Introduction à UML

par Alexandre Brillant

Date de publication : 9/11/07

Dernière mise à jour :

UML est l'**Unified Modeling Language** standardisé par l'**OMG** (*Object Management Group* : http://www.omg.org). Ce n'est pas une méthode, il ne donne pas de solution pour la mise en oeuvre d'un projet. C'est avant tout un **formalisme graphique** issu de notations employées dans différentes méthodes objets.

I - Introduction à UML

Cette documentation ne peut être employée dans le cadre d'un cours ou d'une formation sans mon accord.

Je suis formateur Java, XML tout niveau : plus de renseignements.

UML sert à :

- **Décomposer** le processus de développement,
- **Mettre en relation** les experts métiers et les analystes,
- **Coordonner** les équipes d'analyse et de conception,
- **Séparer** l'analyse de la réalisation,
- Prendre en compte **l'évolution** de l'analyse et du développement,
- **Migrer** facilement vers une architecture objet d'un point de vue statique et dynamique.

Plan :

II - Concepts Objet

La méthode **Merise** utilise deux types de modèles pour décrire une application :

- Un modèle de données,
- Un modèle de traitement.

L'approche objet repose sur :

- L'**association** des données et des traitements dans une même entité,
- L'**encapsulation** masquant les données et traitements internes,
- Un niveau **abstrait** de manipulation des entités à base de classe et un niveau concret à base d'instance,
- L'**identification** de chaque instance,
- Des niveaux d'**accès** aux données et traitements (publique, privé, implémentation),
- La séparation des **interfaces** de manipulation de l'**implémentation** des traitements,
- Un mécanisme d'**héritage** (généralisation et spécialisation),
- Le **polymorphisme.**

III - Démarche d'un développement

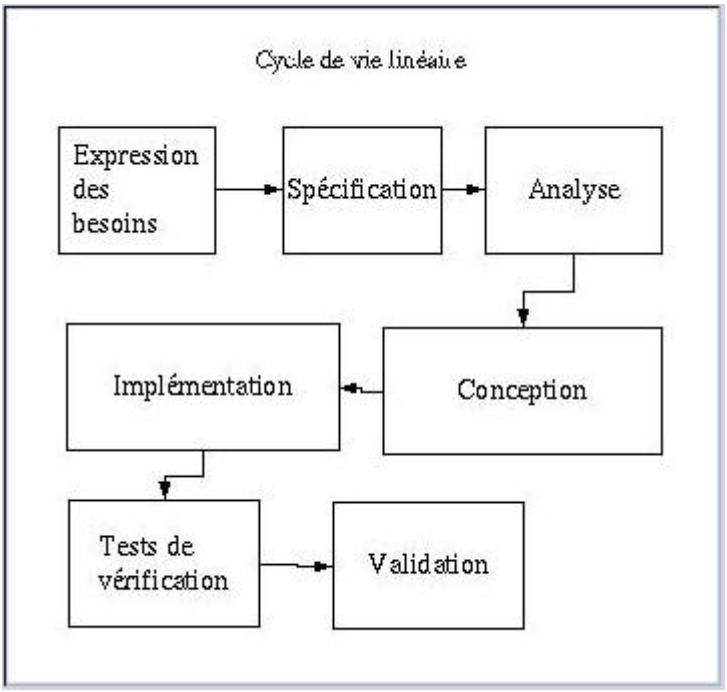
Une démarche de développement repose sur :

- **Un formalisme,**
- **Une méthode,**
- **Un processus et un cycle de vie,**
- **Des buts.**

Les étapes du cycle de vie d'une application :

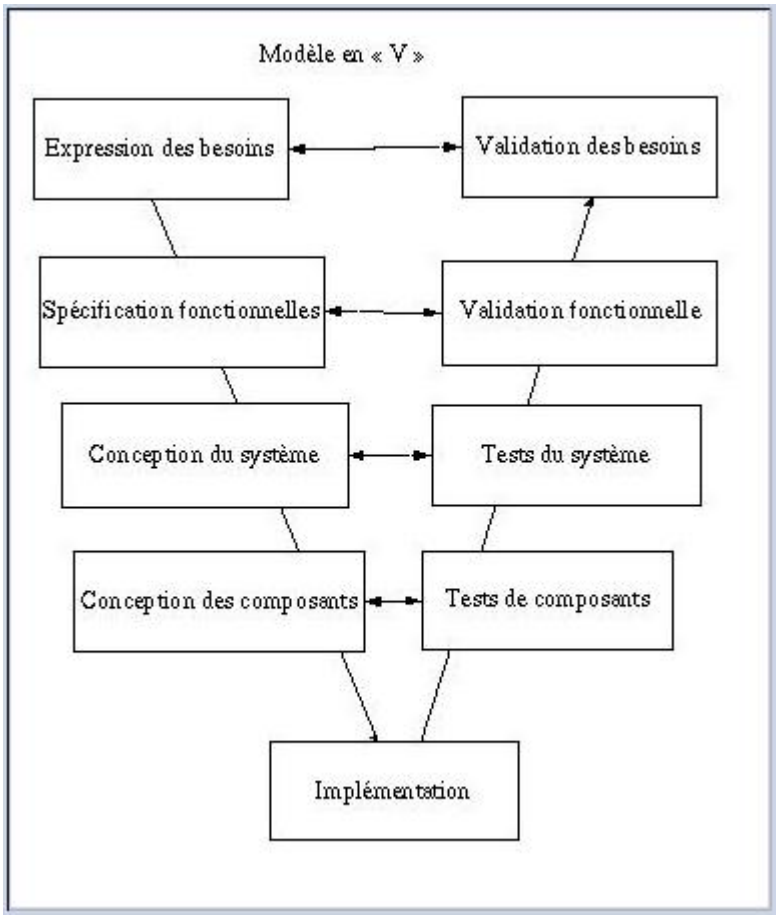
- **Expression des besoins** :Il traduit l'apport du futur système,
- **Spécifications** :Précision avec schémas, modèles et enchainements d'écrans,
- **Analyse** :Détermination des éléments du système,
- **Conception** :Comprend tous les choix techniques,
- **Implémentation** :Génération des squelettes d'une application,
- **Tests de vérification** :Tests unitaires et finals,
- **Validation** :Utilisation d'un cahier de recettes,
- **Maintenance et évolution** :Suivi du logiciel en production.

