Profils UML 2 et projections

© Olivier Caron

12 septembre 2007

LIFL

Plan

√ UML et MDA : points communs
- ▶ Technique des profils

√ Transformation de modèles et profils UML
√ UML 2 : introduction de composants PIM
√ Application MDA : le projet RNTL ACCORD

MDA et UML

√ Ce sont deux normes délivrées par l'OMG
√ MDA : ingénierie des modèles
√ UML : notation graphique de spécification de modèles
√ UML peut être utilisé comme support du MDA :
- ▶ Nécessaire de spécifier des modèles UML PIM
- ▶ mais aussi des modèles UML PSM !
- ▶ Transformation de modèles UML ?

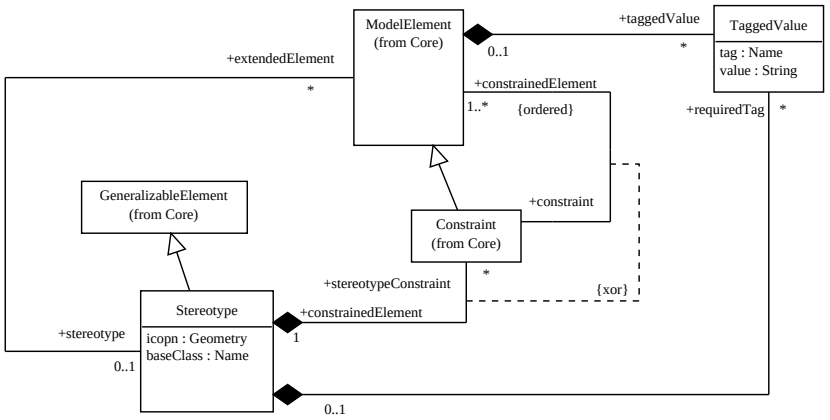
Les technologies préconisées par MDA

√ UML (from OMG)
√ MOF (Meta Oject facilities)
√ profils UML
√ OCL (Object Constraint Language)
√ XML, XMI

Qu'est-ce qu'un profil UML ?

- √ Mécanisme `standard` d'extension d'UML
- √ Ensemble cohérent de stéréotypes, valeur marquées (tagged value) et contraintes
- √ Principe général : on n'ajoute pas de méta-classes mais des annotations aux méta-classes UML existantes.

Mécanisme d'extension UML



Exemple construction profil UML (1/3)

- √ Objectif : créer un profil UML permettant de décrire des bases de données relationnelles :
  Une base de données contient un ensemble de tables. Une table contient un ensemble de colonnes, un sous-ensemble de ces colonnes forme la clé primaire de la table.

Exemple construction profil UML (2/3)

- √ Phase 1 : attacher les concepts du modèle au modèle UML
- √ Profil Bases de données :

Concept modèle	Méta-classe UML	Extension
Base	Package	Stéréotype
Table	Class	Stéréotype
Colonne	Attribute	
ClePrimaire	Attribute	Stéréotype

Exemple construction profil UML (3/3)

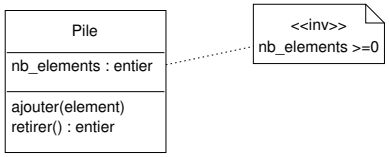
√ Phase 2 : Donner les règles de cohérence d'utilisation du profil
√ Utilisation du langage OCL (Object Constraint Language)

Le langage OCL (1/3)

√ Langage formel, textuel et normalisé (OMG).
√ Des parsers OCL existent
√ Complémentaire à la notation UML
√ Permet de spécifier des contraintes sur tout schéma UML
√ UML est lui-même spécifié avec OCL (méta-modèle).
√ Devoir : étudiez le langage avec la référence :
http ://www.ibisc.univ-evry.fr/ djafri/SupportsCours/ocl.pdf

Le langage OCL (2/3)

√ Liè à un *contexte* : précise le type de l'instance auquel la contrainte se rapporte
√ Les *stéréotypes suivants* sont applicables à une contrainte : `inv, pre, post`
√ Exemple :

