Sesión 23 y 24
Diagrama de Clases (IV)

Unidad 4

Modelado del comportamiento estático del sistema

Mg. Gustavo G. Delgado Ugarte

Clases Abstractas

- A veces, se utiliza la generalización para declarar una bonita, genérica, clase reutilizable, que no será capaz de implementar todos los comportamientos que son necesarios para la clase general
  - Se debe dejar a las subclases decidir

Clases Abstractas

- Para indicar que la implementación de las operaciones almacenar (..) y recuperar (..) se va a dejar a las subclases, al declarar las operaciones como algo abstracto, escribir su firma en cursiva

Clases Abstractas

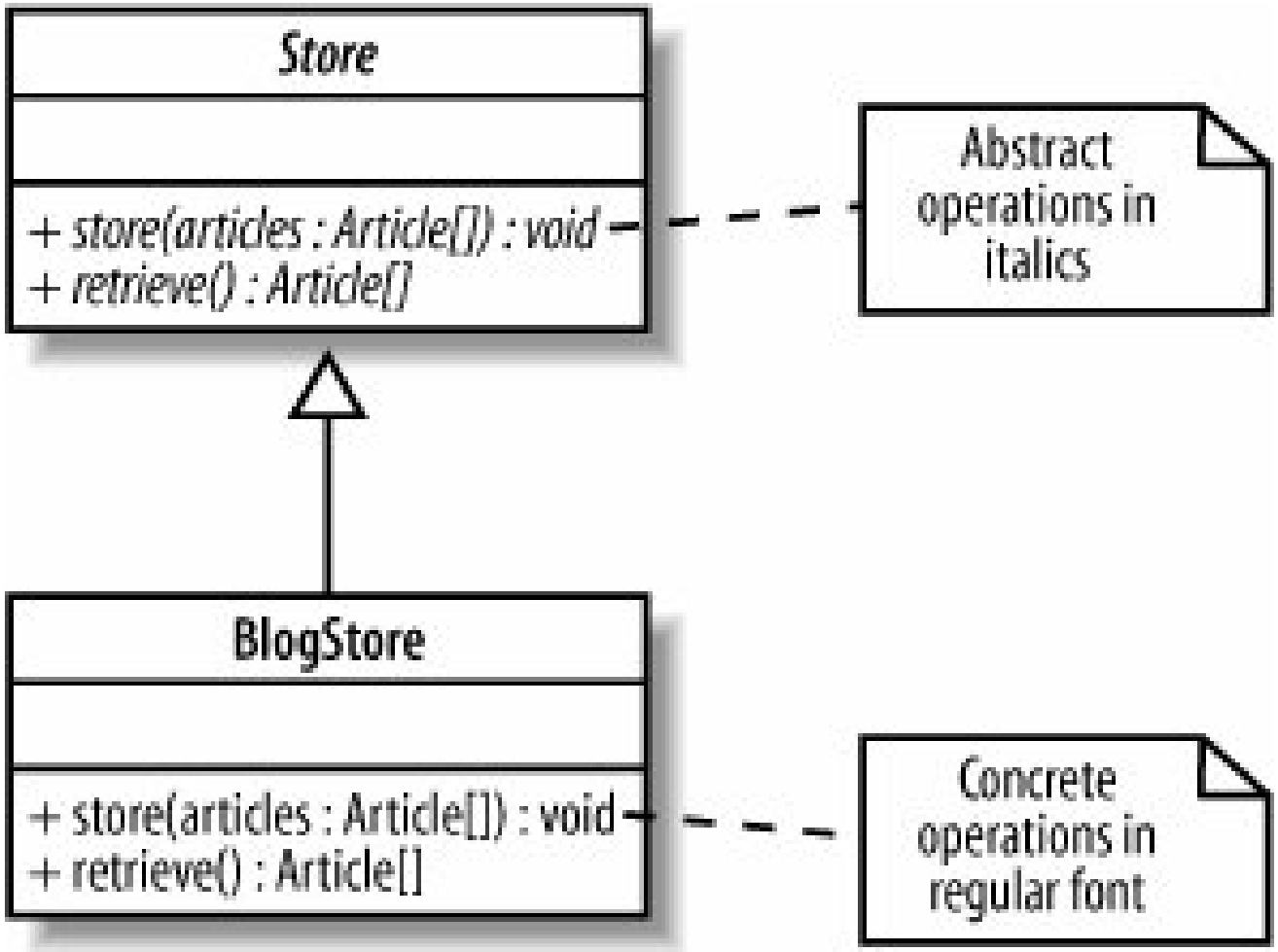
- Una operación abstracta no contiene una implementación del método y es realmente un marcador de posición que dice: "dejo la implementación de este comportamiento a mi subclases.“
- Si alguna parte de una clase se declara abstracta, la clase en sí también debe ser declarado como abstracto la implementación su nombre en letra cursiva

Clases Abstractas

Clases Abstractas

- Una clase abstracta no puede ser instanciada como un objeto porque hay partes de la definición de la clase que falta: las piezas abstractas
- Clases hijo de la clase abstracta pueden crear instancias de objetos, si completan todas las piezas abstractas faltantes en los padres, convirtiéndose así en una clase concreta

Clases Abstractas



Clases Abstractas

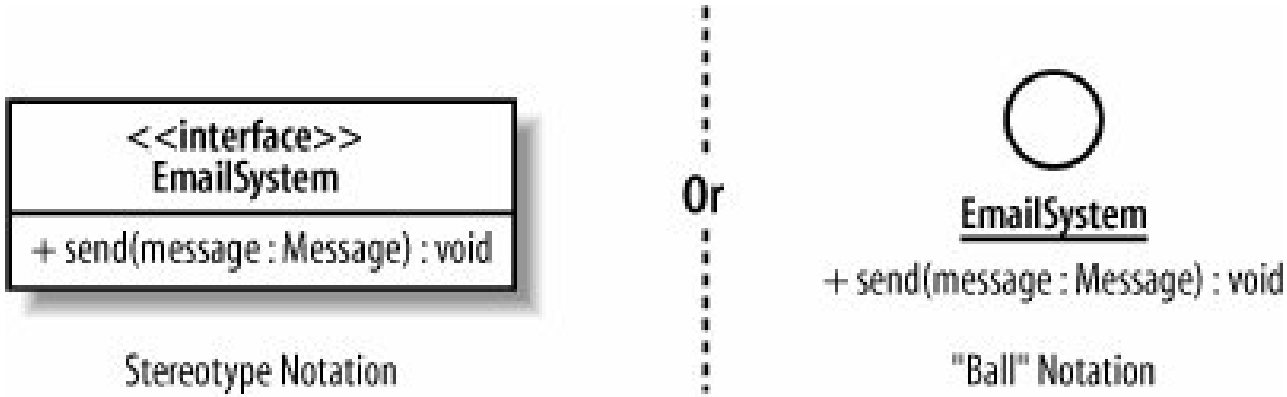
- Las clases abstractas son un mecanismo muy poderoso que le permiten definir el comportamiento y atributos comunes, pero dejan algunos aspectos de cómo una clase trabajará a las subclases concretas
- Un gran ejemplo de las clases abstractas e interfaces donde se utilizan definición de roles genéricos y de comportamiento que conforman patrones de diseño
- Sin embargo, para implementar una clase abstracta, hay que utilizar la herencia, por lo tanto, es necesario estar al tanto de todo el equipaje que viene con la relación de generalización de acoplamiento fuerte y bien

Interfaces

- Una interfaz es una colección de operaciones que no tienen implementaciones de los métodos correspondientes muy similar a una clase abstracta que contiene sólo métodos abstractos
- Pensar en una interfaz como un contrato muy simple que declara: "Estas son las operaciones que deben ser implementadas por las clases que intentan cumplir este contrato."
- A veces, una interfaz contienen atributos también, pero en esos casos, los atributos suelen ser estáticos y con frecuencia son constantes

