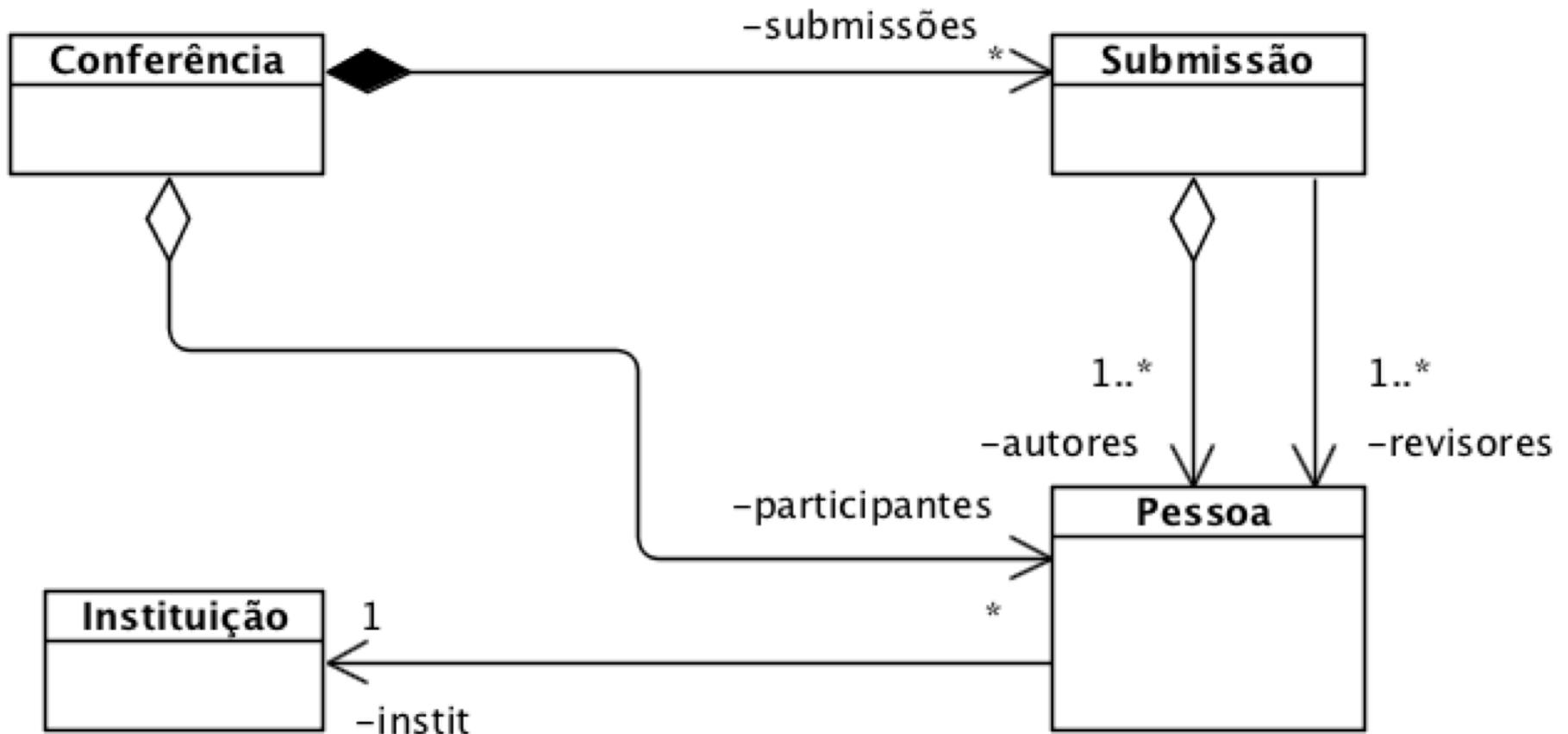


# Desenvolvimento de Sistemas Software

Aula Teórica 16

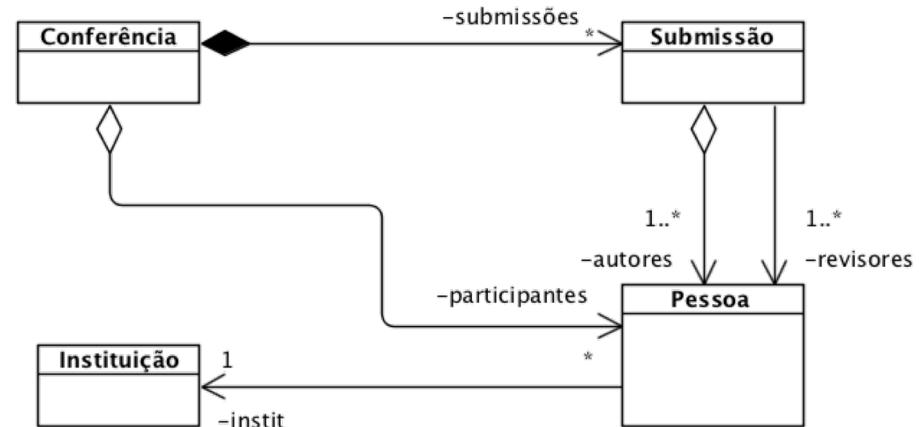
## OCL - Object Constraint Language

# Diagramas UML nem sempre são suficientes



# Diagramas UML nem sempre são suficientes

1. Os revisores de uma submissão não podem ser seus autores!
2. Os revisores de uma submissão não podem ser da mesma instituição dos autores!



