

**MÉTODOS FORMALES EN  
INGENIERÍA DEL SOFTWARE**

**Capítulo 5**

**El lenguaje de restricciones y consultas  
OCL**

**Grado en Ingeniería Informática**

*Espec. Ingeniería del Software*



DEPARTAMENTO DE  
Tecnologías de la  
Información

ETSI  
INGENIERÍA  
INFORMÁTICA

# Índice

- 1. OBJETIVOS
- 2. INTRODUCCIÓN
- 3. OCL:
  - a. Self
  - b. Restricciones
  - c. Precondiciones/Postcondiciones
  - d. Restricciones y Herencia
- 4. TIPOS EN LAS OPERACIONES OCL
- 5. OBJETOS Y PROPIEDADES
- 6. COLECCIONES

EXTENSIONES

BIBLIOGRAFIA

Prof. Ana M<sup>a</sup> Roldán



DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

**1. OBJETIVOS**

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

- Entender la necesidad de utilizar un lenguaje de especificación de restricciones en el modelado estático.
- Utilizar OCL para especificar la semántica de sistemas especificados en UML, evitando estados del sistema incoherentes.
- Otros usos interesantes de OCL.
- Principales herramientas existentes.

Prof. Ana M<sup>a</sup> Roldán

(3)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

**2. INTRODUCCIÓN**

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

- Los modelos que describen los sistemas (hardware, software,..) han de ser **PRECISOS** y **COMPLETOS**:
- **PRECISOS:** Interpretación **sin ambigüedad**, es decir, otros usuarios -incluidos otros programas- han de entender lo mismo.
- **COMPLETOS:** Contienen **todos los detalles** necesarios para representar el sistema desde un punto de vista adecuado.

Prof. Ana M<sup>a</sup> Roldán

(4)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
AUTÓNOMA  
DE MÉXICO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

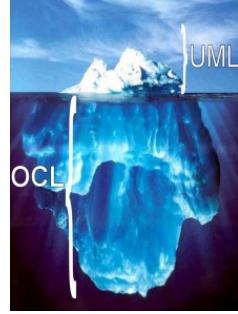
## 2. Introducción

**UML es :**

- Una **notación gráfica** muy popular para modelar.
- Es **visual** y **atractivo** (*ventajas*).
- **No posee suficiente expresividad** para describir toda la información que debe contener un modelo (*desventaja*) → **Solución:**

↓

**OCL**



Prof. Ana M<sup>a</sup> Roldán

(5)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
AUTÓNOMA  
DE MÉXICO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL (Object Constraint Language)

- **Extiende** el Lenguaje de Modelado Unificado (UML).
- **Es** un Lenguaje Formal\* para la definición de restricciones y consultas sobre modelos UML.
- **No es** un lenguaje de programación: *no se puede escribir la lógica del programa ni se pueden invocar procesos o activar operaciones que no sean sólo de consulta.*
- **Es declarativo y tipado**, es decir, *todas las expresiones OCL tienen tipo, y tienen que seguir las reglas del lenguaje* (p.e. cada clase definida en UML representa un tipo distinto en OCL). Además OCL incluye un conjunto de tipos predefinidos.
- **Está libre de efectos laterales**: cuando se evalúa una expresión OCL simplemente se devuelve un mensaje, *no cambia nada en el modelo*.
- **Añade** semántica precisa a modelos visuales.

Prof. Ana M<sup>a</sup> Roldán

(6)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INVESTIGACIONES  
y DESARROLLO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL (Object Constraint Language)

- Se considera que la **evaluación** de las expresiones OCL es **instantánea**.
- Es aceptado por la comunidad.
- Existen **diferentes extensiones**, p.e., para especificar restricciones temporales.
- Es un lenguaje base de otros lenguajes de la OMG (QVT, PRR) --

Prof. Ana M<sup>a</sup> Roldán

(7)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INVESTIGACIONES  
y DESARROLLO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

**Se puede usar como:**

1. Lenguaje de consulta
2. Para especificar:
  - invariantes sobre las clases y los tipos de un modelo de clases,
  - restricciones sobre operaciones:
    - pre-y postcondiciones de operaciones y métodos,
  - el destino de mensajes y acciones y
  - reglas de derivación para atributos de cualquier expresión en un modelo UML.

Prof. Ana M<sup>a</sup> Roldán

(8)

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN ETSI INDUSTRIAL UNED

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

- Todas las expresiones OCL están escritas en el *contexto de una instancia de un tipo específico*.
- Se usa la palabra reservada **self** para referirse a la instancia que hace de contexto.
- Ejemplo:  
-si el contexto es Empresa (ver diagrama), self se refiere a una instancia de empresa

Si el contexto está asociado a una **Clase** (por ej. Cliente, Factura, etc.) **self** se asocia a cada una de las instancias de la clase.