Interfaces

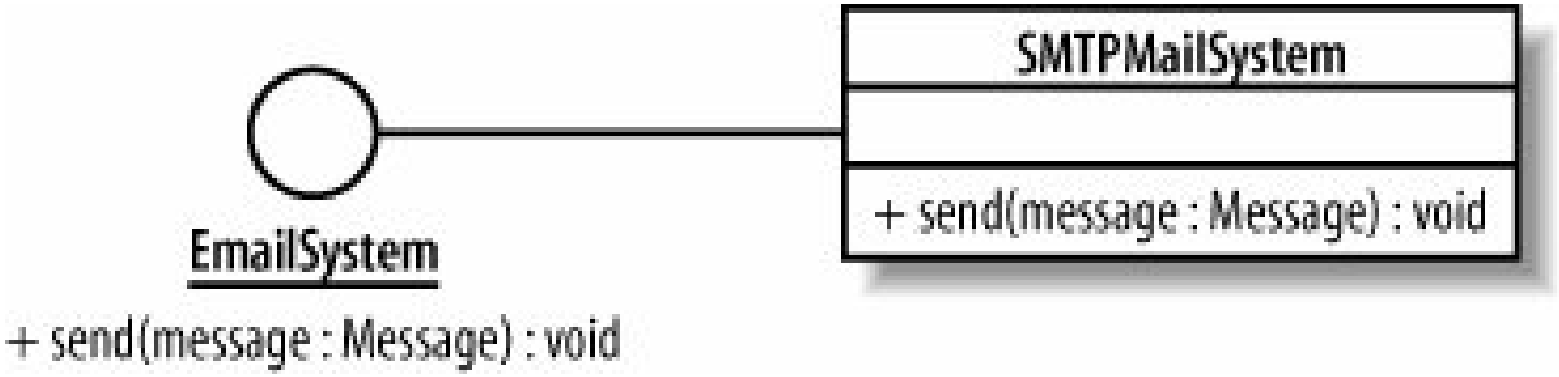
- En UML, una interfaz puede ser mostrado como una notación de clase estereotipada o mediante el uso de su notación propia, la bola



Interfaces

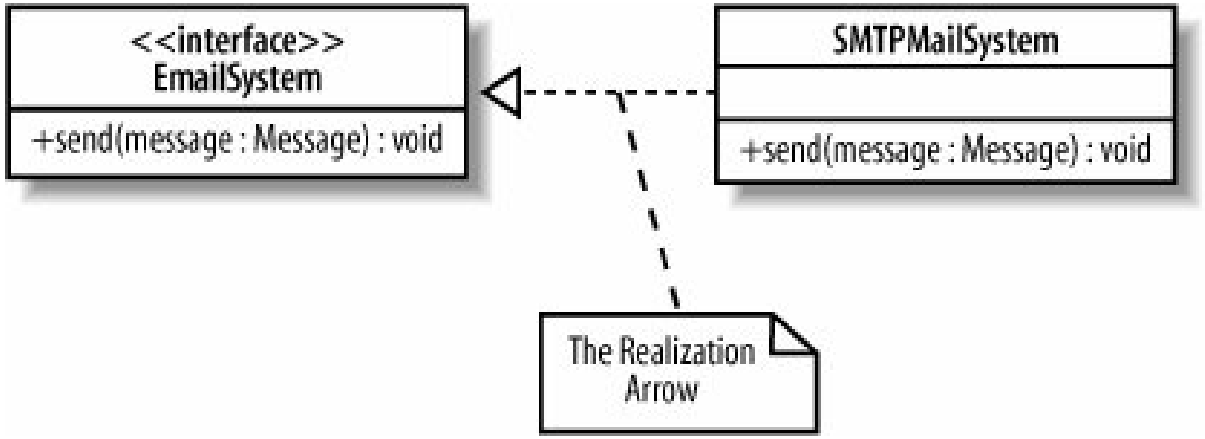
- En UML, una interfaz puede ser mostrado como una notación de clase estereotipada o mediante el uso de su notación propia, la bola
- Usted no puede crear instancias de una interfaz en sí misma, al igual que no se puede crear instancias de una clase abstracta. Esto se debe a todas las implementaciones de las operaciones de una interfaz que faltan hasta que se realiza por una clase. Si está utilizando notación interfaz la "bola", entonces la realización de una interfaz se realiza mediante la asociación con una clase

Interfaces


SMTPMailSystem
+ send(message : Message) : void
EmailSystem
+ send(message : Message) : void

Interfaces

- Si se ha utilizado la notación estereotipo de su interfaz, una nueva flecha es necesaria para demostrar que esto es una relación de realización



METAMODELO DEL DIAGRAMA DE CLASES

Metamodel[1]

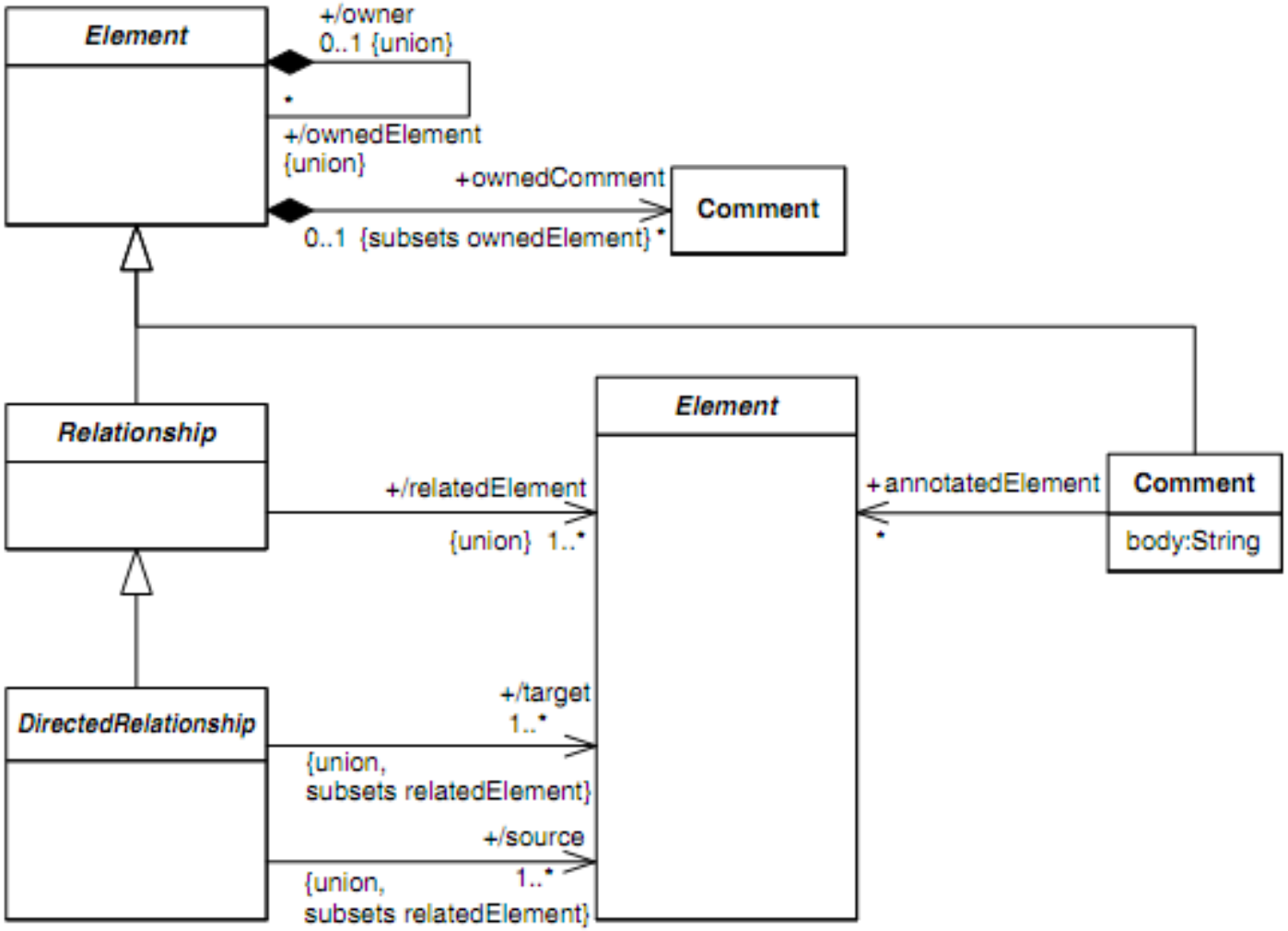


FIGURE 2.12 The basic metamodel concepts.

