

**LMO 2004**  
***Présentation dernière minute***

**Un paradigme de programmation entités**

Jacques Mathon  
LIRIS – UCBLyon 1

# **Sommaire**

- Objectifs de la présentation
- Contexte et hypothèses de travail
- Paradigme entités
- Langage MDSL
- Spécificités du langage
- Exemples
- Perspectives et conclusion

# ***Objectifs de la présentation***

- Discuter le bien fondé voire l'opportunité de l'approche
- Discuter les principes du paradigme entités
- Critiquer le langage MDSL
- Evaluer quelques perspectives
- ...
- Provoquer des réactions... voire des questions

# ***Contexte de travail (prospection)***

- **Evolutivité (au sens commun du terme)**
  - Des applications
  - Du système
  - Du langage lui même
- **Approche expérimentale et exploratoire**
  - Analyse -> modélisation (dans une intention-projet)
  - Intégration de l'action de conception et de modélisation à l'action de programmation (au niveau langage)

# ***Paradigme entités***

Paradigme objets mais...

- Pas de classes/instances ni de prototypes/clones (a priori) -> entités
- Pas d'héritage structurel mais héritage comportemental
- Dynamicité globale
- Pas de gestion de cohérence (a priori)

# Paradigme entités

**Entité:** conteneur nommé d'attribut(s) et de méthodes

**Attribut:** couple (nom attribut, description de la valeur)

**Méthode:** couple (nom de méthode, référence au code)

<i>Nom de l'entité</i>
<i>(Nom de l'attribut, description de la valeur)</i> . .
<i>(Nom de la méthode, référence au code)</i> . .

# ***Paradigme entités***

## **Primitives**

- **Réification simple (création d'une entité)**
- **Affectation d'attribut**
- **Affectation de méthode**
- **Réification d'attribut (ou de méthode)**

## **Envoi de messages**

## **Héritage comportemental**

# ***Langage MDSL***

Une implémentation du paradigme entités

- Une entité initiale possède les comportements primitifs (après amorçage du... système)
- Les primitives sont implémentées par des codes
- L'utilisateur interagit avec le système par envoi de messages

# ***Spécificités de MDSL***

Deux spécificités significatives:

- La réification d'attribut
- La gestion de la *valeur*

# Réification d'attribut

- Ne pas confondre avec la réification de relation
  - « *Un attribut n'est pas la relation... qu'il concrétise éventuellement pour l'entité qui le contient* »
- La réification d'attribut consiste à créer une entité qui *représente* cet attribut pour l'entité qui le contient
  - « *Tout n'est pas entité, tout est réifiable* »

# Réification d'attribut

<b>Nom de l'entité</b>
<b>(Nom de l'attribut, ..●)</b>
.
.

<b>Nom de l'entité-attribut</b>
<b>(« value », description de la valeur)</b>
.
.

# Réification d'attribut

- L'attribut, ainsi réifié, fournit, à l'entité pour laquelle il représente cet attribut, la description de la valeur de celui-ci. Concrètement, le système procède par envoi d'un message `getValue` (primitive non modifiable) à l'attribut réifié. Cette méthode peut elle même être surchargée.
- De manière similaire, l'attribut spécifique « *value* » de l'attribut réifié peut à son tour être réifié.

# *Gestion de la valeur*

Qu'est-ce que la valeur d'un attribut?

- Une valeur est le produit d'une évaluation
- Cette évaluation peut se concevoir distinctement de la *description* de cette valeur

Une valeur doit-elle être une entité?

- En MDSL... non!

