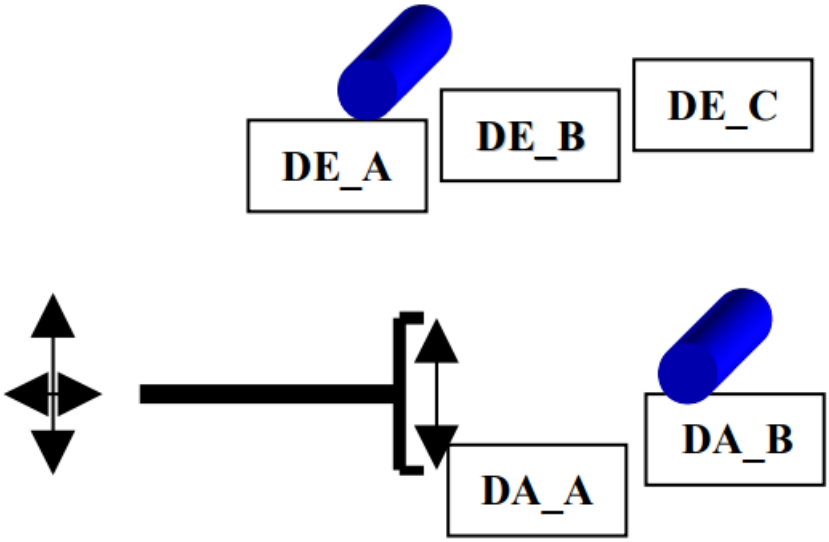
UNIVERSITÉ DE FRANCHE-COMTÉ

**PROJET IBM M2**

**Année 2016/2017**

# IDM
## Projet OCL

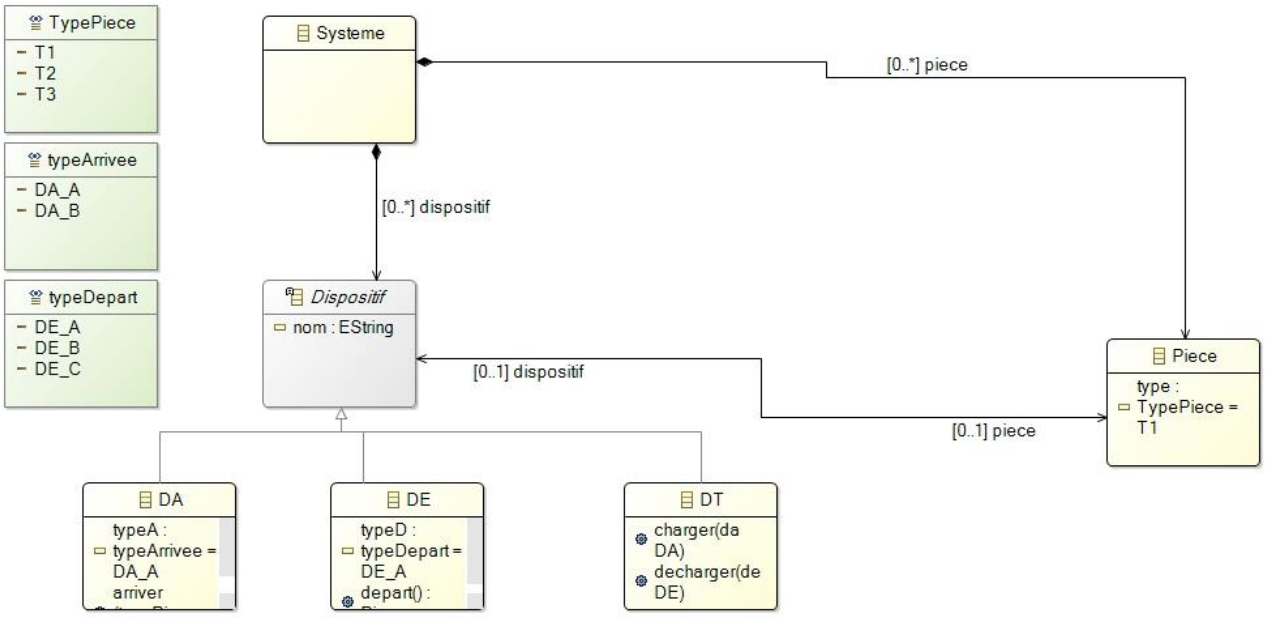



**Enseignant : Bruno TATIBOUET**

Pierre Wargnier

# Sommaire

## Diagramme de Classe




Voici un diagramme de classe qui modélise le système.
Pour ce diagramme, nous avons décidé de modéliser les pièces et les Dispositifs sous forme d'énumération. Une autre solution serait de les faire apparaitre sous forme de classe héritant des classes DA et DE. Nous avons préféré la première car elle apporte une meilleure lisibilité au diagramme, de plus les objets étant des objets simples, ils se prêtent bien à l'énumération. Ce diagramme réalise les deux premières contraintes.

# Contrainte

## 1. Première Contrainte

Une pièce ne peut arriver sur un dispositif d'arrivée que si celui-ci est libre.

Nous avons choisi de gérer l'arrivée de pièce sur un dispositif grâce à la méthode arriver(). Ainsi nous avons dû contraindre la méthode en OCL. De plus, la contrainte est déjà effective grâce au diagramme de classe.