Si el contexto está asociado a **Propiedad multi-valuada** (por ej. clientes, ventas, productos, items, etc.) **self** se refiere a la colección apuntada por la relación

Si el contexto está asociado a una **Propiedad simple** (por ej. destinatario, librería, producto, etc.) **self** se refiere a la **instancia** del objeto apuntados.

9

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN ETSI INDUSTRIAL UNED

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### **RESTRICCIÓN (constraint)**

- Condiciona uno o más valores de o parte de un modelo o sistema orientado a objetos.
- Una restricción **se formula al nivel de las clases**, pero su semántica se aplica al nivel de los objetos
- Se realiza sobre modelos UML.

Prof. Ana M<sup>a</sup> Roldán

10

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### a. Invariantes

- **Es** una restricción que *debería ser cierta para un objeto durante toda su vida.*
- Los invariantes **representan** reglas que tienen que cumplir todas las instancias de la clase que actúa de contexto de la expresión.

#### ▪ Sintaxis

```
context <classifier> inv:  
[<constraint name>]:<Boolean> OCL expression>
```

Prof. Ana M<sup>a</sup> Roldán

(11)

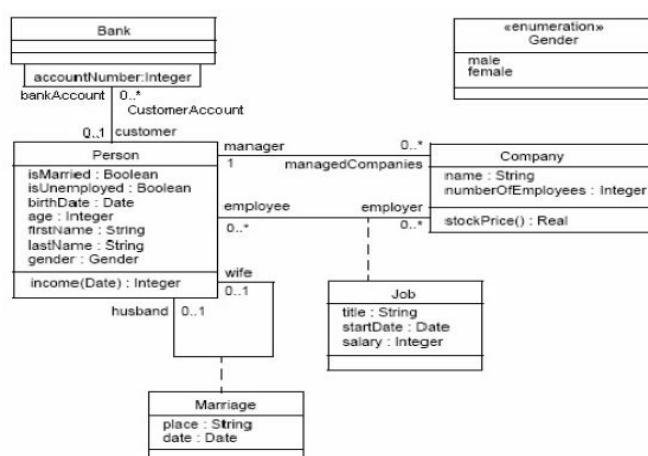
DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### Ejemplo de trabajo



Prof. Ana M<sup>a</sup> Roldán

(12)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### a. Invariantes

- **Sintaxis**

```
context <classifier> inv:  
[<constraint name>]:<Boolean> OCL expression>
```

**Ejemplo**

```
context Empresa inv:  
self.numeroDeEmpleados > 50:
```

Nota: si el contexto está claro, se puede eliminar la palabra **self**

Prof. Ana M<sup>a</sup> Roldán

13

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### a. Invariantes

- **También puede darse un nombre distinto a la instancia que se utiliza de ámbito**

```
context e: Empresa inv:  
e.numeroDeEmpleados > 50
```

- **También es posible dar, un nombre al invariante para referirse posteriormente a ella por el nombre**

```
context e: Empresa inv suficientesEmpleados:  
e.numeroDeEmpleados > 50
```

Prof. Ana M<sup>a</sup> Roldán

14

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
AUTÓNOMA  
DE MÉXICO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### a. Invariantes

**Importante**

- **¿Cómo elegir el contexto de un invariante? Según el contexto elegido la expresión puede ser más o menos compleja:**
  - Si la expresión es sobre atributos de una clase, dicha clase debe elegirse como contexto.
  - Si la expresión es sobre atributos de varias clases asociadas, cualquiera de ellas puede elegirse como contexto considerando que ..
  - *Como regla general, cualquier predicado debería minimizar el número de navegaciones sobre las asociaciones. -ver transp.15-*

Prof. Ana M<sup>a</sup> Roldán

15

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
AUTÓNOMA  
DE MÉXICO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### Nota1:

- **Además, ¿cómo navegamos?...avanzando un poco,..**
  - Desde un objeto se puede navegar a través de las asociaciones que hay en el modelo para referenciar a otros objetos y sus propiedades considerando...

nombreDeObjeto nombreDeRol

En el otro extremo  
de la asociación

Persona	director	empresaDirigida	Compañía
tieneMBA:Boolean	1..1	0..*	
0..*	empleado		empresa 0..*

context Compañía inv:  
self.director.tieneMBA=true and self.empleado->notEmpty()