- Como expressar estas restrições de forma não ambígua e “mecanizável” ?

## OCL: Object Constraint Language

- Linguagem de expressões tipada - expressões não têm efeitos laterais
- Não é uma linguagem de programação

## Breve História da OCL

- Em 1995 a divisão de seguros da IBM desenvolve uma linguagem para modelação de negócio
  - IBEL (Integrated Business Engineering Language)
- IBM propõe a IBEL ao OMG
  - OCL integrado na UML 1.1
  - A OCL é utilizada para definir a UML 1.2

# Vantagens de Formalizar as Restrições

- Melhor documentação
  - As restrições adicionam informação àcerca dos elementos e suas relações aos modelos visuais da UML
  - Permitem documentar o modelo
- Maior precisão
  - As restrições escritas em OCL têm uma semântica formal
  - Ajudam a diminuir a ambiguidade dos modelos
- Melhor Comunicação
  - Se os modelos UML são utilizados para comunicar, as restrições OCL permitem comunicar sem ambiguidade

# Onde utilizar OCL?

- Restrições em operações e associações
- Invariantes de classe e tipos
  - Uma restrição que deve ser verdadeira num objecto durante todo o seu tempo de vida
- Pré- e pós-condições dos métodos
  - Restrições que especificam as condições de aplicabilidade/efeito de uma operação

# Sistema de tipos OCL

- Tipos primitivos

Type	Description	Values	Operations
<b>Boolean</b>		true, false	=, <>, and, or, xor, not, implies, if-then-else-endif
<b>Integer</b>	A whole number of any size	-1, 0, 1, ...	=, <>, >, <, >=, <=, *, +, - (unary), - (binary), / (real) abs(), max(b), min(b), mod(b), div(b)
<b>Real</b>	A real number of any size	1.5, ...	=, <>, >, <, >=, <=, *, +, - (unary), - (binary), / abs(), max(b), min(b), round(), floor()
<b>String</b>	A string of characters	'a', 'John'	=, <> size(), concat(s2), substring(from, to) toInteger(), toReal(),

# Sistema de tipos OCL

- Colecções e Tuplos

Description	Syntax	Examples
Abstract collection of elements of type T	<b>Collection(T)</b>	
Unordered collection, no duplicates	<b>Set(T)</b>	Set{1 , 2}
Ordered collection, duplicates allowed	<b>Sequence(T)</b>	Sequence {1, 2, 1} Sequence {1..4} (same as {1,2,3,4})
Ordered collection, no duplicates	<b>OrderedSet(T)</b>	OrderedSet {2, 1}
Unordered collection, duplicates allowed	<b>Bag(T)</b>	Bag {1, 1, 2}
Tuple (with named parts)	<b> Tuple(field1: T1,           fieldn : Tn)</b>	Tuple {age: Integer = 5, name: String = 'Joe' } Tuple {name = 'Joe', age = 5}



# Colecções - Operações

Operation	Description
<b>size(): Integer</b>	The number of elements in this collection ( <i>self</i> )
<b>isEmpty(): Boolean</b>	<i>size</i> = 0
<b>notEmpty(): Boolean</b>	<i>size</i> > 0
<b>includes(object: T): Boolean</b>	True if <i>object</i> is an element of <i>self</i>
<b>excludes(object: T): Boolean</b>	True if <i>object</i> is not an element of <i>self</i>
<b>count(object: T): Integer</b>	The number of occurrences of <i>object</i> in <i>self</i>
<b>includesAll(c2: Collection(T)): Boolean</b>	True if <i>self</i> contains all the elements of <i>c2</i>
<b>excludesAll(c2: Collection(T)): Boolean</b>	True if <i>self</i> contains none of the elements of <i>c2</i>
<b>sum(): T</b>	The addition of all elements in <i>self</i> (T must support "+")
<b>product(c2: Collection(T2)) : Set(Tuple(first:T, second:T2))</b>	The cartesian product operation of <i>self</i> and <i>c2</i> .

