

Introduction & Fondements

● Préoccupations

- Etablir facilement un logiciel concerné

● Fondements

- Paradigme objet et notation UML

► Moyen de synthèse de logiciels puissant

- Ingénierie Dirigée par les Modèles (IDM)

► Processus de développement qui place les modèles au cœur du processus de développement

Paradigme objet

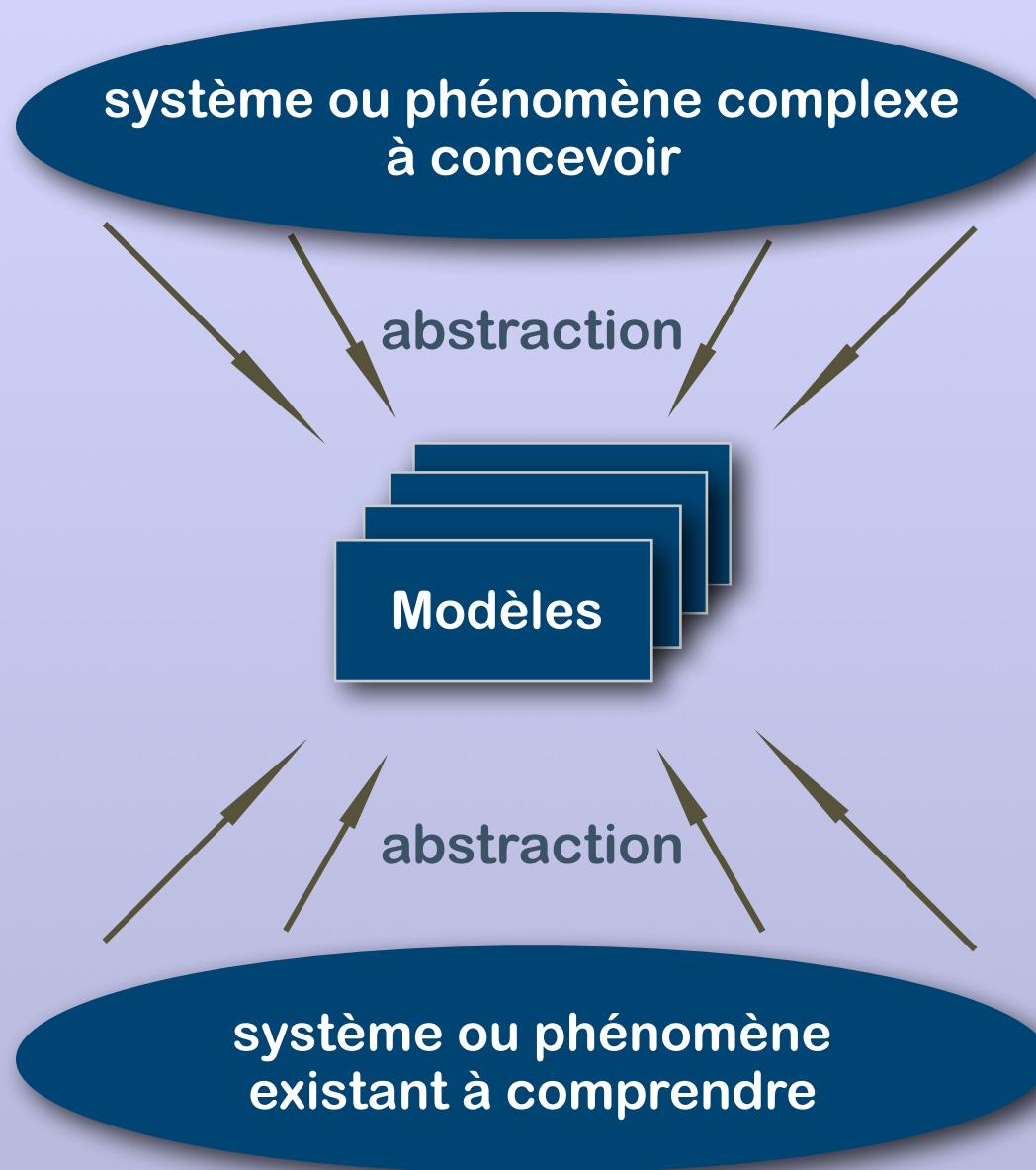
Design Patterns (Micros architectures de conception)

- Solutions génériques pour problèmes récurrents.
- Favorisent l'établissement et la réutilisation d'architectures logicielles éprouvées

Frameworks

- Applications semi-finies fournissant des cadres génériques spécialisables pour la réalisation de partie ou d'application ou d'application
- Ils automatisent en partie la construction des logiciels.

IDM, une approche modèle



Ingénierie des modèles

Modélisation (abstraire)

Architecture (organiser)

Intégration de modèles

Transformation de modèles

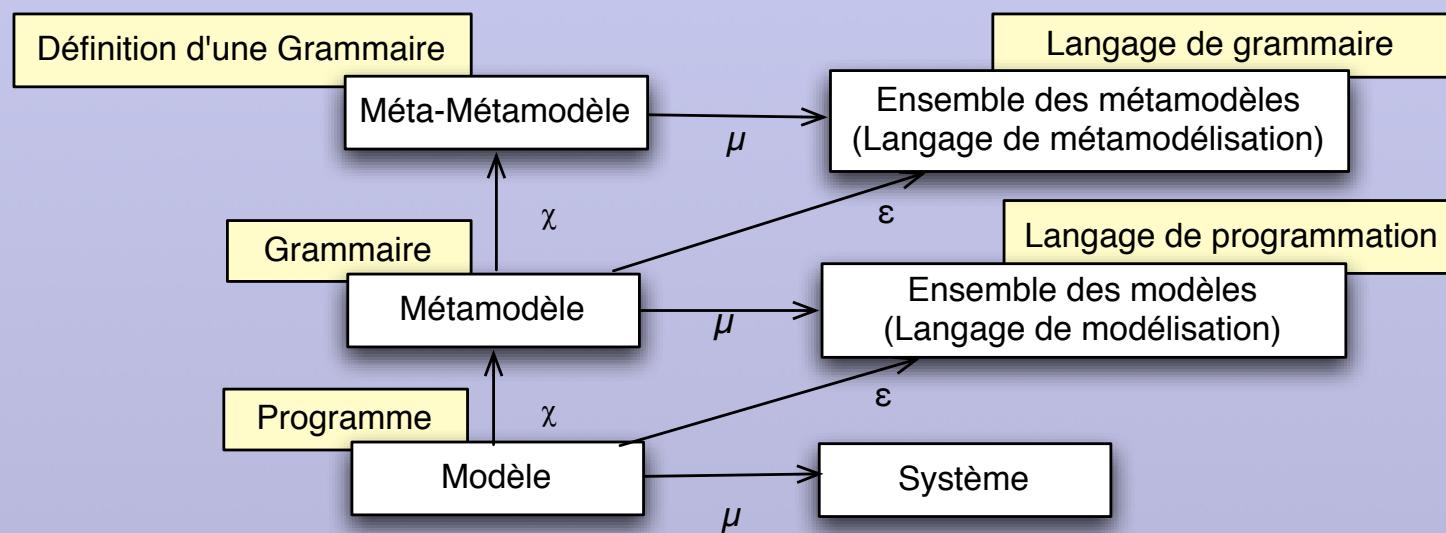
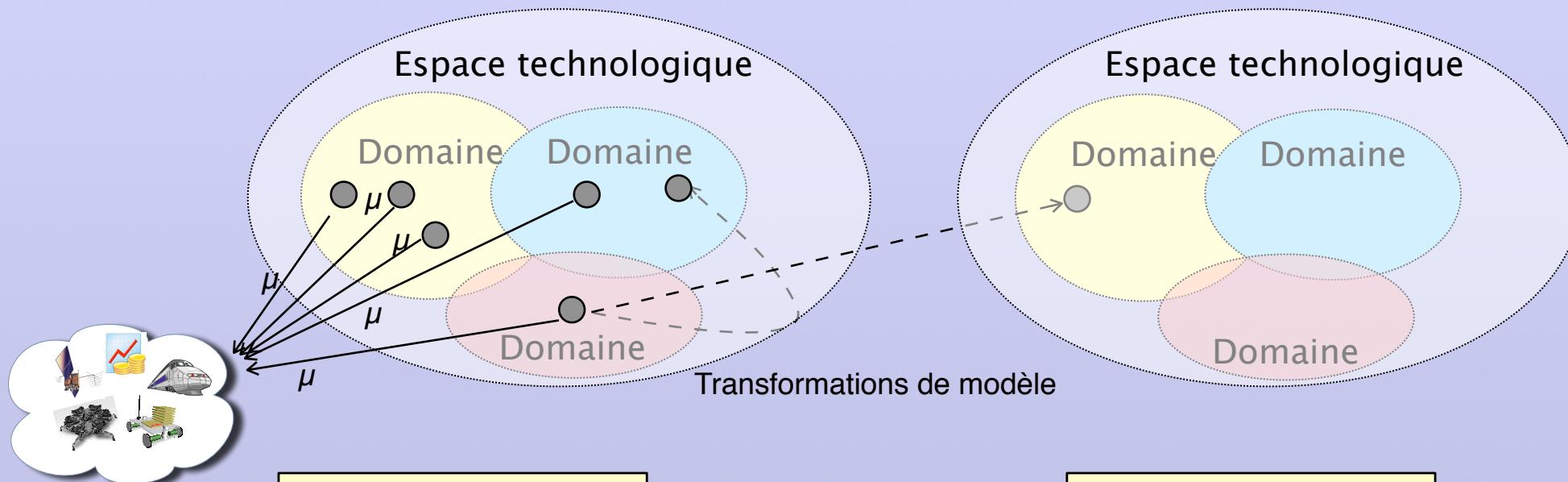
Validation et simulation