III-A - Les principaux cycles de vie



Inconvénients :

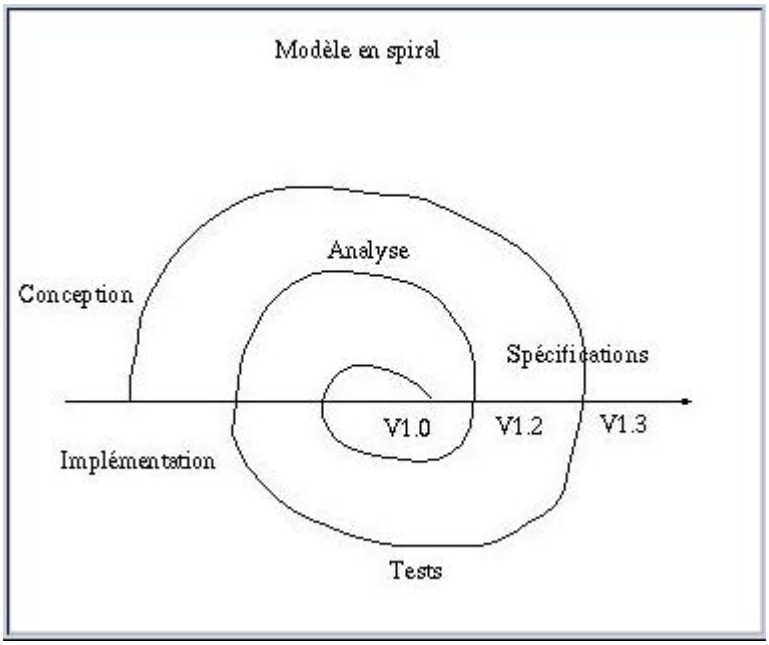
- Pas de travail en parallèle
- Validation tardive



L'un des plus utilisés

Inconvénients :

- Mise en oeuvre tardive
- Erreurs coûteuses



Construction d'une série de prototypes

IV - Phases d'analyse et de conception UML

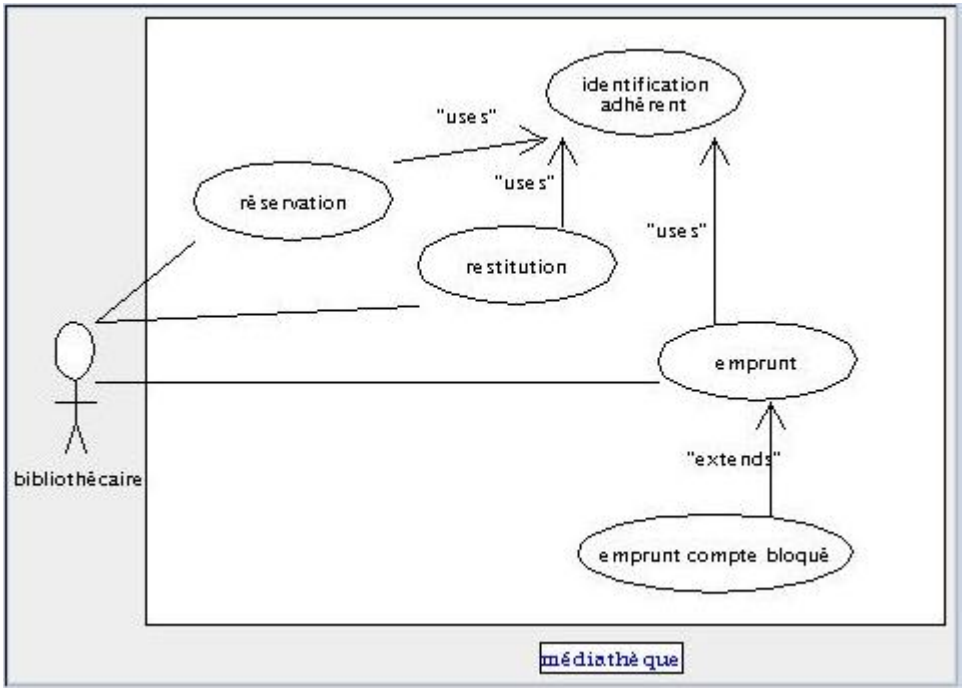
Les étapes d'une construction d'application aboutissent sur l'émergence de **modèles.**

IV-A - Les uses cases

Ils sont issus de la méthode OOSE de **Ivar Jacobson**. Il s'agit d'une représentation orientée "**fonctionnalité**" du système résultant de la **spécification**.

