- 3.5 -

Modélisation structurelle (2)

UNIVERSITÉ DE LORRAINE

Analyse et Conception de Logiciels

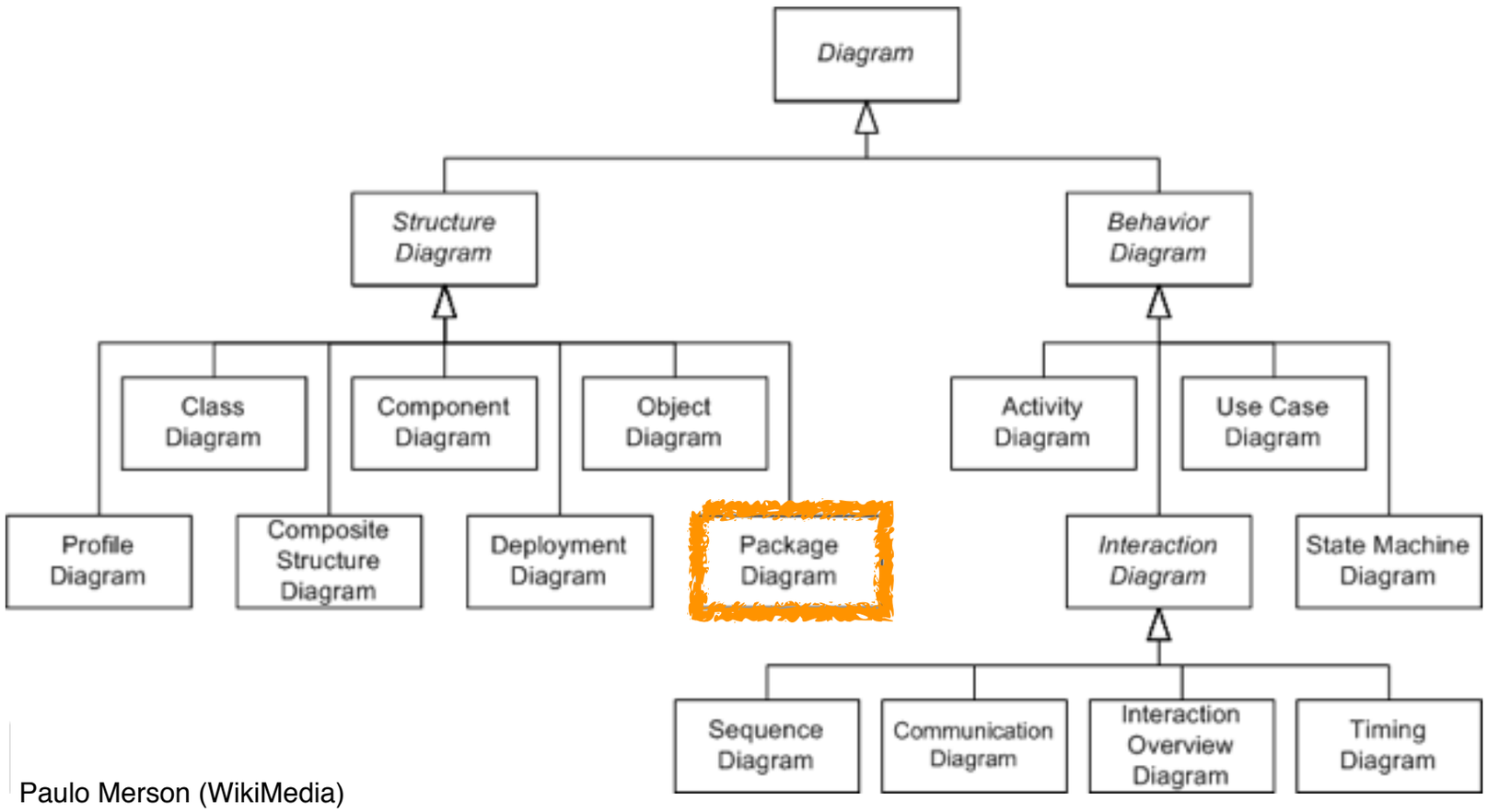
Horatiu CIRSTEA

Exemples du cours

Une partie inspirés de

- Object-Oriented Software Engineering: Practical Software Development using UML and Java - Timothy C. Lethbridge, Robert Laganière
- UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. - Martin Fowler
- UML 2 - de l'apprentissage à la pratique - Laurent Audibert, Ellipses, 2009
- UML 2 par la pratique - Pascal Roques, Eyrolles, 2009

Modélisation avec des diagrammes de package

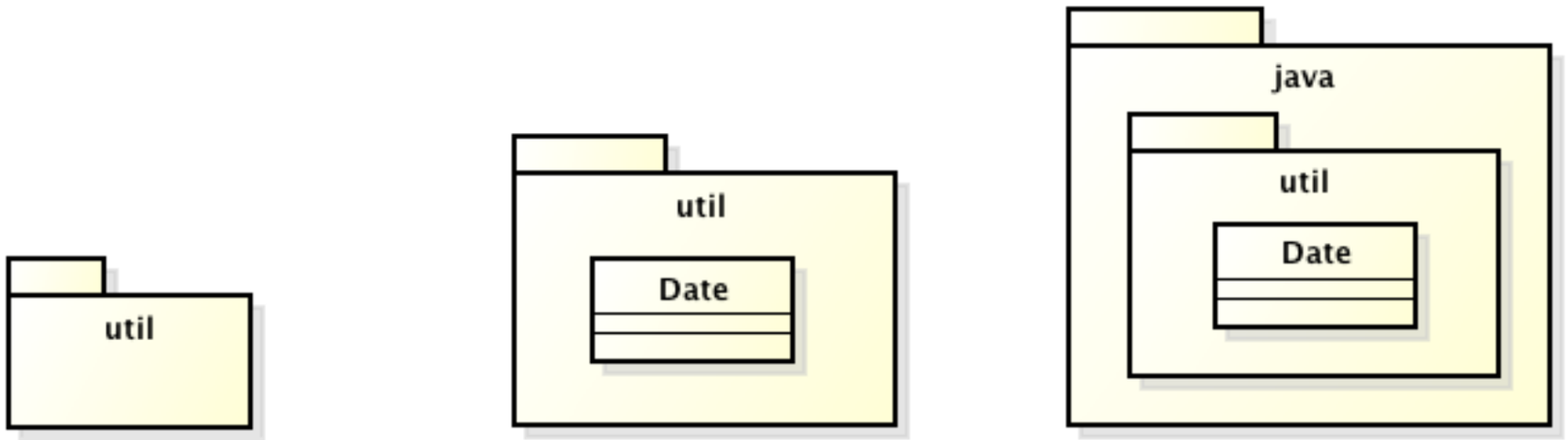


Paulo Merson (WikiMedia)