Vérification (robustesse)

Implantation (réalisation)

Ingénierie Dirigée par les Modèles (IDM)

Modèle - Métamodèle - Langage de modélisation - Transformations



Métamodèle et Framework

● MétaModèle

- Il décrit l'ensemble de tous les modèles possibles
- Il décrit toutes les configurations possibles réalisables à partir des concepts présents.
- Il définit la syntaxe à partir de laquelle les modèles sont construits.

● Framework

- On peut sans trop de risque affirmer qu'un Framework est un métamodèle dont la sémantique opérationnelle est donnée dans le code de réalisation.

Framework Flou

Objectif

- Etablir Framework générique pour les systèmes d'inférence flous.

La philosophie

- Les systèmes seront exprimés par le langage support (c++, java, ...)

- Portabilité.
- La puissance du langage est conservée.
- L'expertise du domaine de la logique floue doit être alors capturée par le Framework.
- Les utilisateurs manipuleront alors de nouveaux idiomes.

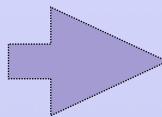
Le Framework doit:

- Etre flexible et extensible.
- Fournir des opérateurs polymorphes

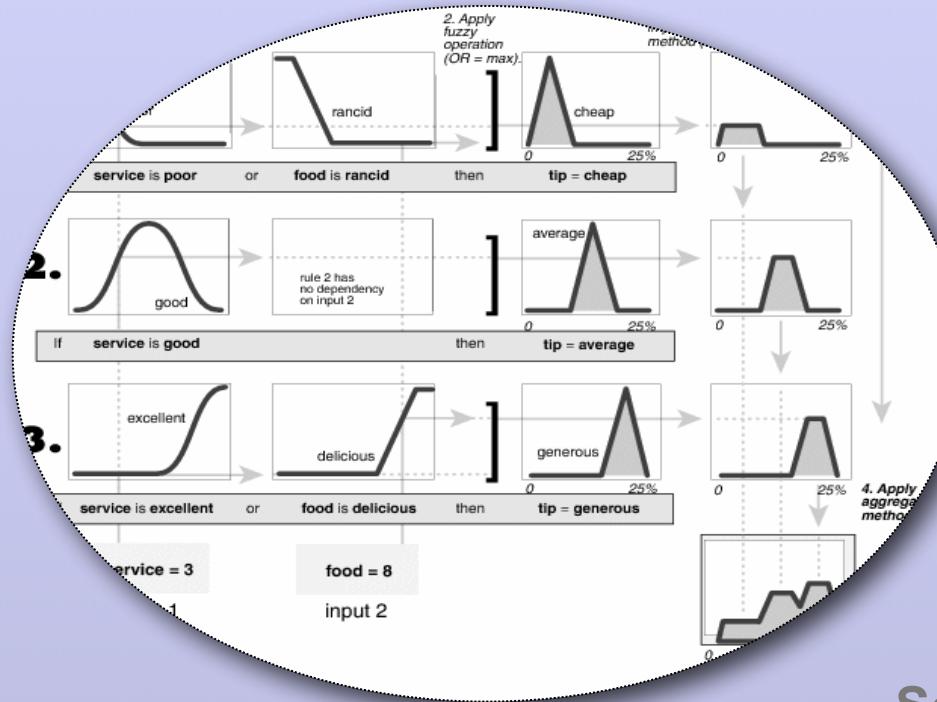
- L'influence de la nature de chaque opérateur doit pouvoir être étudiée sans pour autant refaire un nouveau système.



Une intention



Définir un langage spécifique dédié au domaine de la logique floue



Concepts du domaine

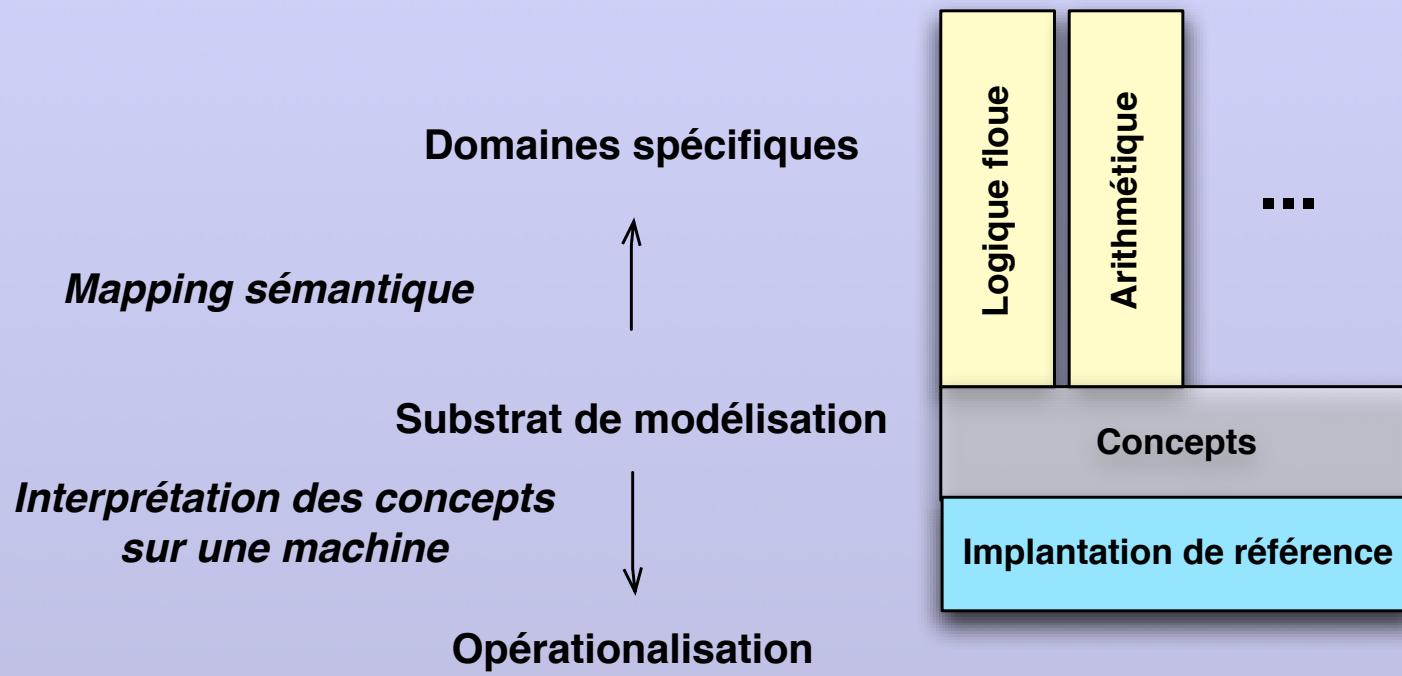
Syntaxe abstraite

Sémantique

statique

dynamique

Un point de vue



- **Un système flou classique doit être étudié**

- **Les concepts pertinents doivent être identifiés**

- Effectuer en quelque sorte un instantané du domaine du problème.

