Université Paris 13- PRES Sorbonne Paris Cité

Introduction au langage UML

A.Osmani

UML: Unified Modeling Language)

Grady Booch, James Rumbaugh, Ivar

Site officiel UML : www.omg.org

UML à quoi sert-il ?

C'est un langage simple, très expressif et offrant plusieurs vues pour développer des applications.

UML est un langage de modélisation graphique pour

- comprendre et décrire des besoins
- spécifier des systèmes, simples ou complexes
- concevoir des solutions
- documenter un système le long de son cycle de vie

UML est un langage à usage général, il est utilisable quel que soit

- le type de système - logiciel, matériel, organisation
- le domaine métier - ingénierie, finance
- l'étape de développement -recueil de besoins, analyse, conception, etc.
- le processus de développement choisi

UML est utilisé pour visualiser, spécifier, construire et documenter le systèmes.

Ou peut-on utiliser UML ?

Il peut être utilisé pour réaliser tout type de logiciel, en particulier dans les domaines :

- ✓ Les entreprises traitant des systèmes d'informations
- ✓ Les banques et assurances
- ✓ Les télécommunications
- ✓ Défense et aérospatiale
- ✓ Médias électroniques
- ✓ Les services WEB distribués
- ✓ etc.

UML n'est pas limité à la modélisation de logiciels. Il peut être utilisé pour définir les plans de travail, la structure et le comportement de systèmes, le design de hardware, etc.

Qu'est ce qui est unifier par UML ?

UML unifie :

Les notations et concepts orientés-objet existants

- UML fournit une définition «standard» de la terminologie et de la notation

les concepts et la notation utilisés au cours du cycle de développement

- aucune traduction des modèles n'est nécessaire, de l'étude des besoins jusqu'au déploiement

Les concepts, quel que soit le domaine d'application

- qu'il soit un système temps réel, complexe, distribué, grand, etc.
- des langages de modélisation spécialisés peuvent suppléer aux manques

Les concepts, quelles que soient les techniques d'implantation.

Ses propres concepts par leur généralisation

- UML est ainsi plus simple et plus facilement applicable

Les composants UML

Le modèle UML est construit

- sur la base d'éléments standards définis par le méta-modèle d'UML
- sur la base d'éléments particuliers créés par extension d'UML

Le modèle utilisateur est représenté par des diagrammes qui montrent

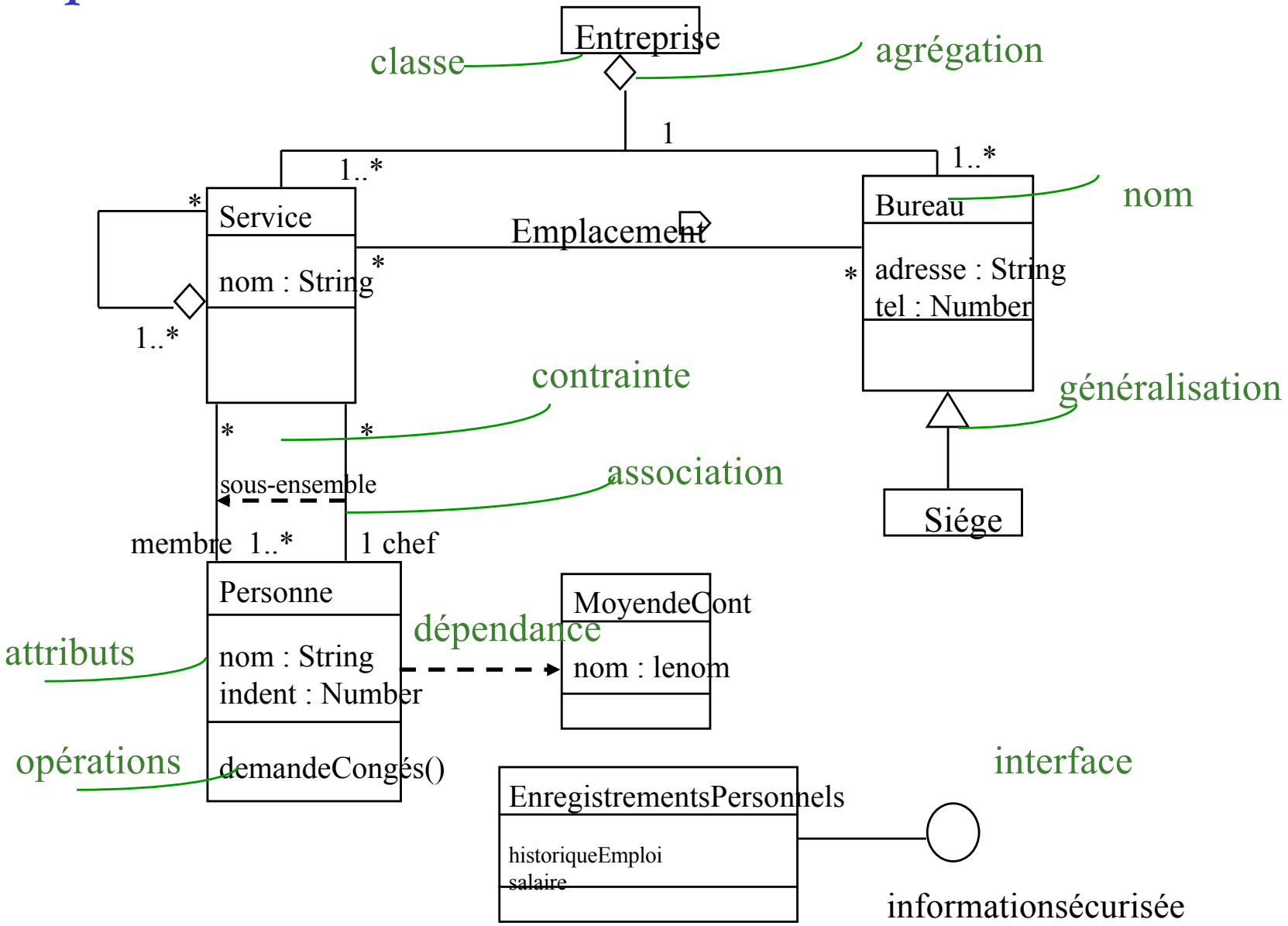
- la structure statique du système (vue logique) : diagramme des classes, diagrammes d'objets, diagrammes des cas d'utilisation)
- la dynamique du système : diagramme d'états, diagramme d'activités, diagramme de séquences, diagramme de collaboration
- la structure d'implantation du système (vue physique) : diagramme de composants et diagramme de déploiement
- l'organisation du modèle lui-modèle : diagramme de classes constitués de représentation de packages.

