Merise & UML

Conception, Spécification & Réalisation de SI

(Alexandre.Saidi@ec-lyon.fr)
Ecole Centrale de Lyon

2007-2008

Plan

Merise et UML

- Première partie Consacrée à Merise.
- **Seconde partie Consacrée à UML.**

Quelques références bibliographiques :

- "MERISE, 60 AFFAIRES CLASSÉES", *Michel DIVINÉ*, Éditions Eyrolles 1990
- Notes Fabrice Jouanot (imag)
- "UML par la pratique", *P. Roques*
- "Merise et UML", *J.Gabay*, Dunod 2004
- "Modélisation Objet avec UML", *P-A. Muller*, Eyrolles, 1997
- Notes personnelles

Ici, partie UML

- Présentation Diagramme des Classes UML.....

Rappel Sommaire

GL/SI : rappels cycles de vie

- Cycle de Vie en V

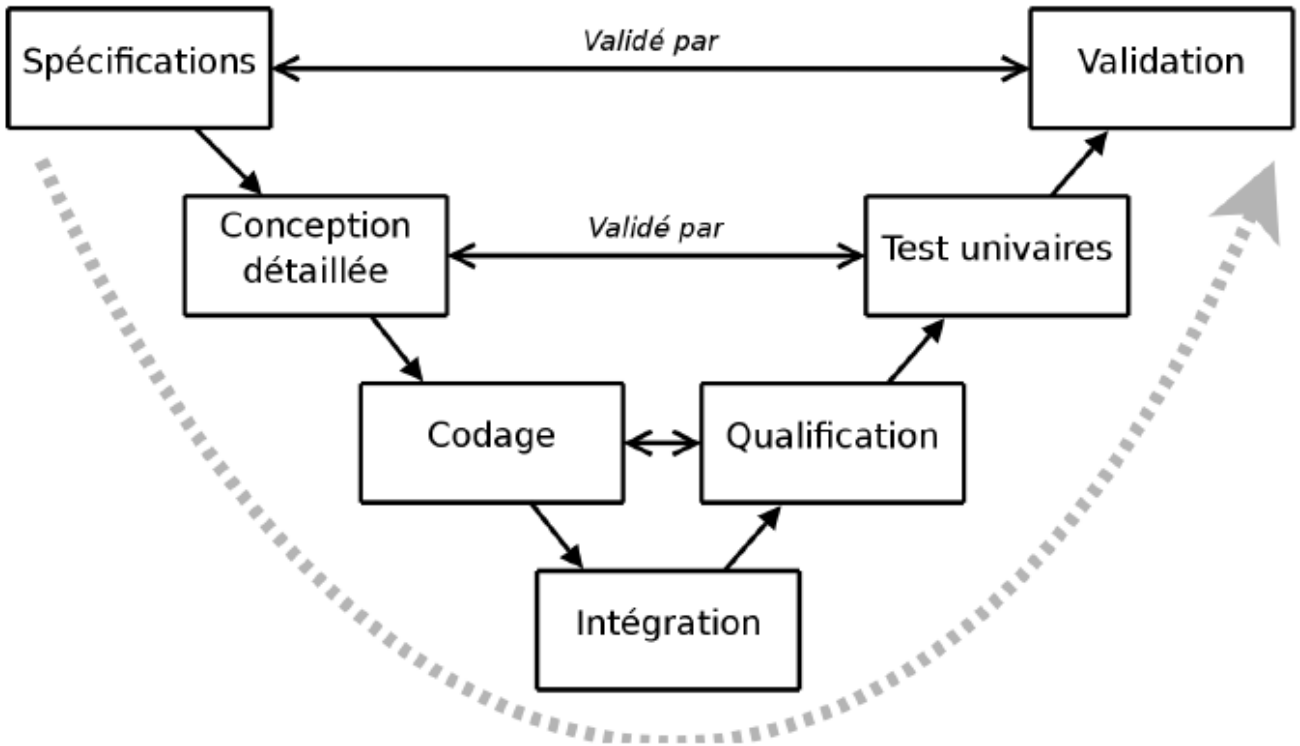


Figure: Cycle de Vie en V

GL/SI : rappels cycles de vie (suite)

- Cycle de Vie en cascade

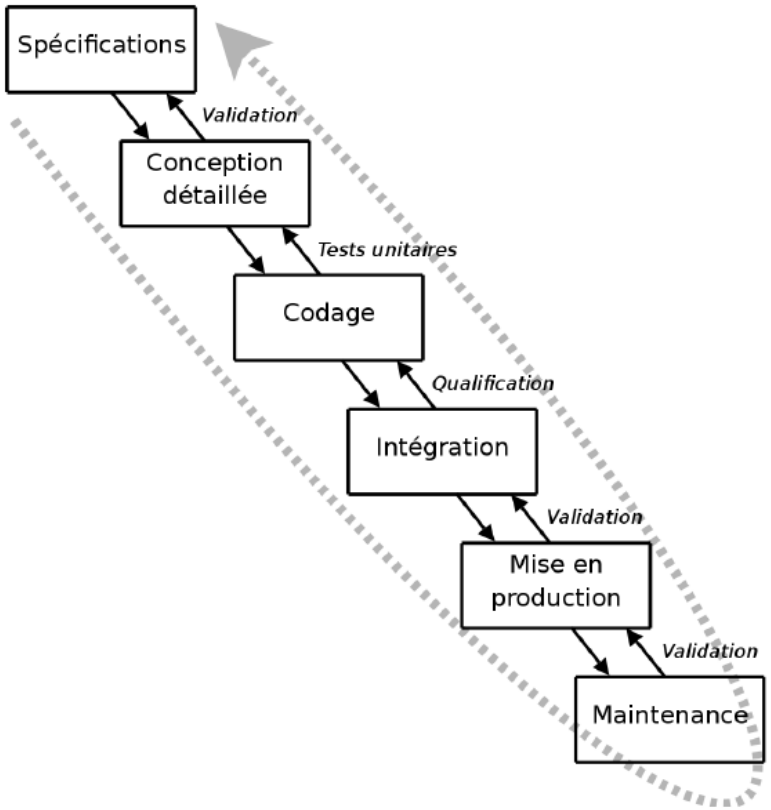


Figure: Cycle de Vie en cascade

UML et la AOO

UNE METHODE ORIENTEE-OBJET en 2 phases

Phase Initialisation

1. Définition de l'étendue du projet
2. Poser le problème : Quel est le système dont l'utilisateur a besoin ?
3. Identifier les acteurs : utilisateurs, gestionnaires du système, plates-formes, interfaces avec autres applications
4. Identifier les cas d'utilisations
5. Préciser les **cas d'utilisation** les plus importants, en fonction du risque, sous forme de scénarios (itération sur les scénarios)

Phase Analyse

1. Analyse du domaine : Modèle Objet statique
2. Scénarios (Diagramme de séquences)
3. Définition de l'Architecture
4. L'architecture est la façon d'organiser les objets pour qu'ils réalisent les fonctionnalités de l'application par leurs collaborations
5. Planification du Développement

- La qualité d'un logiciel est celle de son architecture

UML et la AOO (suite)

Une DEMARCHE possible

1. Analyse du domaine par Analyse textuelle : classes et relations
2. Objets complexes
3. Recherche des agrégation
4. Généralisation
5. Spécialisation
6. Règle de l'utilisation de l'héritage

UML et la AOO (suite)

Exemple : inscription aux cours d'une université

• Les professeurs doivent pouvoir renseigner le système sur les cours qu'ils assurent. Ils doivent aussi savoir quels étudiants se sont enregistrés à leurs cours.

• Lors de chaque début de semestre, il est défini une période de temps où les étudiants peuvent modifier leurs choix. Les étudiants doivent pouvoir accéder au système pour modifier leurs choix. Pendant cette période de temps, le système d'inscription doit créditer les étudiants pour les cours qu'ils ont annulés

Phase d'Initialisation

- Définition des risques.
  - Le risque principal est constitué par l'interface (bonne acceptation par les étudiants et professeurs).
- Identifier les acteurs.
  - Etudiant.
  - Professeur.
  - Employé du service enregistrement.
  - Système de facturation des droits d'inscription (externe).

UML et la AOO (suite)

Ebauche Diagramme de cas d'utilisation (use-case)

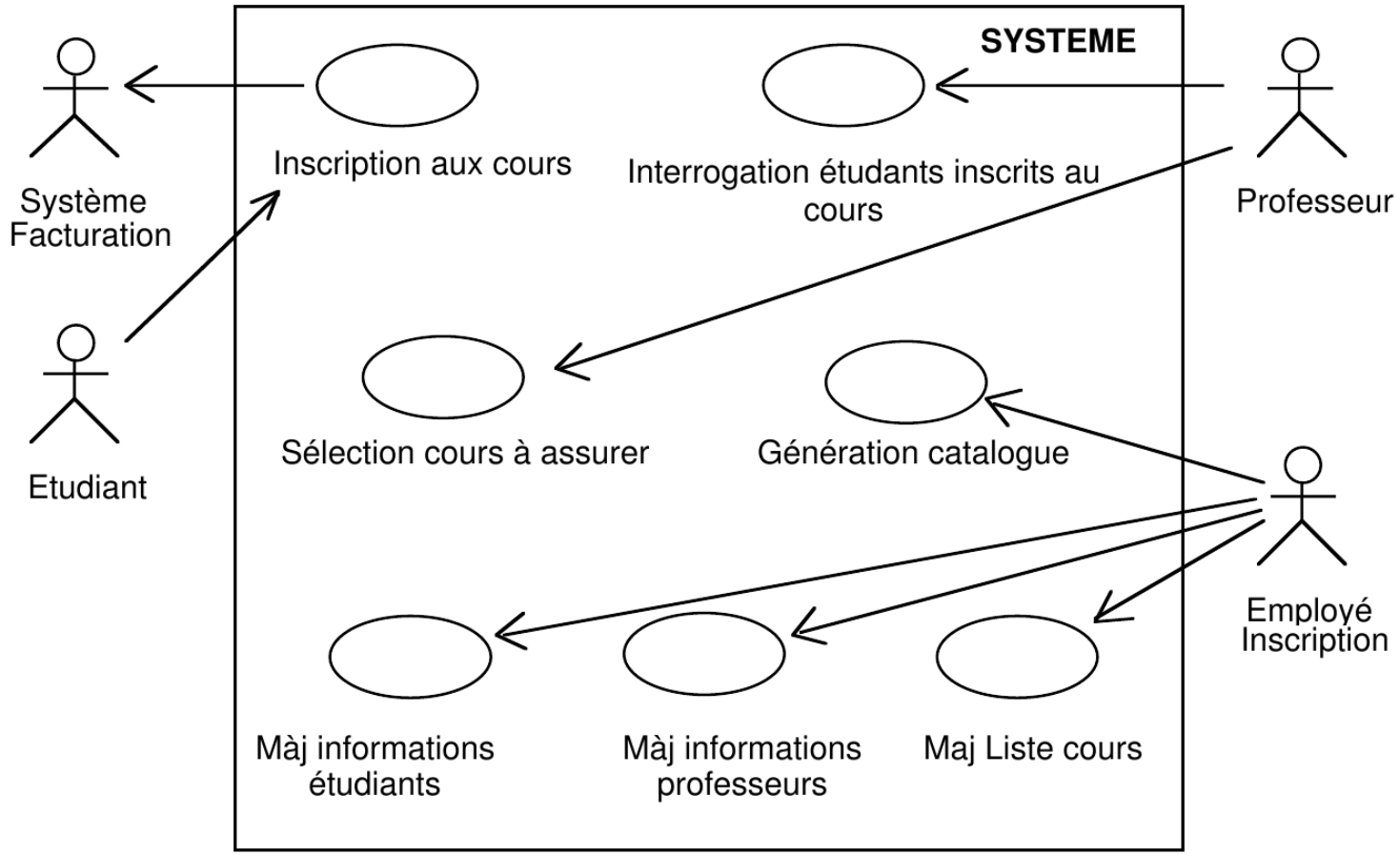


Figure: Diagramme des cas d'utilisation

UML et la AOO (suite)

Phase Analyse Objet : rappels via un exemple

Décomposer les attributs non-atomiques d'un objet

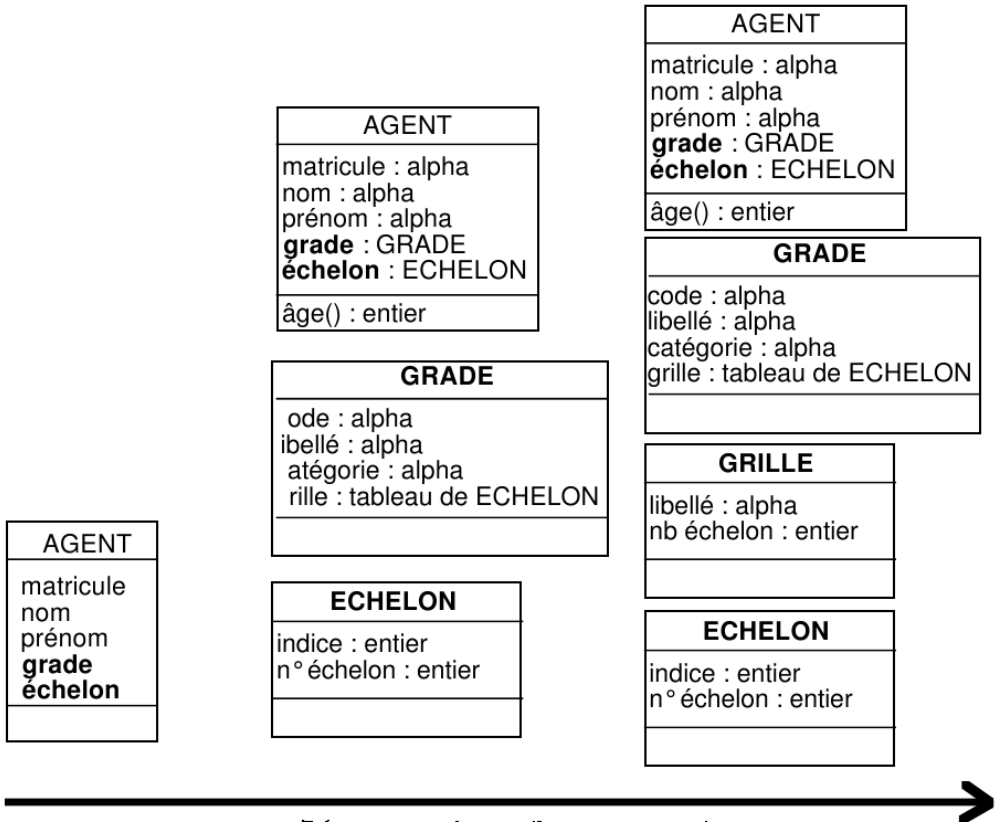
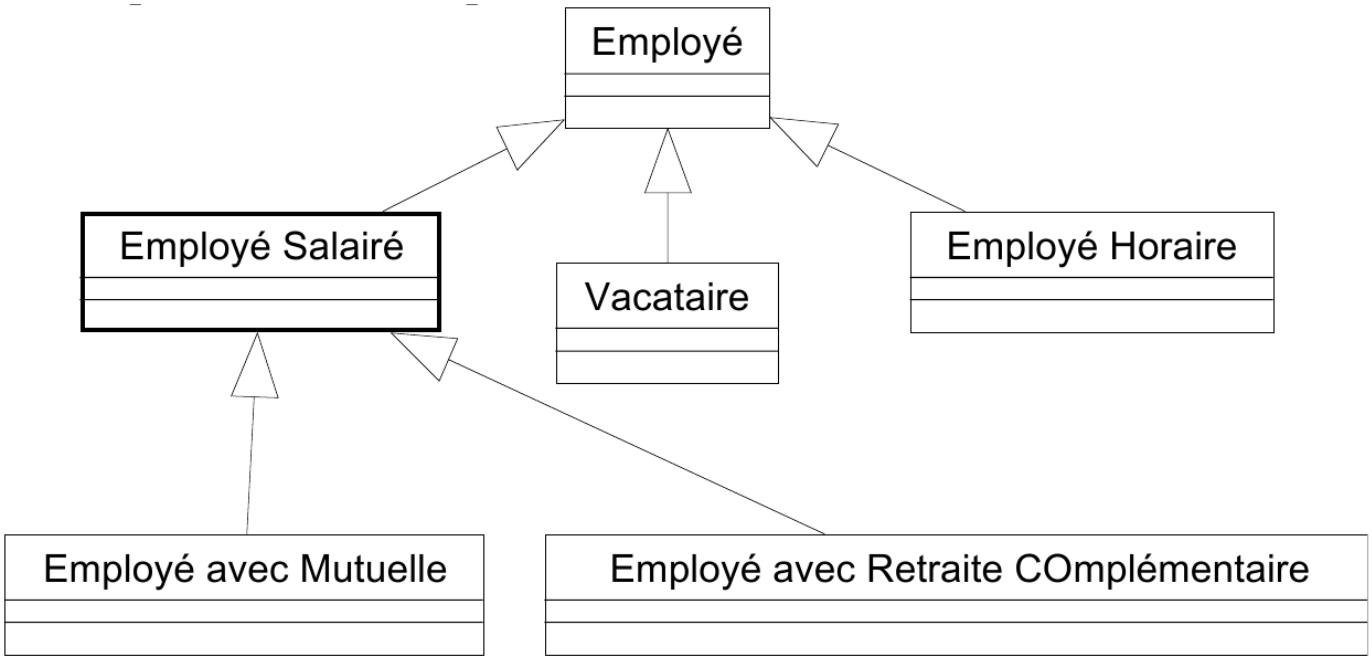


Figure: Décomposer les attributs non-atomiques

UML et la AOO (suite)

REGLES D'UTILISATION DE L'HERITAGE :

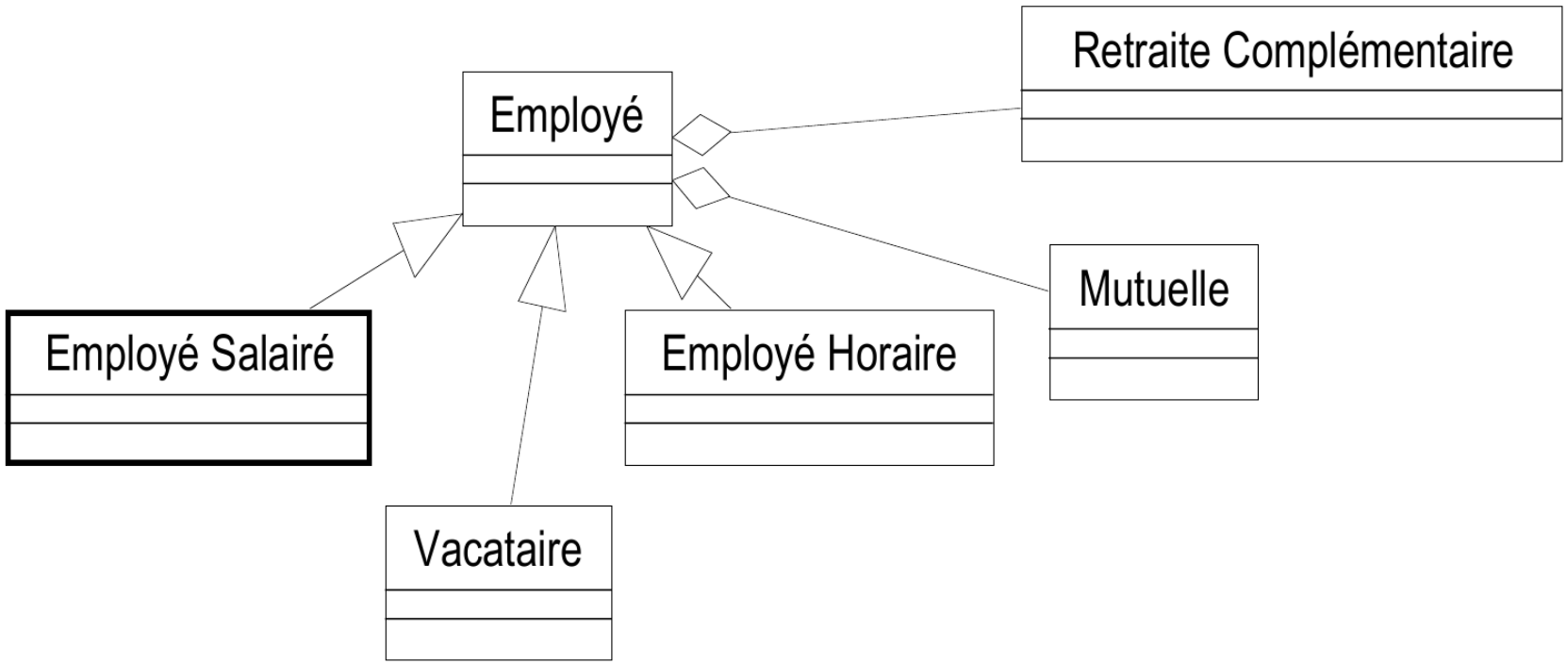
- Il y a un problème d'utilisation de l'héritage lorsque :
- Il y a un nombre important de sous-classes possibles (≥ 5), ou la définition des spécialisations n'est pas un invariant du domaine (elle sera sujette à des remises en cause).
- Une technique à utiliser est celle de la délégation.
- Croisement des "points de vue" de spécialisation



UML et la AOO (suite)

REGLES D'UTILISATION DE L'HERITAGE :

- Hériter de la classe la plus importante et déléguer le reste :
- Cette approche préserve l'identité de l'héritage à travers une spécialisation.



