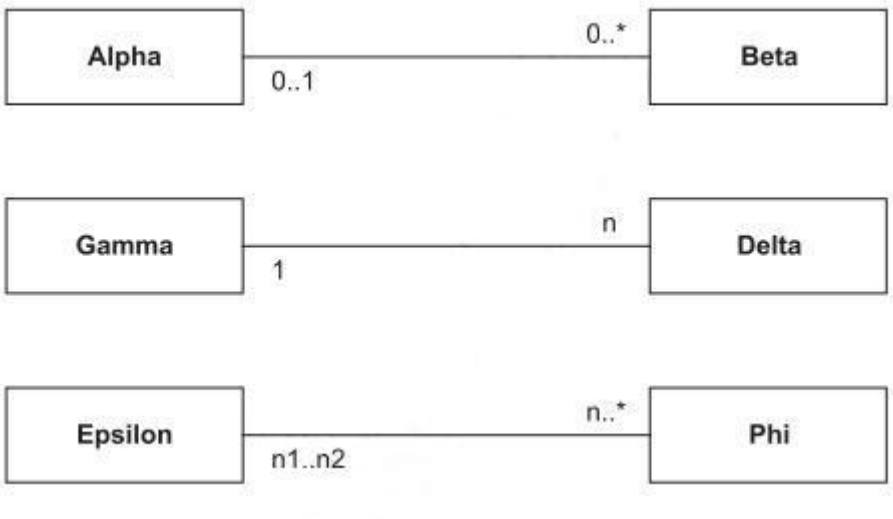

Figura 3 – Multiplicidad UML

11 Navegabilidad

Una flecha situada en el extremo de la línea de una asociación indica que se permite la navegación hacia la clase que apunta la flecha. Por ejemplo, en la figura 4, la asociación es navegable desde el usuario hasta el proveedor. Esto significa que una instancia de la clase Phi tiene acceso a la información disponible en una instancia de la clase Epsilon. Por ejemplo, una operación especificada/definida para Phi podría usar el valor de un atributo de Epsilon.

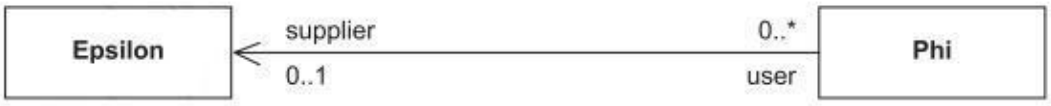

Figura 4 – Navegabilidad UML

12 Agregación

Las asociaciones pueden usarse para mostrar relaciones de agregación o composición entre las clases. Un rombo sin rellenar en el extremo de una asociación indica que la clase de ese extremo de la asociación es una agregado de instancias de la clase del otro extremo de la asociación. Por ejemplo, la clase llamada Gamma, en la figura 5, es un agregado de cero o más instancias de la clase llamada Delta. La agregación se considera una forma débil de composición.
Un ejemplo típico de Agregación es la asociación existente entre un racimo de uvas y los granos de uva que lo forman. Cumplen los dos criterios básicos de las agregaciones: si el racimo de uvas (superclase) se rompe y desaparece, los granos de uva (subclase) no tienen porqué desaparecer; y es posible concebir un grano de uva separado del racimo.

