

Modélisation dans le développement logiciel

Exemples cours

Une partie inspirés de

- Software Engineering, 9th ed. - Ian Sommerville
- 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 système

- le développement de modèles abstraits d'un système
- un modèle  un point de vue (perspective)
- souvent utilisant des notations graphiques (Unified Modeling Language - UML)

Modélisation système

- modéliser le système existant
 - ▶ base pour l'analyse des besoins
- modéliser le nouveau système
 - ▶ expliquer les services

Modèles (graphiques)

- **Faciliter la communication**
 - ▶ peuvent être incomplets, voire incorrects
- **Documenter le système**
 - ▶ modèles précis
- **Générer l'implantation**
 - ▶ modèles précis et corrects

➡ utilisés tout le long du cycle de développement

Modélisation système

- Un **modèle** est créé à plusieurs niveaux (avec plus au moins de détails)
- L'**abstraction** est un principe fondamental de la modélisation
 - ◆ regarder les informations importantes à un moment (niveaux) donné
 - ◆ cacher les détails

Un bon modèle

- utiliser une notation standard
- être compris des clients et utilisateurs
- permettre aux développeurs de bien saisir le système

Modélisation

- Plusieurs points de vues (perspectives)
- Chaque point de vue (view) contient
 - ▶ informations uniques
 - ▶ informations partagées avec d'autres views
 - ▶ informations partagées cohérentes

Perspectives

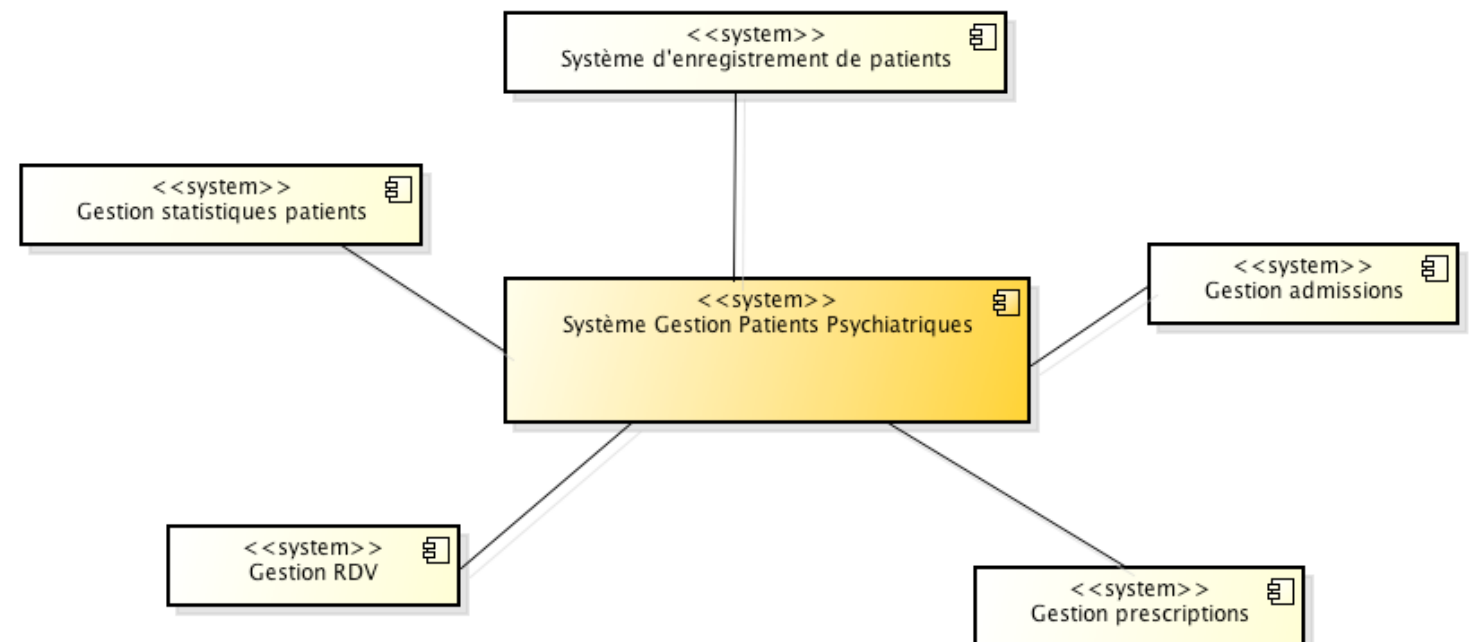
- **externe**
 - ▶ modéliser le contexte (l'environnement)
- **interaction**
 - ▶ modéliser les interactions avec l'environnement et entre les composants
- **structurelle**
 - ▶ modéliser l'architecture et les données
- **comportementale**
 - ▶ modéliser la dynamique

Modélisation du contexte

- illustrer le contexte opérationnel d'un système
- donner des informations sur les frontières du système
- illustrer les relations avec les autres systèmes/acteurs

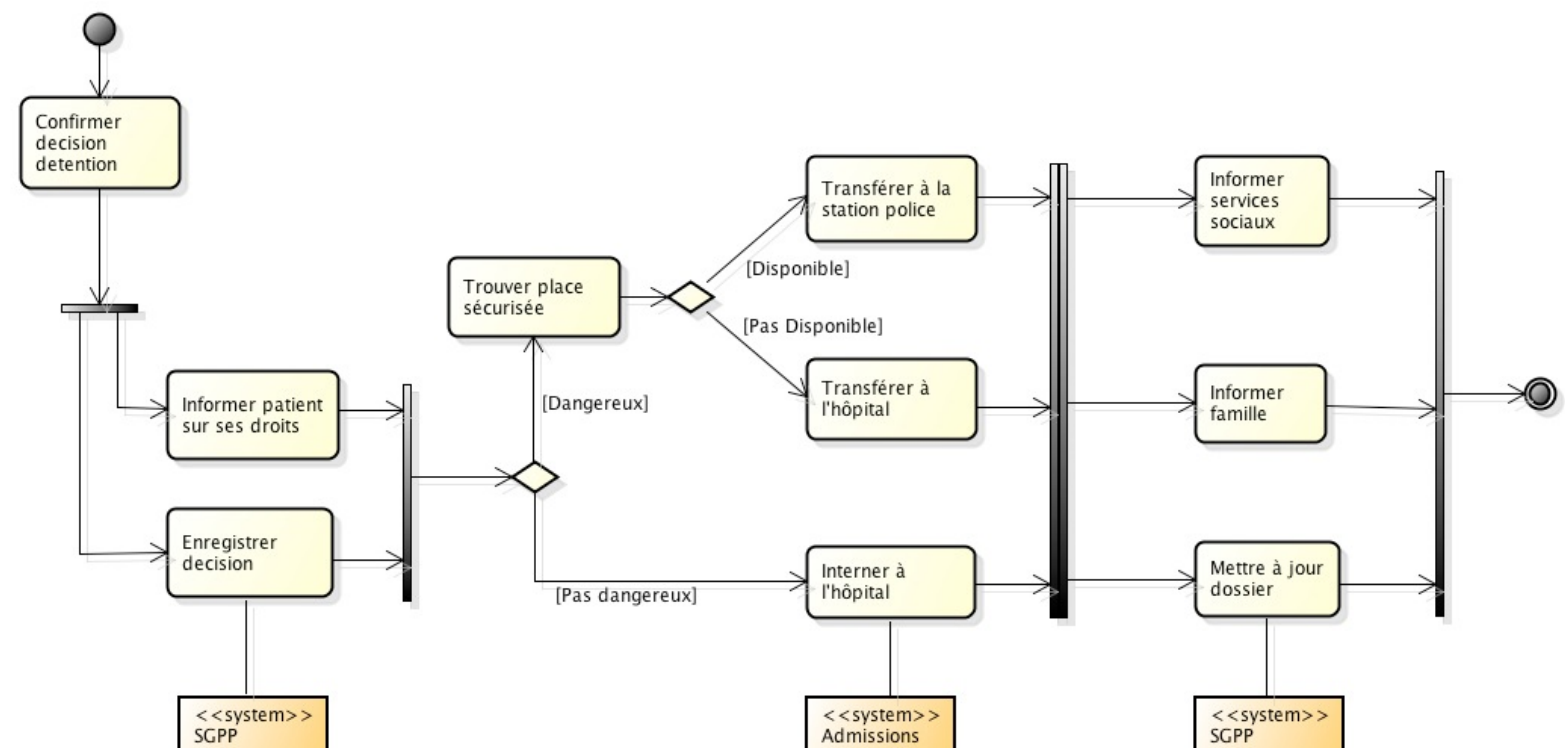
Frontière système

- délimiter les fonctionnalités et montrer les systèmes utilisés ou utilisant le système
- impact important sur le cahier des charges
- peut être politique



Perspective dynamique

- complète la perspective statique de la modélisation du contexte
- illustre l'utilisation du système
- éventuellement avec des diagrammes d'activité