Diagrammes de package

- organiser les éléments du modèle dans des groupes
- les diagrammes devient plus simples (à comprendre)
- applicable à tous les types de diagrammes
- peuvent être imbriquées
- un package représente un « *namespace* »
  - chaque classe doit avoir un nom unique dans son package

Diagrammes de package



- on peut utiliser des noms « fully qualified »

➡ représentent un mécanisme pour grouper les classes au *compile-time*

Dépendances

- plusieurs types
  - use
  - import (package ou classe)
  - access (private import)
  - merge
- une *bonne* structure de packages a un flux de dépendances clair

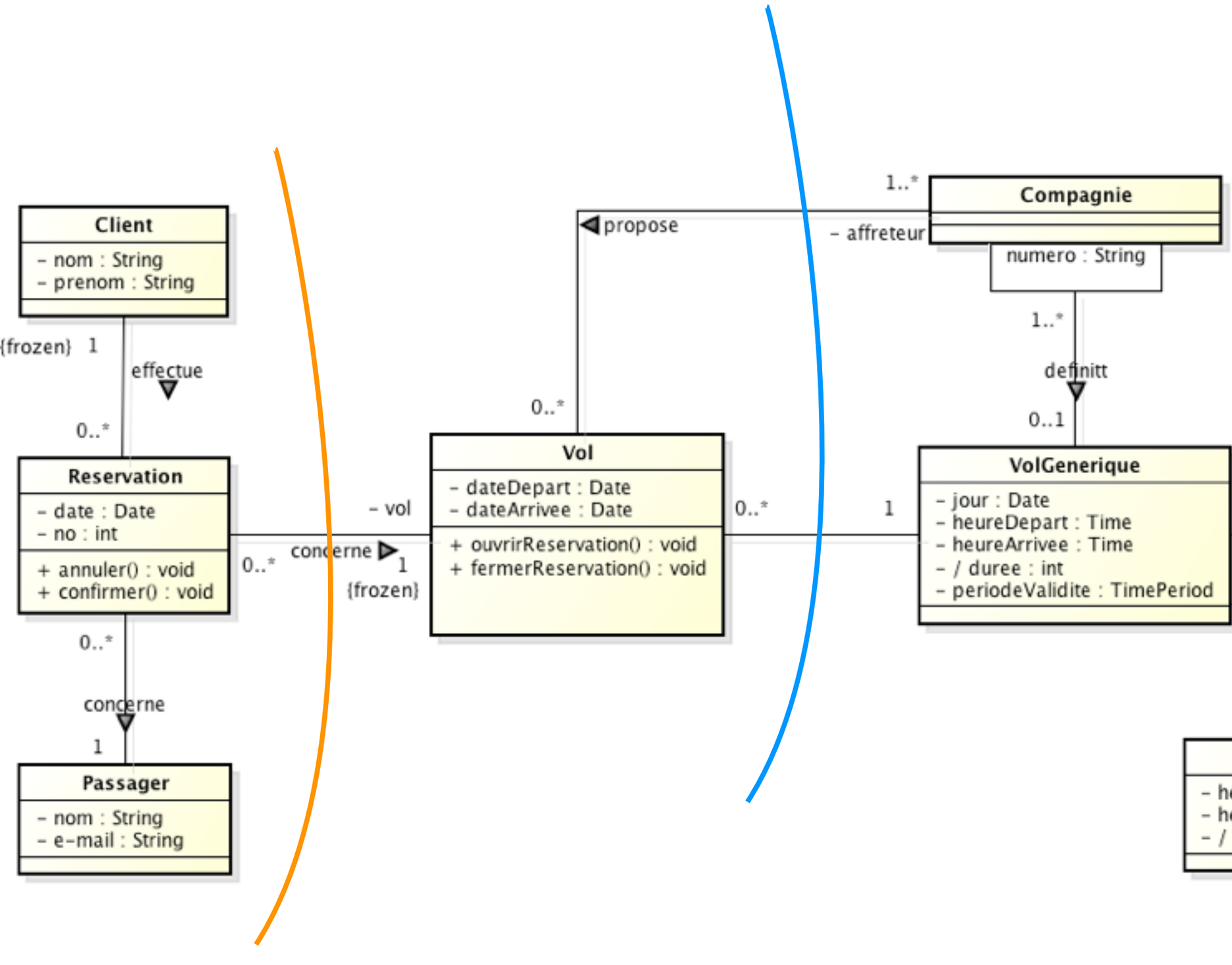
Comment grouper ?