OCL :

```
operation arriver(typePiece : Piece[?])
{
	precondition : self.piece->isEmpty();
	postcondition : self.piece-> size()=1;
}
```

## 2. Deuxième Contrainte

DT ne peut charger une pièce que s'il est libre (et qu'il y a une pièce sur le dispositif d'arrivée).

Nous avons choisi de gérer l'arrivée de pièce sur un DT grâce à la méthode charger(Dispositif d'arrivée). Ainsi nous avons dû contraindre la méthode en OCL. De plus, la contrainte est déjà effective grâce au diagramme de classe.

OCL :

```
operation charger(da : DA[?])
{
	precondition : self.piece->isEmpty() and da.piece->notEmpty();
	postcondition p2: self.piece->notEmpty() and da.piece->isEmpty();
}
```

## 3. Troisième Contrainte

Le dispositif d'évacuation DE_A ne peut recevoir que des pièces du type T1 et T2, le dispositif d'évacuation DE_B ne peut recevoir que des pièces du type T2 et le dispositif d'évacuation DE_C ne peut recevoir que des pièces du type T1 et T3.

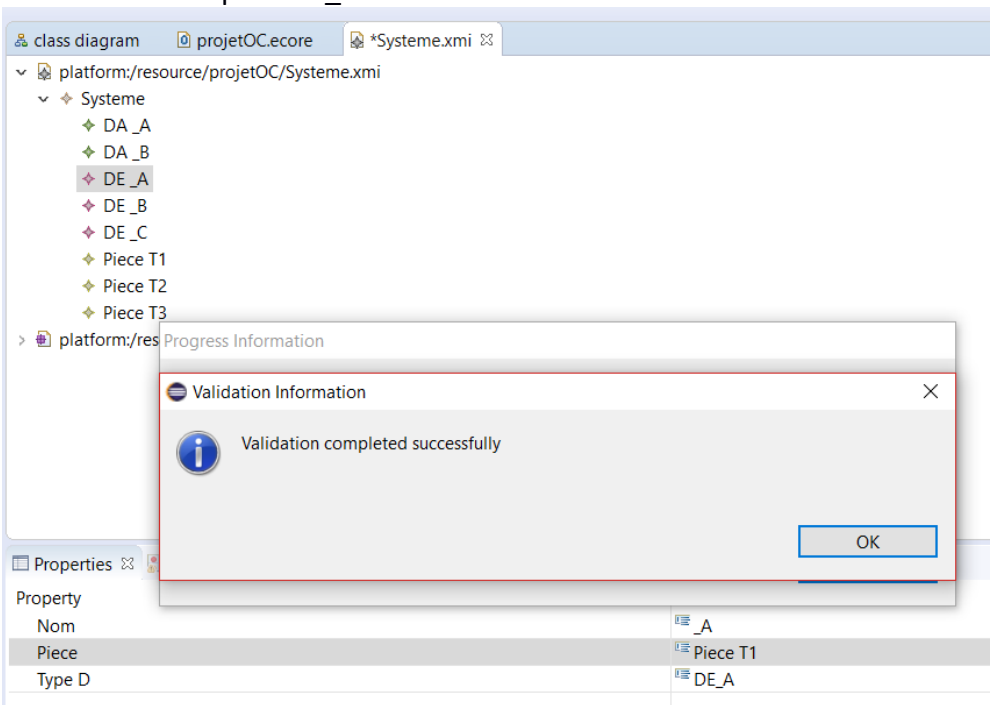
Nous avons représenté la contrainte grâce à un invariant dans la classe DE.