Ils intègrent :

- Des **acteurs** externes au systèmes (primaires ou secondaires)
- Des **actions** (cas d'utilisation) permises par le système



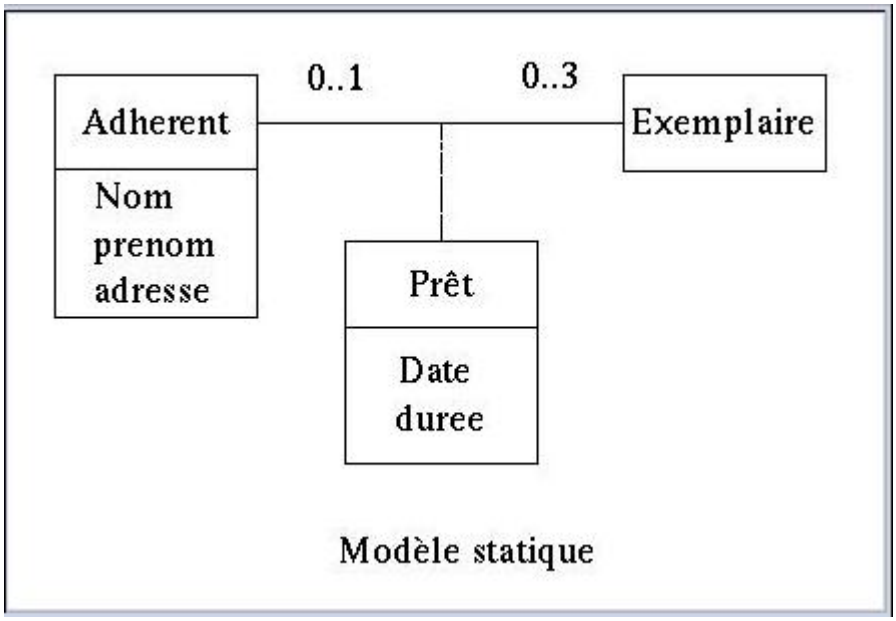
Il est possible de simplifier les actions avec deux types de relations :

- La relation "**extends**" : Intégre les caractéristiques du **use case** pointé
- La relation "**uses**" : Indique qu'un **use case** fait partie d'un sous-ensemble

IV-B - Le modèle statique (objet)

Il est utilisé en **phase d'analyse** sans rapport avec l'implémentation.

Il décrit les **entités** et leurs **relations** en terme objet.



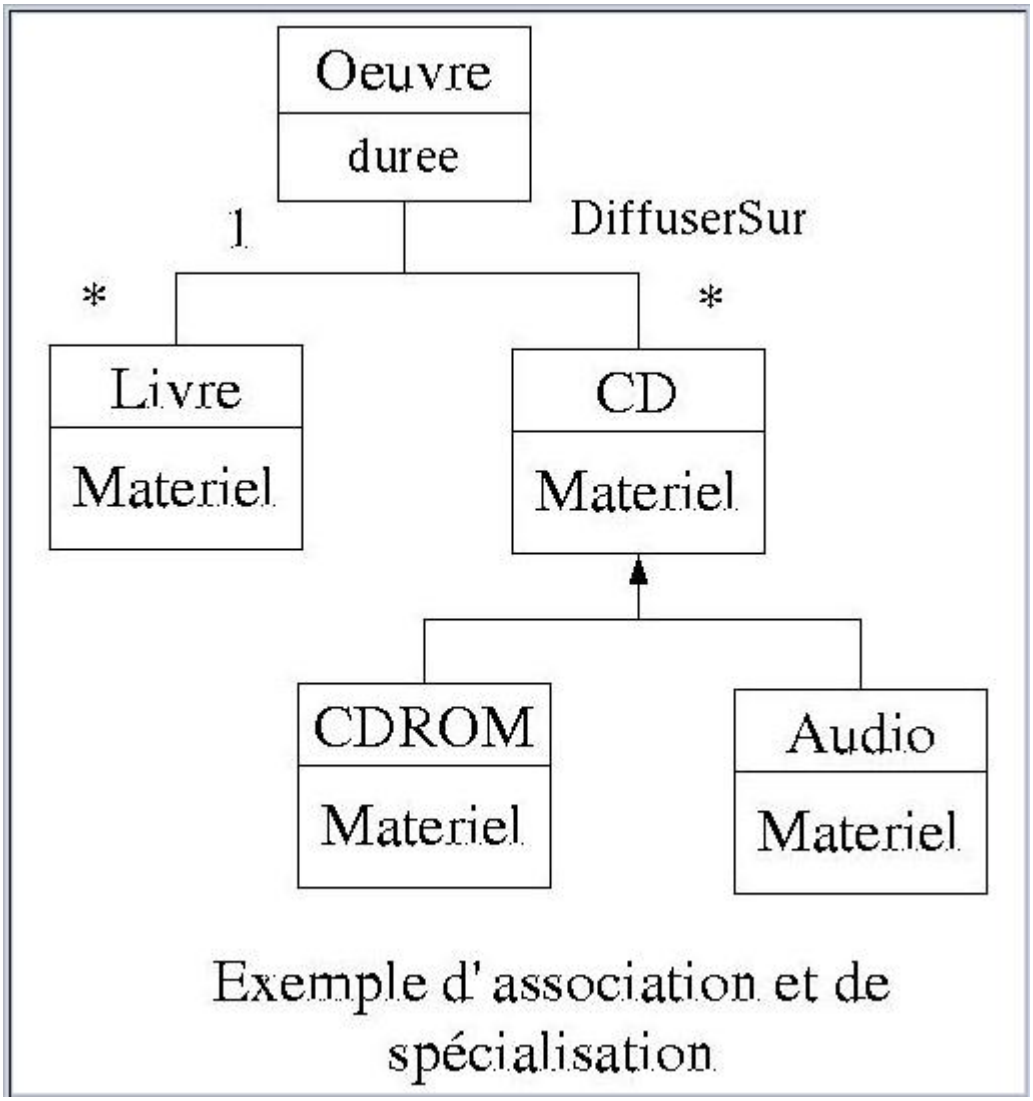
Les attributs d'une entité n'ont pas de type en rapport avec un langage de programmation. Les cardinalités représentent le nombre d'instances impliquées dans l'association.