UML et les aspects dynamiques de l'analyse

UML : Modéliser la Dynamique

• Le contrôle est l'aspect dynamique du système qui décrit le séquencement des opérations, activé par des stimuli internes ou externes, sans tenir compte de l'activité de ces opérations ou de leur champ d'action

• Le modèle dynamique **permet d'examiner le comportement des objets**, et les modifications d'états des objets suite aux réceptions de messages

• En phase d'analyse, les messages échangés entre objets sont vus comme des événements

• Un événement est une transmission d'information (une communication), d'un acteur vers un objet du système, ou d'un objet du système vers un autre objet.

UML et les aspects dynamiques de l'analyse (suite)

Scénarii

• Un scénario est une séquence d'événements se déroulant suivant un cas typique d'exécution du système, lors d'un cas d'utilisation

• La portée d'un scénario peut varier : Il peut inclure
  - tous les événements qui se produisent lors d'un cas d'exécution,
  - ou seulement ceux produits par certains objets

• UML modélise la dynamique sous la forme de deux diagrammes:
  - **Diagramme de séquences** (pour chaque scénario)
  - **Diagrammes d'états** (pour chaque classe d'objet actif)

UML et les aspects dynamiques de l'analyse (suite)

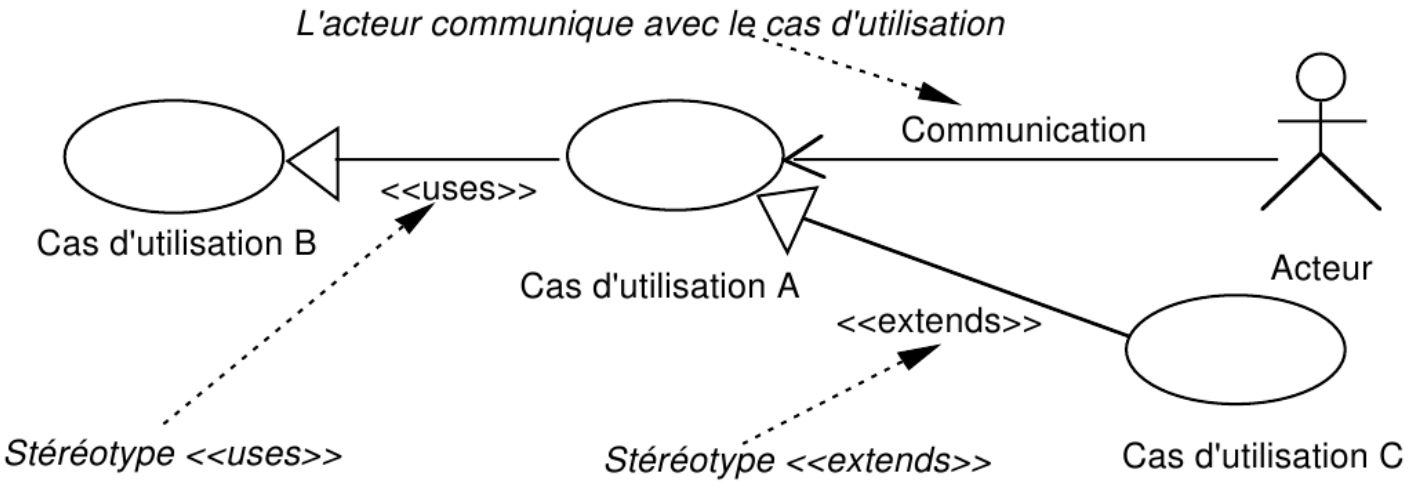
Diagramme de Cas d'Utilisation : détails

• Chaque scénario définit un comportement prototypal pour sa classe "cas d'utilisation".

• Un cas d'utilisation donné est caractérisé par un ensemble de scénarios

• Un cas d'utilisation peut contenir des diagrammes de cas d'utilisation, formant ainsi une arborescence correspondante à l'arborescence du menu de l'application

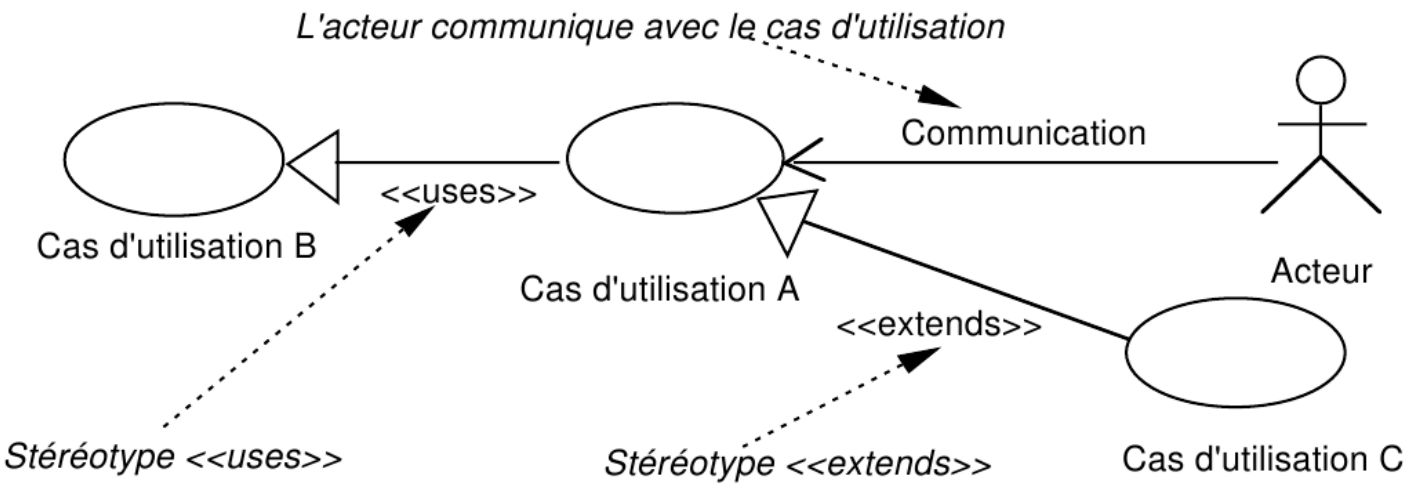
• Les fonctionnalités élémentaires classiques (ajouter un item, modifier un item, supprimer un item) correspondront aux scénarii contenus dans un cas d'utilisation



UML et les aspects dynamiques de l'analyse (suite)

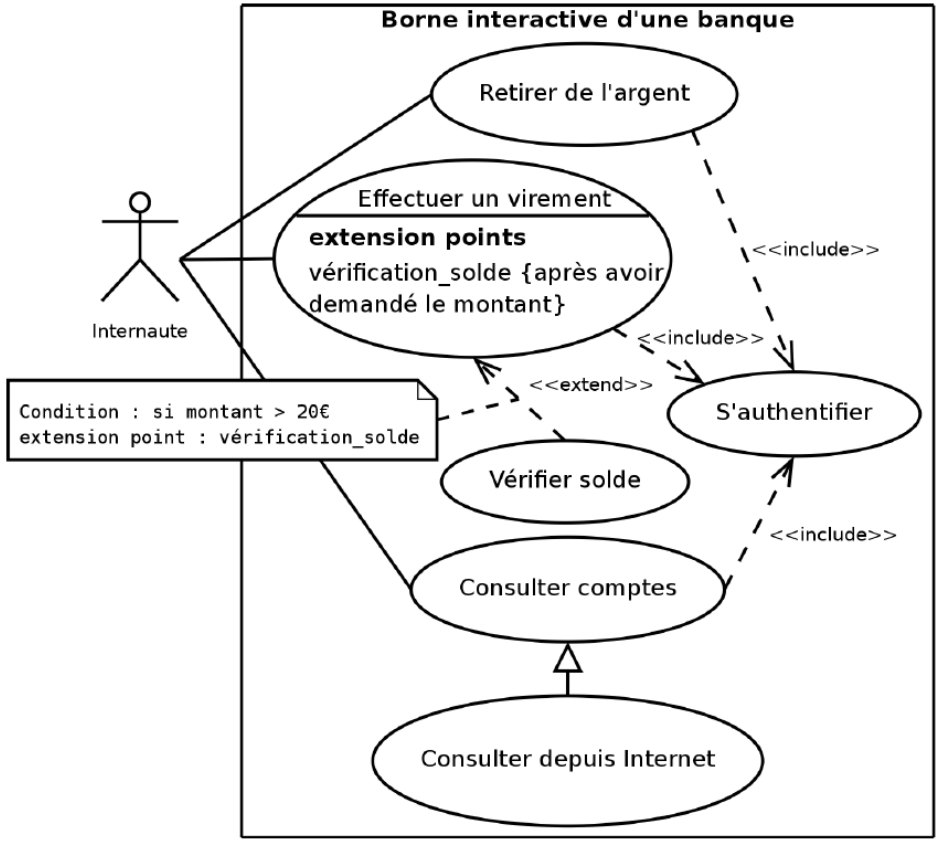
• «**extends**» désigne une relation d'extention entre un cas d'utilisation A et un cas d'utilisation B indique qu'une instance de B peut inclure (en fonction de **conditions** spécifiées dans l'extension) le comportement définit par A

• «**uses**» désigne une relation d'utilisation entre un cas d'utilisation A et un cas d'utilisation B indique qu'une instance de A inclus aussi le comportement spécifié par B



UML et les aspects dynamiques de l'analyse (suite)

Un autre exemple de Use case : GAB



- «**include**» désigne une inclusion d'un cas d'utilisation dans un autre.
- On note l'héritage dans les utilisations, ainsi que la condition de «**extends**»

UML et les aspects dynamiques de l'analyse (suite)

Evénement

• Les valeurs des informations échangées sont les paramètres de l'événement.

• Les événements sans paramètres s'illustrent d'eux-mêmes. L'information échangée est le fait qu'il ait été déclenché, tel un signal

• Un événement, ou transition d'état, est une façon pratique d'identifier une action par un nom donné à son résultat

Exemple :

"vol départ(compagnie, n°, ville)"

Cet événement est le résultat d'une action (complexe).
Ses paramètres qualifient le résultat de cette action.

UML et les aspects dynamiques de l'analyse (suite)

Diagramme de séquences

• Les scénarios sont représentés sous forme de Diagrammes de Séquences.

• Ce diagramme représente chaque objet par une ligne verticale, et chaque événement par une flèche horizontale reliant l'objet émetteur à l'objet récepteur.

• Le temps s'écoule du haut vers le bas sur la figure, mais la largeur des espaces horizontaux entre deux flèches ne sont pas significatifs; seule les séquences d'événements sont représentées, non le temps qui les sépare.

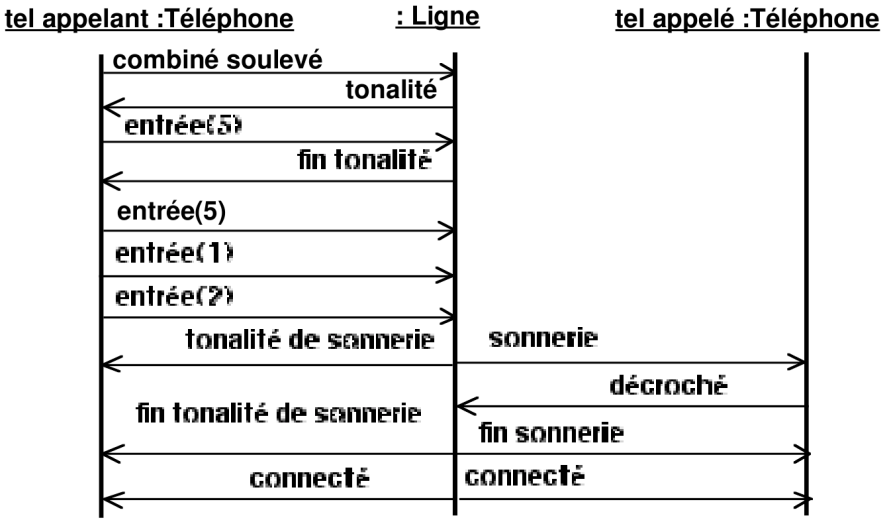
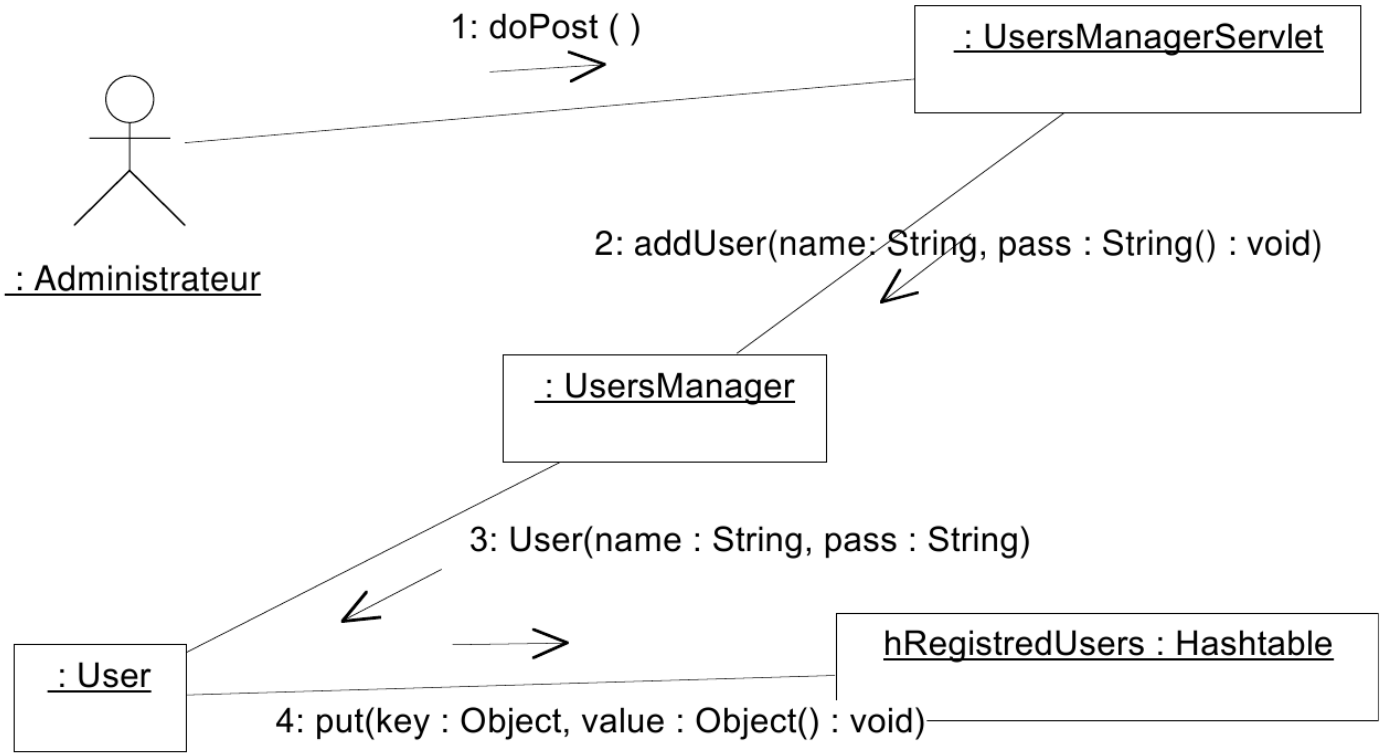


Figure: Diagramme de séquences

UML et les aspects dynamiques de l'analyse (suite)

Diagramme de Collaboration

• Les *diagrammes d'instances* (ou diagrammes de collaboration) d'UML notent les échanges de message.



UML et les aspects dynamiques de l'analyse (suite)

Etats :

Critère pour définir un état :

• La réponse d'un objet à un événement variera suivant son état, et, pour chaque événement reçu, l'objet change d'état.

• Un état correspond donc à l'intervalle de temps entre deux événements reçus par un objet.

• Au cours du temps, les objets se stimulent les uns les autres. Le stimulus d'un objet à un autre est un événement.

• Quand un objet reçoit un événement (un message), il change d'état. A un événement correspond donc une transition d'état.

• La réponse de l'objet à l'événement dépend de son état. Il peut envoyer un autre événement à un objet.

• Les événements représentent un point sur l'axe du temps, et les états représentent un *intervalle* sur cet axe.