Que fait UML ?

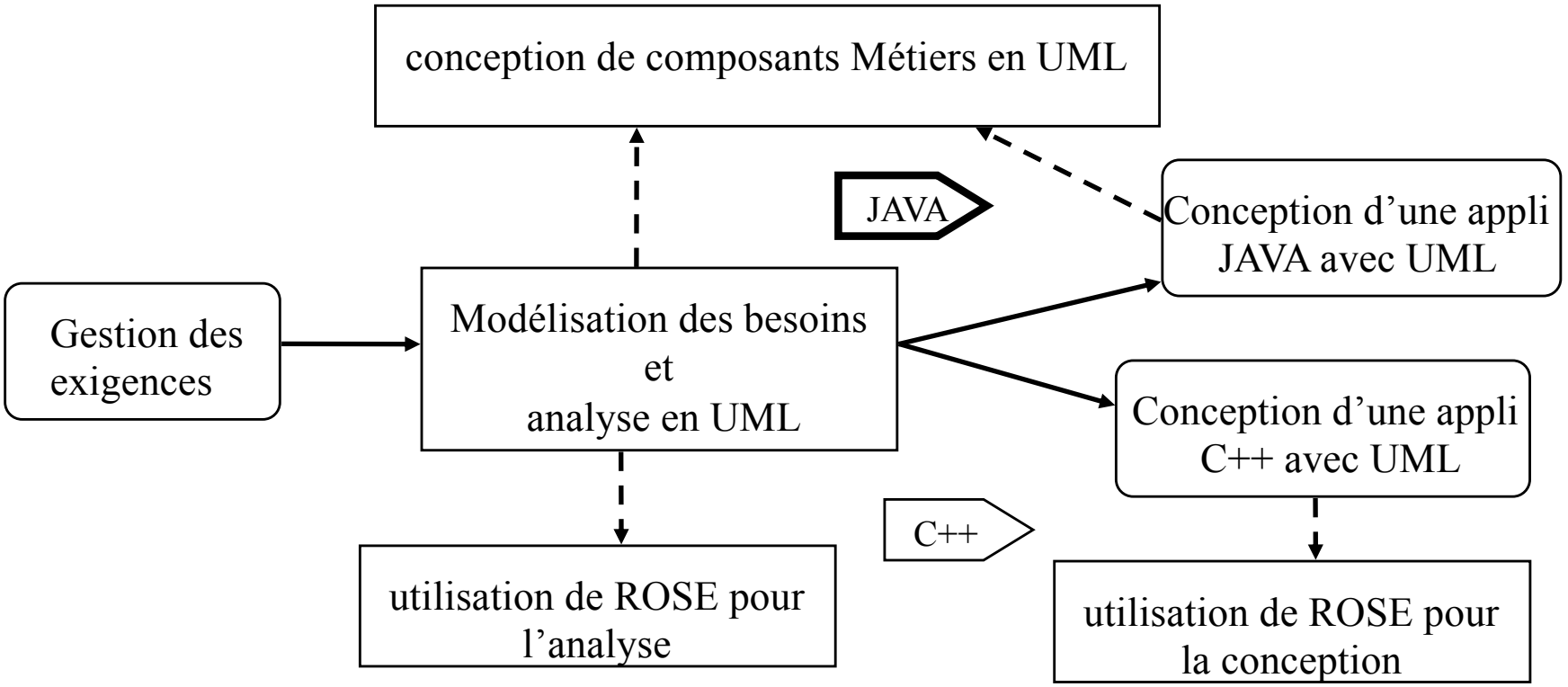
- Visualiser
- spécifier
- construire
- documenter

Les composants d'un système, et en ce qui nous concerne de systèmes à forte composantes logicielles.

Exemple de modèle UML



Ou situer UML par rapport aux langages de programmation ?



UML est un langage de modélisation

La devise de beaucoup de programmeurs est : ***penser le, programmer le***. Cette façon de faire présente beaucoup d'inconvénients :

- la diffusion de ces modèles conceptuels est très délicate pour ceux qui n'utilise pas le même langage;
- Il y a certain aspects du problème qui sont incompréhensible sans la construction d'un modèle qui transcende le programme;
- La maintenance de tel programme est difficile

UML tente de gommer ces inconvénients : un modèle explicite facilite la communication

UML est un langage de visualisation

Certains concepts sont mieux modélisés textuellement, d'autres le sont graphiquement. Dans beaucoup de systèmes, il y a des structures qui transcendent ce qui est représentable graphiquement.

- *UML est un langage graphique*

UML n'est pas uniquement une collection de symboles graphiques. Derrière chaque symbole de la notation UML se cache une définition sémantique précise.

De cette façon, un développeur peut écrire un modèle en UML, et un autre développeur (ou un autre outil) peut interpréter ce modèle sans ambiguïté.

UML est un langage de spécification

Dans ce contexte spécifier signifie : construire un modèle qui soit :

- précis
- Non ambiguë et
- Complet

UML permet la spécification et l'aide à la décision durant les phases d'analyse, de design et d'implantation.

