A

Représentation UML avancée pour XML Schema

Nous présentons dans cette annexe les notations UML à utiliser en regard de tous les mécanismes autorisés mais avancés de XML Schema. Nous analysons chaque difficulté et expliquons comment la traiter. Nous avons baptisé « avancés » les mécanismes de XML Schema rarement utilisés et dont on peut même se demander s'ils doivent raisonnablement être représentés dans les vues UML du système.

Voici les mécanismes avancés de XML Schema dont on trouvera ici la correspondance UML :

- les attributs et les types listes et unions,
- les attributs et autres valeurs par défaut,
- la définition d'annotations,
- les groupes d'attributs,
- les contraintes d'exclusion et les groupes de choix,
- les contraintes de simultanéité et les groupes simples,
- la question de l'ordre des éléments,
- la mixité des modèles.

Attributs et types listes de XML Schema

UML autorise à spécifier tant une multiplicité maximale qu'une multiplicité minimale pour les attributs UML. Les multiplicités servent à indiquer des contraintes sur le nombre de valeurs que doit avoir un attribut. La multiplicité minimale induit les contraintes suivantes :

- Lorsque la multiplicité minimale est égale à `0`, l'attribut est optionnel.
- Lorsque la multiplicité minimale est égale à `1`, l'attribut est obligatoire.
- Lorsque la multiplicité minimale est supérieure à `1`, l'attribut est une liste ayant un nombre de valeurs obligatoires égal à la valeur de la multiplicité minimale.

Par défaut, la multiplicité minimale est égale à `0`.

La multiplicité maximale induit les contraintes suivantes :

- Lorsque la multiplicité maximale est égale à `1`, l'attribut a une seule valeur.
- Lorsque la multiplicité maximale est supérieure à `1`, l'attribut est une liste de valeurs.

Par défaut, la multiplicité maximale est égale à `1`.

Pour savoir si un attribut a une seule valeur ou s'il est une liste, il suffit de regarder sa multiplicité maximale. La multiplicité minimale n'indique, quant à elle, que le nombre de valeurs obligatoires.

Les règles de représentation d'un attribut UML en XML dépendent du type de données de l'attribut et du choix de transformation des attributs en XML.

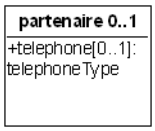
- Lorsqu'un attribut est de type composé, par exemple un type adresse avec rue, ville et code postal, l'attribut UML est obligatoirement représenté par un élément XML.
- Lorsqu'un attribut est de type simple, il peut au choix être représenté par un attribut XML ou par un élément XML. Le choix relève d'une décision de conception des schémas XML.

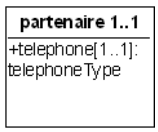
Lorsqu'un attribut UML est représenté par un élément XML, les valeurs des multiplicités minimale et maximale correspondent aux indicateurs d'occurrences de l'élément : `minOccur`, `maxOccur`.

Lorsqu'un attribut UML est représenté par un attribut XML, le type associé à l'attribut XML varie en fonction des valeurs des multiplicités. En effet, il n'est pas possible, en XML, de définir des indicateurs d'occurrences pour les attributs. En XML, un élément ne peut avoir qu'un seul attribut d'un nom donné. En revanche, cet attribut peut avoir une suite de valeurs. Il faut alors recourir au type liste de XML Schema. Le tableau A1-1 donne la liste des combinaisons possibles.

Tableau A–1 Représentation d'un attribut UML par un attribut XML

Exemple UML	Multiplicité minimale	Multiplicité maximale	Attribut XML	Exemple XML

	0	1	Attribut de type simple non obligatoire	`<xs:simpleType name="telephoneType">`
				`<xs:restrictionbase="xs:string"/>`
				`</xs:simpleType>`
				
				`<xs:element name="partenaire">`
				`  <xs:complexType>`
				`  <xs:attribute`
				`  name="telephone"`
				`  type="telephoneType"`
				`  use="optional"/>`
				`  </xs:complexType>`
				`</xs:element>`

	1	1	Attribut de type simple obligatoire	`<xs:simpleType name="telephoneType">`
				`<xs:restrictionbase="xs:string"/>`
				`</xs:simpleType>`
				