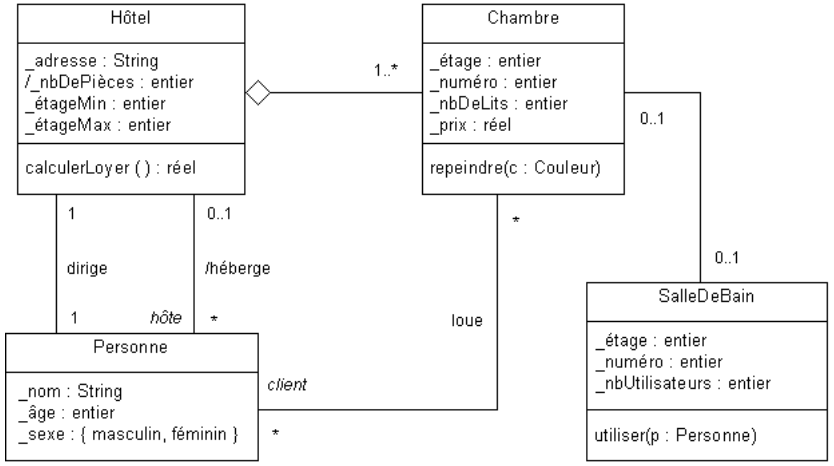


context Pile inv :
nb_elements >=0

Le langage OCL (3/3)

√ Parcours simple d'un schéma UML (via les associations)
√ Expressions arithmétiques, booléennes
√ Conformité des types (`oclIsTypeOf()`)
√ Expressions ensemblistes (`collect, select, forAll`,. . .)

Exemple OCL, source JOOP 1999, uml.free.fr



Contraintes OCL (1/3)

√ Un hôtel ne contient jamais d'étage numéro 13 (superstition oblige).

```
context Chambre inv:
self._étage <> 13
```

√ Le nombre de personnes par chambre doit être inférieur ou égal au nombre de lits dans la chambre louée. Les enfants (accompagnés) de moins de 4 ans ne "comptent pas" dans cette règle de calcul (à hauteur d'un enfant de moins de 4 ans maximum par chambre).

```
context Chambre inv:
  client->size <= _nbDeLits or
  (client->size = _nbDeLits + 1 and
                  client->exists(p : Personne | p._age < 4))
```


Contraintes OCL (2/3)

√ L'étage de chaque chambre est compris entre le premier et le dernier étage de l'hôtel.

```
context Hôtel inv:
 self.chambre->forAll(c : Chambre | c._étage <= self._étageMax and
                                    c._étage >= self._étageMin)
```

√ Chaque étage possède au moins une chambre (sauf l'étage 13, qui n'existe pas...).

```
context Hôtel inv:
 Sequence{_étageMin.._étageMax}->forAll(i : Integer |
   if i <> 13 then
     self.chambre->select(c : Chambre | c._étage = i)->notEmpty)
   endif)
```


Contraintes OCL (3/3)

√ On ne peut repeindre une chambre que si elle n'est pas louée. Une fois repeinte, une chambre coûte 10 pour cent de plus.

```
context Chambre::repeindre(c : Couleur)
pre: client->isEmpty
post: _prix = _prix@pre * 1.1
```

√ Une salle de bain privative ne peut être utilisée que par les personnes qui louent la chambre contenant la salle de bain et une salle de bain sur le palier ne peut être utilisée que par les clients qui logent sur le même palier.

```
context SalleDeBain::utiliser(p : Personne)
pre: if chambre->notEmpty then chambre.client->includes(p)
     else p.chambre._étage = self._étage
     endif
post: _nbUtilisateurs = _nbUtilisateurs@pre + 1
```

Profil Base de données (suite)

√ Quelques règles du profil Bases de données :

```
une table ne contient que des colonnes
Context Class : self.feature->forAll(oclIsKindOf(Attribute))
une table doit posséder une clé primaire
Context Class : self.feature->select(isStereotyped(''ClePrimaire''))
                .notEmpty
...
```

√ Beaucoup d'autres règles (un package ne contient que des tables, pas d'héritage de table,. . .)