OCL :

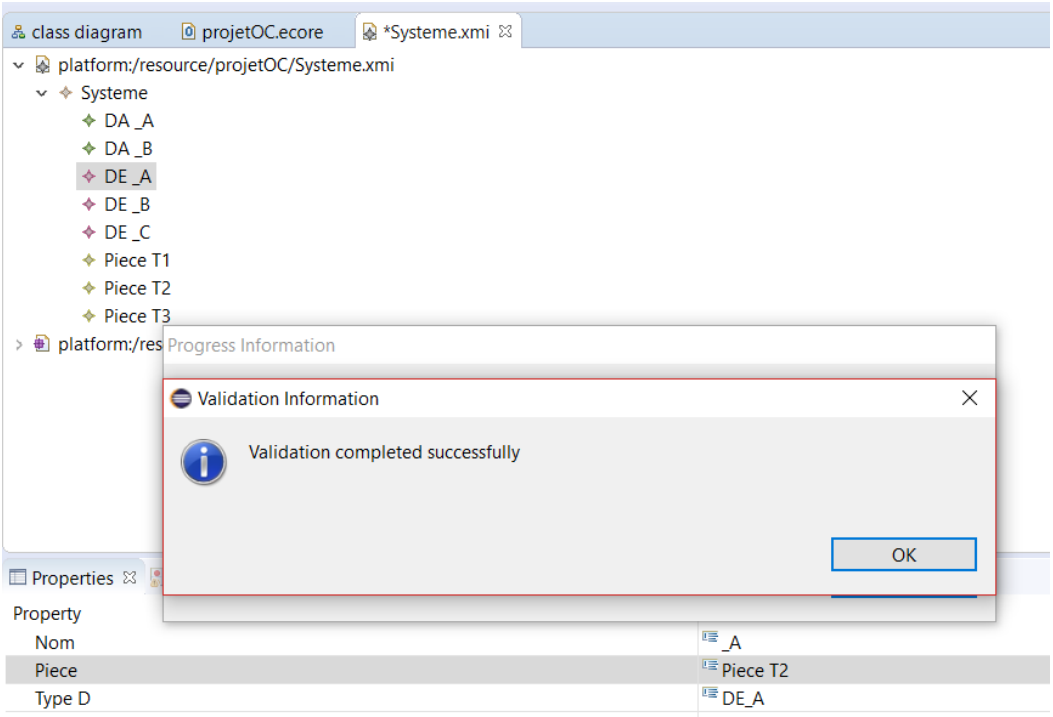
```
Invariant t1:
   if self.typeD = typeDepart::DE_A
      then (self.piece.type = TypePiece::T1 or self.piece.type = TypePiece::T2
      or self.piece->isEmpty())
      else if self.typeD = typeDepart::DE_B
            then (self.piece.type = TypePiece::T2 or self.piece->isEmpty())
                  else if self.typeD = typeDepart::DE_C
                        then (self.piece.type = TypePiece::T1 or
                        self.piece.type = TypePiece::T3 or
                        self.piece->isEmpty())
                        else false
                  endif
      endif
    endif;
```