Modélisation interactions

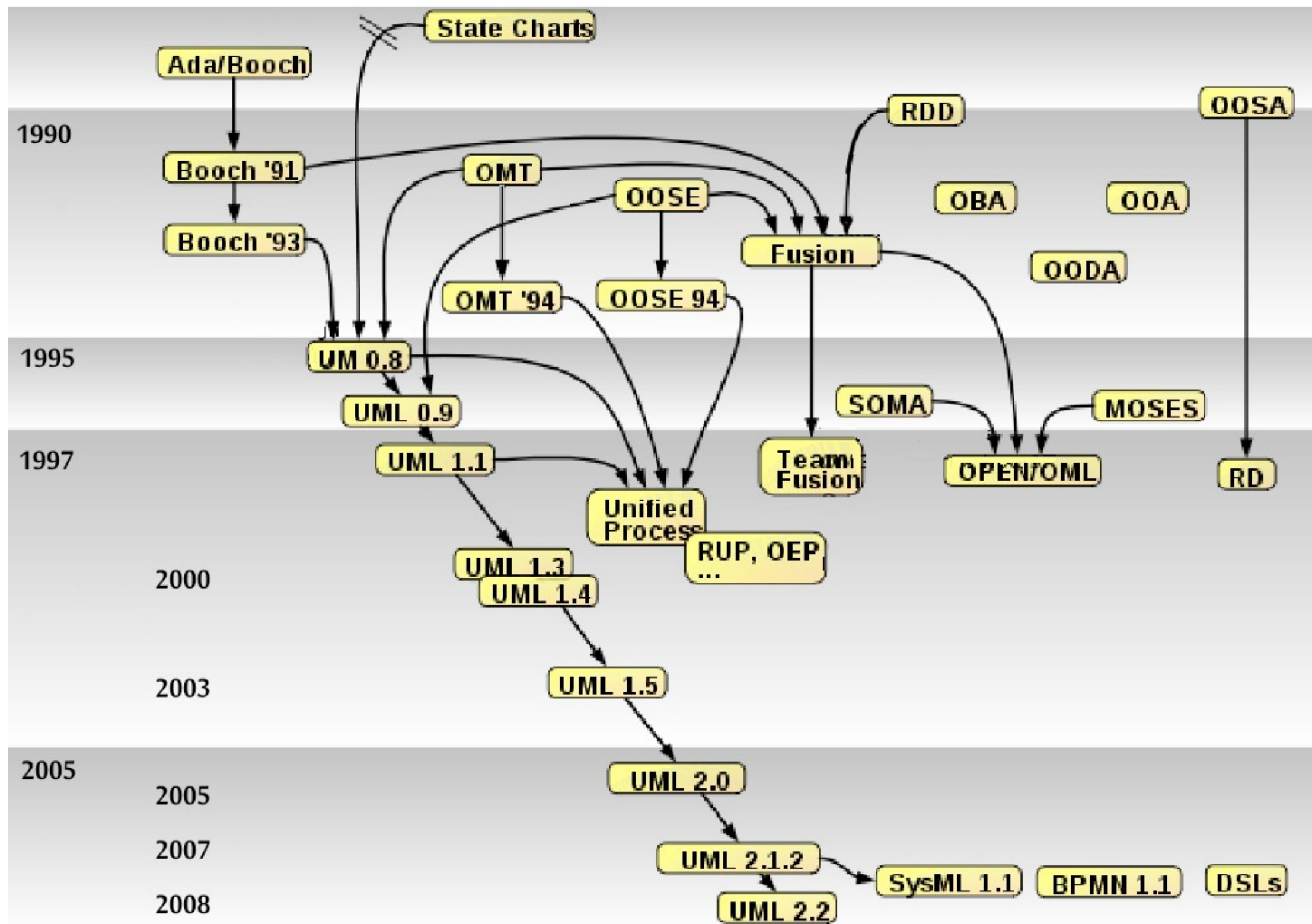
- permet d'identifier les besoins
 - identifier les problèmes de communication
 - identifier les problèmes de fiabilité et performance
- ➡ UML : use-cases, diagrammes d'interaction
(diagrammes de séquence)

UML

Unified Modeling Language

UML

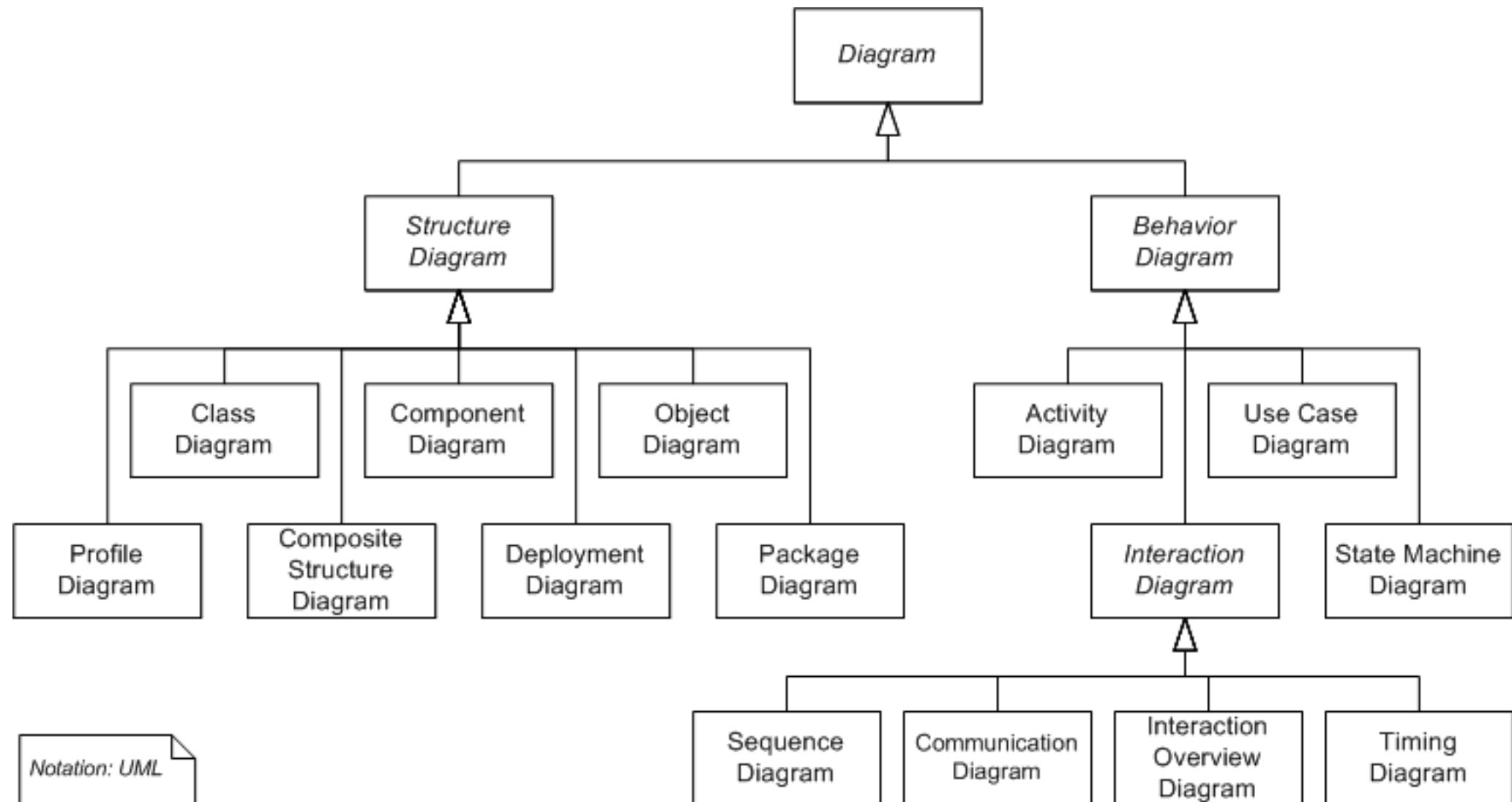
- fin 1980 - début 1990 : premiers processus de développement orienté-objet
 - ▶ OMT (James Rumbaugh)
 - ▶ Booch (Grady Booch)
 - ▶ OOSE (Ivar Jacobson)
- 1997 : « Object Management Group » (OMG) lance le processus de standardisation UML
 - ▶ Standardisation depuis 2000



Utilisation UML

- comprendre et décrire des besoins
- communiquer
- spécifier
- documenter des systèmes
- esquisser des architectures logicielles
- concevoir des solutions

Diagrammes UML



Avantages

- Indépendant des méthodes de développement
- Indépendant des langages de programmation
- Adapté à toutes les phases du développement
- Peut être utilisé pour modéliser autre chose que des logiciels (mécanisme d'extension)

Caractéristiques de UML

Une *sémantique* détaillée

Un *mécanisme* intégré d'extension

Un langage textuel associé utilisé pour la description des contraintes: *Object Constraint Language* (OCL)

L'objectif de UML est d'assister le développement du logiciel

Ce n'est pas une *méthodologie*

Utilisation des diagrammes dans le processus de développement

Analyse des besoins

- ***Use-case diagrams*** : décrire les fonctionnalités du système (par rapport au point de vue des utilisateurs)
- ***Activity diagrams*** : décrire l'activité du système
- ***Sequence diagrams*** : décrire l'interaction entre utilisateurs et système

Analyse - Conception

- **Class diagrams** : structures de données du système, relations, etc. (faire abstraction de ce qui est extérieur au système - IHM, SGBD, ...)
- **Sequence diagrams** : interactions entre objets lors de la réalisation de chaque fonctionnalité
- **State diagrams** : le comportement d'une famille d'objets en termes d'états et de transitions entre états

Conception

- ***Class diagrams*** : relations entre classes (plus détaillées)
- ***Sequence diagrams*** : interactions entre objets (en incluant les nouveaux objets)
- ***Component diagrams*** : architecture des composants
- ***Deployment diagrams*** : déploiement des composants sur les équipements

Implémentation

- Génération automatique du code (squelettes de classes)
- Ecrire **code** en accord avec les différents diagrammes
- Ecrire **tests** en accord avec les différents diagrammes