Exemple profil base de données

√ Exemple :

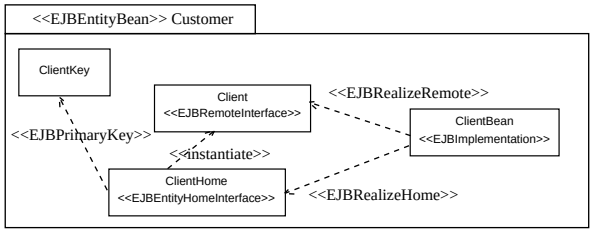
√ Compléter avec des outils automatiques, exemple génération des commandes de création de tables SQL

Exemple profils standards UML

√ Des profils standardisés existent : profil CORBA 2, profil EJB 1.1

√ En cours : profil CCM (Thales, Alcatel, LIFL, . . .)

√ A venir : profil EJB 2.0



Profils UML versus MOF

√ Inconvénients :

- Obligation de se rattacher au méta-modèle UML
- Limitation du mécanisme d'extension
  exemple : impossible de créer une nouvelle méta-classe

√ Avantages :

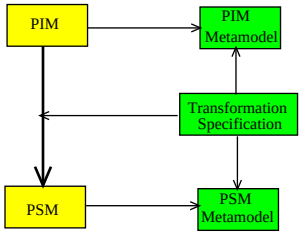
- Utilisation de la notation graphique UML
- Utilisation des ateliers UML existants
- Utilisation des différentes technologies UML associées (XMI, OCL, . . .)

Avantages Atelier UML

√ Spécification plus rapide d'un composant
√ Outil intégré de vérification structure d'un composant
√ Outil de génération de code, de transformation de modèles

Transformation de modèles UML (1/2)

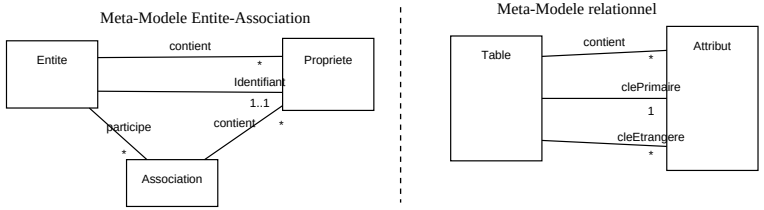
√ Approche par Méta-modélisation



√ Technologies possibles : XMI, XSLT, règles XSL, profils (UML PSM), MOF

Exemple transformation de modèles

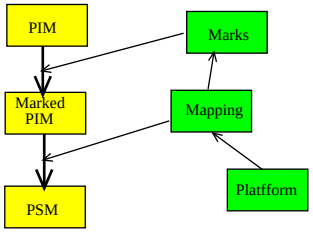
√ Passage modèle entité-association vers modèle relationnel



√ Identifier les règles de traduction :
ex : Entite → Table
identifiant → clePrimaire

Transformation de modèles UML (2/2)

√ Approche par annotations (marquage)



√ Technologies possibles : profils UML, XMI, règles XSL

UML, Processus MDA et composants

- √ Techniquement possible, architecture de profils UML
- √ Un modèle abstrait PIM : le modèle UML
- √ Des profils PSM UML pour les plates-formes de composants (profils EJB, profils CCM, etc)
- √ Difficulté principale : UML 1.4 ne dispose pas de la notion de composants logiques.

solution : UML 2.0

UML 2.0

- √ Norme sortie en juillet 2003
- √ Notion de composant logique
- √ Favoriser un processus MDA vers plates-formes de composants à partir d'un modèle UML 2

Composant UML 2.0

- √ Un composant UML 2 est une classe enrichie.
- √ Un composant UML 2 peut disposer de Ports
- √ Un port est associé à une ou plusieurs interfaces.
- √ Un port peut être requis ou fourni ou les deux.
- √ Des instances de composants (définis par des parts) peuvent se connecter via les ports
- √ Il est possible de définir des composants composites