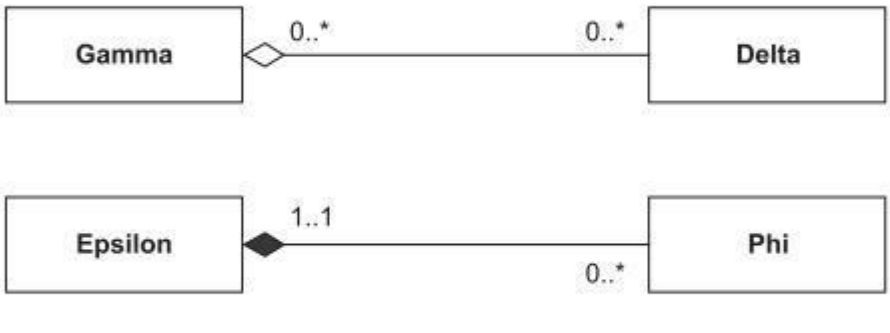

Figura 5 – Agregación y Composición

13 Composición

Un rombo negro en un extremo de la asociación indica que la clase al final de

esa asociación está compuesta por instancias de la clase del otro extremo de la asociación. Por ejemplo, la clase denominada Epsilon en la Figura 5 está compuesta por cero o más instancias de la clase llamada Phi. Los miembros de un compuesto no pueden existir de forma independiente a la clase compuesta, ni pueden ser miembro de más de una clase compuesta.
Un ejemplo típico de Composición es la asociación existente entre una mano y los cinco dedos que la forman. Cumplen los dos criterios esenciales de una composición: si desaparece la mano o se destruye, los cinco dedos también desaparecen o se destruyen; y no tiene sentido concebir un dedo separado de su mano.

En ocasiones puede presentarse un caso límite entre Agregación y Composición que cumple parcialmente ambos criterios y puede considerarse tanto una cosa como la otra. En tal situación y por defecto, lo aconsejable es modelar la asociación como una Agregación.

14 Dependencia

Una dependencia establece que la implementación o funcionamiento de uno o más elementos requiere la presencia de uno u otros elementos más. Una dependencia indica una relación semántica entre dos elementos de modelado (o dos conjuntos de elementos de modelado). La dependencia relaciona los elementos de modelado entre sí y no necesita un conjunto de instancias para tener significado. Una dependencia se representa como una flecha discontinua entre dos elementos de modelado. El elemento de modelado que está en la cola de la flecha (el cliente) depende del elemento de modelado que está en la punta de la flecha (el proveedor).
El tipo de dependencia se puede indicar con una palabra clave entre comillas latinas o angulares, tal como <<import>>, <<refine>>, o <<use>>. En el ejemplo de la Figura 6, Epsilon tiene una dependencia de tipo <<use>> sobre Phi.

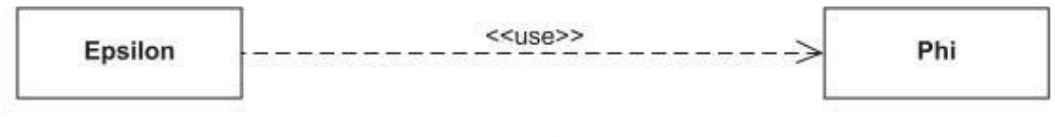

Figura 6 – Dependencia

15 Generalización

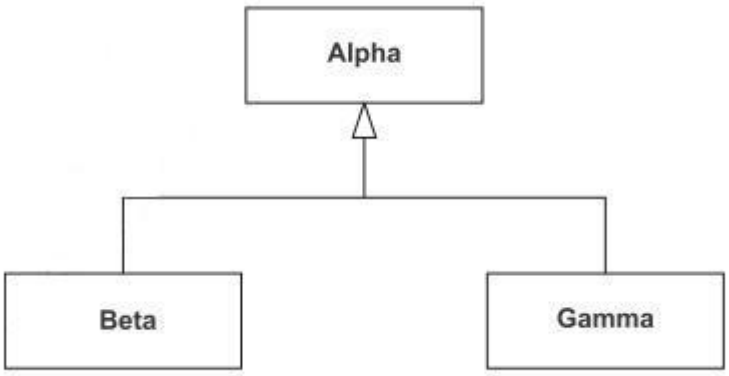

Figura 7 – Generalización UML

La Norma ISO/IEC 19501 define la generalización (figura 7) como una relación taxonómica entre un elemento más general y un elemento más específico. El elemento más específico es completamente consistente con el elemento más general y contiene información adicional. Se puede utilizar una instancia del elemento más específico cuando se permite utilizar el elemento más general. La generalización se representa como una línea continua desde el hijo (elemento más específico, tal como una subclase) hasta el padre (elemento más general, tal como una superclase), con un triángulo vacío en el extremo de la línea donde se encuentra el elemento más general.

16 Realización

La realización especifica una relación de realización entre un tipo o interfaz (el proveedor) y una clase que la implementa (el cliente). Se requiere el cliente para admitir todas las operaciones declaradas por el proveedor. La implementación de un tipo o una interfaz mediante una clase se representa con una línea discontinua con una cabeza de flecha triangular (una “flecha de generalización” discontinua).