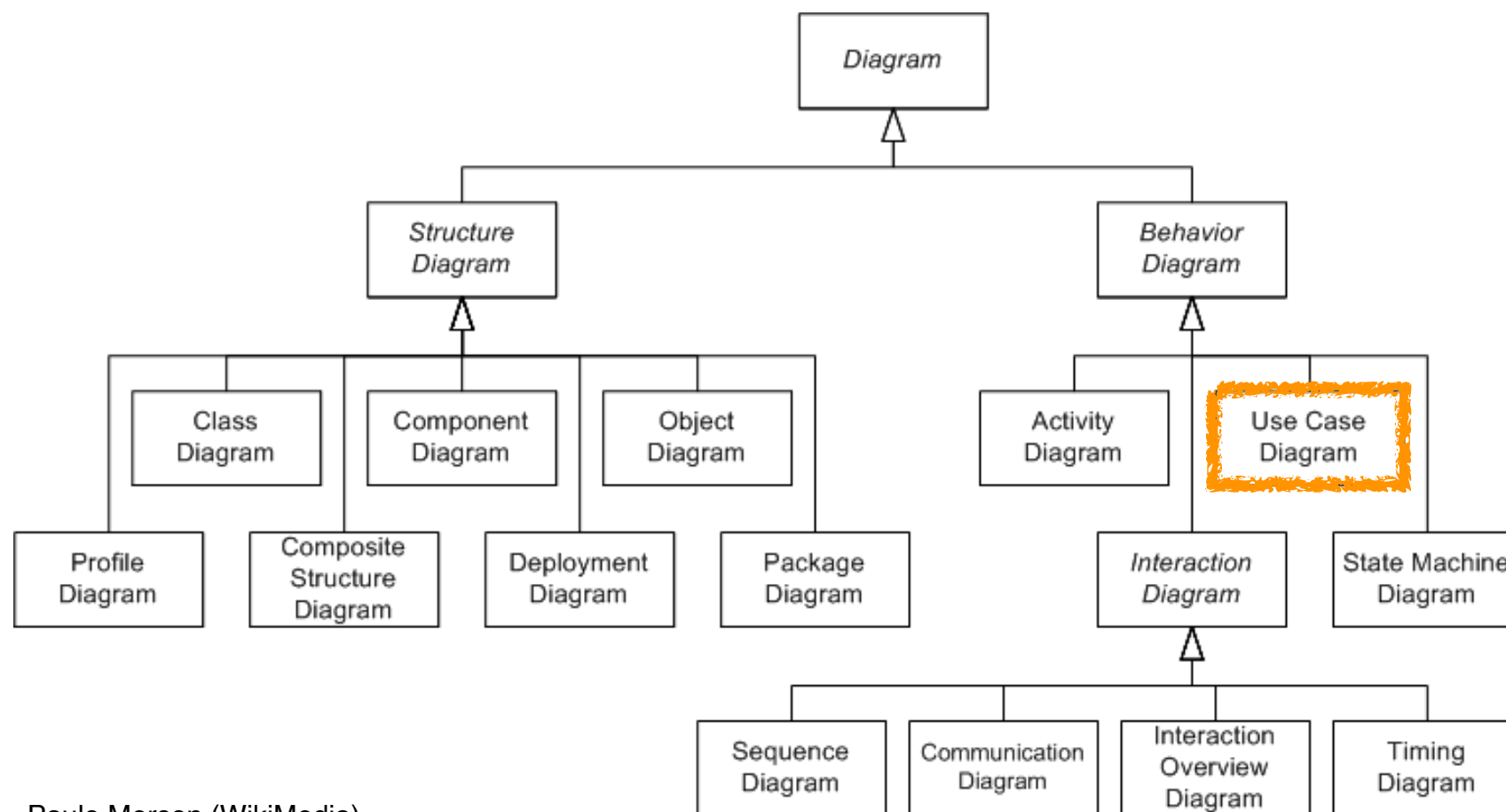
-|-

Modélisation interactions (environnement)

Perspectives

- ***externe***
 - ▶ modéliser le contexte (l'environnement)
- ***interaction***
 - ▶ modéliser les interactions *avec l'environnement* et entre les composants
- ***structurelle***
 - ▶ modéliser l'architecture et les données
- ***comportementale***
 - ▶ modéliser la dynamique

Diagrammes cas d'utilisation (Use-case diagrams)



Paulo Merson (WikiMedia)

Use-cases

- une séquence typique d'actions qu'entreprend un utilisateur afin d'accomplir une tâche donnée
- **Objectif** : *modéliser le système*
 - ▶ ... du point de vue de la façon par laquelle l'utilisateur **interagit** avec le système
 - ▶ ... afin d'accomplir ses **objectifs**

Identification use-case

Modélise un service rendu par le système

- regroupe une famille de scénarios d'utilisation selon un critère fonctionnel

Comment construire les use-cases ?

- Pour chaque acteur, identifier les intentions avec lesquelles il communique avec le système

Use-cases

- un moyen de déterminer les besoins fonctionnels de l'application
 - ▶ partie intégrante du cahier des charges
 - ▶ formalisme basé sur le langage naturel,
 - ▶ compréhensibles par les experts et les informaticiens

Identification use-case

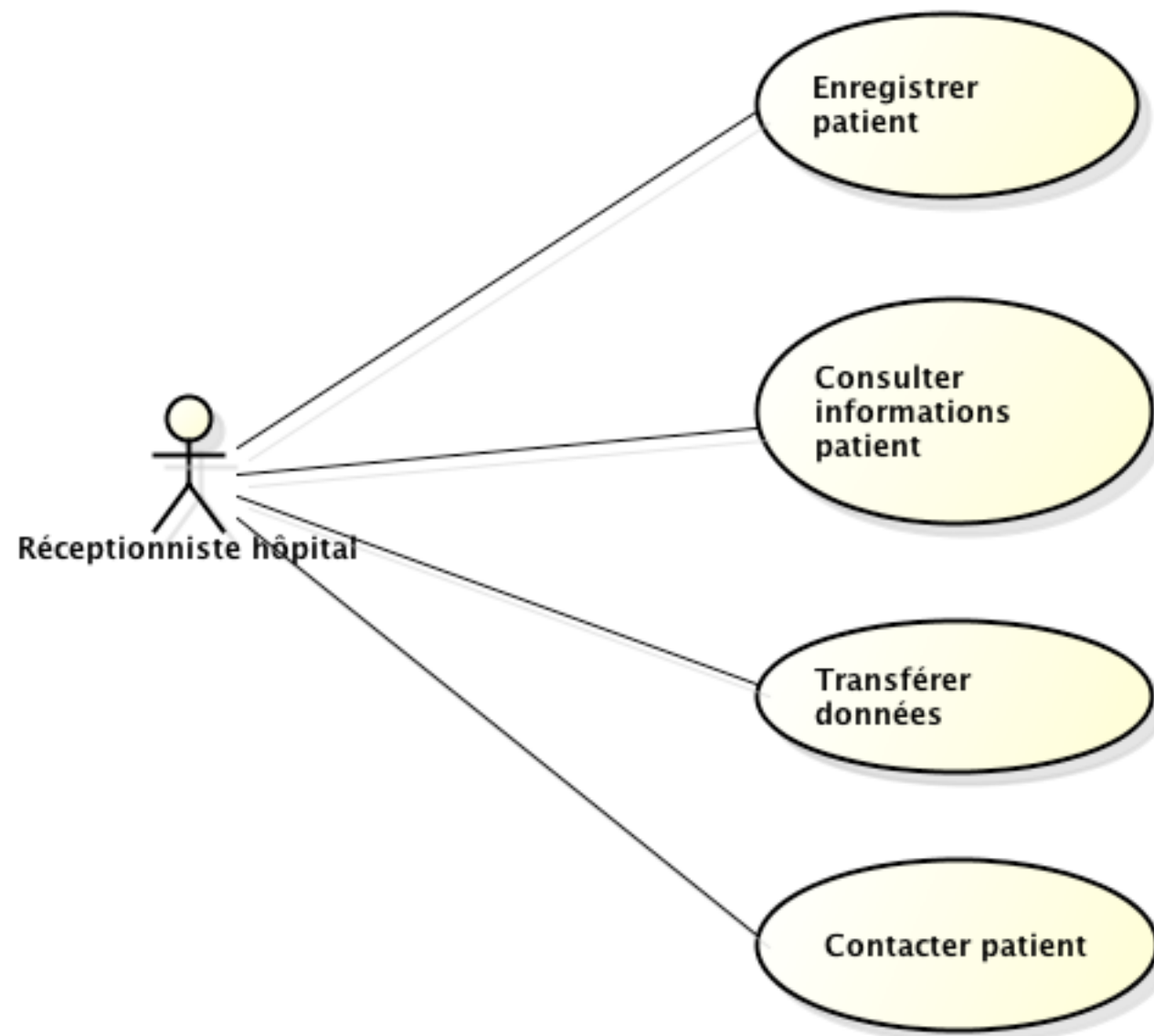
*Exemple : **client GAB***

- retirer de l'argent
- consulter le solde
- déposer de l'argent

Exemple



Diagramme composite use-cases



Les use-case

- Un **use-case** doit couvrir l'ensemble des étapes à suivre dans l'accomplissement d'une tâche donnée
- Le **use-case** doit être écrit d'une façon indépendante de toute interface utilisateur
- Un **modèle** avec use-cases contient
 - un ensemble de use-cases
 - un diagramme expliquant comment ils sont reliés