Prof. Ana M<sup>a</sup> Roldán

16

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN 

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

**Nota2:**

**Sobre la elección del contexto:**

```

classDiagram
    Persona "tieneMBA:Boolean" --> "1..1" director
    Persona "0..*" --> "0..*" empleado
    director --> "0..*" empresaDirigida
    empleado --> "0..*" empresa
    class Compania {
        <<Compañía>>
    }
    Compania "0..*" --> "0..*" empleado
    empleado --> "0..*" empresa
  
```

```

context Compañía inv:
    self.director.tieneMBA=true and self.empleado ->notEmpty()
  
```

**equivalentes pero..**

```

context Persona
    self.empresa->forall(c|c.director.estáEmpleado=true and c.empleado ->notEmpty() )
  
```

Prof. Ana M<sup>a</sup> Roldán 

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN 

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

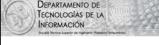
## 3. OCL

**Ejemplo: Reuniones de Trabajo**

```

classDiagram
    class Person {
        name : String
        age : Integer
        title : String
        gender : Gender
    }
    class Date {
        day : Integer
        month : Integer
        year : Integer
    }
    class TeamMember {
        role : String
        numMeetings() : Integer
        numConfMeetings() : Integer
        members : 2..*
        BelongsTo team
    }
    class Meeting {
        title : String
        day : Date
        mstart : Integer
        mend : Integer
        isConfirmed : Boolean
        cancel()
        confirm()
        numParticipants() : Integer
        shift(d : Integer)
        duration() : Integer
        inConflict(m : Meeting) : Boolean
    }
    class Team {
        name : String
        forTeam
    }
    class TeamMeeting {
        * teamMeeting
    }
    <<enumeration>>
    class Gender {
        male
        female
    }
  
```

Prof. Ana M<sup>a</sup> Roldán 

 <b>ETSI</b> INGENIERÍA DE LA INFORMACIÓN	1. Objetivos. 2. Introducción. 3. OCL 4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones. Referencias.
-------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

## 3. OCL

### Reuniones de Trabajo/Invariantes

*Cómo indicamos que la hora de finalización de una reunión tiene que ser mayor que la de inicio?*

```
context Meeting inv: self.mend > self.mstart
```

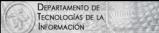
Formulaciones equivalentes:

```
context Meeting inv: mend > mstart
-- "self" se refiere al identificador del objeto en cuyo
-- contexto se evalúa la restricción
```

```
context Meeting inv startEndConstraint:
self.mend > self.mstart
-- Podemos darles nombres a las restricciones
```

Prof. Ana M<sup>a</sup> Roldán

19

 <b>ETSI</b> INGENIERÍA DE LA INFORMACIÓN	1. Objetivos. 2. Introducción. 3. OCL 4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones. Referencias.
---------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

## 3. OCL

### b. Pre y postcondiciones

- Una expresión OCL puede ser parte de la precondición o de la postcondición asociada a una operación o a un método → la instancia de contexto será entonces una instancia del tipo al que pertenece la operación o el método.
- El contexto **se especifica** poniendo detrás de la palabra reservada context la declaración del método (signatura).
- La precondición se indica con la palabra reservada **pre** y la postcondición con la palabra reservada **post**:
- **PRECONDICIÓN -Definición-**: Es una restricción que debe ser cierta justo antes de la ejecución de una **operación**
- **POSTCONDICIÓN- Definición-**: Es una restricción que debe ser cierta justo después de la ejecución de una operación. Permiten describir el efecto de las operaciones en OCL.

Prof. Ana M<sup>a</sup> Roldán

20

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### b. Pre y postcondiciones

**Sintaxis general:**

```
context NombreTipo::nombreOperacion(param1 : Tipo1
, ... ): tipoDevuelto
pre : param1 > ...
post: result = ...
    ↗ palabra reservada que se refiere al resultado de la operación
```

**Ejemplo:**

```
context Persona::sueldo(f : Fecha) : Integer
post: result = 5000
```

- Como en el caso de los invariantes, también se le puede dar un **nombre** a la precondition y postcondición:

```
context NombreTipo::nombreOperacion(param1 : Tipo1
, ... ): tipoDevuelto
pre [parametroCorrecto]: param1 > ...
post: result = ...
    ↗ Prof. Ana Ma Roldán
```

(21)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### Precondiciones

- Sintaxis**

```
context <classifier>::<operation> (<parameters>)
pre [<constraint name>]: <Boolean OCL expression>
```

**Ejemplos:**

```
context Meeting::shift(d : Integer)
pre: self.isConfirmed = false
```

```
context Meeting::shift(d : Integer)
pre: d > 0
```

```
context Meeting::shift(d : Integer)
pre: self.isConfirmed = false and d > 0
```