**Note:** Operations on collections are applied with “->” and not “.”

# Tipos de Colecções

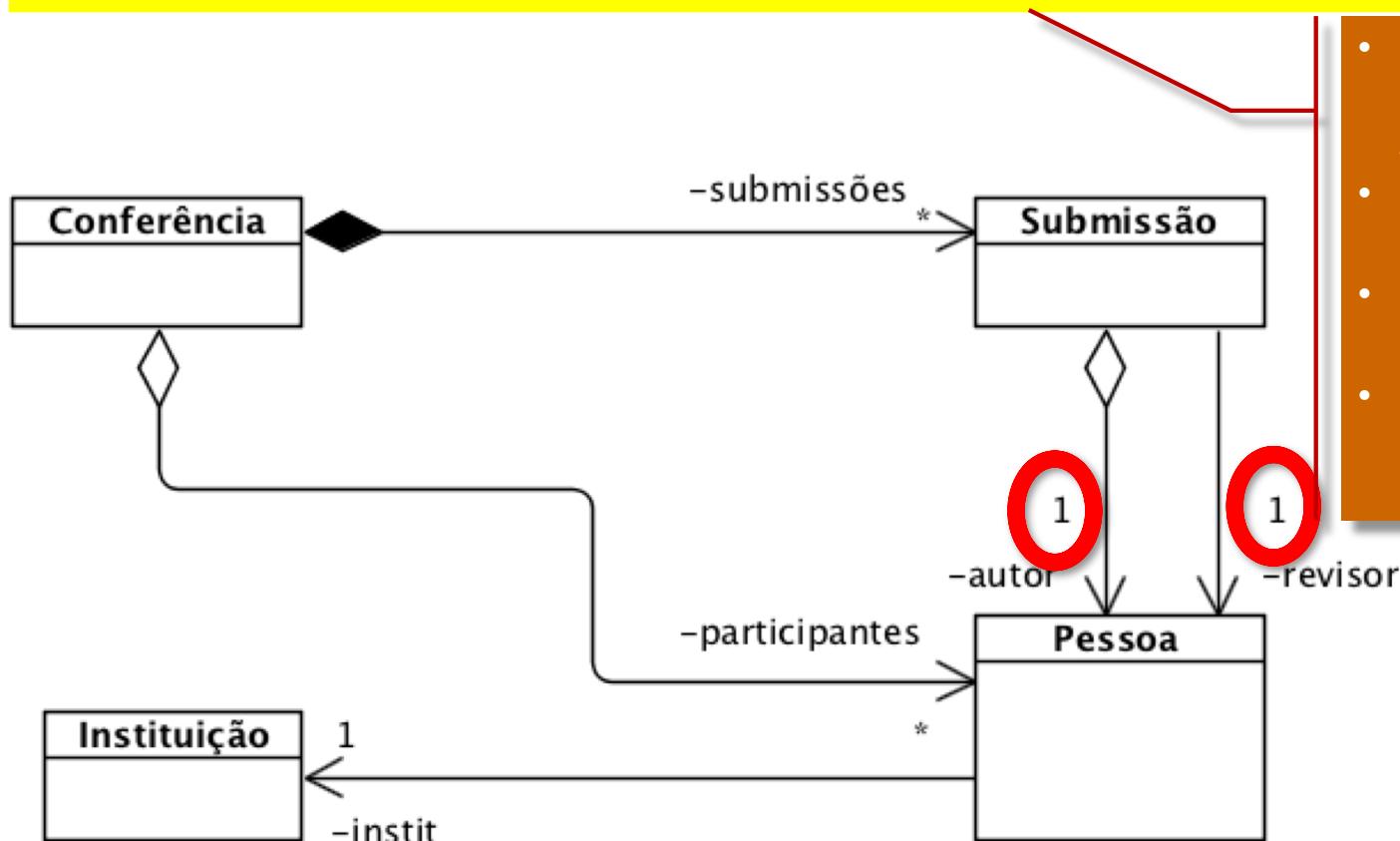
- Colecções podem ser
  - Set - sem ordem e sem duplicados
  - OrderedSet - com ordem e sem duplicados
  - Bag - sem ordem e com duplicados
  - Sequence - com ordem e com duplicados
- Cada um tem operações apropriadas ao tipo (herdam as das coleções)
  - Set: =, union, intersection, -(difference), ...
  - OrderedSet: =, union, intersection, ...
  - Bag: =, union, intersection, flatten, ...
  - Sequence: =, append, prepend, insertAt, subSequence, ...

