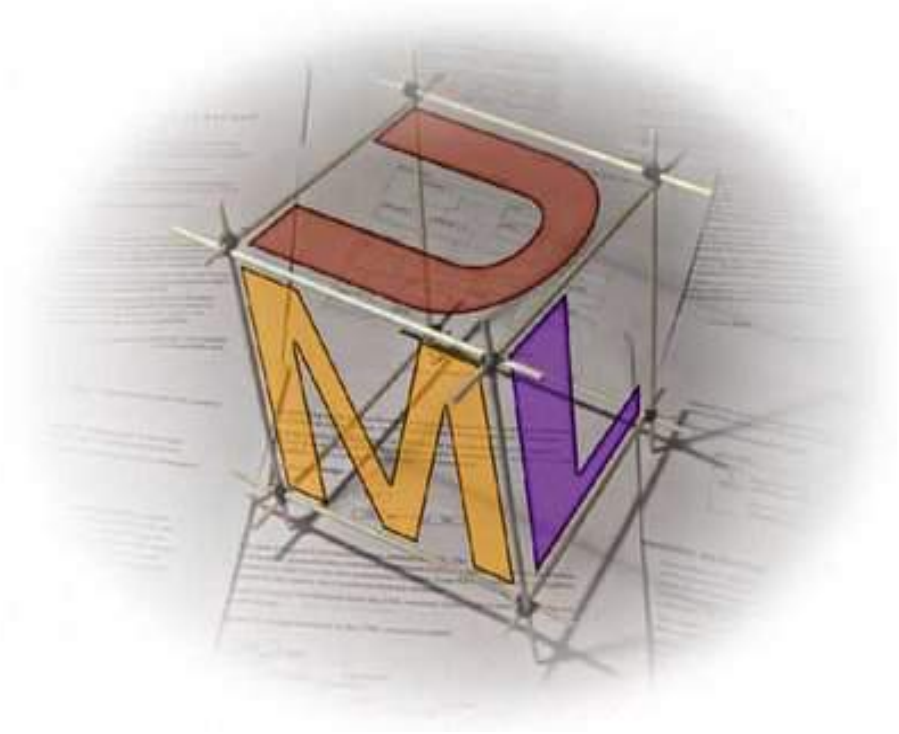
L'approche Orientée Objet et UML

UML

Rémy Courdier



UML
Les Diagrammes
De Modélisation

Remy.Courdier@univ-reunion.fr

Plan du cours

UML

- Introduction au Génie Logiciel
- L'approche Orientée Objet et Notation UML
- **Les diagrammes de modélisation**
- Relations entre les différents diagrammes
- De l'analyse à la conception
- Relation entre les notations OMT et UML

Chapitre 3 : Les diagrammes de modélisation

UML

Diagrammes : Moyens de visualiser et de manipuler des éléments de modélisation avec UML

- **Les diagrammes de classes**
- **Les diagrammes d'objets**
- **Les diagrammes de cas d'utilisation**
- **Les diagrammes de séquence**
- **Les diagrammes d'états-transitions**
- **Les diagrammes d'activités**
- **Les diagrammes de composants**
- **Les diagrammes de déploiement**

3.1. Diagramme de classes

Domain Model Diagram

UML

Diagramme de Domaine
(Domain Model Diagram)

Le modèle de domaine est une représentation de concepts significatifs du monde réel et qui concernent le domaine à modéliser dans le logiciel.

- structure abstraite statique en terme de classes et de relations.
- description abstraite des liens potentiels d'un objet vers d'autres objets.

Diagramme d'implémentation
(Diagram of Implementation Classes)

Le diagramme d'implémentation est réalisé en second temps en détaillant l'ensemble des attributs de classes générer un code informatique.

- Précise les types de variables manipulées et les paramètres des opérations
- Précise les règles de visibilités des attributs de classes
- Permet de générer le code informatique des squelette des classes
- description abstraite des liens potentiels d'un objet vers d'autres objets.

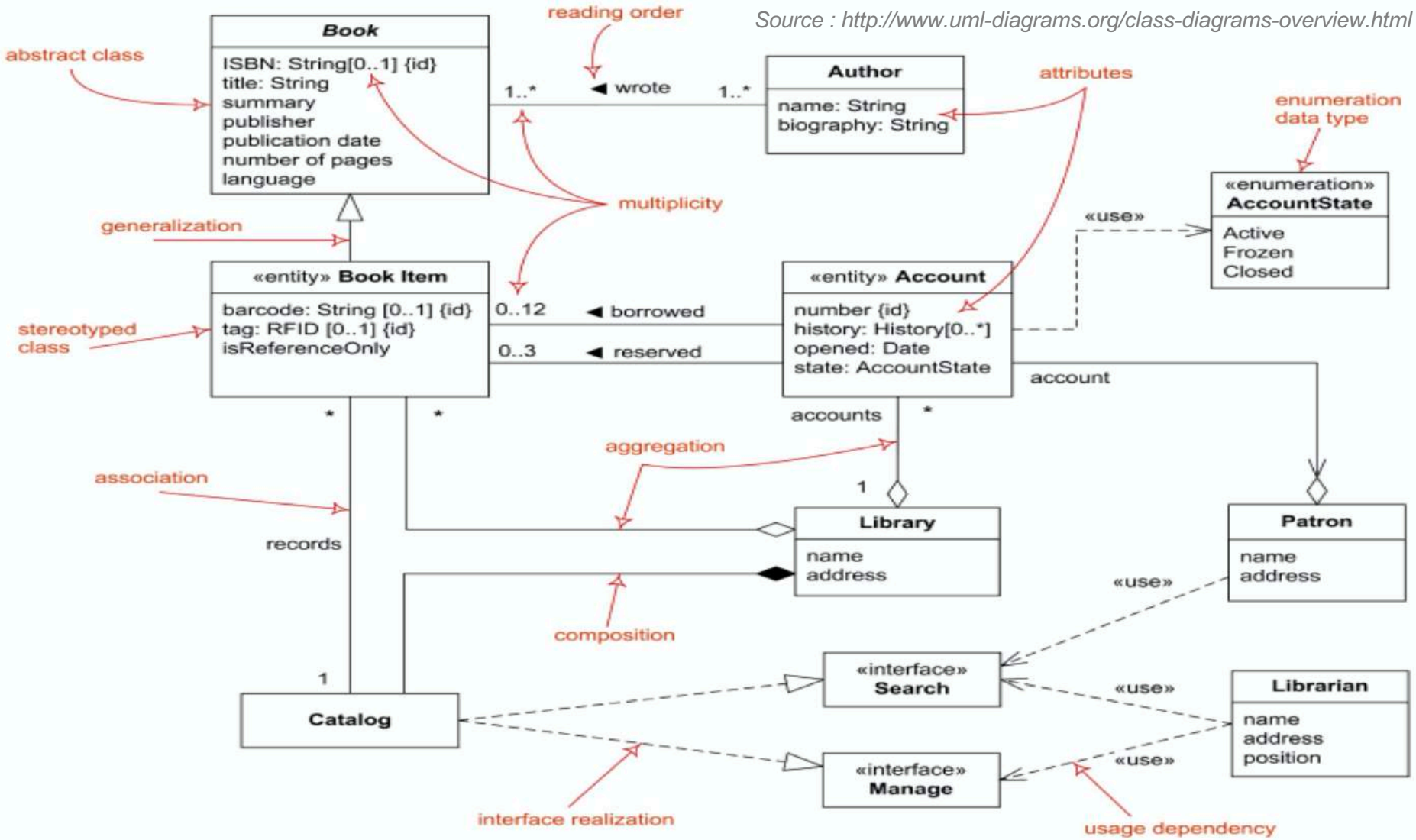
Première étape → Seconde étape

3.1. Diagramme de classes

Domain Model Diagram

UML

Source : http://www.uml-diagrams.org/class-diagrams-overview.html



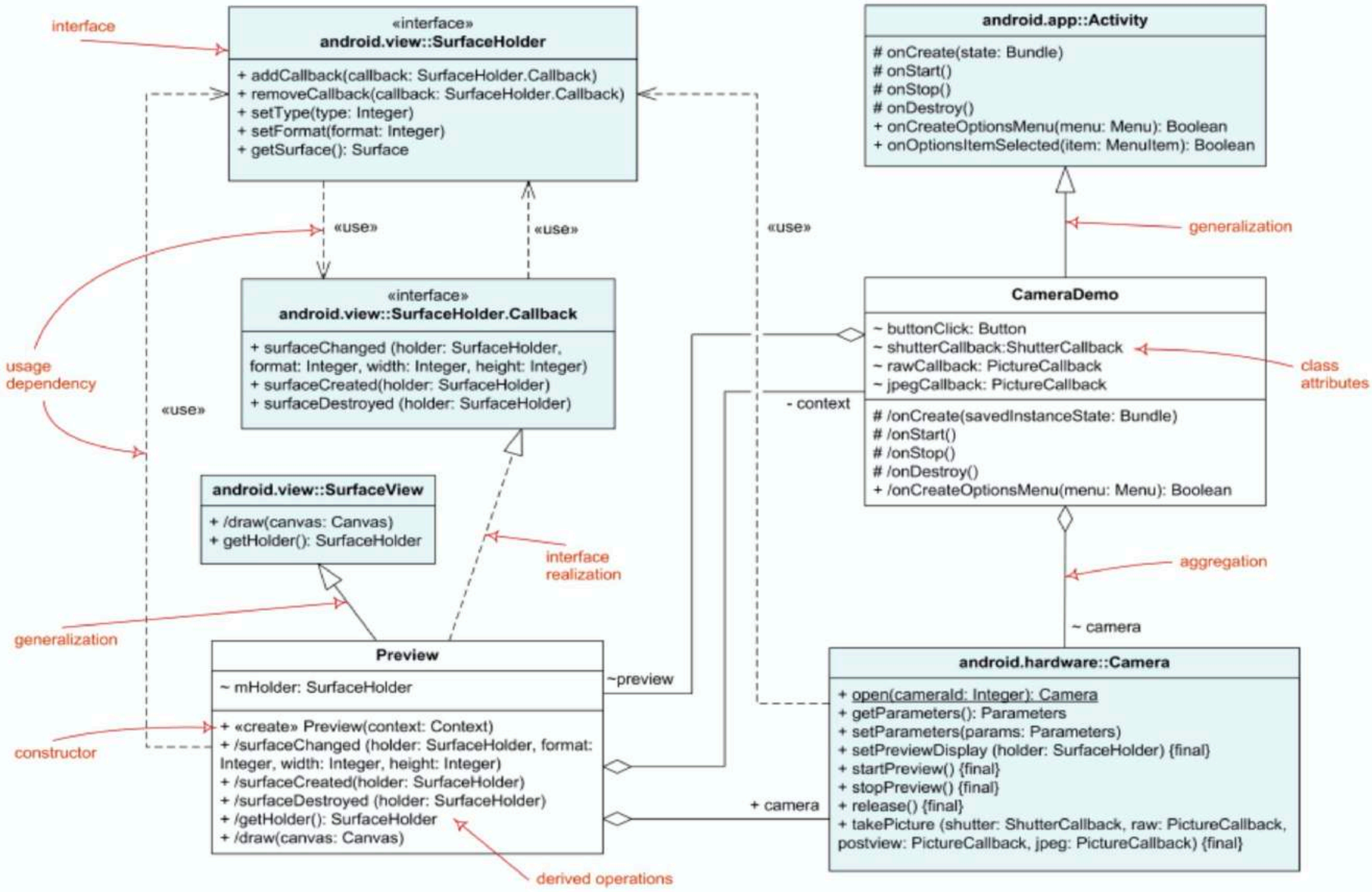
Domain diagram overview - classes, interfaces, associations, usage, realization, multiplicity.