UML et les aspects dynamiques de l'analyse (suite)

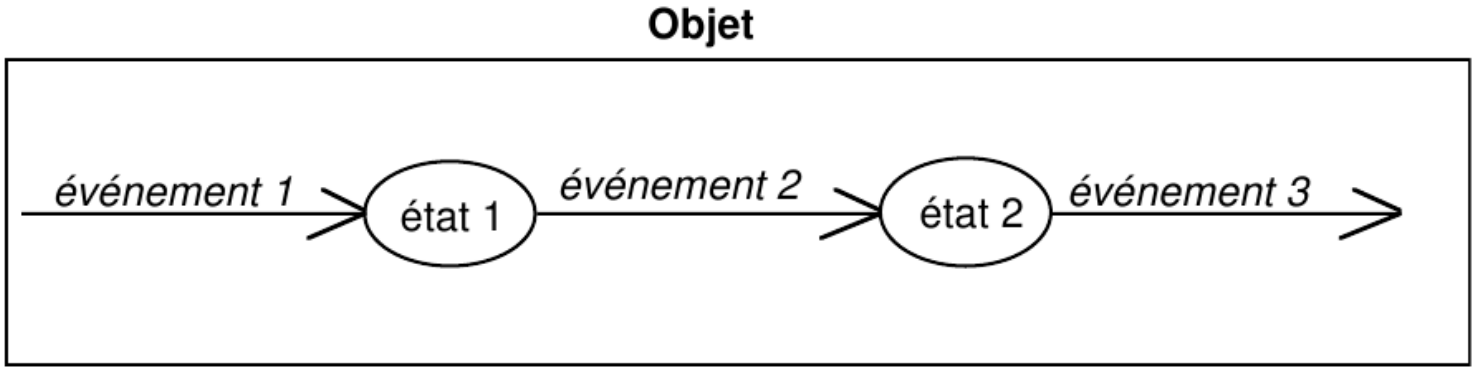


Figure: Les états d'un objet

• La définition des événements et états dépend du point de vue d'observation et du but dans lequel on modélise

UML et les aspects dynamiques de l'analyse (suite)

Diagramme d'états

• L'organisation des événements, des états et des transitions d'états pour une classe donnée est abstraite dans un diagramme d'états.

• Un diagramme d'états est un réseau d'états et d'événements.

• Dans un diagramme d'état, les événements sont vus sous forme de transitions

• L'organisation des états et des transitions d'états pour une classe donnée est abstraite dans un diagramme d'états.

• Un diagramme d'état montre les relations entre événements et états.

• Quand un événement est reçu, l'état suivant de l'objet dépend de l'état courant et de l'événement reçu.

• Un diagramme d'états est un graphe dont les oeuds sont les états, et les arcs les événements.

UML et les aspects dynamiques de l'analyse (suite)

• Un diagramme d'états décrit le comportement d'une seule classe d'objets.

• Chaque machine à états (diagramme d'état) s'exécute concurremment et peut changer d'état de façon indépendante.

• Les diagrammes des différentes classes s'unissent en un seul modèle par l'intermédiaire des événements partagés.

• Les événements déclenchent une ou des actions de l'objet récepteur.

• La modélisation dynamique doit spécifier ce que fait l'objet (l'action réalisée) en réponse aux événements.

• Le diagramme d'état doit donc aussi indiquer les actions réalisées par les objets en réponse aux événements (lorsqu'ils passent dans un état donné).

UML et les aspects dynamiques de l'analyse (suite)

Exemple de diagrammes d'état en UML :

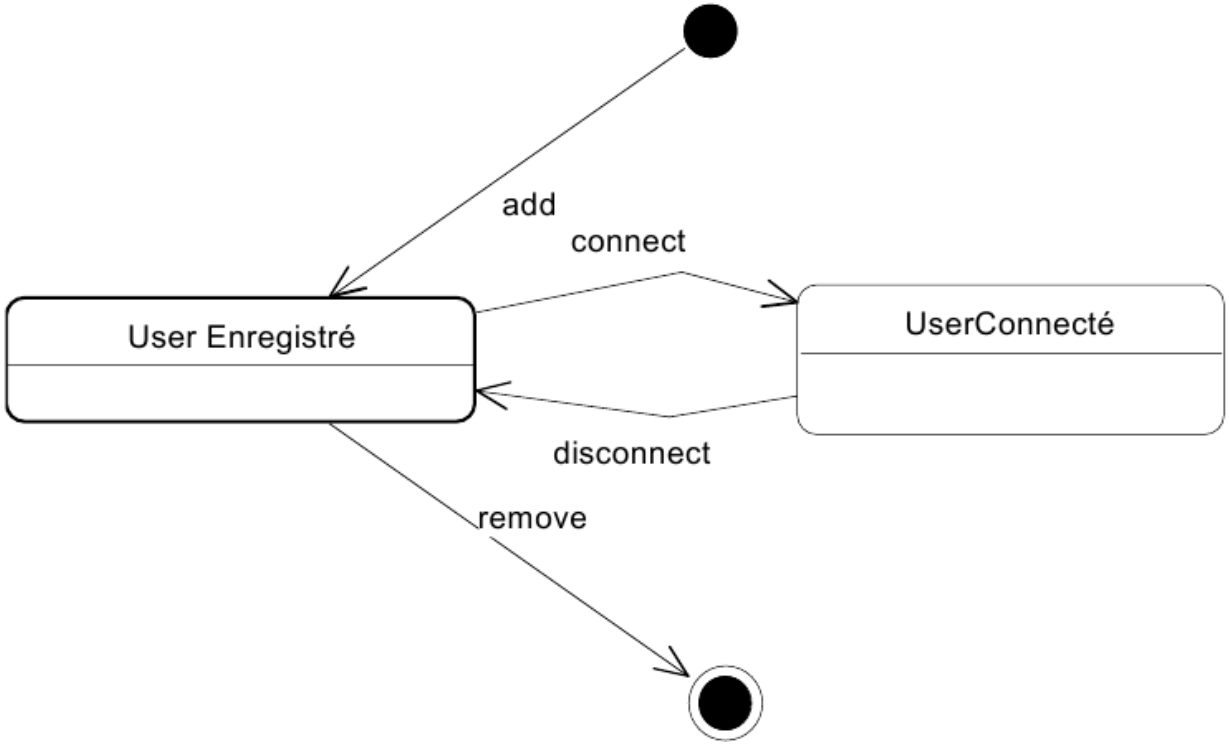
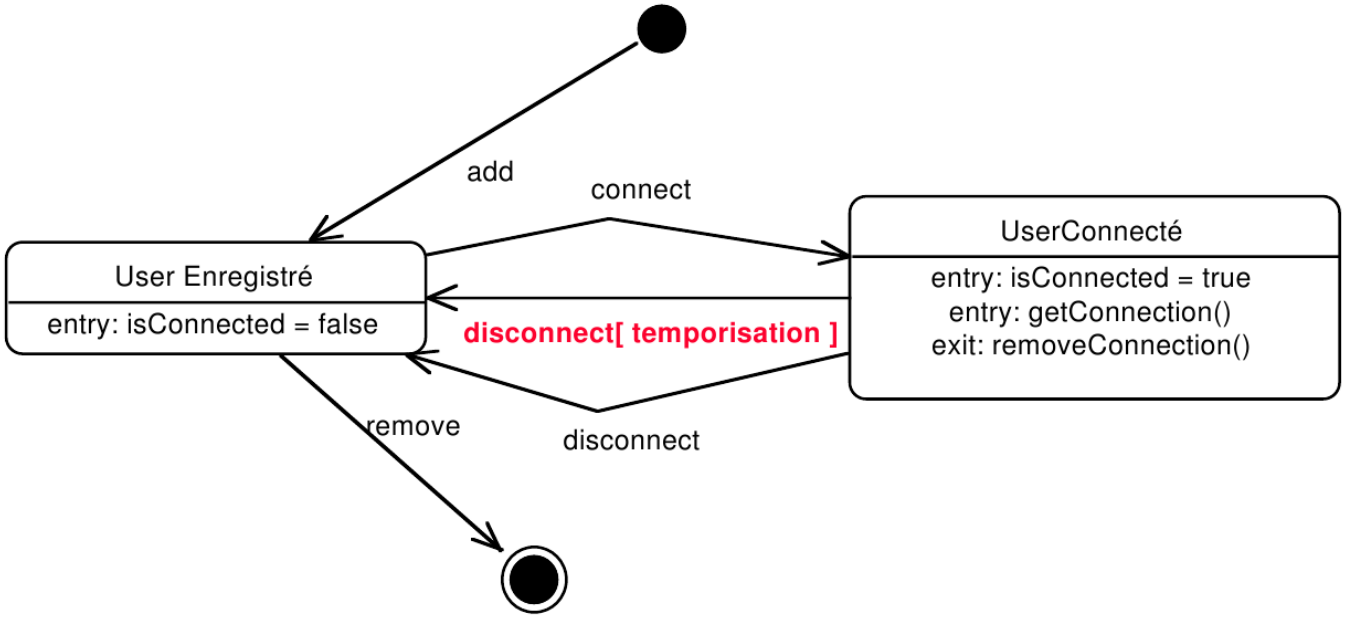


Figure: Diagramme d'états de la classe User

UML et les aspects dynamiques de l'analyse (suite)

Généralisation d'événements

Exemple : au bout d'un certain temps (temporisation) un utilisateur non actif est déconnecté



- Un arc sans nom d'événement indique une transition automatique.
- Une seule transition automatique par état source
  Une transition automatique est une transition gardée.

UML et les aspects dynamiques de l'analyse (suite)

Les opérations dans un diagramme d'états

- Les événements déclenchent une ou des actions de l'objet récepteur
- La modélisation dyn. spécifie l'action réalisée par l'objet en réponse aux évts.
- Le diagramme d'état doit donc aussi indiquer les actions réalisées par les objets en réponse aux événements (lorsqu'ils passent dans un état donné)

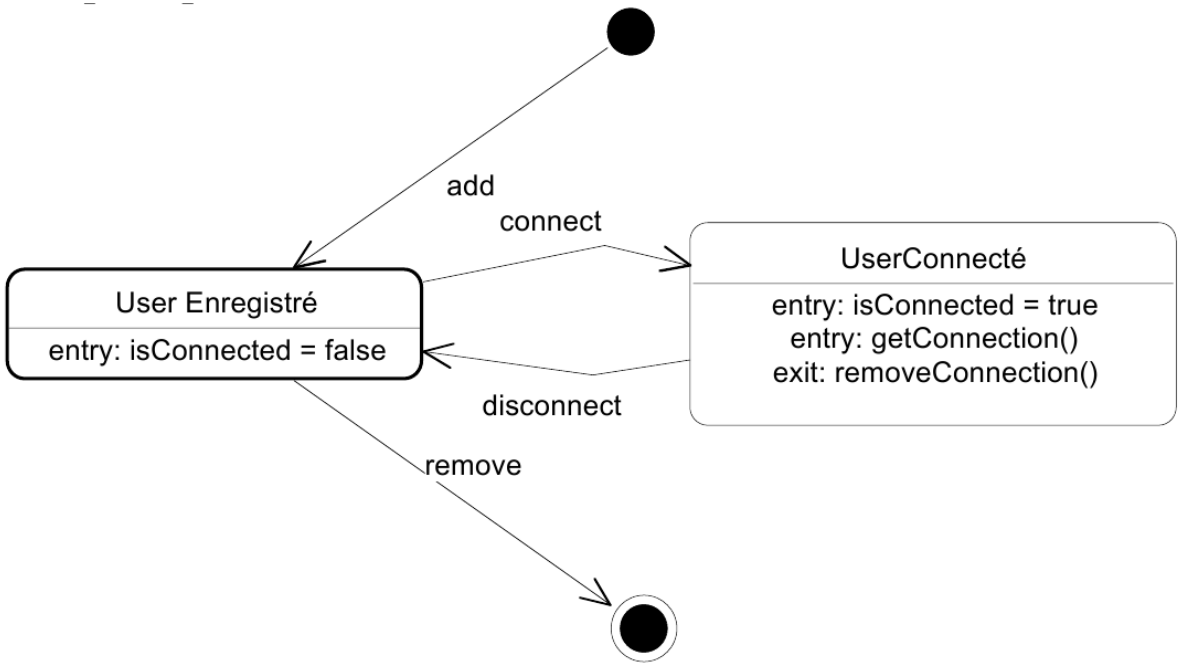


Figure: Diagramme d'états de la classe User avec actions

UML et les aspects dynamiques de l'analyse (suite)

LES OPERATIONS

LES OPERATIONS

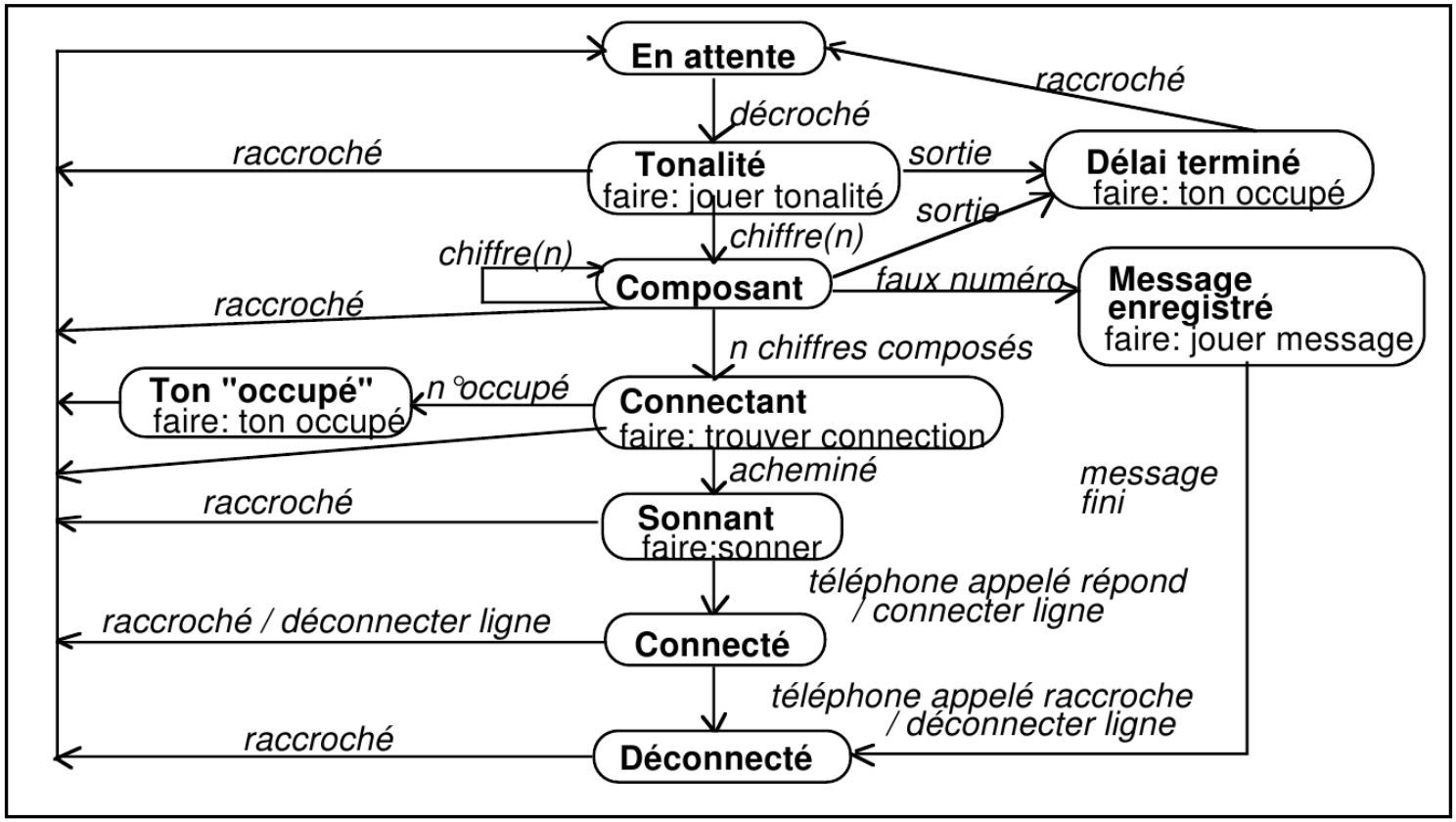


Figure: Diagramme d'états avec activités et actions (notation NON UML)

UML et les aspects dynamiques de l'analyse (suite)

RELATIONS ENTRE MODELE OBJET ET MODELE DYNAMIQUE

- La structure du modèle dynamique est rattachée à la structure du modèle objet.
- Les états sont des classes d'équivalence sur les valeurs des attributs et des liens.
- Un événement est le résultat d'une opération du modèle objet.
- Pour l'analyse, le nom d'un événement est plus expressif que le nom d'opération, car l'effet d'un événement dépend aussi de l'état de l'objet.
- Les différences intrinsèques sont modélisées sous forme de classes différentes; les différences temporaires sont modélisées sous la forme d'états différents.
- Le modèle dynamique d'une classe est hérité par ses sous-classes. Les sous-classes héritent des états et des transitions de leurs ancêtres.
- Les sous-classes peuvent avoir leur propre modèle d'états, surchargeant le modèle hérité.
- La hiérarchie des événements est indépendante de la hiérarchie du modèle objet.
- La hiérarchie des événements exprime des niveaux plus ou moins abstraits de description des fonctionnalités de l'application.

➥ Une "bonne" architecture objet est une architecture qui permet de traduire les évts en envoi de messages d'une manière simple et efficace.

Etude de cas : GAB et son use-case

Etude de cas : GAB et son use-case

Enoncé simplifié

• Le GAB offre les services suivants:

- Distribuer de l'argent à tout porteur de carte bancaire (visa ou de la banque)
- Consultation du solde, dépôt en numéraire et dépôt de chèques pour le clients de la banque porteurs de la carte de la banque

• Il faut savoir que

- Toutes les transactions sont sécurisées
- Il et nécessaire parfois de recharger le distributeur en argent, en papier pour l'impression des tickets, récupérer les chèques déposés, les cartes avalées, et les numéraires déposés etc.

Etude de cas : GAB et son use-case (suite)

Identification des acteurs du GAB

• Un acteur représente une catégorie d'individus externes au système et qui ont le même comportement vis-à-vis de celui-ci et qui consiste à attendre un service.
• Un acteur peut être un agent humain ou un autre système
• Un acteur est dit *primaire* lorsqu'il est initiateur du cas d'utilisation.
Il est dit *secondaire* lorsque le déroulement du cas d'utilisation nécessite son intervention.