Les use-case

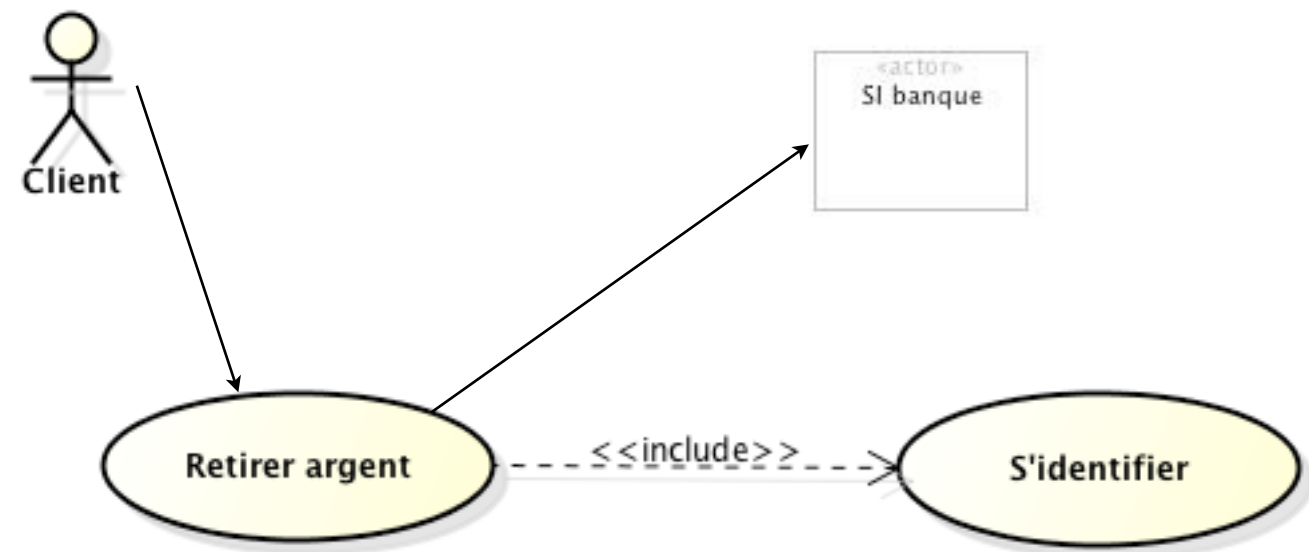
- Le **use-case** doit décrire l'interaction de l'utilisateur avec le système
 - ▶ et **non** les opérations que doit réaliser le système
- Un **use-case** ne doit inclure que les interactions avec le système
 - ▶ et **non** d'autres opérations réalisées par l'utilisateur indépendamment du système

Utilisation use-cases

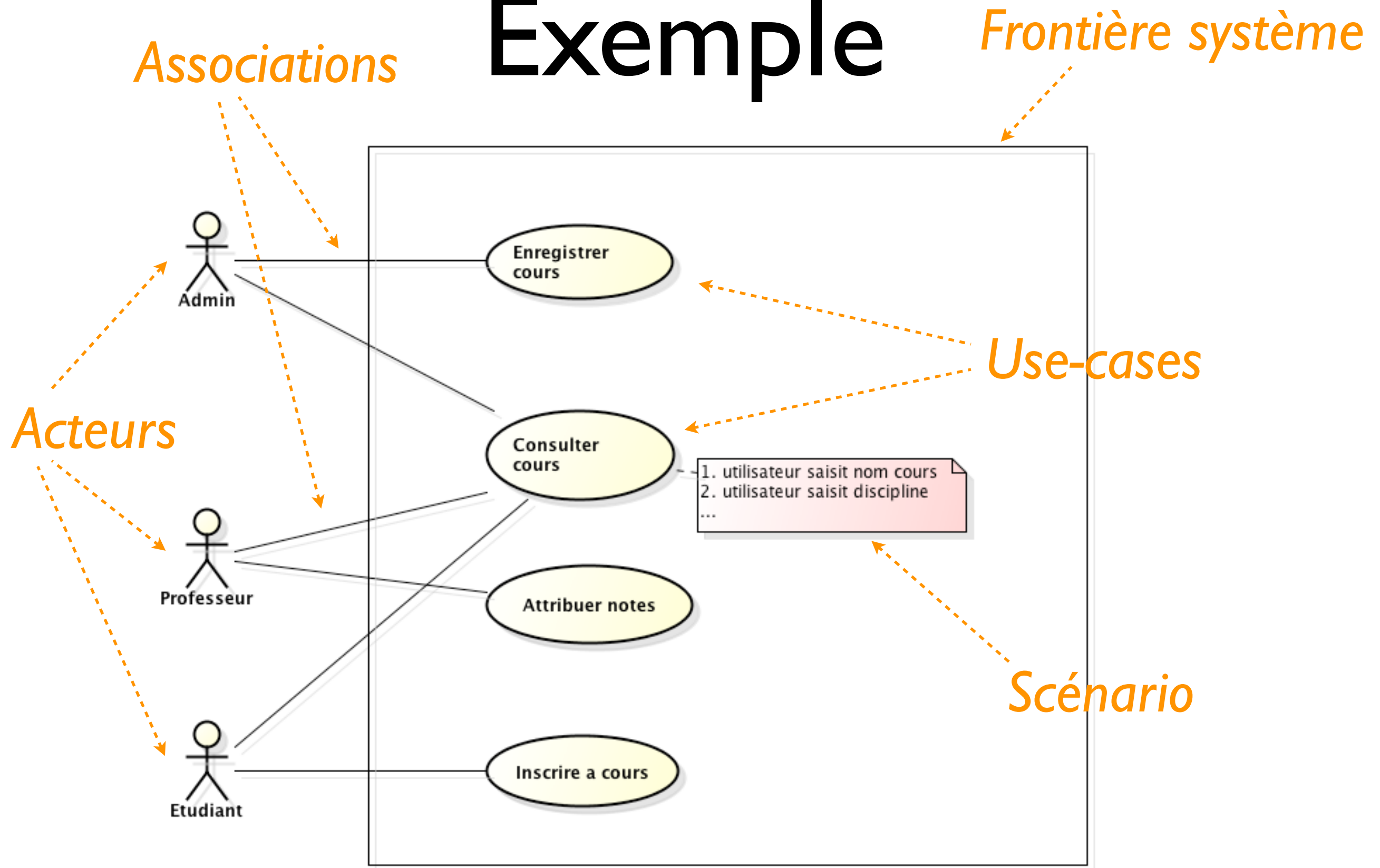
- décrire le comportement du système selon un point de vue *extérieur* (des utilisateurs)
 - décrire les *fonctionnalités attendues* du système
 - préciser les *contraintes non fonctionnelles*, concernant par exemple le matériel, le temps de réponse, etc.
- ➔ Décrire le « **quoi** » et non le « **comment** »

Concepts

- Acteurs
- Use-case (et relations)
- Associations
- Scénarios (documents textuels pour les cas)
- Frontières système, packages
- prototypes IHM (optionnel)
- diagrammes d'activités (optionnel)



Exemple



L'acteur

- représente un **rôle** joué par un utilisateur (extérieur) qui interagit avec le système
 - ➔ personne, système, entité externe, le temps
- lié à un ou plusieurs use-cases
 - ➔ permet de grouper les use-cases
- n'a pas d'autres propriétés que les associations aux use-cases

Utilisateur vs. Acteur

- un ***utilisateur*** peut jouer plusieurs rôles
(un professeur peut aussi participer à un cours)
- plusieurs ***utilisateurs*** peuvent jouer le même rôle

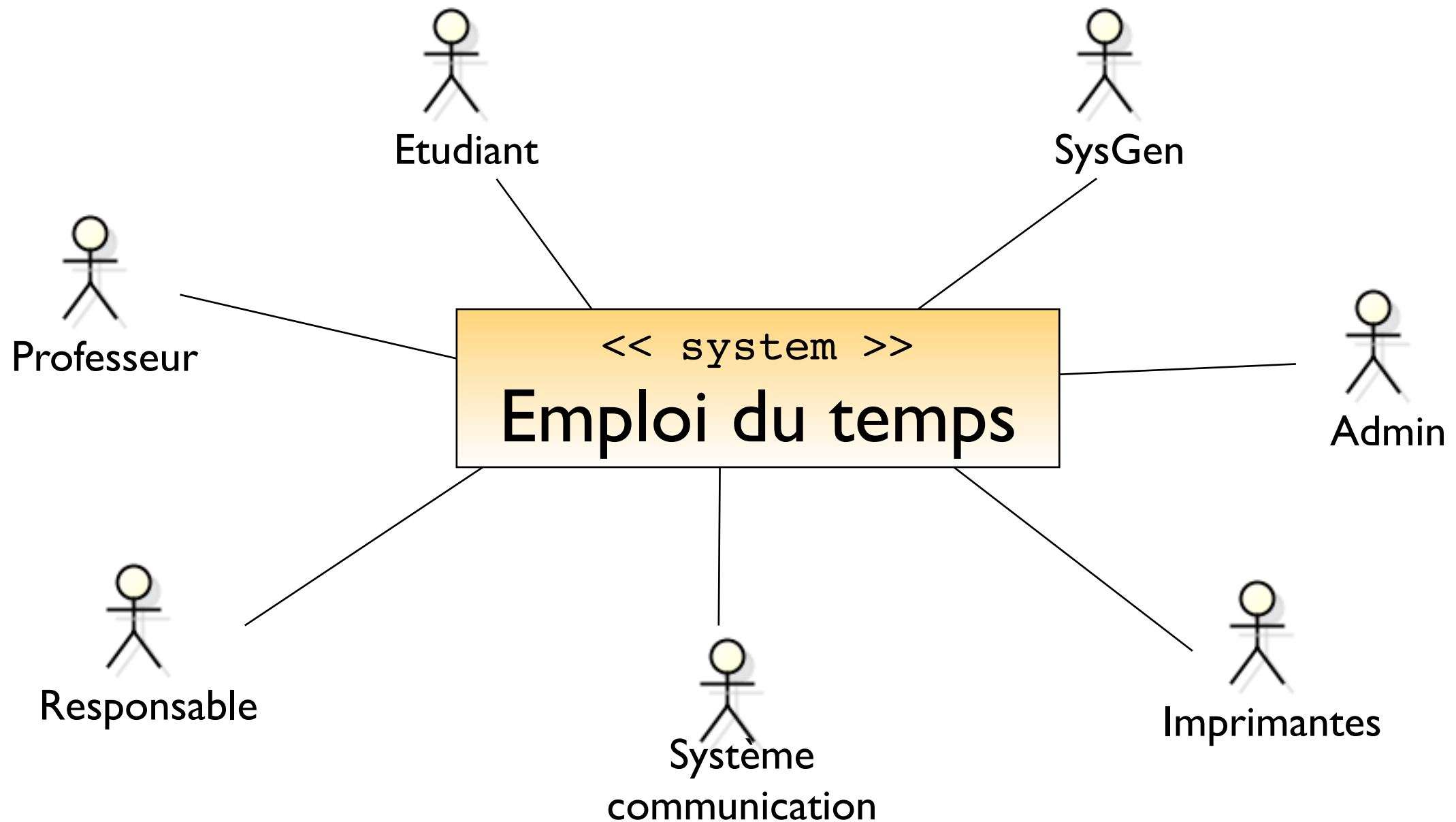
Les acteurs

- Les acteurs communiquent directement avec le système et non pas par un intermédiaire.
 - ◆ le caissier est un acteur, mais pas le client
- Les échanges entre acteurs ne concernent pas la modélisation du système !