# Invariante

1. Os revisores de uma submissão não podem ser seus autores

**context** Submissão

**inv** SemConflito: self.autor <> self.revisor

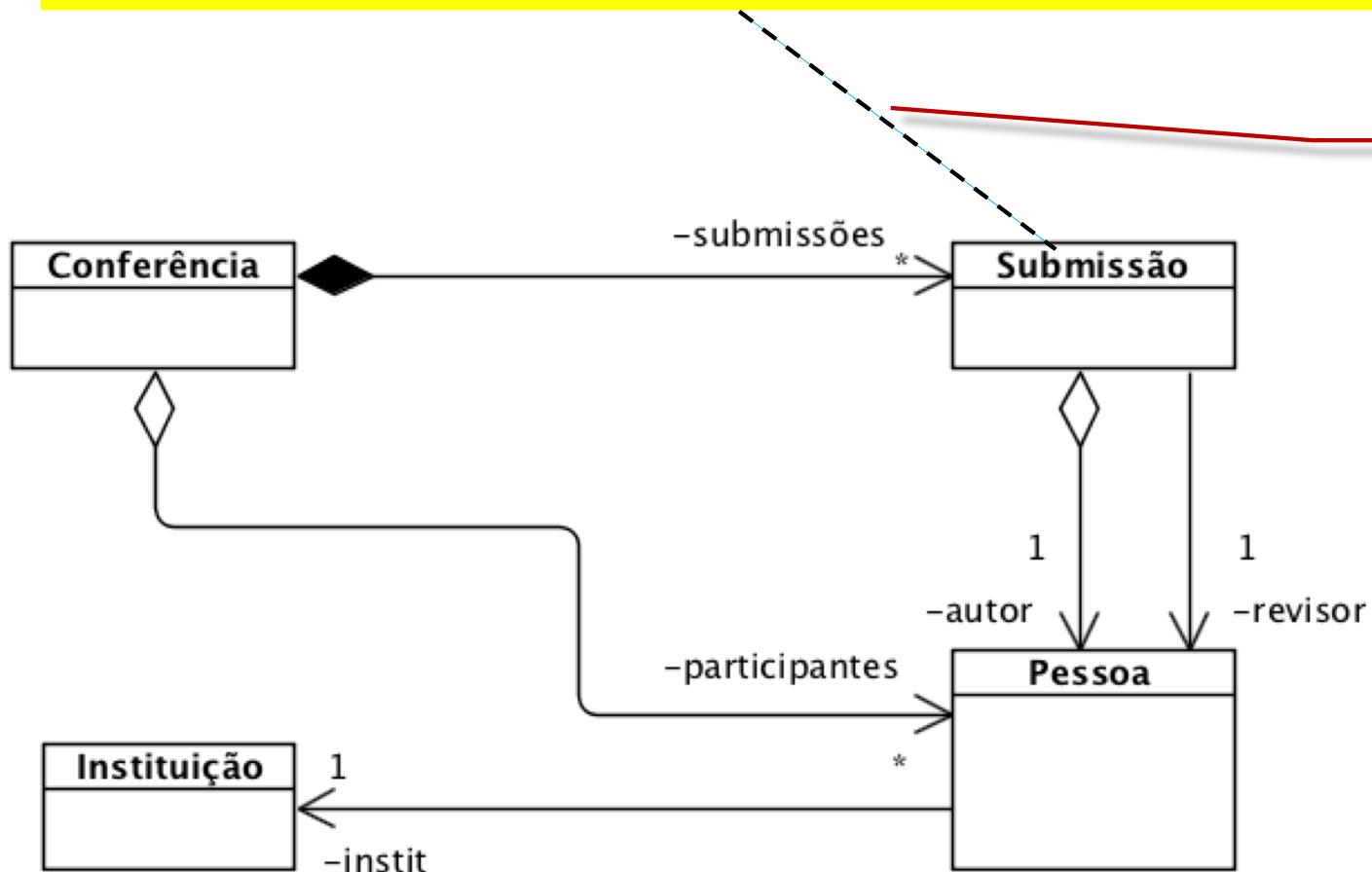


- Cada invariante tem um contexto (Classe, Interface ou Classe de Associação)
- O contexto pode ser referido utilizando *self* (opcional)
- Os invariantes podem ter um nome (neste caso é SemConflito)
- Os invariantes são expressões booleanas

# Invariante

- Os revisores de uma submissão não podem ser seus autores

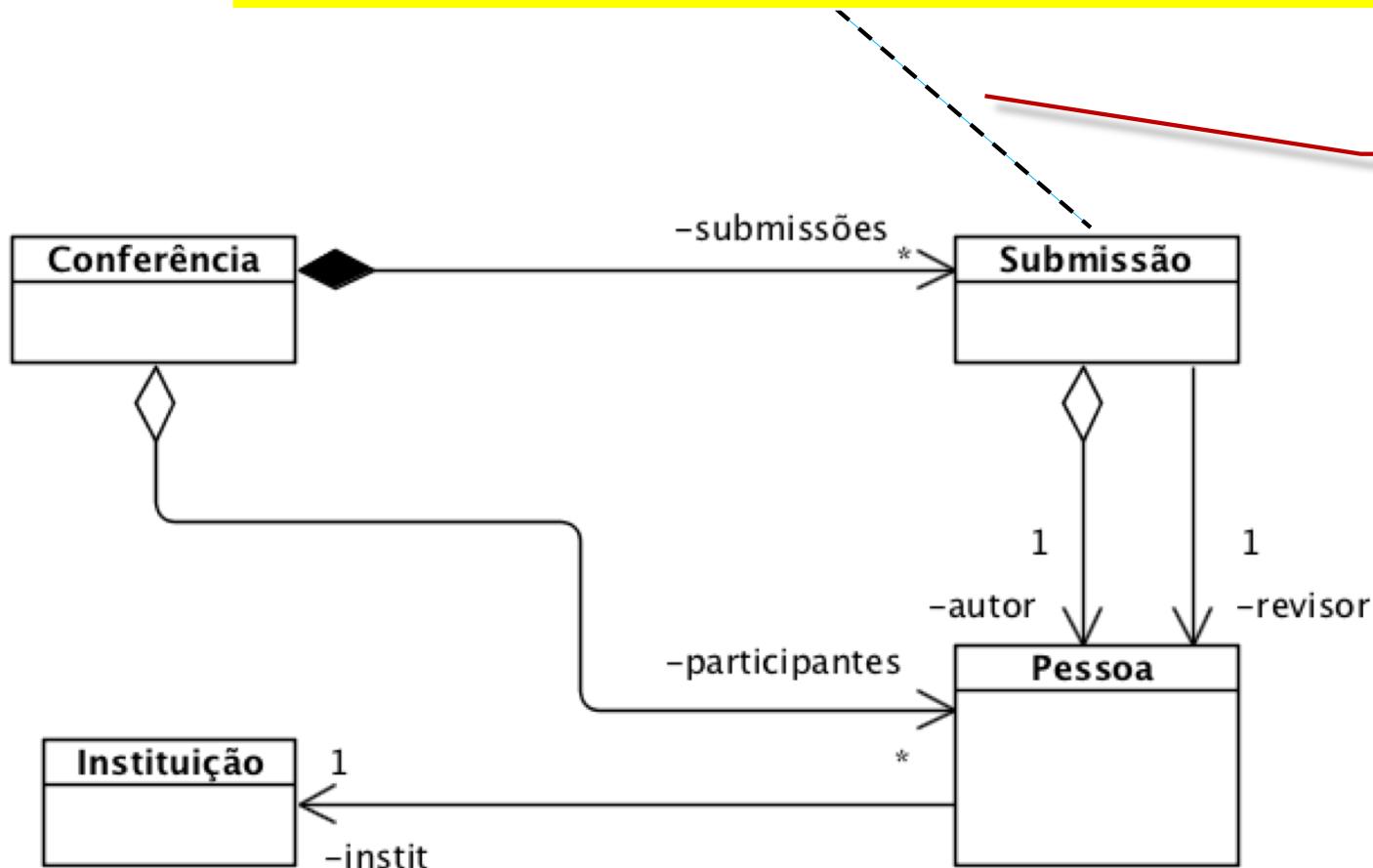
```
inv SemConflito: self.autor <> self.revisor
```



