Annexe du cours Conception des sites web marchands et mobiles

Nassim BAHRI {contact@nassimbahri.ovh}

4 Novembre 2016

1 Diagramme de séquence système

Les cas d'utilisation décrivent les interactions des acteurs avec le site web que nous voulons spécifier et concevoir. Lors de ces interactions, les acteurs produisent des messages qui affectent le système informatique et appellent généralement une réponse de celui-ci. Nous allons isoler ces messages et les représenter graphiquement sur des diagrammes de séquence UML.

Pour les messages propres à un cas d'utilisation, les DSS (diagrammes de séquence système) montrent non seulement les acteurs externes qui interagissent directement avec le système, mais également ce système (en tant que boîte noire) et les événements système déclenchés par les acteurs. L'ordre chronologique se déroule vers le bas et l'ordre des messages doit suivre la séquence décrite dans le cas d'utilisation.

Nous allons représenter le DSS d'un scénario représentatif pour le cas d'utilisation "chercher des ouvrages". (Voir figure 1)

2 Diagramme de séquence détaillé

L'expression diagramme d'interactions englobe principalement deux types de diagrammes UML spécialisés, qui peuvent servir tous les deux à exprimer des interactions de messages similaires :

- les diagrammes de communication (collaboration)
- les diagrammes de séquence.

Les diagrammes de séquence représentent les interactions dans un format où chaque nouvel objet est ajouté en haut à droite. On représente la ligne de vie de chaque objet par un trait pointillé vertical. Cette ligne de vie sert de point de départ ou d'arrivée à des messages représentés eux-mêmes par des flèches horizontales (voir figure 2). Par convention, le temps coule de haut en bas. Il indique ainsi visuellement la séquence relative des envois et réceptions de messages, d'où la dénomination : diagramme de séquence.

3 Diagramme des classes participantes

Il s'agit de diagrammes de classes UML qui décrivent, cas d'utilisation par cas d'utilisation, les trois principales classes d'analyse et leurs relations. Les diagrammes de classes participantes sont importants car ils font la jonction entre les cas d'utilisation, la maquette et les diagrammes de conception logicielle (diagrammes d'interaction et diagrammes de classes).
Pour compléter ce travail d'identification, nous allons ajouter des attributs et des opérations dans les classes d'analyse, ainsi que des associations entre elles.

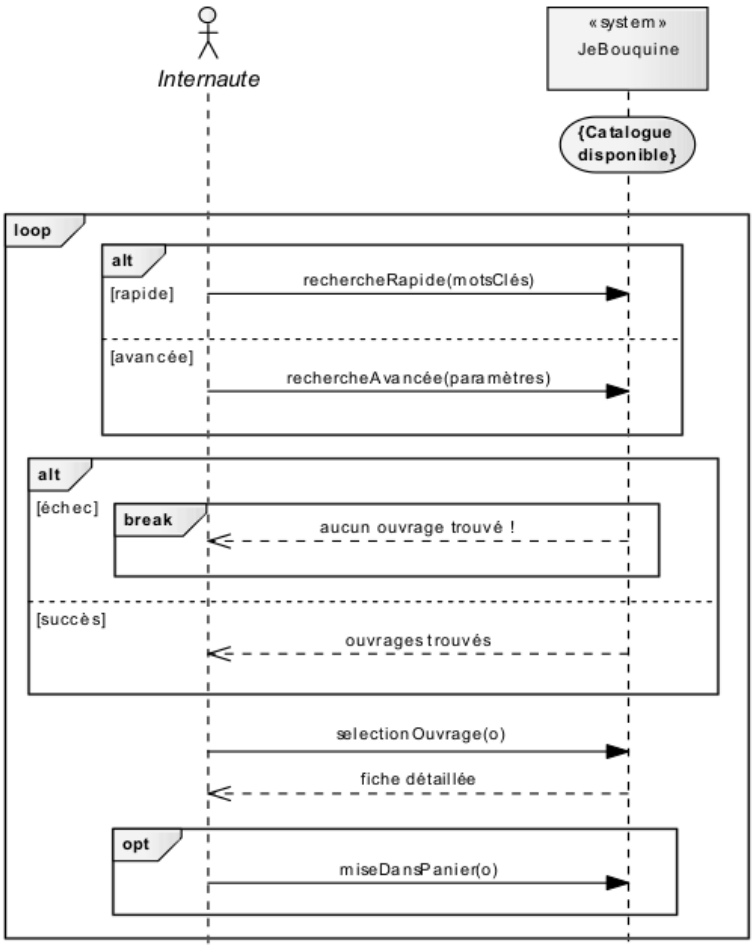

Figure 1: DSS de Chercher des ouvrages [1]