Acteur principal/secondaire

- l'acteur **principal** utilise/modifie le système
 - ➔ le réceptionniste hôpital
 - ➔ le client d'un GAB
- l'acteur **secondaire** est sollicité pour obtenir une information complémentaire
 - ➔ système d'enregistrement de patients
 - ➔ le système d'autorisation de CB

Contexte statique



▮▮▮▮ ➡ résumé de la liste des acteurs

Scénarios

Un scénario est une instance d'un use-case

- un acteur spécifique ...
- à un moment spécifique ...
- avec des données spécifiques

↳ Décrit les étapes d'interaction entre un utilisateur et le système

Use-case

- collection de scénarios de base avec alternatives
- *nom* : le but de l'acteur qui le déclenche
 - ▶ exemple: *consulter solde, inscrire étudiant*
- une procédure **complète** et **self-contained**

Use-case

A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state.

C. Larman, Applying UML and Patterns

Use-case

- facile à comprendre
- non ambigu
- complet - contient toutes les alternatives
- confirmé par les utilisateurs du système
- mappé vers un prototype d'interface
- ~~un morceau de (pseudo)code~~
- ~~une liste de fonctions~~

Organisation use-cases

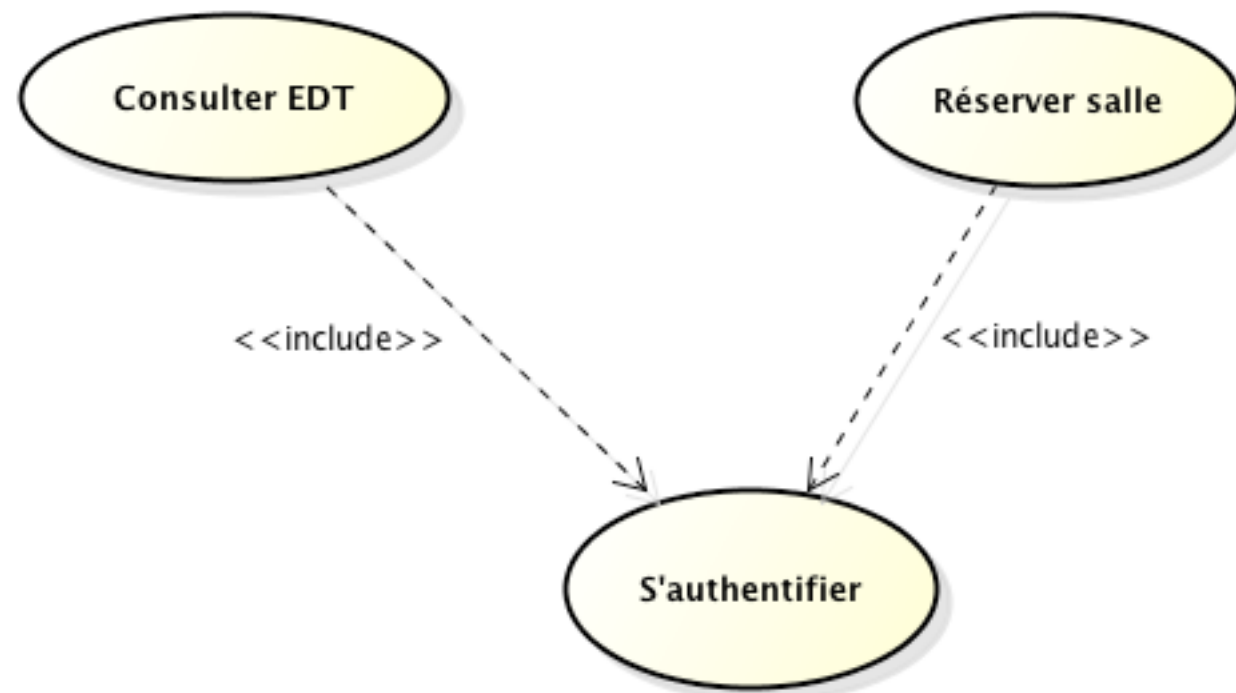
Il est possible (et souhaitable) d'organiser les use-cases

- ▶ en regroupant les use-cases en packages, par acteur et/ou par use-case,
- ▶ en utilisant les relations d'**inclusion**, d'**extension** et de **généralisation** entre use-cases.

Inclusions

- décrire les points communs contenus dans plusieurs use-cases différents (et éviter la répétition de ces détails dans chacun des use-cases)
- inclus dans d'autres use-cases
- représente l'exécution d'une tâche de plus bas niveau pour accomplir un but aussi de plus bas niveau

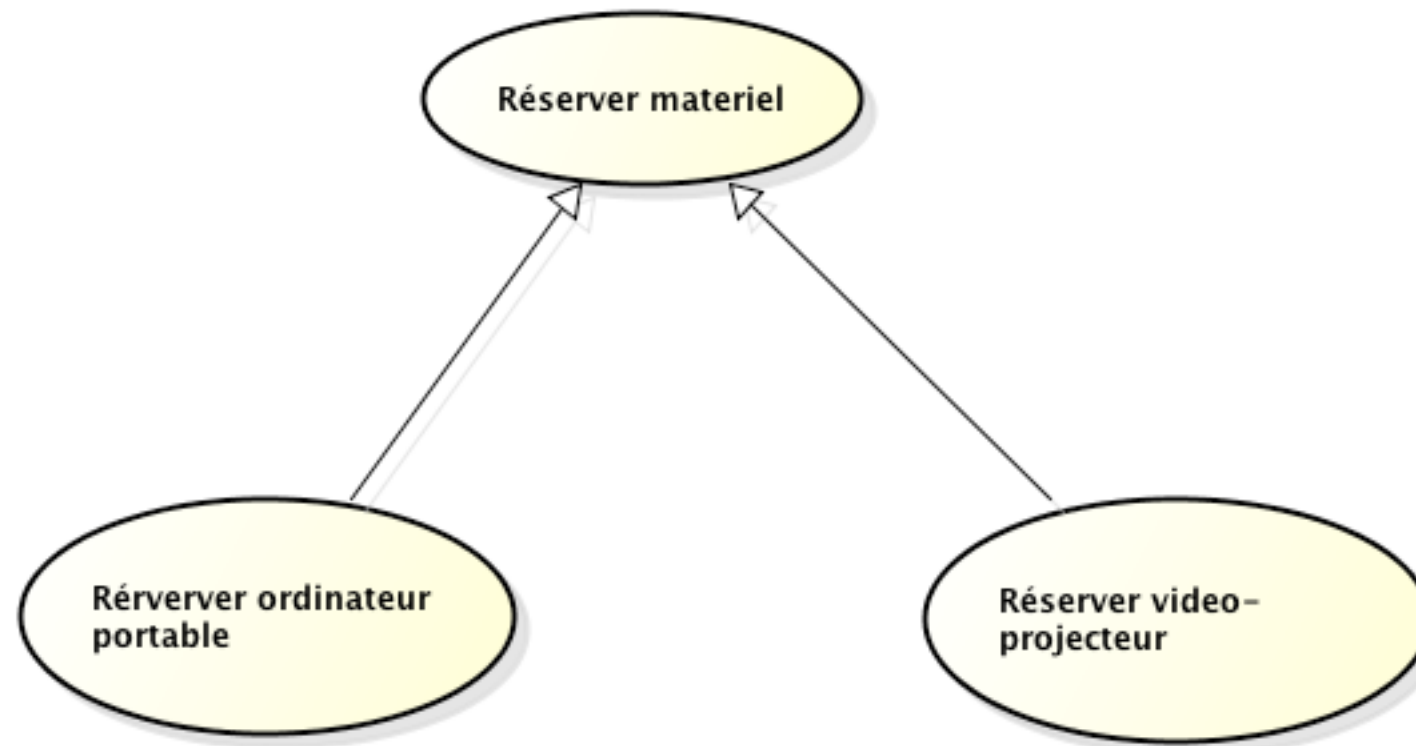
Inclusion



Généralisations

- Similaire aux super-classes dans les (diagrammes de) classes
- Un use-case généralisé représente plusieurs use-case similaires
- La spécialisation d'un use-case fournit les détails spécifiques relatifs à un de ces use-cases similaires
 - ▶ peut remplacer un ou plusieurs enchaînements d'actions du use-case hérité

