Modélisation des comportements élémentaires

Cas d’utilisation

Cas d’utilisation

- **Un cas d'utilisation précise le comportement d'un système ou d'une partie d'un système et décrit un ensemble de séquences d'actions, y compris des variantes, comportant les variantes qu'un système exécute pour produire un résultat tangible pour un acteur.**
- **Les cas d'utilisation servent à saisir le comportement attendu d'un système en cours de développement, sans avoir à préciser la façon dont ce comportement est réalisé. Les cas d'utilisation permettent aux développeurs d'atteindre une compréhension commune avec les utilisateurs finaux du système et les experts du domaine.**
- **De plus, les cas d'utilisation aident à valider l'architecture et à suivre l'évolution du système au cours de son développement.**
- **Tout au long de l'implémentation du système, ces cas d'utilisation sont réalisés à l'aide de collaborations dont les éléments travaillent en commun pour faire fonctionner chaque cas d'utilisation.**
- **Les cas d'utilisation bien structurés représentent uniquement les comportements essentiels des systèmes ou des sous-systèmes et ne sont ni trop généraux, ni trop spécifiques.**

Cas d’utilisation (suite)

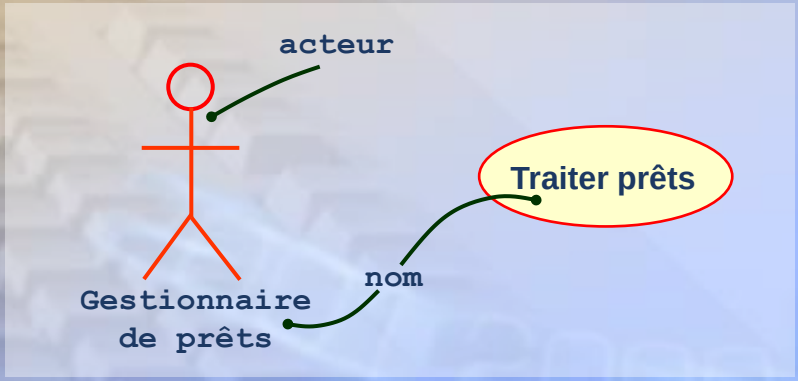
- **Un cas d'utilisation décrit un ensemble de séquences dans lequel chaque séquence représente l'interaction des éléments qui se trouvent à l'extérieur du système (ses acteurs) avec le système lui-même (et ses abstractions clés).**
- **Ces comportements sont en fait des fonctions système qui servent à visualiser, spécifier, construire, et documenter le comportement attendu du système pendant la saisie et l'analyse des exigences.**
- **Un cas d'utilisation représente une exigence fonctionnelle du système dans son ensemble.**
  - **Par exemple, pour une banque, un des cas d'utilisation classiques consiste à examiner des demandes de prêt.**
  - **Un cas d'utilisation implique l'interaction des acteurs et du système. Un acteur représente un ensemble cohérent de rôles interprétés par les utilisateurs de cas d'utilisation quand ils sont en interaction avec ces cas d'utilisation.**
  - **Les acteurs peuvent aussi bien être des êtres humains que des dispositifs automatisés.**

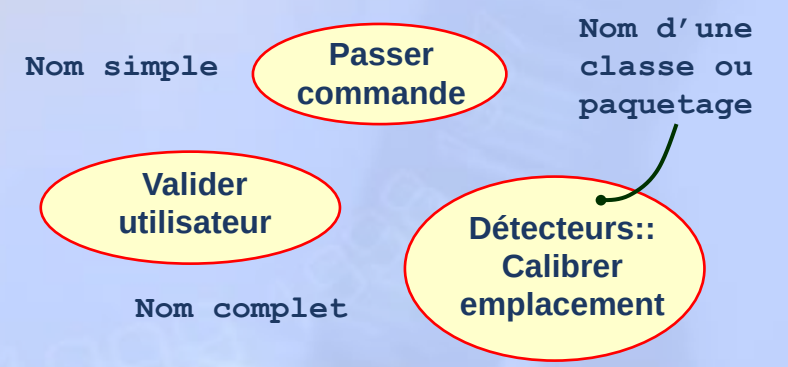
Cas d'utilisation (suite)

- Un cas d'utilisation comporte une quantité tangible de travail.
- Du point de vue d'un acteur donné, un cas d'utilisation remplit une fonction qui a une valeur pour l'acteur:
  - calculer un résultat, générer un nouvel objet ou modifier l'état d'un autre objet.
  - Pour reprendre l'exemple d'une banque, l'examen d'une demande de prêt aboutit à l'acceptation du prêt, qui se matérialise par une somme d'argent remise au client.
- On peut appliquer les cas d'utilisation à un système dans son intégralité.
- On peut également les appliquer à une partie d'un système, aux sous-systèmes ainsi qu'aux classes individuelles et aux interfaces.
- Dans chacune des situations, les cas d'utilisation représentent non seulement le comportement attendu de ces éléments, mais peuvent également servir de base aux cas de test pour ces éléments au fur et à mesure de leur évolution dans le cycle de développement.

