INTRODUCCIÓN AL DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS, Y AL META-MODELADO

Desarrollo Automatizado de Software

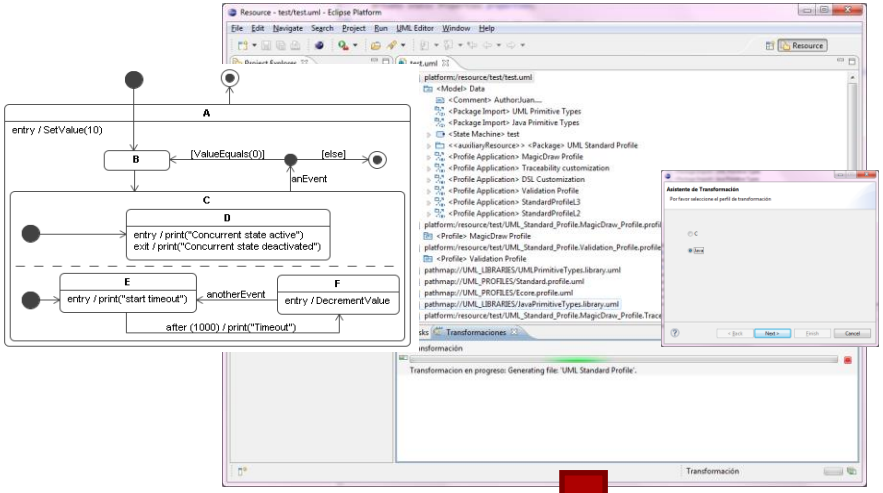
4º Ingeniería Informática

Universidad Autónoma de Madrid

Desarrollo de Sw Dirigido por Modelos

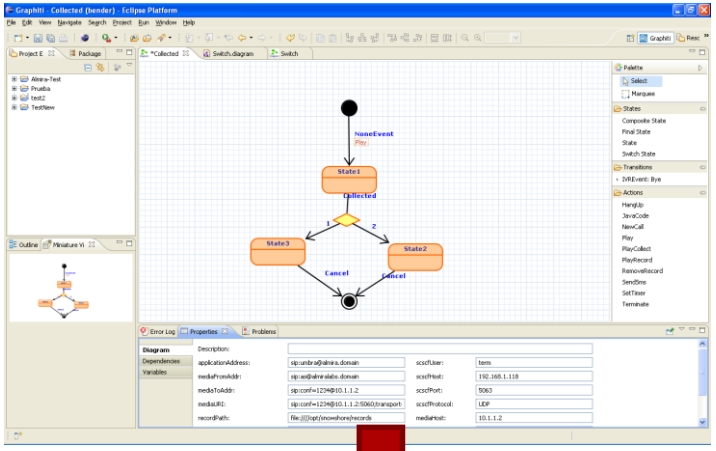
- ¿Objetivos?
  - desarrollar software de mayor calidad más rápido
  - evitar codificar la mismas soluciones una y otra vez
- ¿Cómo?
  - elevando el nivel de abstracción: modelos
  - automatización: los modelos no son sólo documentación, de ellos se genera código para parte o toda la aplicación final
  - implica diseñar lenguajes de dominio específico (textuales o visuales)
  - menos detalles “accidentales”, notaciones más cercanas al problema
- Reutilización: lenguaje de dominio específico + arquitectura + generación de código (o configuración)

Ejemplos



generador de código

generación de código desde máquinas de estado para software de señalización ferroviaria



generador de código

modelado, validación y generación de código para servicio de telefonía

Lenguajes de dominio específico

- El Desarrollo de Software Dirigido por Modelos (DSDM) se basa en el concepto de lenguaje de dominio específico
  - lenguajes orientados a un dominio de aplicación particular
  - primitivas del lenguaje de alto nivel, expresivas

- Premisa: mejoran la productividad comparado con usar lenguajes de propósito general

- En dominios software restringidos, es posible generar el 100% del código de la aplicación