- Acteurs principaux
  - Porteur de CB visa
  - Client de la banque
  - Agent de maintenance
- Acteurs secondaires
  - SA Visa (organisme de cartes bancaires)
  - SI banque (système d'information de la banque)

Etude de cas : GAB et son use-case (suite)

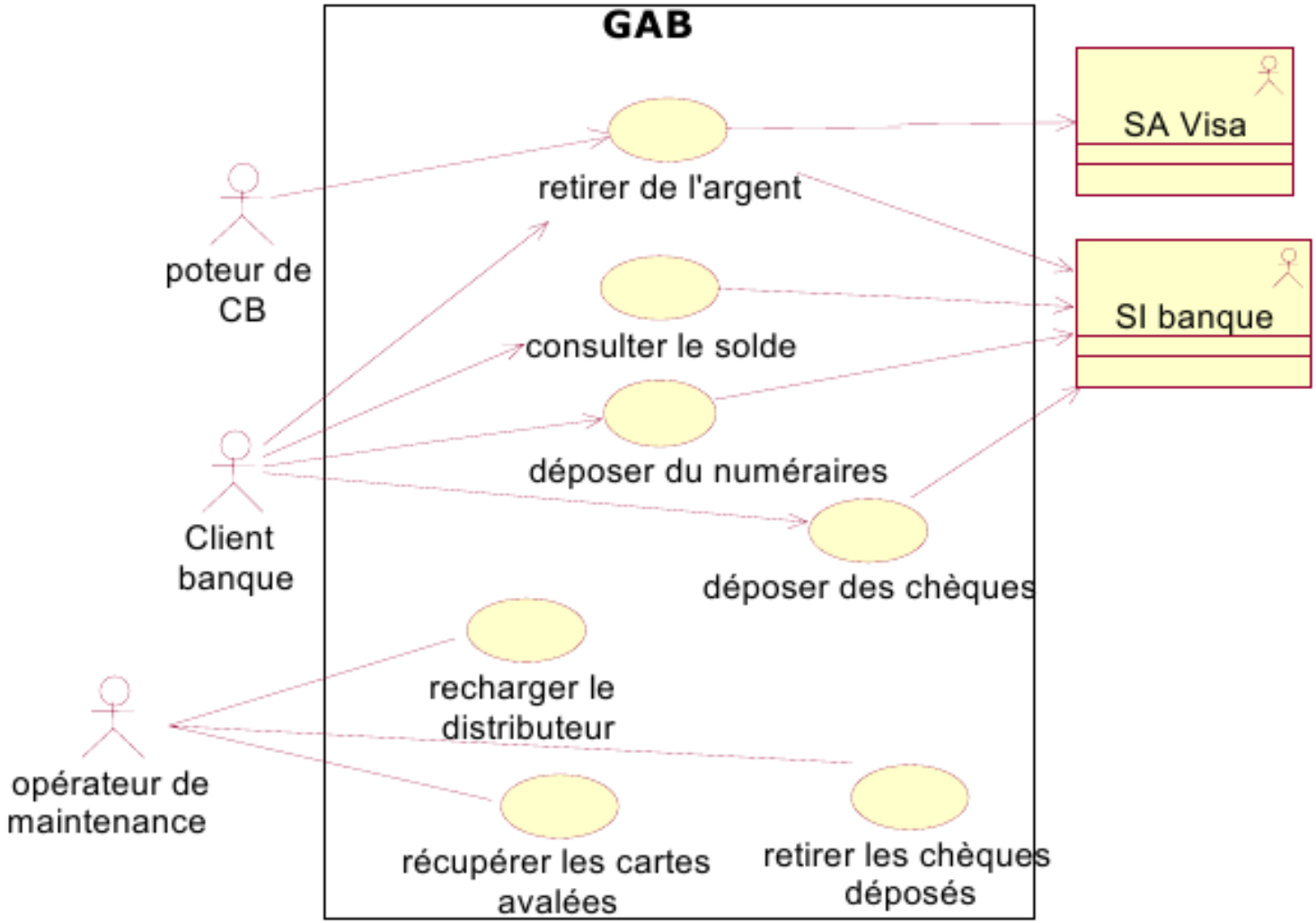
Identification des cas d'utilisation

- Pour un porteur de carte visa
  - Retirer de l'argent
- Pour un client de la banque
  - Retirer de l'argent
  - Consulter le solde
  - Déposer des chèques
  - Déposer des numéraires
- Pour l'agent de maintenance
  - Retirer les chèques et numéraires
  - Récupérer les cartes avalées
  - Recharger le distributeur

../..

Etude de cas : GAB et son use-case (suite)

Diagramme de cas d'utilisation du GAB



Etude de cas : GAB et son use-case (suite)

Documentation les cas d'utilisation

• Un cas d'utilisation ne se limite pas à la représentation graphique, sa documentation n'est pas optionnelle mais nécessaire et obligatoire pour la définition du cas.

• La documentation doit expliciter les activités à dérouler, le déroulement nominal (quand tout se passe bien), le déroulement exceptionnel, les déroulements alternatifs (interruption du déroulement nominal du cas initial par un autre cas d'utilisation pour les besoins du cas initial) et les pré et post conditions pour s'assurer du bon déroulement du cas.

Titre: Retirer de l'argent avec une carte visa
Résumé: ce cas permet à porteur de CB, qui n'est pas client de la banque, de retirer de l'argent, si son crédit hebdomadaire le permet.
Date de création: 02/03/00 **Date de mise à jour:** 09/11/00
Version: 2.2 **Responsable**: P. Rocques
Pré conditions:
la caisse du GAB est alimentée
Aucune carte bancaire ne se trouve dans le lecteur

Etude de cas : GAB et son use-case (suite)

Documentation les cas d'utilisation (suite) : **Scénario nominal:**

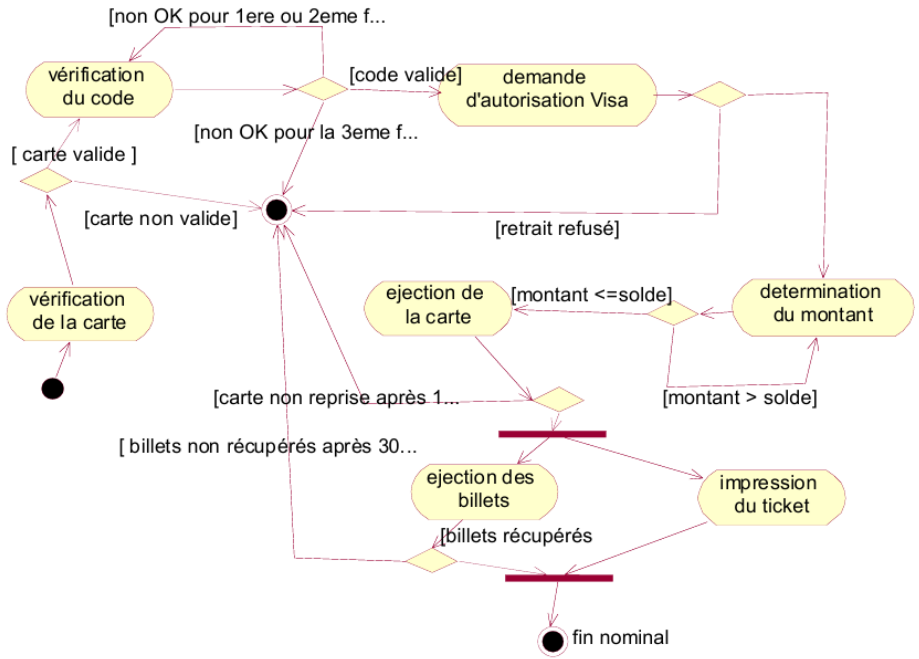
1. Le porteur de CB introduit la carte dans le lecteur du GAB
2. Le GAB vérifie que la carte introduite est bien une carte Visa
3. Le GAB demande au porteur de saisir son code d'identification
4. Le porteur saisit son code d'identification
5. Le GAB compare le code saisi avec celui inscrit dans la puce de la carte
6. Le GAB demande une autorisation au système d'autorisation SA Visa
7. SA Visa donne son accord en indiquant le solde hebdomadaire
8. Le GAB demande au porteur de CB d'indique le montant souhaité
9. Le porteur saisit le montant
10. Le GAB contrôle le montant demandé par rapport au solde hebdomadaire
11. Le GAB demande au porteur de CB s'il veut un ticket
12. Le porteur de CB demande un ticket
13. Le GAB rend la carte au porteur de CB

Etude de cas : GAB et son use-case (suite)

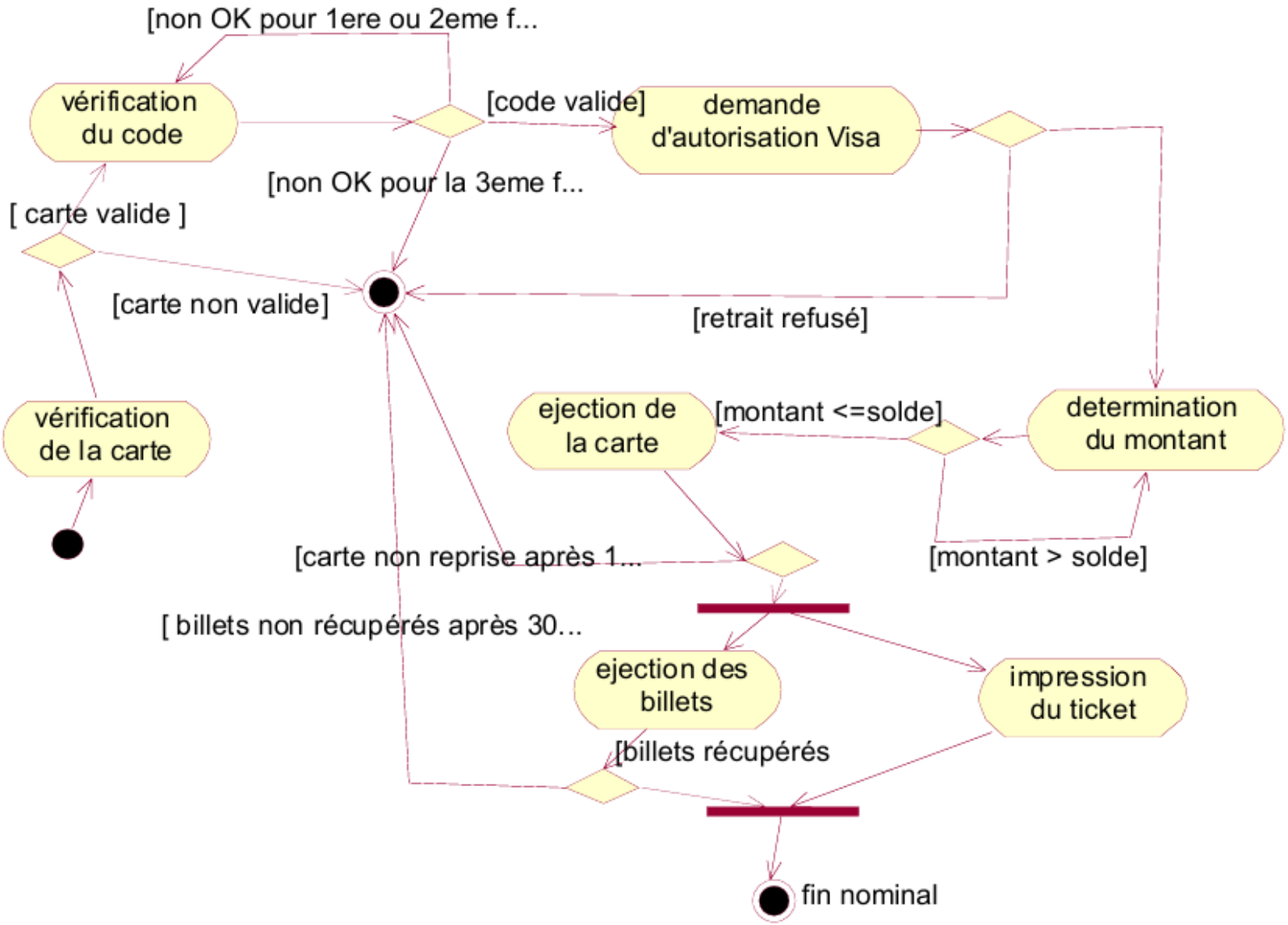
14. Le porteur de CB reprend sa carte
15. Le GAB délivre les billets et un ticket
16. Le porteur de CB prend les billets et le ticket.

N.B. : Il y a bien entendu d'autres scénarii d'utilisation possibles.

Important : Détailler un cas d'utilisation avec un diagramme d'activités



Etude de cas : GAB et son use-case (suite)

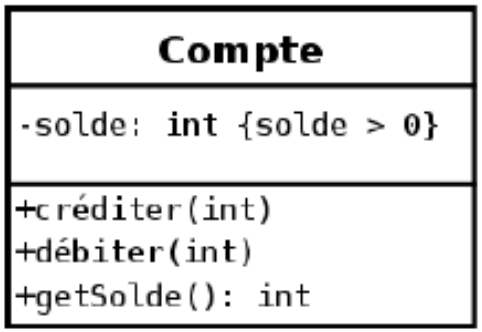


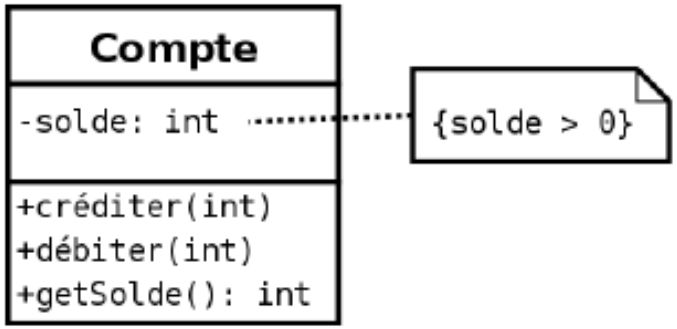
Contraintes en UML

Contraintes en UML

• UML permet d'associer une contrainte à un élément de modèle de plusieurs façons.

Exemple-1 : la contrainte **{frozen}** précise que le nombre de roues d'un véhicule ne peut pas varier.

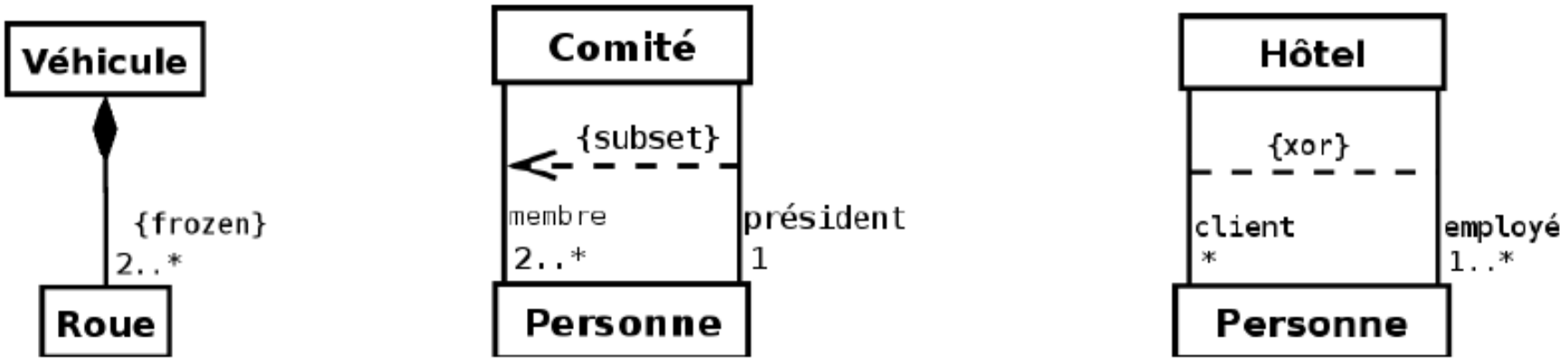




../..

Contraintes en UML (suite)

• Suite exemple 1 :



• Ci-dessus, au milieu, la contrainte **{subset}** précise que le président est également un membre du comité.
• A droite, la contrainte **{xor}** (ou exclusif) précise que les employés de l'hôtel n'ont pas le droit de prendre une chambre dans ce même hôtel.

Contraintes en UML (suite)

Exemple 2 :

- **{frozen}** : une personne est née dans un pays, et que cette association ne peut être modifiée ;

- **{Addonly, Ordered }** : une personne a visité un certain nombre de pays, dans un ordre donné , et que le nombre de pays visités ne peut que croître ;

- une personne aimerait encore visiter tout une liste de pays, et que cette liste est ordonnée (probablement par ordre de préférence).

Contraintes en UML (suite)

- **Langage de contraintes OCL** *(Object Constraint Language)*

Un exemple : application bancaire.

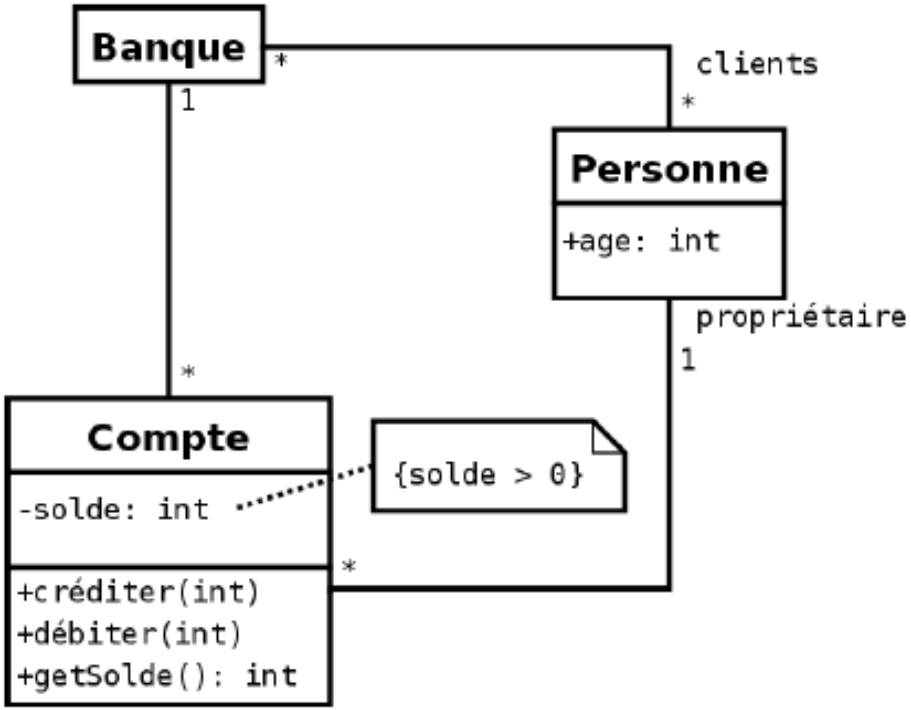
- Il faut gérer :
  - des comptes bancaires,
  - des clients,
  - et des banques.

- De plus, on aimerait intégrer les contraintes suivantes dans notre modèle :
  - un compte doit avoir un solde toujours positif ;
  - un client peut posséder plusieurs comptes ;
  - une personne peut être cliente de plusieurs banques ;
  - un client d'une banque possède au moins un compte dans cette banque ;
  - un compte appartient forcément à un client ;
  - une banque gère plusieurs comptes ;
  - une banque possède plusieurs clients.

Contraintes en UML (suite)

Diagramme de classes modélisant une banque, ses clients et leurs comptes :

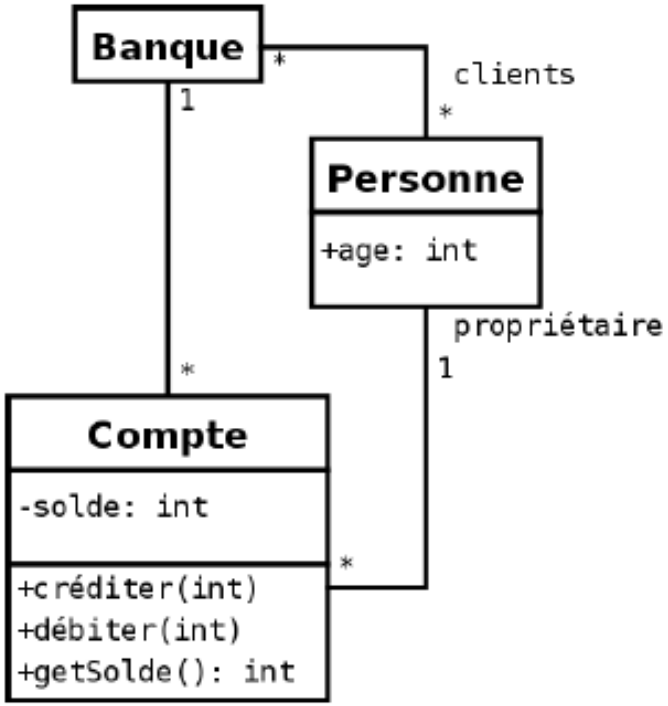
La contrainte solde positif ajoutée.