Cas d'utilisation (suite)

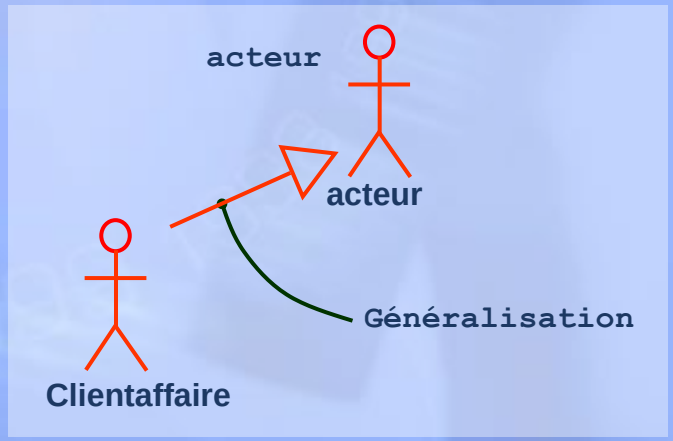
- Les cas d'utilisation appliqués aux sous-systèmes forment d'excellentes sources de tests de non-régression; les cas d'utilisation appliqués à un système dans son intégralité sont d'excellentes sources de tests d'intégration et de tests système.
- Cette représentation permet de visualiser un cas d'utilisation en dehors de sa réalisation et en contexte avec d'autres cas d'utilisation.





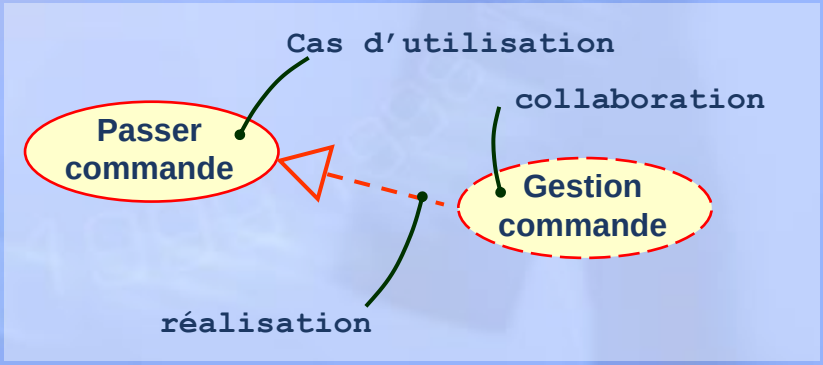
Les cas d'utilisations et le acteurs

- Un acteur représente un ensemble cohérent de rôles joués par les utilisateurs des cas d'utilisation en interaction avec ces cas d'utilisation.
- En règle générale, un acteur représente un rôle qu'un homme, une machine ou même un autre système joue avec le système.
  - Par exemple, une personne qui travaille pour une banque sera le `Gestionnaire de prêts`. Si elle a son compte dans la même banque, elle jouera aussi le rôle du `Client`.
  - Une instance d'acteur montre donc un individu en interaction avec le système d'une façon particulière. Même si on les utilise dans les modèles, les acteurs ne font pas partie intégrante du système puisqu'ils résident en dehors de celui-ci.
- On peut établir des catégories général d'acteurs et en suite utiliser les relations de généralisation pour les définir de manière plus précise