Quelques remarques

- √ De nombreux éléments restent flous dans la spécification
- √ exemple : validité de l'assemblage ?
  Stricte compatibilité des interfaces requises et fournies ? Même nombre d'interfaces dans chaque port ? . . .
- √ Solution : proposer un profil avec règles de cohérence

Le projet RNTL ACCORD

√ Objectifs :
- ▶ Fournir un atelier de conception orienté composant
- ▶ Intégrer des outils de validation de l'assemblage de composants (contrats)
- ▶ Fournir des outils (automatiques ?) de génération vers des plates-formes technologiques de composants

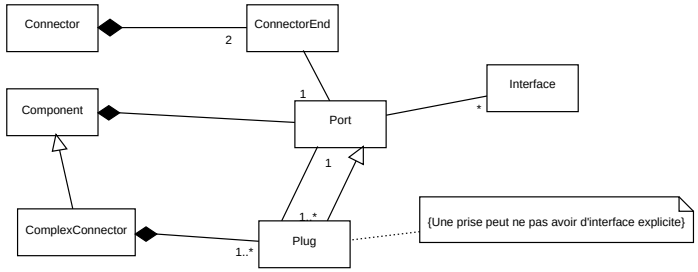
√ Partenaires : EDF, FT, Softeam, LIFL, INRIA Rennes, ENST Paris, ENST Bretagne, CNAM

√ Technologie : profils UML, atelier Objecteering UML 2

Le modèle abstrait de composants

√ Basé sur UML 2 (ensemble de ports, plusieurs interfaces par ports,. . .)

√ Notion de connecteur-complexe



Notion de contrats

√ Intégration de contrats d'assemblage au niveau du méta-modèle

√ Contrat syntaxique, Contrat sémantique (pré-post conditions), Contrat pragmatique (temps, QoS, . . .)

√ Profil UML : syntaxe XML stocké dans des notes UML

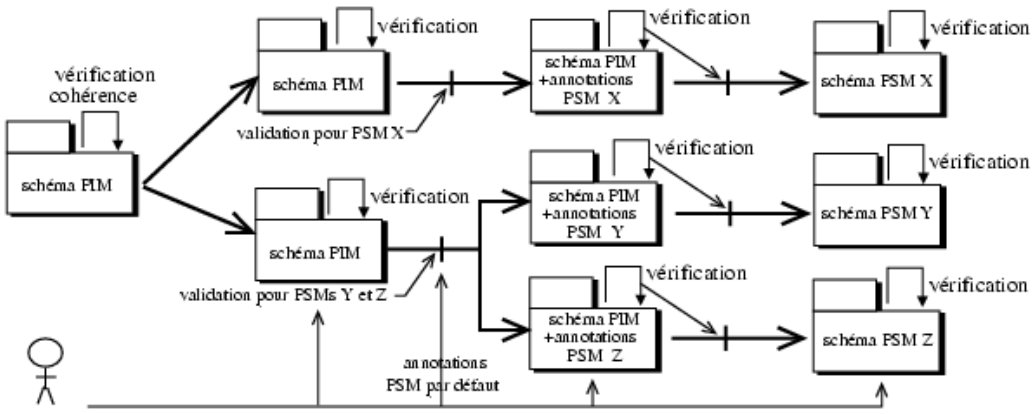
Transformations de modèles PIM vers PSMs

√ Objectif : à partir du modèle abstrait, générer les projections vers EJB et/ou CCM

√ Etude des correspondances ACCORD vers PSM :
- ▶ Un concept ACCORD se traduit en aucun concept PSM (jamais)
- ▶ Un concept ACCORD se traduit en un concept PSM (5%)
- ▶ Un concept ACCORD peut se traduire en plusieurs concepts PSM (95%)

√ Offrir une projection la plus automatique et la plus flexible possible.

Méthodologie de transformation retenue



Profil CCM

√ Modèle abstrait ACCORD génère le modèle abstrait CCM
√ Intégration d'outil pour le modèle de programmation CCM (CIDL)
√ Génération de code IDL et CIDL