### Validation du modèle :

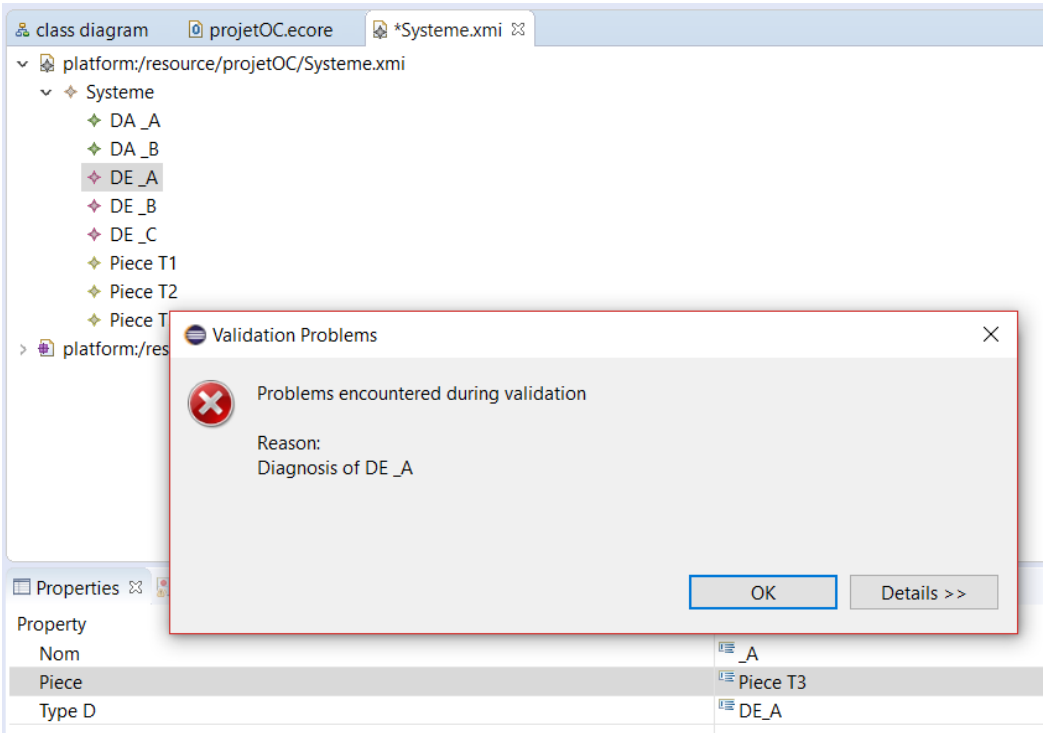
On obtient donc pour DE_A :




La pièce T1 est acceptée.

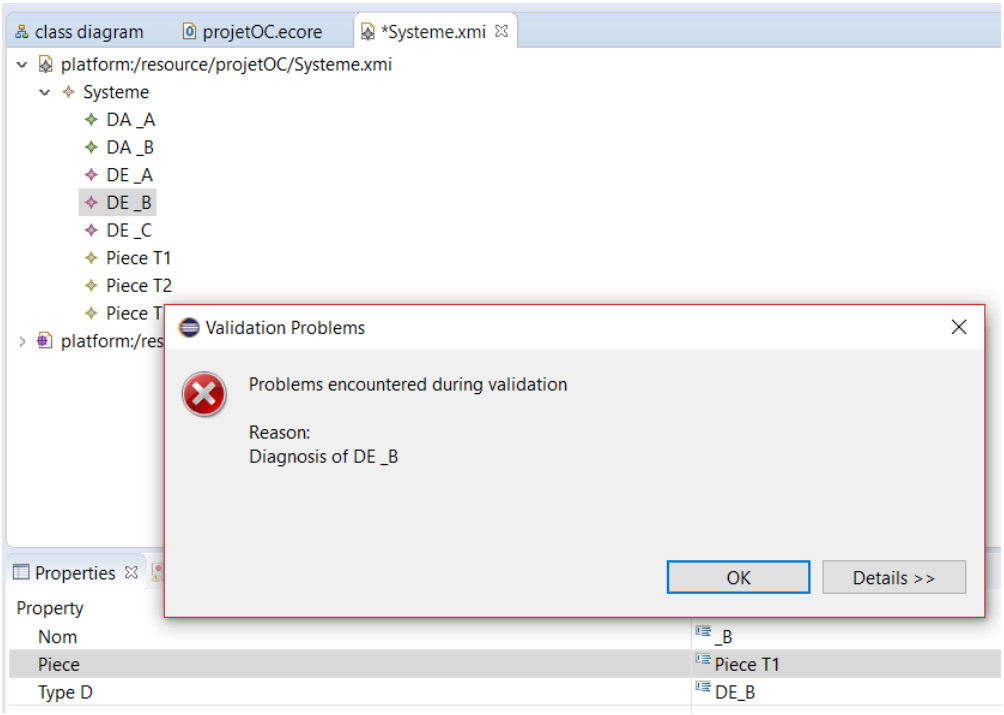



La pièce T2 est acceptée.