«Une valeur n'est pas l'éventuelle entité qu'elle réfère»

# ***Gestion de la valeur***

- Si la valeur d'un attribut est *modifiable*, la valeur en elle même ne l'est pas.
- S'il est possible de distinguer un *objet* valeur par son type non modifiable (voir *Jvalues*), en MDSL une valeur est *décrite*
- le couple (attribut, valeur) devient:  
(nom de l'attribut, *description* de la valeur)

# Gestion de la valeur

- La *responsabilité* de l'interprétation de cette *description* en incombe à son utilisateur, que celui-ci soit un programme ou un humain
- En MDL, cette *description* doit être *interprétée* par un code qui l'utilise. Ce code est écrit dans un langage de programmation support

« *La description n'est pas l'évaluation* »

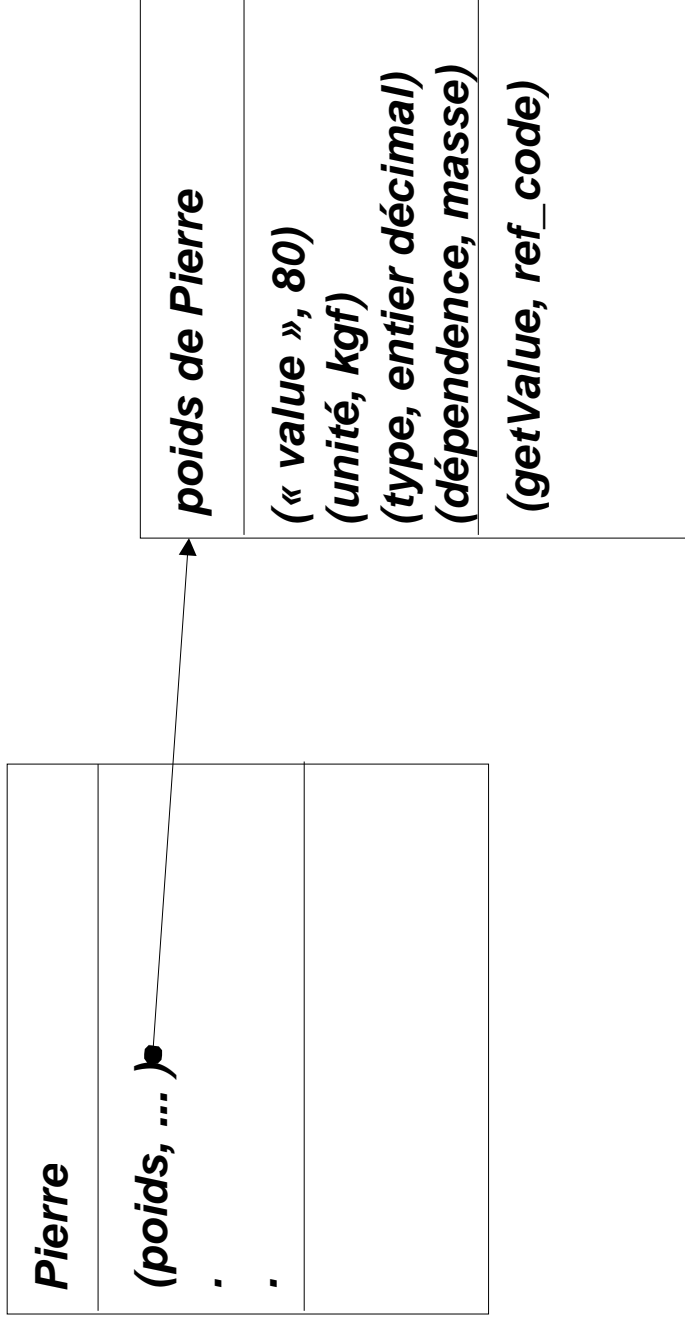
# Exemple 1

- Pierre nomme l'entité
- poids nomme un attribut
- 80 décrit la valeur de l'attribut 80 de poids

<i>Pierre</i>
<i>(poids, 80)</i>
.
.

# Exemple 1

- Qu'est-ce qu'un attribut?



# Exemple 2

- Cas particulier

<i>Pierre</i>	
<i>(conjoint, Claire)</i>	
.	.
.	.