Contraintes en UML (suite)

Exemple d'utilisation du langage de contrainte OCL sur l'exemple bancaire.

➥ Les contraintes peuvent être exprimées sous forme de notes.



context Compte
inv : solde > 0

context Compte : : débiter(somme : int)
pre : somme > 0
post : solde = solde@pre - somme

context Compte
inv : banque.clients -> includes (propriétaire)

- On peut exprimer les contraintes sous forme de **texte** ➙ OCL

Contraintes en UML (suite)

OCL : un exemple plus complet appliqué au diagramme de classes :

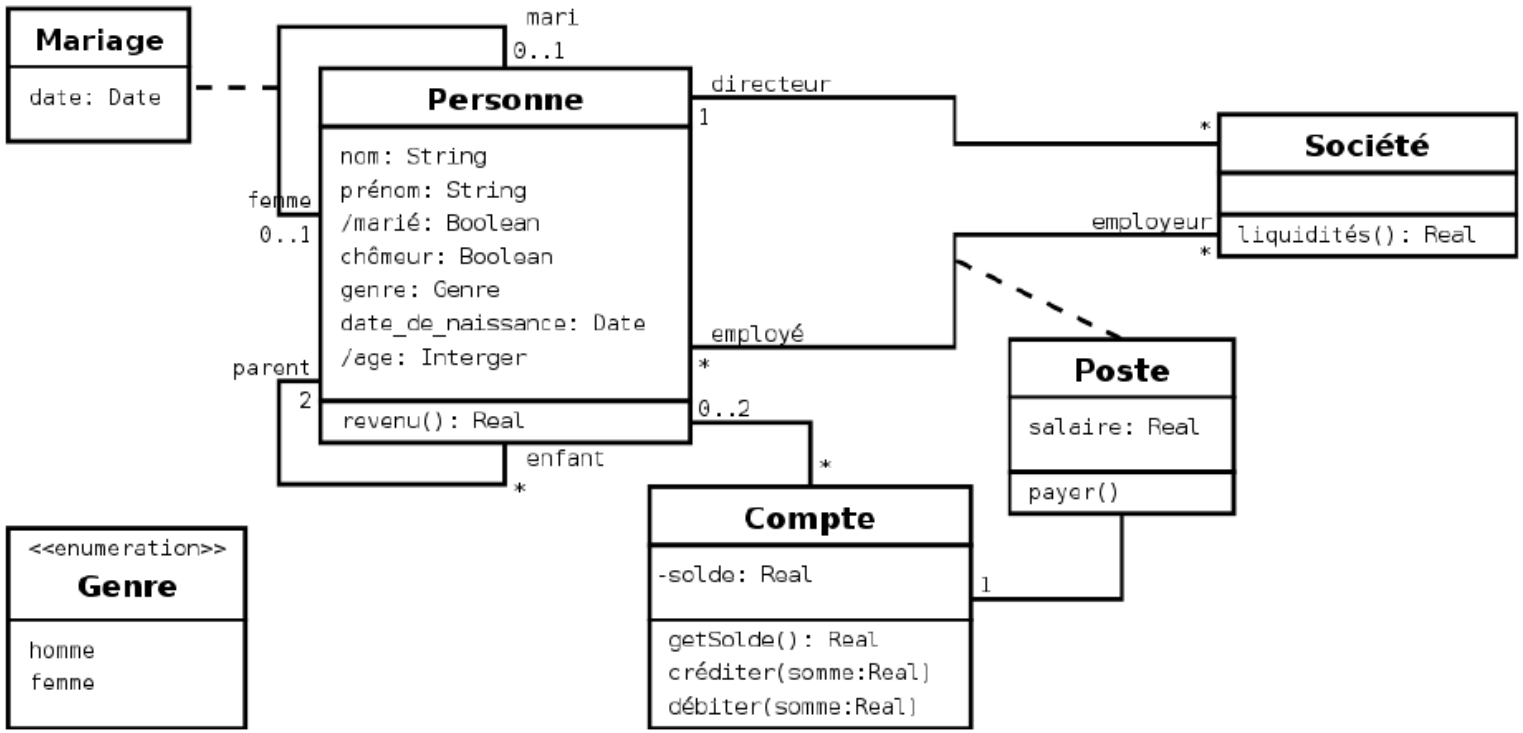


Figure: Diagramme des classe (support des contraintes OCL)

Contraintes en UML (suite)

1. Dans une société, le directeur est un employé, n'est pas un chômeur et doit avoir plus de 40 ans.

   De plus, une société possède exactement un directeur et au moins un employé.

```
context Société
inv :
    self.directeur->size()=1 and                    '->' sur des fonctions
    not(self.directeur.chômeur) and
    self.directeur.age > 40 and
    self.employé->includes(self.directeur)
```

2. Une personne considérée comme au chômage ne doit pas avoir des revenus supérieurs à 100 Euros.

```
context Personne
inv :
    let revenus : Real = self.poste.salaire->sum() in
    if chômeur then revenus < 100
    else revenus ≥ 100
    endif
```

Contraintes en UML (suite)

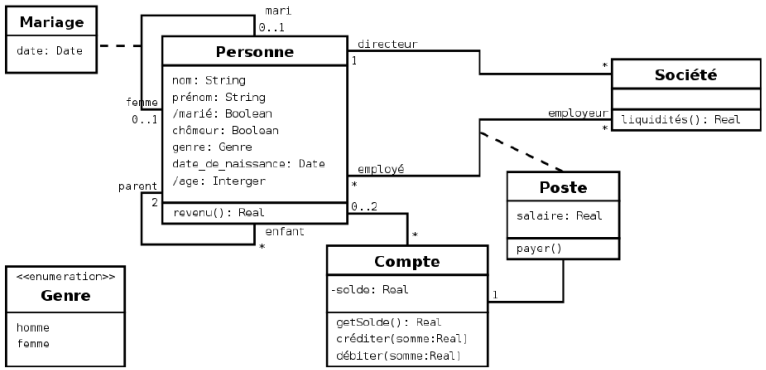
3. Une personne possède au plus 2 parents (référencés).

```
context Personne inv : parent->size()<=2
```

4. Si une personne possède deux parents, l'un est une femme et l'autre un homme.

NB : '::' : pour l'accès à une composition (désigner un élément dans un englobant)

```
context Personne
inv :
    parent->size()=2 implies
    ( parent->exists(genre=Genre::homme) and
    parent->exists(genre=Genre::femme) )
```



Contraintes en UML (suite)

5. Tous les enfants d'une personne ont bien cette personne comme parent et inversement.

➥ Expression de deux contrainte séparées sur Personne (invariant modifié)

```
context Personne
inv :
    enfant->notEmpty() implies
    enfant->forAll( p : Personne | p.parents->includes(self))
```

```
context Personne
inv :
    parent->notEmpty() implies
• parent->forAll ( p : Personne | p.enfant->includes (self))
```

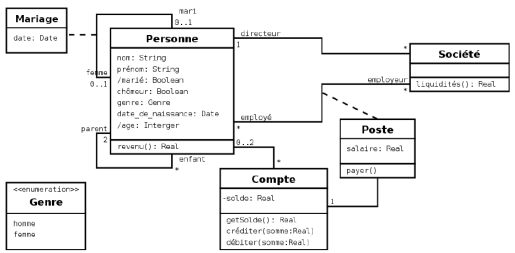
6. Pour être marié, il faut avoir une femme ou un mari.

```
context Personne::marié
derive : self.femme->notEmpty() or self.mari->notEmpty()
```

Contraintes en UML (suite)

7. Pour être marié, il faut avoir plus de 18 ans. Un homme est marié avec exactement une femme et une femme avec exactement un homme.

```
context Personne
inv :
    self.marié implies
    self.genre=Genre::homme implies (
    self.femme->size()=1 and
    self.femme.genre=Genre::femme)
    and self.genre=Genre::femme implies (
    self.mari->size()=1 and
    self.mari.genre=Genre::homme)
    and self.age >=18
```



Les Diagramme Etat-Transition

Notion et exemple d'automate à étatsfinis

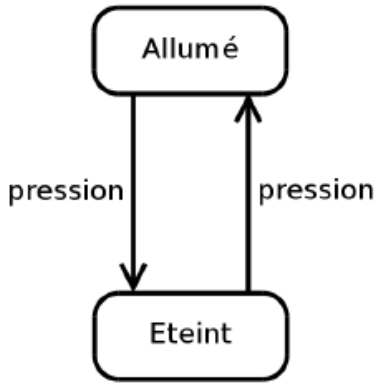


Figure: Un diagramme d'états-transitions simple

Figure: état simple, initial et final

Événement

Événement
Événement de type signal (signal)

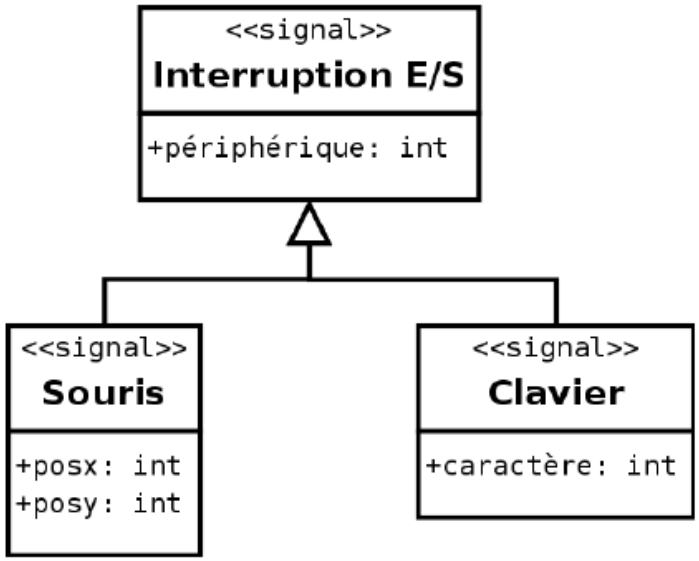


Figure: Déclaration de signaux et héritage.

Evénement Temporel :

after(durée)

when (date=une date)

Événement (suite)

Transition (réponse à un événement)

- La syntaxe d'une transition est la suivante :
  [<événement>]['[' <garde> ']'] ['/' <activité>]

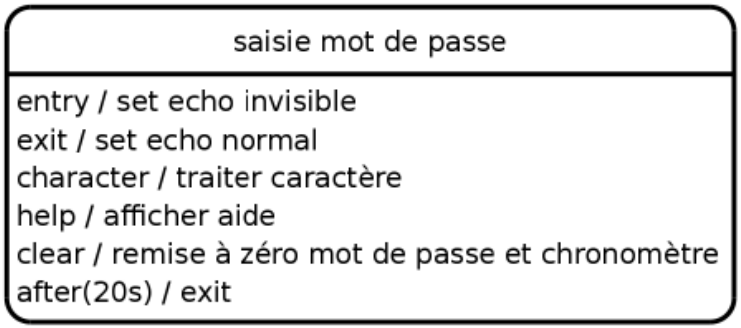


Figure: Représentation de la saisie d'un mot de passe dans un état unique en utilisant des transitions internes.

Événement (suite)

• Les transitions internes possèdent des noms d'événement prédfinis correspondant à des déclencheurs particuliers : *entry, exit, do et include.*

entry permet de spécifier une activité qui s'accomplit quand on entre dans l'état.

exit permet de spécifier une activité qui s'accomplit quand on sort de l'état.

do Une activité do commence dès que l'activité entry est terminée.

Lorsque cette activité est terminée, une transition d'achèvement peut être déclenchée, après l'exécution de l'activité exit bien entendu.

Si une transition se déclenche pendant que l'activité do est en cours, cette dernière est interrompue et l'activité exit de l'état s'exécute.

include permet d'invoquer un sous-diagramme d'états-transitions.

Événement (suite)

Evénement-Transition (suite)
Point de choix : ... voir point de décision

Point de jonction

• Un point de jonction (artefact graphique = un pseudo-état) permet de partager des segments de transition,et d'aboutir à une notation plus compacte ou plus lisible des chemins alternatifs.

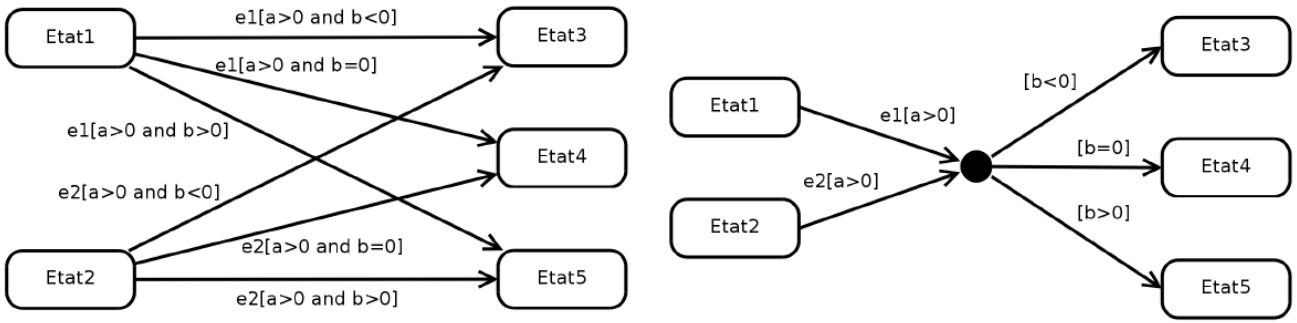


Figure: A gauche, un diagramme sans point de jonction. A droite, son équivalent utilisant un point de jonction.

Événement (suite)

Point de jonction (suite)

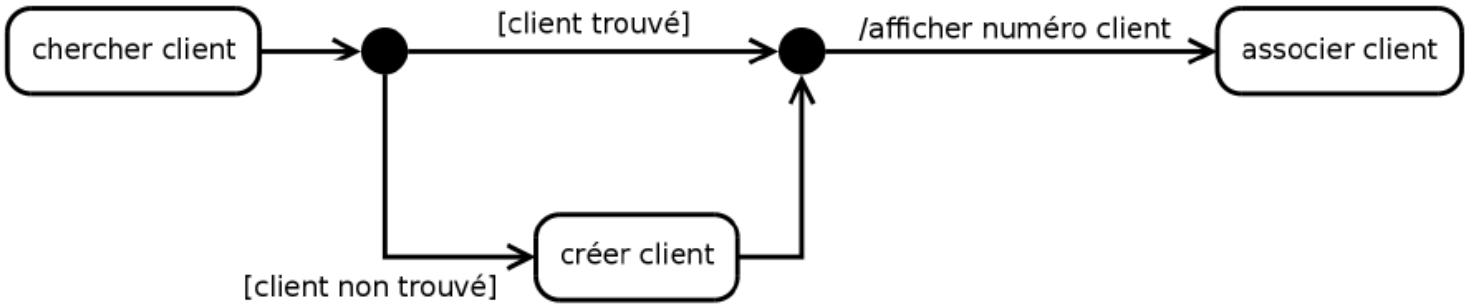


Figure: Exemple d'utilisation de deux points de jonction pour représenter une alternative.

Événement (suite)

Point de décision

Point de choix.

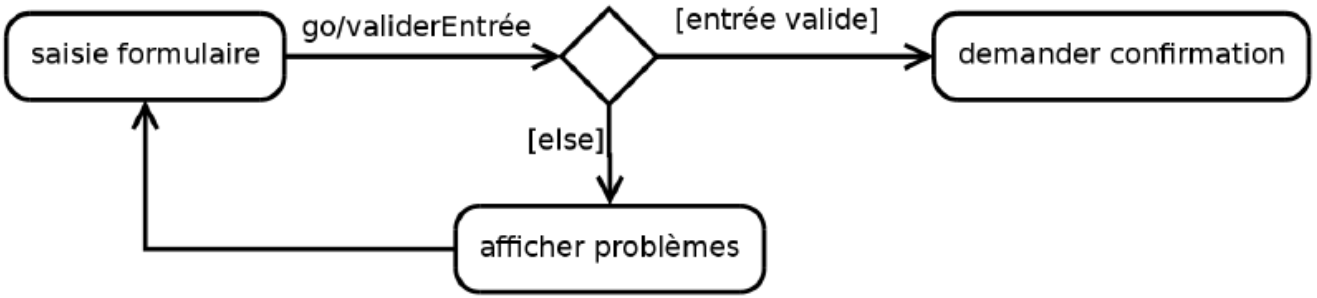


Figure: Exemple d'utilisation d'un point de décision.

Événement (suite)

États composites

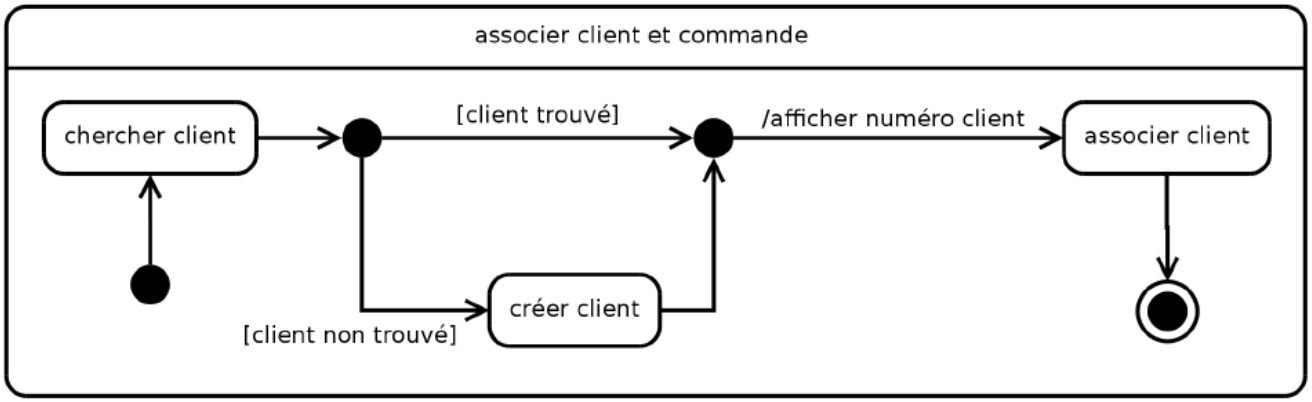


Figure: Exemple d'état composite modélisant l'association d'une commande à un client.

- Il existe d'autre notions dans UML.... (plus poussées)

Autres Diagrammes UML

Diagramme d'activités

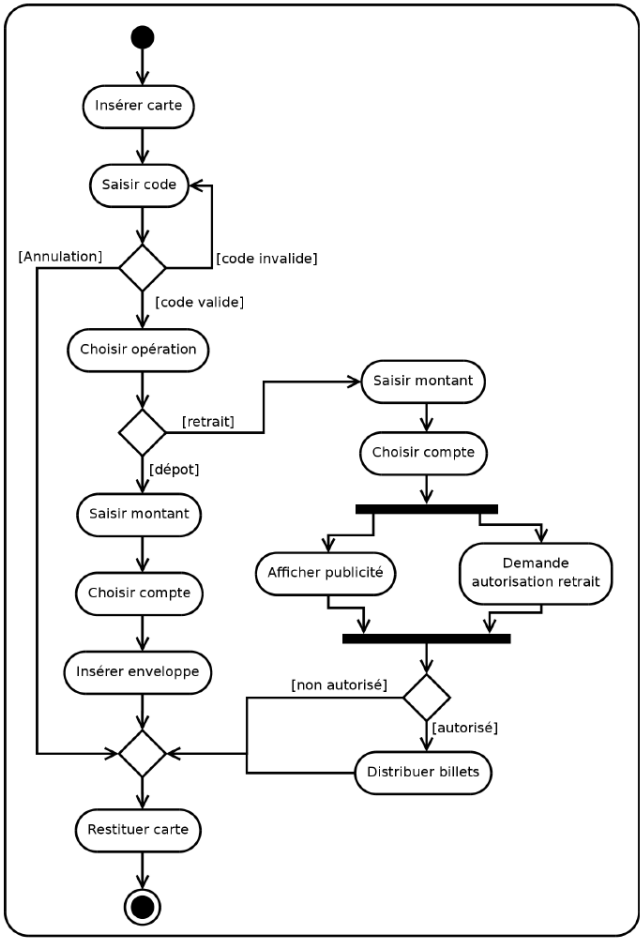


Figure: Exemple de diagramme d'activités : fonctionnement d'une borne bancaire.