Lenguajes de dominio específico

Lenguajes de Dominio Específico

espacio del problema

- menor distancia semántica
- orientados a la forma de pensar del usuario

transformación (semi-)automática

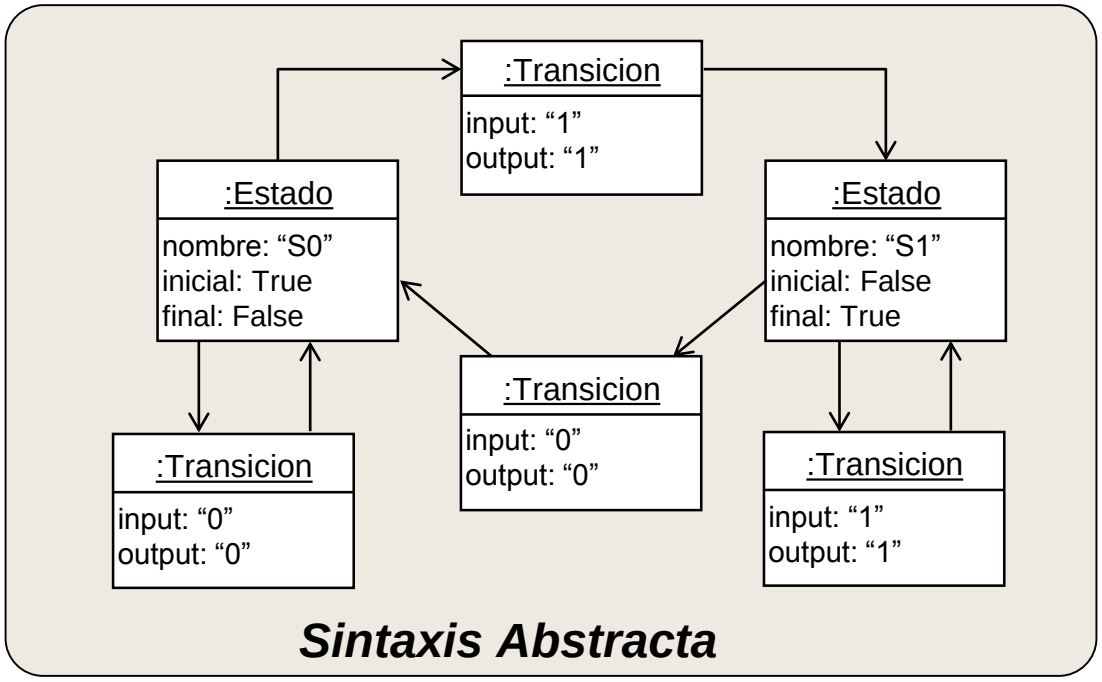
espacio de la solución

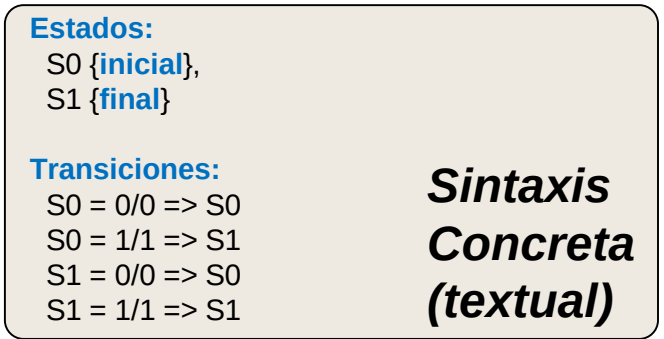
Lenguajes de Propósito General

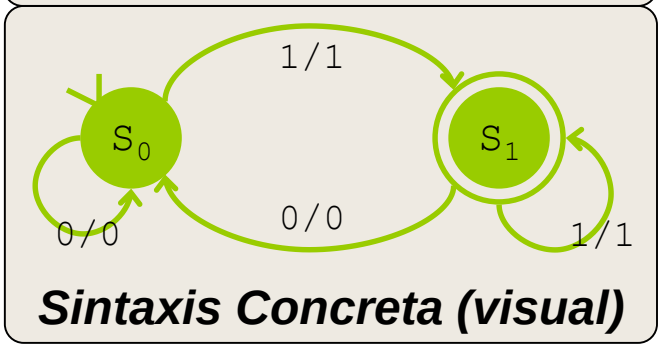
- necesario transformar a un dominio técnico
- orientados a la forma de pensar del desarrollador