Cas d'utilisation et collaborations

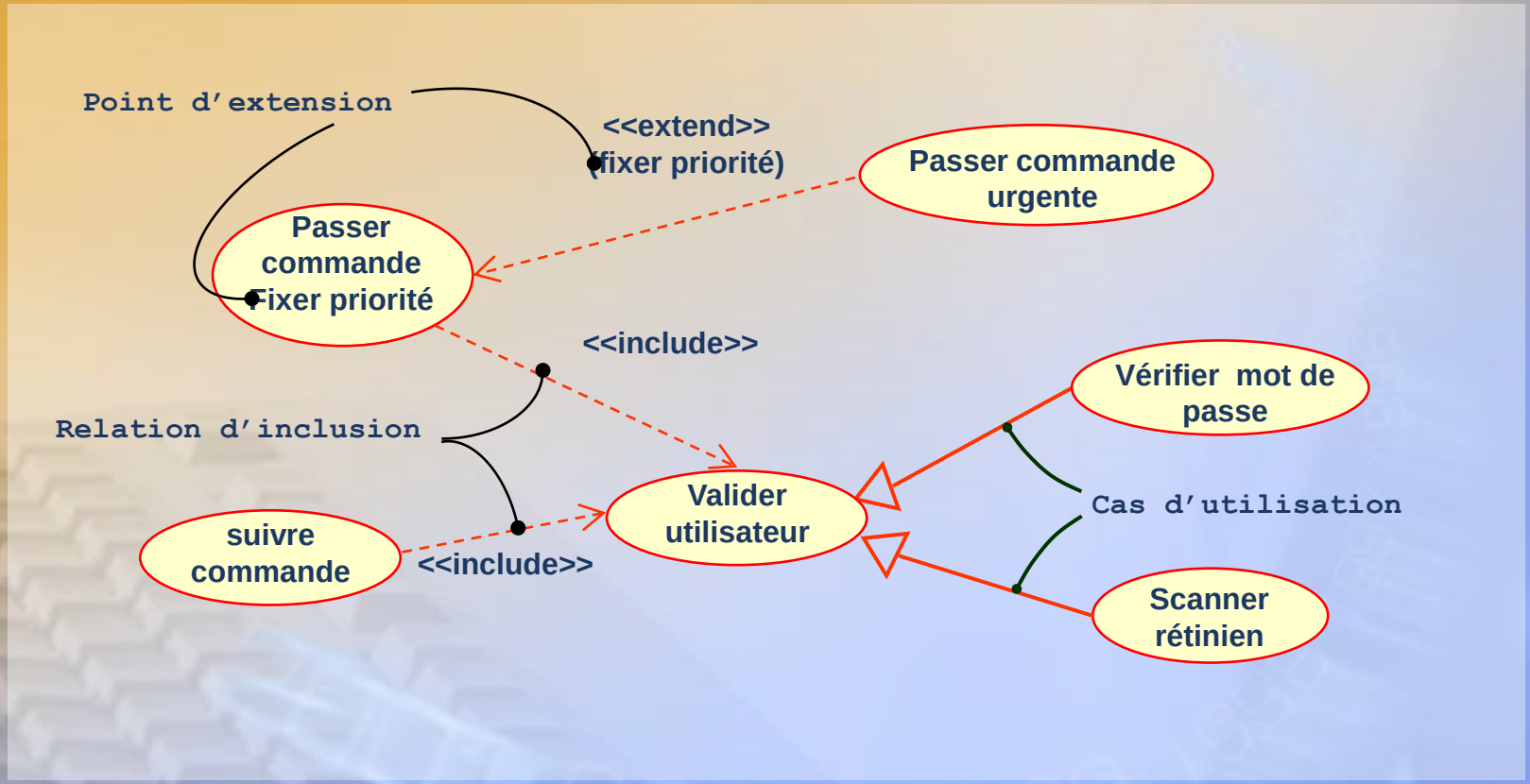
- Un cas d'utilisation saisit le comportement attendu du système (ou du sous-système, de la classe ou de l'interface) que l'on développe, sans que l'on ait besoin de préciser comment ce comportement est réalisé.
- Cette différenciation est importante car, dans la mesure du possible, l'analyse d'un système (qui précise le comportement) ne doit pas être influencée par des questions d'implémentation (qui précisent la façon dont le comportement est réalisé).
- Ensuite, il faut quand même implémenter les cas d'utilisation en créant une société de classes et d'autres éléments qui fonctionnent ensemble pour réaliser le comportement de ce cas d'utilisation. Cette société d'éléments, qui comporte à la fois une structure statique et dynamique, est modélisée en UML sous le nom de collaboration.



Organisation des cas d'utilisation

- On peut organiser les cas d'utilisation en les groupant en paquetages, de la même manière que l'on organise les classes.
- On peut aussi organiser les cas d'utilisation en précisant leurs relations mutuelles de généralisation, d'inclusion et d'extension.
- On applique ces relations pour factoriser le comportement commun, par extraction de ce comportement lorsqu'il est partagé par plusieurs cas d'utilisation et par insertion lorsqu'on veut étendre le comportement d'autres cas d'utilisation.
- La généralisation des cas d'utilisation est identique à celle des classes. Cela signifie que le cas d'utilisation enfant hérite du comportement et de la signification du cas d'utilisation parent.
- On traduit une relation d'extension par une dépendance à laquelle on a attribué le stéréotype extend.
- Les points d'extension du cas d'utilisation principal peuvent être classés dans un compartiment supplémentaire. Ces points d'extension ne sont que des étiquettes qui peuvent apparaître dans le flot du cas d'utilisation principal.