Metamodel

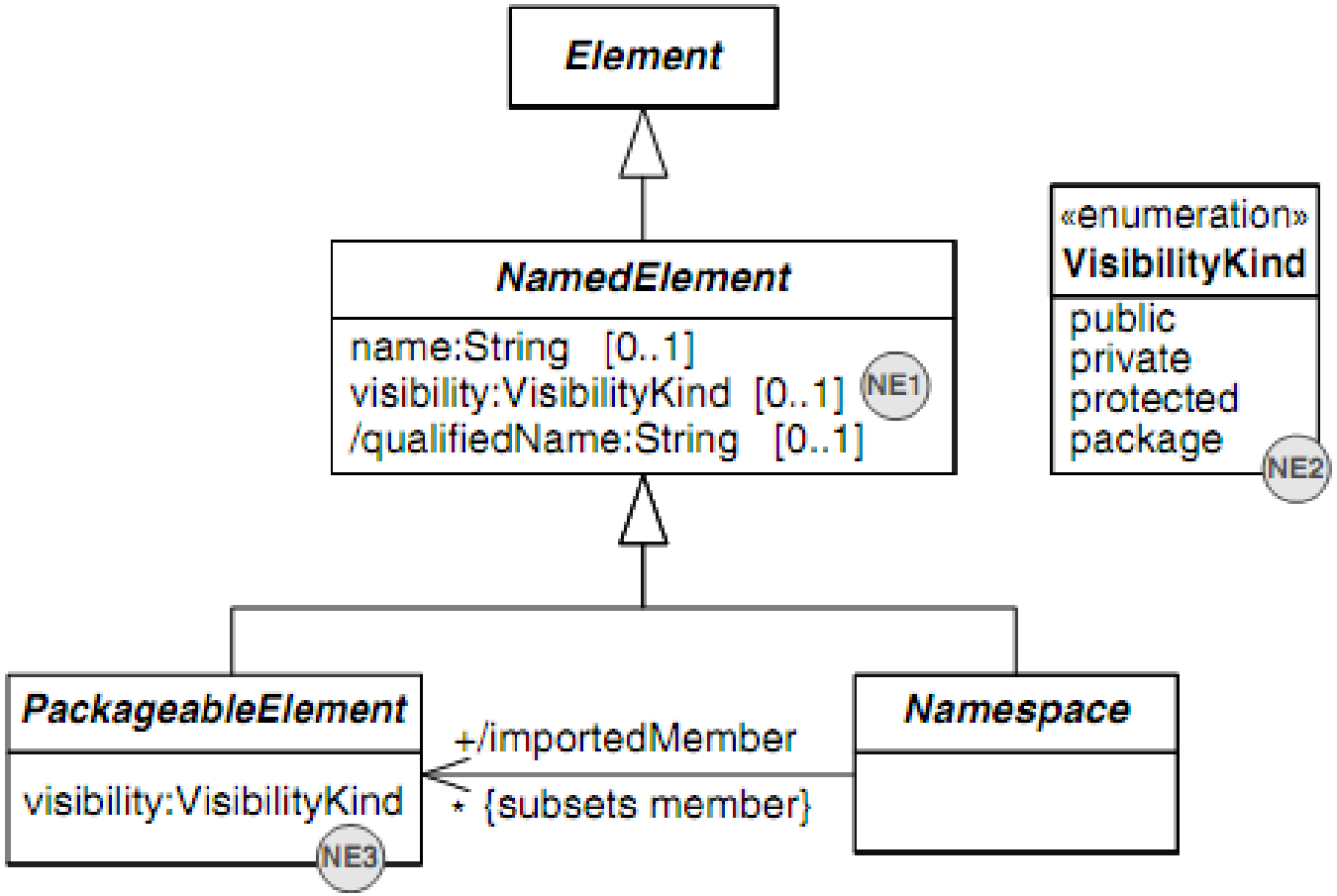


FIGURE 2.15 The metamodel for NamedElement.

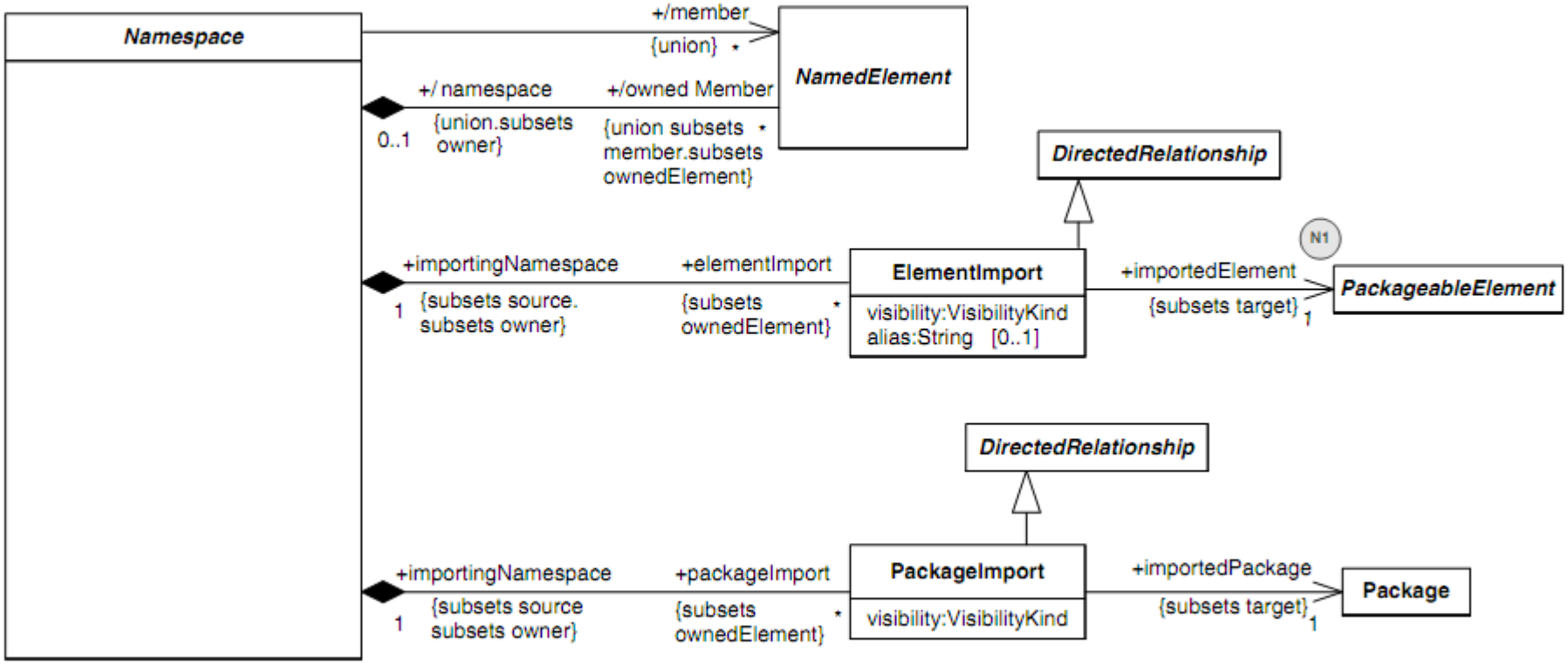


FIGURE 2.16 The metamodel for namespaces.

Metamodel

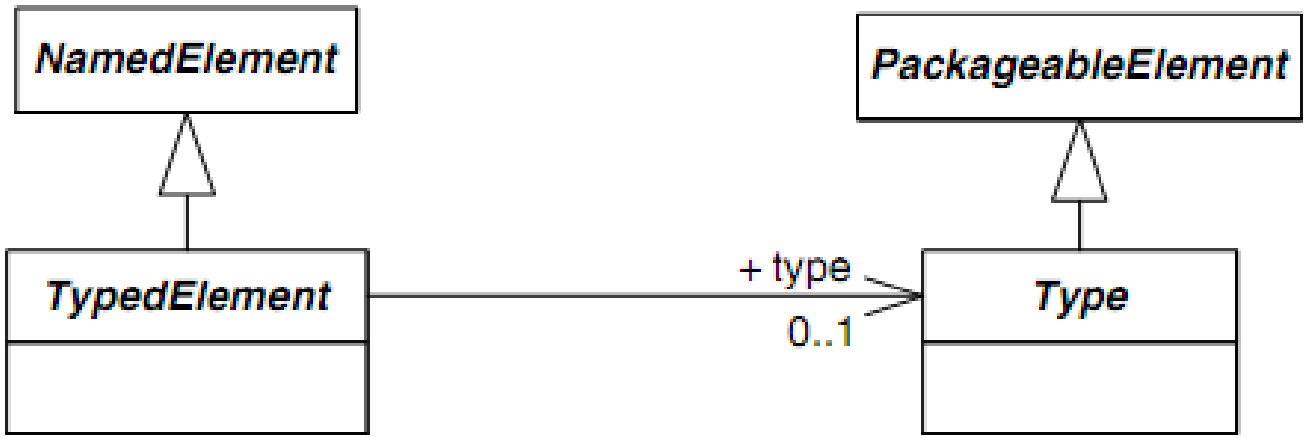


FIGURE 2.18 The metamodel for typed elements.