Quelques cardinalités :

- **0..1** : Peut avoir aucune ou une instance
- *: Peut avoir aucune ou plusieurs instances

Les entités représentent des classes, il est possible de représenter des instances en les préfixant par

classe :: .

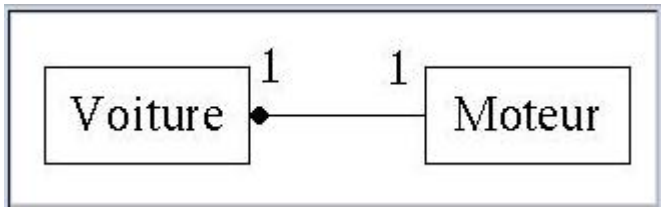


La spécialisation ajoute une relation de type "**est une sorte de**".

Pour renforcer **l'association, l'agrégation** fait intervenir l'idée de **dépendance** pour une instance.

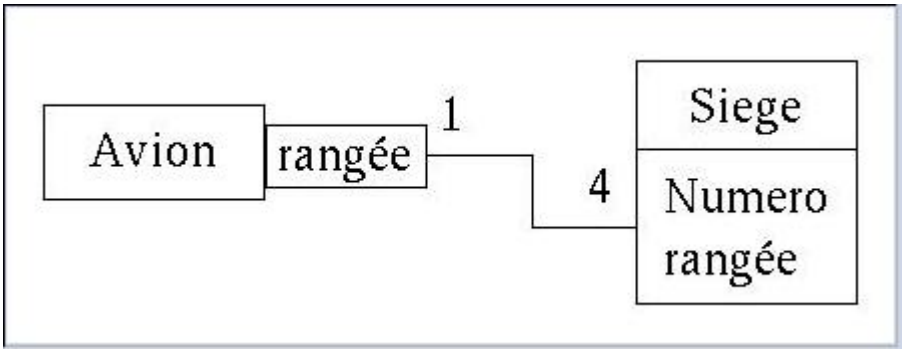
Elle introduit une relation de type "**fait partie de**".

Exemple:



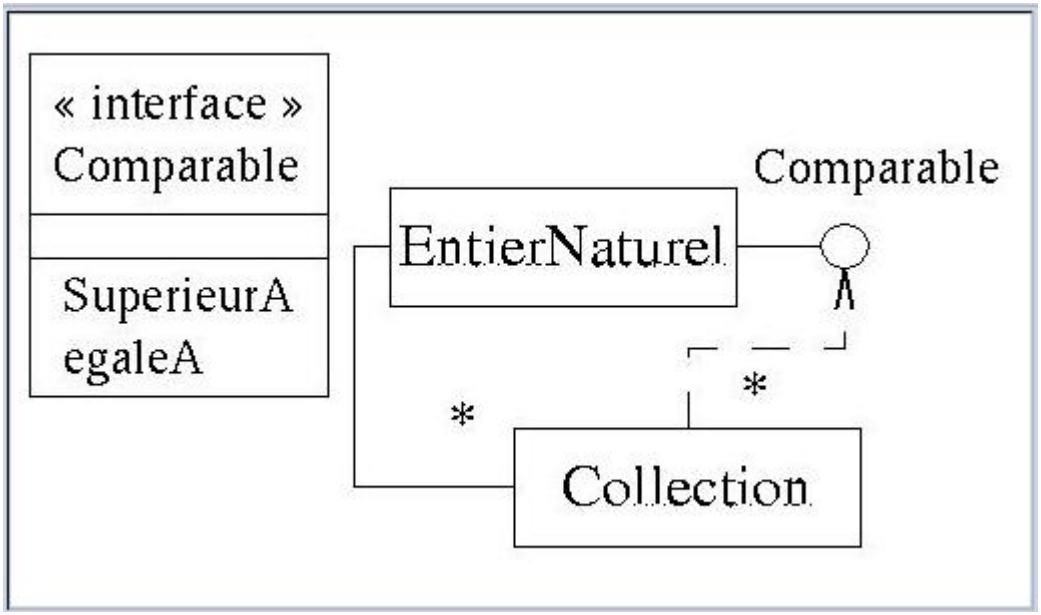
Les **qualificateurs** peuvent améliorer les domaines de valeur des attributs.