# Invariante

- Os revisores de uma submissão não podem ser seus autores

**inv SemConflito: autor <> revisor**



Quando não existe ambiguidade, *self* é opcional

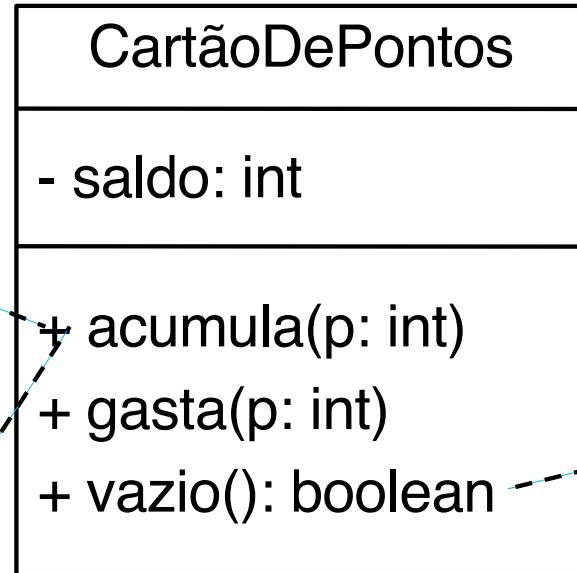
# Pré- e pós-condições

Podem referir-se  
atributos nos invariantes

**inv:** saldo  $\geq 0$

**pre:**  $p > 0$

1



2

**post:**

saldo = saldo@pre + p

saldo@pre: valor do  
saldo antes da operação

result: resultado  
da operação

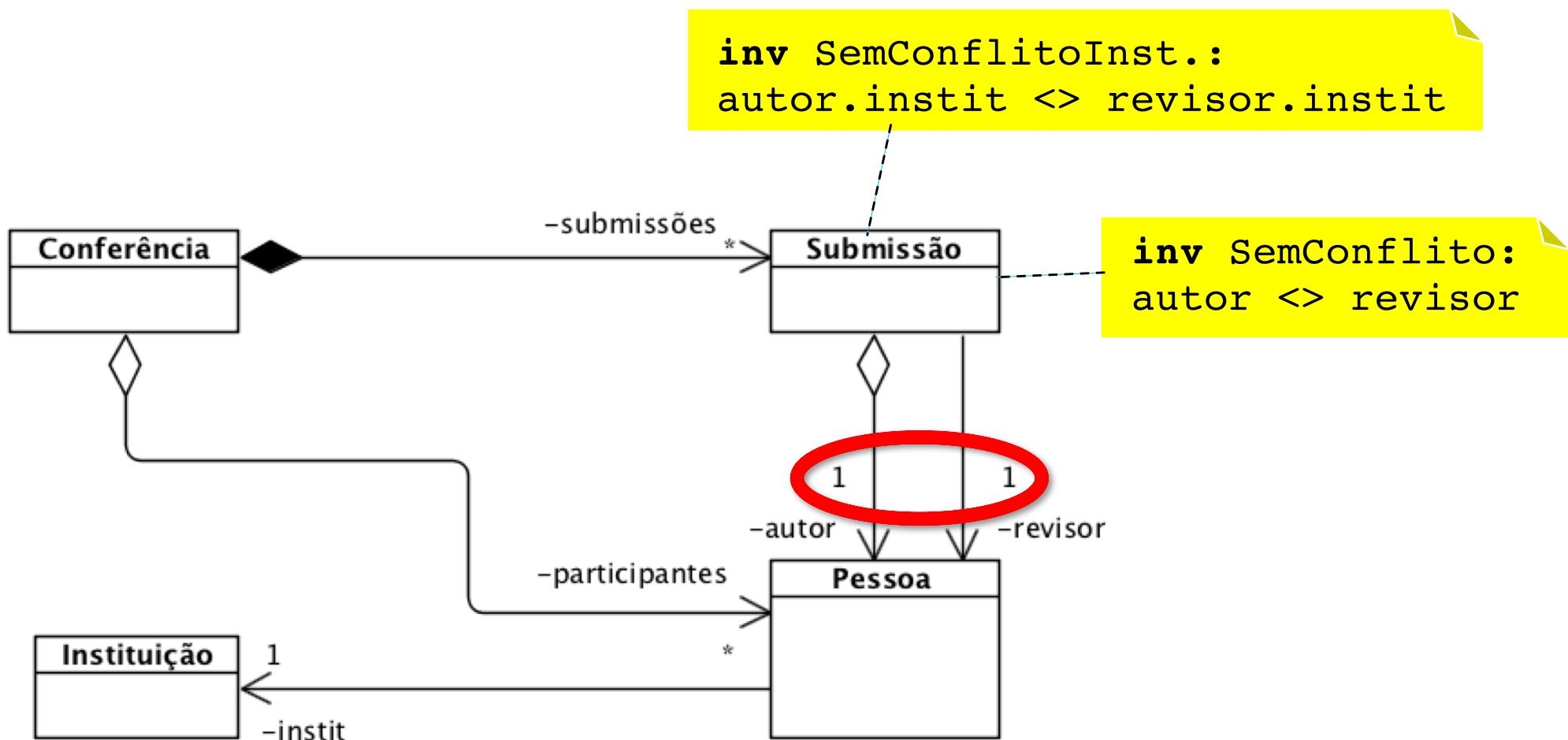
**post:** result = (saldo=0)