3.1. Diagramme de classes

Diagram of Implementation Classes

UML

Source : http://www.uml-diagrams.org/class-diagrams-overview.html



3.1Diagramme d'objets / Object Diagramme

UML

■ Les diagramme Objet (ou d'instances)

- ✓ représente les objets et leurs liens sans les envois de messages
- ✓ permet la compréhension générale du système
- ✓ facilite la compréhension des structures complexes (récursives)

Source : http://www.uml-diagrams.org/class-diagrams-overview.html

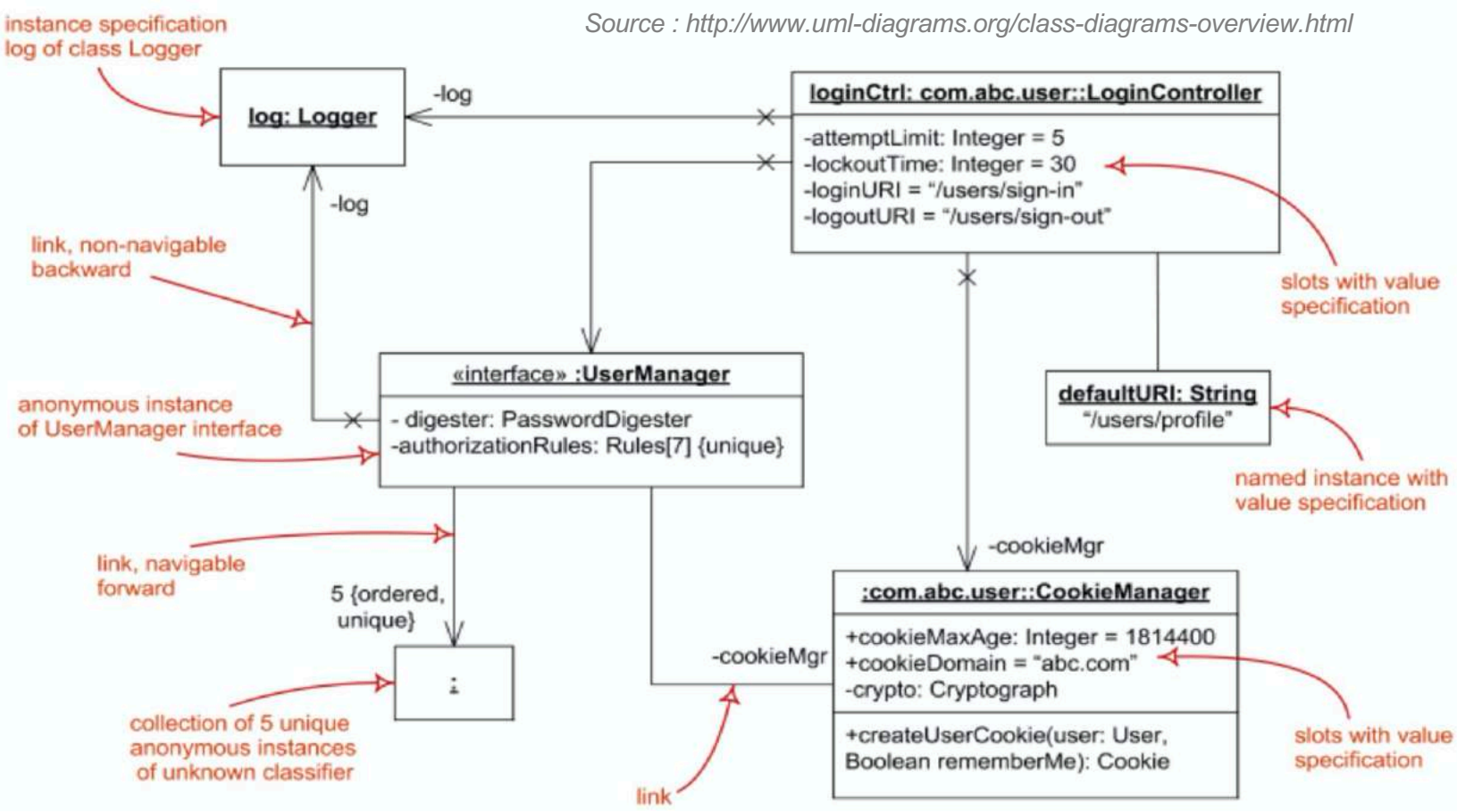
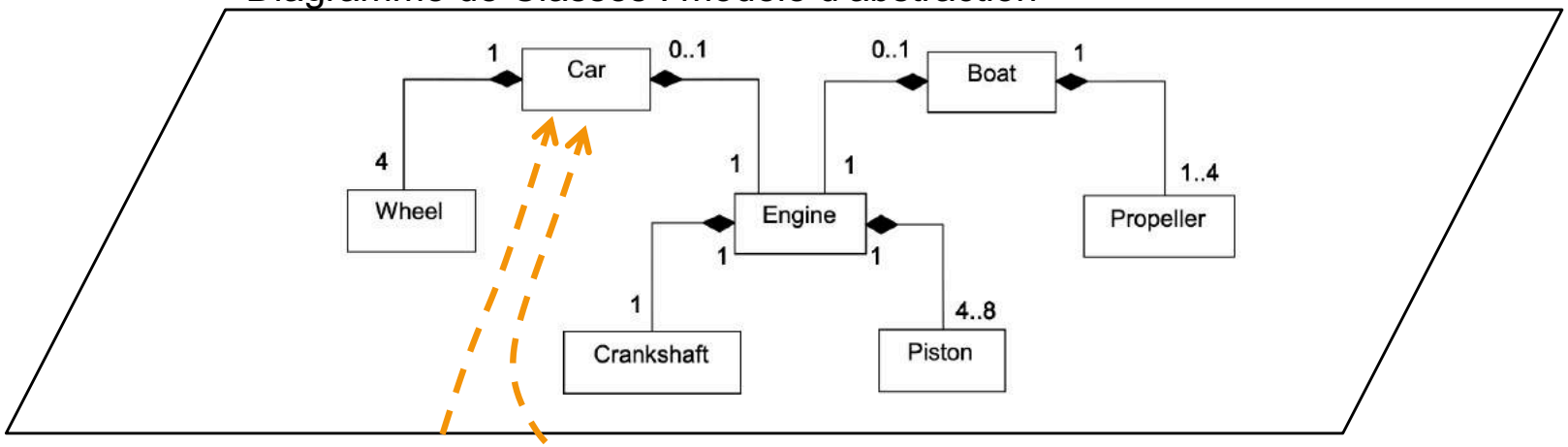


Diagramme de Classes vs d'Objets

UML

Diagramme de Classes : modèle d'abstraction



<< instance of >>

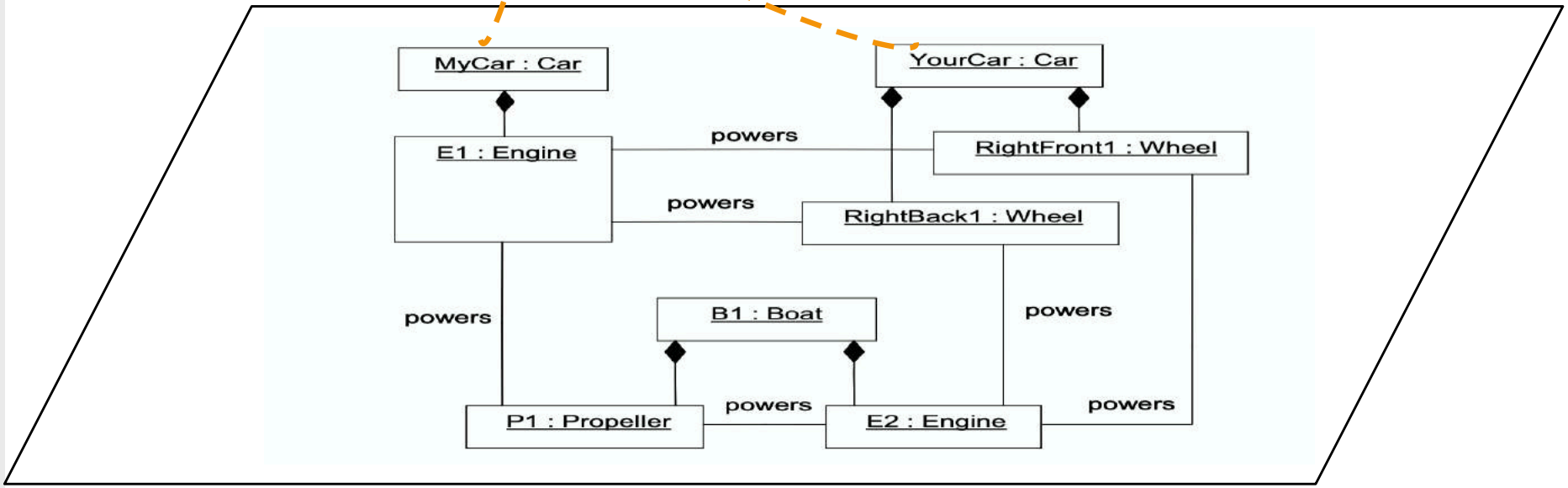


Diagramme Objets : modèle illustrant une situation concrète

Conseils pour la conception de Diagrammes UML

UML

- *Lorsque l'on passe à la conceptualisation d'une application réelle, on garde souvent les mauvaises habitudes datant de diagrammes plus simples.*

- *Voici quelques règles pour réaliser des diagrammes clairs et lisibles par tous.*

  - ✓ **Placer avant de relier**
  - ✓ **Ne pas croiser les associations**
  - ✓ **Mettre des codes couleurs**
  - ✓ **Diviser pour mieux régner**
  - ✓ **Mettre du sens**

3.2 Les Diagrammes de Cas d'Utilisation

UML

Définition

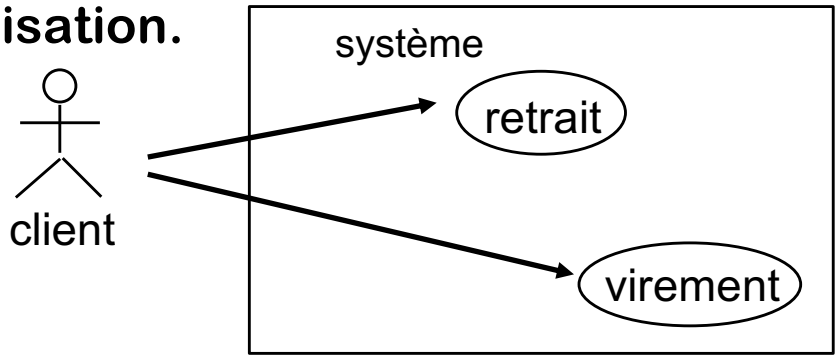
- ✓ Description sous forme d'actions et de réactions du comportement d'un système du point de vue de l'utilisateur

Objectif

- ✓ Définir les limites du système à concevoir
- ✓ Définir les relations entre le système et l'environnement
- ✓ Partitionner l'ensemble des besoins d'un système

Notation UML

- ✓ Un utilisateur ou acteur est représenté par un petit personnage, un cas d'utilisation ou fonctionnalité du système par un ovale, les flèches issues d'un personnage pointent vers les cas d'utilisation.



Les acteurs

UML

Quatre catégories d'acteurs

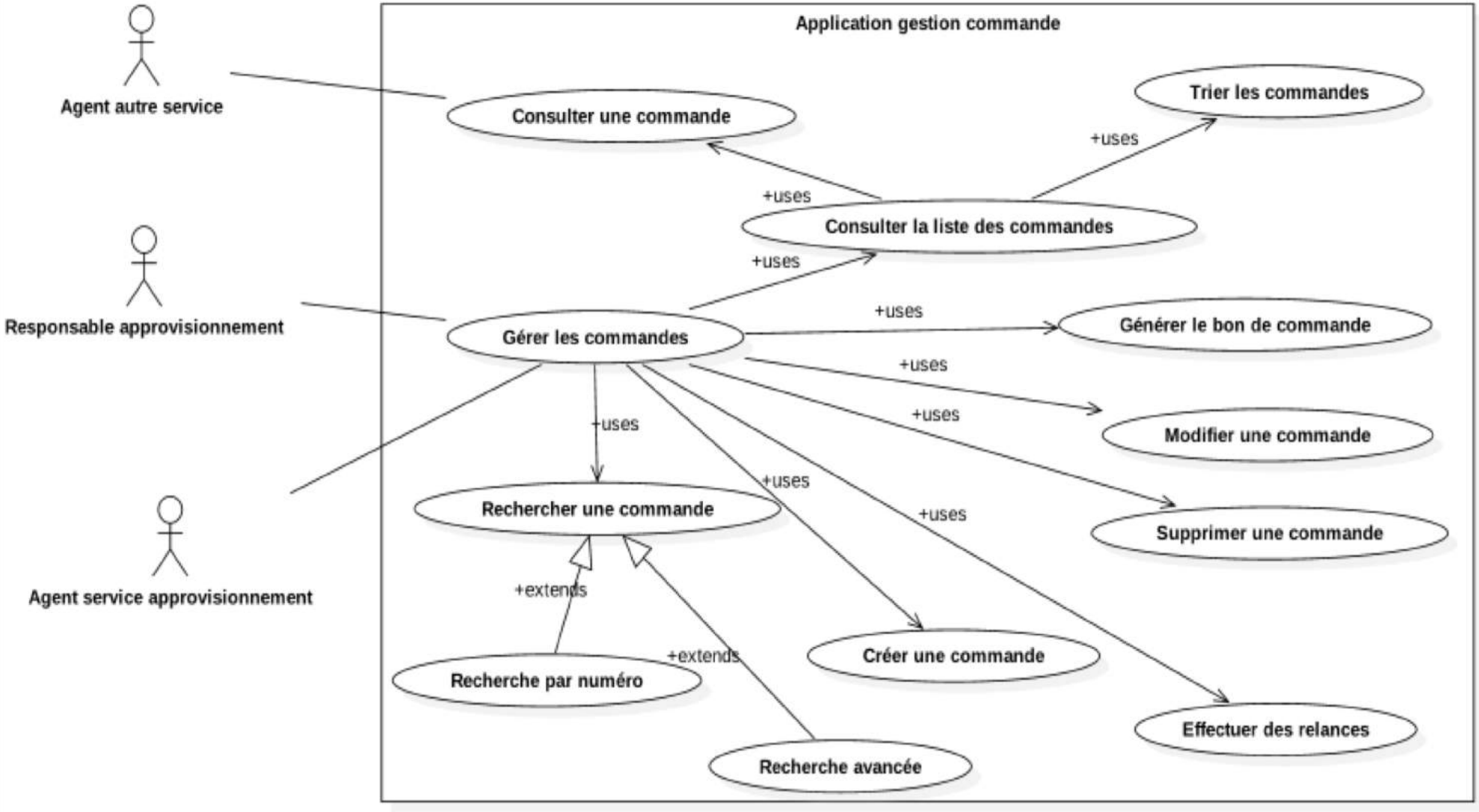
- ✓ acteurs principaux : utilisent les fonctions principales du système
- ✓ acteurs secondaires : tâches administratives ou de maintenance
- ✓ matériel externe : dispositifs matériels incontournables utilisés
- ✓ autres systèmes : systèmes avec lesquels le système doit interagir

Les 3 types de relations avec les acteurs

- ✓ relation de communication : **Déclenche**
  Déclenchement d'un cas d'utilisation par un un acteur
- ✓ relation d'utilisation : **Utilise**
  Un cas d'utilisation utilise le comportement complet d'un autre
- ✓ relation d'extension : **Etend**
  Le cas d'utilisation source étend ou enrichi le comportement du cas d'utilisation destination.