Definiendo lenguajes de dominio específico

- **Sintaxis abstracta**: conceptos, relaciones y atributos (meta-modelado)
- **Semántica estática:** otras restricciones (lenguaje de restricciones, ej. OCL)
- **Sintaxis concreta**: visualización de los elementos de la sintaxis abstracta
- **Semántica operacional** (simulador) / **semántica denotacional** (otro lenguaje)



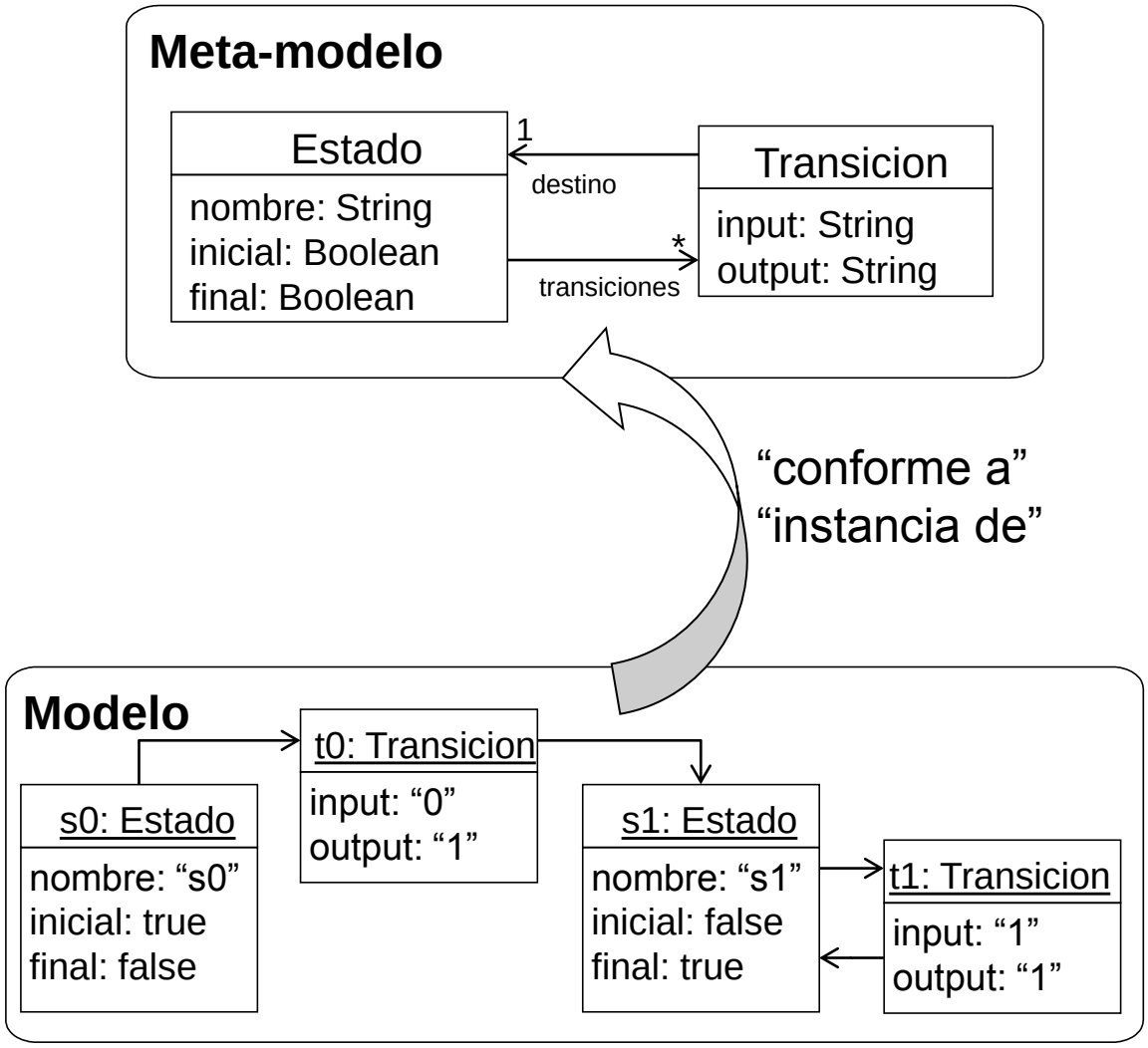




Meta-modelado

- Un modelo es una descripción de un sistema, usando para ello un lenguaje (de modelado)
- Un meta-modelo es un modelo que describe un lenguaje, i.e., describe todos los modelos que son válidos sintácticamente
- Un meta-modelo describe la sintaxis abstracta de un lenguaje
- Los meta-modelos suelen definirse mediante diagramas de clases, e incluyen restricciones OCL adicionales