**context** CartãoDePontos::gasta(p:int)  
**pre:** saldo $\geq p$  and  $p \geq 0$

**context** CartãoDePontos::gasta(p:int)  
**post:** saldo = saldo@pre - p

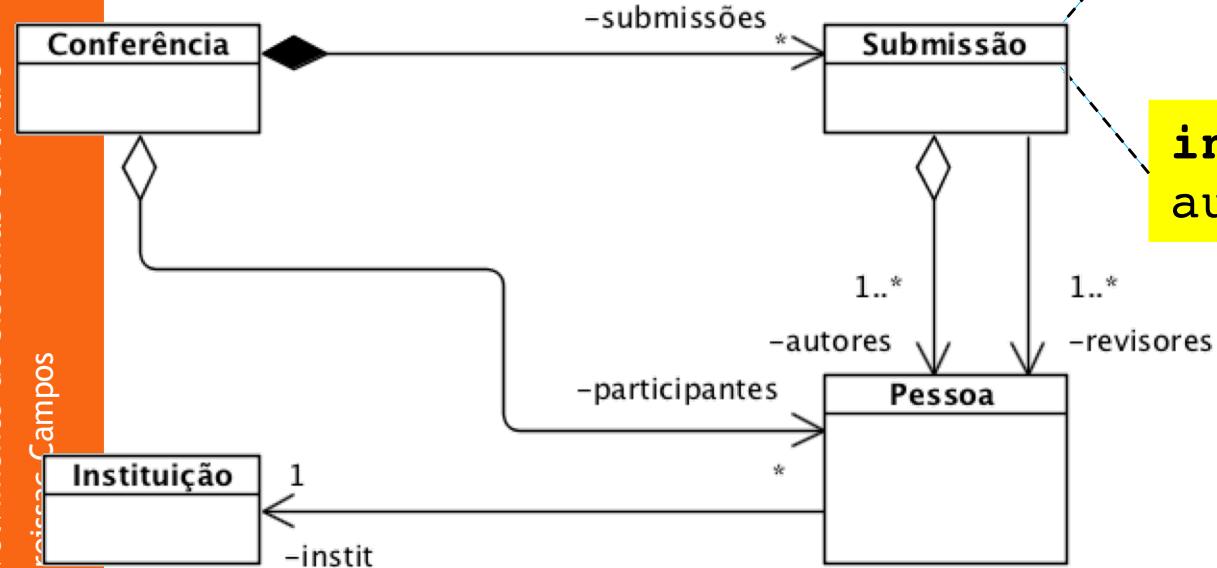
# Navegação nas associações

1. Os revisores de uma submissão não podem ser seus autores
2. Os revisores de uma submissão não podem ser da mesma instituição dos autores



## Colecções - exemplos

1. Os revisores de uma submissão não podem ser seus autores
2. Os revisores de uma submissão não podem ser da mesma instituição dos autores



**inv SemConflitoInst.:**  
????

**inv SemConflito:**  
autores->excludesAll(revisores)

# Colecções - iteradores (tipo reduce)

Iterator expression	Description
<b>iterate(iterator: T; accum: T2 = init   body) : T2</b>	Returns the final value of an accumulator that, after initialization, is updated with the value of the <i>body</i> expression for every element in the <i>source</i> collection.
<b>exists(iterators   body) : Boolean</b>	True if <i>body</i> evaluates to true for at least one element in the <i>source</i> collection. Allows multiple iterator variables.
<b>forAll(iterators   body): Boolean</b>	True if <i>body</i> evaluates to true for each element in the <i>source</i> collection. Allows multiple iterator variables.
<b>one(iterator   body): Boolean</b>	True if there is exactly one element in the <i>source</i> collection for which <i>body</i> is true
<b>isUnique(iterator   body): Boolean</b>	Results in true if <i>body</i> evaluates to a different value for each element in the <i>source</i> collection.
<b>any(iterator   body): T</b>	Returns any element in the <i>source</i> collection for which <i>body</i> evaluates to true. The result is null if there is none.