Exemple de Diagramme de Cas d'Utilisation

UML



Exemple sur un système de gestion de commandes

Spécification d'un cas d'utilisation

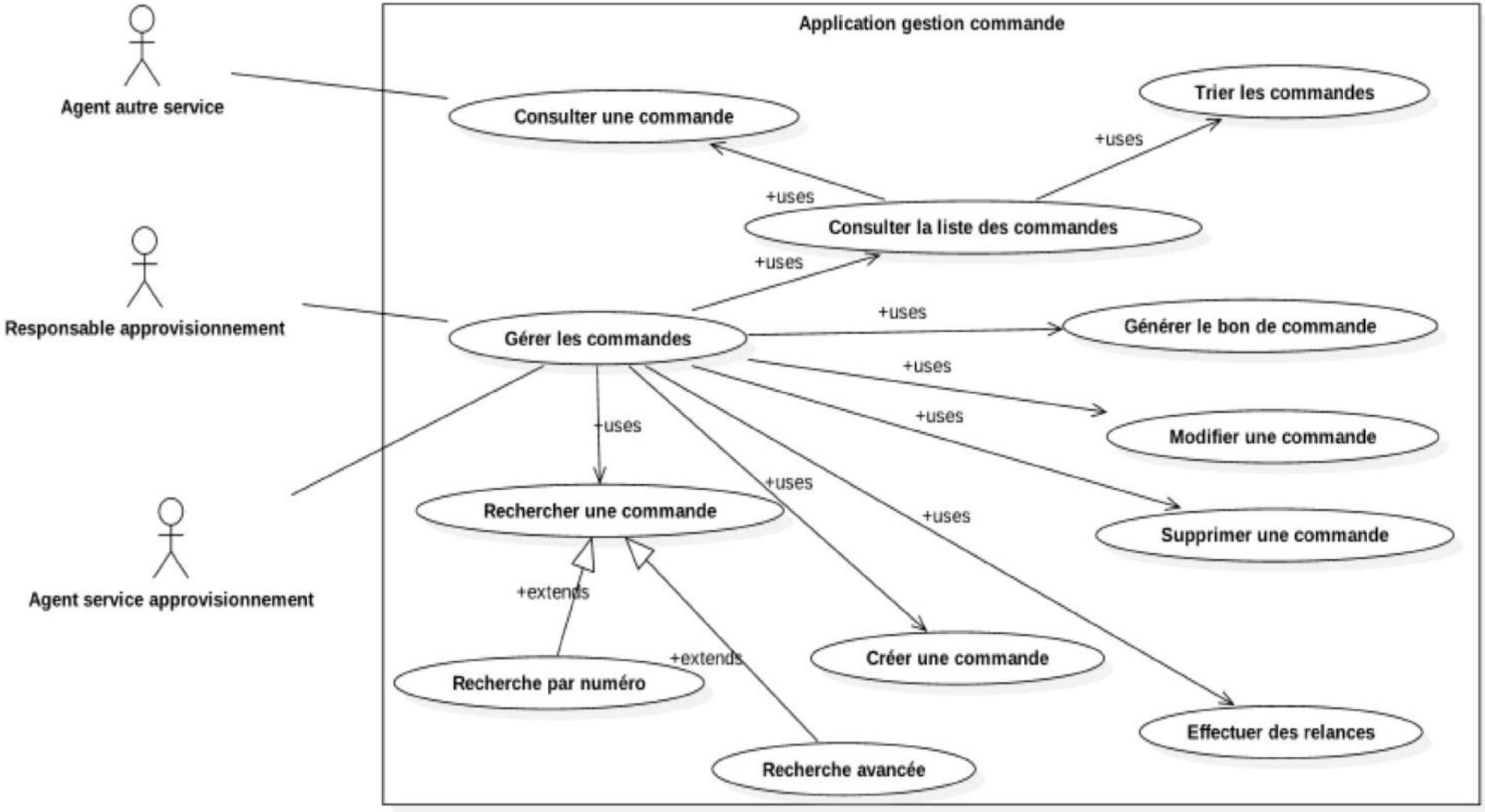
UML

La spécification d'un cas d'utilisation comprend :

- **L'événement qui déclenche le cas d'utilisation**
- **L'événement qui cause l'arrêt du cas d'utilisation**
- **L'interaction entre le cas d'utilisation et les acteurs**
- **Les échanges d'informations**
- **La chronologie et l'origine des informations**
- **Les répétitions de comportements**
- **Les situations optionnelles (Choix a, Choix b, Choix c)**

Exemple de Diagramme de Cas d'Utilisation

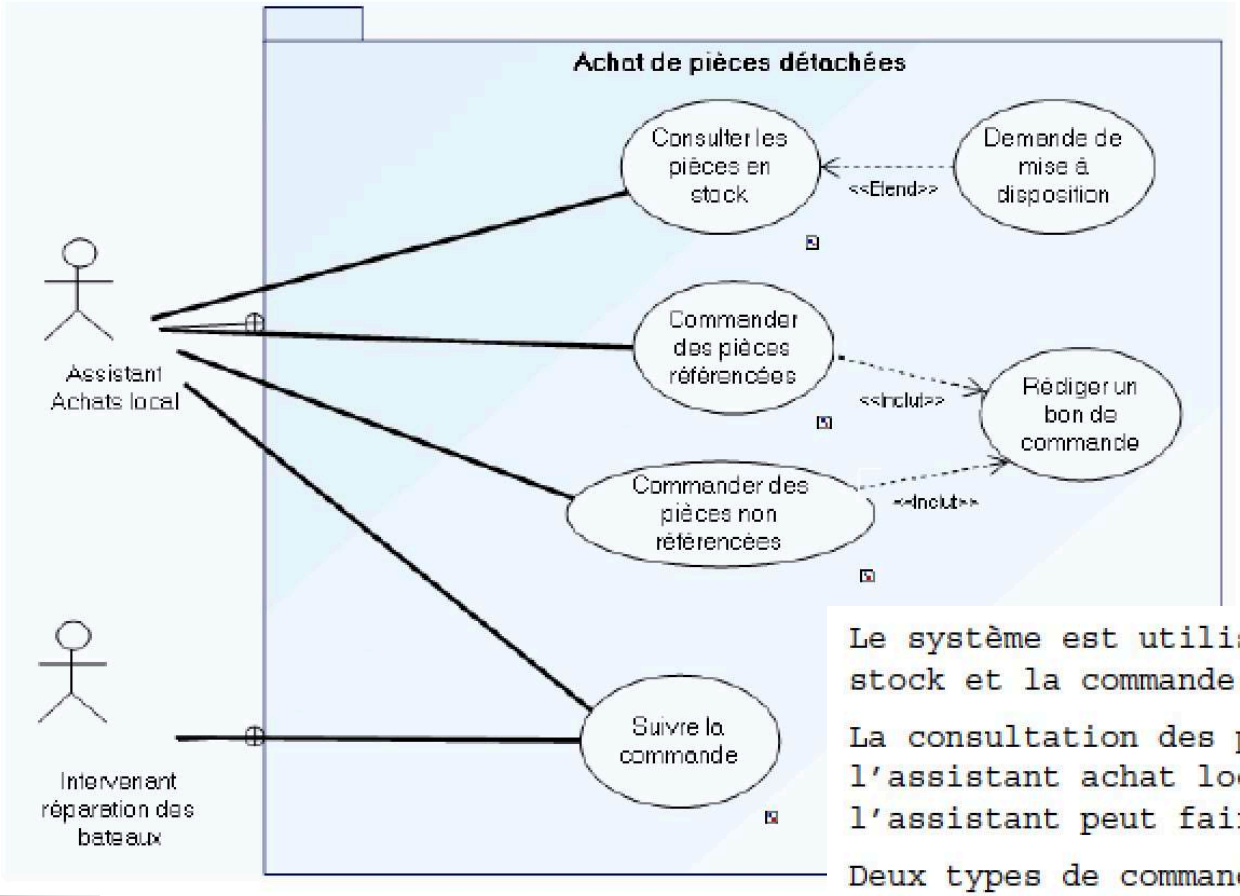
UML



Exemple sur un système de gestion de commandes

Exemple de Diagramme de Cas d'Utilisation

UML



Exemple sur une application d'achats de pièces détachées

Le système est utilisé pour la consultation des pièces en stock et la commande de nouvelles pièces détachées.

La consultation des pièces en stock est effectuée par l'assistant achat local. Suite à la consultation, l'assistant peut faire une demande de mise à disposition.

Deux types de commandes sont possibles; une commande de pièces déjà référencées ou une commande de pièces non référencées. Dans les deux cas, un bon de commande doit être rédigé.

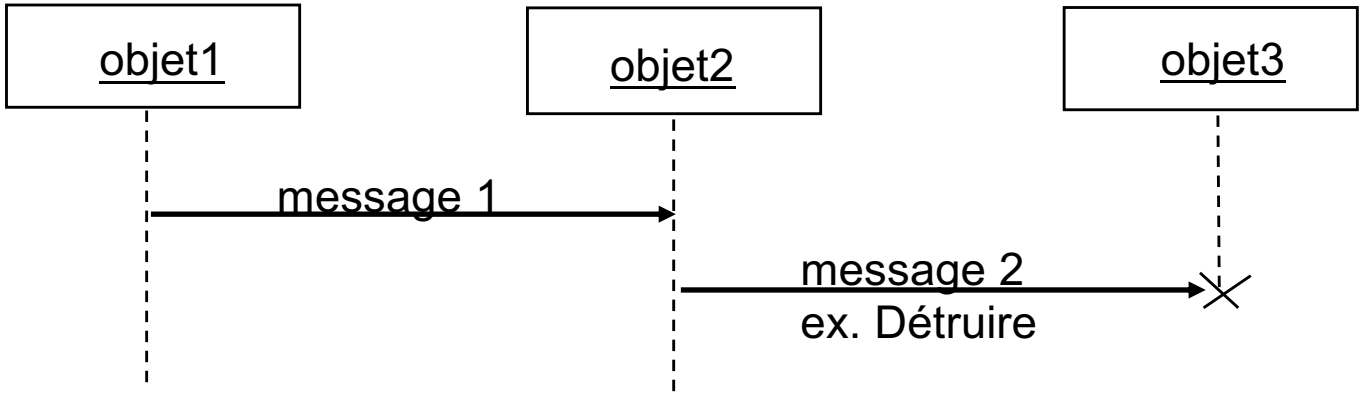
Le suivi de la commande est assuré à la fois par l'assistant achat local et l'intervenant chargé de la réparation des bateaux.

Illustrations : crédits
© MEGA International, 2018

3.3 Les diagrammes de séquence

UML

- Un diagramme de séquence montre des interactions entre objets selon un point de vue temporel
- La représentation se concentre sur l'expression des interactions
- Notation :



envoi synchrone (expéditeur bloqué jusqu'à acceptation du destinataire)

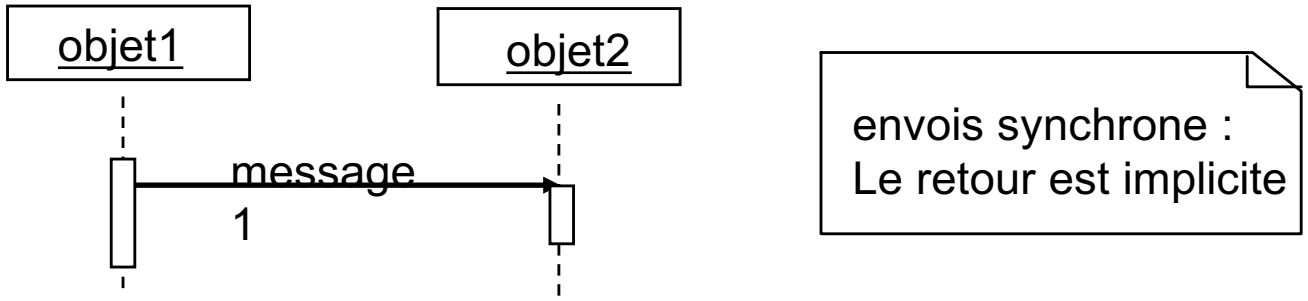
envoi réflexif (envoi d'un objet à soi-même)

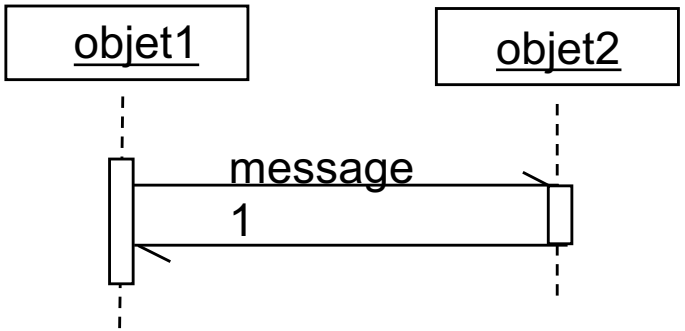
envoi asynchrone (n'interrompt pas l'exécution de l'expéditeur)