- cohérence : regrouper les classes proches sémantiquement
  - **finalité :** les classes doivent rendre des services de même nature
  - **évolution :** isoler les classes réellement stables de celles qui évoluent
  - **cycle de vie des objets :** distinguer les classes dont les objets ont des durées de vie très différentes
- indépendance

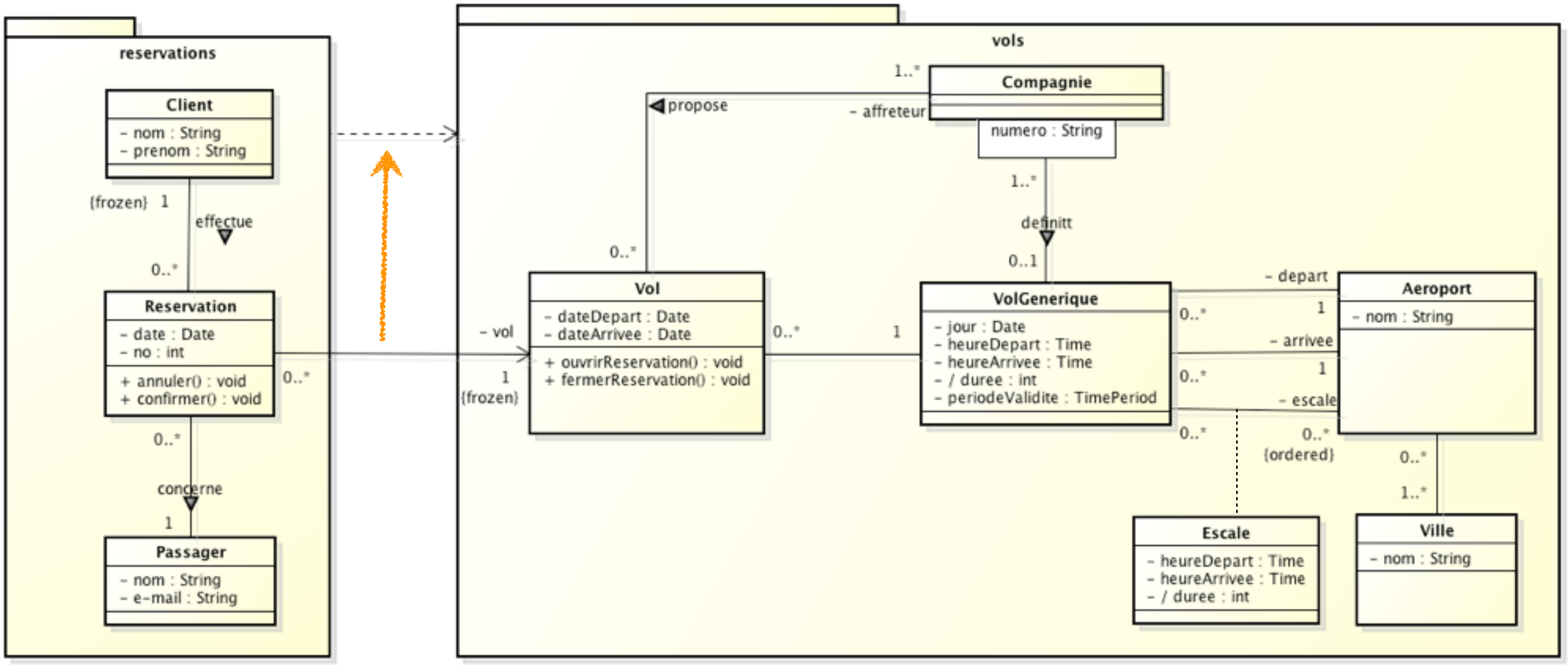
Comment grouper ?

- *Common Closure Principle*
  - les classes dans un package doivent avoir besoin d'évoluer pour les mêmes raisons
- *Common Reuse Principle*
  - les classes dans un package doivent être réutilisées ensemble