Généralisations

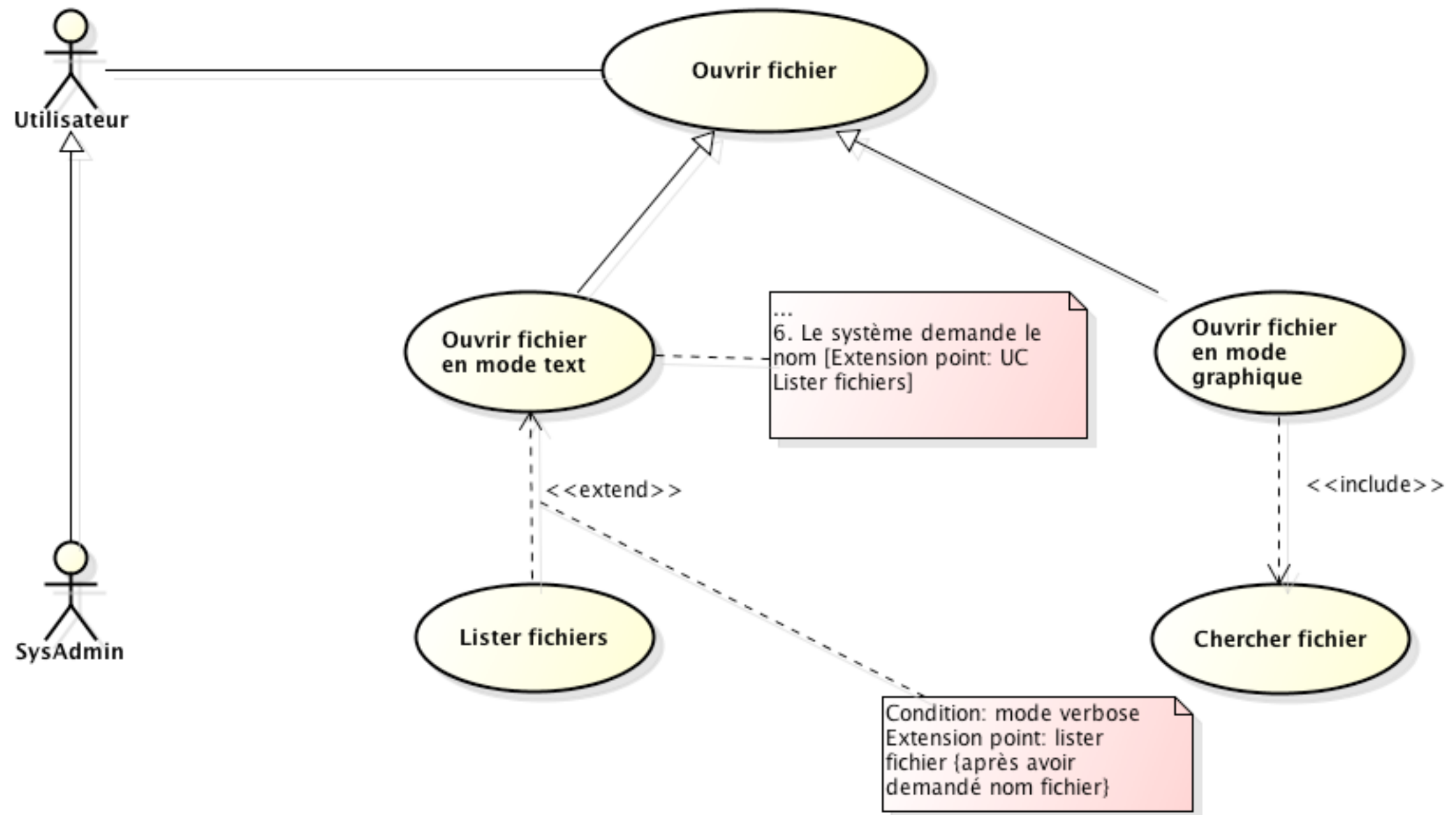


▣➡ Le cas fils hérite du comportement et de la signification du cas père (il peut le compléter ou le remplacer)

Extensions

- Utilisées afin de spécifier des interactions optionnelles et permettent de modéliser
 - ▶ des partie vues comme optionnelles par l'acteur
 - ▶ des flux conditionnels
 - ▶ l'insertion de flux conditionnés par l'interaction avec l'acteur
- En créant un use-case d'extension, la description principale peut demeurer simple

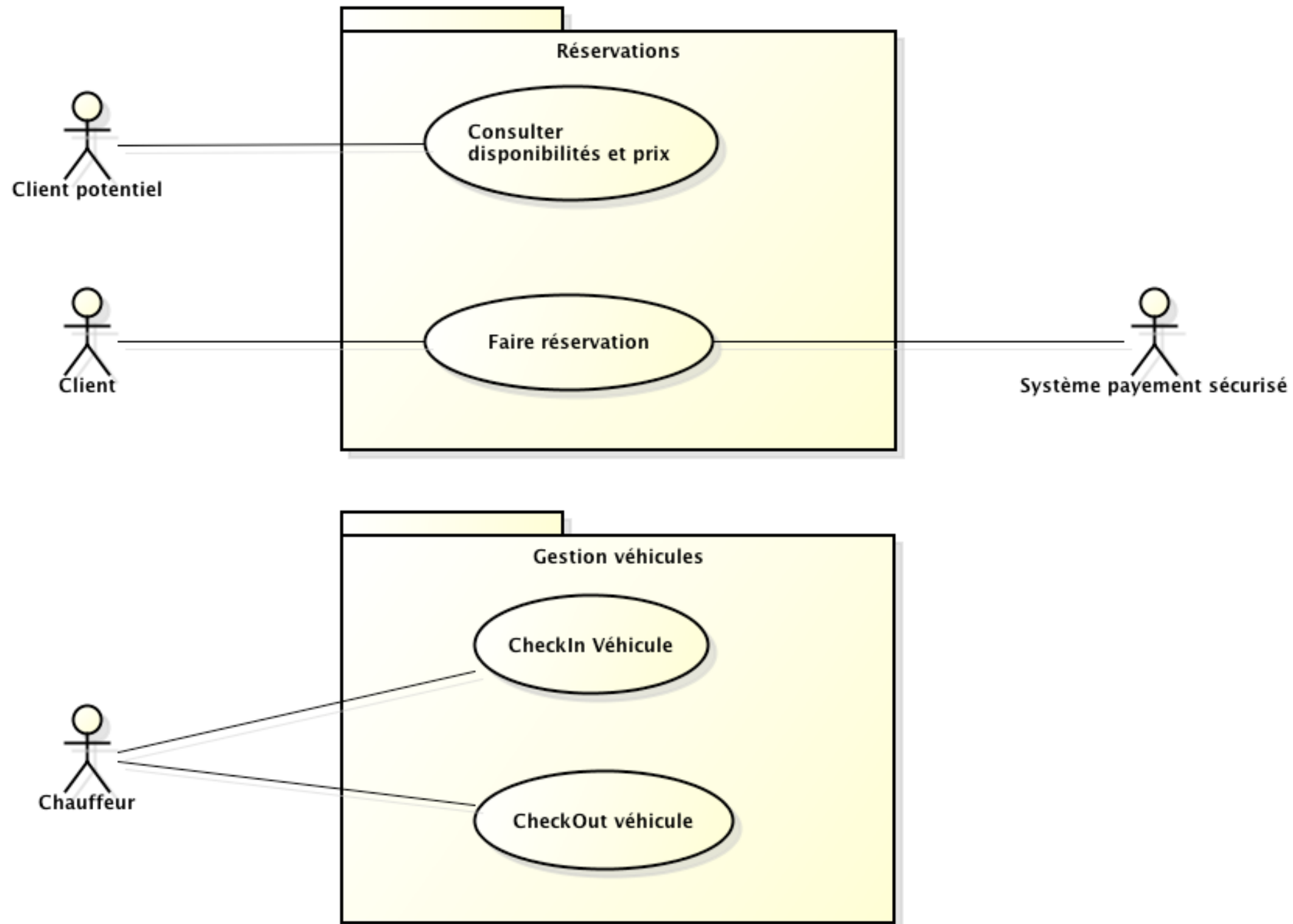
Exemple



Extends vs. Generalises

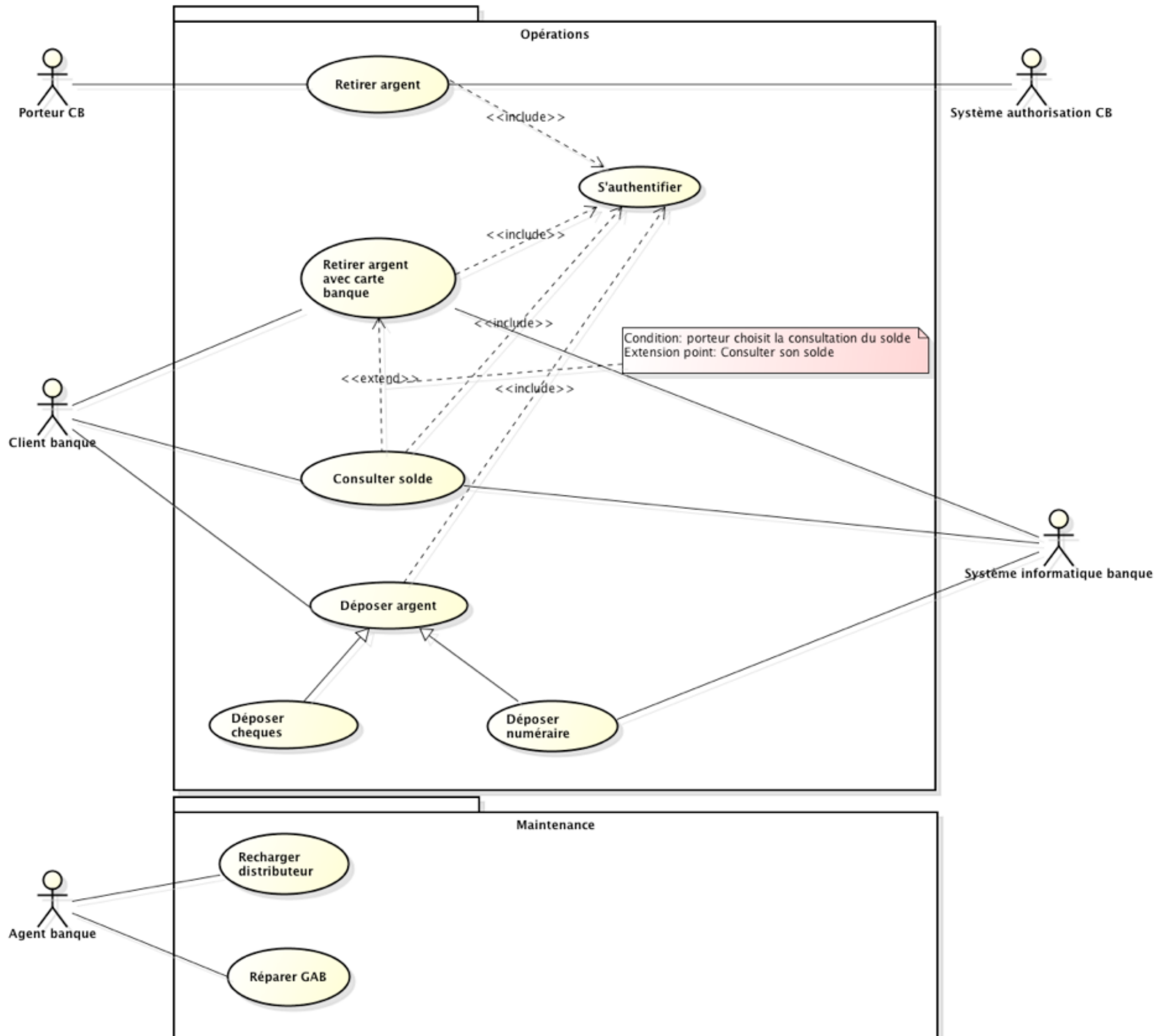
- ▶ distinguer les variantes d'un use-case
- ▶ acteur lié au use-case de base
- ▶ acteur exécute le use-case et les extensions
- ▶ identifier comportements communs
- ▶ pas d'acteur lié au use-case de base (en général)
- ▶ acteurs différents pour les généralisations