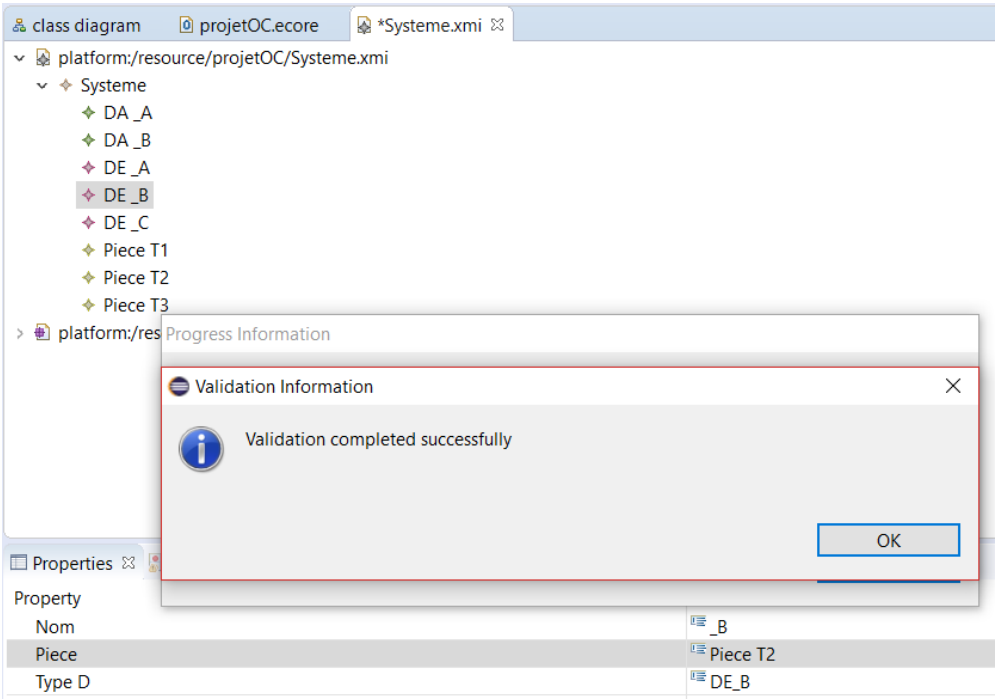



La pièce T3 est refusée car elle viole la contrainte.

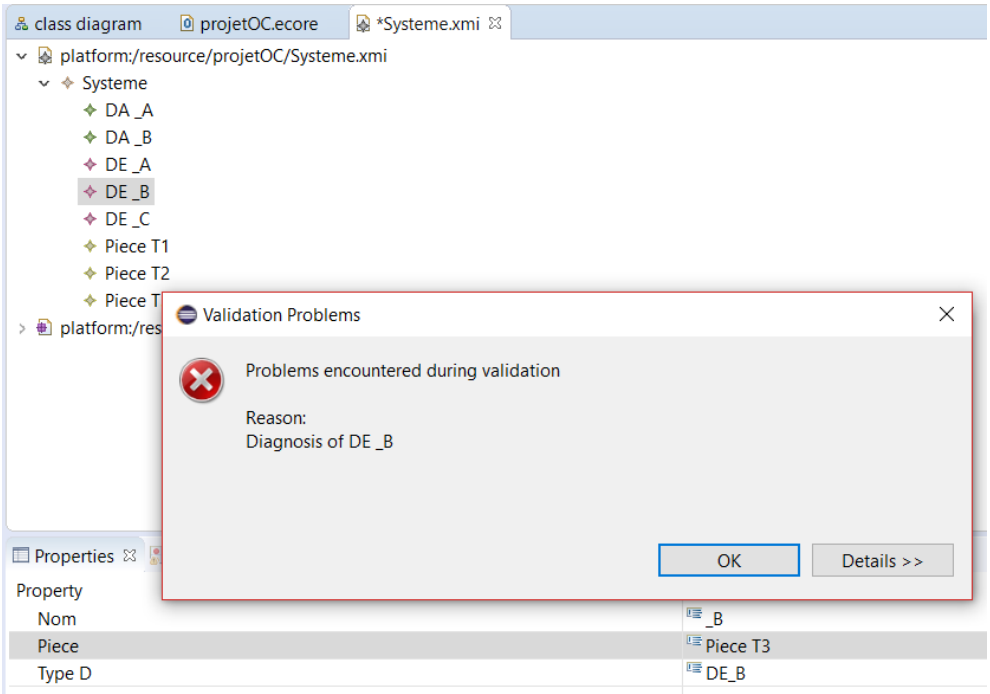
On obtient pour DE_B :




La pièce T1 est refusée car elle viole la contrainte.