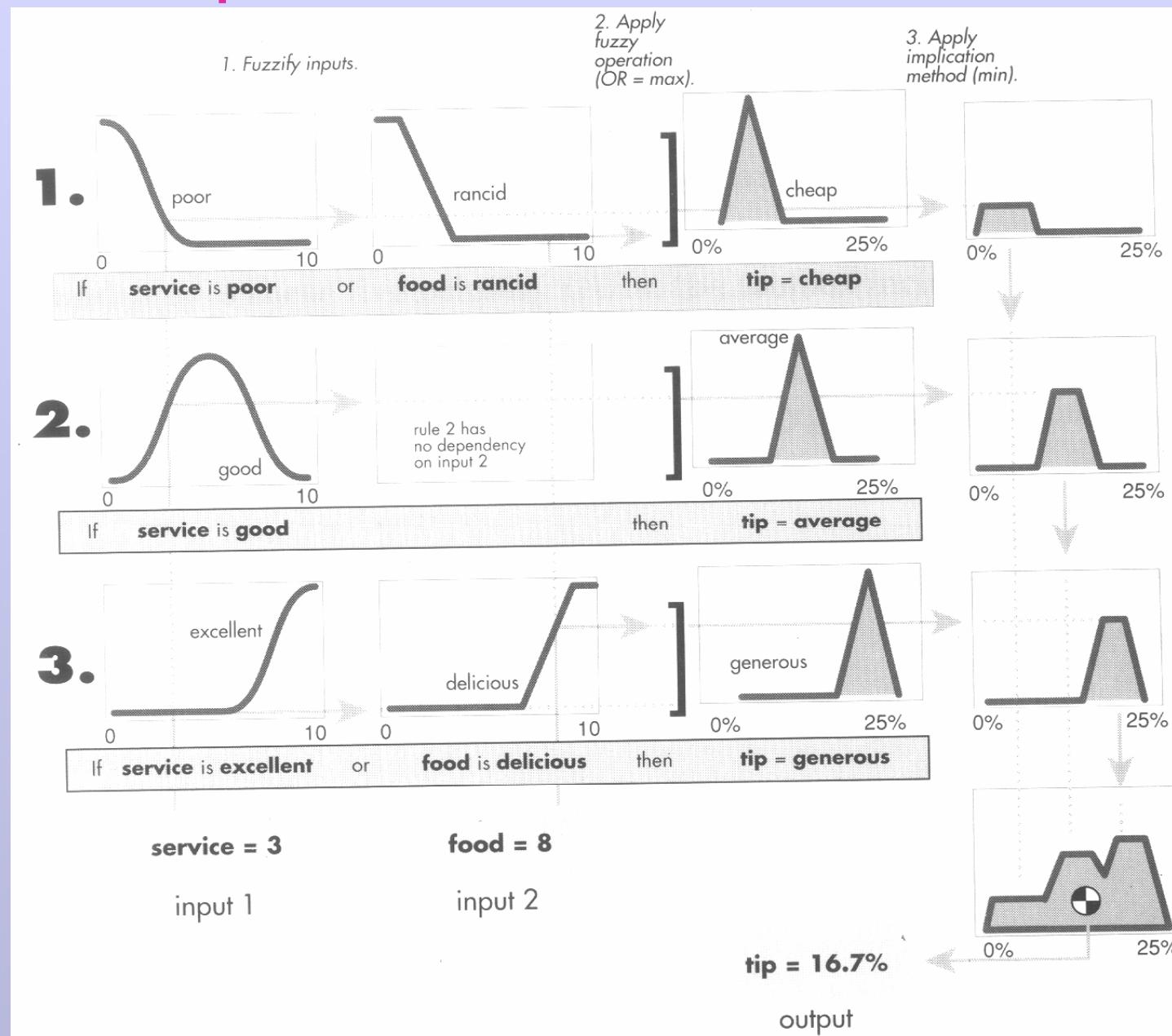
- **Un framework de logique floue doit être développé**

- Etablir le métamodèle de logique floue et une sémantique opérationnelle
 - Il devra utiliser les éléments découverts afin de faciliter la tâche du développeur (du système flou).
 - Il doit permettre la mise en œuvre aisée d'un système flou par sa seule *spécification*.



Systèmes Flous

Calcul de pourboire



Concepts:

● Des variables,

- Des fonctions d'appartenance,
- Des opérateurs flous (and, or , not, then, aggregation, defuzzification).

● Deux types de système flous:

- Mamdani,
- Sugeno.
- La différence réside dans l'implication et la méthode de défuzzification.