				`<xs:element name="partenaire">`
				`  <xs:complexType>`
				`  <xs:attribute`
				`  name="telephone"`
				`  type="telephoneType"`
				`  use="required"/>`
				`  </xs:complexType>`
				`</xs:element>`

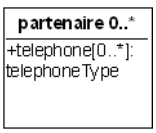
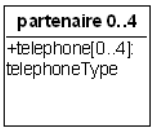
	0	*	Attribut de type liste	`<xs:simpleType name="telephoneType">`
				`<xs:restrictionbase="xs:string"/>`
				`</xs:simpleType>`
				
				`<xsd:simpleType name="telListe">`
				`<xsd:list itemType="telephoneType"/>`
				`</xsd:simpleType>`
				
				`<xs:element name="partenaire">`
				`  <xs:complexType>`
				`  <xs:attribute`
				`  name="telephone"`
				`  type="telListe"`
				`  use="required"/>`
				`  </xs:complexType>`
				`</xs:element>`

Tableau A–1 Représentation d'un attribut UML par un attribut XML

Exemple UML	Multiplicité minimale	Multiplicité maximale	Attribut XML	Exemple XML

	0	>1
		+
		Valeur limite
			Attribut de type liste avec facette maxLength	`<xs:simpleType name="telephoneType">`
				`<xs:restrictionbase="xs:string"/>`
				`</xs:simpleType>`
				
				`<xs:simpleType name="telListe">`
				`<xs:list itemType="telephoneType"/>`
				`</xs:simpleType>`
				
				`<xs:simpleType`
				`name="telListeMaxQuatre">`
				`<xs:restriction base="telList">`
				`<xs:maxLength value="4"/>`
				`</xs:restriction>`
				`</xs:simpleType>`
				
				`<xs:element name="partenaire">`
				`<xs:complexType>`
				`<xs:attribute`
				`name="telephone"`
				`type="telListeMaxQuatre"`
				`use="required"/>`
				`</xs:complexType>`
				`</xs:element>`

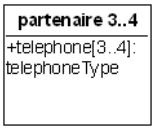
	>0
	+
	Valeur limite
		>1
		+
		Valeur limite
			Attribut de type liste avec facettes maxLength et min-Length	`<xs:simpleType name="telephoneType">`
				`<xs:restrictionbase="xs:string"/>`
				`</xs:simpleType>`
				
				`<xs:simpleType name="telListe">`
				`<xs:list itemType="telephoneType"/>`
				`</xs:simpleType>`
				
				`<xs:simpleType`
				`name="telListeMinMax">`
				`<xs:restriction base="telList">`
				`<xs:minLength value="3"/>`
				`<xs:maxLength value="4"/>`
				`</xs:restriction>`
				`</xs:simpleType>`
				
				`<xs:element name="partenaire">`
				`<xs:complexType>`

Tableau A–1 Représentation d'un attribut UML par un attribut XML

Exemple UML	Multiplicité minimale	Multiplicité maximale	Attribut XML	Exemple XML
				`<xs:attribute name="telephone" type="telListeMinMax" use="required"/> </xs:complexType> </xs:element>`

À la lecture du tableau A-1, on aura pu noter que la correspondance UML/XML n'est pas directe. Ce qui est spécifié au niveau de l'attribut dans UML l'est au niveau du type de l'attribut dans XML.

En ce qui concerne le type union de XML Schema, on ne trouve pas de représentation correspondante dans UML. XML Schema dispose de capacités d'expression plus avancées en ce qui concerne le contrôle des valeurs d'attributs et d'éléments.

Attributs et valeur par défaut

Il est possible de donner une valeur par défaut aux attributs UML. La correspondance de ces valeurs par défaut en XML est donnée dans le tableau A-2.

La définition d'annotations

UML permet de définir des commentaires sur tout objet de modélisation. À chaque commentaire associé à un objet UML correspond dans le schéma XML une annotation XML de type documentation.

Groupe d'attributs

La notion de groupe d'attributs n'existe pas en UML. Le concept le plus proche est celui de type de données (*DataType* en anglais) avec la restriction suivante : les attributs du type de données doivent eux-mêmes être de type simple, c'est-à-dire ne pas se décomposer en d'autres attributs. Pour exprimer l'utilisation d'un groupe d'attributs par une classe, il faut utiliser une relation de généralisation entre la classe et le type de données.