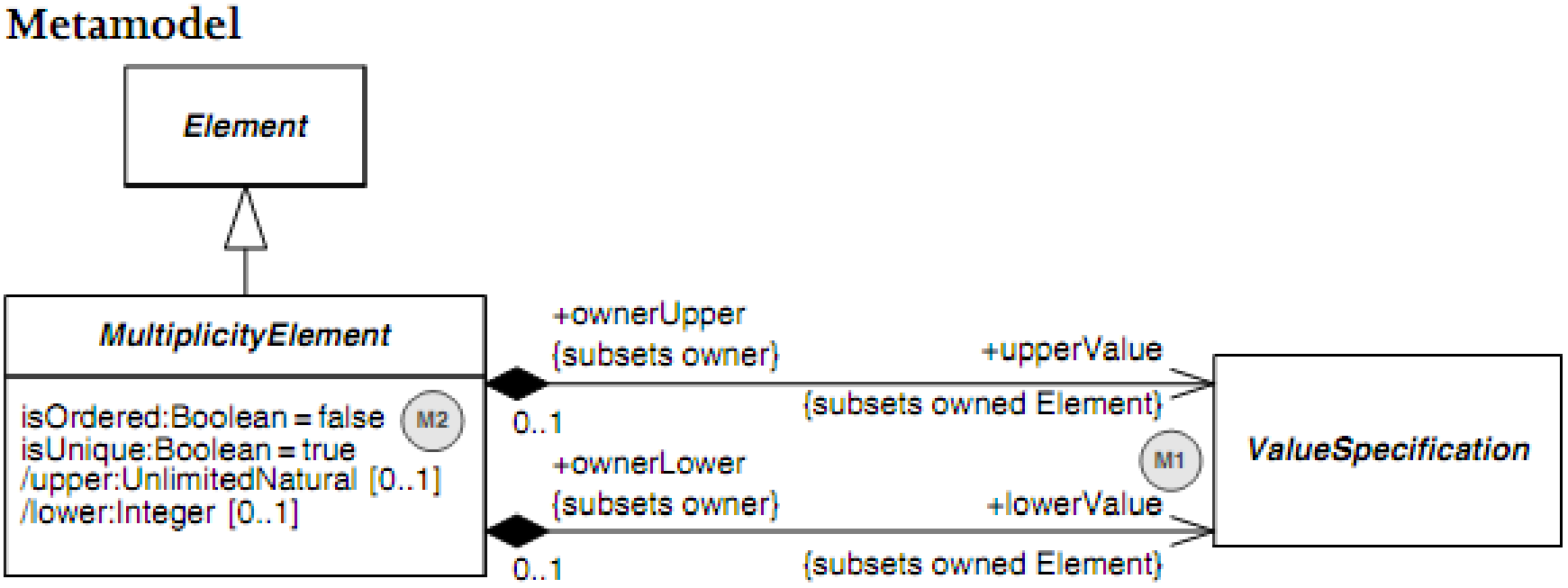


FIGURE 2.20 The multiplicity metamodel.

Metamodel

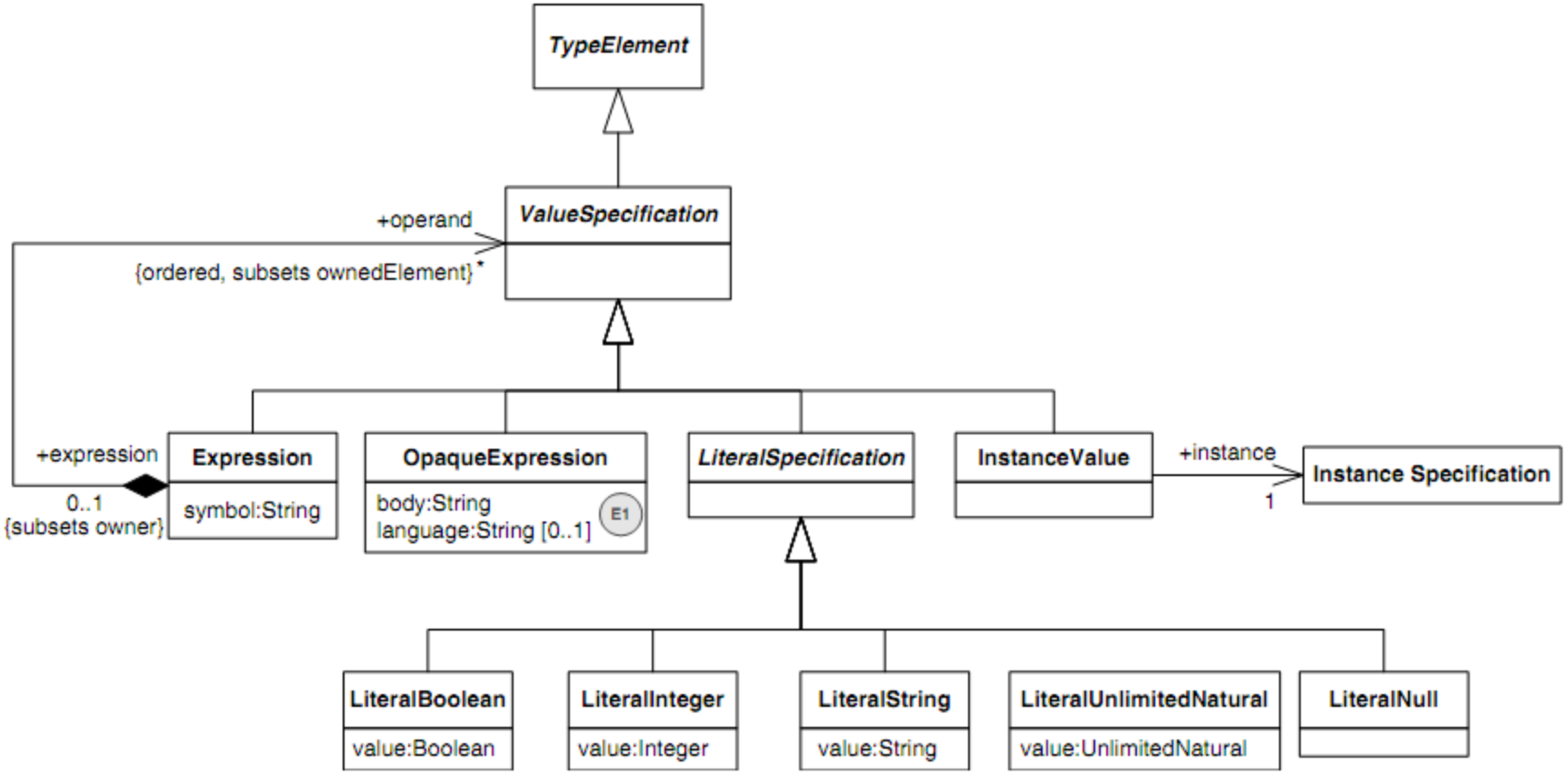


FIGURE 2.23 The metamodel for value specifications.

Metamodel

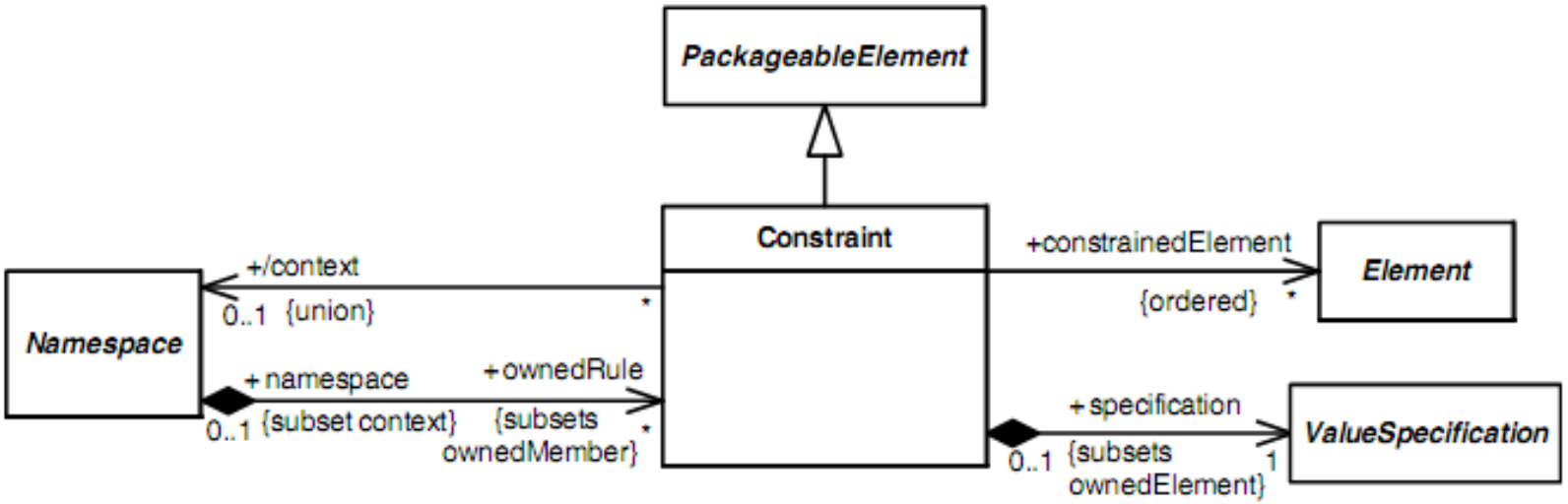


FIGURE 2.25 The metamodel for constraints.