Concepts identifiés

Expressions

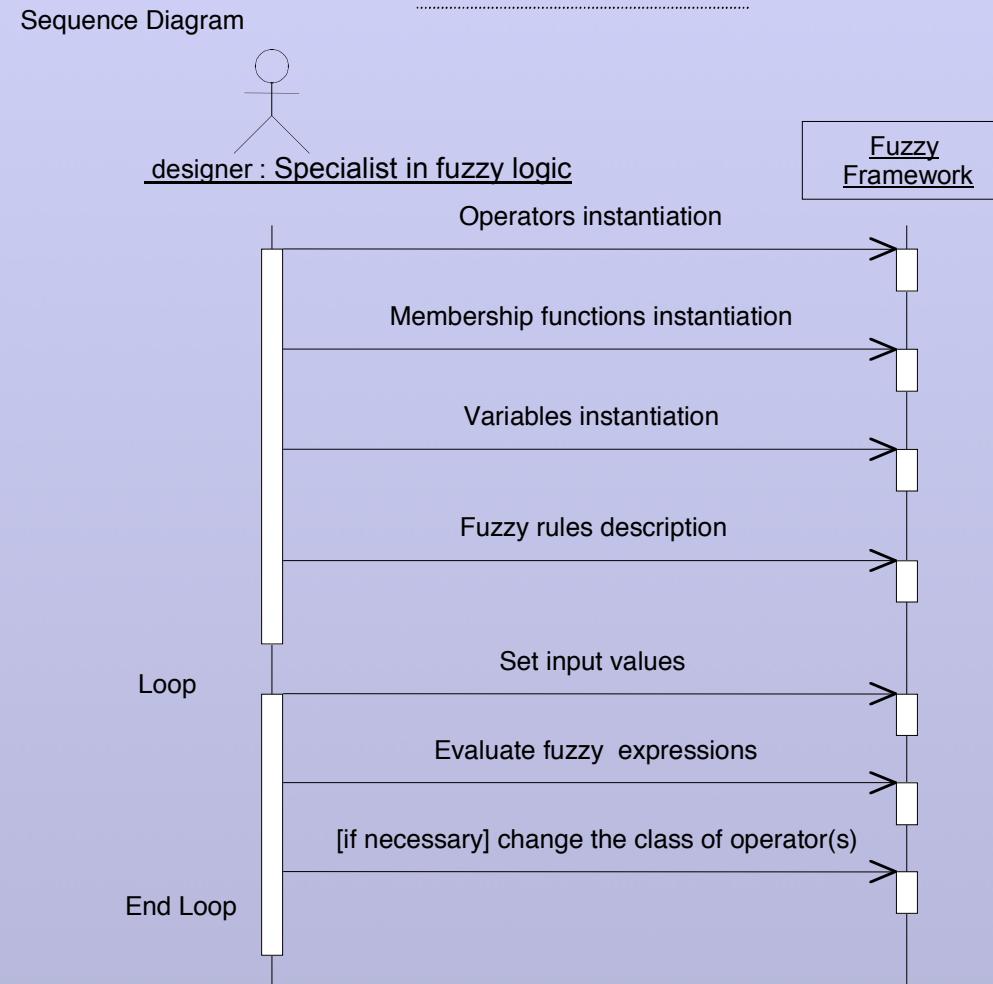
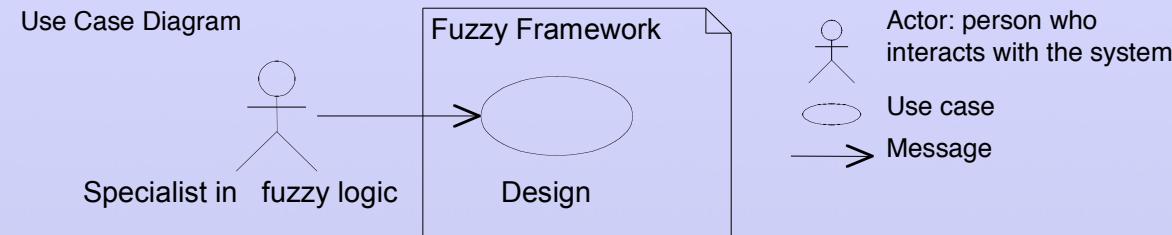
- Règles floues sugèrent des expression floues.
- Les valeurs et opérateurs ressemblent à ceux impliqués dans des expressions arithmétiques.

Les différents types d'entités (expressions):

```
If x is "low" and y is "medium" then w is "low"  
If x is "low" or y is "high" then w is "high"
```

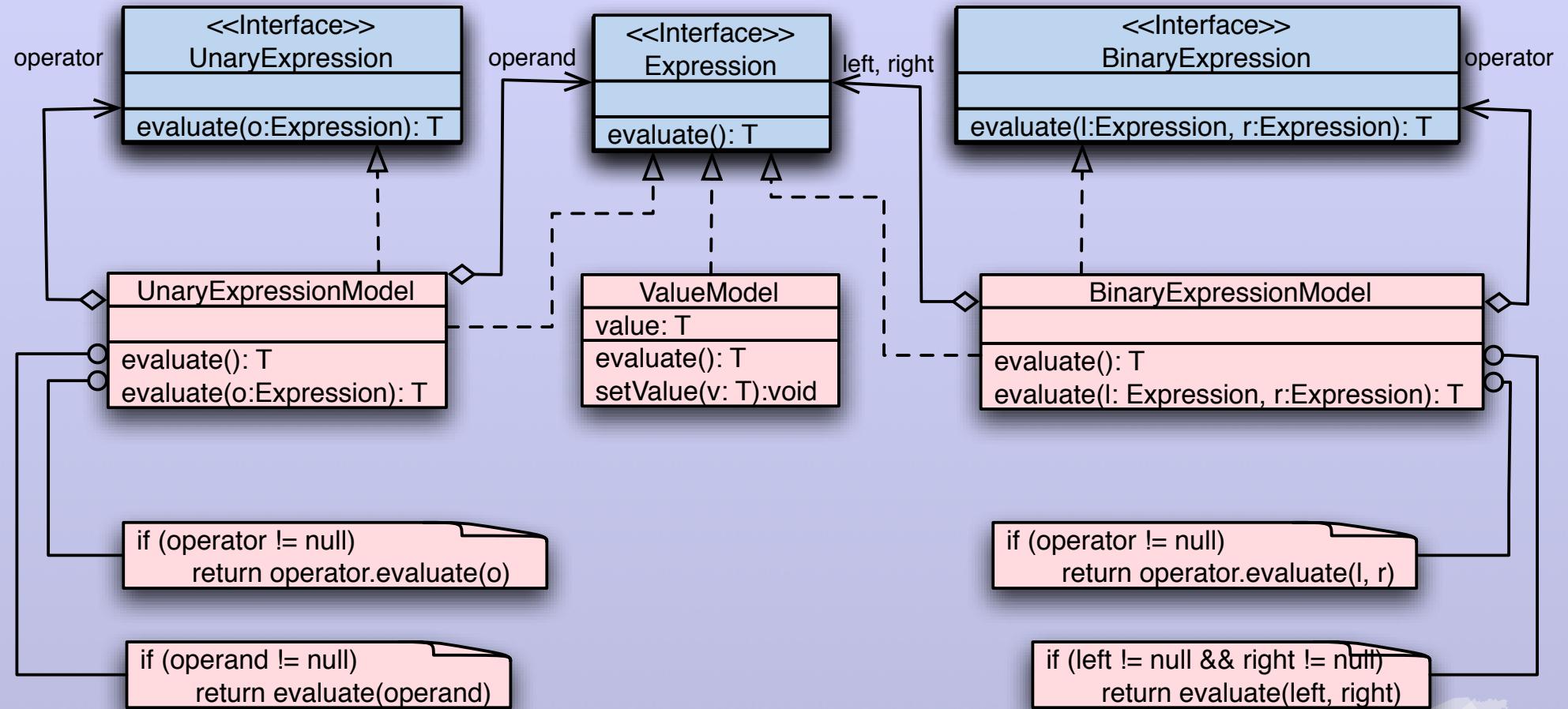
- Variables: x, y, w
- Fonctions d'appartenance : "low", "medium", "high"
- Opérateurs flous : and (T-norm operator), or (T-co norm operator), then, aggrégation.
- Opérateur de défuzzification.