Tableau A–2 Valeur d'attribut par défaut

Modèle UML	Représentation XML

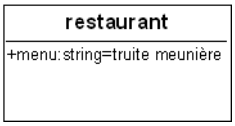
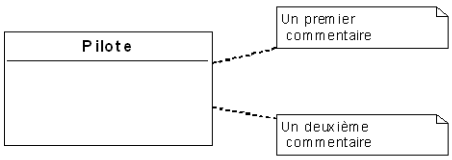
Attribut avec valeur par défaut
 
 	
	Cas où l'attribut UML donne lieu à un élément XML
	
	`<xs:element name="restaurant">`
	`  <xs:complexType>`
	`  <xs:sequence>`
	`  <xs:element name="menu"`
	`  type = "xs:string"`
	`  default="truite meunière"/>`
	`  </xs:sequence>`
	`  </xs:c omplexType>`
	`</xs:element>`
	Cas où l'attribut UML donne lieu à un attribut XML
	`<xs:element name="partenaire">`
	`  <xs:complexType>`
	`  <xs:attribute`
	`  name="menu"`
	`  type="xs:string"`
	`  value="truire meunière"/>`
	`  </xs:complexType>`
	`</xs:element>`

Tableau A–3 Annotations

Modèle UML	Représentation XML

Une classe et ses commentaires
 
 	
	Annotation XML
	
	`<xs:element name="pilote">`
	`  <xs:annotation>`
	`  <xs:documentation>Un premier commentaire`
	`  </xs:documentation>`
	`  <xs:documentation>Un deuxième commentaire`
	`  </xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`  …`
	`  </xs:complexType>`
	`</xs:element>`

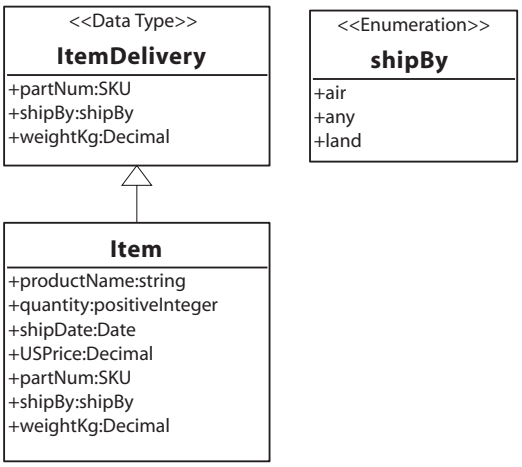
Sur le plan de la conformité à UML, cette relation de généralisation entre une classe et un type de données est inhabituelle, mais ne viole pas les règles de généralisation

établies par la norme UML 2.0. Si l'on avait considéré le groupe d'attributs comme une classe standard, nous aurions été confrontés à des difficultés pour la transposition du modèle de classe vers le modèle XML. Au chapitre 3, nous avons vu que, dans le cas général, chaque classe donne lieu à un élément XML ; or ce n'est pas ce que nous recherchons ici, les groupes d'attributs n'étant justement pas des éléments XML Schema.

Tableau A–4 Groupe d'attributs

Modèle UML

Une classe héritant d'un type de données



Représentation XML

Groupe d'attributs XML

```
<xsd:attributeGroup name="ItemDelivery">
  <xsd:attribute name="partNum" type="SKU"/>
  <xsd:attribute name="weightKg"
                 type="xsd:decimal"/>
  <xsd:attribute name="shipBy">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:enumeration value="air"/>
      <xsd:enumeration value="land"/>
      <xsd:enumeration value="any"/>
     </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
```

Tableau A–4 Groupe d'attributs

```
<xsd:element name="item">
<xsd:complexType>
  <xsd:sequence>
   <xsd:element name="productName"
                type="xsd:string"/>
   <xsd:element name="quantity">
    <xsd:simpleType>
     <xsd:restriction base="xsd:positiveInteger">
      <xsd:maxExclusive value="100"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="USPrice"
                type="xsd:decimal"/>
   <xsd:element ref="comment"   minOccurs="0"/>
   <xsd:element name="shipDate" type="xsd:date"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="ItemDelivery"/>
</xsd:complexType>
</xsd:element>
```