Metamodel

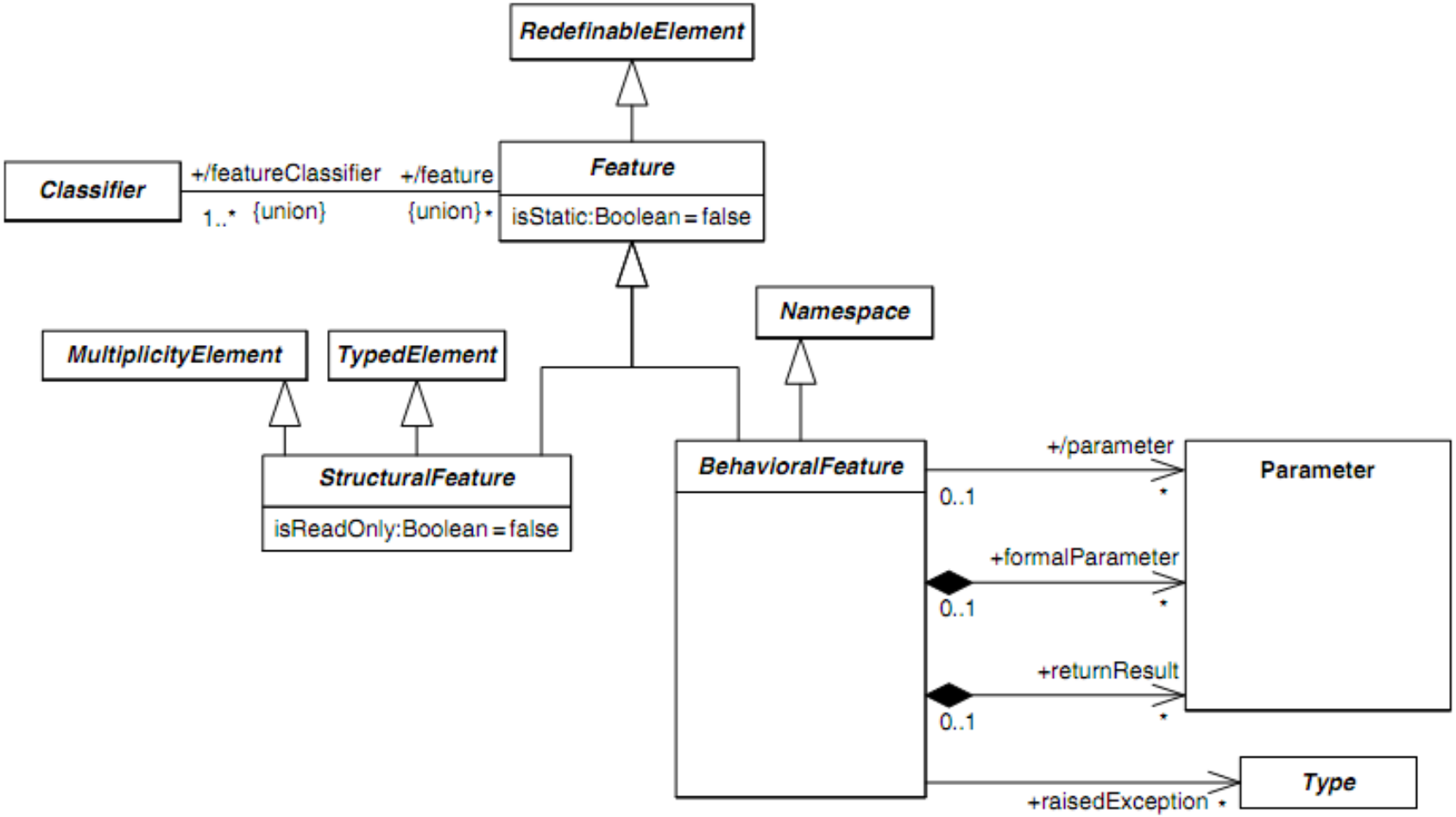


FIGURE 2.33 The metamodel for features.

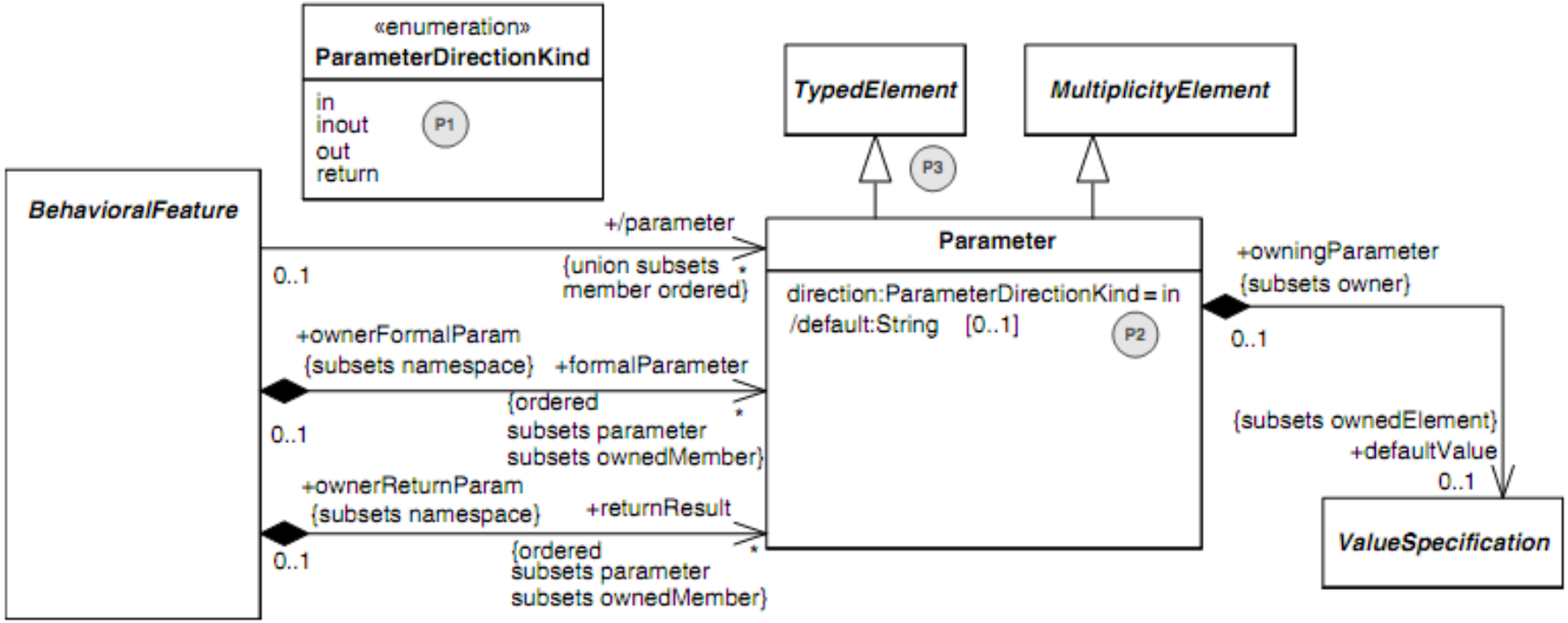


FIGURE 2.34 The metamodel for parameters.