Période d'activités

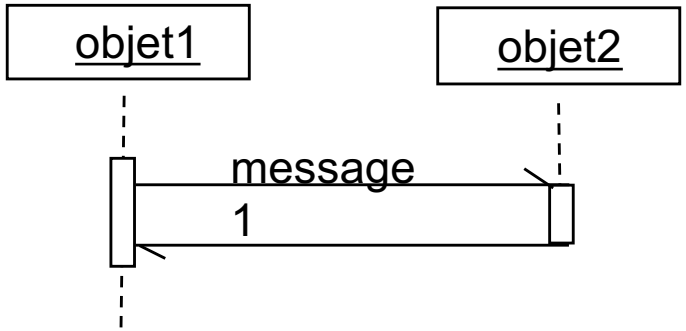
UML

- **Une période d'activité correspond au temps pendant lequel un objet effectue une action**





Retour explicite d'envoi asynchrone



Retour explicite avant suicide

3.4 Les diagrammes d'états-transitions

UML

Un diagramme d'états-transitions est un automate d'états fini déterministe associé à une classe

- **Abstraction des comportements possibles**
  - ✓ chaque objet suit le comportement décrit dans l'automate associé à sa classe, son état caractérise ses conditions dynamiques
- **On associe un tel automate à toute classe qui présente un comportement réactif marqué**
- **Statecharts de D. Harel**
  - ✓ D. Harel, "Statecharts : A Visual Formalisme for Complexe Systems", Science of Computer Programming, vol. 8, 1987.
  - ✓ Automates hiérarchiques, possédant les concepts d'orthogonalités, d'agrégation et de généralisation

Sémantique du diagramme

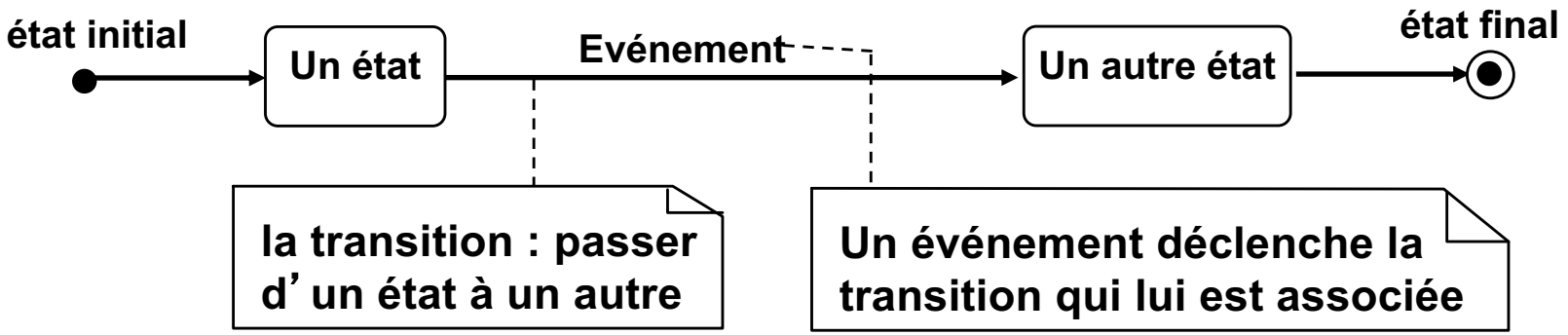
UML

- Ce diagramme sert à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions.
- Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.
- Un état se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des valeurs des attributs d'un objet.
- Une transition représente le passage instantané d'un état vers un autre.
- Une transition est déclenchée par un événement. En d'autres termes : c'est l'arrivée d'un événement qui conditionne la transition.
- Les transitions peuvent aussi être automatiques, lorsqu'on ne spécifie pas l'événement qui la déclenche.
- En plus de spécifier un événement précis, il est aussi possible de conditionner une transition, à l'aide de "gardes" : il s'agit d'expressions booléennes, exprimées en langage naturel (et encadrées de crochets).

Notations

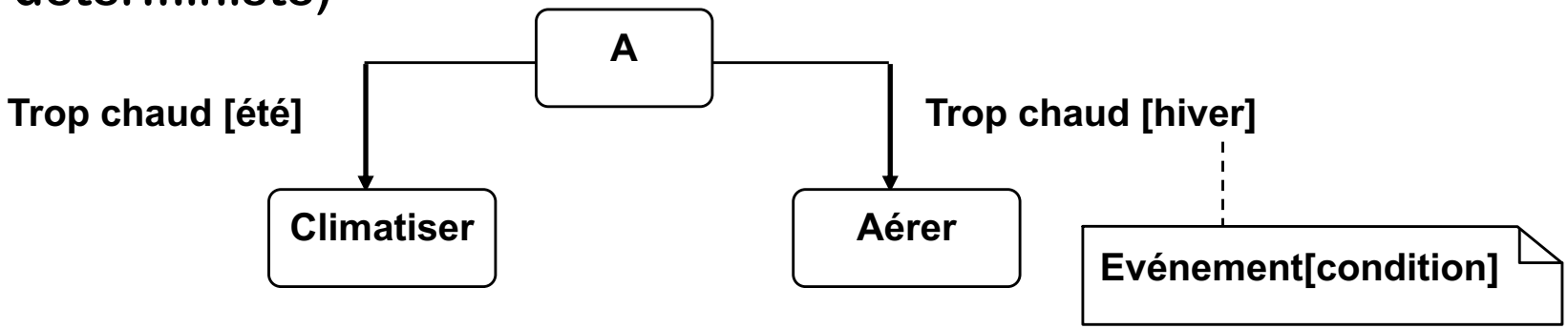
UML

Etat, Transition et Evénement



Les gardes

- ✓ Une garde est une condition qui valide ou non le déclenchement d'une transition lors de l'occurence d'un événement
- ✓ Elles doivent être mutuellement exclusives (aspect déterministe)



Evénements & Opérations, Actions

UML

- **Le lien entre Evénements et Opération du diagramme de classe apparaît dans l'automate**
  - ✓ Chaque transition peut être décorée par le nom d'une action à exécuter lorsque la transaction est déclenchée par un événement
  - ✓ L'action correspond à une des opérations déclarées dans la classe de l'objet destinataire de l'événement
- **Les états peuvent contenir des actions**
  - ✓ exécutées à l'entrée ou à la sortie
  - ✓ exécutées lors de l'occurrence d'un événement pendant que l'objet est dans l'état

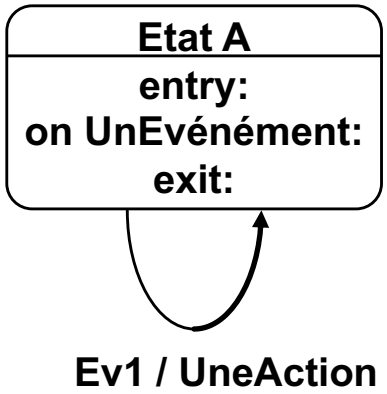
Les mot-clés "*entry:*" et "*exit:*"

- indique les actions d'entrée et de sortie

Le mot-clé "*on*" *Un événement:*

- indique une action sur événement externe

Une transition réflexive entraîne les actions de sortie et d'entrée



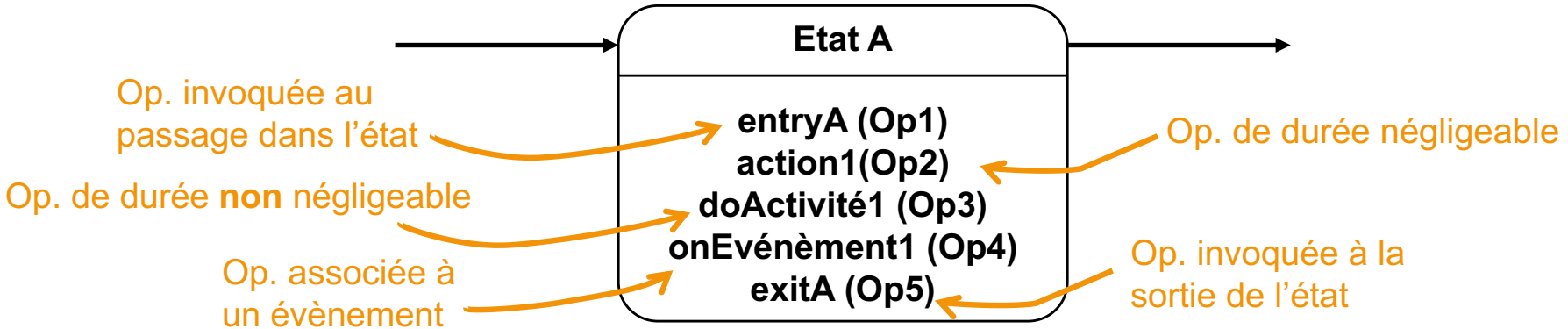
Actions et activités

UML

Définitions

- ✓ Une action correspond à une opération dont le temps d'exécution est négligeable
- ✓ Une activité est une opération qui prend du temps qui est exécutée pendant qu'un objet est dans un état donné.
- ✓ Le mot-clé "*do:*" indique une activité
- ✓ Activité cyclique : s'arrête lorsqu'une transition de sortie est déclenchée. activité séquentielle : l'état peut être quitté si l'activité parvient à son terme

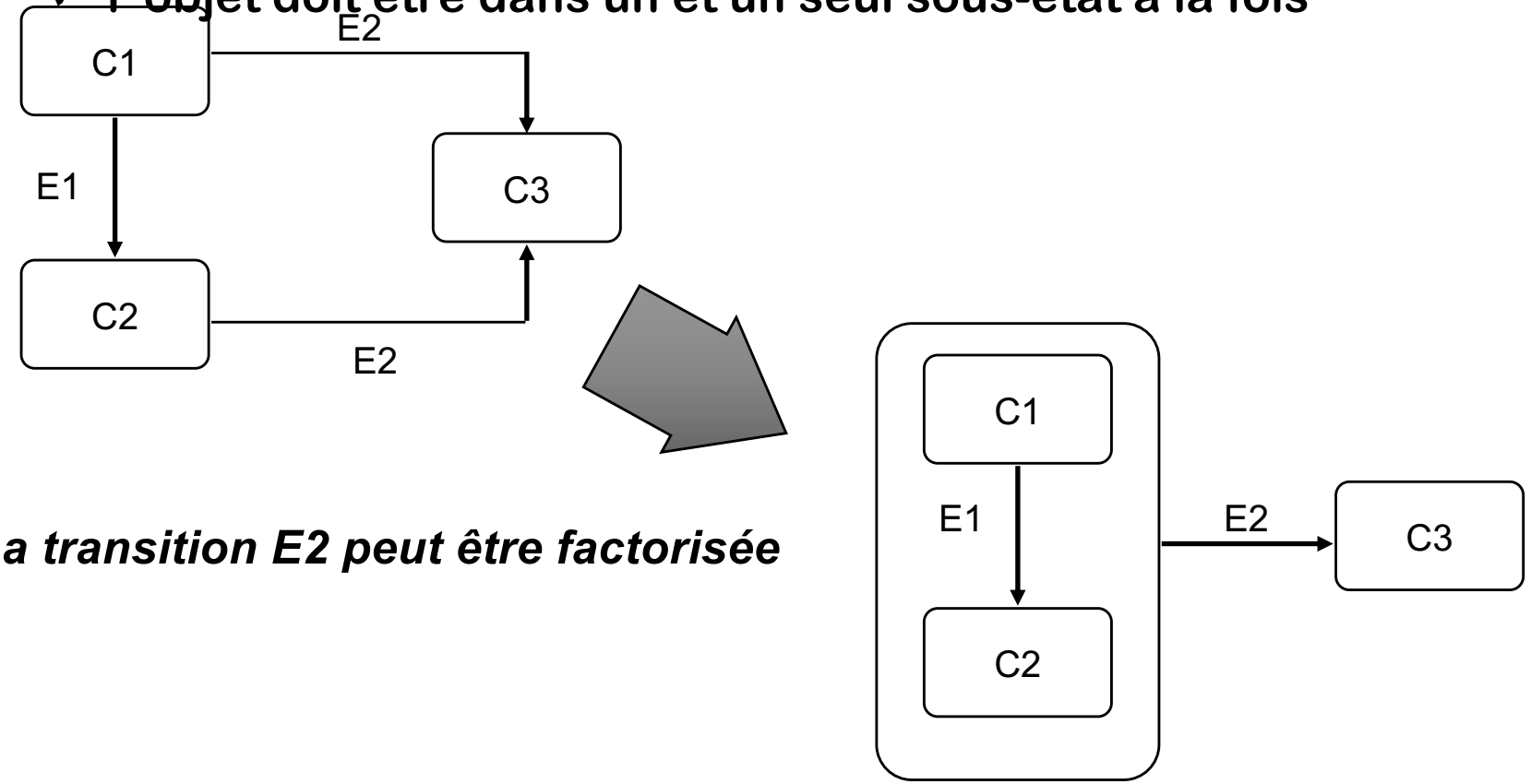
Types d'Actions/Activités



Notion avancée : Généralisation d'état

UML

- Un état peut être décomposé en sous-états disjoints
  - ✓ états généraux = super-états, états spécialisés = sous-états
  - ✓ l'objet doit être dans un et un seul sous-état à la fois