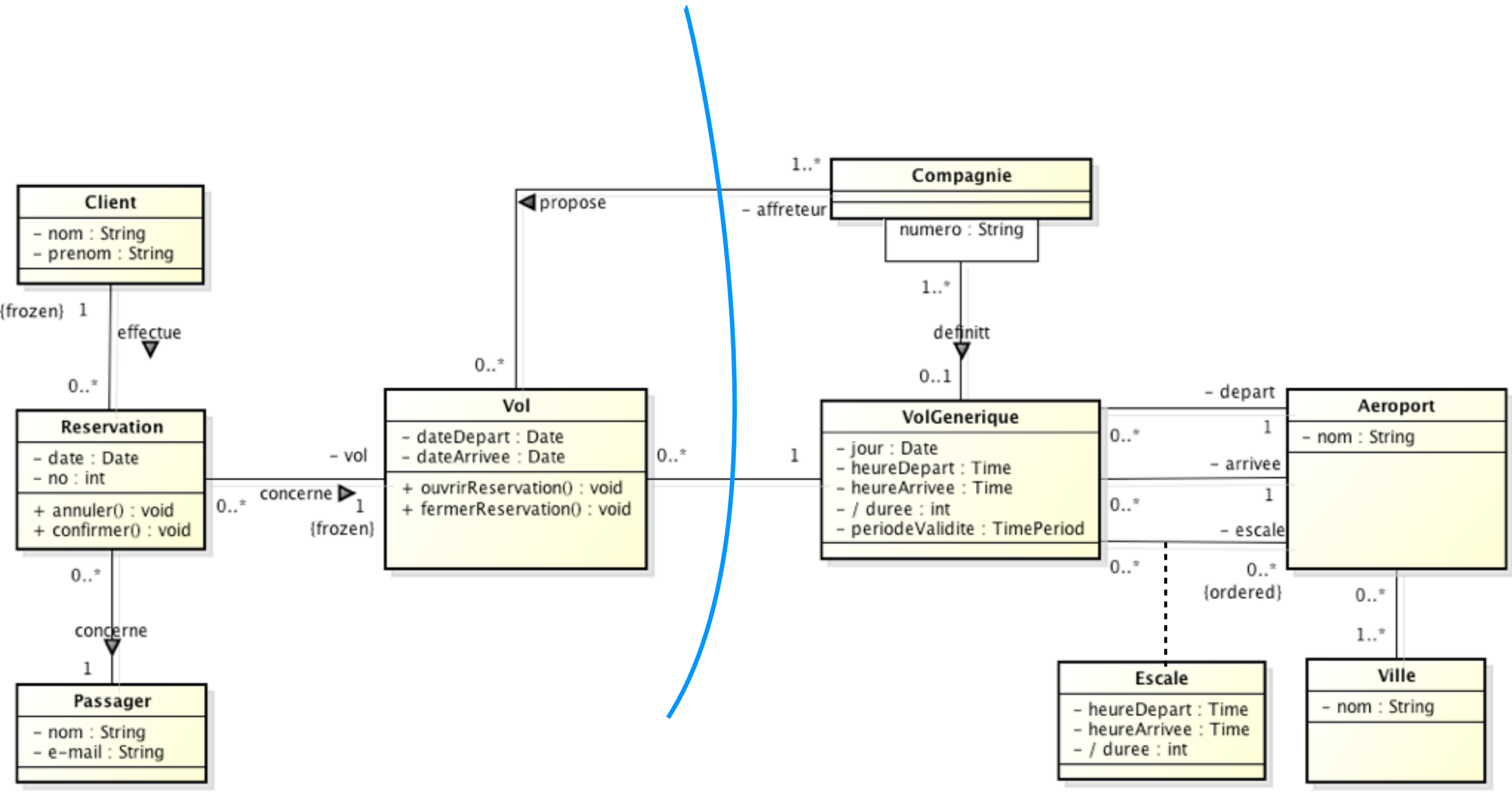
Exemple: Vols



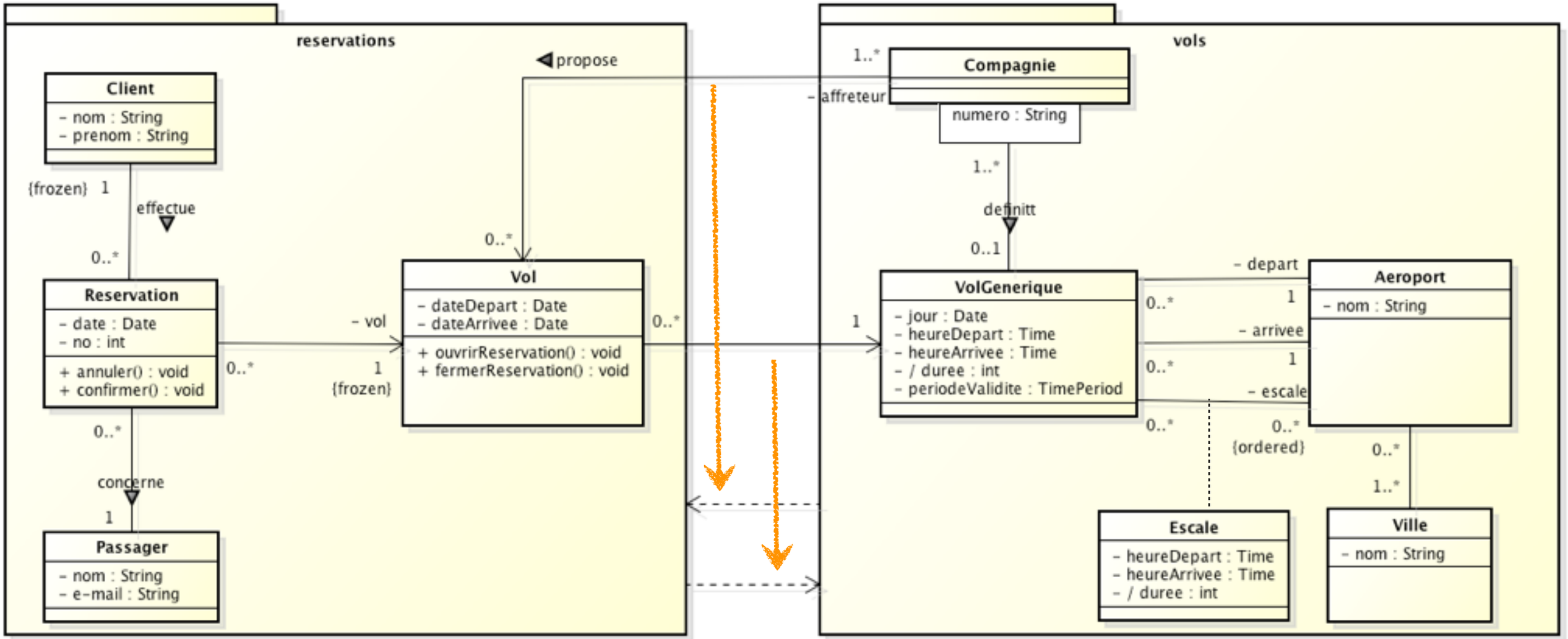
Exemple: Vols



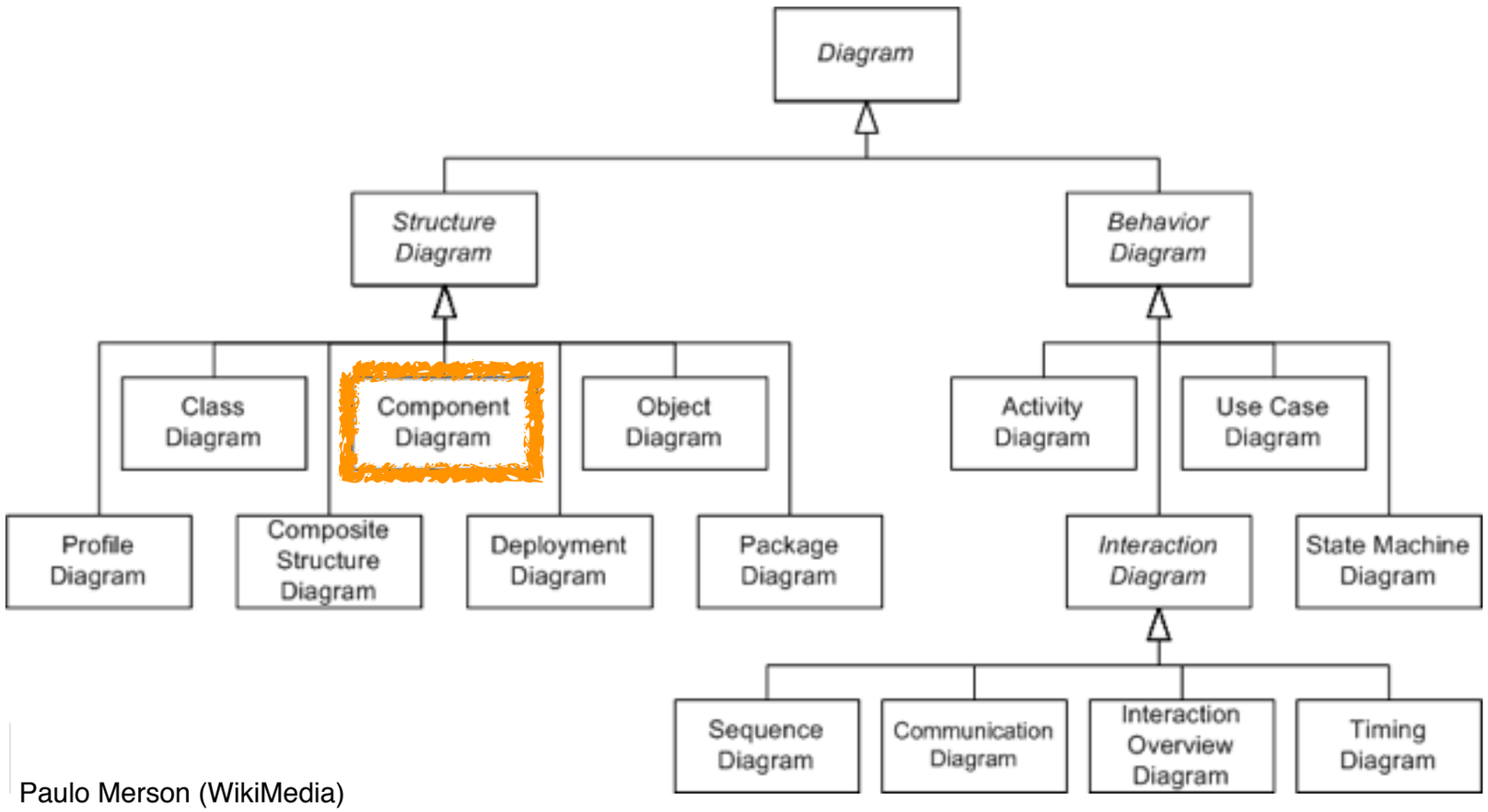
Exemple: Vols



Exemple: Vols



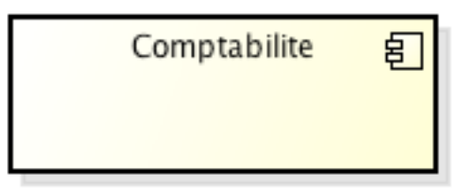
Modélisation avec des diagrammes de composants

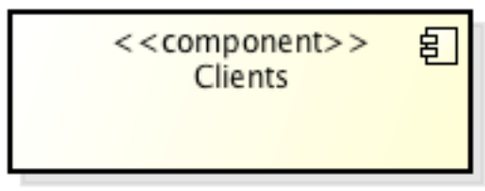


Paulo Merson (WikiMedia)

Diagrammes de composants

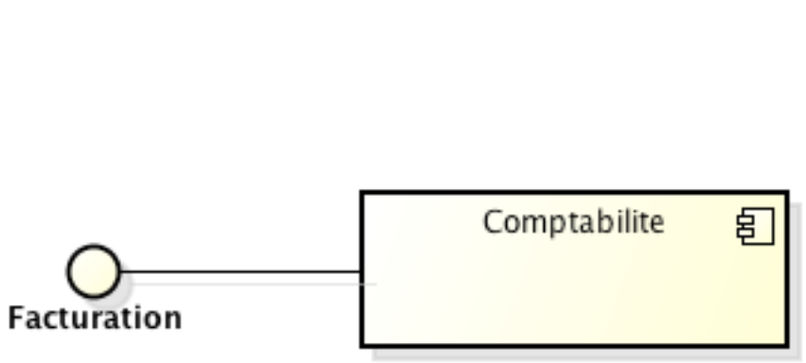
- permettent la description des composants logiciels de haut niveau et leurs interactions
- *composant*
  - partie d'un système modulaire avec un contenu encapsulé
  - remplaçable dans son environnement