La mixité des modèles

Tableau A–5 Modèle mixte

Modèle UML
Classe héritant du type de données string

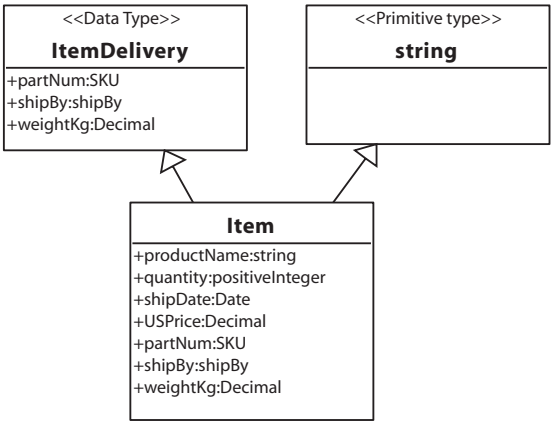


Tableau A–5 Modèle mixte

Représentation XML
Modèle de contenu mixte

```
<xsd:element name="item">
<xsd:complexType mixed=true>
  <xsd:sequence>
   <xsd:element name="productName"
                type="xsd:string"/>
   <xsd:element name="quantity">
    <xsd:simpleType>
     <xsd:restriction base="xsd:positiveInteger">
      <xsd:maxExclusive value="100"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="USPrice"
                type="xsd:decimal"/>
   <xsd:element ref="comment"   minOccurs="0"/>
   <xsd:element name="shipDate" type="xsd:date"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="ItemDelivery"/>
</xsd:complexType>
</xsd:element>
```

Dans un cas particulier d'héritage, le type de données hérité est le type primitif string. C'est un cas limite du point de vue de la conformité du modèle UML, même s'il reste valide. Il permet de représenter les modèles de contenu mixte de XML Schema comme indiqué dans le tableau A-5.

Comme pour d'autres correspondances entre modèles de classes et XML Schema, la solution proposée dans ce paragraphe pousse le modèle UML dans ses retranchements. Voilà une indication supplémentaire comme quoi il n'y a pas recouvrement un pour un du modèle de classes UML et de XML Schema.

Contrainte d'exclusion et groupe de choix

UML dispose d'une contrainte standard, appelée `{xor}`, permettant de définir une exclusion entre deux associations.

Les contraintes d'exclusion `xor` sont représentées par un groupe de choix XML.

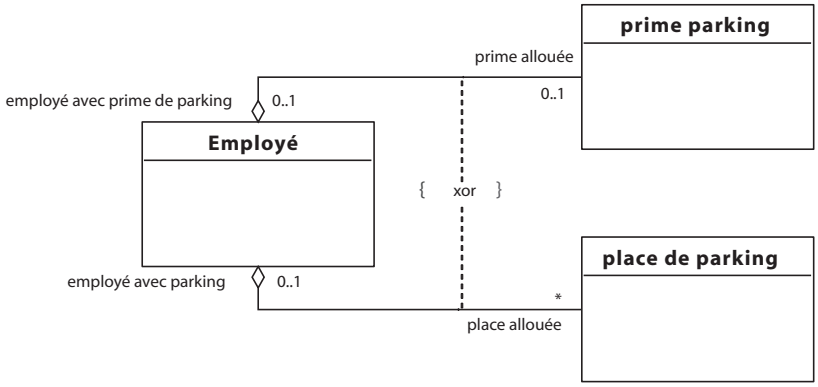
VOCABULAIRE Contrainte d'exclusion

Une contrainte d'exclusion indique qu'un objet ne peut être relié que par l'une ou l'autre des associations référencées par la contrainte. Dans l'exemple du tableau A-6, un employé est soit un employé avec prime de parking, soit un employé avec place de parking ; ou, ce qui est équivalent, il a soit une prime de parking, soit une place de parking.