UML est un langage de construction

UML n'est pas un langage de programmation. Il peut être connecté directement à divers langages de programmation (java, c++, base de données relationnelles…). Les aspects pouvant être mieux exprimer graphiquement sont décrits avec UML, les autres avec le langage choisi.

La correspondance entre UML et les langages de programmation permet la génération automatique de code. L'inverse est aussi possible (l'inversion est souvent assistée par des experts humains)

Quelques outils de modélisation UML

- ✓ Rational Rose/XDE : IBM
- ✓ Objecteering/UML : Objecteering Software
- ✓ Together/ModelMaker : Borland
- ✓ Poseidon UML : Gentleware
- ✓ Rhapsody Modeler : I-logix
- ✓ PowerAMC/PowerDesigner : Sybase
- ✓ Visio : Microsoft
- ✓ Win'design
- ✓ Plugin Omondo : Eclipse
- ✓ Umbrello UML Modeller (Paul Hensgen-kde)
- ✓ ClassBuilder
- ✓ ArgoUML
- ✓ TAU UML : Telelogic

Modèles UML

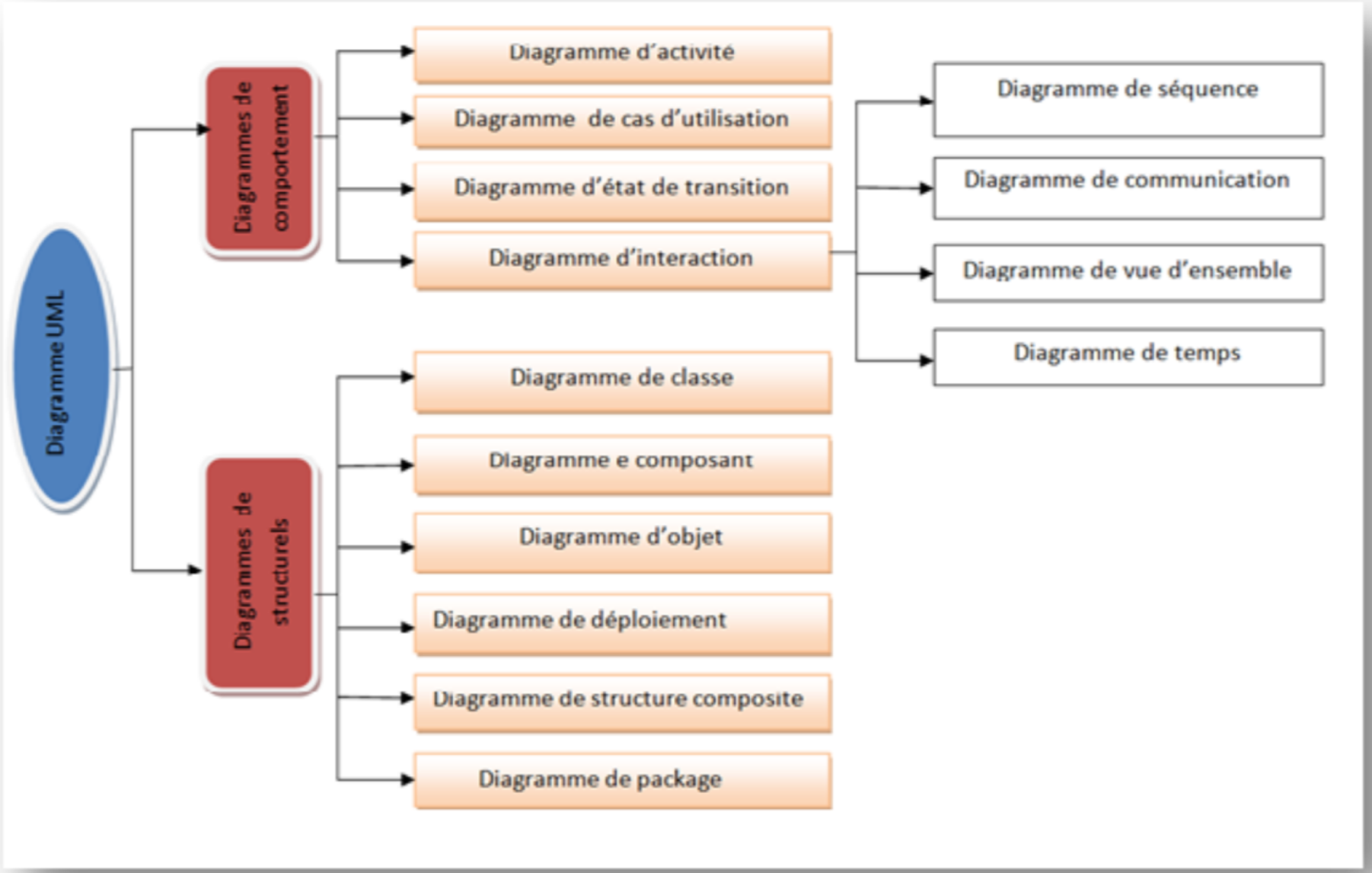


Diagramme UML
Diagrammes de comportement
Diagrammes de structurels
Diagramme d'activité
Diagramme de cas d'utilisation
Diagramme d'état de transition
Diagramme d'interaction
Diagramme de séquence
Diagramme de communication
Diagramme de vue d'ensemble
Diagramme de temps
Diagramme de classe
Diagramme e composant
Diagramme d'objet
Diagramme de déploiement
Diagramme de structure composite
Diagramme de package

Les cas d'utilisation

A.Osmani

Sommaire

1. Définition et intérêt
2. Diagramme de cas d'utilisation
3. Exemple
4. Eléments mis en jeu et leur notations
5. Description textuelle
6. Exemple complet

Définition et intérêt

Le développement d'un logiciel permet de répondre à un ensemble de besoins exprimés par l'utilisateur futur du système.