Structurer en package

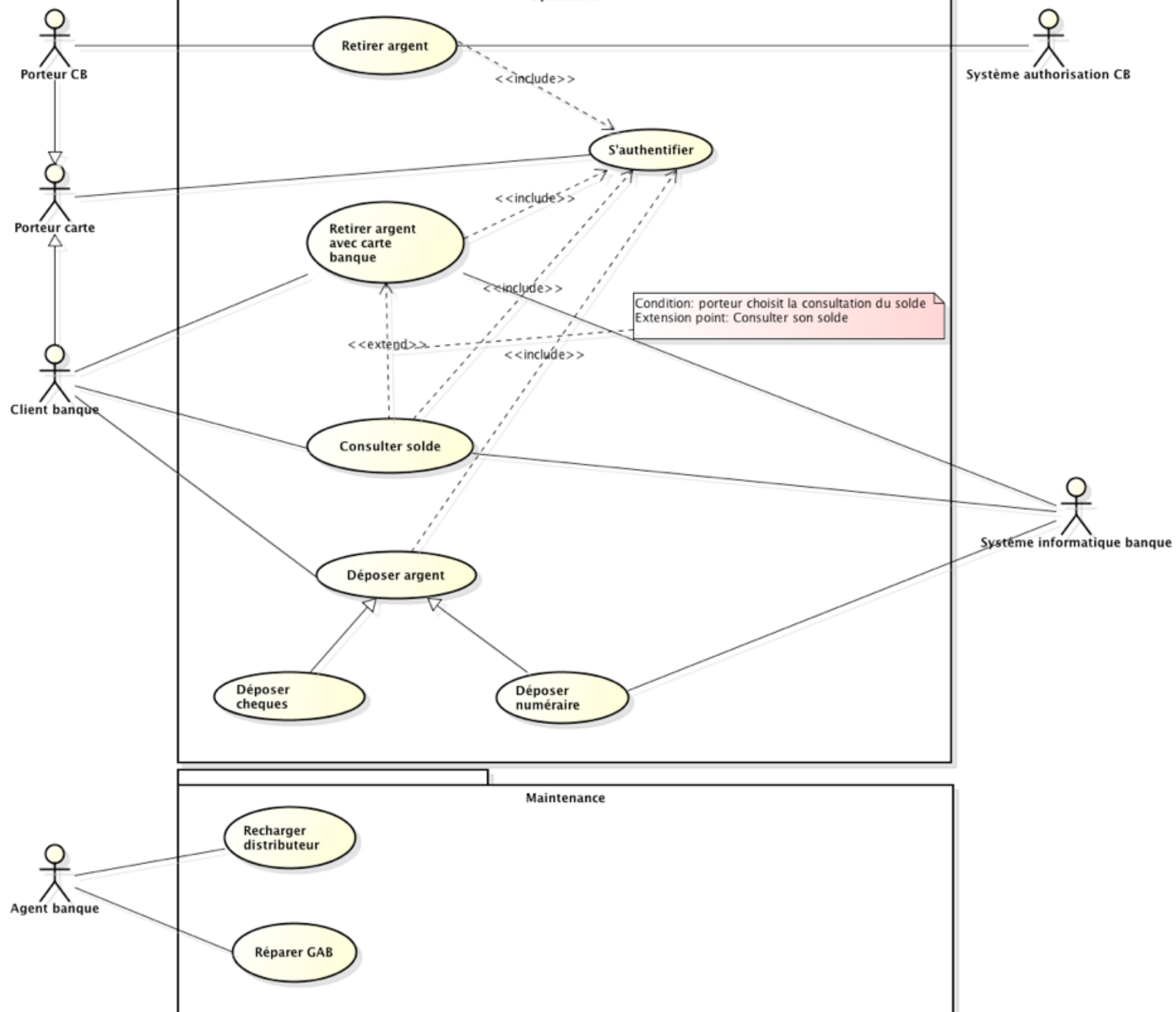


Modélisation Guichet Automatique de Banque (GAB)

- **Ses principales fonctions :**
 - ▶ distribution d'argent à tout porteur d'une carte de la banque (autorisation d'un certain montant par le Système d'Information de la banque) ou d'une carte bancaire (autorisation à distance par le Système d'Autorisation CB),
 - ▶ consultation du solde, dépôt en numéraire et de chèques pour les possesseurs d'une carte de la banque.
- **Toutes les transactions sont sécurisées (code personnel vérifié avec le code enregistré sur la puce de la carte ; la carte est avalée après trois échecs). Il faut parfois recharger le GAB et retirer du numéraire et des chèques.**



uc



Détailler les use-case

- Un diagramme de use-cases doit être détaillé
 - ▶ Détailler la dynamique
 - ▶ Décrire le flot d'événements qui le constitue

Description textuelle

- Identifier des scénarios possibles pour un use-case.
 - ▶ Scénario nominal
 - ▶ Scénario(s) alternatif(s)
 - ▶ Scénario(s) d'erreur/exceptionnel(s)
- Aucune normalisation UML

Exemple :

l'intention de l'acteur
(pas comment)

P. Roques

qui fait quoi

Scénario nominal

1. Le porteur de CB introduit sa carte dans le lecteur du GAB.
2. Le GAB vérifie que la carte introduite est bien une CB.
3. Le GAB demande au porteur de saisir son code.
4. Le porteur de CB saisit son code.
5. Le GAB compare le code saisi avec celui inscrit sur la puce.
6. Le GAB demande une autorisation au SA CB.
7. Le SA CB donne son accord et indique le solde.

Exemple : GAB

P. Roques

Scénario nominal

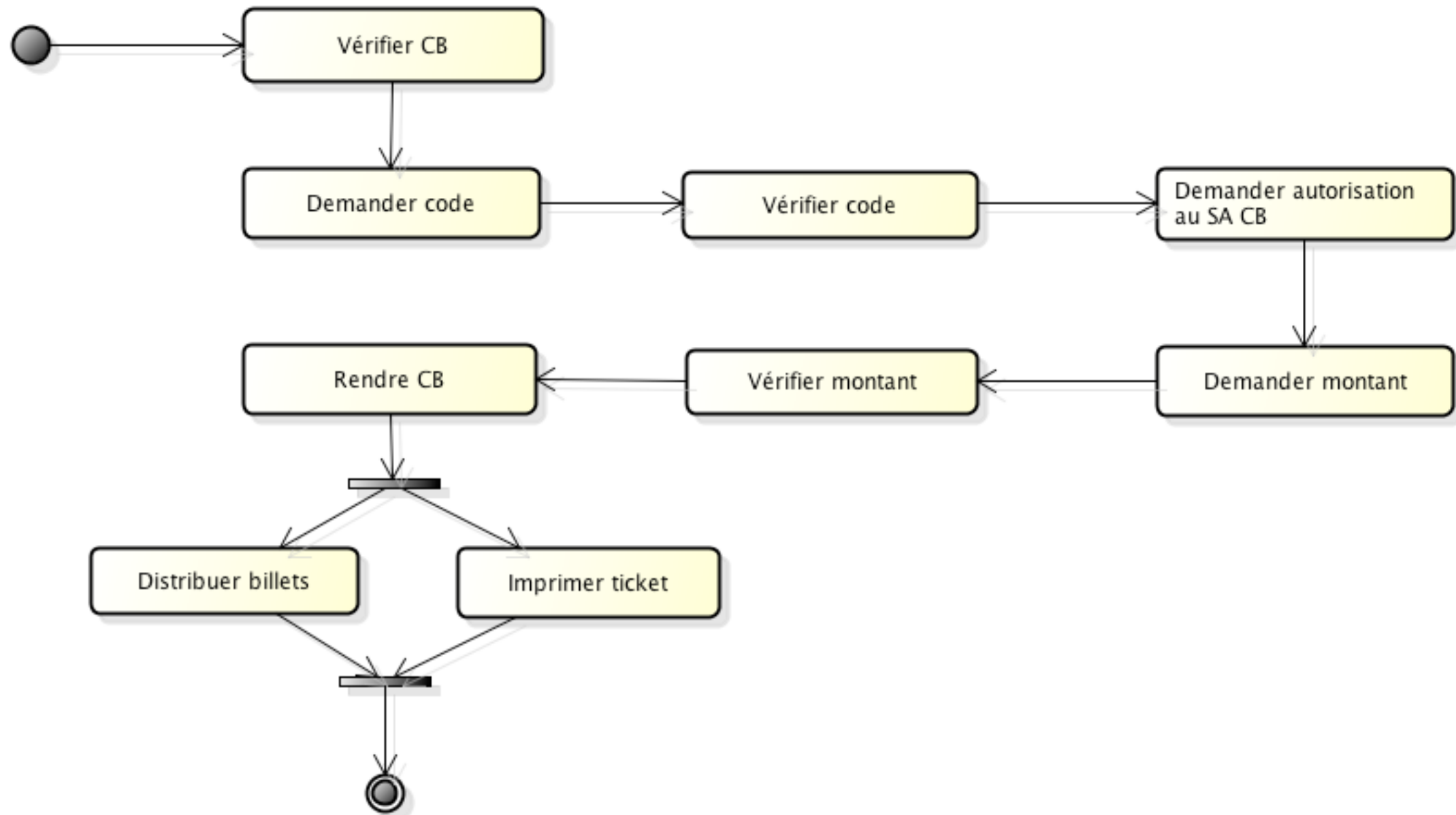
1. Le porteur de CB introduit sa carte dans le lecteur du GAB.
2. Le GAB vérifie que la carte est bien une CB.
3. Le GAB demande au porteur de saisir son code.
4. Le porteur saisit son code.
5. Le GAB vérifie que le code saisi avec celui inscrit sur la puce.
6. Le GAB demande une autorisation au SA CB.
7. Le SA CB donne son accord.
8. ...

alternatives ?