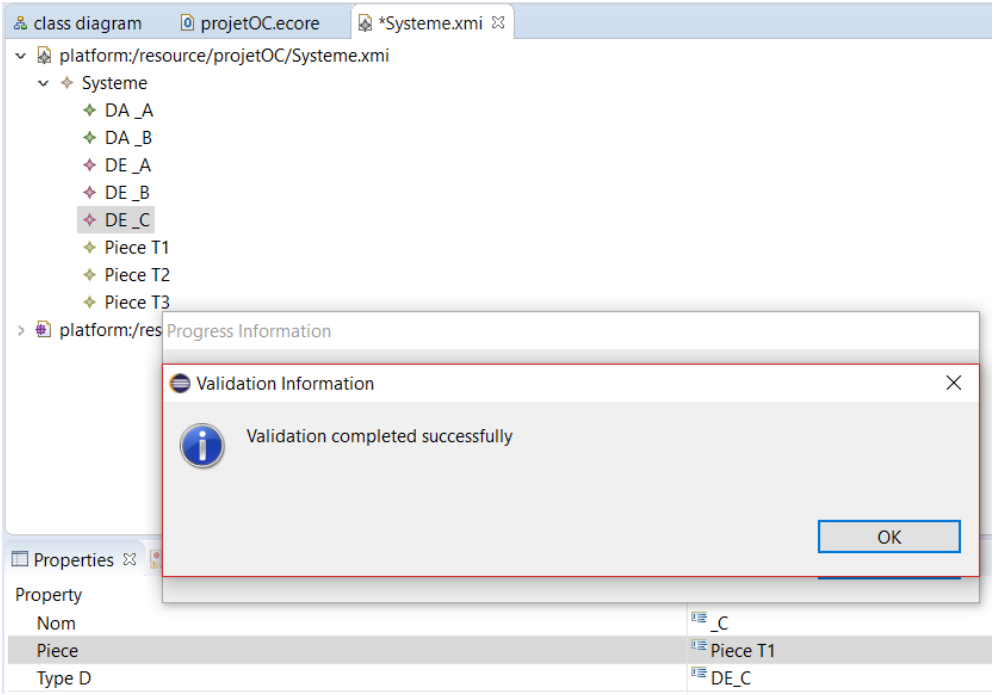



La pièce T2 est acceptée.