# Exemple 2

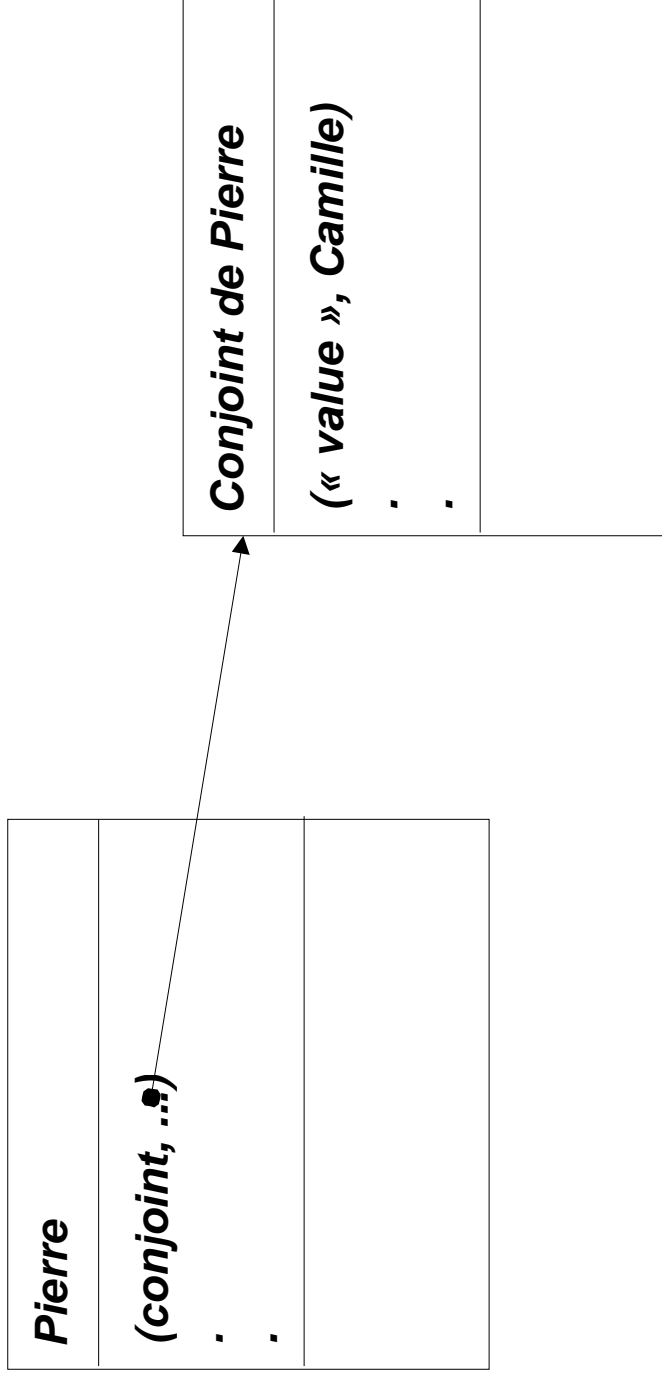
- « La valeur n'est pas l'entité qu'elle réfère »
- *Conjoint de Pierre* n'est pas *Claire* mais *Claire*

<i>Pierre</i>
( <i>conjoint, .•</i> )
.
.

<i>Conjoint de Pierre</i>
(« <i>value</i> », <i>Claire</i> )
.
.

# Exemple 2

- Si Pierre change de conjoint...



## ***Exemple 2***

- Claire ne se change pas en Camille!
  - surtout si Camille n'est pas une femme ;--)
- L'énoncé de la description (la valeur de la description) permet de construire UNE valeur pour celui qui l'utilise (code ou utilisateur)
- Est-ce que c'est clair?
  - Ben non c'est Camille! ;--)

# *Perspectives théoriques*

- Explicitation de la sémantique d'une représentation donnée (classification, points de vue, perspectives, etc.)
- Explicitation de la sémantique → dialecte(s)
- Clarification => compliquer ou complexifier?
- Utiliser des *connaissances représentées*
- LPO/RCO: une approche langage avant d'être architecturale

# ***Perspectives appliquées***

- Plateforme(s)... mais vers quels objectif(s) théorique(s)?
- La réification de méthode pour les implémentations ouvertes → « design space »
- Exploration–expérimentation de l’héritage dynamique
- ...

# ***Conclusion***

- Les perspectives sont clairement exploratoires
- L'investissement n'est pas négligeable
- Ça n'intéresse personne
- Alternatives?
- De *bonnes* raisons de laisser tomber