Ejemplo de meta-modelo y modelo

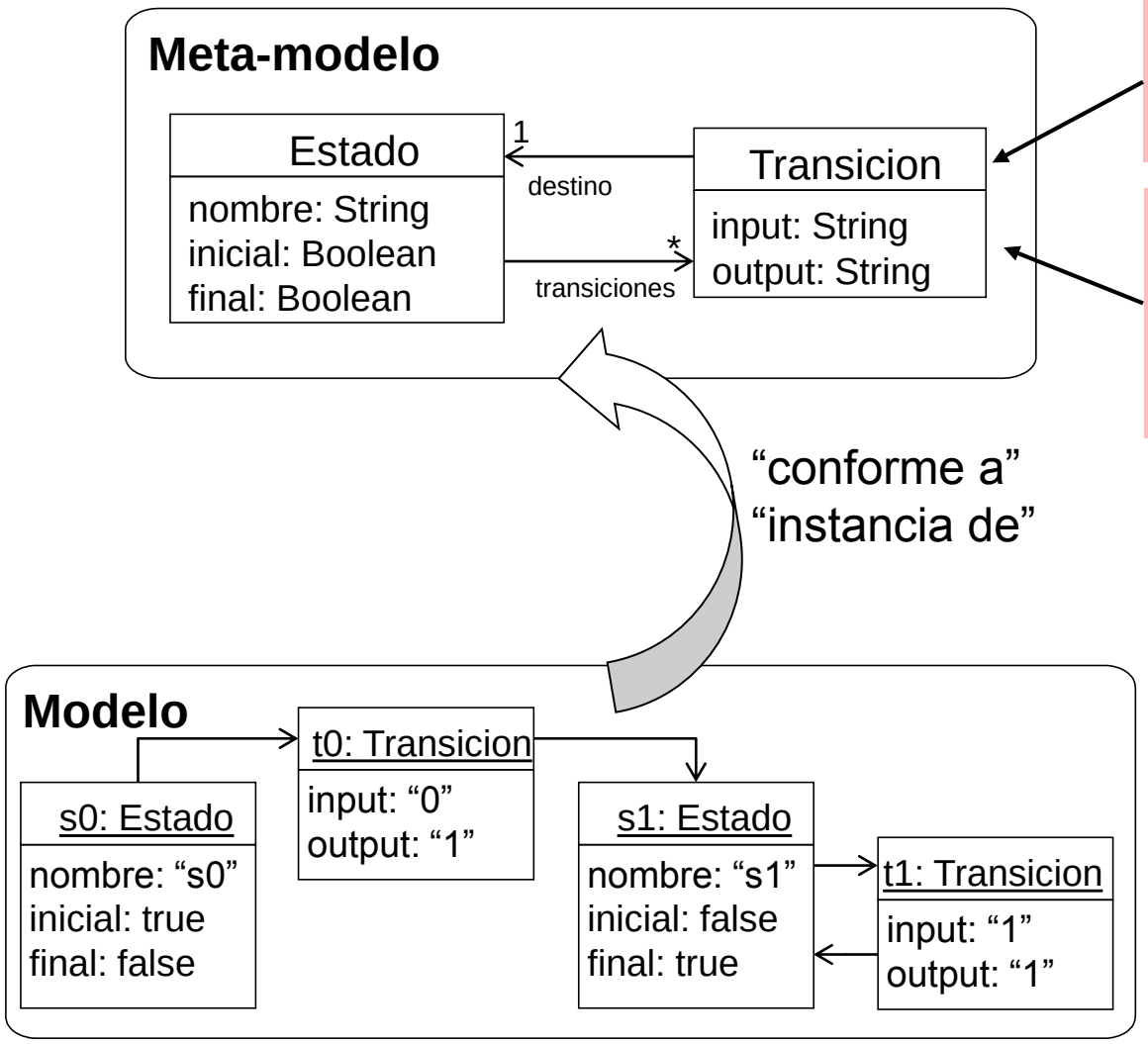


Un modelo es una instancia válida de un meta-modelo si:

- El modelo es estructuralmente válido:
  - Los objetos y enlaces del modelo son instancias de las clases y asociaciones del meta-modelo, respectivamente
- El modelo satisface las siguientes restricciones:
  - cardinalidad en asociaciones
  - restricciones OCL adicionales

(similar a la relación entre un diagrama de clases y un diagrama de objetos)

Ejemplo de meta-modelo y modelo



¿Cómo expresamos que los autómatas deben ser deterministas?

¿Cómo expresamos que debe haber exactamente un estado inicial, y al menos un estado final?

cia

o si:

s

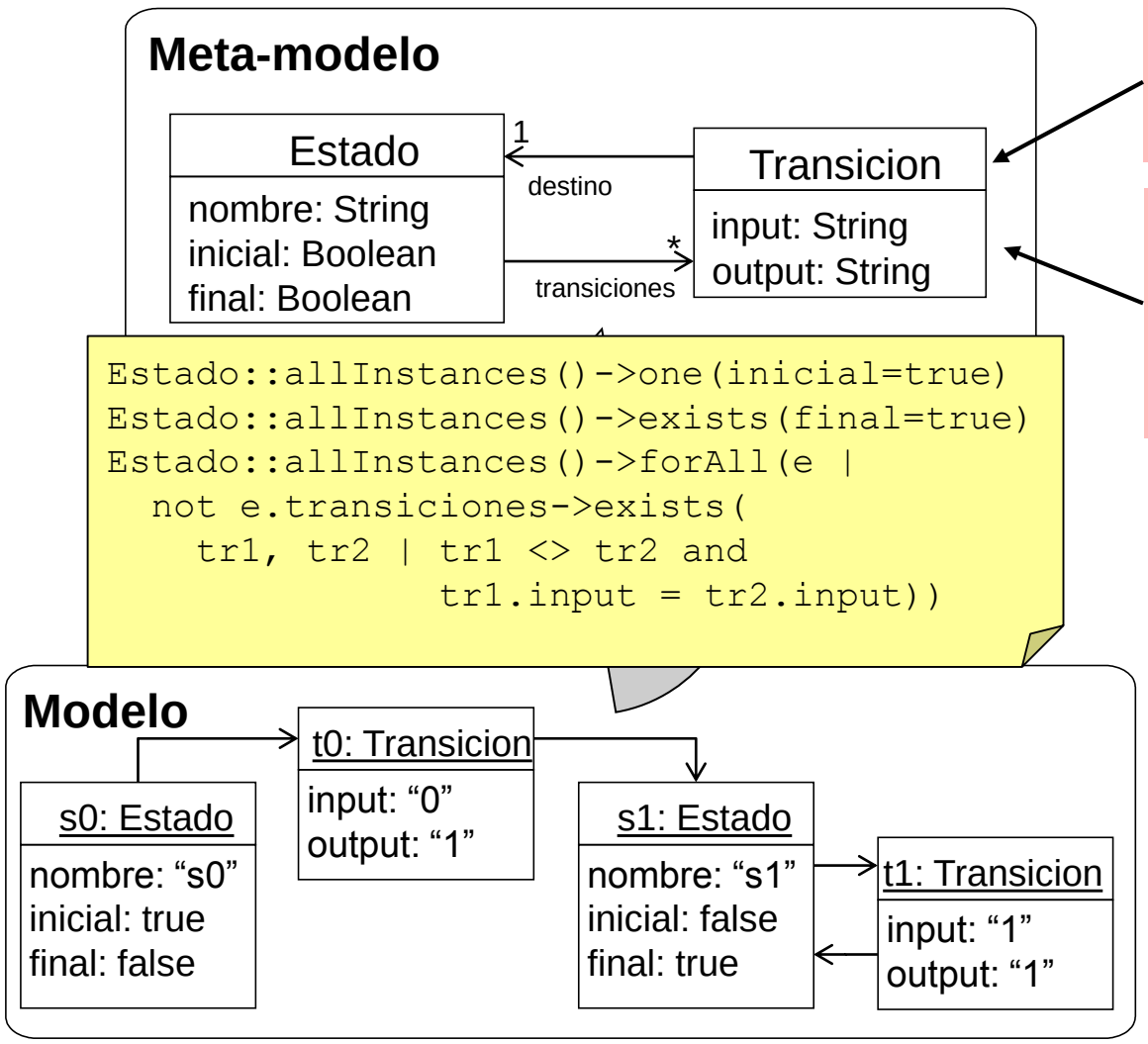
eta-

modelo, respectivamente

- El modelo satisface las siguientes restricciones:
  - cardinalidad en asociaciones
  - restricciones OCL adicionales

(similar a la relación entre un diagrama de clases y un diagrama de objetos)

Ejemplo de meta-modelo y modelo