Exemple : Pour une rangée d'avion il existe 4 sièges.

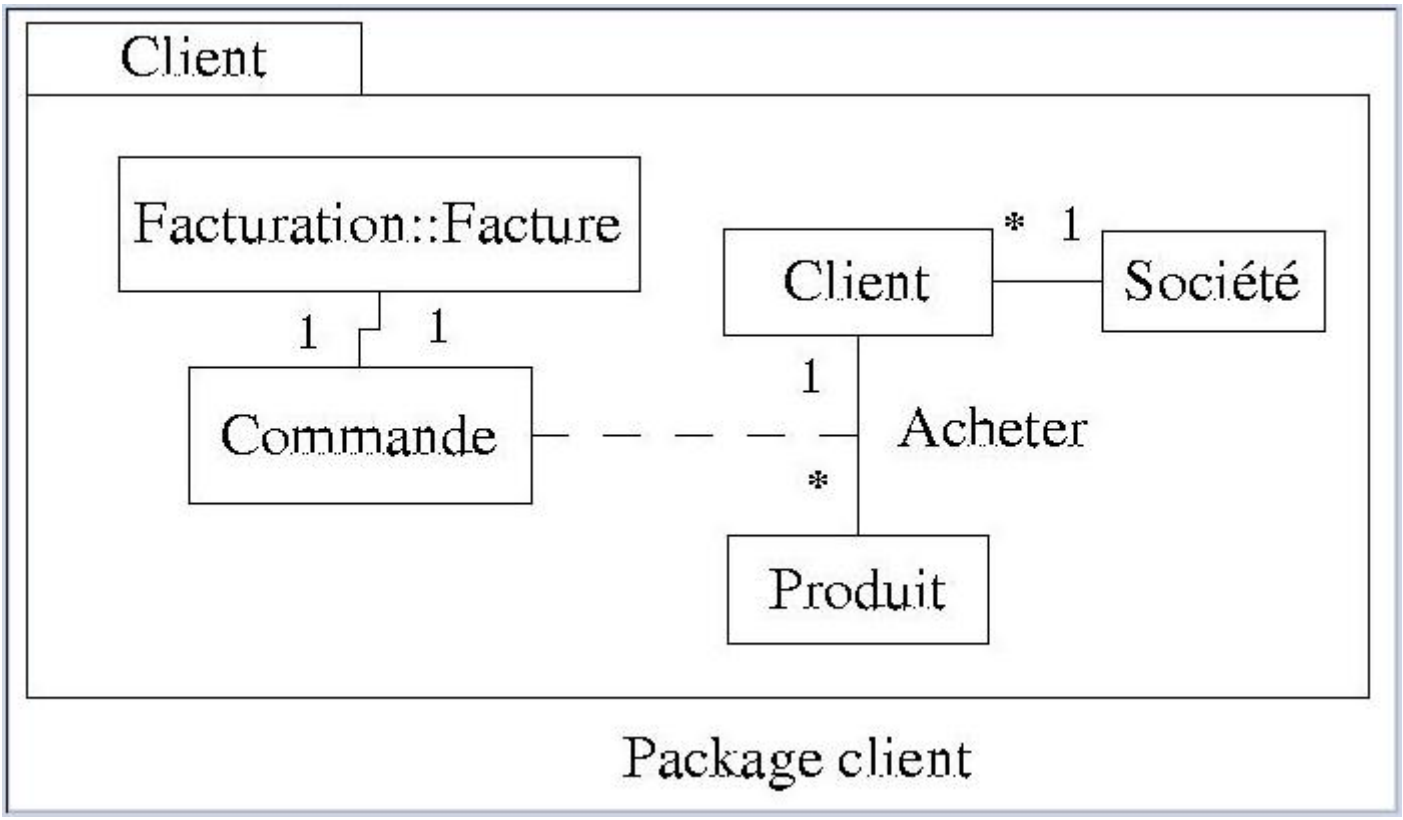


Un **stéréotype** sert à subdiviser les classes en grandes familles.

Exemple : "interface".



Les **packages** décrivent les relations macroscopiques entre partie du système.



Un **package** peut aussi se représenter sans contenu en ne mettant en évidence que les relations importantes.