Metamodel

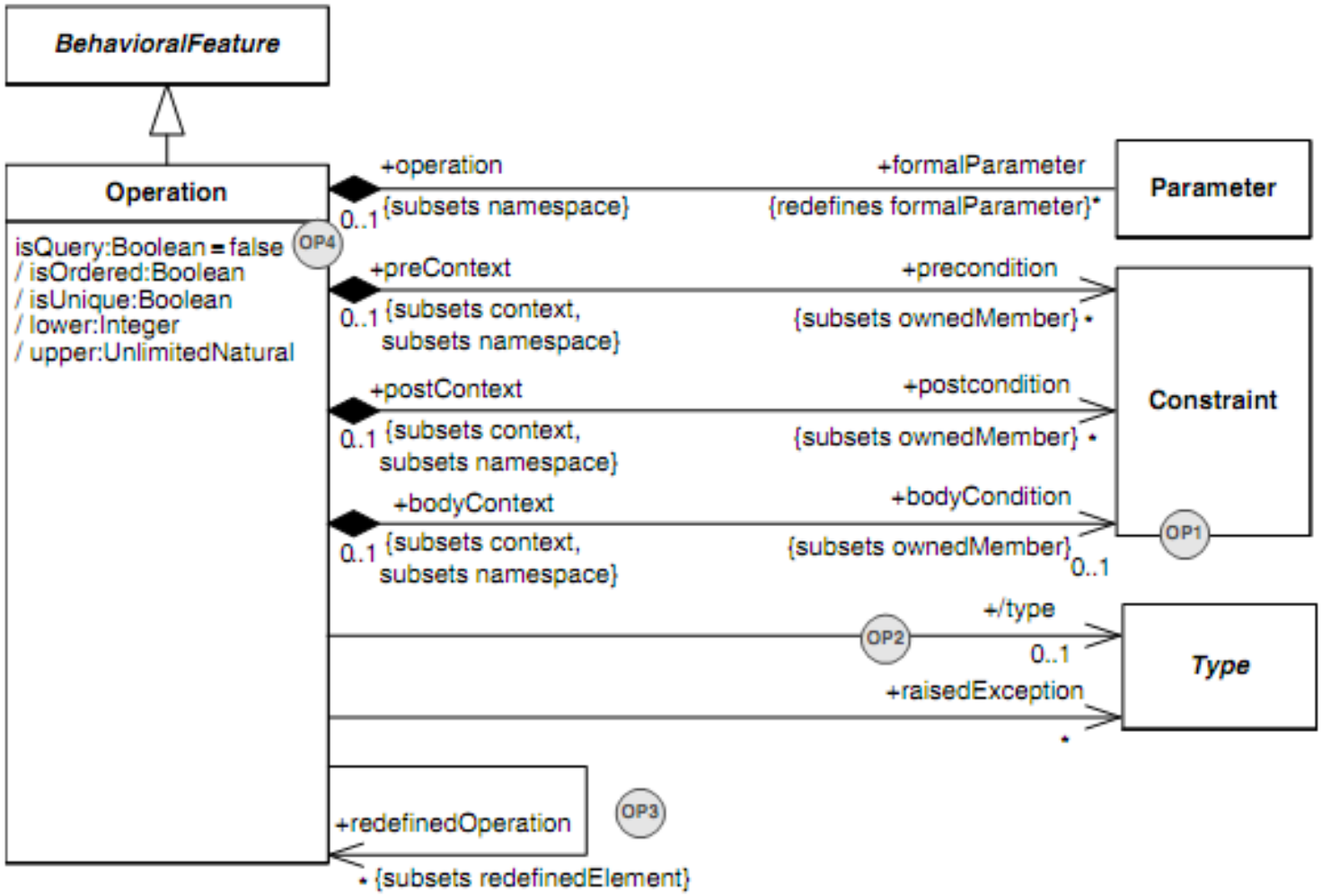


FIGURE 2.35 The metamodel for operations.

Metamodel

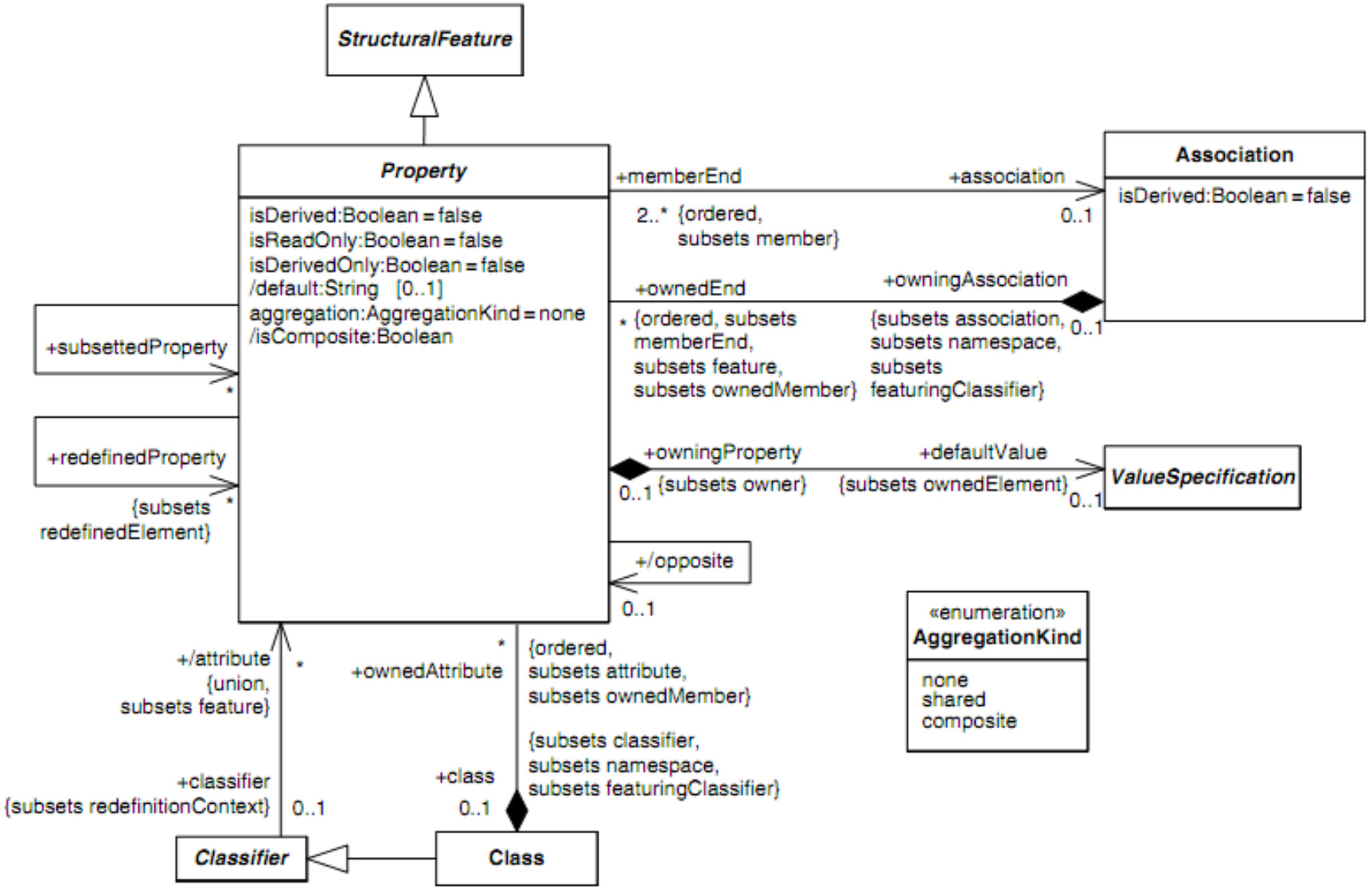


FIGURE 2.39 The metamodel for properties.