Prof. Ana M<sup>a</sup> Roldán

(22)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### **Postcondiciones**

- **Sintaxis**

```
context <classifier>::<operation>(<parameters>
post [<constraint name>]: <Boolean OCL expression>
```

**Ejemplos:**

```
context Meeting::duration(): Integer
post: result = self.mend - self.mstart
-- la palabra reservada result representa el resultado de
-- la operación
```

```
context Meeting::confirm()
post: self.isConfirmed = true
```

Prof. Ana M<sup>a</sup> Roldán

(23)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### **Postcondiciones**

- El caso de @

```
context Meeting::shift(d : Integer)
post: mstart = mstart@pre + d and mend = mend@pre + d
```

- **@pre:**
  - Sólo puede utilizarse en las **postcondiciones**.
  - Permite referirnos a una expresión que se evalúa en el estado previo a la ejecución de la operación:
    - mstart@pre se evalúa al valor del atributo mstart antes de la ejecución de la operación.
    - mstart se evalúa al valor del atributo mstart tras la ejecución de la operación.

Prof. Ana M<sup>a</sup> Roldán

(24)

 DEPARTAMENTO DE TÉCNICAS DE LA INFORMACIÓN  <b>ETSI</b> ESTADÍSTICA INVESTIGACIÓN INNOVACIÓN	1. Objetivos. 2. Introducción. 3. OCL 4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones. Referencias.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

## 3. OCL

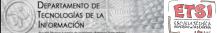
### Postcondiciones

- El caso del operador “enviado”
- El operador ^ sólo puede usarse en **postcondiciones**.
- **Especifica que se han enviado los mensajes indicados**, que la comunicación ha ocurrido.

```
context Subject::hasChanged()
post: observer^update(2,4)
-- la postcondición se satisface si se envió un mensaje
-- update al objeto observer con argumentos 2 y 4 durante
-- la ejecución del mensaje hasChanged
```

Prof. Ana M<sup>a</sup> Roldán

(25)

 DEPARTAMENTO DE TÉCNICAS DE LA INFORMACIÓN  <b>ETSI</b> ESTADÍSTICA INVESTIGACIÓN INNOVACIÓN	1. Objetivos. 2. Introducción. 3. OCL 4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones. Referencias.v
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------

## 3. OCL

### c. Restricciones OCL y herencia



- **Las restricciones se heredan.**
- En cualquier sitio donde se espera una instancia de una clase, uno puede utilizar una instancia de una subclase.
- **Un invariante de una superclase es heredado por su subclase:**
  - Una subclase puede fortalecer un invariante pero nunca debilitarlo.
- **En la redefinición de un operador en una subclase:**
  - Una precondición puede ser habilitada pero no fortalecida.
  - Una postcondición puede ser fortalecida pero no debilitada.

Prof. Ana M<sup>a</sup> Roldán

(26)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### c. Restricciones OCL y herencia

#### Ejemplos

```

context TeamMember::numConfMeeting(): Integer
post: result = meetings->select(isConfirmed)->size()

classDiagram
    class Person {
        name : String
        age : Integer
        title : String
        gender : Gender
    }
    class Date {
        day : Integer
        month : Integer
        year : Integer
    }
    class Meeting {
        title : String
        day : Date
        mstart : Integer
        mend : Integer
        isConfirmed : Boolean
    }
    class Gender {
        <<enumeration>>
        male
        female
    }
    class TeamMember {
        role : String
        numMeetings() : Integer
        numConfMeetings() : Integer
        participants : Set<Meeting>
        moderates : Set<Meeting>
        moderatedMeetings : Set<Meeting>
        members : Set<TeamMember>
        BelongsTo team
    }
    class Team {
        name : String
        forTeam : Set<TeamMember>
    }

    Person "2..*" -- "2..*" TeamMember : participants
    TeamMember "*" -- "*" Meeting : moderates
    TeamMember "*" -- "*" Meeting : moderatedMeetings
    TeamMember "2..*" -- "2..*" TeamMember : members
    TeamMember "*" -- "1" Team : BelongsTo
    Team "1" -- "1" TeamMember : forTeam
  
```

Prof. Ana M<sup>a</sup> Roldán

(27)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### c. Restricciones OCL y herencia

#### Ejemplos

- El moderador ha de ser una mujer

```

context Meeting
inv: self.moderator.gender = Gender::female -ver transp. 34
  
```

Prof. Ana M<sup>a</sup> Roldán

(28)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### c. Restricciones OCL y herencia

#### Ejemplos

- El moderador ha de ser una mujer
 

```
context Meeting
inv: self.moderator.gender = Gender::female -ver transp. 31
```
- El número de participantes en una reunión tiene que ser 2 o más
 

```
context Meeting inv: self.participants->size() >= 2
```