Généralisation, inclusion et extension



Relation d’inclusion

- **Une relation d'inclusion entre des cas d'utilisation signifie que le cas d'utilisation principal incorpore le comportement d'un autre cas d'utilisation en un point d'insertion bien déterminé.**
- **Le cas d'utilisation inclus n'est jamais tout seul, il fait toujours partie d'un cas d'utilisation qui l'englobe.**
- **L'inclusion peut être assimilée au cas d'utilisation principal qui tire un comportement du cas d'utilisation fournisseur**
- **On utilise une relation d'inclusion pour éviter de décrire le même flot d'événements plusieurs fois, et ce en rangeant le comportement commun dans un cas d'utilisation propre (le cas d'utilisation inclus dans le cas principal).**
- **La relation d'inclusion est avant tout un exemple de délégation.**

Relation d'extension

- **Une relation d'extension entre les cas d'utilisation signifie que le cas d'utilisation de base englobe implicitement le comportement d'un autre cas d'utilisation en un point spécifié indirectement par le cas d'utilisation qui étend le cas de base.**
- **Le cas d'utilisation de base peut exister tout seul, mais sous certaines conditions, son comportement peut être étendu au comportement d'un autre cas d'utilisation.**
- **Ce cas d'utilisation de base peut être étendu seulement à certains points, logiquement appelés points d'extension. L'extension peut être assimilée au cas d'utilisation étendu poussant un comportement vers le cas d'utilisation principal.**
- **On utilise une relation d'extension pour modéliser la partie d'un cas d'utilisation considérée par l'utilisateur comme facultative dans le comportement du système.**
- **Ainsi, on différencie le comportement facultatif du comportement obligatoire. On peut aussi utiliser une relation d'extension pour modéliser un flot secondaire qui n'est exécuté que dans certaines conditions. Enfin, on utilise une relation d'extension pour modéliser plusieurs flots, qui peuvent être insérés à un certain point, le tout étant géré par une interaction explicite avec un acteur.**

Diagrammes de cas d’utilisation

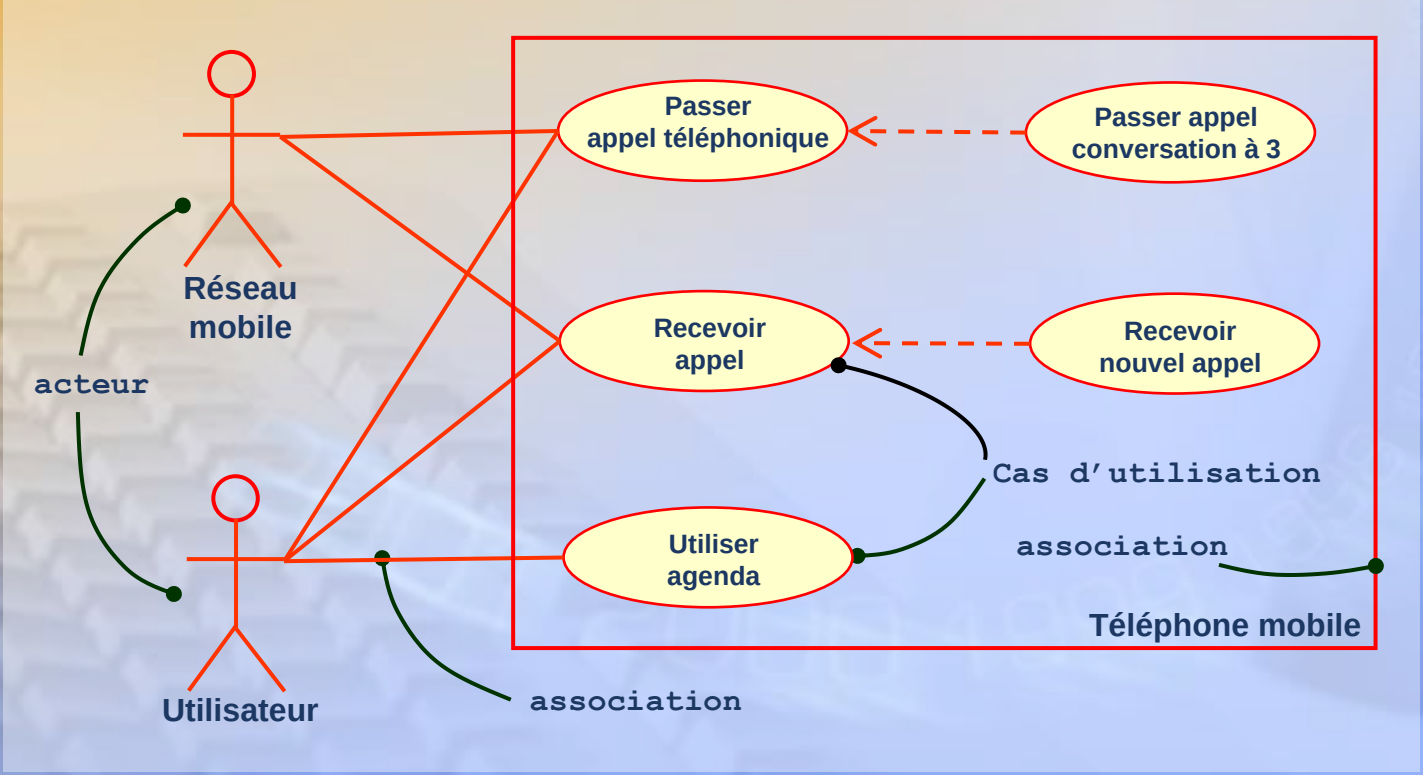
Modélisation des comportements élémentaires