Cas d'utilisation



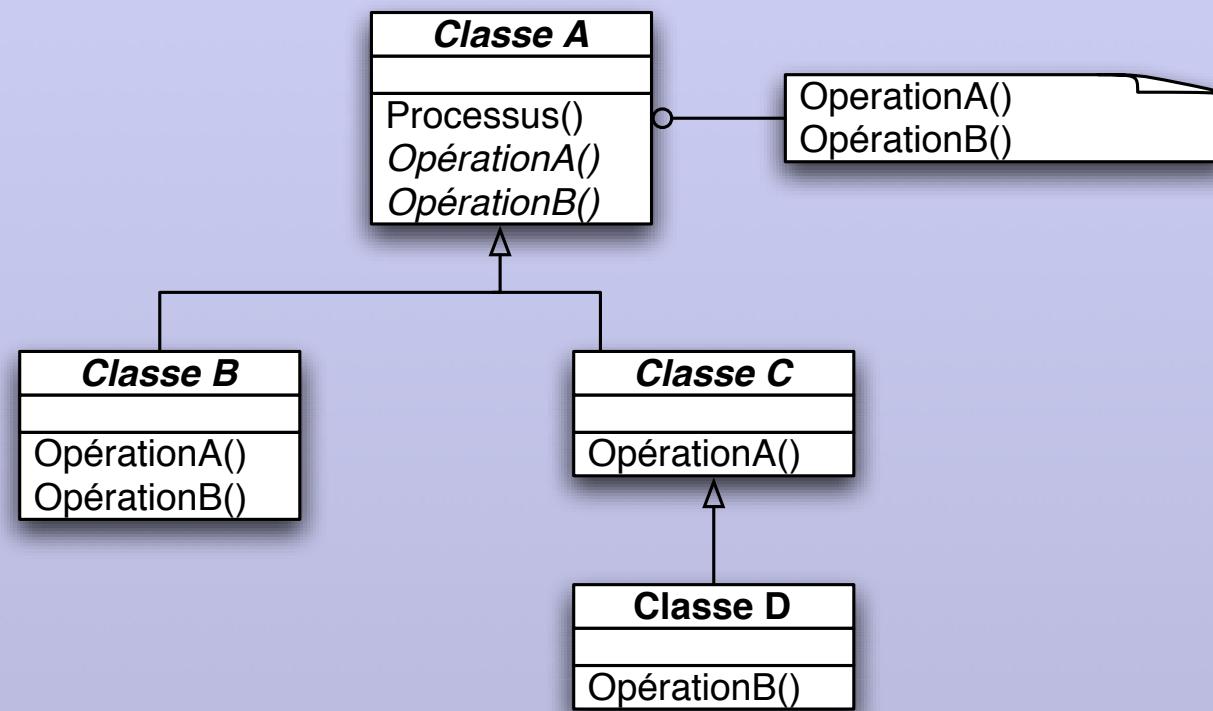
Métamodèle

Concepts



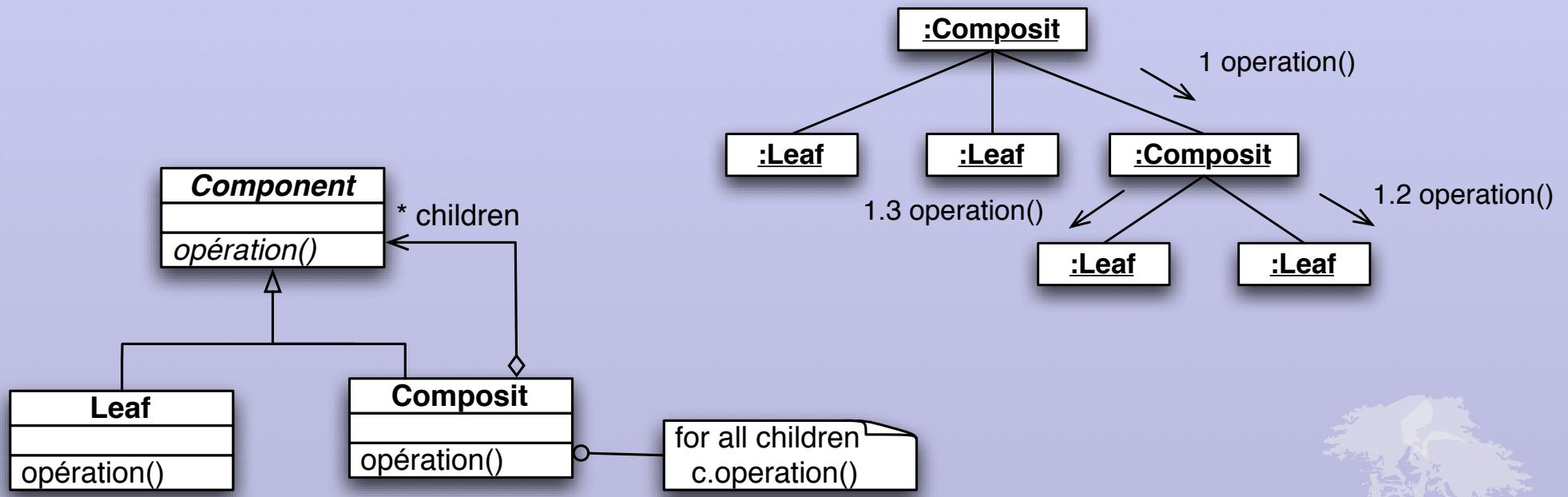
Template Method

- Les parties spécifiques de l'algorithme sont déléguées aux sous-classes.
- Séparation de deux points de vue



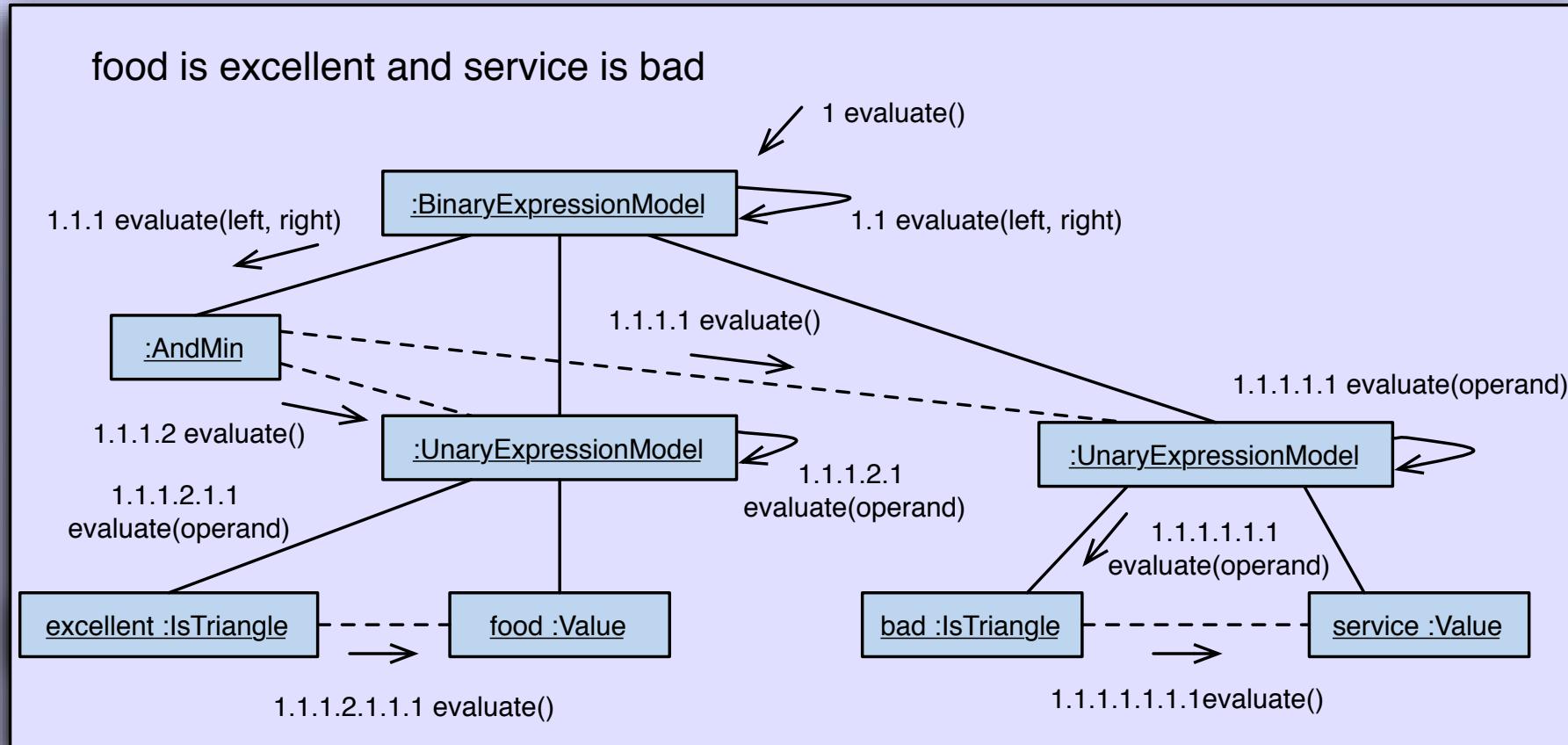
Composit & Interpret

- Une structure d'expressions hiérarchique.
- Chaque nœud de la structure est traitée uniformément.
- Chaque expression est évaluée.
- L'arbre est évalué récursivement.