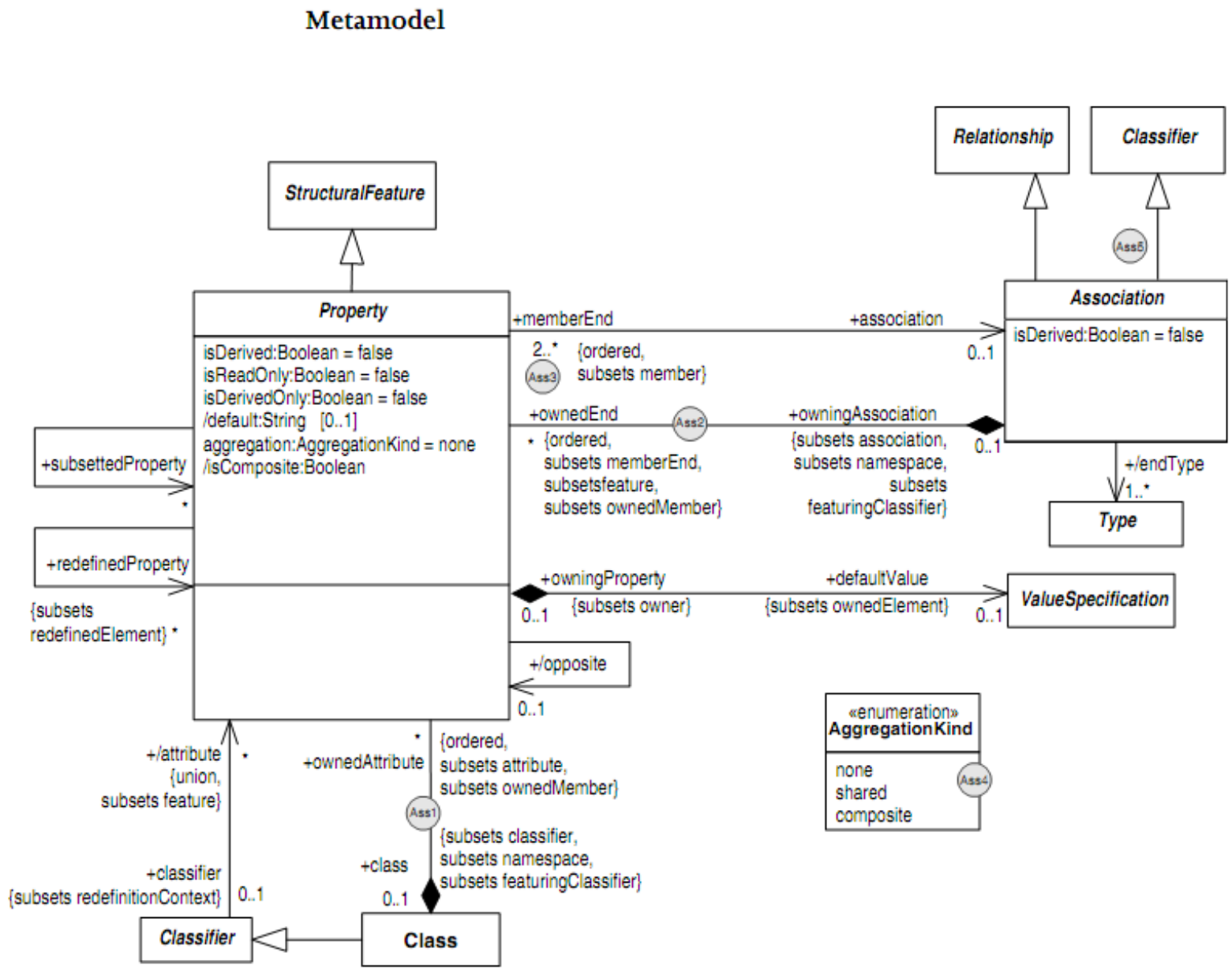


FIGURE 2.51 The metamodel for associations.

Metamodel

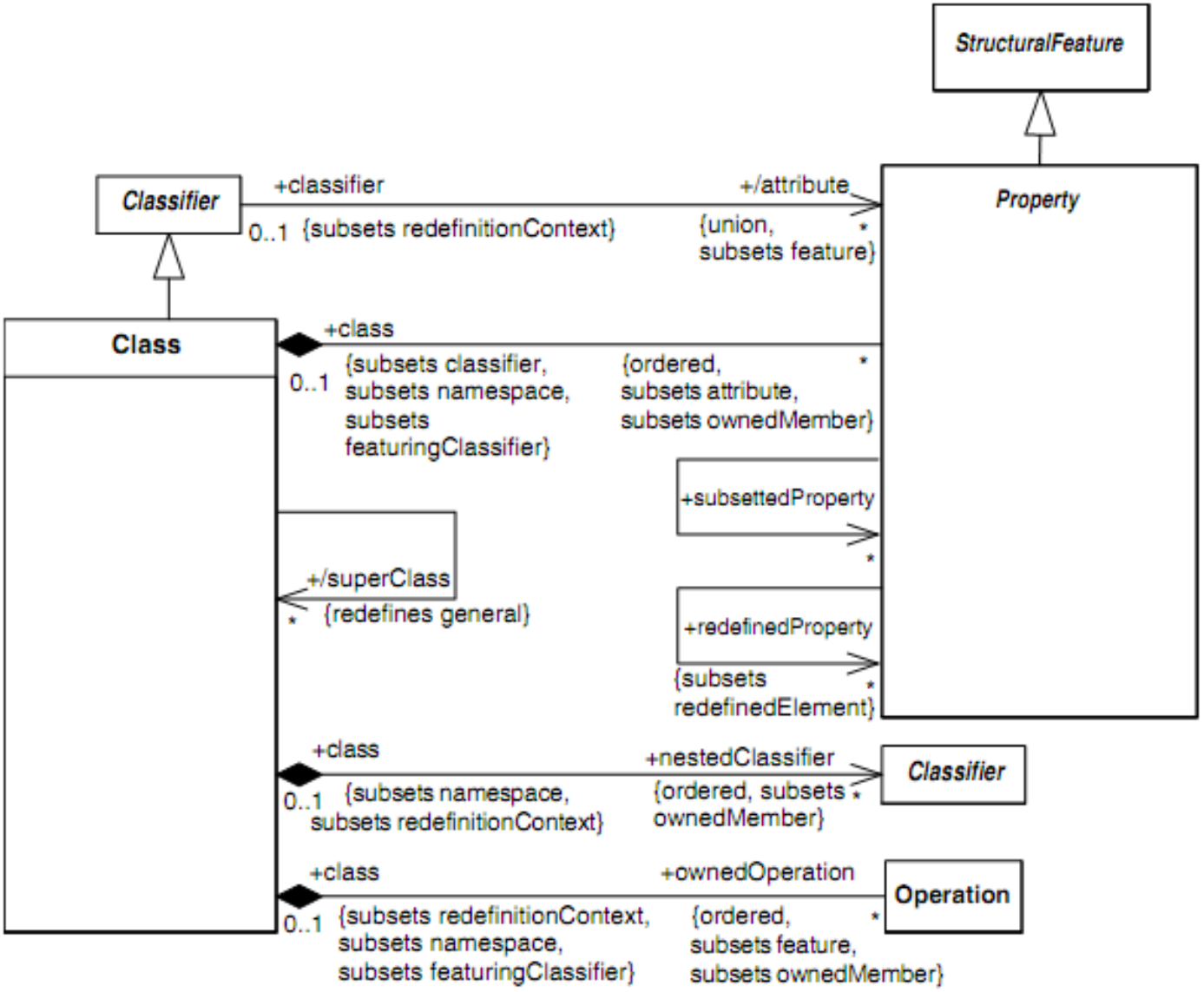


FIGURE 2.53 The metamodel for Class.

Metamodel

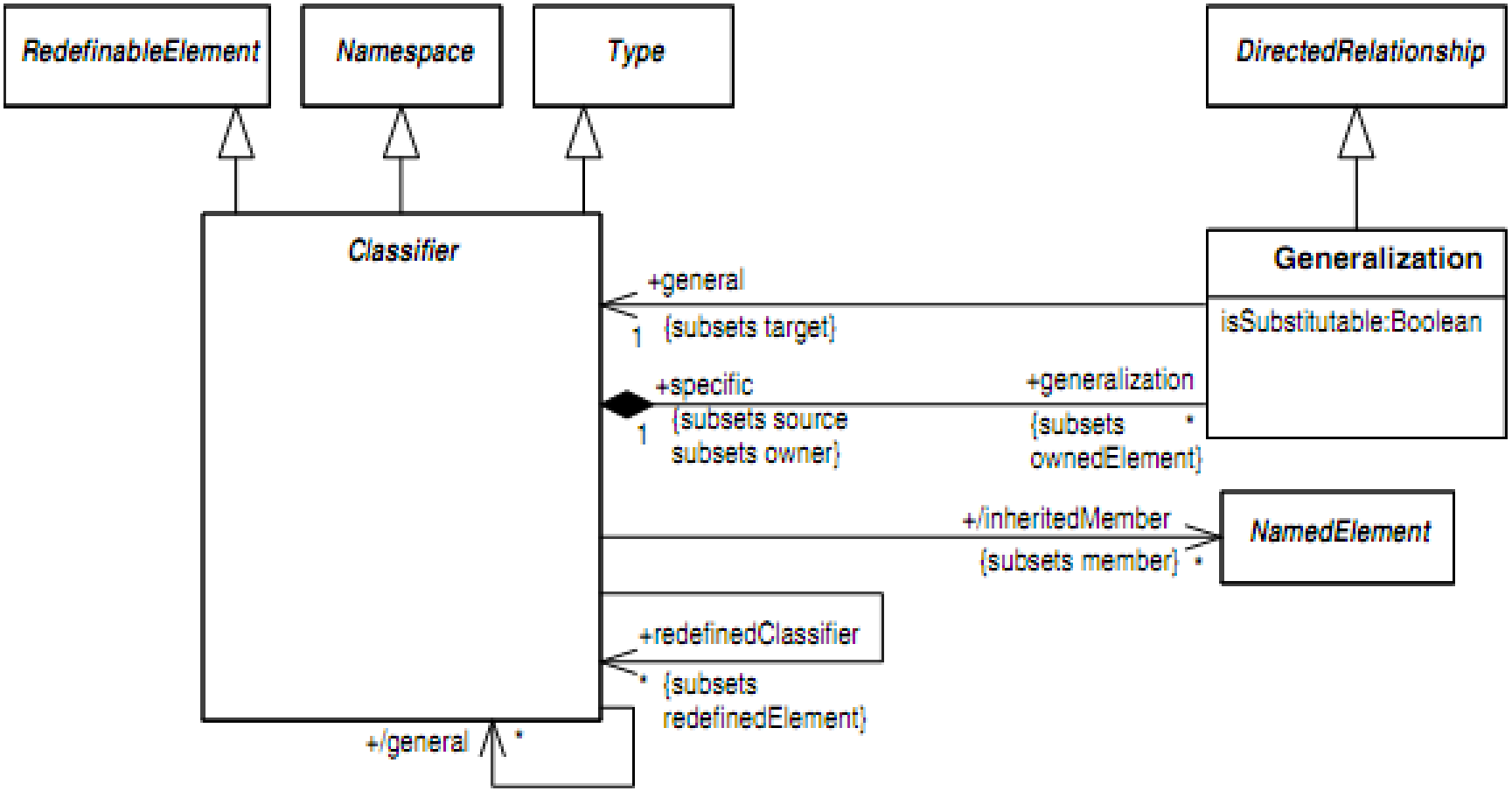


FIGURE 2.56 The metamodel for generalizations.