¿Cómo expresamos que los autómatas deben ser deterministas?

¿Cómo expresamos que debe haber exactamente un estado inicial, y al menos un estado final?

cia

o si:

s

eta-

modelo, respectivamente

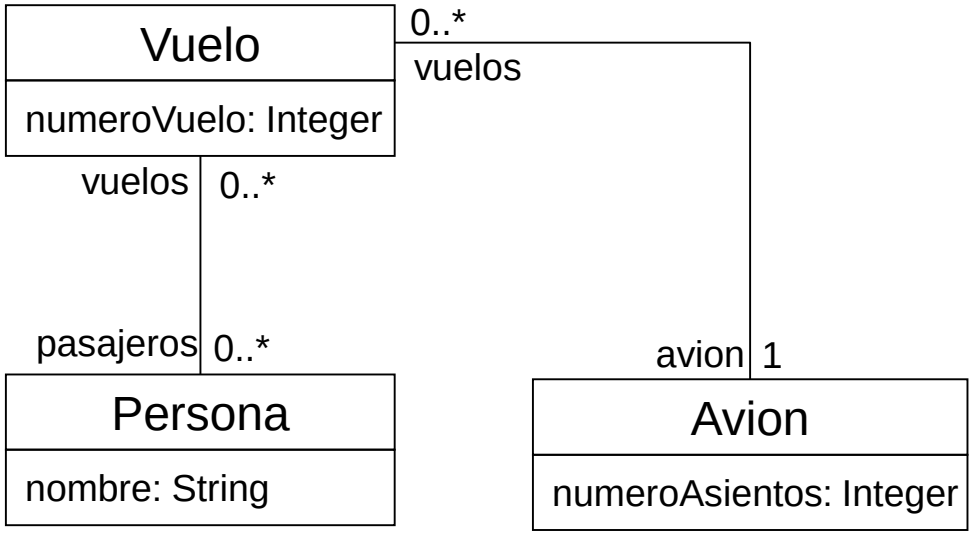
- El modelo satisface las siguientes restricciones:
  - cardinalidad en asociaciones
  - restricciones OCL adicionales

(similar a la relación entre un diagrama de clases y un diagrama de objetos)

OCL: Object Constraint Language

- Lenguaje de restricciones para expresar condiciones que no pueden expresarse con diagramas y cardinalidades
- Lenguaje preciso, no ambiguo, declarativo, tipado, basado en matemáticas (lógica de predicados y teoría de conjuntos)
- Útil en DSDM: definición precisa de los modelos (no comentarios en lenguaje natural)
- Meta-modelo: diagrama UML + restricciones OCL textuales

Ejemplo



Una restricción OCL:

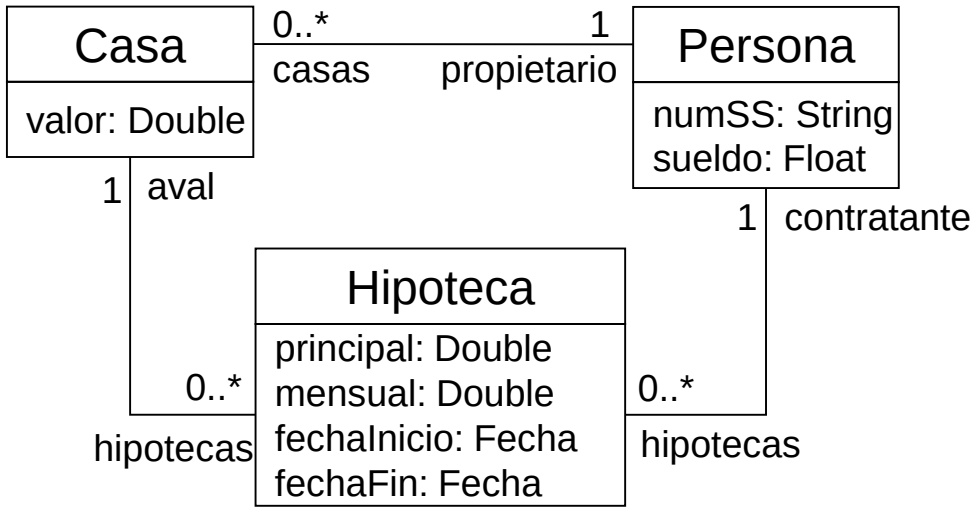
- se define en el contexto de un tipo específico
- se evalúa sobre todas las instancias de ese tipo

- ¿Cómo expresamos el hecho de que en ningún vuelo puede haber más pasajeros que asientos tiene el avión?

  Restricción OCL:

```
context Vuelo
inv: pasajeros->size() <= avion.numeroAsientos
```

Ejemplo



- Una persona puede tener una hipoteca sobre una casa sólo si es el propietario:

```
context Hipoteca
inv: aval.propietario = contratante
```

- La fecha de inicio de cada hipoteca debe ser anterior a la final:

```
context Hipoteca
inv: fechaInicio < fechaFin
```

- El número de la seguridad social de cada persona es único:

```
context Persona
inv: Persona.allInstances()->isUnique(numSS)
```

OCL: operaciones sobre colecciones

Tipos de colecciones: Set, OrderedSet, Bag, Sequence

`select(expr):` selecciona los elementos que cumplen una condición

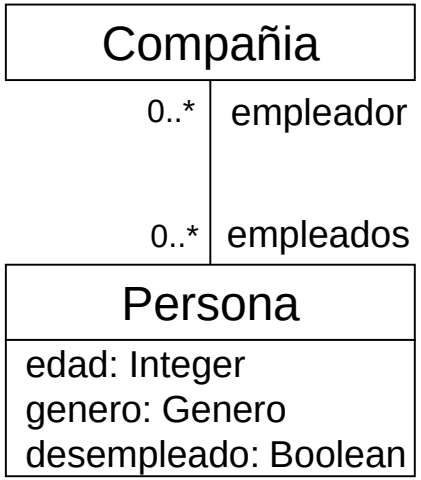
colección->select(expresión-lógica)
colección->select(var | expresión-lógica-con-var)
colección->select(var : Tipo | expresión-lógica-con-var)

```
context Compañia inv:
self.empleados->select(edad<18)->isEmpty()
```

```
context Compañia inv:
self.empleados->select(genero=mujer)->notEmpty()
```

`collect(expr):` devuelve la colección que resulta de evaluar `expr` sobre cada elemento de la colección fuente

```
empleados->collect(edad)->asSet()
```



OCL: operaciones sobre colecciones

`exists(expr):` *true* si algún elemento de la colección cumple `expr`

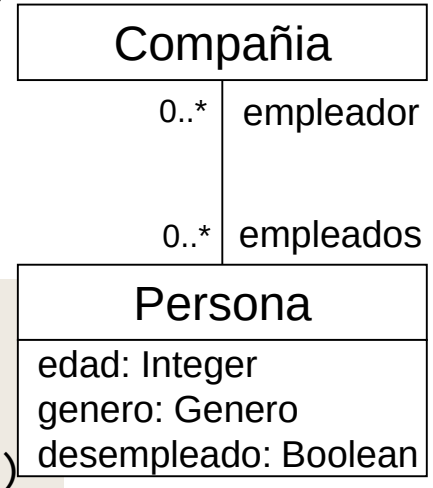
colección->exists(expresión-lógica)
colección->exists(var | expresión-lógica-con-var)
colección->exists(var : Tipo | expresión-lógica-con-var)

```
context Compañia
inv: empleados->exists(edad>50)
inv: empleados->exists(p | p.edad>50)
inv: empleados->exists(p:Persona | p.edad>50)
```