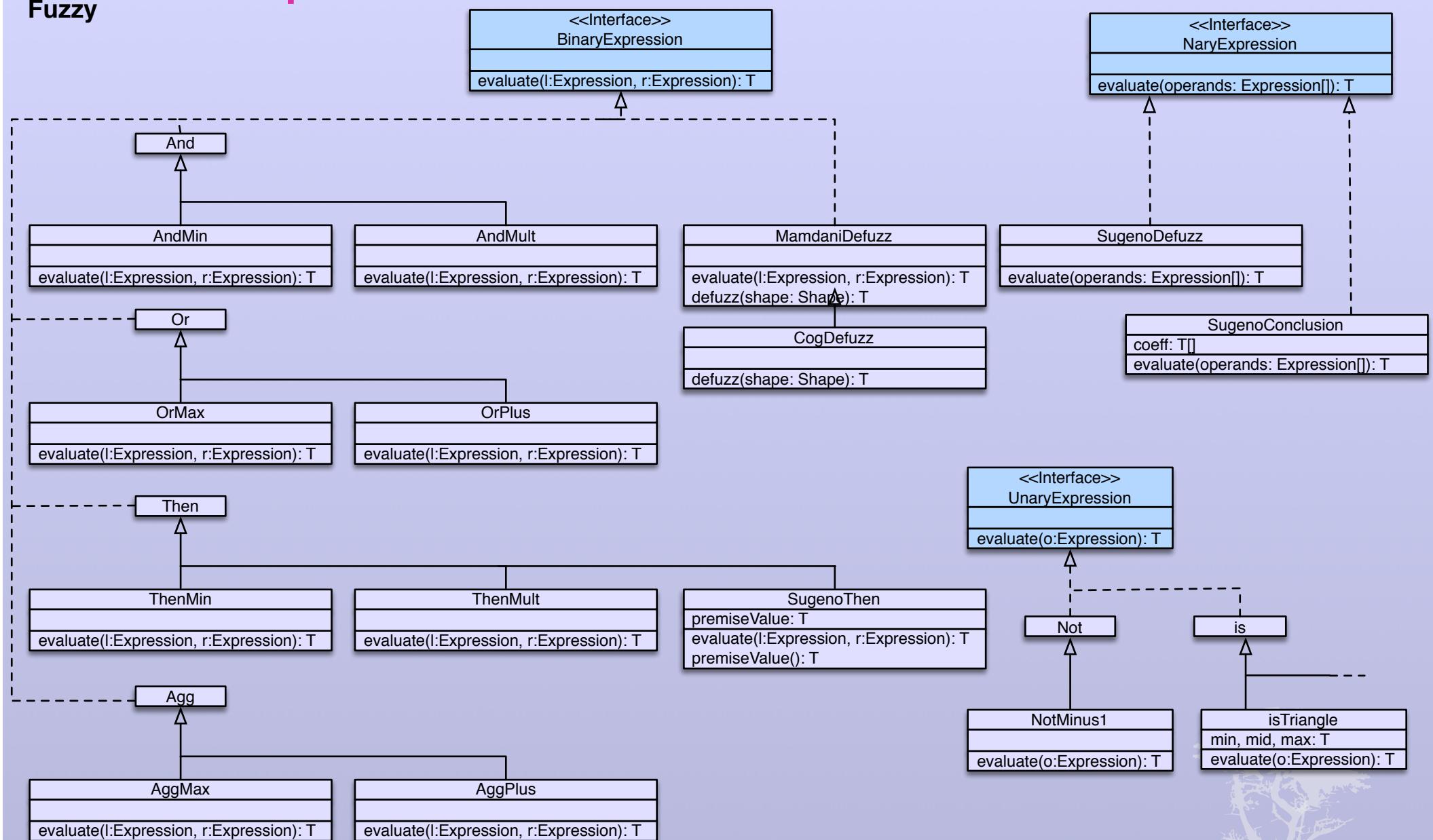
Un modèle





Sémantique floue

Fuzzy



Comportements polymorphes

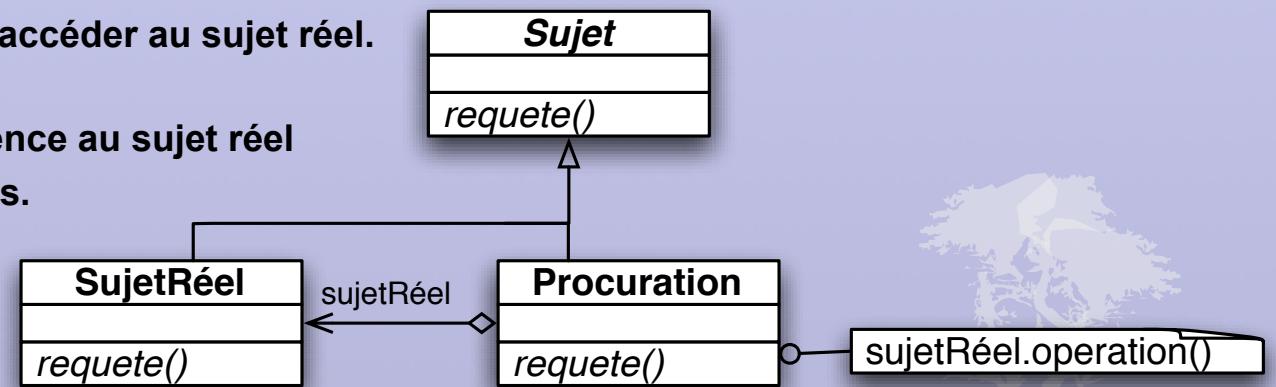
Shadow expressions: UnaryShadowExpression, BinaryShadowExpression

- L'évaluation polymorphe nécessite que chaque opérateur doit apparaître comme si il changeait de classe (méta modélisation).
- A shadow expression joue le rôle d'un opérateur.
- Il maintient un lien vers l'opérateur réel.
- Le lien peut être reconnecté à chaque instant vers n'importe quel opérateur.

Design Pattern

Procuration

- Gère une référence qui permet d'accéder au sujet réel.
- Une procuration peut faire référence au sujet réel si les interfaces sont compatibles.