Tableau A–6 Contrainte d'exclusion et groupe de choix

Modèle UML

Contrainte d'exclusion



Représentation XML

Groupe de choix XML

```
<xs:element name="Employé">
  <xs:complexType>
    <xs:choice>
      <xs:element name="parkingAlloue"
                  type="placeDeParking"/>
      <xs:element name="primedeparkingAllouee"
                  type="primeDeParking"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

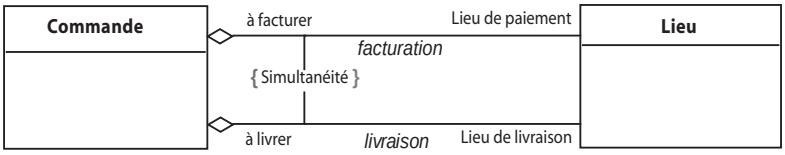
Contraintes de simultanéité et groupes simples

Outre la contrainte standard `{xor}` de UML, il existe d'autres contraintes sur associations qui font l'objet d'un large consensus, en particulier la contrainte de simultanéité.

VOCABULAIRE **Contrainte de simultanéité**

Une contrainte de simultanéité indique que si un objet participe à l'une des associations de la contrainte, il participe alors également aux autres associations. Dans l'exemple suivant, si une commande est à facturer, elle est aussi à livrer, et réciproquement.

Figure A–1
Contrainte de simultanéité



Il est tentant d'utiliser directement les groupes simples XML (<xsd:group>) pour exprimer les contraintes de simultanéité dans les schémas XML. Malheureusement, contrairement aux groupes de choix, les groupes simples doivent être déclarés comme des éléments globaux pour être réutilisés dans plusieurs autres éléments XML. Tel n'est pas l'objectif de la contrainte de simultanéité qui vise seulement la présence simultanée d'associations et non pas leur réutilisation.

En UML, la réutilisation s'exprime toujours à l'aide de relations de généralisation. Il faut donc revoir le schéma de la figure A-1 en conséquence : une nouvelle classe abstraite indique la mise en facteur des associations facturation et livraison ; la contrainte de simultanéité est conservée. Une représentation XML possible de ce modèle de classe est présentée dans le tableau A-7.

Tableau A–7 Contrainte de simultanéité et groupe simple

Modèle UML
Contrainte d'exclusion

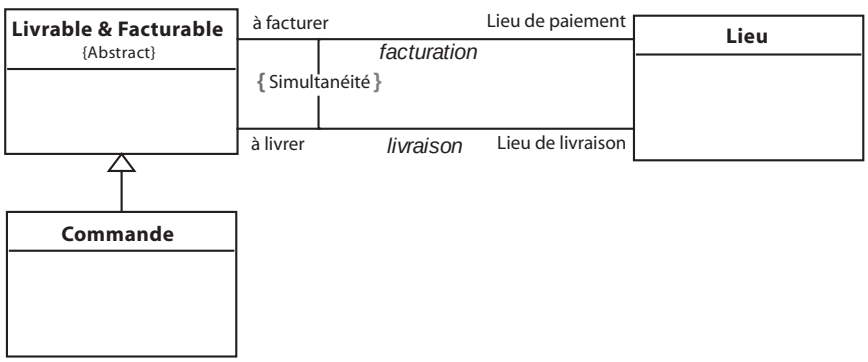


Tableau A–7 Contrainte de simultanéité et groupe simple

Représentation XML
Groupe XML

```
<xs:schema elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   xmlns:xl=http://www.w3.org/1999/xlink
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import
     namespace=http://www.w3.org/1999/xlink
     schemaLocation="xlink.xsd"/>
  <xs:complexType name="entityRef">
    <xs:attribute ref="xl:type" fixed="simple"/>
    <xs:attribute ref="xl:href" use="required"/>
  </xs:complexType>
  <xs:group name="LivrableFacturable">
    <xs:sequence>
      <xs:element name="LieuLivraison"
                  type="entityRef"/>
      <xs:element name="LieuFacturation"
                  type="entityRef"/>
    </xs:sequence>
  </xs:group>
  <xs:element name="Commande">
    <xs:complexType>
      <xs:group ref="LivrableFacturable"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Il faut ici remarquer un nouveau cas où il n'y a pas de correspondance claire et directe entre le modèle de classes UML et XML Schema. Le tableau A-7 indique un choix particulier de transformation des relations de généralisation en XML, qui vient s'ajouter à celles exposées dans le chapitre 3.