Autres Diagrammes UML (suite)

Diagramme d'activités et noeuds de contrôle

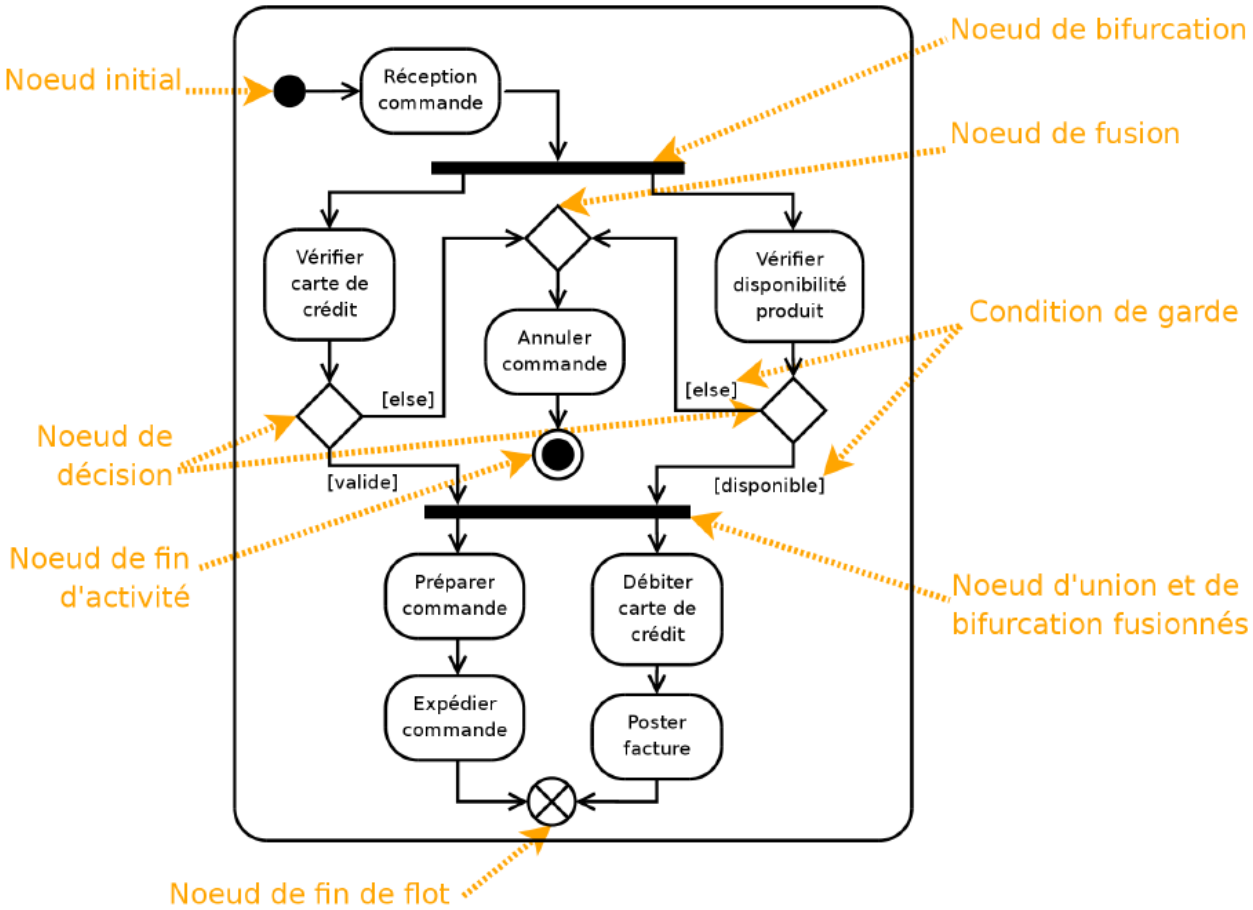


Figure: Différents Noeuds dans un diagramme d'activités../..

Autres Diagrammes UML (suite)

Noeuds

- Il existe plusieurs types de noeuds de contrôle :
  - noeud initial (initial node en anglais) ;
  - noeud de fin d'activité (final node en anglais)
  - noeud de fin de flot (loww final en anglais) ;
  - noeud de décision (decision node en anglais) ;
  - noeud de fusion (merge node en anglais) ;
  - noeud de bifurcation (fork node en anglais) ;
  - noeud d'union (join node en anglais).

Autres Diagrammes UML (suite)

Partitions dans un Diagramme d'activités (cf. MOT Merise)

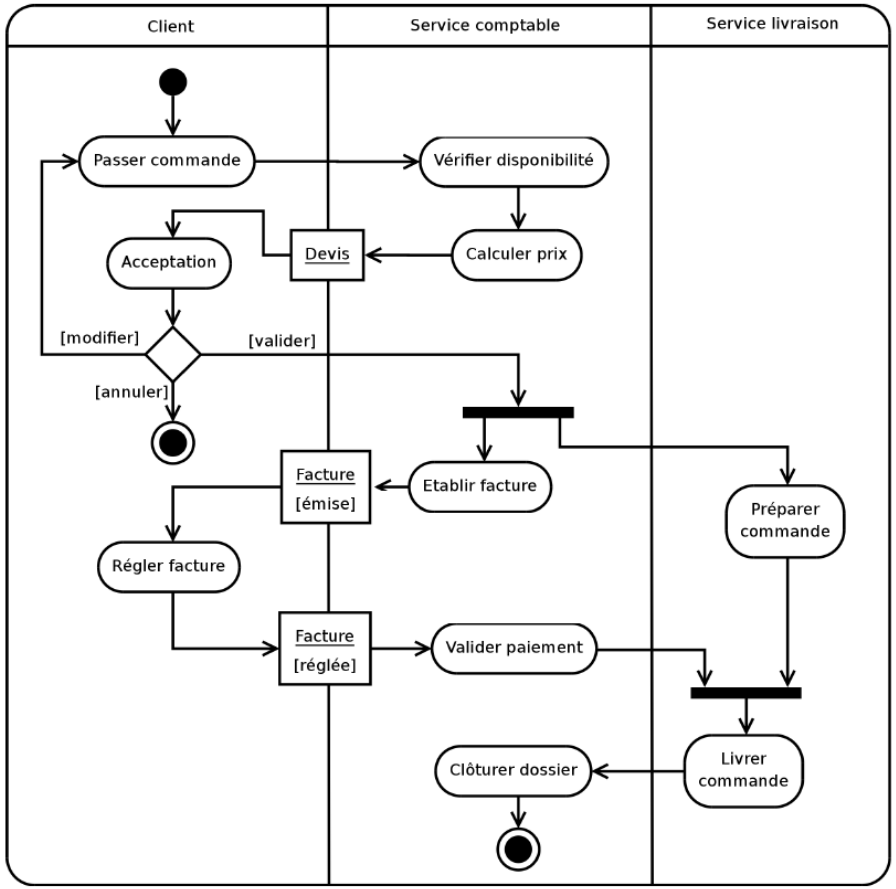


Figure: Utilisation de noeuds d'objets et de partitions dans un diagramme d'activités.

Autres Diagrammes UML (suite)

Diagramme de Collaboration

• Une collaboration permet de décrire la mise en $\frac{1}{2}$uvre d'une fonctionnalité par un jeu de participants.
• Un rôle est la description d'un participant.
• Contrairement aux paquetages et aux classeurs structurés, une collaboration ne détient pas les instances liées à ses rôles.
• Une collaboration peut traverser plusieurs niveaux d'un système et un même élément peut apparaître dans plusieurs collaborations.

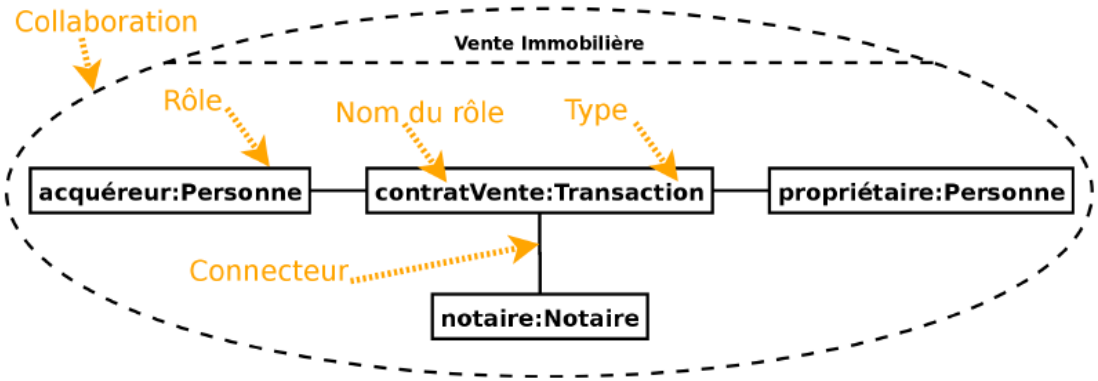


Figure: Diagramme de collaboration d'une transaction immobilière.

Autres Diagrammes UML (suite)

Diagrammes de communication

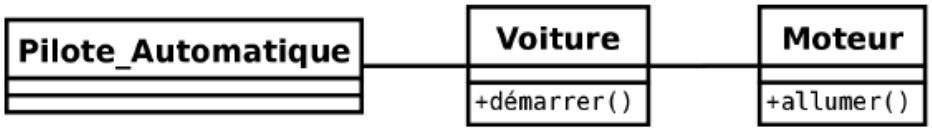


Figure: Diagramme de classe d'un système de pilotage.

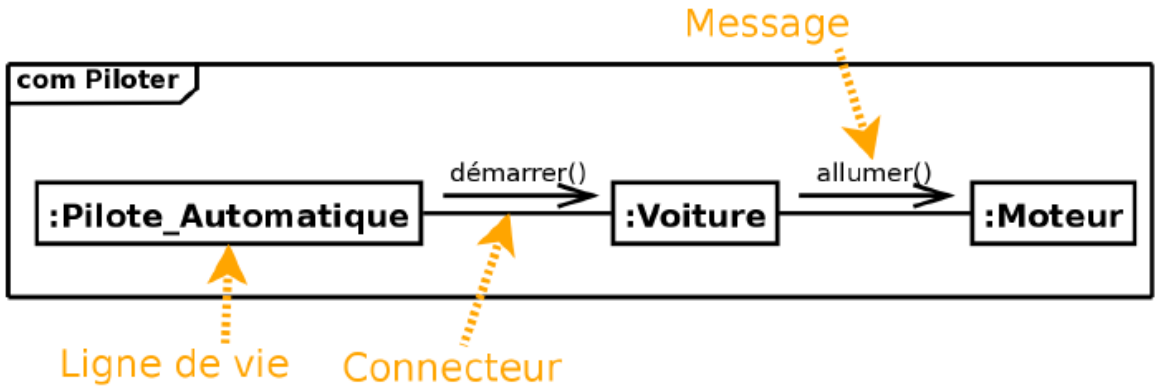


Figure: Diagramme de communication d'un système de pilotage.

Autres Diagrammes UML (suite)

Diagramme de Séquence

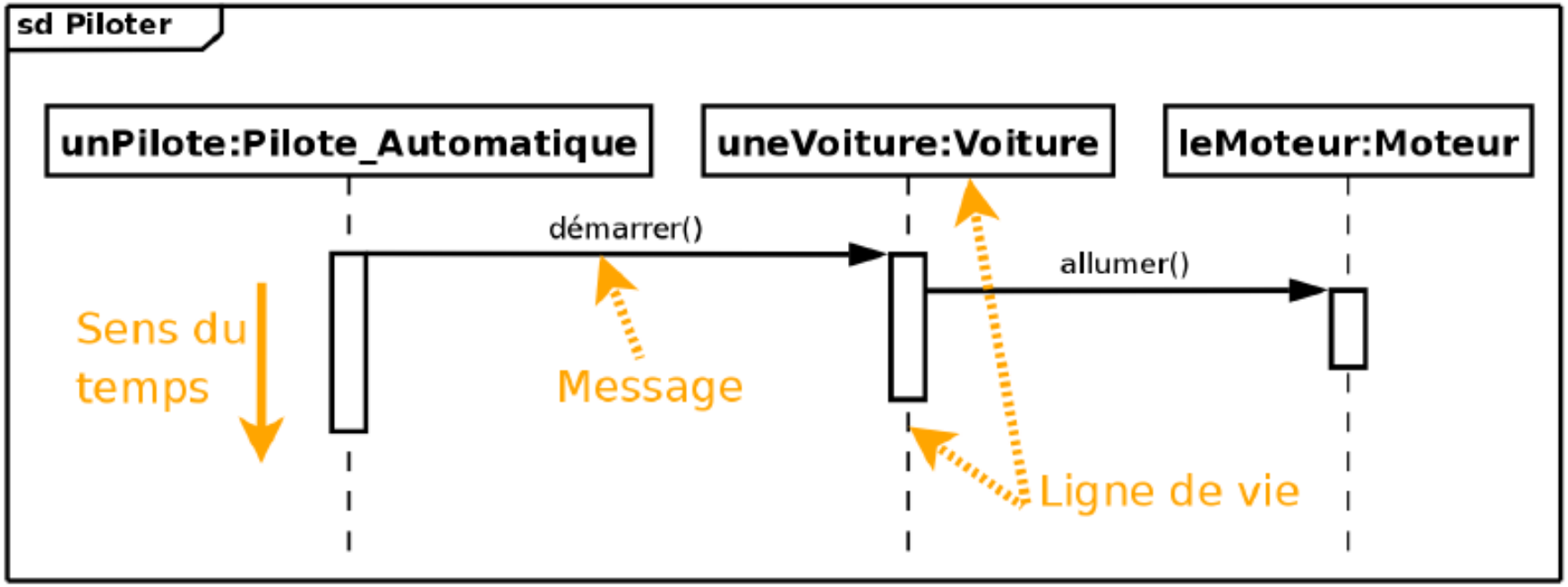


Figure: Diagramme de séquence d'un système de pilotage.

Autres Diagrammes UML (suite)

Un autre exemple de diagramme de communication

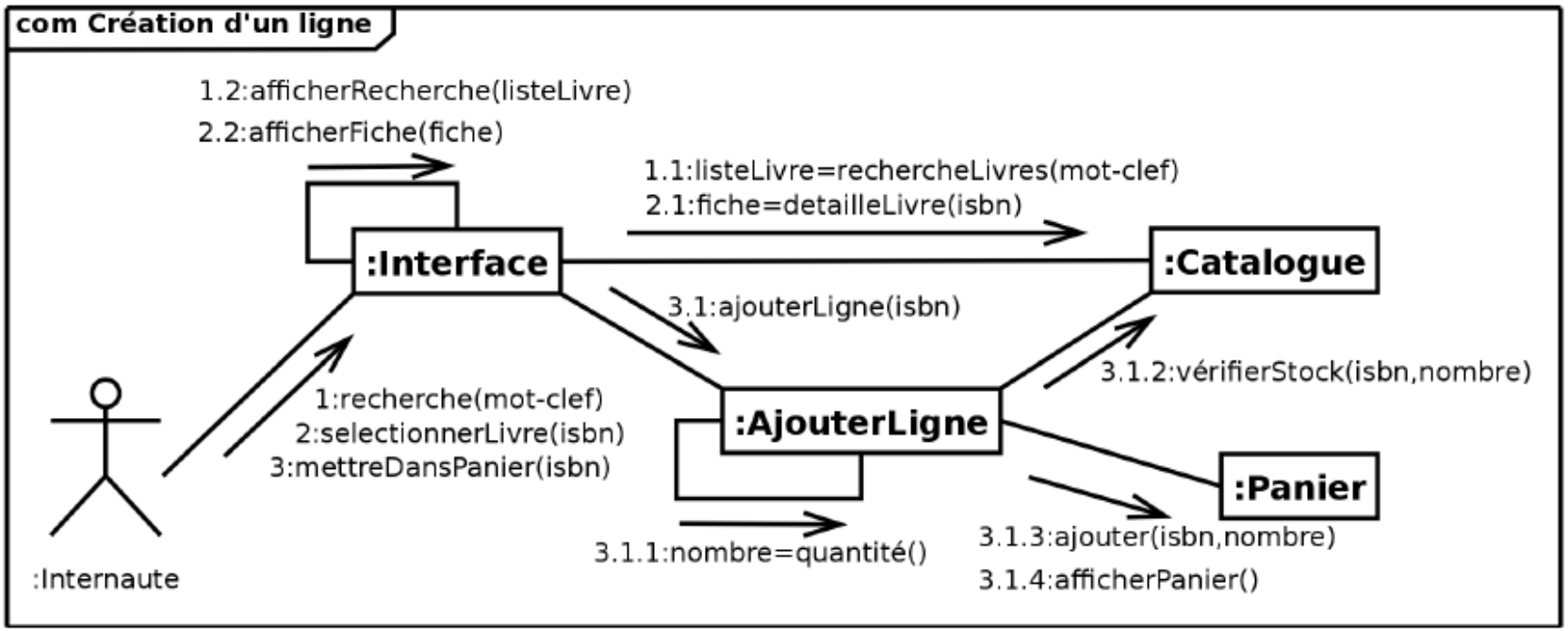


Figure: Diagramme de communication illustrant la recherche puis l'ajout, dans son panier virtuel, d'un livre lors d'une commande sur Internet.