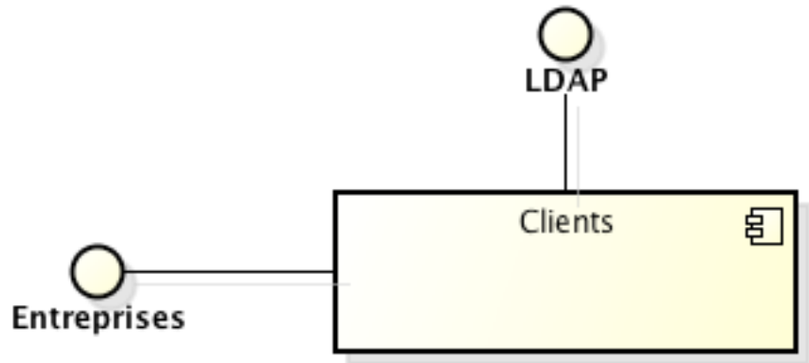




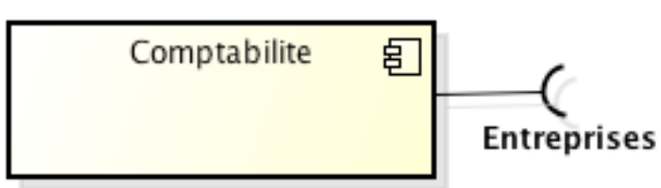
Comportement composant

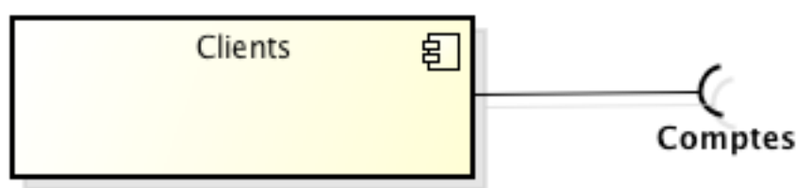
- interfaces offertes





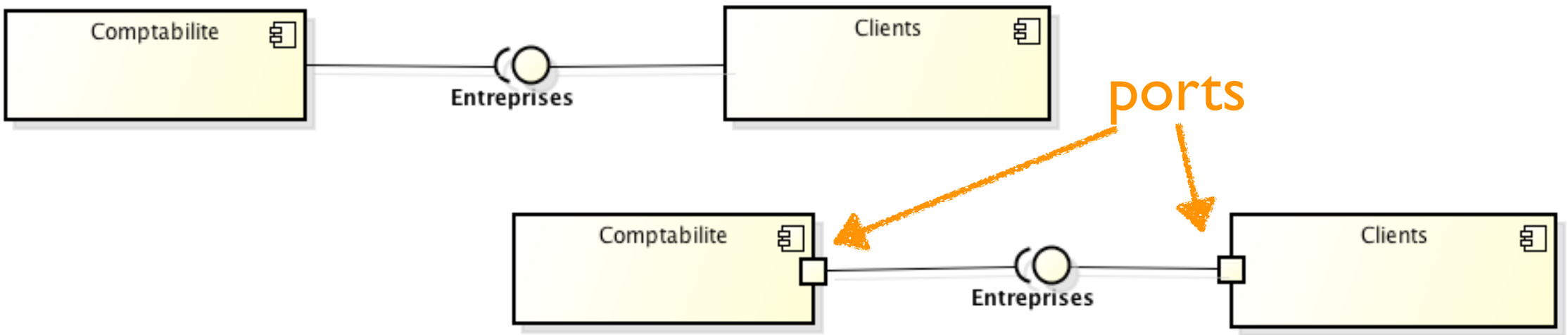
- interfaces requises



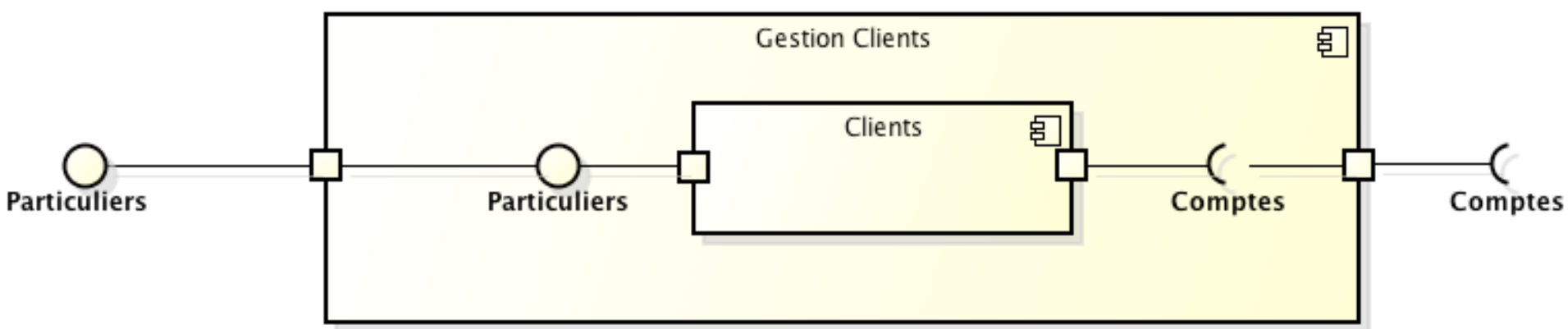


Connecteurs

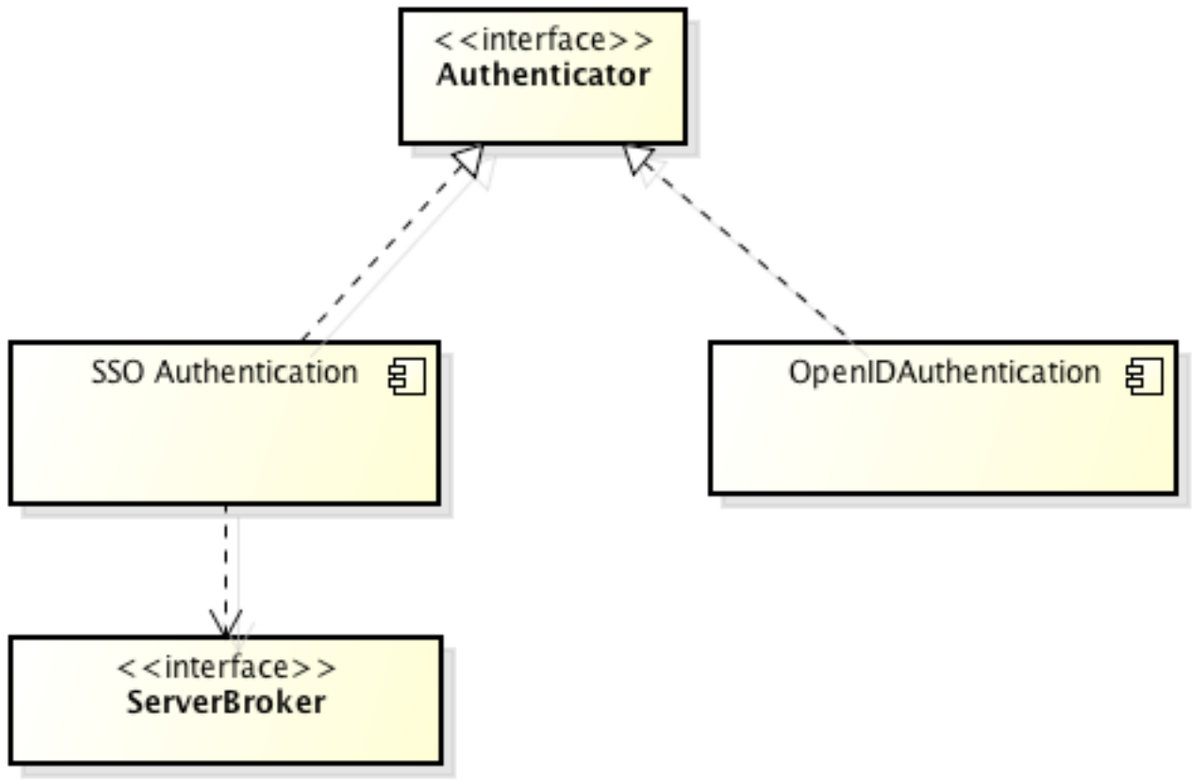