Diagrammes de cas d’utilisation

- **Les diagrammes des cas d'utilisation font partie des cinq diagrammes d'UML qui servent à modéliser les aspects dynamiques des systèmes (les quatre autres sont: les diagrammes d'activités, les diagrammes d'états-transitions, les diagrammes de séquence et les diagrammes de collaboration).**
- **Les diagrammes de cas d'utilisation sont indispensables à la modélisation du comportement d'un système, d'un sous-système ou d'une classe.**
- **Tous montrent un ensemble de cas d'utilisation et d'acteurs ainsi que leurs relations.**
- **On applique les diagrammes de cas d'utilisation pour modéliser la vue des cas d'utilisation d'un système. Pour l'essentiel, cela implique la modélisation du contexte d'un système, d'un sous-système ou d'une classe, ou encore la modélisation des exigences sur le comportement de ces éléments.**
- **Ils rendent les systèmes, les sous-systèmes et les classes plus faciles à aborder et plus compréhensibles en présentant une vue externe de la façon dont ces éléments peuvent être utilisés en contexte.**
- **Ils sont également importants pour tester des systèmes exécutables grâce à l'ingénierie vers l'aval et pour comprendre des systèmes exécutables au moyen de la rétro-ingénierie.**

Représentation

- **Avec UML, on applique les diagrammes de cas d'utilisation pour visualiser le comportement d'un système, d'un sous-système ou d'une classe, de telle sorte que l'utilisateur puisse comprendre comment utiliser cet élément et que le développeur puisse l'implémenter.**