Le détail des entités peut être défini dans les modèles statiques en analyse. Cependant, seule la conception prendra en compte tous les impératifs du développement.

Syntaxe des attributs de classe :

```
[visibilité] nom [ [:type] [=valeur initial] [{propriété}]
```

Visiblité peut prendre les valeurs :

- **-**: Champ privé
- **+** : Champ publique
- #: Champ protected

Propriété affine l'utilisation des attributs :

- **changeable** : Attribut en lecture/écriture
- **frozen** : Constante
- **readOnly** : Attribut en lecture

Syntaxe des méthodes de classe :

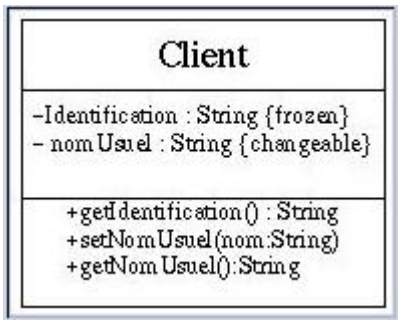
```
[visibilité] nom [(liste des paramètres)] [: type de la valeur de retour] [{propriété}]
```

Visiblité peut prendre les valeurs :

- **-**: Champ privé
- **+** : Champ publique
- #: Champ protected

Une méthode soulignée désigne une **méthode de classe.**

Liste de paramètres sous la forme : paramètre1 : Type1, paramètre2 : Type2,...



IV-C - Le modèle dynamique

Le modèle dynamique est une **vision microscopique** du fonctionnement du système. Il sert à mettre en évidence les **relations temporelles** inter-objets et la représentation sous forme d'un **automate** du comportement de chaque objet.

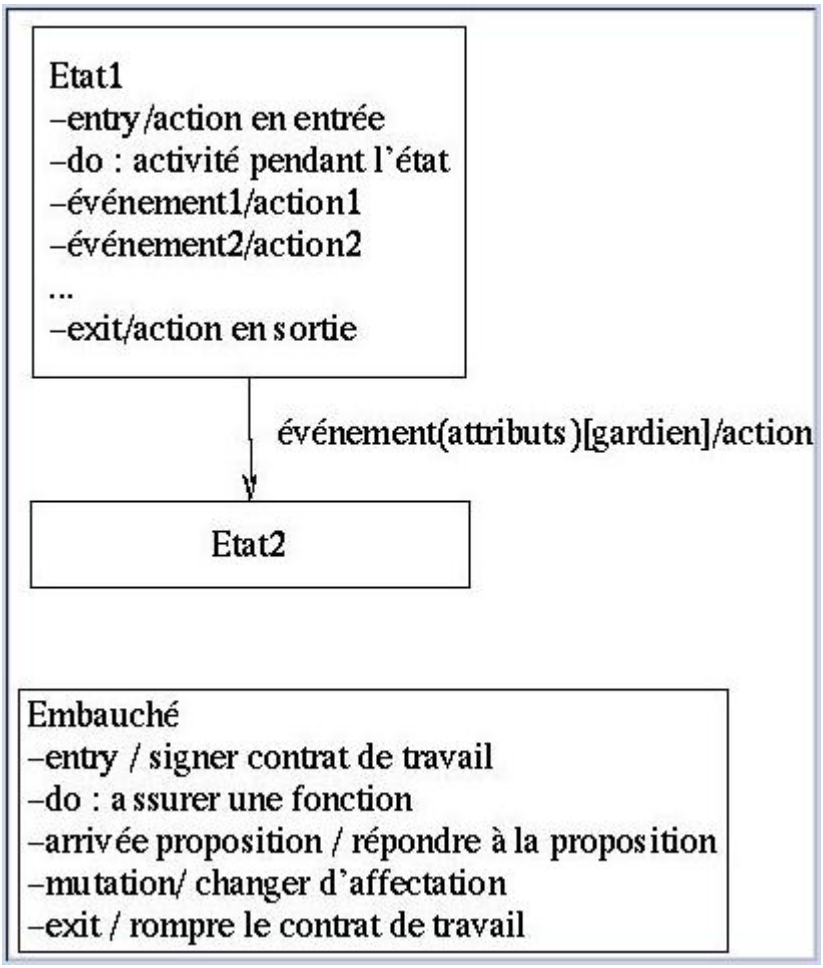
Il intervient aprés la définition du modèle statique

IV-C-1 - Le diagramme d'état

Il découpe un objet en un ensemble d'**états**. Le passage d'un état à un autre se fait par des **événements**.

Un **événement** est composé d'**attributs** et de **gardiens**. Un **attribut** est un **paramètre** alors qu'un **gardien** est une **condition** pour réaliser une transition.

Exemple :



Dans l'exemple ci-dessous, l'état initial est représenté par un rond noir, l'état final par un rond noir dans un cercle.