- assemblage



- délégation

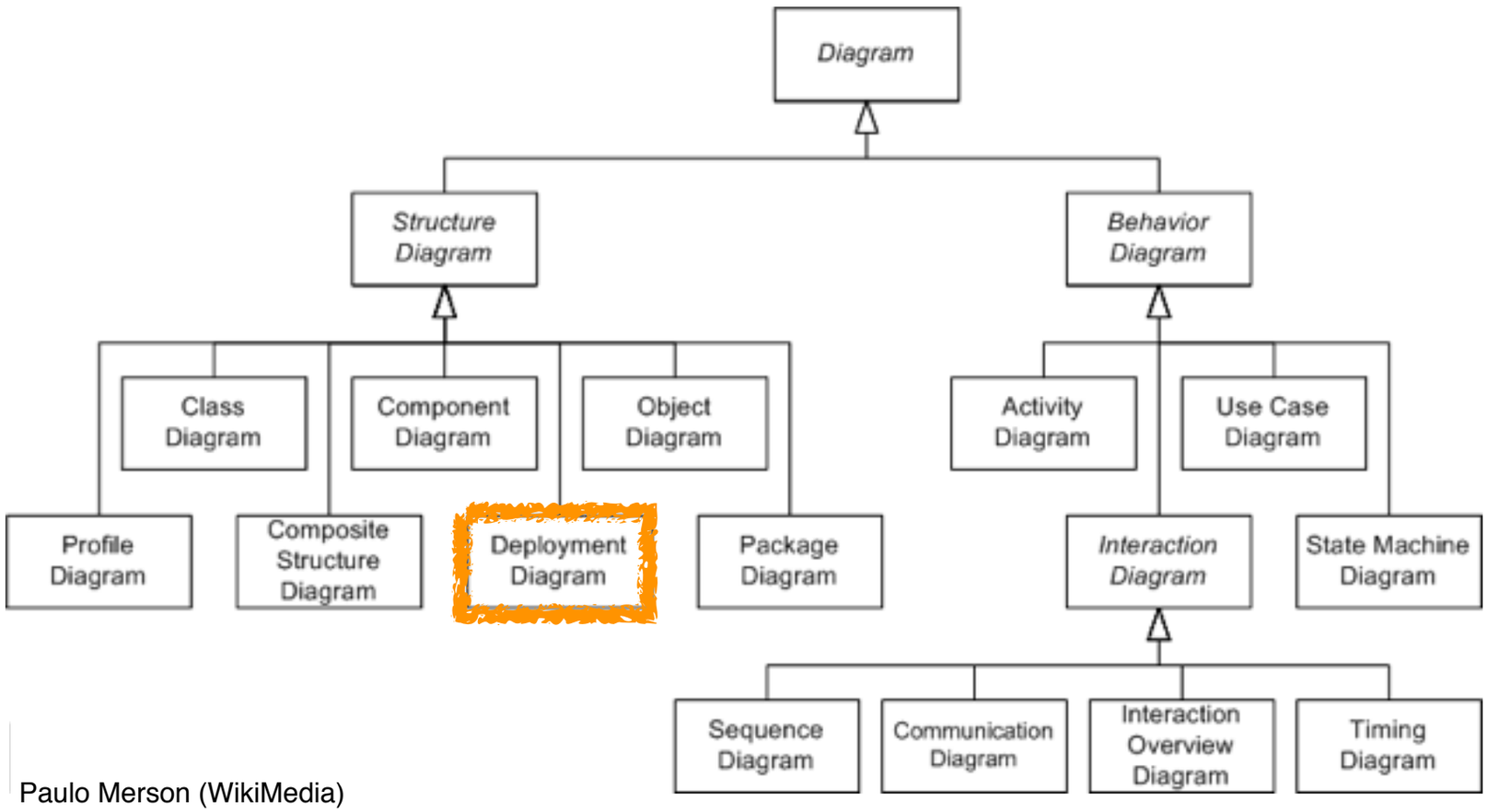


Réalisation et utilisation explicite



- **on peut utiliser des stéréotypes standards pour les composants :** subsystem, process, service, specification, realization, implement

Modélisation avec des diagrammes de déploiement



Paulo Merson (WikiMedia)