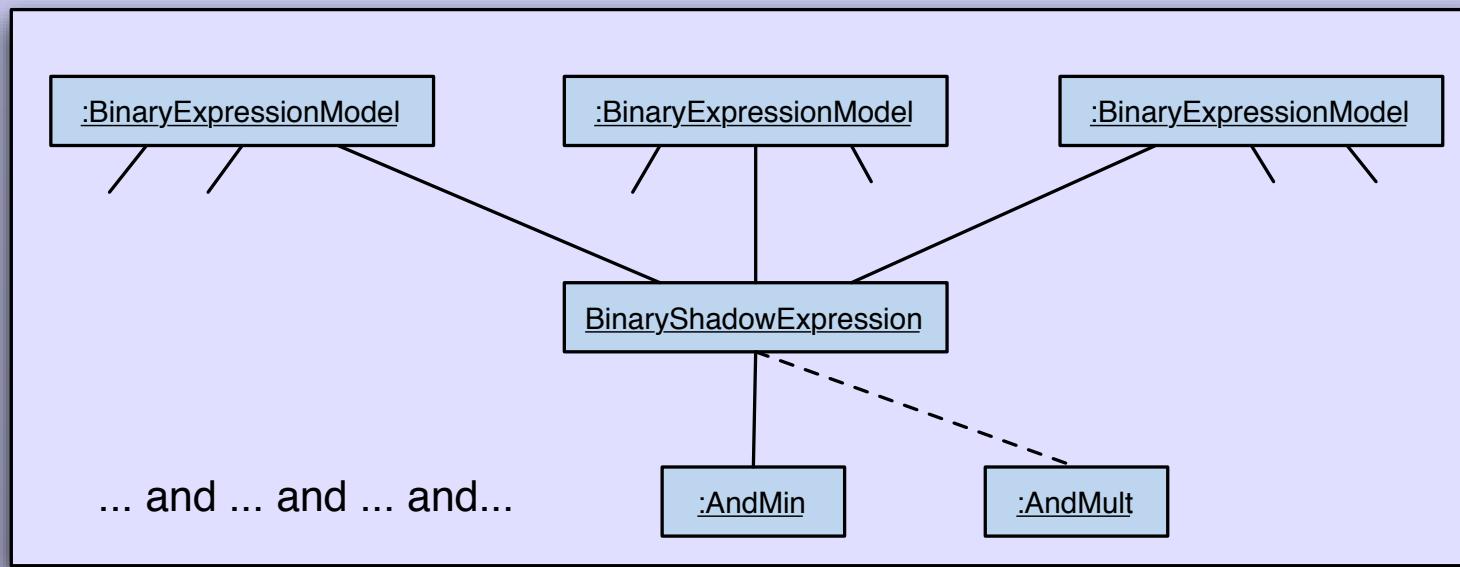


Comportements

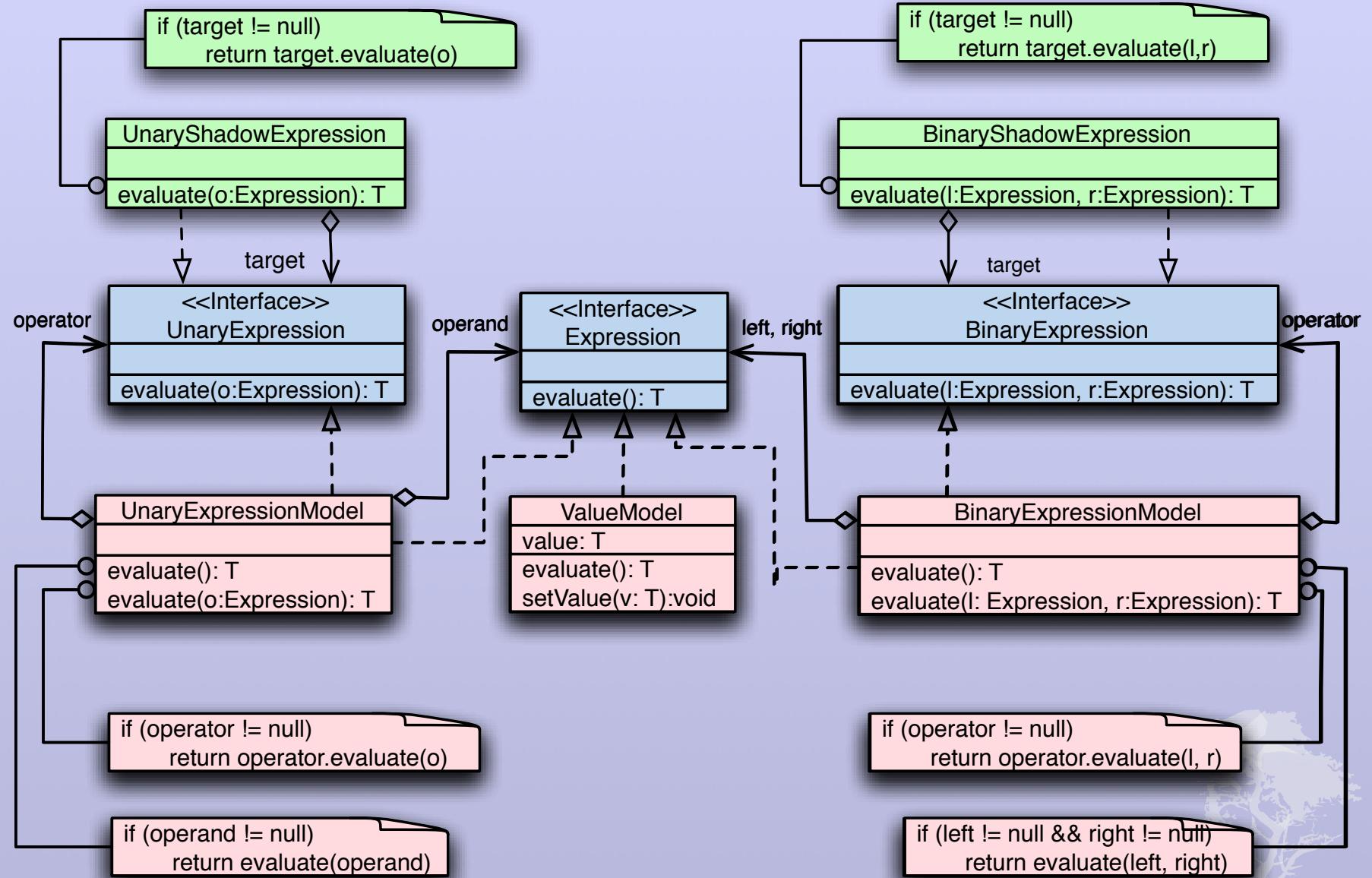
Shadow expressions: UnaryShadowExpression, BinaryShadowExpression

- L'évaluation polymorphe nécessite que chaque opérateur doit apparaître comme si il changeait de classe (méta modélisation).
- A shadow expression joue le rôle d'un opérateur.
- Il maintient un lien vers l'opérateur réel.
- Le lien peut être reconnecté à chaque instant vers n'importe quel opérateur.