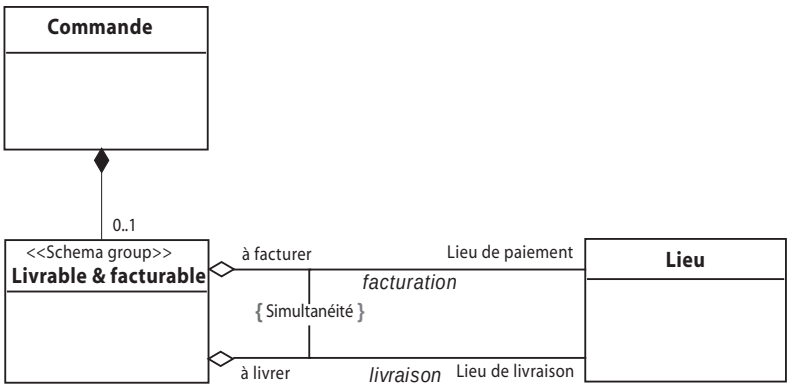
Certaines recommandations conseillent l'utilisation de stéréotypes UML pour faire correspondre directement les concepts de XML Schema et ceux de UML.

VOCABULAIRE Stéréotypes UML

Les stéréotypes sont une technique de marquage des classes UML utilisée pour étendre le métamodèle UML et l'adapter à un cas d'emploi spécifique, par exemple XML Schema.

La figure A-2 montre l'exemple du stéréotype « Schema group » utilisé pour représenter un groupe XML à l'aide d'une classe UML spécifique. Cependant, l'usage de tels stéréotypes ne fait que singer les concepts XML dans les modèles de classes UML, sans apporter véritablement de valeur ajoutée. La classe `Livrable & Facturable` n'est pas à proprement parler une classe et ne fournit pas une aide à la découverte des classes entités et des modules de données, ce qui est l'objectif premier d'un modèle de conception UML, ainsi que nous l'avons exposé au chapitre 3. L'usage des stéréotypes pour représenter les spécificités de XML Schema tient plus de la recette de cuisine que de la véritable modélisation de données.

Figure A–2
Utilisation d'un stéréotype pour représenter un groupe XML



La question de l'ordre des éléments

Pour XML Schema, l'ordre des éléments revêt une grande importance et fait partie intégrante de la validation de la conformité des documents XML par rapport à leurs schémas. Tel n'est pas le cas des modèles de classes UML. Si l'on peut bien spécifier un ordre pour les attributs, les rôles d'associations et les généralisations d'une classe, cet ordre ne change pas substantiellement la nature du modèle de classe.

Les éléments XML correspondant à une classe UML proviennent soit d'attributs et de rôles d'associations directement reliés à cette classe, soit d'attributs et de rôles d'associations issus des classes héritées par les relations de généralisation. Il n'est donc pas possible d'obtenir un ordre absolu des éléments XML, mais seulement un ordre relatif : ordre des éléments provenant de généralisations + ordre des éléments provenant d'attributs + ordre des éléments provenant de rôles d'associations.

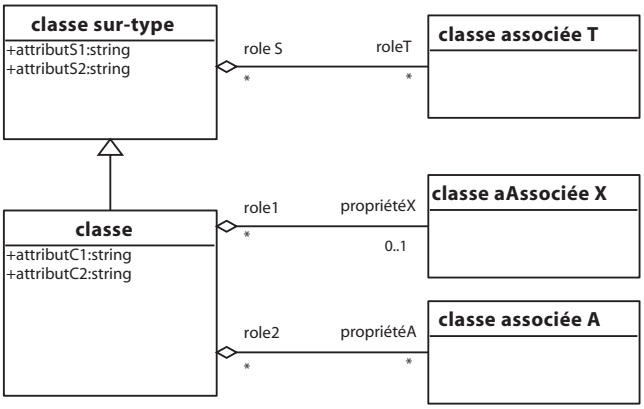
La règle le plus souvent appliquée est la suivante :

- Les éléments en provenance des classes héritées viennent en premier. Ils sont eux-mêmes ordonnés en fonction des deux règles suivantes.
- Les éléments en provenance d'attribut de classe viennent en second, dans l'ordre des attributs de la classe.
- Les éléments en provenance de rôle d'associations viennent en troisième, dans l'ordre des rôles de la classe.

Le tableau A-8 offre une illustration de l'application de ces règles.

Tableau A–8 Ordre des éléments

Modèle UML



Représentation XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="Classe">
       <xs:complexType>
      <xs:sequence>
        <xs:element name="attributS1"/>
        <xs:element name="attributS2"/>
        <xs:element name="RoleT"/>
        <xs:element name="attributC1"/>
        <xs:element name="attributC2"/>
        <xs:element name="proprieteX"/>
        <xs:element name="proprieteA"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```