Diagrammes de déploiement

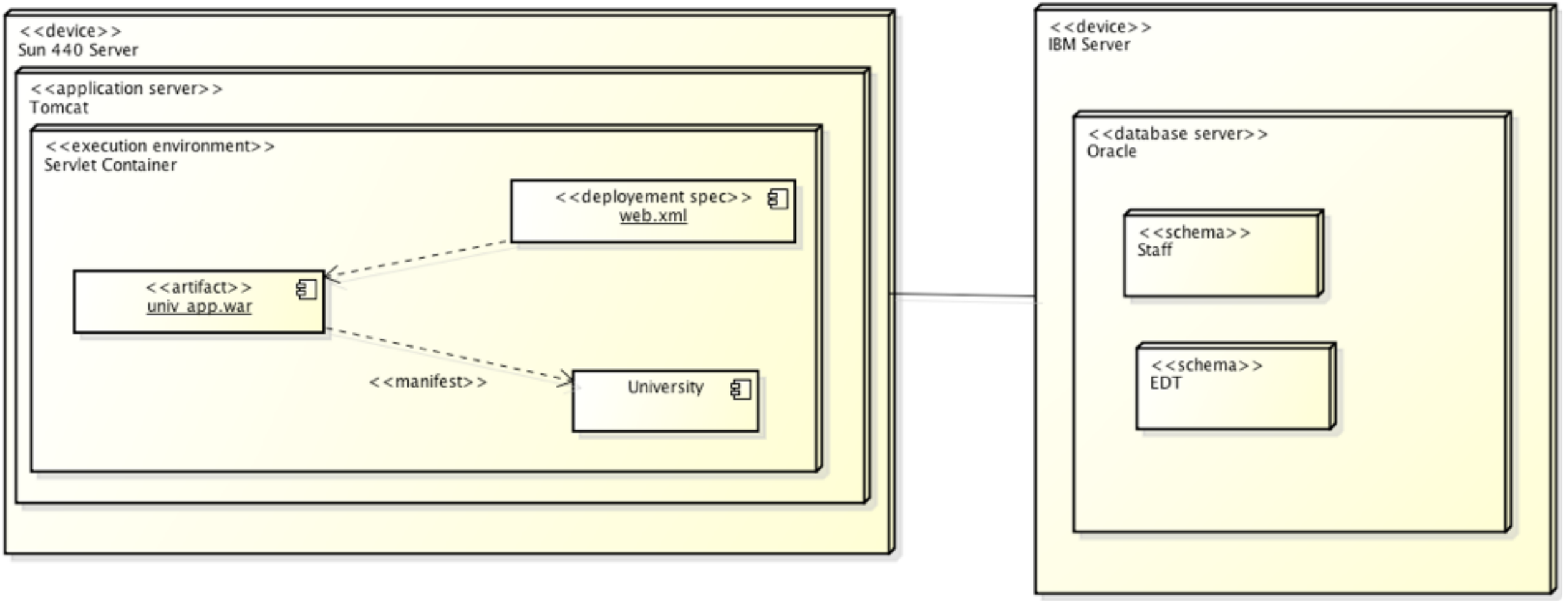
- disposition physique des ressources matérielles qui composent le système
- répartition des composants sur ces matériels
- *déploiement de artefacts logiciels sur des cibles*
  - *artefacts* → elements concrets dans le monde réel : fichier executables, librairies, BDD, fichiers config, ...
  - *cibles* → representées par des ***nodes*** (device hardware ou environement software d'execution) reliés par des chemins de communication

Diagrammes de déploiement

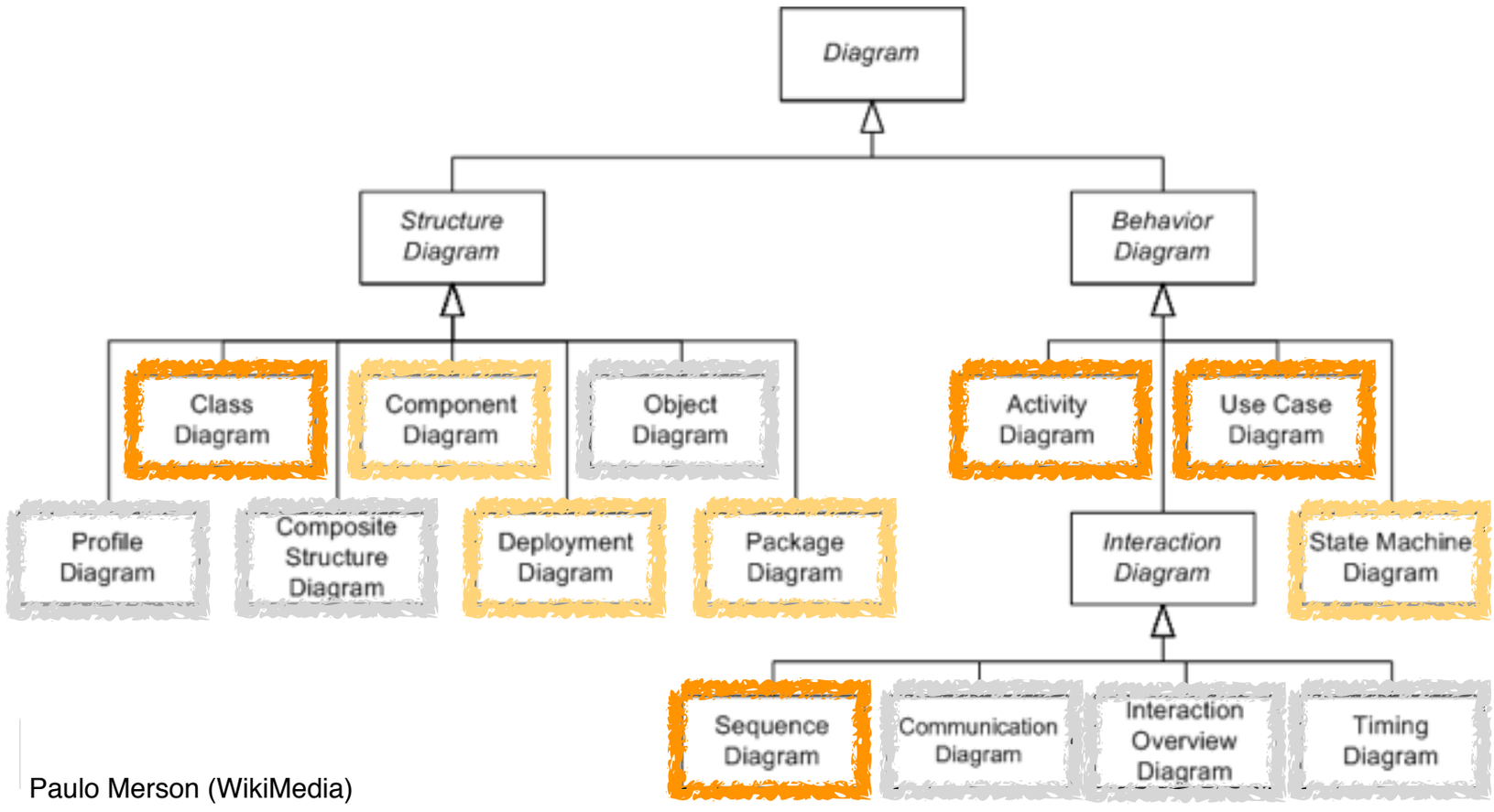
- types usuels de diagrammes de déploiement
  - implémentation (manifestation) de composants par artefacts
  - diagramme de déploiement niveau spécification
  - diagramme de déploiement niveau instance
  - architecture réseau du système

Diagrammes de déploiement

- diagramme de déploiement niveau spécification



Diagrammes UML



Paulo Merson (WikiMedia)

Questions ?