Prof. Ana M<sup>a</sup> Roldán

(29)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3. OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### c. Restricciones OCL y herencia

#### Ejemplos

- El moderador ha de ser una mujer
 

```
context Meeting
inv: self.moderator.gender = Gender::female -ver transp. 31
```
- El número de participantes en una reunión tiene que ser 2 o más
 

```
context Meeting inv: self.participants->size() >= 2
```
- Todos los miembros del equipo participan en una reunión de equipo
 

```
context TeamMember::numMeeting(): Integer
post: result = meetings->size()
```

Prof. Ana M<sup>a</sup> Roldán

(30)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 3. OCL

### c. Restricciones OCL y herencia

#### Ejemplos

- El moderador ha de ser una mujer
 

```
context Meeting
inv: self.moderator.gender = Gender::female -ver transp. 31
```
- El número de participantes en una reunión tiene que ser 2 o más
 

```
context Meeting inv: self.participants->size() >= 2
```
- Todos los miembros del equipo participan en una reunión de equipo
 

```
context TeamMember::numMeeting(): Integer
post: result = meetings->size()
```
- Postcondiciones de numMeeting() y numConfMeeting:
 

```
context TeamMember::numConfMeeting(): Integer
post: result = meetings->select(isConfirmed)->size()
```

Prof. Ana M<sup>a</sup> Roldán

(31)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.v

## 4. TIPOS EN OP. OCL

### BÁSICOS

Tipo	Valores
Boolean	true, false
Integer	1, -5, 2, 34, 26524, ...
Real	1.5, 3.14, ...
String	'En un lugar de ...'
UnlimitedNatural	0, 1, 2, 42, ..., *
OclInvalid	invalid
OclVoid	null, invalid

Prof. Ana M<sup>a</sup> Roldán

(32)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.v

## 4. TIPOS EN OP. OCL

### OPERACIONES

Tipo	Operaciones
Boolean	and, or, xor, not, implies, if-then-else
Integer	*, +, -, /, abs()
Real	*, +, -, /, floor()
String	concat(), size(), substring()
UnlimitedNatural	*, +,

Prof. Ana Mª Roldán

33

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.v

## 4. TIPOS EN OP. OCL

### Integer

```
i = (i2:Integer) : Boolean      i . mod (i2 : Integer) : Integer
i <> (i2:Integer) : Boolean    i . max (i2 : Integer) : Integer
i + (i2:Integer) : Integer     i . min (i2 : Integer) : Integer
i - (i2:Integer) : Integer     i < (i2 : Integer) : Boolean
i * (i2:Integer) : Integer     i > (i2 : Integer) : Boolean
i / (i2:Integer) : Real       i <= (i2 : Integer) : Boolean
i . abs () : Integer          i >= (i2 : Integer) : Boolean
i . div (i2 : Integer) : Integer
```

### Boolean

```
b = (b2:Boolean) : Boolean
b <> (b2: Boolean) : Boolean
b or (b2:Boolean) : Boolean
b xor (b2:Boolean) : Boolean
b and (b2:Boolean) : Boolean
not b : Boolean
b implies (b2:Boolean) : Boolean
if b then (b2:Boolean) else (b3:Boolean) : Boolean
```

### String

```
string = (string2:String) : Boolean
string <> (string2: String) : Boolean
string . size( ) : Integer
string . concat (string2 : String) : String
string . substring (lower : Integer, upper : Integer) : String
string . toInteger () : Integer
string . toReal () : Real
```

### Real

```
r = (r2:Real) : Boolean
r <> (r2:Real) : Boolean
r + (r2:Real) : Real
r - (r2:Real) : Real
r * (r2:Real) : Real
r / (r2:Real) : Real
r . abs () : Real
r . floor ( ) : Integer
r . round ( ) : Integer
r . max (r2 : Real) : Real
r . min (r2 : Real) : Real
r < (r2 : Real) : Boolean
r > (r2 : Real) : Boolean
r <= (r2 : Real) : Boolean
r >= (r2 : Real) : Boolean
```

Prof. Ana Mª Roldán

34

## 4. TIPOS EN OP. OCL

### Tipos del modelo UML

- Todas las clases que se definen en el modelo UML que sirve de contexto a una expresión OCL, se pueden usar en esa expresión.
- *En el caso particular de los tipos enumerados, en las expresiones OCL se puede hacer referencia a los valores de esos tipos usando el nombre de tipo como prefijo del valor.*

```
context Persona inv:  
    sexo =Sexo :: hombre
```

Prof. Ana Mª Roldán

35