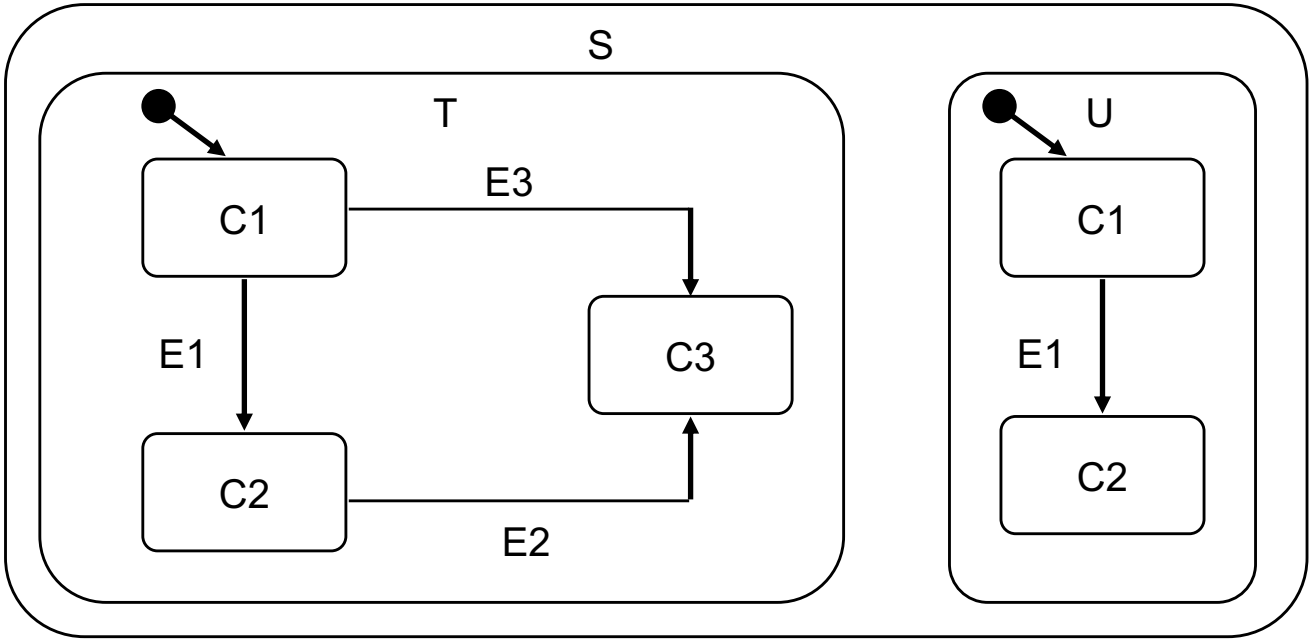


La transition E2 peut être factorisée

Notion avancée : Agrégation d'état

UML

- **Composition d'un état à partir de plusieurs autres états indépendants**
  - ✓ on parle d'agrégation d'états et d'état composant
  - ✓ l'objet doit être simultanément dans tous les états composant l'agrégation d'états



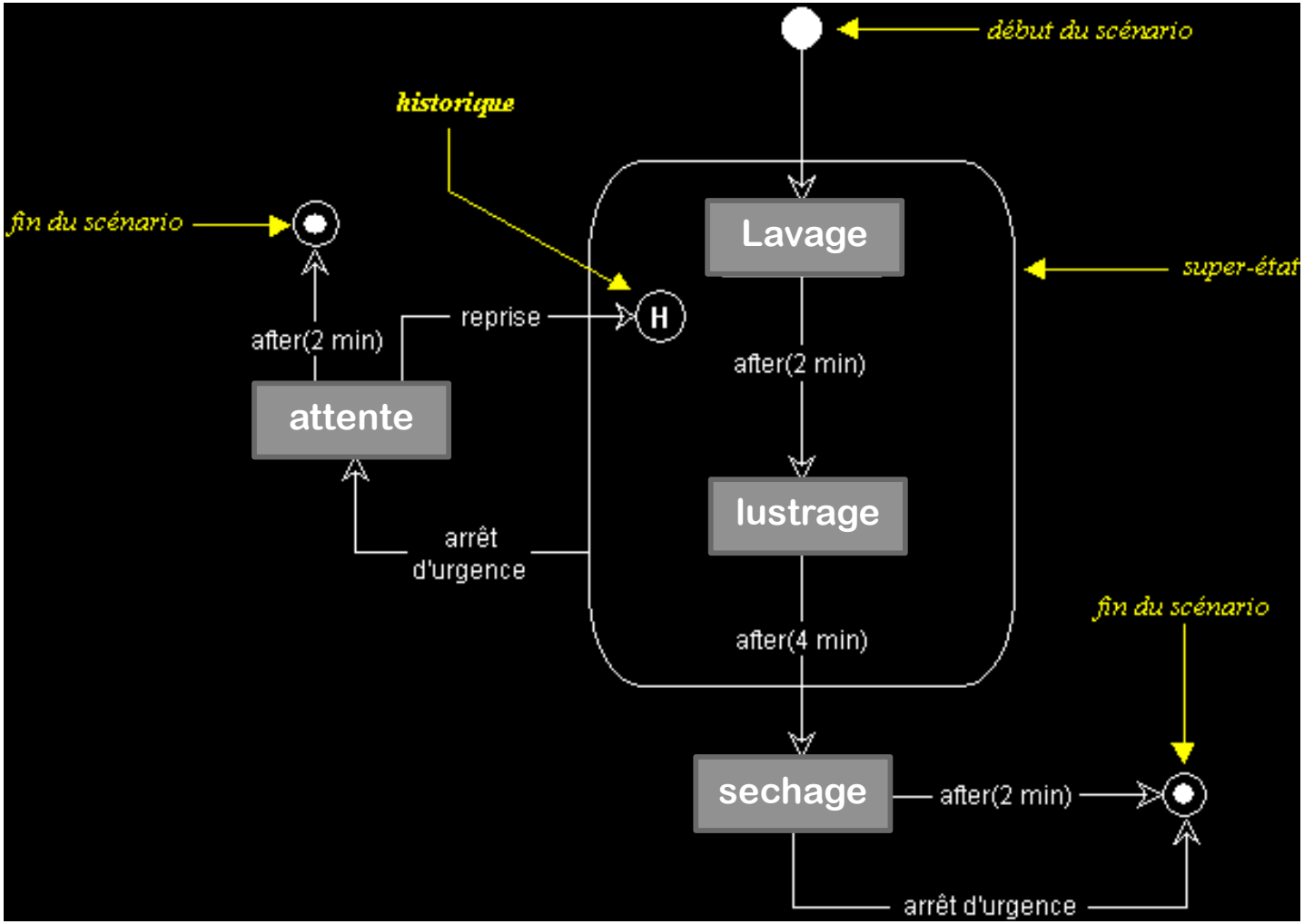
L'état S - produit cartésien des états T et U

Les diagrammes d'états-transitions : exemple

UML

Le diagramme d'états-transitions présenté, montre les différents états par lesquels passe une machine à laver les voitures.

En phase de lustrage ou de lavage, le client peut appuyer sur le bouton d'arrêt d'urgence. S'il appuie sur ce bouton, la machine se met en attente. Il a alors deux minutes pour reprendre le lavage ou le lustrage (la machine continue en phase de lavage ou de lustrage, suivant l'état dans lequel elle a été interrompue), sans quoi la machine s'arrête. En phase de séchage, le client peut aussi interrompre la machine. Mais dans ce cas, la machine s'arrêtera définitivement (avant de reprendre un autre cycle entier).



3.5 Les Diagrammes d'activités

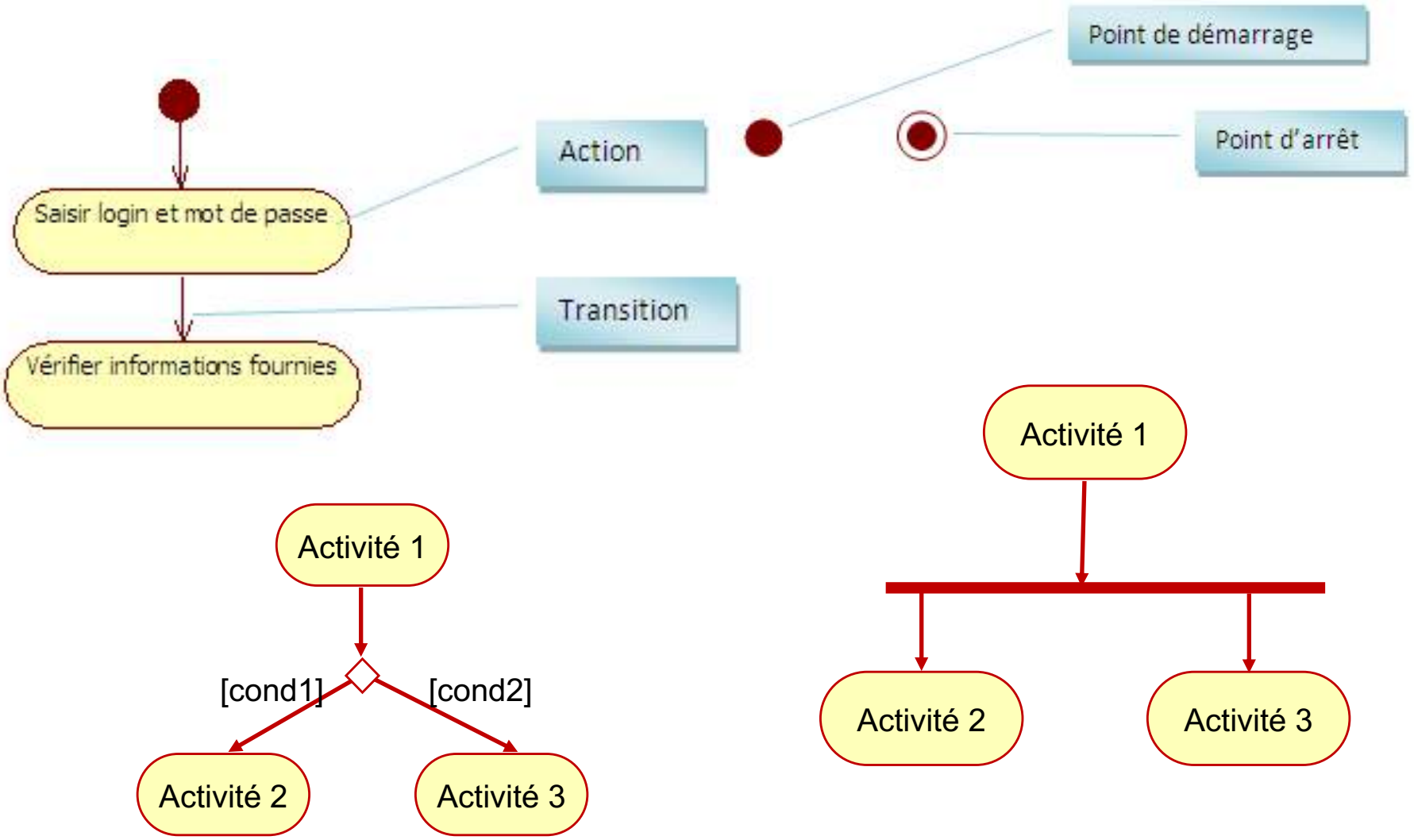
UML

- **Un diagramme d'activité expose les activités séquentielles et parallèles d'un cas d'utilisation**
  - ✓ Un diagramme d'activité est une formalisation graphique des actions qui sont réalisées dans un cas d'utilisation.
  - ✓ Le diagramme est donc organisé en actions réalisées soit par un acteur, soit par le système, relié par une flèche indiquant l'enchaînement des actions.
  - ✓ C'est un complément à la fiche descriptive d'un cas d'utilisation complexe, permet de donner une vision globale de l'ensemble des scénarios possibles.