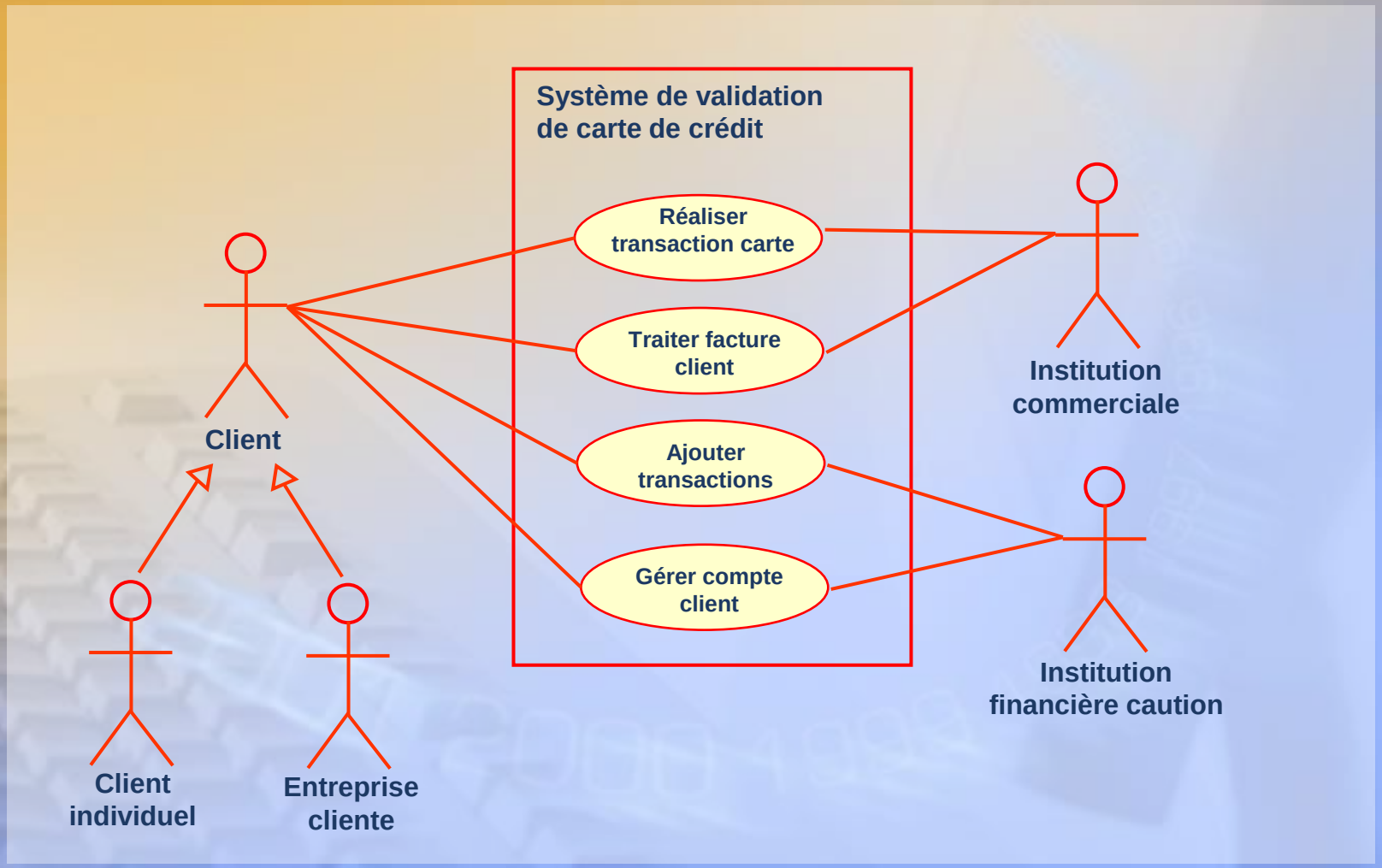
Contenus

- En règle générale, les diagrammes de cas d'utilisation contiennent:
  - des cas d'utilisation,
  - des acteurs,
  - des relations de dépendance,
  - de généralisation et d'association.
- Comme tous les autres diagrammes, les diagrammes de cas d'utilisation peuvent également contenir des notes et des contraintes.
- Les diagrammes de cas d'utilisation peuvent aussi inclure des paquetages qui servent à rassembler certains éléments du modèle en plus grands groupes.
- On peut également ajouter des instances de cas d'utilisation dans les diagrammes, plus particulièrement quand il s'agit de visualiser un système particulier en exécution.

Technique de modélisation

- **Pour modéliser le contexte d'un système, il faut:**
  - **identifier les acteurs qui entourent le système en définissant les groupes qui ont besoin de l'aide du système pour accomplir leurs tâches,**
    - **ceux qui sont nécessaires pour exécuter les fonctions du système,**
    - **ceux qui dialoguent avec le matériel ou d'autres systèmes externes**
    - **et enfin les groupes chargés des fonctions secondaires d'administration et de maintenance;**
- **organiser les acteurs semblables selon une hiérarchie de *généralisation/spécialisation.***
- **dans les cas où cela est nécessaire, attribuer un stéréotype à chaque acteur utile pour garantir une meilleure compréhension;**
- **intégrer ces acteurs au diagramme de cas d'utilisation et spécifier les chemins de communication entre chaque acteur et les cas d'utilisation du système.**

Modélisation du contexte d'un système



Modélisation des exigences d'un système

- Une exigence est une caractéristique de conception, une propriété ou un comportement d'un système.
- Lorsqu'on met en place les exigences d'un système, on définit les termes d'un **contrat**, entre les éléments qui existent en dehors du système et le système lui-même, contrat dans lequel on précise ce que l'on attend du système.
- En règle générale, on ne cherche pas à savoir comment le système agit, mais on veut s'assurer ***qu'il agit***. Un système qui se comporte correctement exécutera toutes ses fonctions fidèlement, de manière sûre et prévisible.
- Quand on construit un système, il est important de bien définir ce que l'on attend de lui, même si la compréhension des exigences évolue au fur et à mesure qu'on l'implémente de manière itérative et incrémentale.
- De même, quand on doit utiliser un système, il est indispensable de savoir comment il se comporte si l'on veut être capable de bien l'utiliser.
- Les exigences peuvent être exprimés sous plusieurs formes,
  - depuis le texte non structuré
  - jusqu'aux expressions en langage formel,
  - en passant par tout ce qui peut exister entre les deux.
- La plus grande partie des exigences fonctionnelles d'un système, pour ne pas dire l'intégralité, peut être exprimée sous forme de cas d'utilisation, et les diagrammes de cas d'utilisation d'UML sont essentiels à la gestion de ces exigences.
- Pour modéliser les exigences d'un système, il faut:
  - définir le contexte du système en identifiant les acteurs qui l'entourent;
  - pour l'ensemble des acteurs, tenir compte du comportement que chacun attend ou exige du système;
  - nommer ces comportements communs comme des cas d'utilisation;