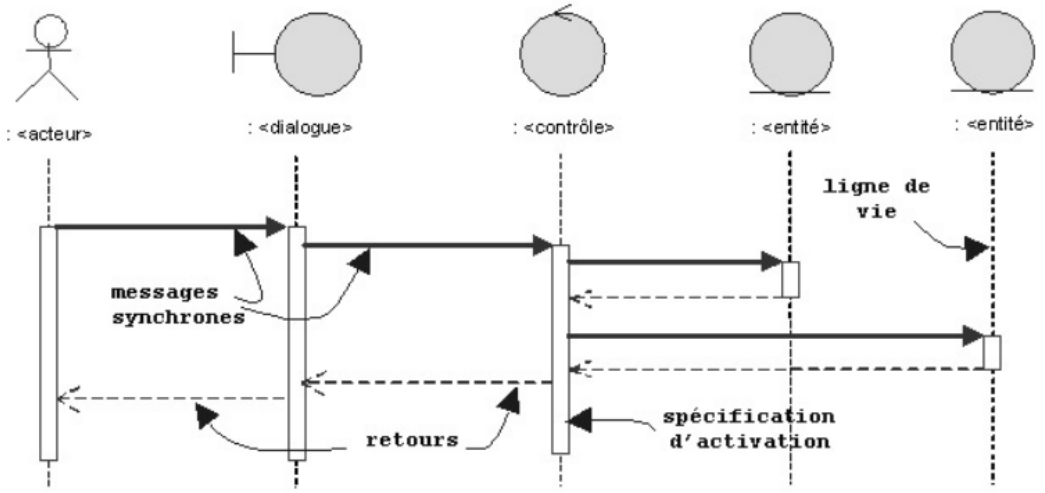

Figure 2: Notations du diagramme de séquence détaillé [1]

- Les entités vont seulement posséder des attributs. Ces attributs représentent en général des informations persistantes de l'application (stocké dans des base de données, fichiers,...).

- Les contrôles vont seulement posséder des opérations. Ces opérations montrent la logique de l'application, les règles transverses à plusieurs entités, bref les comportements du système informatique. Il y a souvent un seul contrôle par cas d'utilisation, mais il peut également y en avoir plusieurs, en fonction du nombre et de la cohérence des comportements associés.
- Les dialogues vont posséder des attributs et des opérations. Les attributs représenteront des champs de saisie ou des résultats. Les résultats seront distingués en utilisant la notation de l'attribut dérivé. Les opérations représenteront des actions de l'utilisateur sur l'IHM

Nous allons également ajouter des associations entre les classes d'analyse, mais en respectant des règles assez strictes :

- Les dialogues ne peuvent être reliés qu'aux contrôles ou à d'autres dialogues, mais pas directement aux entités.
- Les entités ne peuvent être reliées qu'aux contrôles ou à d'autres entités.
- Les contrôles ont accès à tous les types de classes, y compris d'autres contrôles.

La figure 4 illustre un exemple de diagramme des classe participante pour le cas d'utilisation "Gérer son panier".

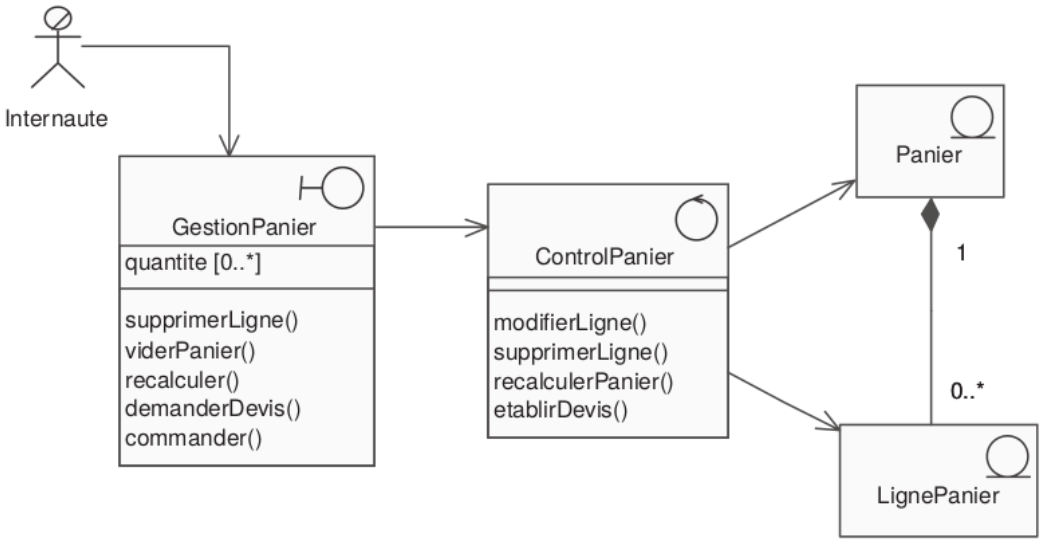

Figure 3: Diagramme des classes participante du cas d'utilisation Gérer son panier [1]

4 Diagramme d'états de navigation

Pour modéliser la navigation dans un site web, nous allons utiliser un nombre restreint d'éléments standards, à savoir :

- des états pour représenter les classes dialogues,
- des transitions entre états déclenchées par des événements et pouvant porter des conditions, pour représenter les actions IHM.

Conventions spécifiques
Nous allons pousser un peu plus loin et nous servir du concept d'état pour modéliser plusieurs concepts différents, grâce aux conventions graphiques suivantes :

- une page complète du site («page»),
- un frame particulier à l'intérieur d'une page («frame»),
- une erreur ou un comportement inattendu du système («exception» avec un niveau de gris intermédiaire),
- une liaison vers un autre diagramme d'activité, pour des raisons de structuration et de lisibilité («connector» avec un niveau de gris soutenu).