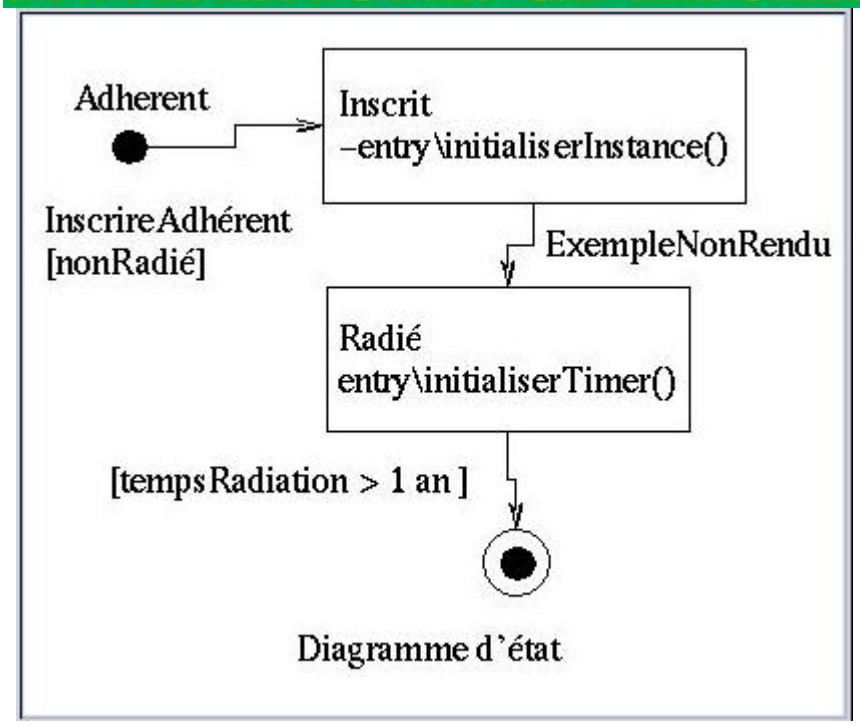
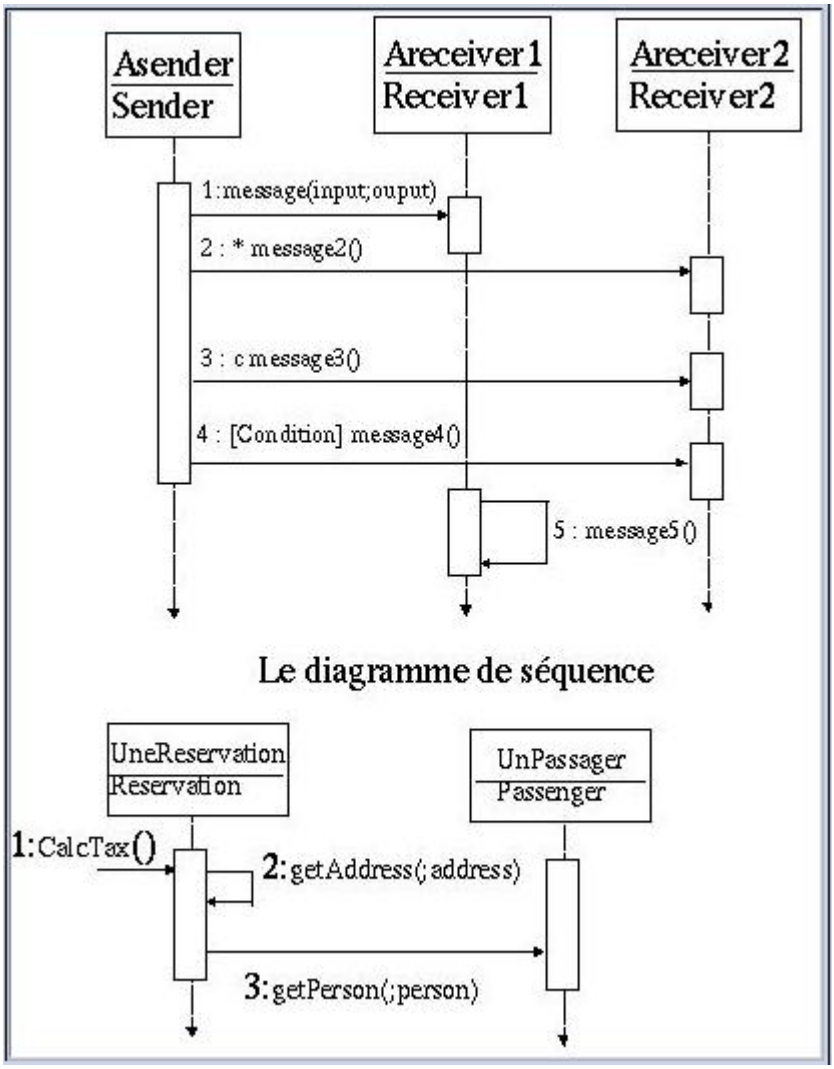


Diagramme d'état

IV-C-2 - Le diagramme de séquence

Ce diagramme met en évidence les relations temporelles inter-objet.