`forAll(expr):` *true* si `expr` es cierto para cada elemento de la colección

colección->forAll(expresión-lógica)
colección->forAll(var | expresión-lógica-con-var)
colección->forAll(var : Tipo | expresión-lógica-con-var)

```
context Compañia
inv: empleados->forAll(desempleado=false)
inv: empleados->forAll(p | p.desempleado=false)
inv: empleados->forAll(p:Persona | p.desempleado=false)
```



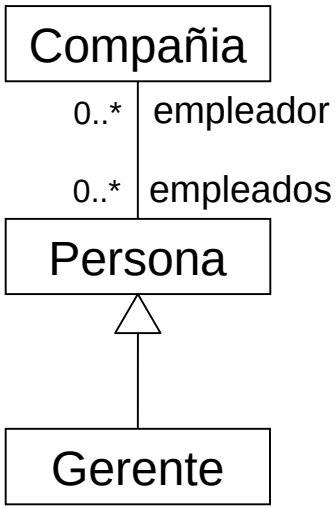
OCL: operaciones sobre colecciones

`any(expr)`: *true* si algún elemento de la colección cumple `expr`
`count(objeto)`: número de veces que el objeto está en la colección
`excludes(objeto)`: *true* si la colección no contiene el objeto
`excluding(objeto)`: copia de la colección que no contiene el objeto
`includes(objeto)`: *true* si la colección contiene el objeto
`isEmpty()`: *true* si la colección no contiene elementos
`notEmpty()`: *true* si la colección contiene elementos
`one(expr)`: *true* si sólo un elemento de la colección satisface `expr`
`reject(expr)`: selecciona los elementos que no cumplen `expr`
`size()`: número de elementos de la colección

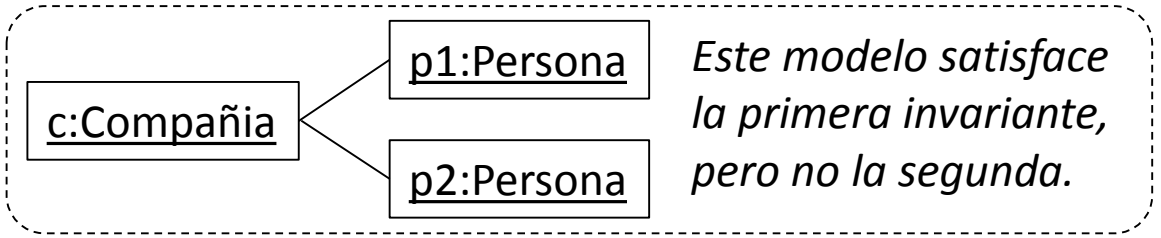
Listado completo de operadores, con ejemplos, aquí.

OCL: operaciones sobre elementos

`oclIsKindOf(t):` *true* si el tipo del elemento es `t` o un supertipo de `t`
`oclIsTypeOf(t):` *true* si el tipo del elemento es `t`
`oclIsUndefined():` *true* si elemento es inválido o nulo




```
context Compañia
inv: empleados->exists(p | p.oclIsKindOf(Gerente))
inv: empleados->exists(p | p.oclIsTypeOf(Gerente))
```



Listado completo de operadores, con ejemplos, aquí.

Bibliografía

- Desarrollo de Software Dirigido por Modelos. Völter Stahl. Wiley (2006)

- Model-Driven Software Engineering in Practice (Synthesis Lectures on Software Engineering). Marco Brambilla, Jordi Cabot & Manuel Wimmer. Morgan & Claypool Publishers, 1st Edition (2012)

- Web de la OMG sobre UML: http://www.uml.org
- Web de la OMG sobre OCL: http://www.omg.org/spec/OCL/2.3.1/PDF

- The Object Constraint Language: Getting your Models Ready for MDA. Warmer, Kleppe. Addison-Wesley Professional (2003)