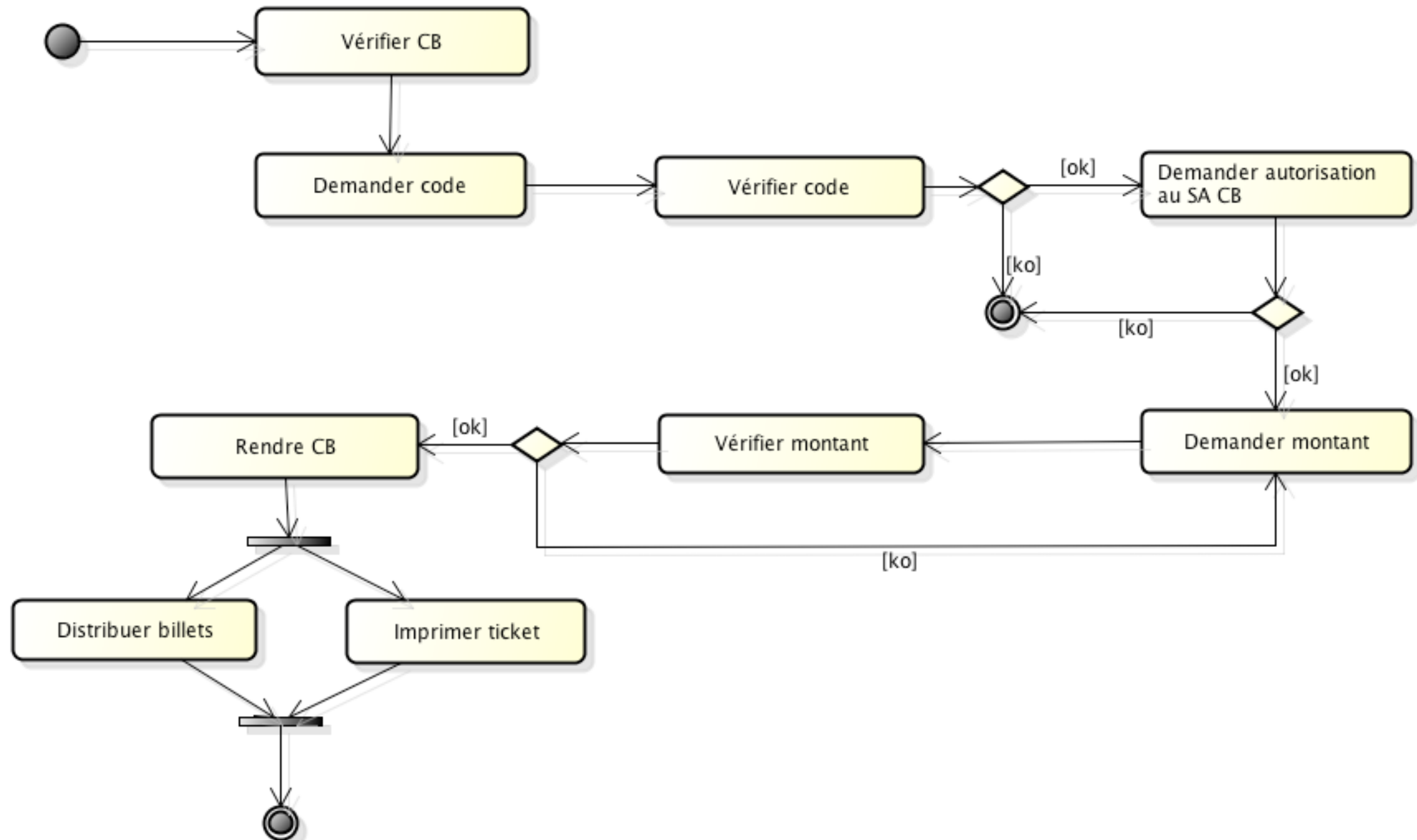
Pour chaque
étape

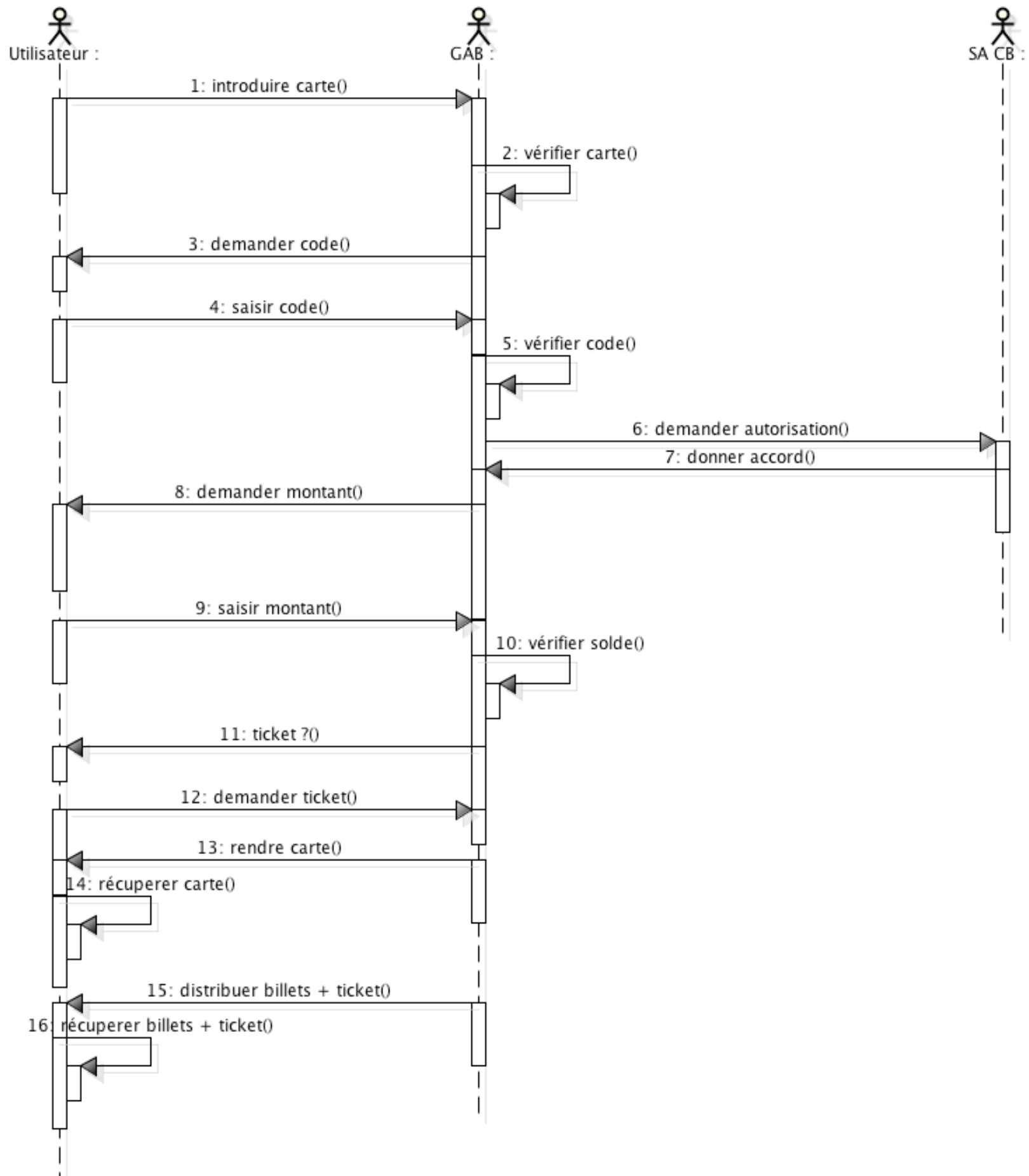
dysfonctionnements ?

Description avec diagrammes



Description avec diagrammes





Les bénéfices d'une modélisation avec use-cases

- Aide à mieux définir la portée du système
- Permet de définir et de valider les exigences
- Permet de définir les scénarios de tests
- Aide à mieux planifier le développement du logiciel
- Permet de mieux structurer les manuels d'utilisation

Inconvénients

- Ne prennent pas toujours en compte les besoins non fonctionnels
- Apprentissage nécessaire pour les lire
- ...

Résumé use-cases

- Le(s) diagramme(s) de use-cases ne représentent qu'une (petite) partie de la description des besoins.
 - *acteurs*
 - *use-cases*
 - *relation entre use-cases*
- La description textuelle est absolument indispensable pour une description détaillée car le diagramme lui-même est peu instructif.

Quizz



1

Allez sur wooclap.com

2

Entrez le code d'événement dans le bandeau supérieur

Code d'événement
ABEYAY



Activer les réponses par SMS