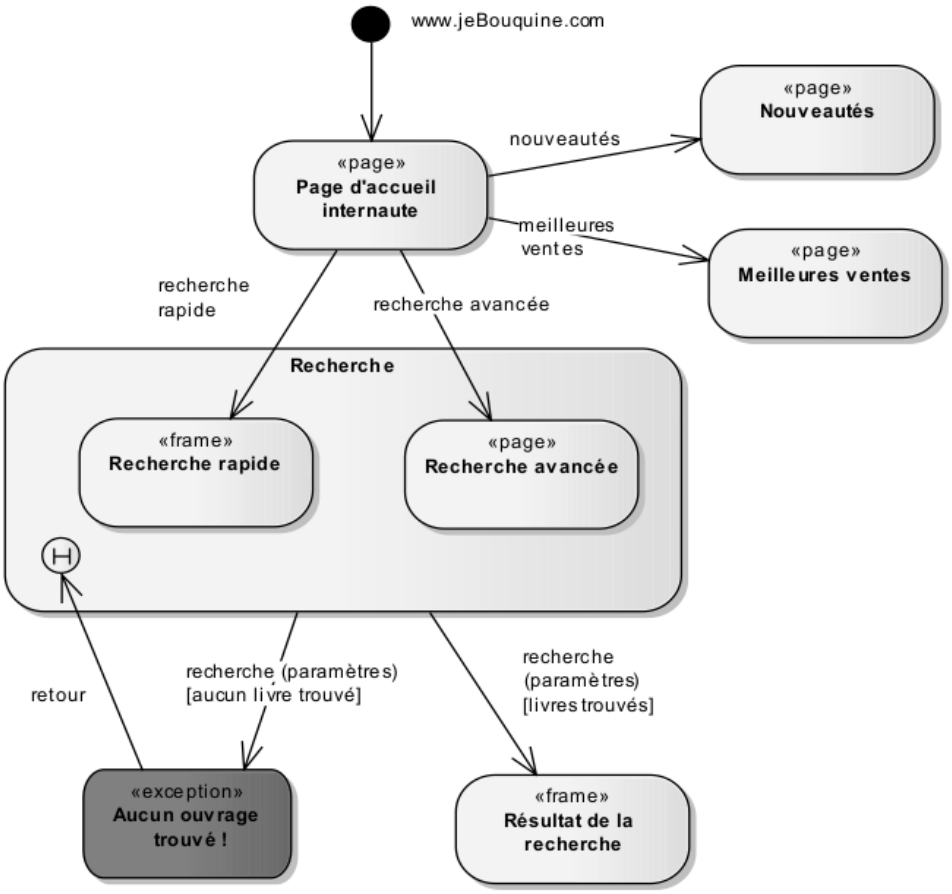

Figure 4: Navigation de la recherche [1]

5 Diagramme de collaboration

Le diagramme de collaboration comme le diagramme de séquence font parties des diagrammes d'interaction UML. Ces diagrammes sont utiles au concepteur pour représenter graphiquement ses décisions d'allocation de responsabilités. Chaque diagramme va ainsi représenter un ensemble d'objets de classes différentes collaborant dans le cadre d'un scénario d'exécution du système. Dans ces diagrammes, les objets communiquent en s'envoyant des messages qui invoquent des opérations sur les objets récepteurs.

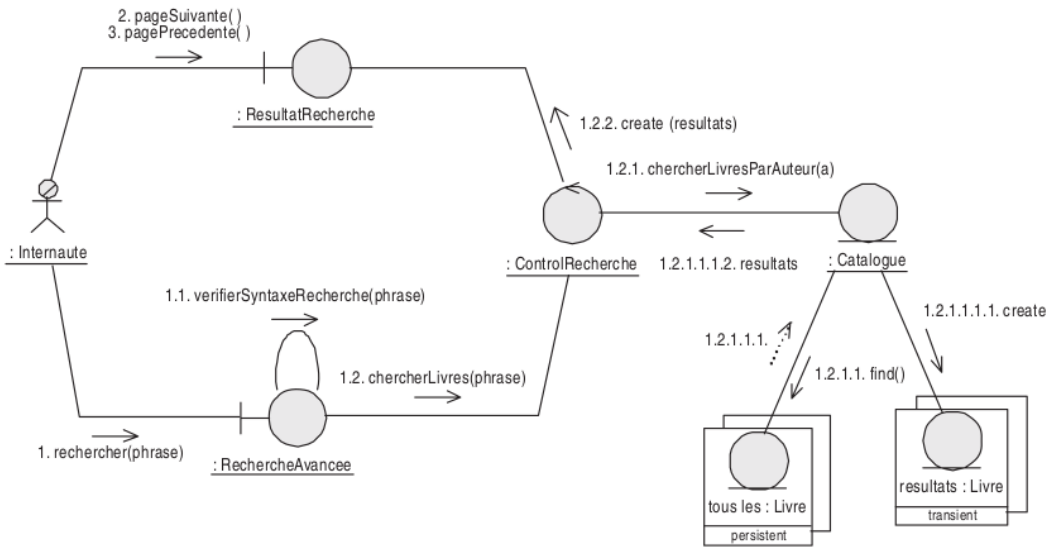

Figure 5: Diagramme de collaboration du scénario nominal de recherche avancée [1]