Par exemple, l'IUT demande un logiciel pour :

- Gérer les salles de td et de tp;
- Gérer les absences des étudiants
- Assurer le suivi des stages
- Etc.

Chacun des besoins exprimés par l'utilisateur futur du logiciel (que nous appelons **acteur** en UML) est un cas d'utilisation.

Un **acteur** est un utilisateur possible du système. Il est donc modéliser à l'extérieur du système.

Définition et intérêt

Un cas d'utilisation est un ensemble de séquences d'actions exécutées par le système pour rendre un service à un ou plusieurs acteurs.

Le cas d'utilisation :

- modélise un aspect dynamique du système.
- permet de visualiser le comportement d'un système ou d'un sous-système
- exprime les interactions entre les acteurs et le système (sauf exceptions)
- décrit le comportement attendu sans imposer le mode de réalisation
- Correspond à une fonction métier du système selon le point de vue des acteurs.

Faire attention : un cas d'utilisation

- ne décrit pas la manipulation de l'IHM (interface homme machine)
- ne décrit pas la gestion des problèmes matériels
- n'est pas une fonction
- ne doit pas se réduire à une seule séquence d'actions

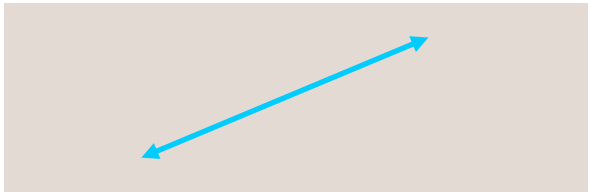
Diagramme des cas d'utilisation

C'est l'ensemble de cas d'utilisations et d'acteurs du système.

Le diagramme des cas d'utilisation :

- Introduit un découpage fonctionnel du système
- Permet une analyse des besoins et aide à démarrer l'analyse orientée objet
- Décrit exhaustivement les exigences fonctionnelles du système (décrit tout ce que le futur système doit faire sans spécifier comment le faire)

Remarque :
Le graphique ci-dessous ne fait pas parti du langage UML. Il me permet uniquement de pointer les éléments à expliquer.



Exemple de diagramme de cas d'utilisation

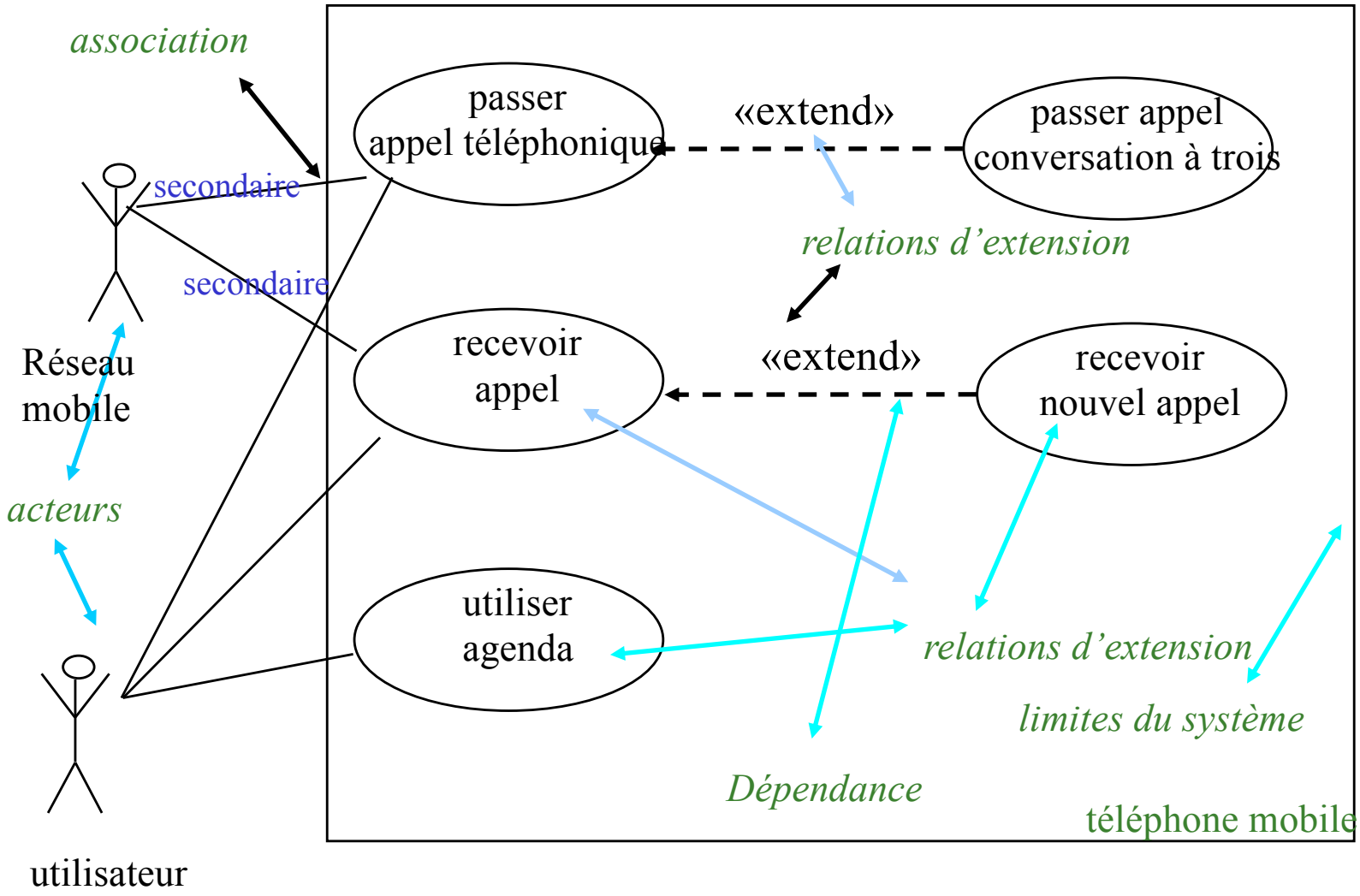


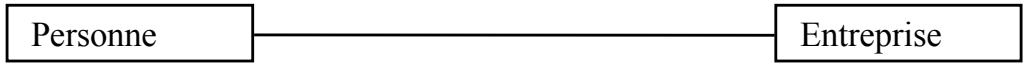
diagramme de cas d'utilisation - téléphone mobile

Relation association

C'est une relation structurelle entre objets. Une association est souvent utilisée pour agréger les liens.