Modélisation des exigences d'un système

- Pour modéliser les exigences d'un système, il faut:
  - définir le contexte du système en identifiant les acteurs qui l'entourent;
  - pour l'ensemble des acteurs, tenir compte du comportement que chacun attend ou exige du système;
  - nommer ces comportements communs comme des cas d'utilisation;
  - factoriser un comportement commun dans de nouveaux cas d'utilisation utilisés par les autres et factoriser les variations de comportement dans de nouveaux cas d'utilisation qui étendent les flots d'interaction principaux;
  - modéliser les cas d'utilisation, les acteurs et leurs relations dans un diagramme de cas d'utilisation;
  - décorer ces cas d'utilisation avec des notes qui font valoir des exigences non fonctionnelles. Certaines de ces notes s'appliquent au système entier.

Nota: Les cas d'utilisation ne bénéficie pas de d'ingénierie vers l'aval ni de la rétro-ingénierie

A vous de jouer

Expression graphique

- Exercice 1 : Le distributeur de billet
- Exercice 2 : Consultation médecin
- Exercice 3 : Connexion à un serveur

Expression littérale

- Exercice 4 : Magasin

Corrections exercices 1

Diagrammes d’activités

Modélisation des comportements élémentaires

Diagrammes d’activités

- Le diagramme d'activités fait partie des cinq diagrammes d'UML utilisés pour là modélisation des aspects dynamiques des systèmes.
- Un diagramme d'activités est principalement un organigramme qui montre le flot de contrôle d'une activité à l'autre.
- On utilise les diagrammes d'activités pour modéliser les aspects dynamiques d'un système.
  - Pour l'essentiel, cela implique la modélisation des étapes séquentielles (voire concurrentes) dans un processus de calcul.
- Un diagramme d'activités permet également de modéliser le flot d'un objet lorsqu'il passe d'un état à l'autre à différents points dans le flot de contrôle.
- Les diagrammes d'activités peuvent exister de manière indépendante pour visualiser, spécifier, construire et documenter la dynamique d'une société d'objets
  - mais peuvent aussi servir à modéliser le flot de contrôle d'une opération.
  - Alors que les diagrammes d'interaction mettent l'accent sur le flot de contrôle d'un objet à l'autre,
  - les diagrammes d'activités insistent sur le flot de contrôle d'une activité à l'autre.

Diagrammes d’activités (suite)

- **Qu’est ce une activité :**
  - **c’est une exécution non atomique en cour à l'intérieur d'un automate à états finis.**
- **Les activités finissent par produire une action, composée de calculs atomiques exécutables qui aboutissent à un changement dans l'état du système ou au retour d'une valeur.**
- **Les diagrammes d'activités sont importants non seulement pour modéliser les aspects dynamiques d'un système, mais également pour construire des systèmes exécutables grâce à l'ingénierie vers l'aval et à la rétro-ingénierie.**
- **Prenons par exemple le workflow lié à la construction d'une maison.**
  - **On commence par choisir un terrain,**
  - **puis on contacte un architecte pour,**
  - **concevoir la maison.**
  - **Une fois que les plans sont plus ou moins définitifs, le maître d'œuvre fait une proposition de prix.**
  - **Lorsqu'on est d'accord sur les plans et le prix, la construction peut commencer.**

Diagrammes d’activités (suite)

- On obtient les permis nécessaires,
- on creuse le sol,
- on coule les fondations,
- on monte la charpente, etc.
- jusqu'à ce que tout soit terminé.
- On reçoit alors les clefs, un certificat de conformité et la maison nous appartient.

- Même s'il s'agit d'une simplification à l'extrême de ce qui ce passe réellement lors du processus de construction d'une maison, elle permet de saisir le chemin principal du workflow.
- Dans la réalité, il existe de nombreuses activités parallèles dans des domaines très variés. Ainsi, les électriciens peuvent travailler en même temps que les plombiers et les charpentiers.