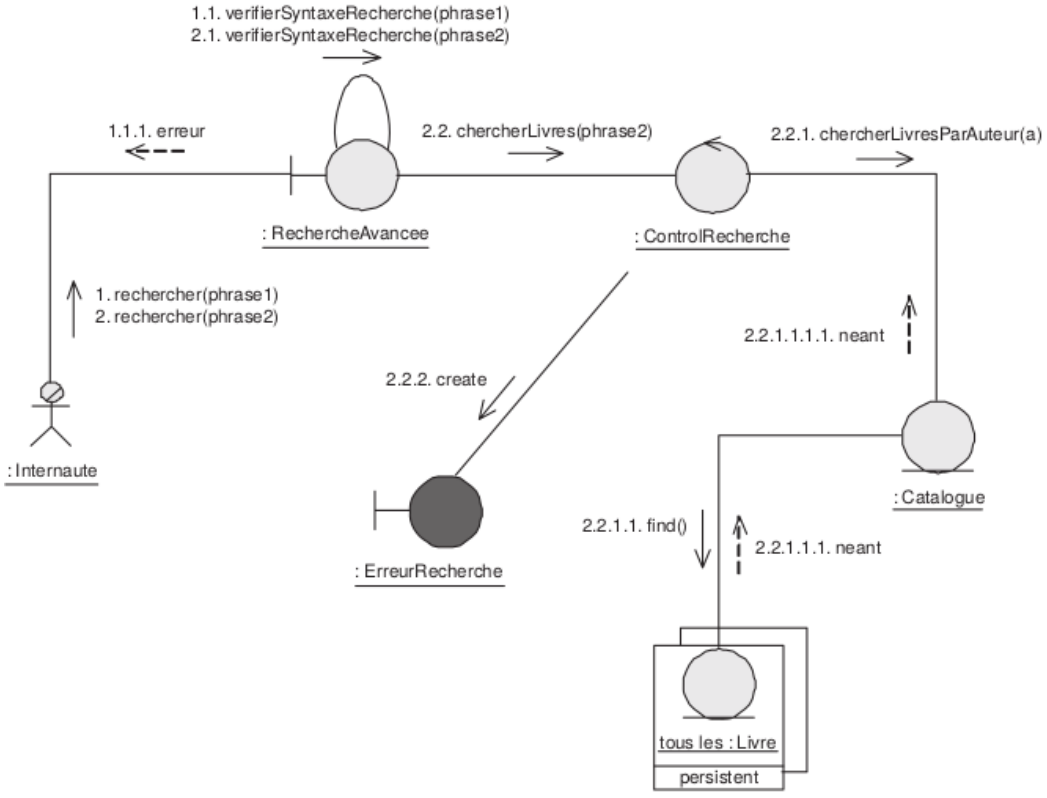

Figure 6: Diagramme de collaboration des scénarios d'erreur de recherche avancée [1]

6 Diagramme des classes de conception

Montre les briques de base statiques : classes, associa- tions, interfaces, attributs, opérations, généralisations, etc.

Classe : description abstraite d'un ensemble d'objets qui partagent les mêmes propriétés (attributs et associations) et comportements (opérations et états).

Attribut : donnée déclarée au niveau d'une classe, éventuellement typée, à laquelle chacun des objets de cette classe donne une valeur. Un attribut peut posséder une multiplicité et une valeur initiale. Un attribut dérivé (« / ») est un attribut dont la valeur peut être déduite d'autres informations disponibles dans le modèle.

Opération : élément de comportement des objets, défini de manière globale dans leur classe. Une opération peut déclarer des paramètres (eux-mêmes typés) ainsi qu'un type de retour.

Association : relation sémantique durable entre deux classes, qui décrit un ensemble de liens entre instances. Une association est bidirectionnelle par défaut, sauf si l'on restreint sa navigabilité en ajoutant une flèche.

Rôle : nom donné à une extrémité d'une association ; par extension, manière dont les instances d'une classe voient les instances d'une autre classe au travers d'une association.

Multiplicité : le nombre d'objets (min..max) qui peuvent participer à une relation avec un autre objet dans le cadre d'une association. Multiplicités fréquentes :

- 0..1 = optionnel (mais pas multiple)
- 1 = exactement 1
- 0..* = * = quelconque
- 1..* = au moins 1

Agrégation : cas particulier d'association non symétrique exprimant une relation de contenance.

Composition : forme forte d'agrégation, dans laquelle les parties ne peuvent appartenir à plusieurs agrégats et où le cycle de vie des parties est subordonné à celui de l'agrégat.

Super-classe : classe générale reliée à d'autres classes plus spécialisées (sous-classes) par une relation de généralisation.

Classe d'association : association promue au rang de classe. Elle possède tout à la fois les caractéristiques d'une association et celles d'une classe et peut donc porter des attributs qui prennent des valeurs pour chaque lien entre objets.