Autres Diagrammes UML (suite)

Un autre exemple de diagramme de séquence jeu du démineur

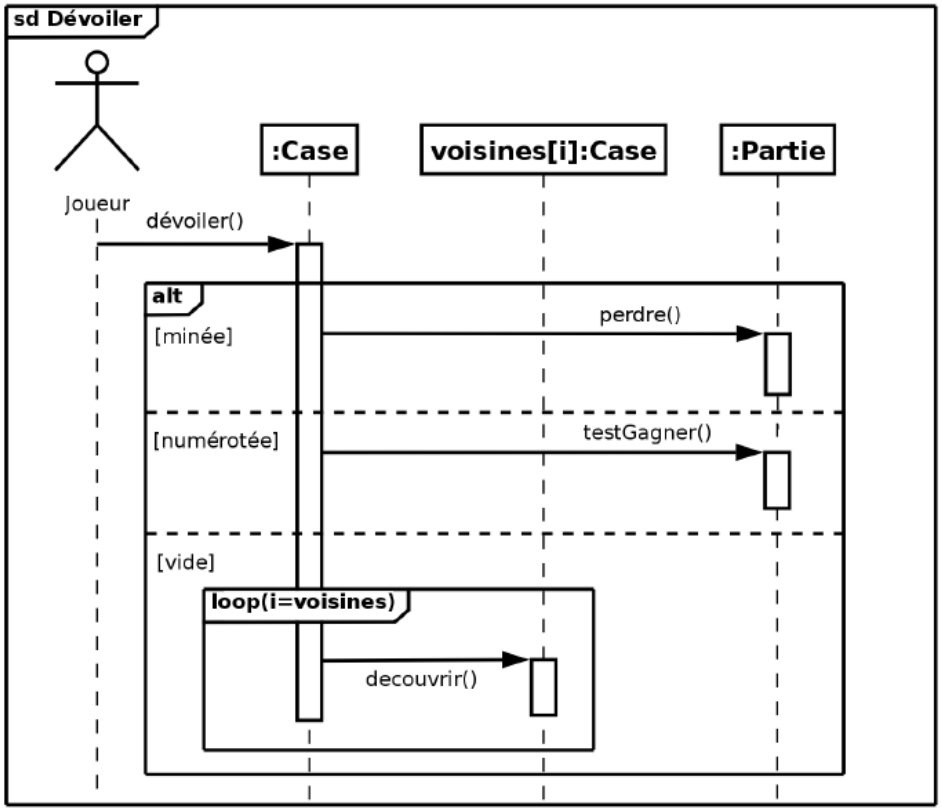


Figure: Représentation d'un choix dans un diagramme de séquence illustrant le découvrement d'une case au jeu du démineur.

Autres Diagrammes UML (suite)

Prise en compte d'activités parallèles dans un diagramme de séquence

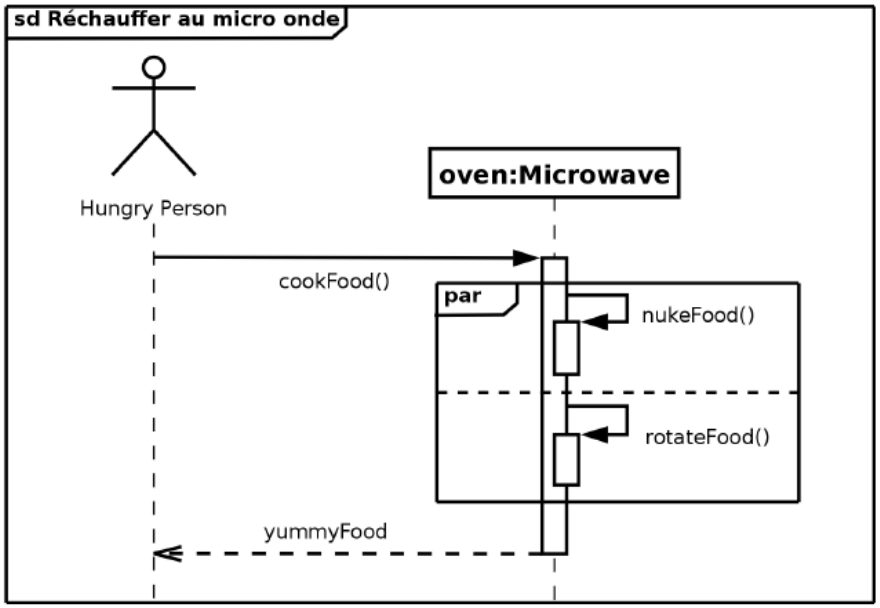
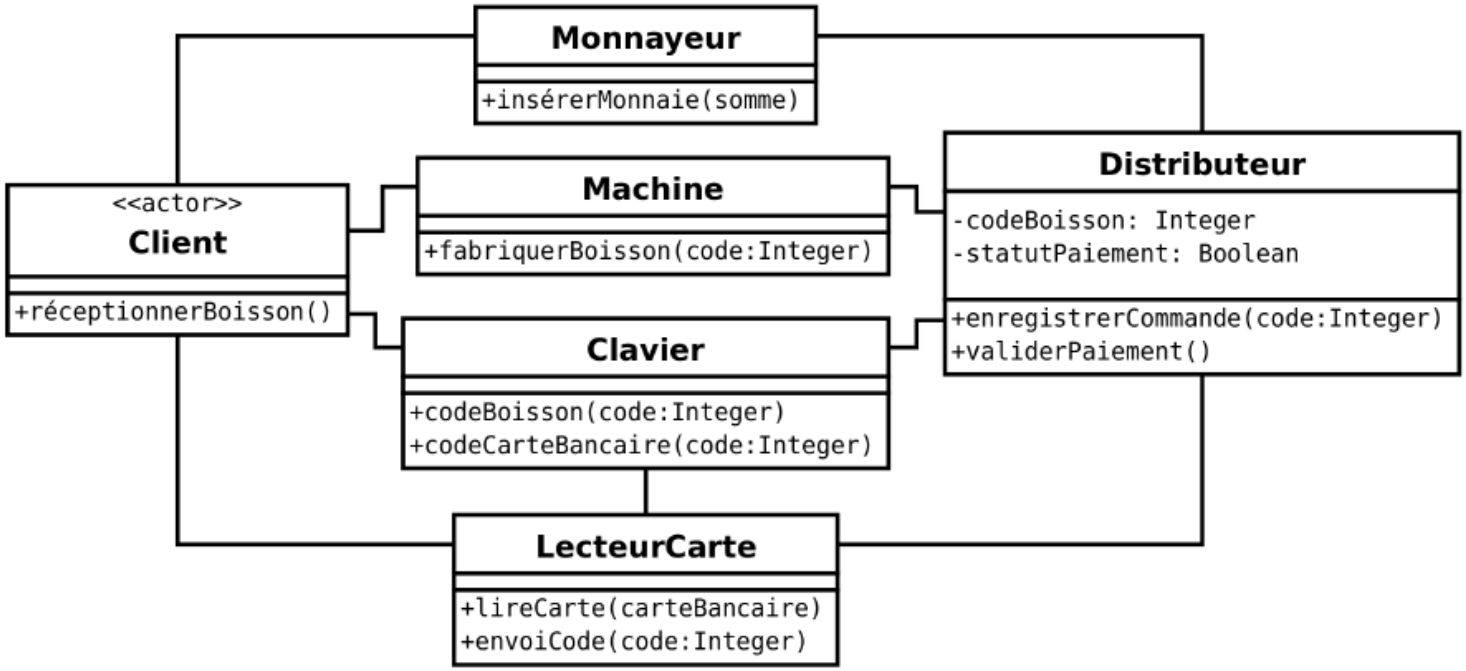


Figure: *Microwave* est un exemple d'objet effectuant deux tâches en parallèle.

Autres Diagrammes UML (suite)

Exercice :

• Un distributeur de boisson permet d'obtenir la boisson de son choix après avoir tapé le code de la boisson désirée puis payé par carte bancaire ou avec de la monnaie.



Autres Diagrammes UML (suite)

- En vous appuyant sur le diagramme de classes ci-dessus, proposez un diagramme de séquence illustrant une interaction allant de la commande d'une boisson à sa distribution et traitant les deux types de paiement.

  On ne vous demande pas de prendre en compte les cas exceptionnels (code de carte bancaire erroné, monnaie manquante, etc.)
- Traduisez votre diagramme de séquence en diagramme de communication

Autres Diagrammes UML (suite)

UML : On retient

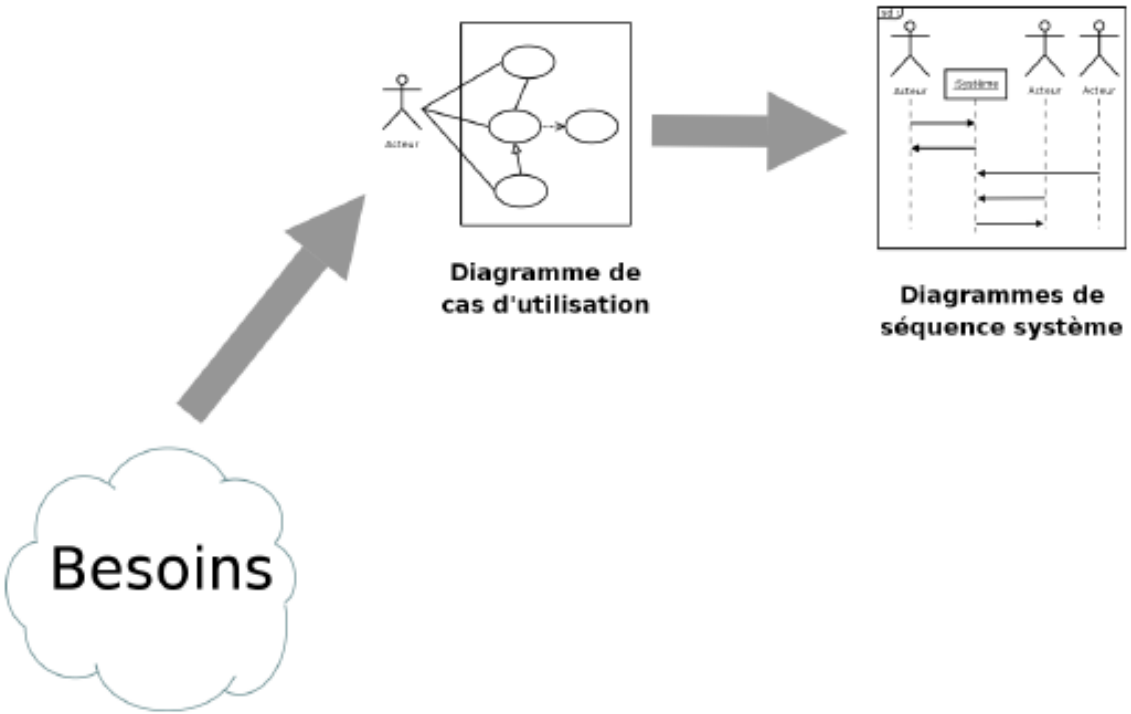


Figure: Les diagrammes de séquence système illustrent la description textuelle des cas d'utilisation.

Autres Diagrammes UML (suite)

On retient ...

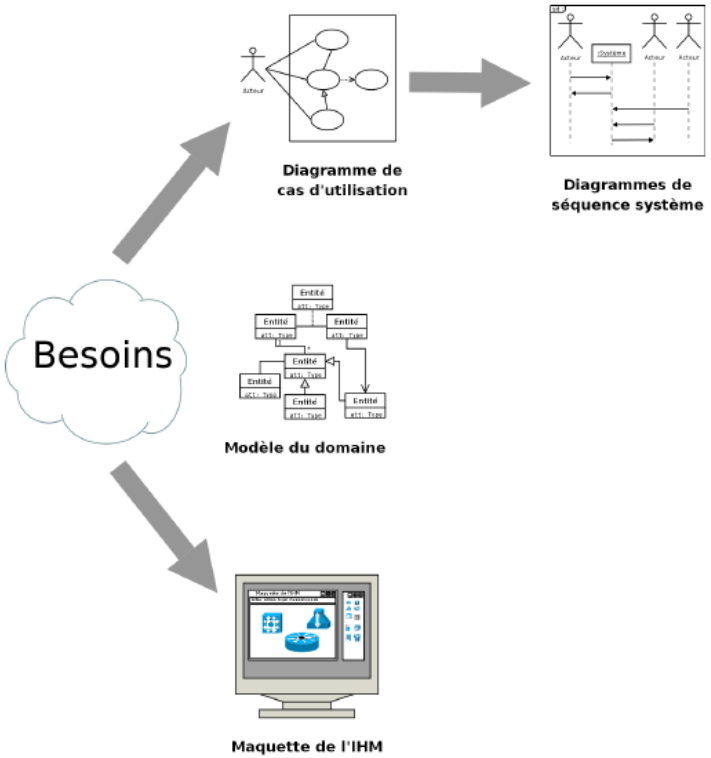


Figure: La phase d'analyse du domaine permet d'élaborer la première version du diagramme de classes.

Autres Diagrammes UML (suite)

On retient ...

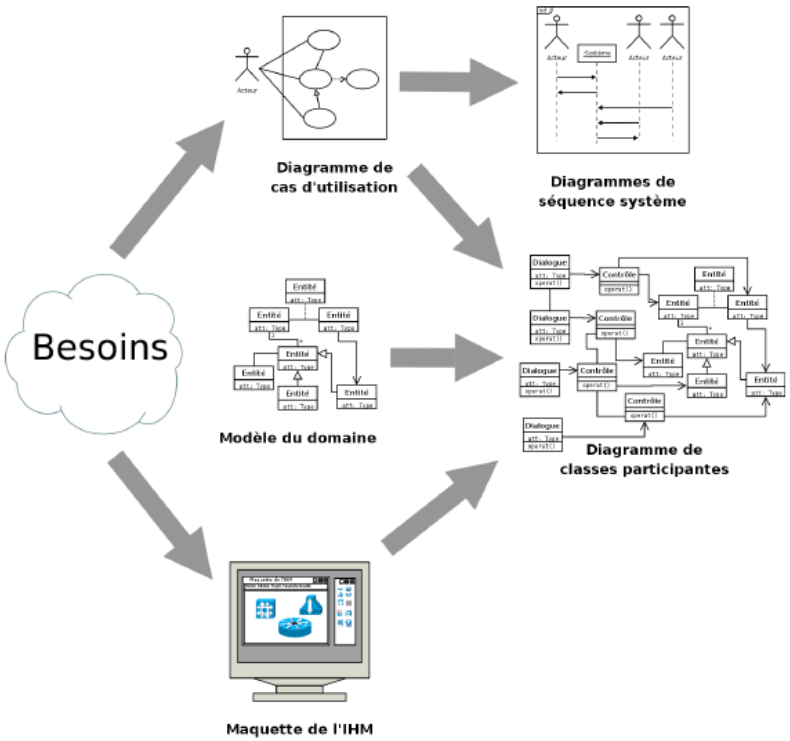


Figure: Le diagramme de classes participantes effctue la jonction entre les cas d'utilisation, le modèle du domaine et les diagrammes de conception logicielle.

Autres Diagrammes UML (suite)

On retient ...

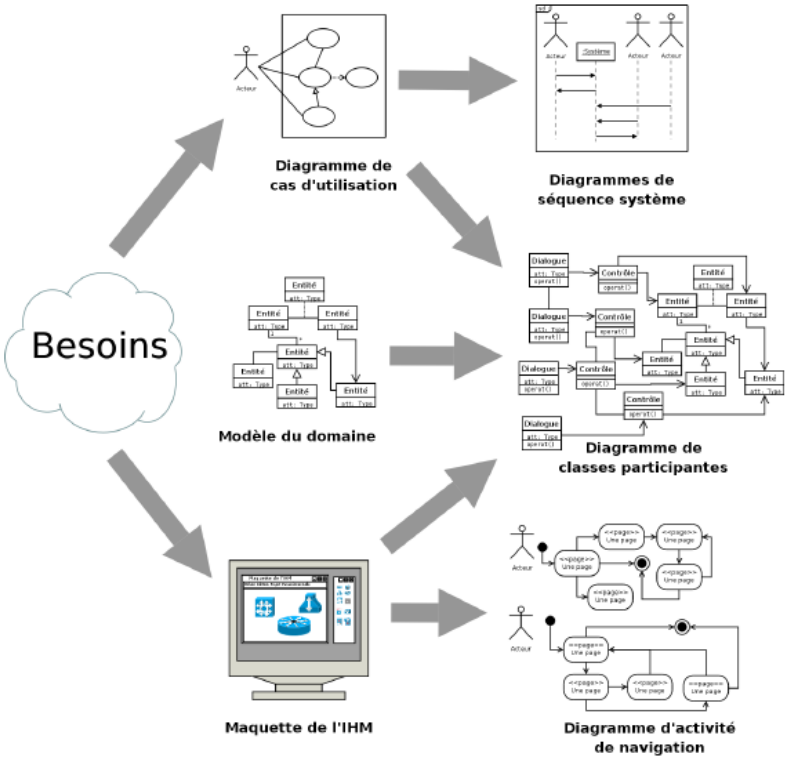


Figure: Les diagrammes d'activités de navigation représentent graphiquement l'activité de navigation dans l'IHM.

Autres Diagrammes UML (suite)

On retient ...

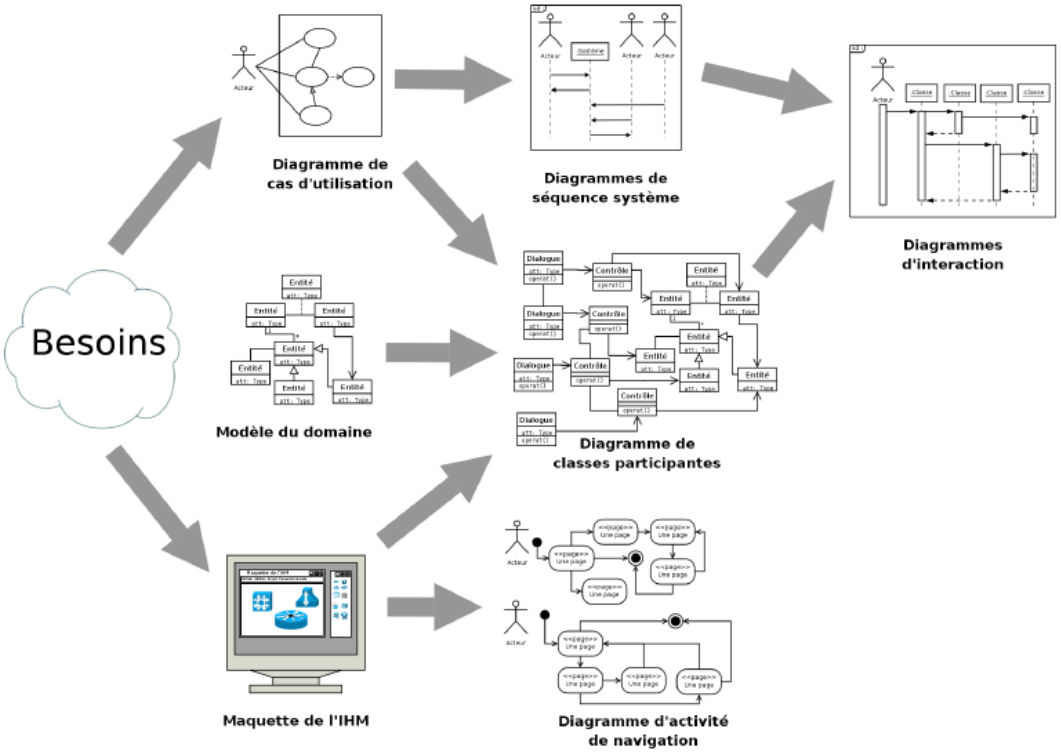


Figure: Les diagrammes d'interaction permettent d'attribuer précisément les responsabilités de comportement aux classes d'analyse.

Autres Diagrammes UML (suite)

On retient ...