**Note:** The iterator variable declaration can be omitted when there is no ambiguity.

# Colecções - iteradores (tipo map)

Iterator expression	Description
<b>select(iterator   body): Collection(T)</b>	The Collection of elements of the <i>source</i> collection for which <i>body</i> is true. The result collection is of the same type of the <i>source</i> collection.
<b>reject(iterator   body): Collection(T)</b>	The Collection of elements of the <i>source</i> collection for which <i>body</i> is false. The result collection is of the same type of the <i>source</i> collection.
<b>collectNested(iterator   body): CollectionWithDuplicates(T2 )</b>	The Collection of elements (allowing duplicates) that results from applying <i>body</i> (of type T2) to every member of the <i>source</i> collection. The result is not flattened. Collection type conversions: Set -> Bag, OrderedSet -> Sequence.
<b>sortedBy(iterator   body): OrderedCollection(T)</b>	Returns an ordered Collection of all the elements of the <i>source</i> collection by ascending order of the value of the <i>body</i> expression. The type T2 of the <i>body</i> expression must support “<”. Collection type conversions: Set -> OrderedSet, Bag -> Sequence.
<b>collect(iterator  body): Collection(T2 )</b>	The Collection of elements resulting from applying <i>body</i> to every member of the <i>source</i> set. The result is flattened.

# Colecções - iteradores (tipo map)

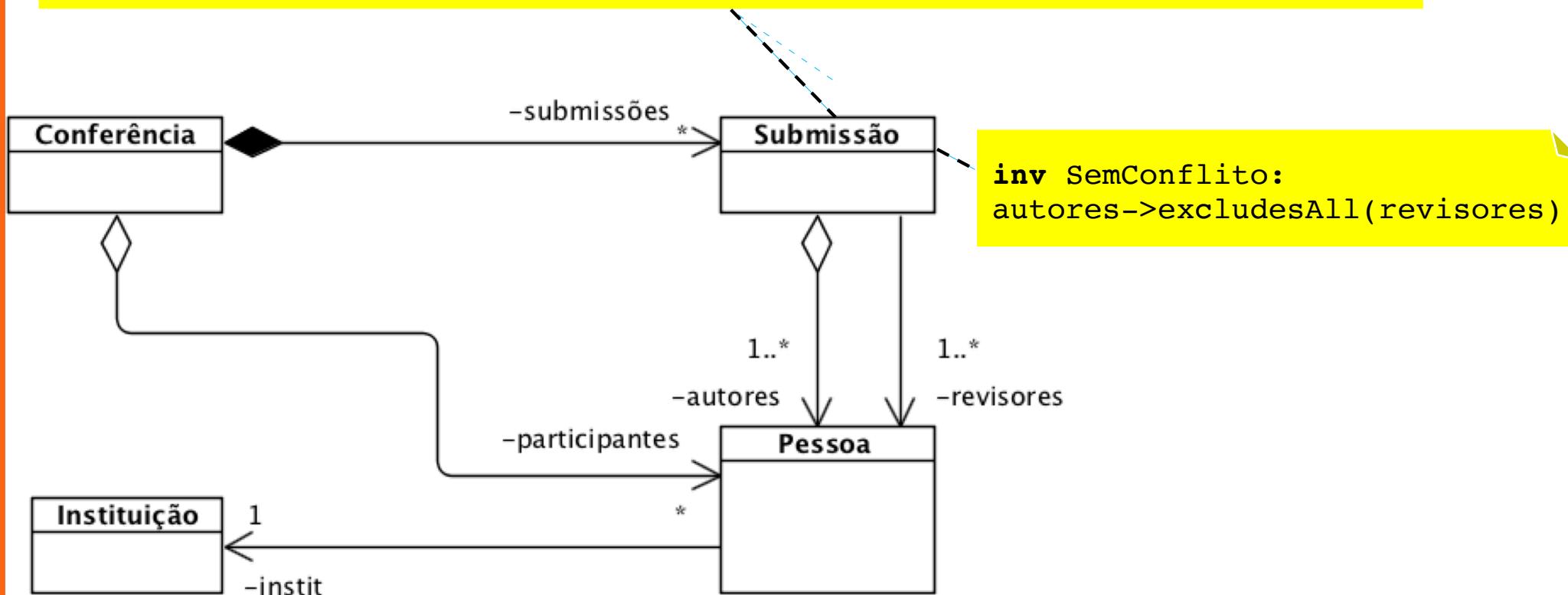
Iterator expression	Description
<b>select(iterator   body): Collection(T)</b>	The Collection of elements of the <i>source</i> collection for which <i>body</i> is true. The result collection is of the same type of the <i>source</i> collection.
<b>reject(iterator   body): Collection(T)</b>	The Collection of elements of the <i>source</i> collection for which <i>body</i> is false. The result collection is of the same type of the <i>source</i> collection.
<b>collect(iterator   body): Collection(T2)</b>	The Collection of elements resulting from applying <i>body</i> to every member of the <i>source</i> set. The result is flattened.
<b>collectNested(iterator   body): CollectionWithDuplicates(T2)</b>	The Collection of elements (allowing duplicates) that results from applying <i>body</i> (of type T2) to every member of the <i>source</i> collection. The result is not flattened. Collection type conversions: Set -> Bag, OrderedSet -> Sequence.
<b>sortedBy(iterator   body): OrderedCollection(T)</b>	Returns an ordered Collection of all the elements of the <i>source</i> collection by ascending order of the value of the <i>body</i> expression. The type T2 of the <i>body</i> expression must support “<”. Collection type conversions: Set -> OrderedSet, Bag -> Sequence.