- **Ils sont utiliser pour représenter :**
  - ✓ Le déroulement du cas d'utilisation
  - ✓ Des enchainements d'activités regroupées séquentiellement
  - ✓ les synchronisations d'activités (workflows)

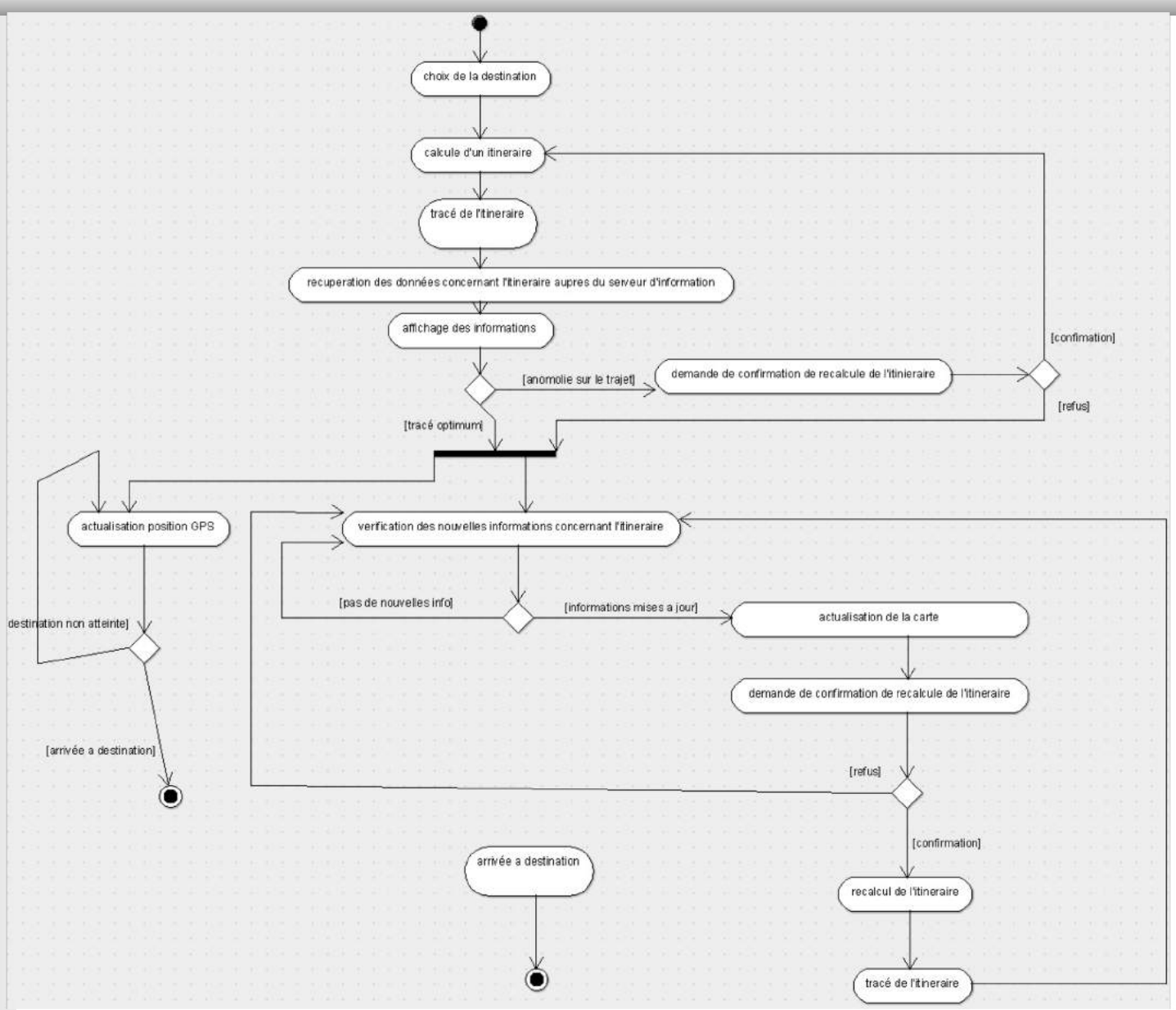
Diagramme d'activité : Notations

UML



Exemple de Diagramme d'activité

UML



Système de navigation GPS

3.6 Composants et Les diagrammes de composants

UML

- **Classe** : bonne réutilisation au sein d'une application via héritage et encapsulation

Cependant…

- faible granularité qui ne correspondant pas au facteur d'échelle de réutilisabilité inter-application
- associations entre classes qui constituent des connexions figées: avec les autres classes matérialisant des liens structurels fixes

La Classe ne constitue pas une réponse adaptée à la problématique de la réutilisation.

Approche Composant

Met l'accent sur la réutilisation du composant et l'indépendance de son évolution vis-à-vis des applications qui l'utilisent.

Application ~~monolithique~~ vs. Approche composants

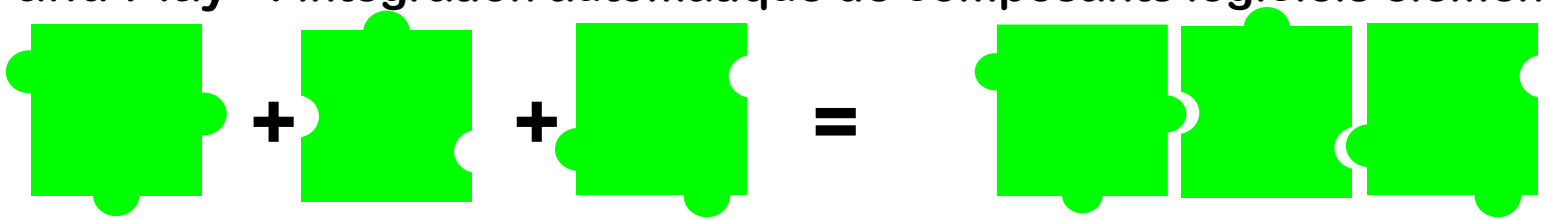
UML

- **Programmer = assembler**
  - ✓ informatique = assemblage de composants par des langages simples : scripts ou langage graphique
- **Petits développeurs**
  - ✓ possibilités à de nombreux développeurs de distribués des briques s'intégrant dans de gros logiciels
- **Grands développements**
  - ✓ réduction de la complexité des grands logiciels
  - ✓ réduction des tests de non régression en cas d' évolution Vx à Vx+1
  - ✓ proposition de solutions sur mesure par assemblage de composants au lieu de proposer des monstres invendables

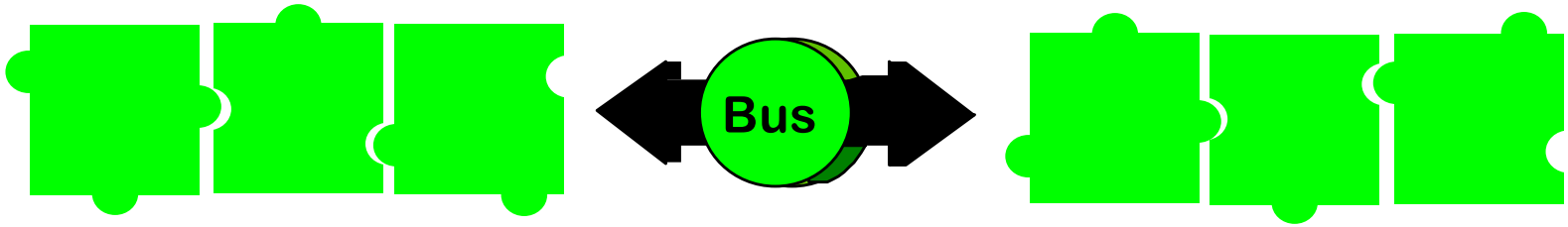
Vers les architectures de composants distribuées (1)

UML

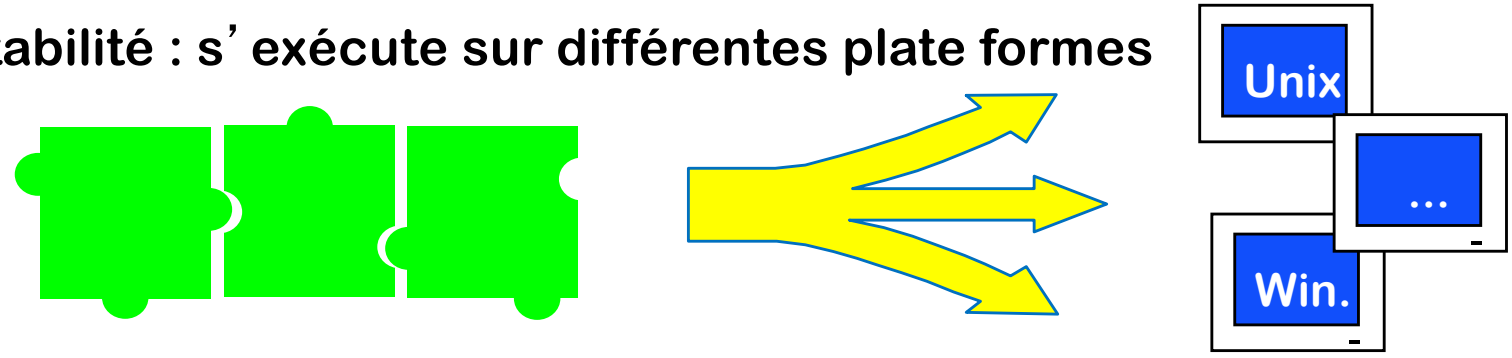
"Plug and Play" : intégration automatique de composants logiciels élémentaires



Interopérabilité : messagerie transparente n'importe ou dans le monde



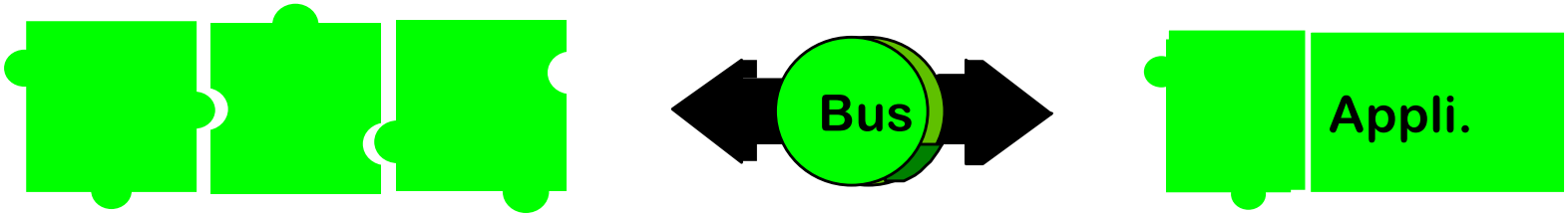
Portabilité : s'exécute sur différentes plate formes



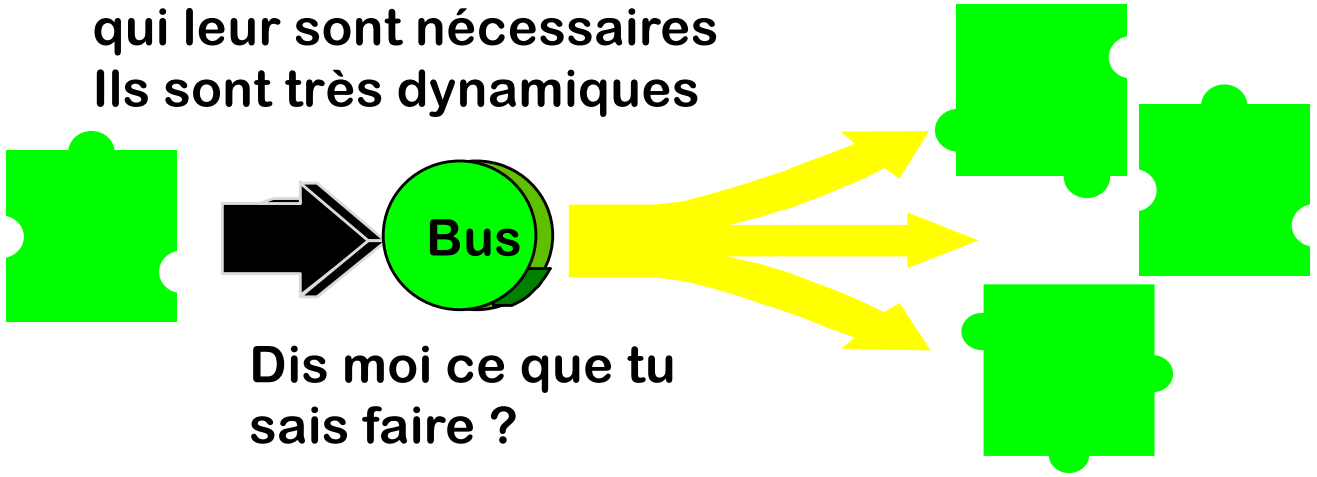
Objets Distribués & Composants (3)

UML

Coexistence : coexiste avec des applications traditionnelles avec des interfaces objet



Autonomie : gèrent leurs liens avec les autres objets et les ressources qui leur sont nécessaires
Ils sont très dynamiques



De la Classe au Composant en conception objet

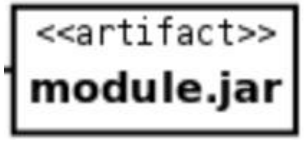
UML

- entité logicielle au niveau d'échelle de réutilisation supérieure à la classe mais dont le comportement interne est généralement réalisé par un ensemble de classes.
- ne correspond pas à une application complète mais peut comporter sa propre version et/ou sa propre licence
- répond à une fonction précise qu'il réalise dans une conception génériques (par opposition à spécialisées) puisque sa vocation est d'être réutilisable.
- doit être une entité logicielle substituable par un composant respectant les mêmes interfaces
- boite noire logicielle autonome avec des interfaces de communication. Le fonctionnement interne est totalement encapsulé (ou masqué) : seules ses interfaces sont visibles.