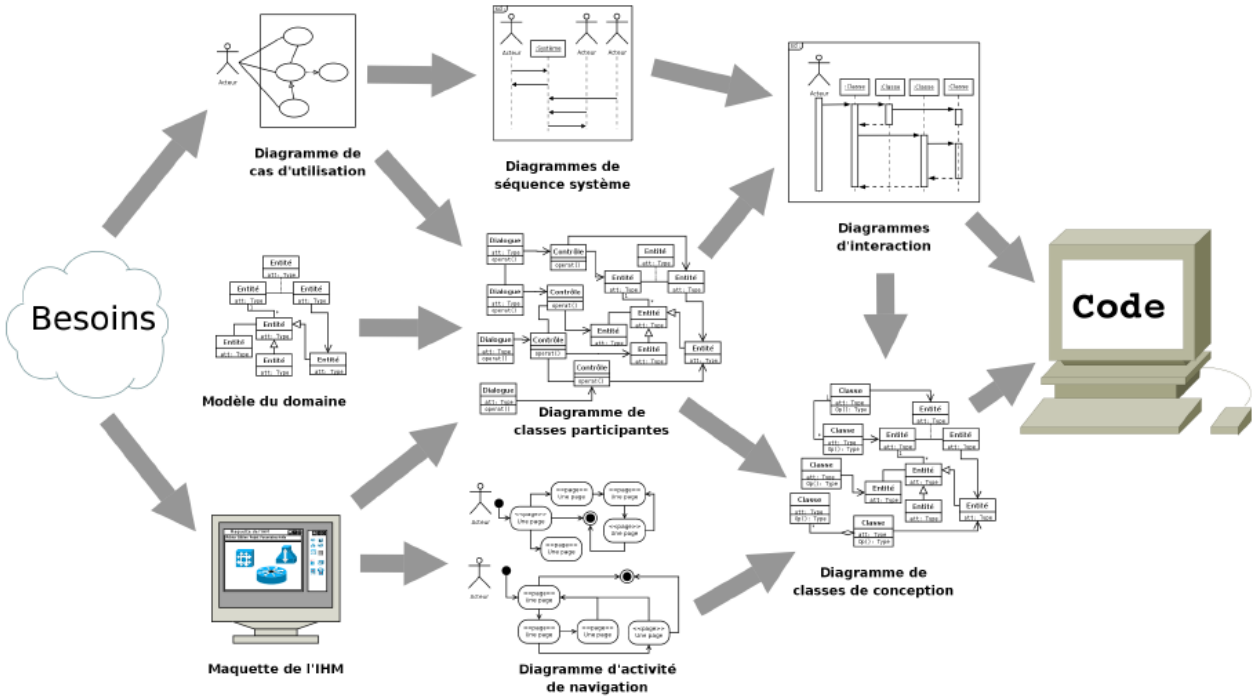


Figure: Chaîne complète de la démarche de modélisation du besoin jusqu'au code.

Rappel Sommaire

Notes sur Diagrammes UML

Le modèle des états et le modèle d'interaction

- Nous avons décrit le formalisme modélisant statiquement un domaine, à savoir le modèle des classes.
- Ce modèle ne permettait pas de modéliser l'aspect dynamique d'un domaine.
- Le modèle des états et le modèle d'interaction permettent la modélisation du point de vue dynamique et de modéliser l'évolution des objets au cours du temps.
- Le modèle des états comprend les diagrammes d'états-transitions.
- Le modèle d'interaction comprend les diagrammes de collaboration et les diagrammes de séquence.

Notes sur Diagrammes UML (suite)

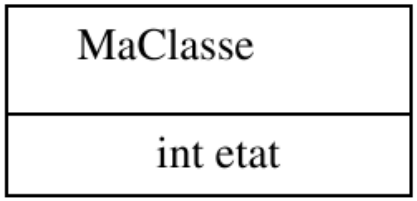
Etat

• Un objet possède un état à un instant donné.

• L'état de l'objet est une notion durable à l'échelle de temps d'évolution des objets.

Exemple : si l'objet est un dé à jouer, on peut dire que le dé possède 6 états possibles et que le dé posé sur la table est dans un état durable (tant qu'on ne le lance pas).

➥ L'état peut être spécifié explicitement :



• L'état peut aussi ne pas être explicitement présent sous forme d'un attribut.

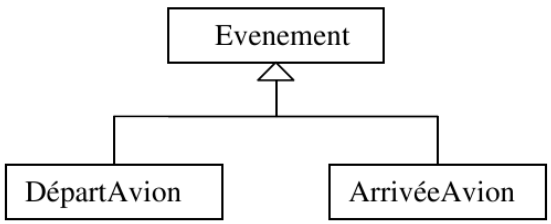
➥ Dans ce cas, l'état de l'objet au sens strict correspond à l'ensemble des attributs de l'objet.

➥ Et l'état au sens large correspond à l'ensemble des attributs et liens de l'objet.

Notes sur Diagrammes UML (suite)

Evénements

- Un événement est un *stimuli* externe ou interne à l'ensemble des objets.
- Il se produit instantanément à l'échelle d'évolution du système.
- Un événement est noté entre guillemets.
- Par exemple : *"Le vol AF-123 part de Chicago "*.
- Un événement peut être reçu ou envoyé par un objet.
- Quand l'objet est spécifié par le contexte, on note avec une flèche vers le bas un événement reçu par l'objet et une flèche vers le haut un événement envoyé par l'objet.
- Par exemple, voici un "événementReçu" ↓, et voici un "événementEnvoyé" ↑.
- Si l'objet n'est pas spécifié, les flèches n'ont pas de signification car un même événement peut être envoyé par un objet et reçu par un autre.
- Les événements peuvent être hiérarchisés dans un diagramme de généralisation.
- Par exemple :



Notes sur Diagrammes UML (suite)

Diagramme de séquence

• Pour commencer à décrire l'évolution d'un ensemble d'objets, il est possible de dessiner d'abord un diagramme de séquence.

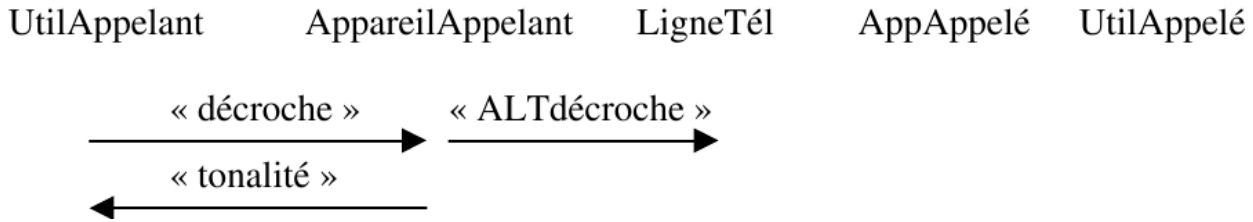
• Horizontalement, on place les instances concernées par un scénario et on relie les instances par des flèches indiquant le flux d'événements.

• Verticalement, le temps est représenté.

• Ce type de diagramme donne une première idée des événements qui pourront être pertinents dans la modélisation.

• On répète ce diagramme autant de fois qu'il existe de scénarii d'événements possibles

•**Exemple du téléphone :** dans le suivi d'événements ci-dessous, un utilisateur appelant décroche le téléphone qui envoie un signal sur la ligne téléphonique et la tonalité à l'utilisateur appelant :

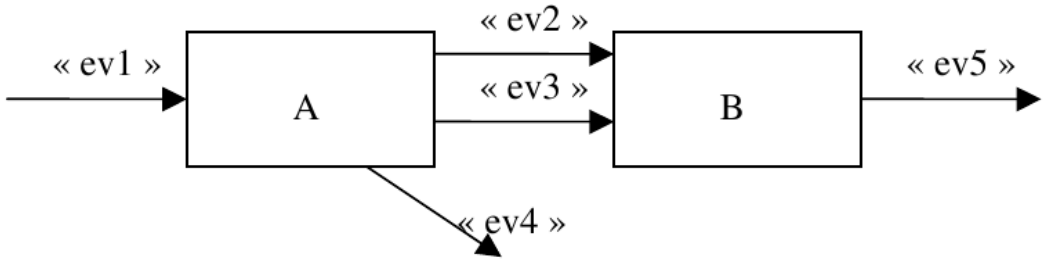


Notes sur Diagrammes UML (suite)

Diagramme de collaboration

• Un diagramme de flux ou diagramme de collaboration est un diagramme sur lequel les classes d'objets sont représentées avec des rectangles reliés par des flèches représentant les flux d'événements entre les classes.

Exemple :



Ici, l'événement " ev1 " est recevable par la classe 'A'.

• Les événements " ev2 ", " ev3 " et " ev4 " sont émissibles par 'A'.

• Les événements " ev2 " et " ev3 " sont recevable par la classe 'B'.

• Enfin, l'événement " ev5 " est émissible par la classe 'B'.

• Ce type de diagramme sert à répertorier tous les événements, reçus et envoyés, relatifs à chaque classe.

Notes sur Diagrammes UML (suite)

Diagramme d'état-transitions :

• Un diagramme d'état est un graphe dont les noeuds sont les valeurs possibles de l'état de l'objet et les arcs sont les transitions entre ces valeurs.

• Par abus de langage dans la suite du document, on dit que les noeuds du graphe sont les états de l'objet au lieu de dire les valeurs des états de l'objet.

• Le diagramme d'états est le diagramme que l'on cherche à obtenir lorsque l'on modélise la dynamique des objets.

• Il peut être obtenu soit directement dans les cas simples, soit indirectement dans les cas complexes, après avoir faits les diagrammes de suivi d'événements et/ou les diagrammes de flux d'événements.

• Sur un diagramme d'états, on place les événements reçus et envoyés par l'objet et les transitions associées.

• On place aussi les *actions* et les *activités*.

• Il existe plusieurs types d'actions : *entrée, sortie, interne ou de transition*.

Notes sur Diagrammes UML (suite)

Les actions et activités

- Une action est supposée avoir une durée nulle à l'échelle d'évolution des objets.
- Une activité par contre a une durée non nulle à l'échelle d'évolution des objets.
- Une action est effectuée à la suite de la réception d'un événement (notion instantanée) alors qu'une activité est effectuée pendant que l'objet est dans un état (notion durable).

- Une activité est spécifiée avec le mot-clé **do :** .

✓**Action d'entrée**

- Une action d'entrée est effectuée lorsque l'on rentre dans un état.
- Elle est annoncée avec le mot-clé **entry :** .

✓**Action de sortie**

- Une action de sortie est effectuée lorsque l'on sort d'un état.
- Elle est annoncée avec le mot-clé **exit :** .

Notes sur Diagrammes UML (suite)

✓**Action interne**

• Une action interne est effectuée lorsque l'objet reçoit un événement sans faire changer l'état de l'objet. Elle est annoncée avec le mot-clé **on** suivi du nom de l'événement entre guillemets. exemple : *on "evenement" :* .

✓**Action de transition**

• Une telle action est une action associée à une transition.

• Dans ce cas, une *condition* ou *garde* peut être associé(e) à la réception de l'événement associé à la transition.

• Le garde est spécifié entre crochets **[]** et l'action est précédée du **/**.

• Par exemple, dans la phrase suivante :
"Quand je sors le matin, si la température est glaciale, je mets mes gants.",

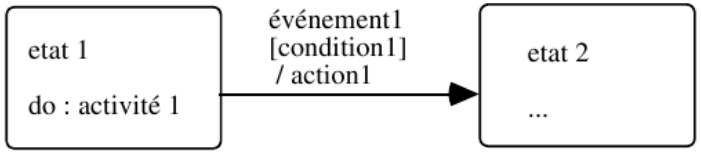
on a les correspondances suivantes :

"Quand je sors le matin,	(événement)
si la température est glaciale,	(condition)
je mets mes gants."	(action de transition)

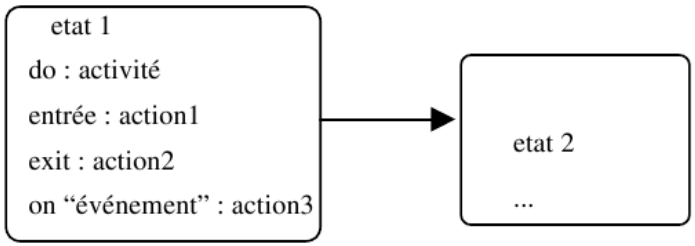
Notes sur Diagrammes UML (suite)

• Elle est formulée avec " événement " ↓ : [condition] / action

• Par exemple, le diagramme suivant exprime une transition entre deux états avec une activité 'activité1' associée à l'état 'état1', une action de transition 'action1', une condition 'condition1', sur réception de l'événement 'événement1'.



• Sur l'exemple suivant, l'activité 'activité' est effectuée pendant que l'objet est dans l'état 'état1' ; en entrée de l'état 'état1', l'action 'action1' est effectuée ; en sortie, l'action 'action2' est effectuée ; enfin, si l'événement 'événement' est recçu dans l'état 'état1' alors l'action interne 'action3' est effectuée.



Notes sur Diagrammes UML (suite)

• Une action interne est différente d'une transition réflexive.
• Une transition réflexive entraîne l'exécution de l'action de sortie de l'état juste avant et l'exécution de l'action d'entrée juste après.
• Lorsque plusieurs transitions arrivent sur un état, le choix du placement d'une action sur une transition ou bien en entrée de l'état doit être fait avec précision.
• De même, lorsque plusieurs transitions partent d'un état, le choix du placement d'une action sur une transition ou bien en sortie de l'état est déterminant.

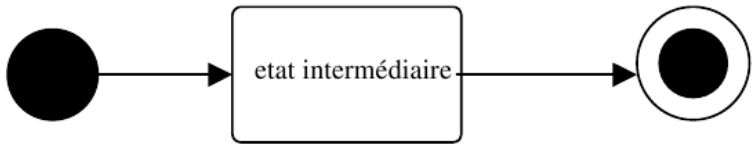
Transition automatique

• Une transition automatique est une transition qui se déclenche à la fin d'une activité de l'objet.
• Elle est dite automatique car elle se déclenche sans que l'objet ne reçoive d'événement.
• Une transition automatique peut avoir un garde et une action associés.

Notes sur Diagrammes UML (suite)

Etat initial et états finaux

• L'état initial d'un objet est spécifié avec un rond noir plein et les états finaux avec des ronds noirs encerclés.

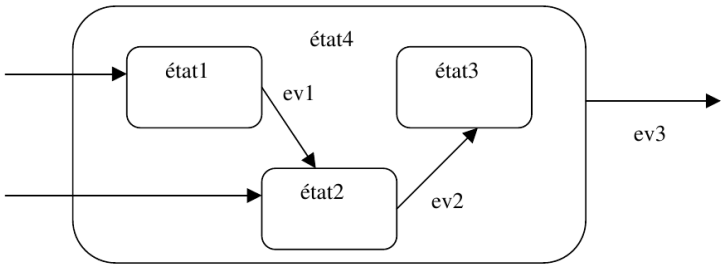


Notes sur Diagrammes UML (suite)

Hiérarchie d'états

• En cours de modéliation avec un diagramme d'états, il peut être judicieux de regrouper plusieurs états en un seul ou bien de décomposer un état en plusieurs sous-états.

• Exemple :



• Ici, l'état 'état4' est un sur-état des états 'état1', 'état2', 'état3'.

• Les deux transitions qui rentrent dans l'état 'état4' spécifient dans quel sous-état on arrive.

• La transition sortante de l'état 'état4' exprime que, quel que soit le sous-état dans lequel l'objet se trouve, si l'événement 'ev3' arrive, alors l'objet sort de l'état 'état4'.

• On peut bien sûr placer des actions d'entrée ou de sortie sur des sur-états ou des sous-états.

Notes sur Diagrammes UML (suite)

Relations entre le modèle objet le modèle dynamique

• Le modèle dynamique spécifie les séquences acceptables de modification d'objet.

• Le modèle dynamique d'une classe est hérité dans les sous-classes ; il est possible de redéfinir un diagramme d'états dans une sous-classe.

• Ce qui est naturel puisque un état d'objet est un attribut d'objet (donc héritable et redéfinissable).

• Toutes les classes ne nécessitent pas de diagramme d'états

• Selon les domaines à modéliser, on peut commencer ou non par faire les scénarios d'événements et les diagrammes de flux avant de faire les diagrammes d'états.

Rappel Sommaire

Le modèle des cas d'utilisation

Le modèle des cas d'utilisation

• Les *cas d'utilisation* (use cases) ont été formalisés par Ivar Jacobson.
Ils décrivent sous forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur.

• Avant UML, ils n'étaient pas formalisés par les autres méthodes objet telles que OMT.

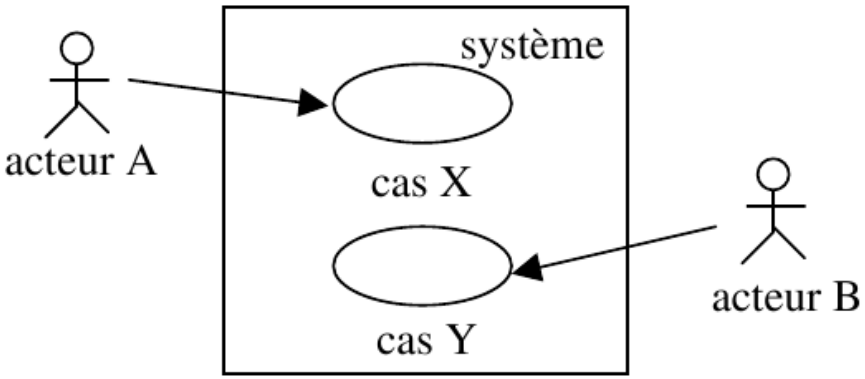
• Les cas d'utilisation sont utiles lors de l'élaboration du cahier des charges ou du document de spécifications des besoins du logiciel.

• Le *modèle des cas d'utilisation* comprend les *acteurs*, le *système* et les *cas d'utilisation*.

• L'ensemble des fonctionnalités du système est déterminé en examinant les besoins de chaque acteur, exprimés sous forme de famille d'interactions dans les cas d'utilisation.

• Les acteurs se représentent sous forme de petits personnages qui déclenchent les cas. Ces derniers se représentent par des ellipses contenues dans un rectangle représentant le système.

Le modèle des cas d'utilisation (suite)



Dans cet exemple, l'acteur A déclenche le cas X et l'acteur B déclenche le cas Y.

- Il existe quatre catégories d'acteurs :
  - les acteurs principaux,
  - les acteurs secondaires,
  - le matériel externe,
  - les autres systèmes.
- Chaque acteur doit être décrit en 3 ou 4 lignes de manière claire et concise.
- Un cas d'utilisation décrit un ensemble de scénarios du point de vue de l'utilisateur grâce a des diagrammes de séquence ou des diagrammes de collaboration.

Le modèle des cas d'utilisation (suite)

Les relations entre cas d'utilisation

Règles de mise en oeuvre

- La description du cas d'utilisation comprend les points suivants :
  - le début du cas exprimé par : " les cas débute quand X se produit ",
  - la fin du cas exprimé par : " les cas se termine quand Y se produit ",
  - l'interaction entre le système et les acteurs qui décrit clairement la frontière du système,
  - les échanges d'informations
  - la chronologie et l'origine des informations utilisant :
    * les diagrammes de séquence
    * ou les diagrammes d'activités.

../..

Le modèle des cas d'utilisation (suite)

- les répétitions de comportement du type (2 fromes d'itération) :

boucle
quelquechose
fin de boucle

pendant que
autre chose
fin pendant

- les situations optionnelles

Diagrammes d'activités

• Les diagrammes d'activités suivent les même formalisme que les diagrammes d'état-transitions sauf que les états sont remplacés par des activités, avec la possibilité pour les activités de se synchroniser.