# Colecções - exemplos

Se o iterador não é ambíguo, pode ser omitido (para autores apresenta-se a notação completa)

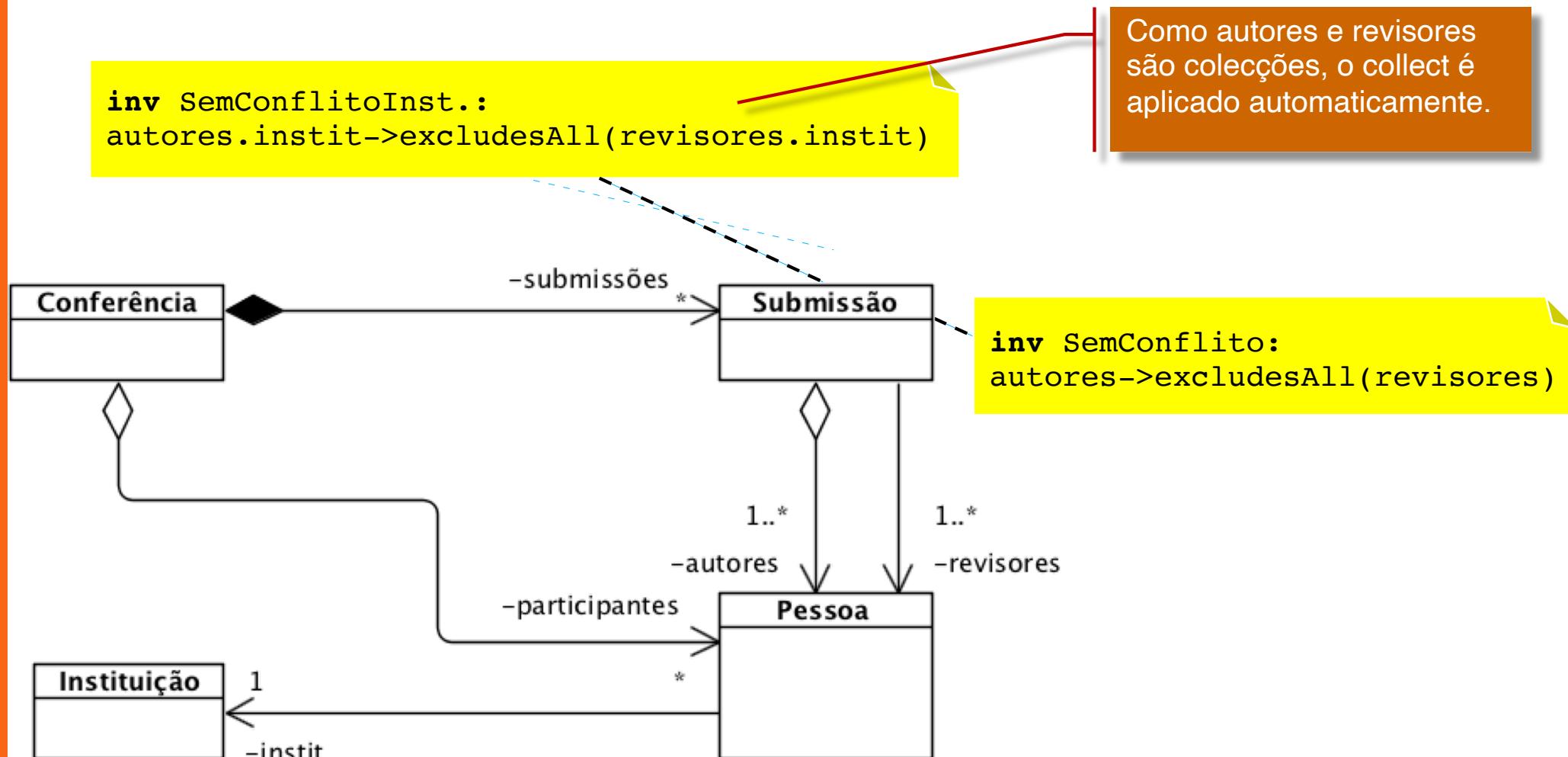
1. Os revisores de uma submissão não podem ser seus autores
2. Os revisores de uma submissão não podem ser da mesma instituição dos autores

```
inv SemConflitoInst.:
(autores->collect(a | a.instit))->excludesAll(revisores->collect(instit))
```



# Colecções - exemplos

1. Os revisores de uma submissão não podem ser seus autores
2. Os revisores de uma submissão não podem ser da mesma instituição dos autores



# OCL

## Sumário

- Object Constraint Language - OCL
  - História e objectivos da OCL
  - Sistema de tipos
  - Invariantes
  - Pré- e pós-condições