7 Règles de validation du diagramme des classes

En se référant au livre "Conception d'une base de données avec UML" de Gilles ROY [2], et par analogie avec le modèle conceptuel de données (notation MERISE), la validation d'un diagramme de classe se conclut en cinq points qui peuvent être décrits comme suit:

1. **Règle d'identité:** Chaque entité (classe) doit posséder un identifiant. Cet identifiant est généralement explicite. Il peut être implicite dans trois cas : soit.
   (a) pour une entité (classe) d'association où il est formé par défaut de la combinaison des identifiants des entités (classes) de l'association.
   (b) pour une entité de type composant où il est formé de la combinaison de l'identifiant du composant avec celui de son composite.

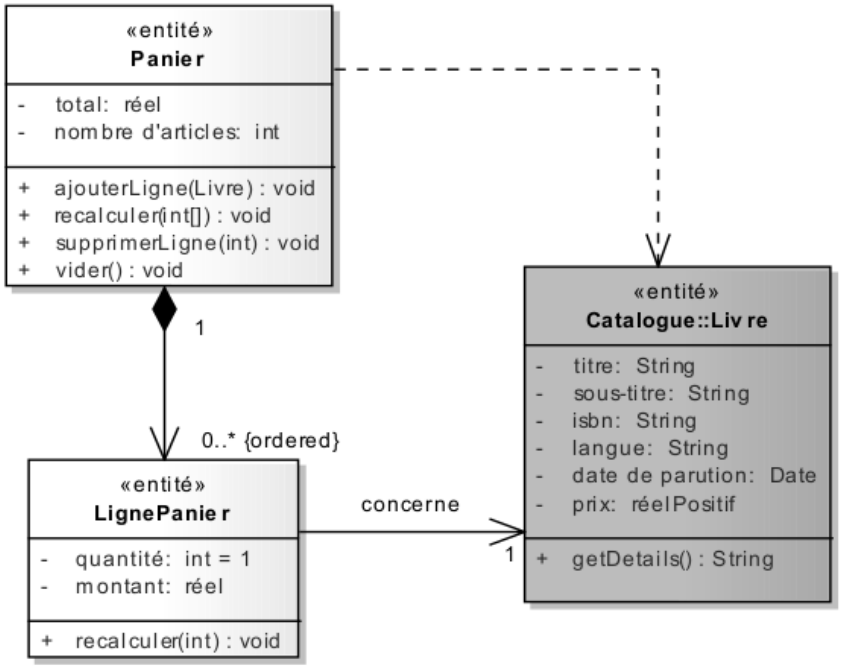

Figure 7: Diagramme des classes du package panier

(c) pour une association d'héritage où une entité qui hérite des attributs d'une autre reçoit par conséquent l'identifiant de cette dernière et ne possède jamais d'identifiant qui lui est propre.

Il faut éviter de faire apparaître explicitement un identifiant qui soit en fait implicite. Cela Conduit à la transgression des règles 3 et 4

2. **Règle de description:** Chaque attribut ne peut avoir qu'une valeur à la fois et cette donnée doit avoir un caractère élémentaire, c'est-à dire posséder un type de données simple.

3. **Règle de non-redondance:** Un attribut ne peut figurer qu'une seule fois dans le diagramme entité-association (diagramme des classe en UML). Prendre garde aux synonymes, la même donnée sous deux noms différents, et à la polysémie, une même appellation pour deux données différentes.

4. **Règle de construction:** Les attributs d'une entité décrivent bien cette entité et lui appartiennent en propre. Aucun ne peut appartenir à une autre entité. La valeur de chaque attribut est déterminée de manière unique par la valeur de l'identifiant. Cette règle possède un corollaire: il faut éviter de répliquer un attribut dans une entité provenant d'une deuxième entité pour signifier qu'elles sont liées. Pour ce faire, il faut employer une association.

5. **Règle de décomposition:** Il est souhaitable de décomposer les associations de degré supérieur le cas échéant. Deux situations se prêtent à la décomposition:

   (a) une des multiplicités maximales est égale à 1.

   (b) il existe une dépendance fonctionnelle entre deux identifiants parmi les entités associées.

8 Passage au code

Les diagrammes de classe permettent de décrire la squelette du code, nous prenons comme exemple de langage JAVA:

- Classe UML : Classe JAVA
- Attribut UML : Variables d'instance JAVA
- Opération UML : Méthode JAVA

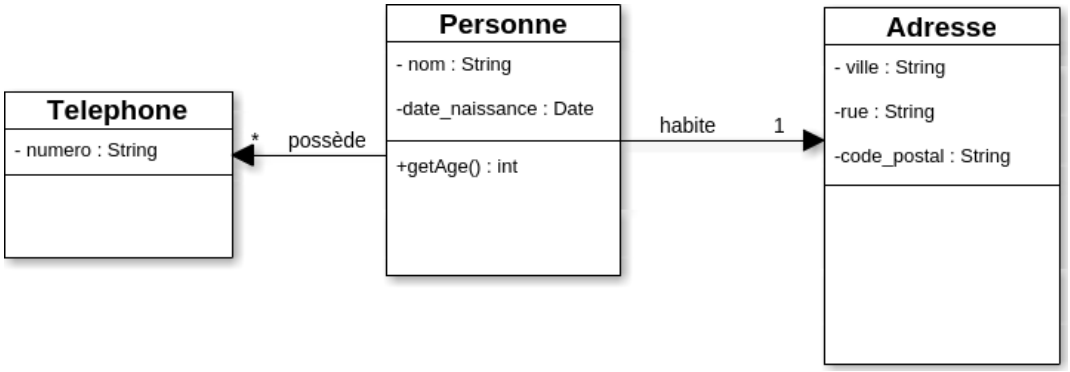

Figure 8: Exemple de diagramme des classes

```
// Adresse.java
package monpackage;
public class Adresse{
   private String ville;
   private String rue;
   private String code_postal;
}
```

```
// Personne.java
package monpackage;
import java.util.Date;
public class Personne{
   private String nom;
   private Date date_naissance;
   private Adresse adresse;
   private ArrayList<Telephone> telephones;
   public int getAge(){
      return 0;
   }
}
```