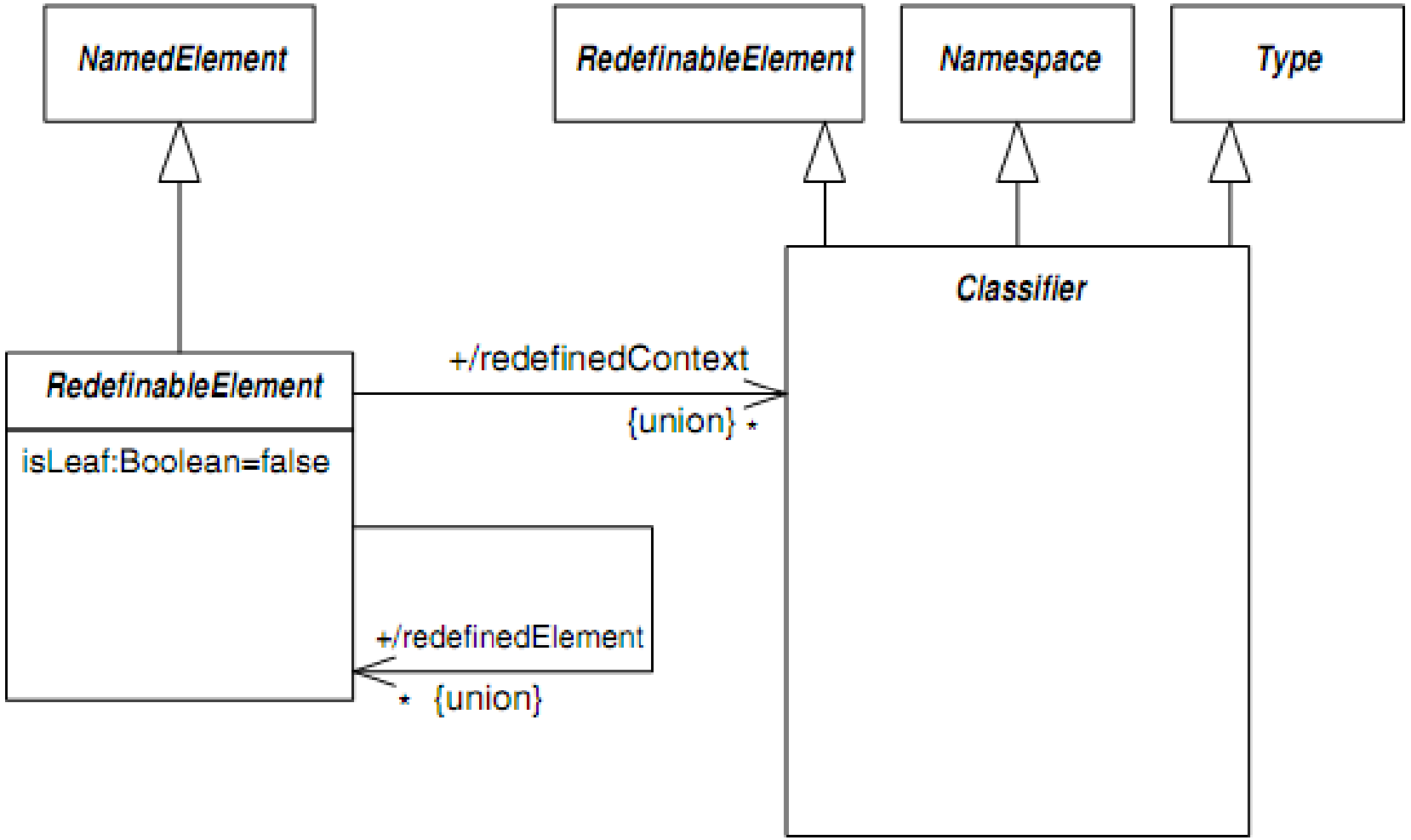


FIGURE 2.57 The metamodel for redefinable elements.

Metamodel

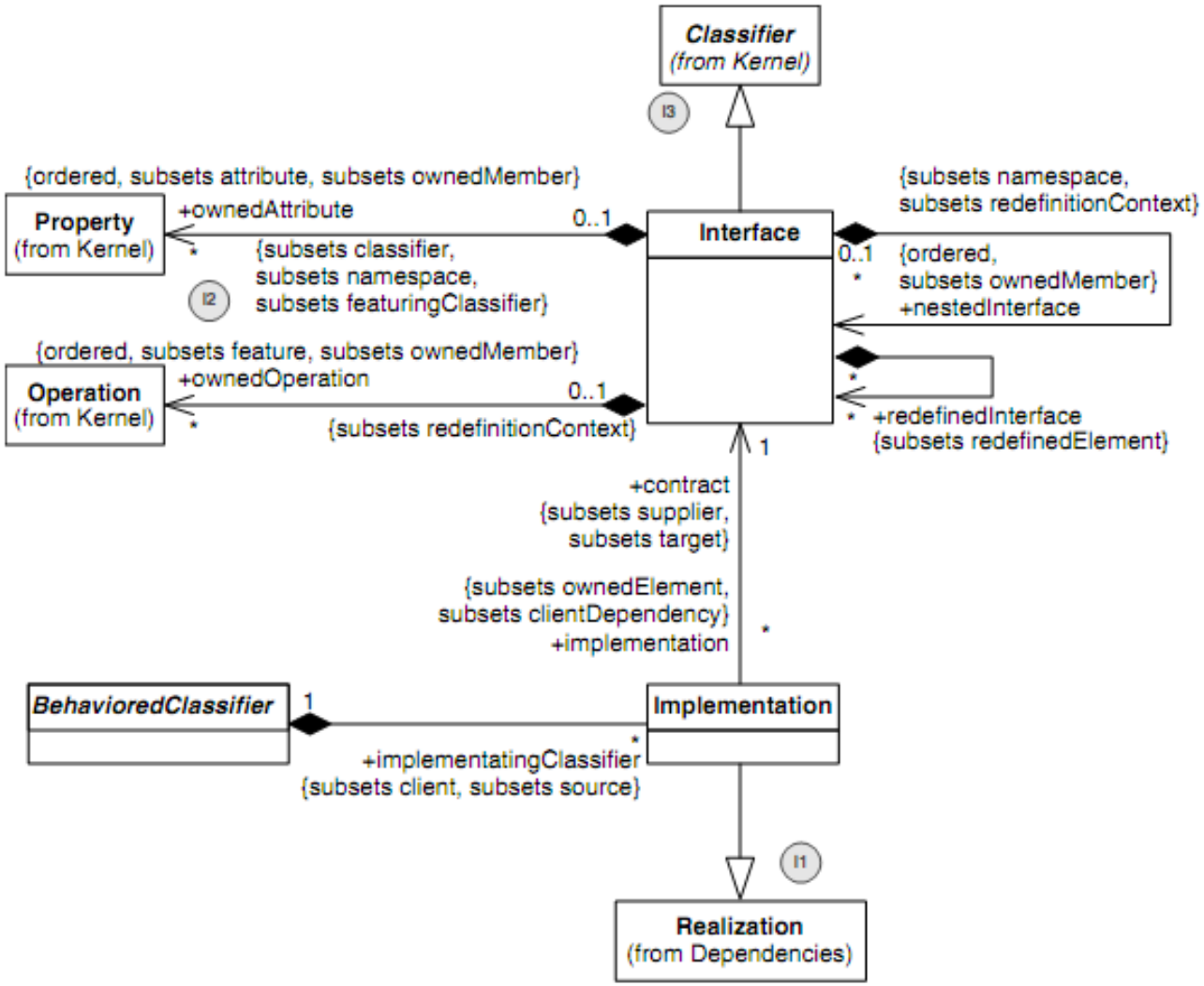


FIGURE 2.79 The metamodel for interfaces.