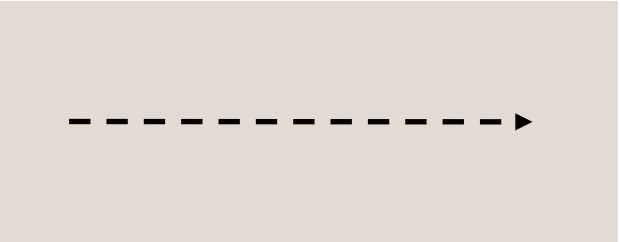
Représentation graphiquement d'une association « simplifiée » :

Exemple :

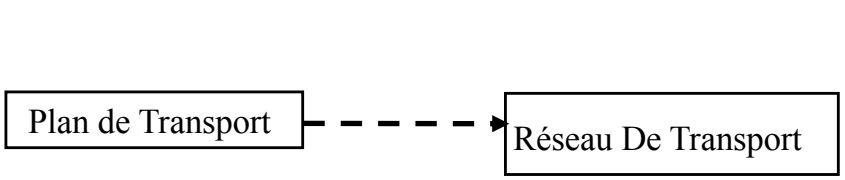


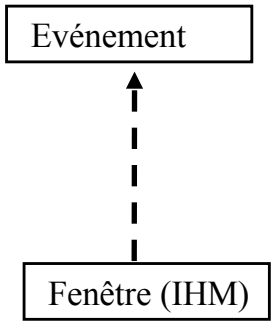
Relation de dépendance

La dépendance est une relation de causalité sémantique entre les objets. Un objet A dépend d'un objet B si le changement sémantique de A affecte celui de B.



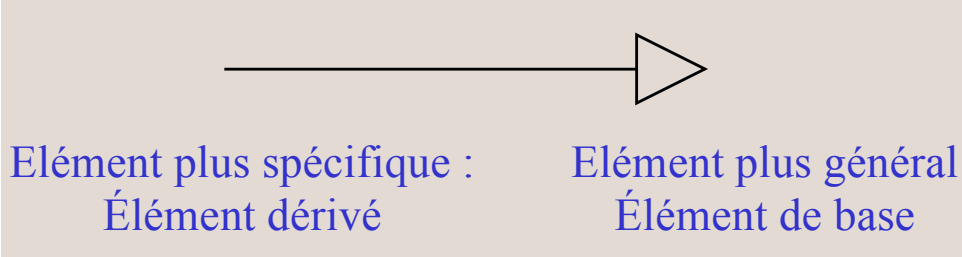
Exemples :



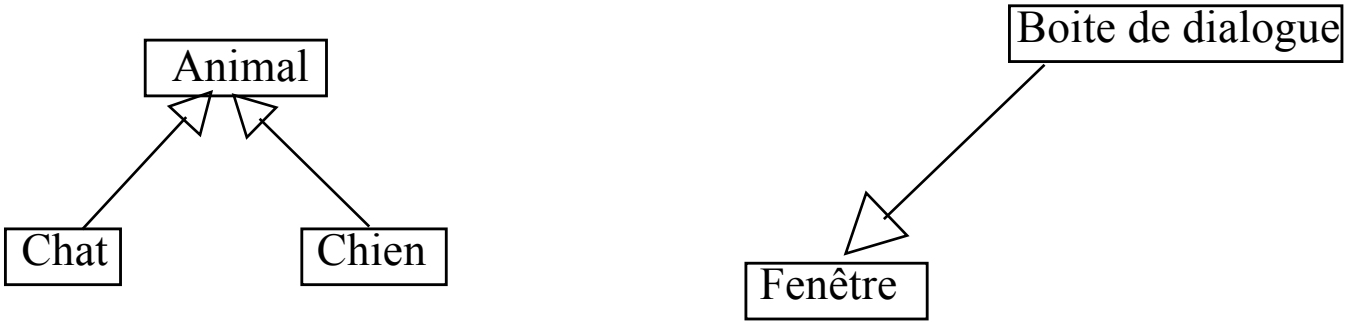


Relation d'héritage

C'est une relation de spécialisation/généralisation. Les éléments spécialisés héritent de la structure et du comportement des éléments plus généraux.



Exemple :