Pour plus de détails sur la transformation du diagramme des classes en classe JAVA, visiter le lien `http://goo.gl/tdQL2W`

9 Passage à la base de données relationnelle

Nous donnons ci-après quatre règles (de R1 à R4) pour traduire un schéma conceptuel entité-association ou UML en un schéma relationnel équivalent [3].

9.1 Transformation des classes (Règle 1)

Chaque classe du diagramme UML devient une relation. Il faut choisir un attribut de la classe pouvant jouer le rôle d'identifiant (voir règle 1 de validation du diagramme des classes).

Si aucun attribut ne convient en tant qu'identifiant, il faut en ajouter un de telle sorte que la relation dispose d'une clé primaire.

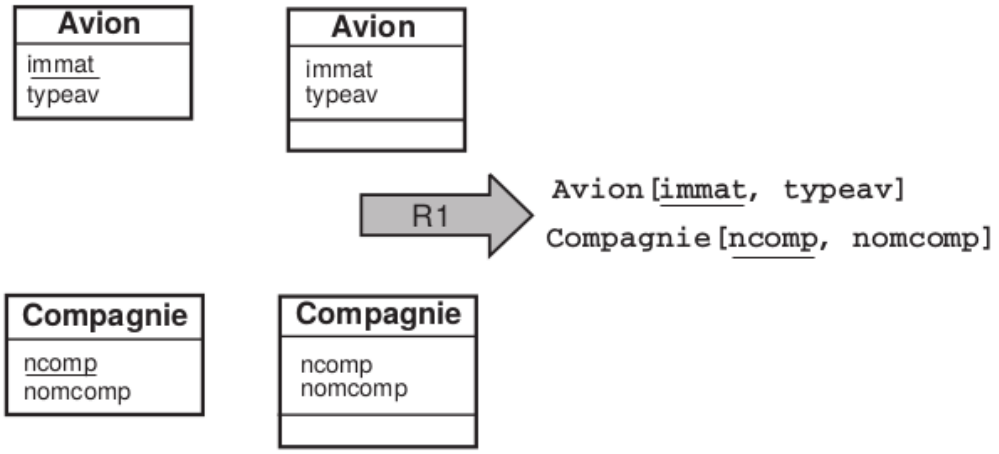

Figure 9: Transformation des classes [3]

9.2 Transformation des associations

Les règles de transformation que nous allons voir dépendent des cardinalités/multiplicités maximales des associations. Nous distinguons trois familles d'associations :

- un-à-plusieurs
- plusieurs-à-plusieurs ou classes-associations, et n-aires
- un-à-un

9.2.1 Associations un-à-plusieurs (Règle 2)

Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.
On peut se rappeler cette règle de la manière suivante : *la clé de la relation père migre dans la relation fils.*

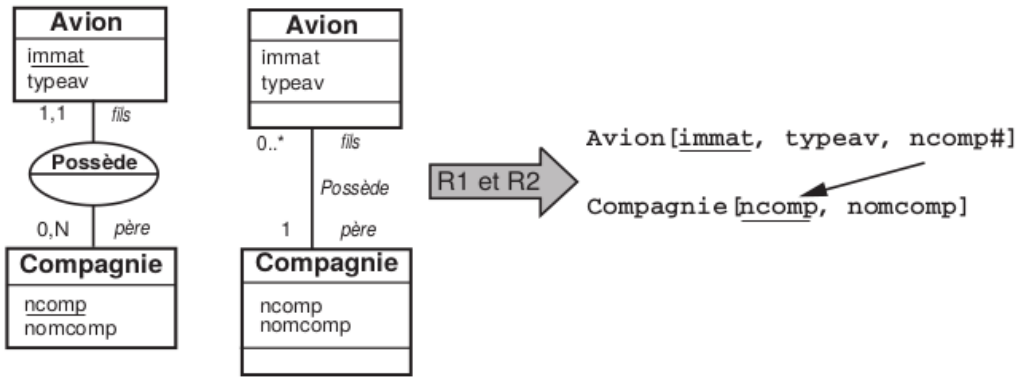

Figure 10: Transformation d'une association un-à-plusieurs [3]

9.2.2 Associations plusieurs-à-plusieurs et n-aires (Règle 3)

L'association (classe-association) devient une relation dont la clé primaire est composée par la concaténation des identifiants des entités (classes) connectés à l'association. Chaque attribut devient clé étrangère si l'entité (classe) connectée dont il provient devient une relation en vertu de la règle R1.
Les attributs de l'association (classe-association) doivent être ajoutés à la nouvelle relation. Ces attributs ne sont ni clé primaire, ni clé étrangère.