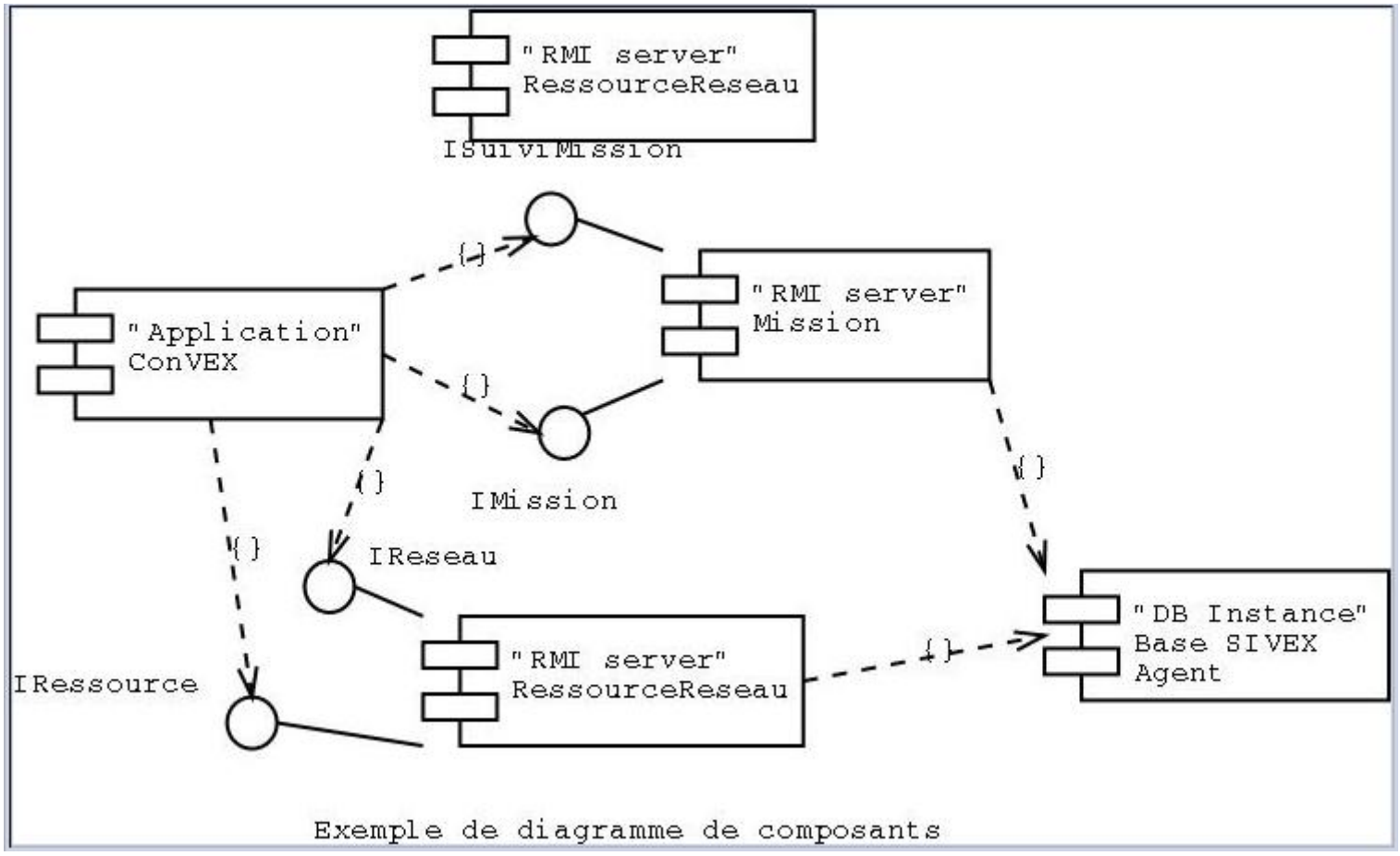


IV-D - La conception

Elle met en place les **grands blocs applicatifs**

La conception reprend les modèles de l'analyse statique et dynamique et **détaille** l'organisation des classes dans un langage cible. Elle utilise les **design-pattern**

IV-D-1 - La conception préliminaire



Exemple de diagramme de composants

Les composants peuvent représenter n'importe quel partie importante d'une architecture (Servlet, Base de données...). Les relations inter-composants sont garanties par des interfaces pour un diagramme plus proche de l'implémentation. Un **stéréotype** classe en grandes familles les composants :

- **"SGBDR"**
- **"EXE"**
- **"Servlet"**
- **"WEB"**
- **...**

IV-D-2 - La conception détaillée

Elle fait intervenir les étapes :

- Conception des classes
- Conception des associations
- Conception des attributs
- Conception des opérations
- Validation du modèle

Elle peut être réalisée en partie par des outils de génération de code pour les diagrammes de classes, d'états et de séquences.

Quelques outils **UML** pour Java :

Payant :

Magic Draw (http://www.nomagic.com)

TogetherJ (http://www.togethersoft.com)

Rational Rose 2000 (http://www.rational.com)

Gratuit :

ArgoUml (http://argouml.tigris.org)

V - Bibliographie

"**UML en Action**" de Pascal Roques et Franck Vallée chez Eyrolles

"**Intégrer UML dans vos projets**" de M.Lopez, E.Pichon et J.Migueis chez Eyrolles

Les schémas ont été réalisés avec **dia (http://wwww.lysator.liu.se/~alla/dia)** et **starOffice 5.2 (http://www.staroffice.org)**.