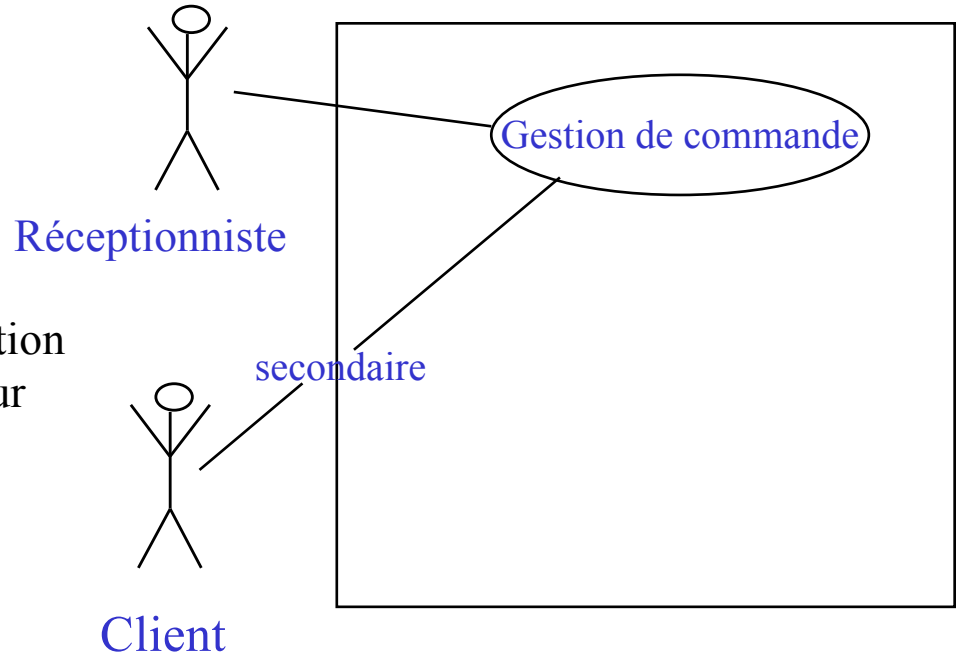


Notations à connaître

Les acteurs

Le cas d'utilisation comporte deux types d'acteurs :

- L'acteur principal pour lequel le cas d'utilisation apporte une plus-value métier. Il est obligatoire. Il est souvent le déclencheur du cas.
- Un ensemble d'acteurs secondaires.



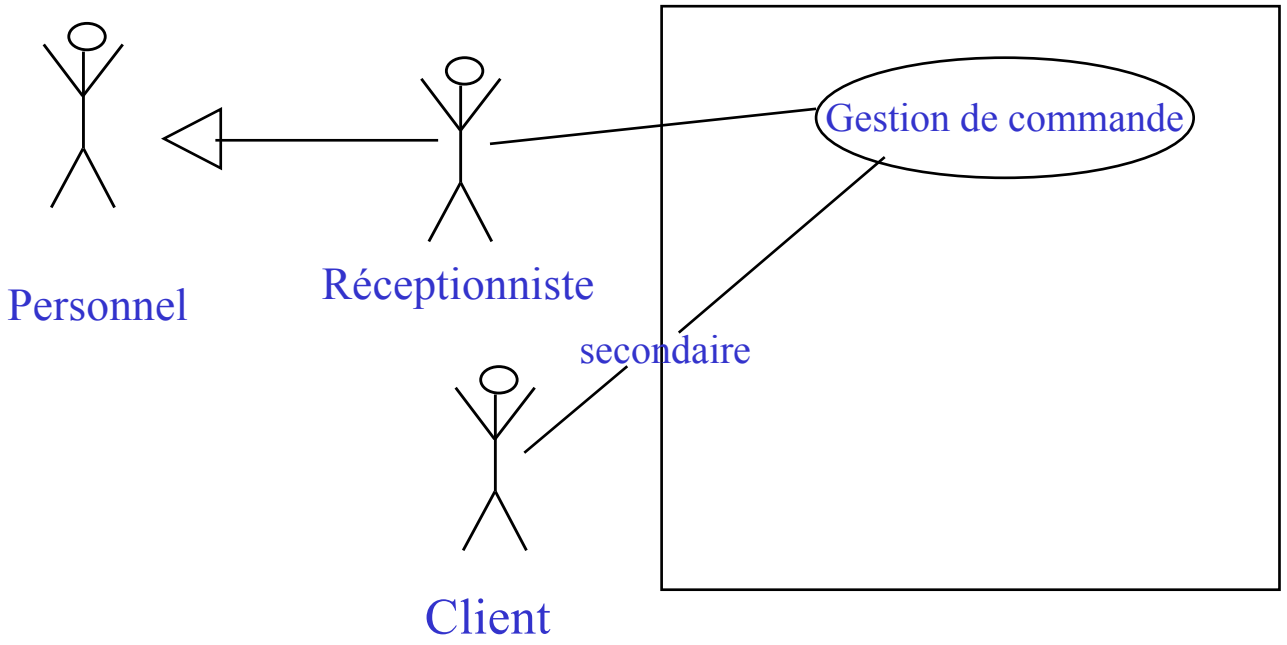
Convention : une association sans rôle indique un acteur principal, sinon préciser explicitement le le rôle

Notations à connaître

Les acteurs

Il est possible de spécialiser les acteurs en utilisant la relation d'héritage.

S'il est utile d'ajouter l'acteur Personnel dans la description du diagramme des cas d'utilisation alors il faut préciser que l'acteur réceptionniste « est un » Personnel Particulier.

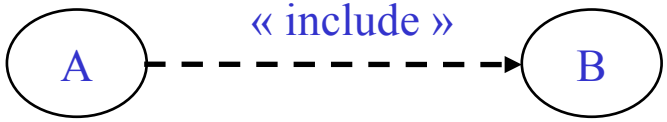


cas d'utilisation

Un cas d'utilisation est donné par le graphique suivant

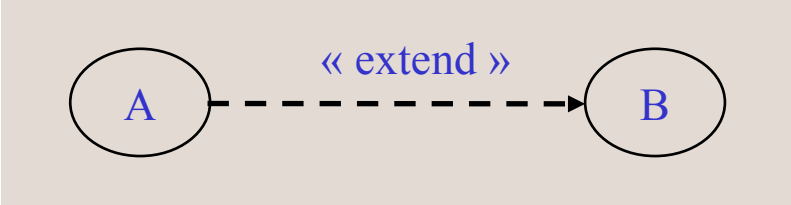
Parmi les relations standards entre les cas d'utilisation on distingue les trois relations suivantes :

1. « include » : le cas de base (A) utilise un autre cas (B)