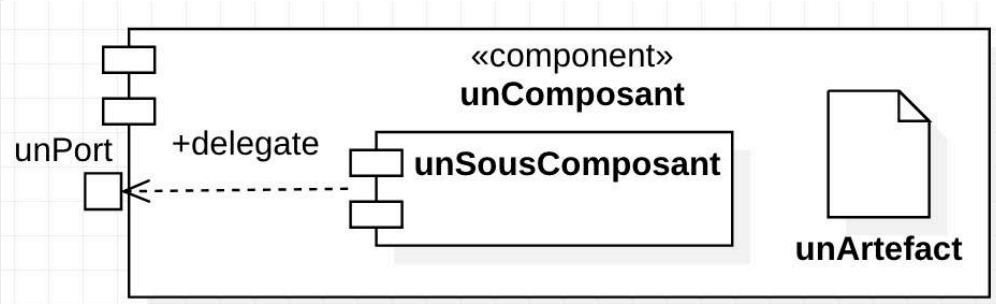
Diagramme de composant UML

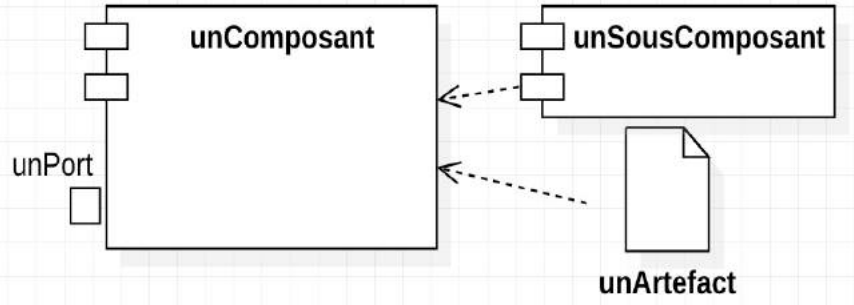
UML

- **Un composant peut être réalisé par :**
  - ✓ une partie masquée : ensemble de classes
  - ✓ une partie visible : ensembles d'interfaces
  - ✓ d'autres composants et des artefacts.
- **L'artefact**
  - ✓ élément concret existant dans le monde réel
  - ✓ exécutable, fichier, tables de bases de données, …
  - ✓ notation UML : stéréotype « artifact » suivi de son nom



- **Représentation UML**
  - ✓ representé par un classeur, stéréotype optionnel <<component>>
  - ✓ ses constituants peuvent être placés à l'intérieur du classsur ou être placés à côté en les reliant au composant par une dépendance



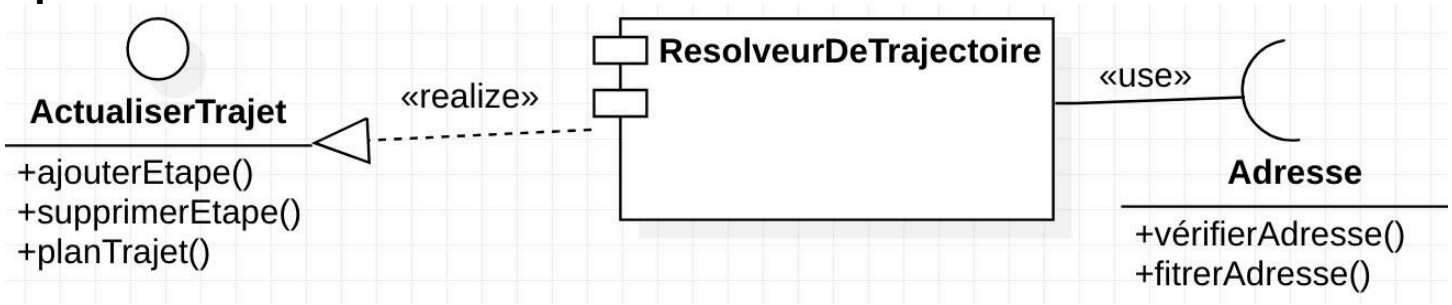


Interface et point de connexion d'un composant

UML

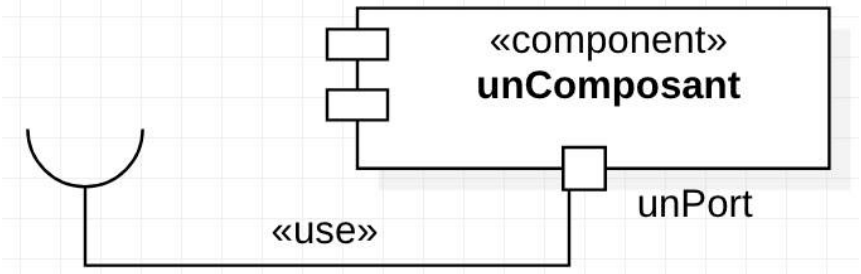
■ Interface :

- ✓ Un composant interagit avec d'autres composants au travers d'interfaces de communication offertes « realize » ou requises « use ».
- ✓ Notation UML: l'interface offerte est représentée par un cercle, l'interface requise par un demi-cercle.



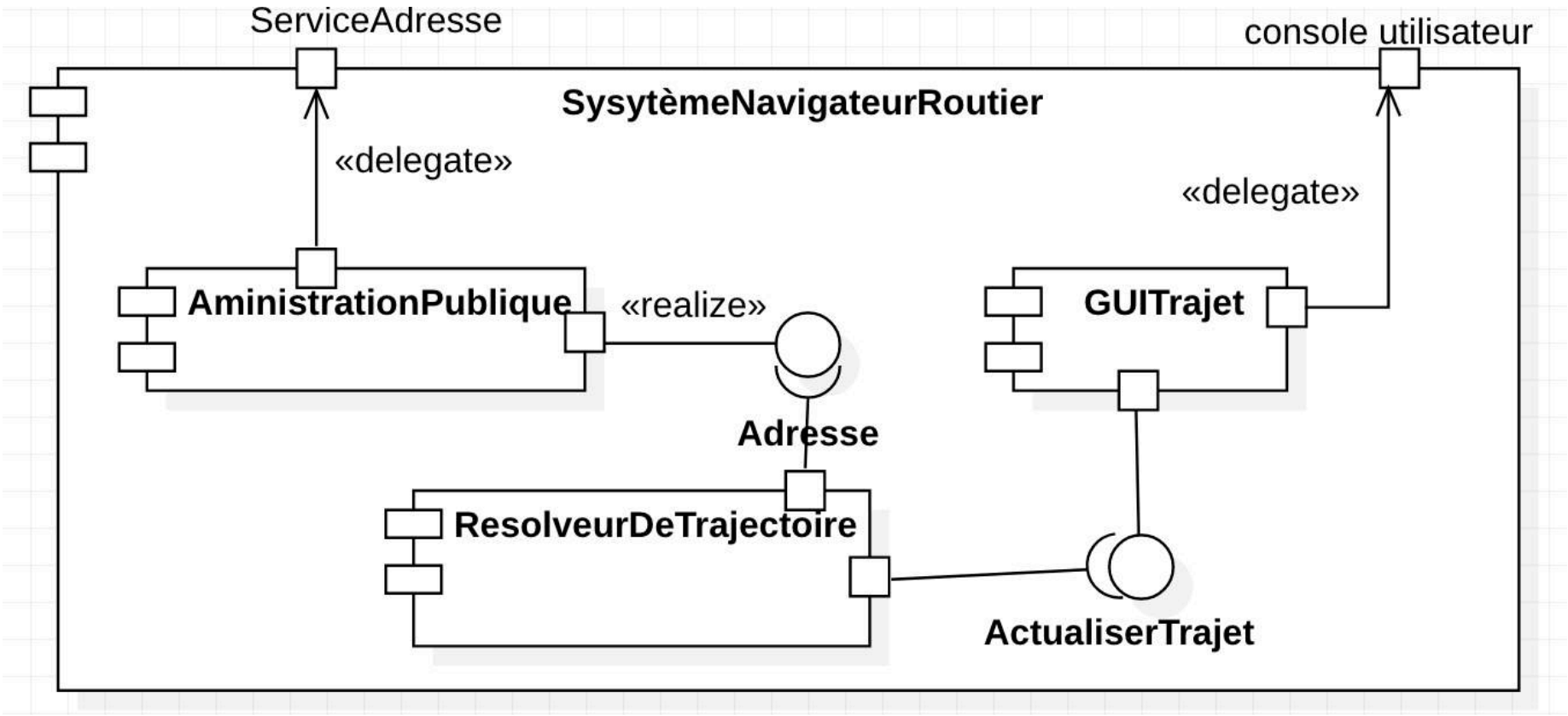
■ Port :

- ✓ point de connexion entre le composant et l'interface
- ✓ l'utilisation des ports permet de modifier la structure interne d'un composant sans affecter les clients externes.
- ✓ un port est représenté par un petit carré sur la bordure du composant



Interface et point de connexion d'un composant

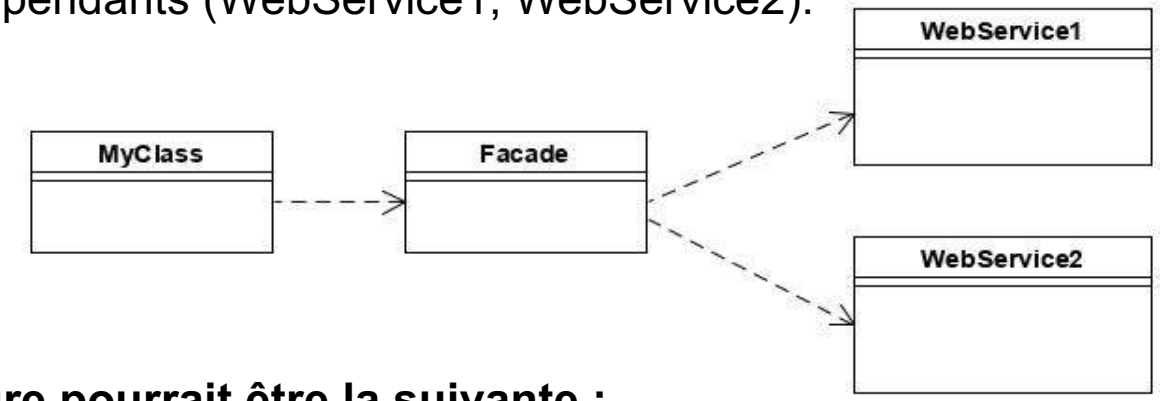
UML



Interface et point de connexion d'un composant

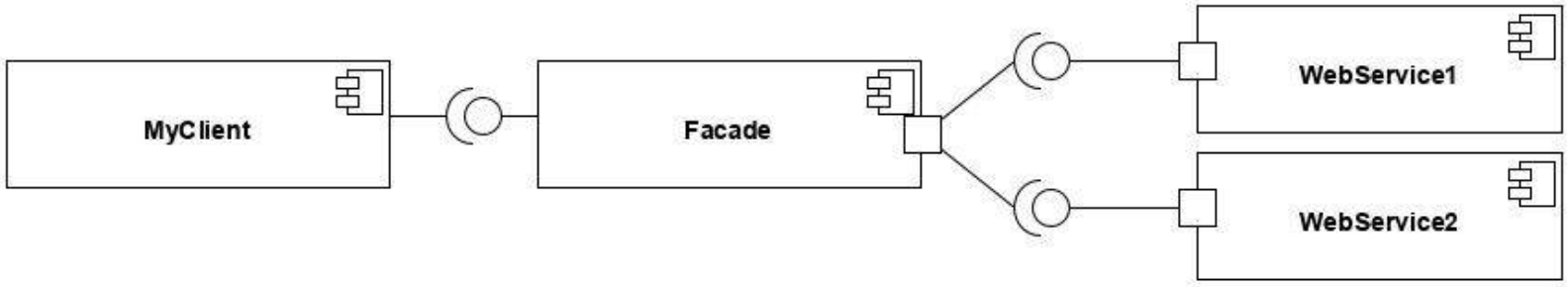
UML

Soit une architecture d'application assez classique avec une classe ou un composant rassemblant des classes applicatives (MyClass) de votre système connecté à une classe ou composant de façade (Facade) établissant le lien avec des web services totalement indépendants (WebService1, WebService2).



La proposition d'architecture pourrait être la suivante :

- Les ports sur le composant de façade et les web services assurent que toute modification interne des composants indépendants du système n'aura aucune répercussion sur les composants en connexion.
- Le composant applicatif (MyClient) et le frontal (Facade) sont des composants applicatifs pas totalement indépendants et ne nécessite pas l'usage de ports.



Le diagramme de déploiement : la notion de noeud

UML

Le diagramme de déploiement spécifie :

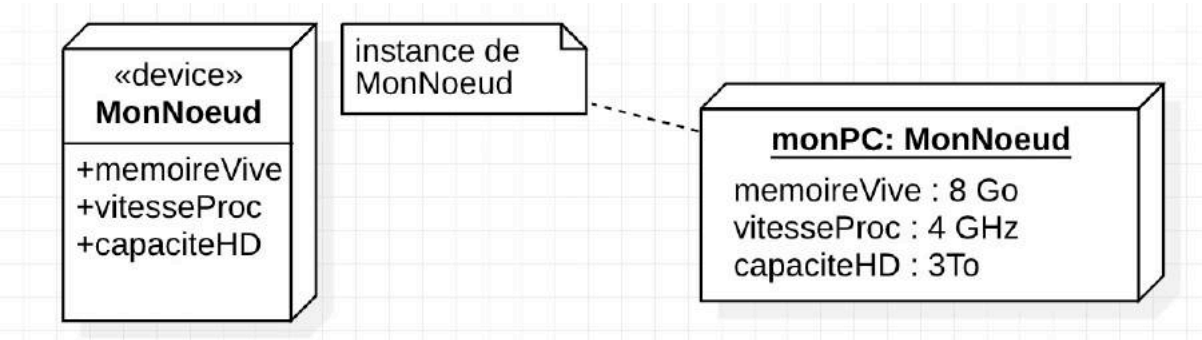
- ✓ la composition physique des matériels constituant le système
- ✓ les chemins de communication entre les composants
- ✓ la répartition des composants sur ces matériels

■ Le nœud

- ✓ Un nœud est une ressource matérielle physique
- ✓ Dispositif (ex. Modem), Processeur (ex. PC), Mémoire (ex. Disque)
- ✓ Il peut posséder des attributs

■ Notation UML

- ✓ Un nœud est représenté par un cube comprenant un nom
- ✓ Dans une instance de nœud, le nom est soulignée

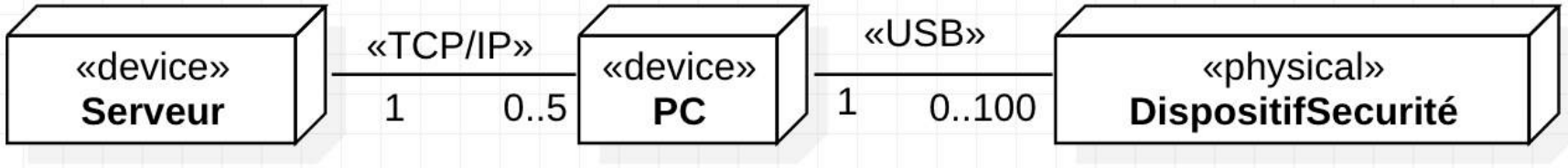


Le diagramme de déploiement : intercations

UML

■ Chemins de communication entre les nœuds

- ✓ association qui symbolisent la nature des connexions entre des nœuds (ex. TCP/IP,HTTP, websocket,...)