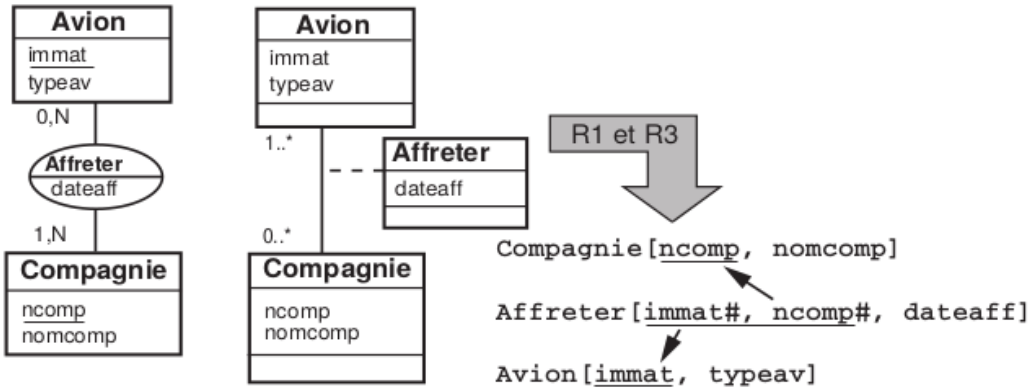

Figure 11: Transformation d'une association plusieurs-à-plusieurs [3]

9.2.3 Associations un-à-un (Règle 4)

La règle est la suivante, elle permet d'éviter les valeurs NULL dans la base de données.
Il faut ajouter un attribut clé étrangère dans la relation dérivée de l'entité ayant la cardinalité minimale égale à zéro. Dans le cas de UML, il faut ajouter un attribut clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de l'entité (classe) connectée à l'association.
Si les deux cardinalités (multiplicités) minimales sont à zéro, le choix est donné entre les deux relations dérivées de la règle R1. Si les deux cardinalités minimales sont à un, il est sans doute préférable de fusionner les deux entités (classes) en une seule.

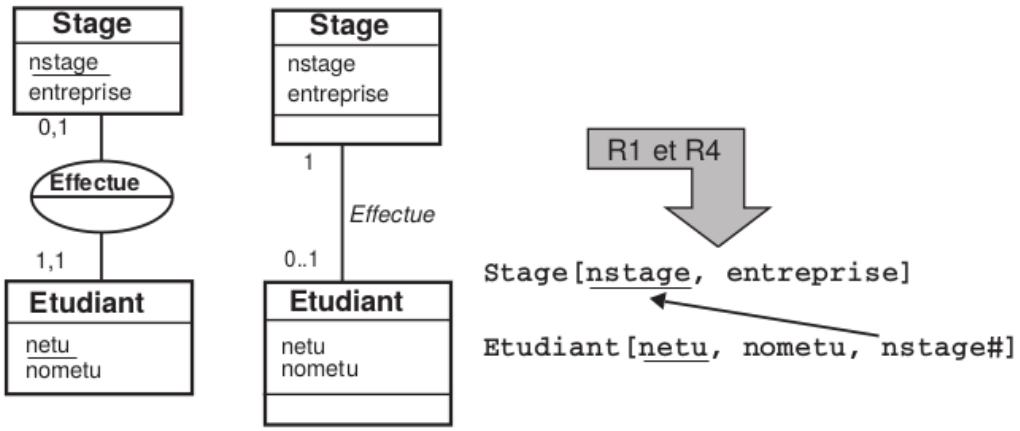

Figure 12: Transformation d'une association un-à-un [3]

9.3 Transformation de l'héritage

Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- décomposition par distinction
- décomposition descendante (push-down). S'il existe une contrainte de totalité ou de partition sur l'association d'héritage, il y aura deux cas possibles de décomposition
- décomposition ascendante (push-up).

9.3.1 Décomposition par distinction

Il faut transformer chaque sous-classe en une relation. La clé primaire de la sur-classe migre dans la (les) relation(s) issue(s) de la (des) sous-classe(s) et devient à la fois clé primaire et clé étrangère.

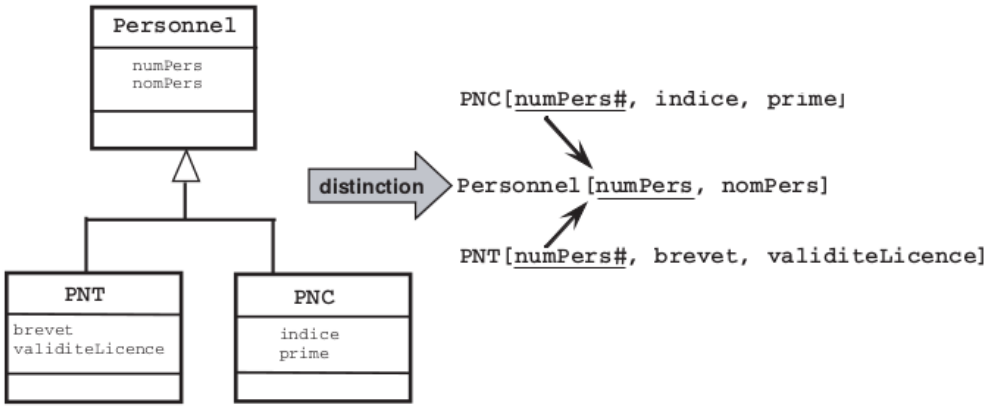

Figure 13: Décomposition par distinction d'une association d'héritage [3]