Nouveau modèle

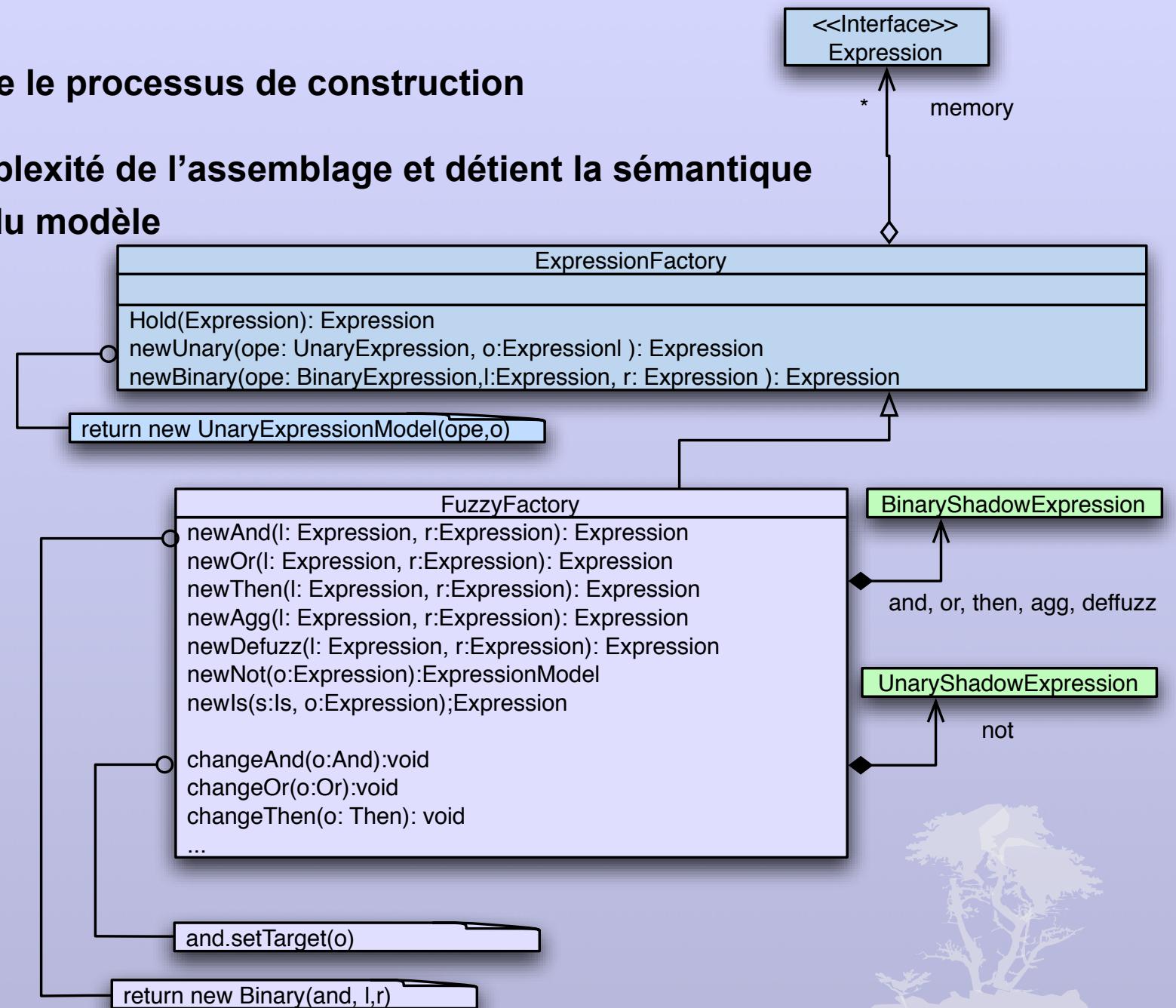


Evolution



Factory

- Prend en charge le processus de construction
- Masque la complexité de l'assemblage et détient la sémantique d'assemblage du modèle



Implantation de l'exemple simplifié

```

//operators
NotMinus1 opNot;
AndMin opAnd;
OrMax opOr;
ThenMin opThen;
CogDefuzz opDefuzz;

//fuzzy expression factory
FuzzyExpressionFactory f(&opNot,&opAnd,&opOr,&opThen,&opOr,&opDefuzz);

//membership function
IsTriangle poor(-5,0,5);
IsTriangle good(0,5,10);
IsTriangle excellent(5,10,15);

IsTriangle cheap(0,5,10);
IsTriangle average(10,15,20);
IsTriangle generous(20,25,30);

//values
Value service(0);
Value food(0);
Value tips(0);

```

Expression *r =

```

f.NewAgg(
  f.NewAgg(
    f.NewThen(
      f.NewIs(&service,&poor),
      f.NewIs(&tips,&cheap)
    ),
    f.NewThen(
      f.NewIs(&service,&good),
      f.NewIs(&tips,&average)
    )
  ),
  f.NewThen(
    f.NewIs(&service,&excellent),
    f.NewIs(&tips,&generous)
  )
);

```

//defuzzification

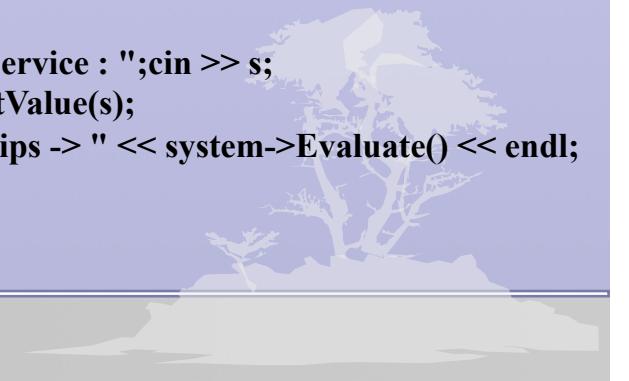
Expression *system = f.NewDefuzz(&tips, r, 0, 25, 1);

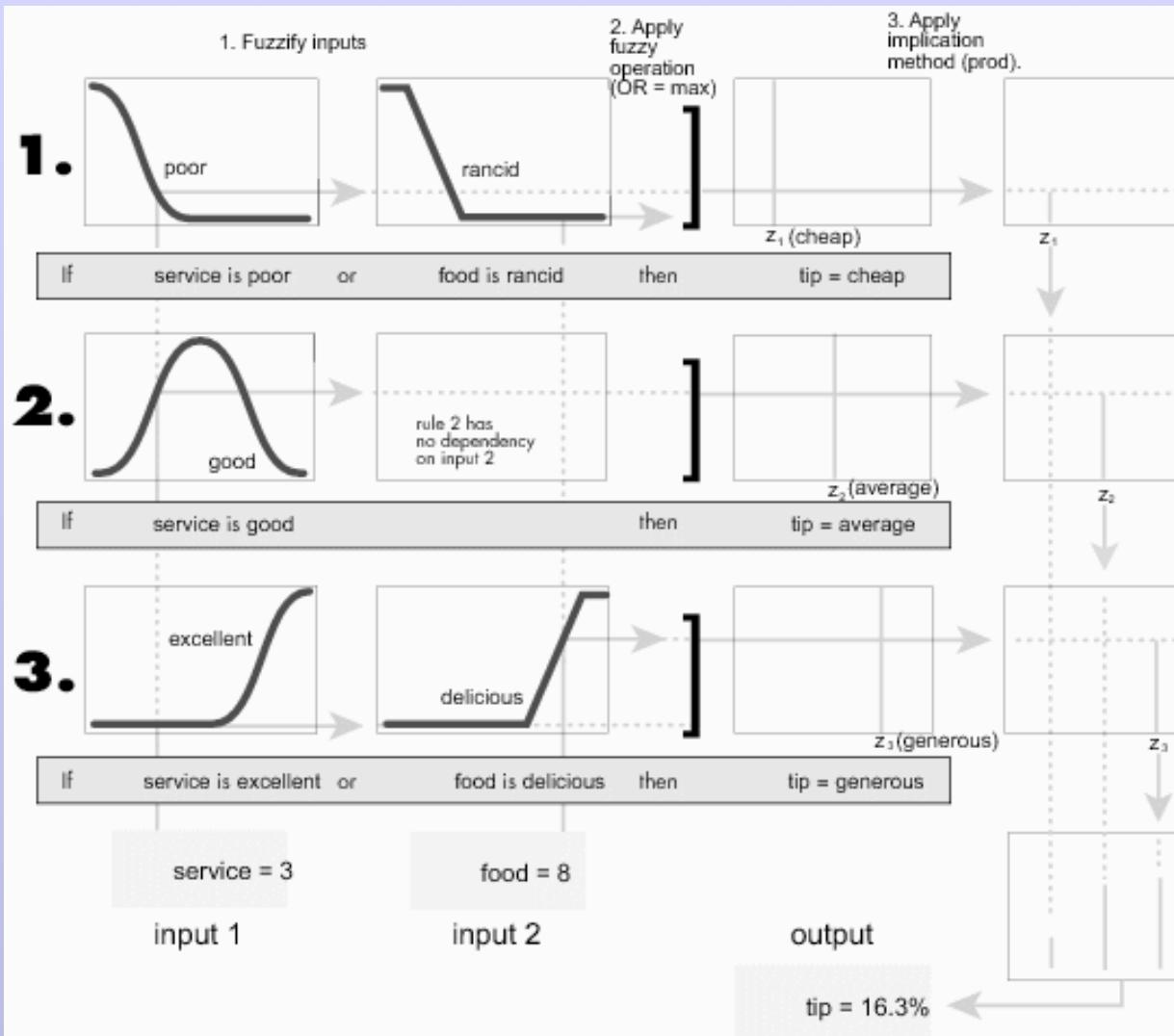
//apply input

```

float s;
while(true)
{
  cout << "service : ";cin >> s;
  service.SetValue(s);
  cout << "tips -> " << system->Evaluate() << endl;
}

```





$$\text{Final Output} = \frac{\sum_{i=1}^N w_i z_i}{\sum_{i=1}^N w_i}$$

wi: premise as Mamdani system

zi: conclusion

If Input 1 = x and Input 2 = y, then $z = ax + by + c$

Sémantique floue

Fuzzy