■ Affectation d'un composant à un nœud

Deux possibilités en UML :

- ✓ placer le composant dans le nœud,
- ✓ relier par une relation de dépendance stéréotypée « support » orientée du composant vers le nœud

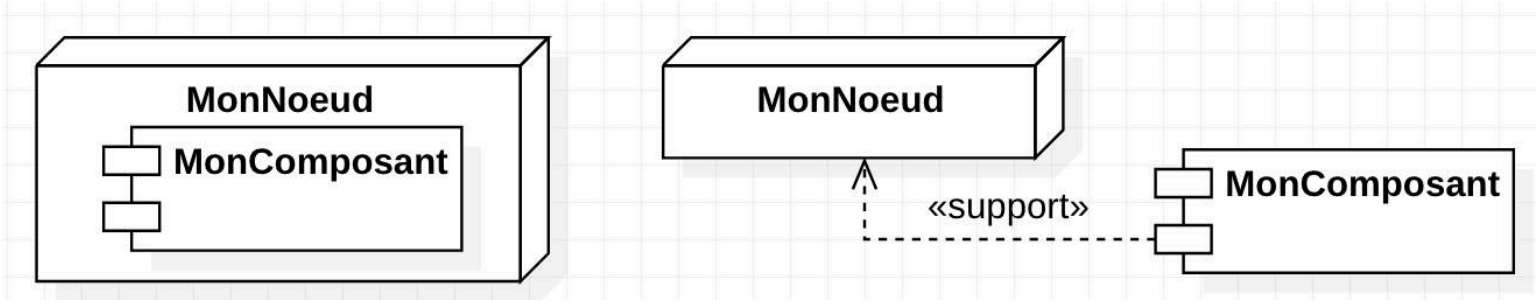
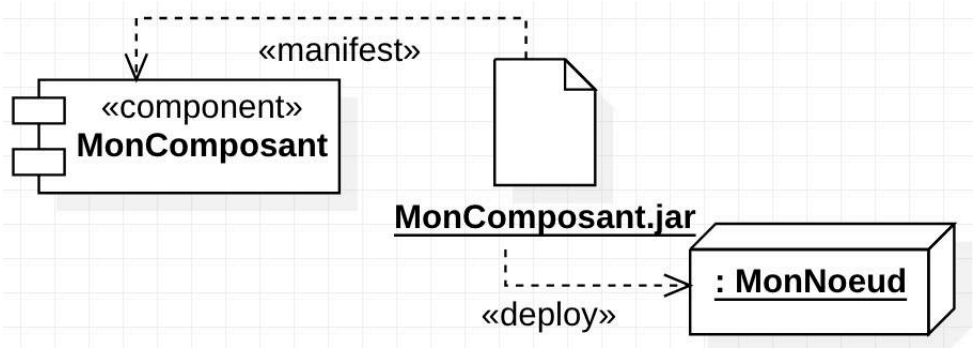


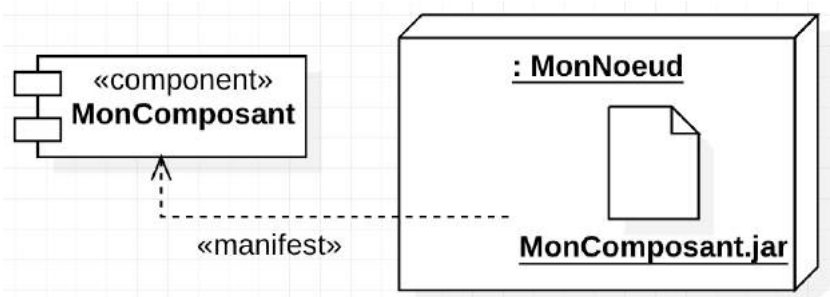
Diagramme de déploiement : Artefact

UML

Rappel : L'artefact est le terme générique qui désigne n'importe quel entité numérique concrète existante dans la réalité physique.

- Composant et artefact
  - ✓ On dit qu'un composant se manifeste sous la forme d'un artefact ou qu'un artefact est la manifestation d'un composant.
  - ✓ Notation UML : le composant est relié à un artefact par une dépendance stéréotypée <<manifest>> orienté de l'artefact vers le composant
- Artefact et nœud
  - ✓ Un composant peut être affecté à un nœud alors qu'un artefact peut lui être déployé sur un nœud
  - ✓ Notation UML : l'artefact est relié à un par une relation de dépendance stéréotypée « deploy » pointant de l'artefact vers le nœud. L'artefact peut également être placé à l'intérieur du nœud sur lequel il est déployé.

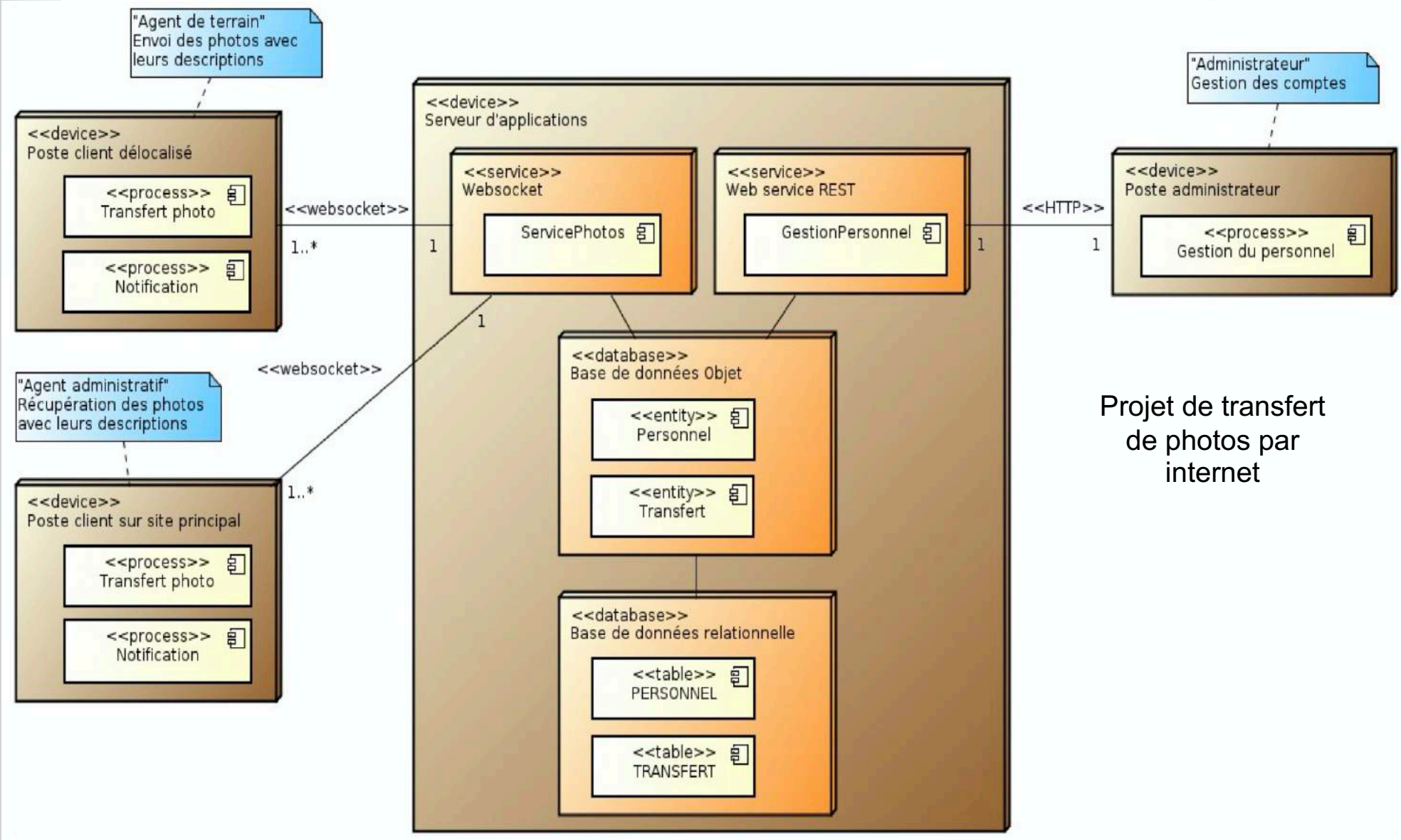




Exemple issu du support suivant :

http://remy-manu.no-ip.biz/UML/Cours/coursUML9.pdf

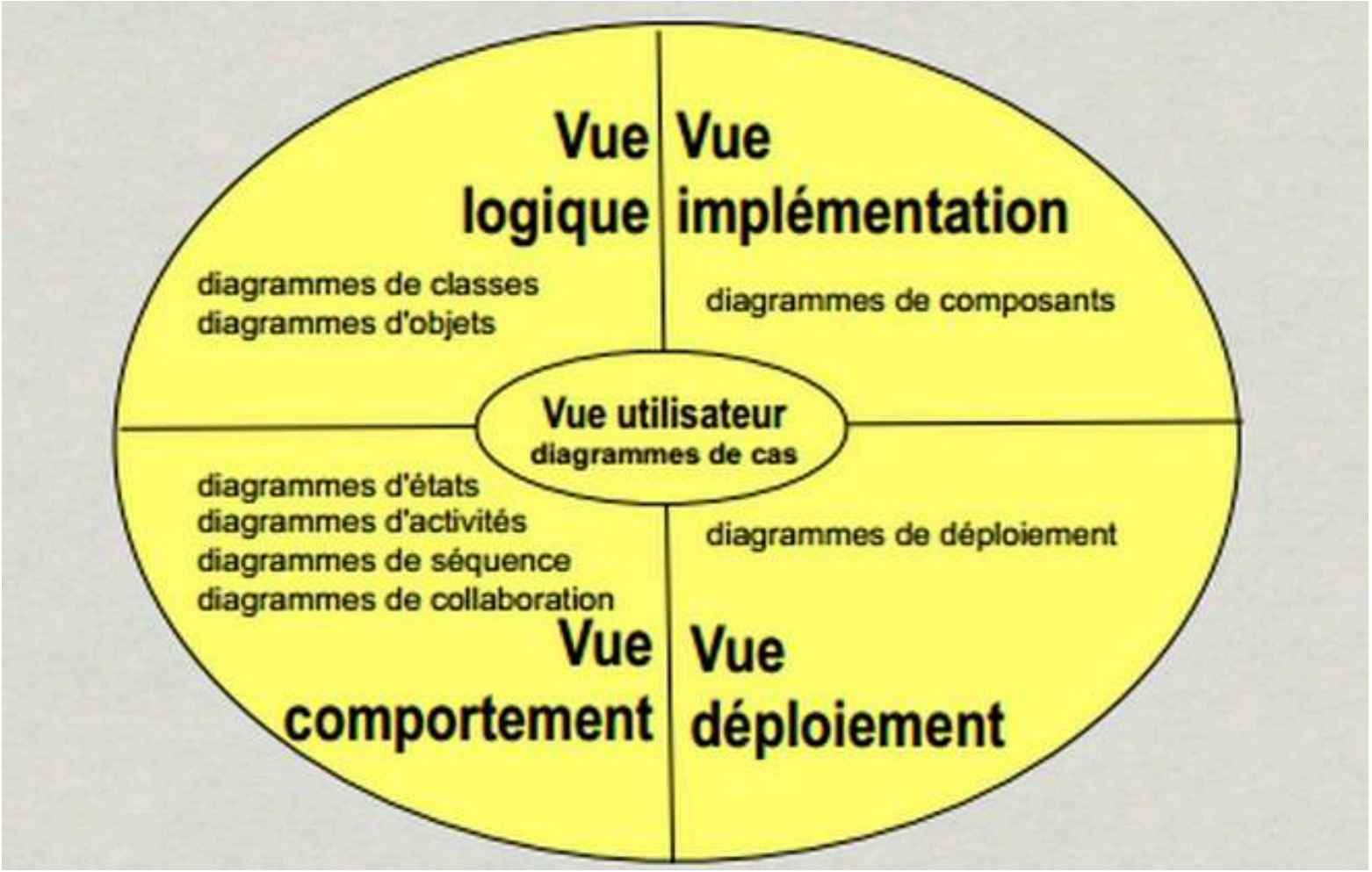
UML



Projet de transfert de photos par internet

Synthèse des diagrammes

UML



Source : http://anubis.polytech.unice.fr/iut/_media/2012_2013/lp/idse/gl/architecture.pdf

UML

Fin du Chapitre 3

Les diagrammes de modélisation