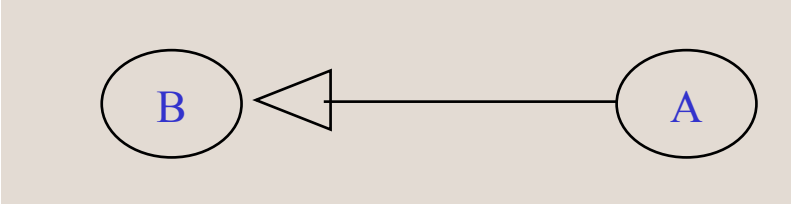


cas d'utilisation

2. « extend » : le cas d'utilisation (A) étend (optionnellement) le cas de base (B)



3. héritage le cas de base (A) est une spécialisation du cas (B)

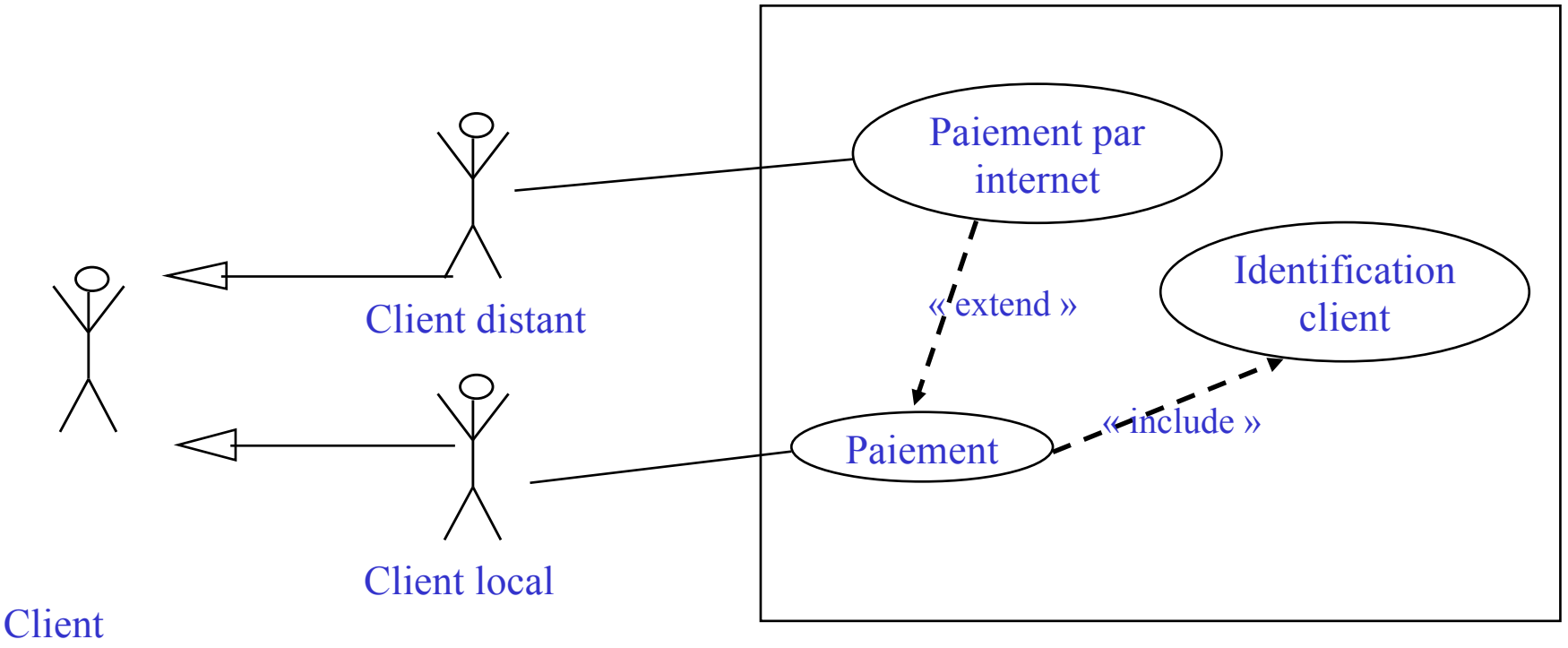


Client local

Notations à connaître

cas d'utilisation

Exemple :



Description textuelle d'un cas d'utilisation

Le diagramme de cas d'utilisation UML permet de visualiser tous les cas mais ne précise pas leur contenu.

Une description textuelle doit accompagner chaque cas d'utilisation. Il n'existe pas en UML de standard d'écriture des cas. La description suivante est rigoureuse et relativement complète :

Structure :

2 rubriques obligatoires

1. Identification : elle comporte le nom, le but, les acteurs, la date, les responsables et la version du cas
2. Les séquencements : ils décrivent l'ensemble des pré-conditions, les post-conditions, séquences nominales et les exceptions éventuelles.

Des rubriques optionnelles

1. Les contraintes non fonctionnelles : fiabilité, temps de réponse, confidentialité, fréquences, concurrence, etc.
2. Des indications sur l'interface homme-machine : ce qui doit appaître, dans quel ordre, etc.