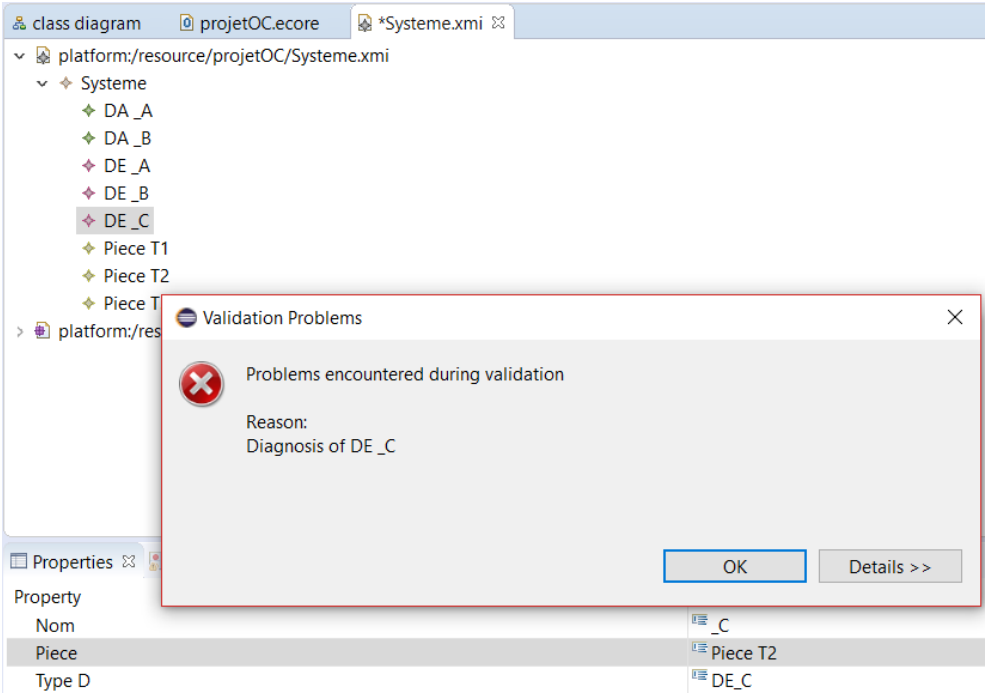



La pièce T3 est refusée car elle viole la contrainte.

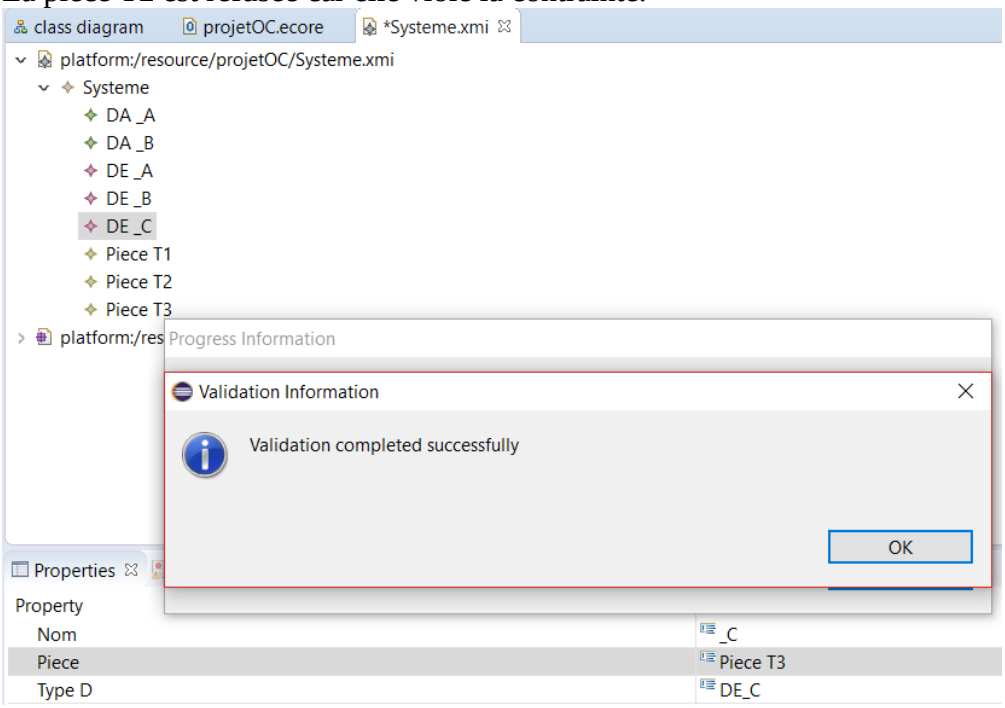
Et au final pour DE_C :




La pièce T1 est acceptée.




La pièce T2 est refusée car elle viole la contrainte.




Et la pièce T3 est acceptée.

## 4. Quatrième Contrainte

DT ne peut charger une pièce sur un dispositif d'évacuation que si ce dispositif est libre et si le dispositif d'évacuation est prévu pour ce type de pièces.

Nous avons prévu que DT interagirai avec les autres dispositifs à l'aide des fonctions charger/Decharger. Ainsi, il nous a fallu contraindre la méthode decharger(Dispositif d'évacuation) pour qu'elle respect les contraintes.

OCL :

```
operation decharger(de : DE[?])
{
    precondition
    p1: if de.typeD = typeDepart::DE_A then (self.piece = TypePiece::T1 or self.piece = TypePiece::T2) and de.piece->isEmpty()
    else if de.typeD = typeDepart::DE_B then self.piece = TypePiece::T2 and de.piece->isEmpty()
    else if de.typeD = typeDepart::DE_C then (self.piece = TypePiece::T1 or self.piece = TypePiece::T3) and de.piece->isEmpty() else false endif endif endif;
}
```