9.3.2 Décomposition descendante (push-down)

Deux cas sont possibles :

1. S'il existe une contrainte de totalité ou de partition sur l'association, il est possible de ne pas traduire la relation issue de la sur-classe. Il faut alors faire migrer tous ses attributs dans la (les) relation(s) issue(s) de la (des) sous-classe(s).
2. Dans le cas contraire, il faut faire migrer tous ses attributs dans la ou les relation(s) issue(s) de la (des) sous-classe(s) dans la (les) relation(s) issue(s) de la (des) sous-classe(s).

9.3.3 Décomposition ascendante (push-up)

Il faut supprimer la (les) relation(s) issue(s) de la (des) sous-classe(s) et faire migrer les attributs dans la relation issue de la sur-classe.

9.4 Transformation de la composition

La clé primaire des relations déduites des classes composantes doit contenir l'identifiant de la classe composite (quelles que soient les multiplicités).

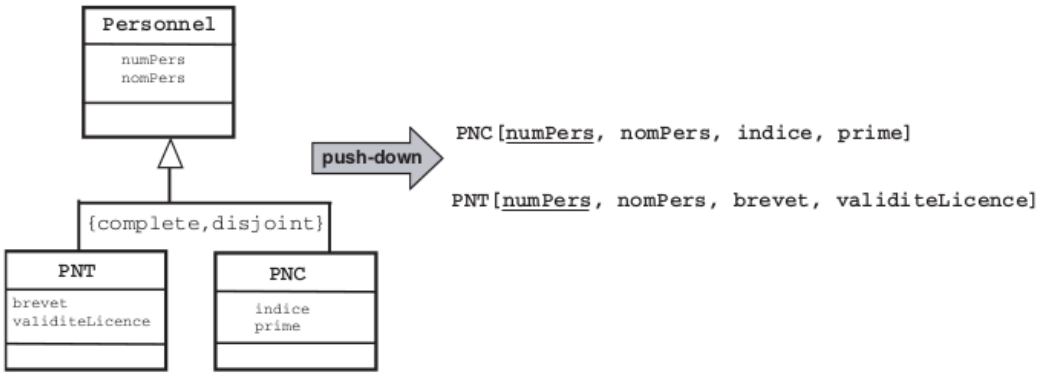

Figure 14: Décomposition descendante (push-down) d'une association d'héritage [3]

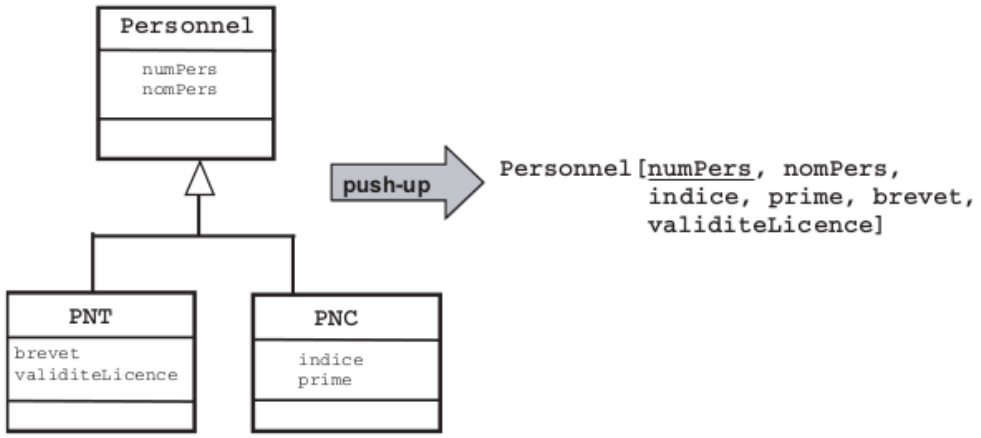

Figure 15: Décomposition ascendante (push-up) d'une association d'héritage [3]

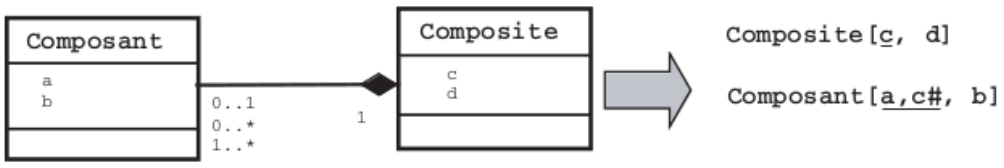

Figure 16: Transformation d'une composition [3]

References

[1] Pascal Roques. *UML 2 Modéliser une application web*. Eyrolles, 2008.

[2] Gilles Roy. *Conception d'une base de données avec UML*, pages 123–124. Presses de L'Université dU QUébec, 2009.

[3] Christian Soutou. *UML2 pour les bases de données*, pages 128–133. Eyrolles, 2007.