Exemple complet

Application bancaire

Pour des raisons de simplicité nous réduisons les opérations bancaires aux :

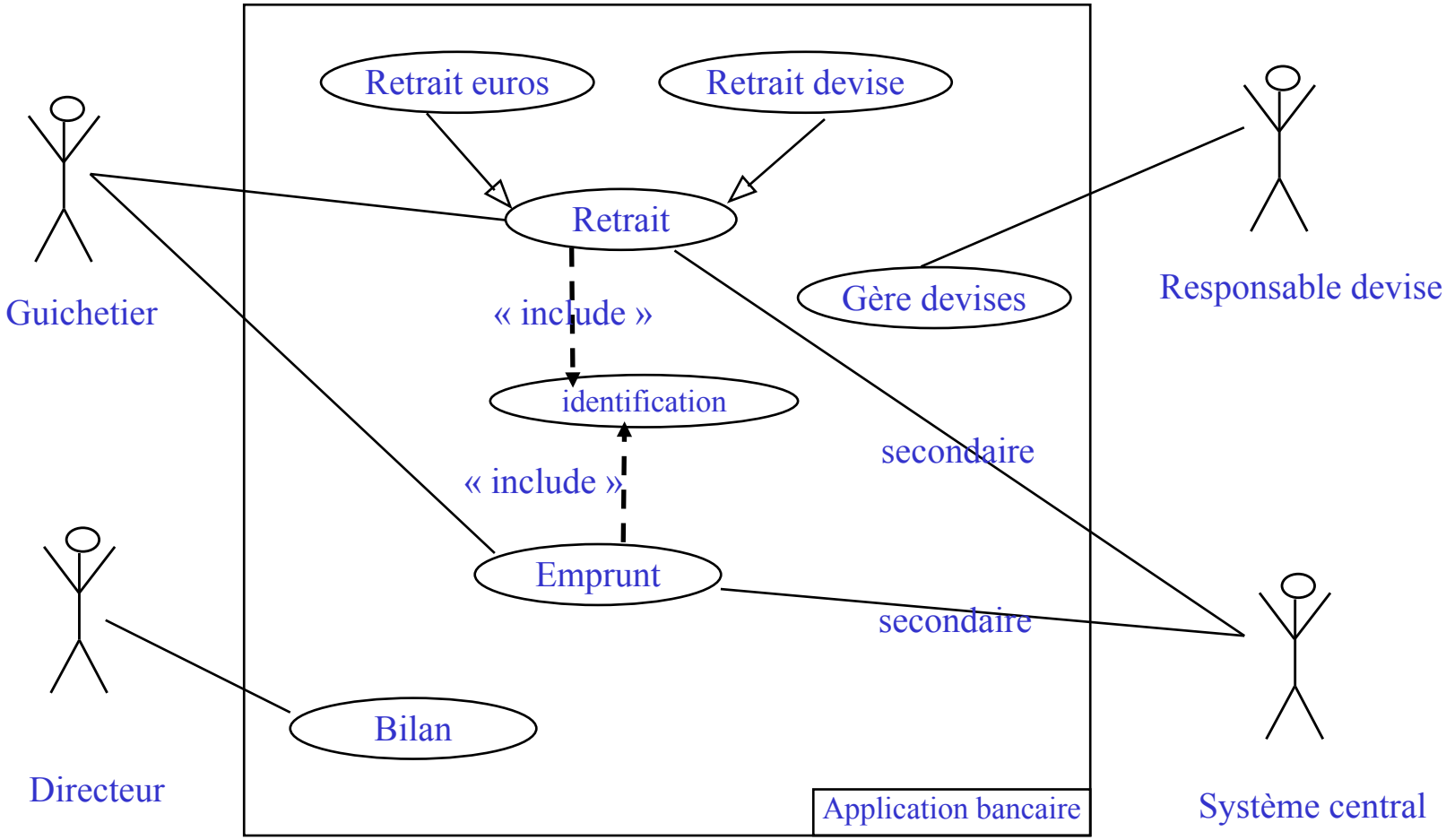
- Retrait d'espèces en euros ou en devises
- Effectuer des emprunts
- Effectuer un bilan
- Saisie des cours des devises

Les acteurs du système sont :

- Le guichetier
- Le directeur
- Le système central
- Le responsable des devises

NB : on sait qu'il est nécessaire de saisir le numéro de compte du client avant d'effectuer des opérations sur son compte.

Exemple complet : diagramme des cas d'utilisation



Exemple complet : description d'un cas

Cas d'utilisation : retrait d'espèces dans une agence

Supposons que les opérations effectuées par le guichetier pour réaliser l'opération de retrait se limite à :

- Le guichetier saisit le numéro de compte client
- l'application valide le compte auprès du système central
- l'application demande le type d'opération au guichetier
- le guichetier sélectionne un retrait d'espèces de 200 euros
- le système «guichet» interroge le système central pour s'assurer que le compte est suffisamment approvisionné
- le système central effectue le débit du compte
- le système notifie au guichetier qu'il peut délivrer le montant demandé

Exemple complet : description textuelle

Identification :

- **nom du cas** : retrait d'espèces en euros
- **but** : détaille les étapes permettant à un guichetier d'effectuer l'opération de retrait d'euros pour un client.
- **acteurs** : Guichetier, système central (secondaire)
- **date** : le 18/02/2002
- **responsables** : A.Osmani
- **version** : 1.0

Les séquencements :

Le cas d'utilisation commence lorsqu'un client demande le retrait d'espèces en euros

- **Pré-conditions**
  - Client possède un compte (donne son numéro de compte)
- **Enchaînements**
  - Le guichetier saisit le numéro de compte client
  - l'application valide le compte auprès du système central
  - l'application demande le type d'opération au guichetier
  - le guichetier sélectionne un retrait d'espèces de 200 euros

Exemple complet : description textuelle

Enchaînements (suite)

- le système «guichet» interroge le système central pour s'assurer que le compte est suffisamment approvisionné
- le système central effectue le débit du compte
- le système notifie au guichetier qu'il peut délivrer le montant demandé

- **Post-conditions**
  - Le guichetier ferme le compte
  - Le client récupère l'argent

Rubriques optionnelles

- **Contraintes non fonctionnelles**
  - fiabilité : les accès doivent être extrêmement sécurisés
  - Confidentialité : les informations concernant le client ne doivent pas être divulguer
- **Contraintes liées à l'interface homme-machine :**
  - Donner la possibilité d'accéder aux autres comptes du client
  - Ne pas accéder à plusieurs comptes au même temps
  - Toujours demander la validation des opérations de retrait