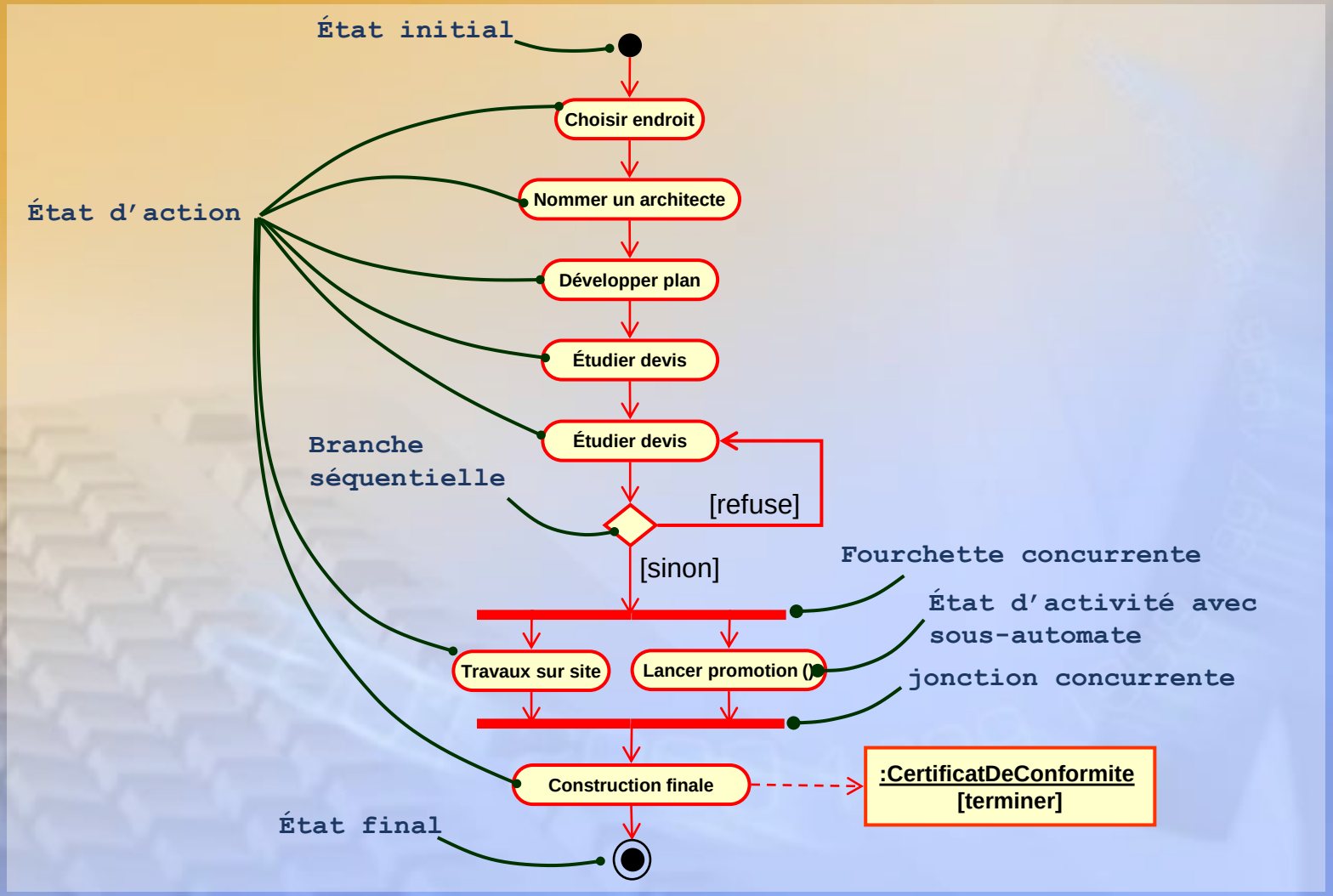
Diagrammes d'activités (suite)

- La modélisation de systèmes à forte composante logicielle pose le même type de problème. Comment modéliser au mieux un workflow ou une opération, qui sont tous deux des aspects dynamiques du système?
- La réponse consiste en deux options simples, semblables à l'utilisation des graphiques de Gantt et des graphiques de Pert.
  - D'une part, il est possible de construire des story-boards de scénarios, qui impliquent l'interaction de certains objets intéressants et des messages qui peuvent circuler entre eux.
  - En UML, on peut modéliser ces story-boards de deux façons:
    - en mettant en évidence le classement par ordre chronologique des messages (en utilisant des diagrammes de séquence)
    - ou en faisant ressortir les relations structurelles entre les objets en interaction (en utilisant les diagrammes de collaboration).
  - Ces diagrammes d'interaction ressemblent aux graphiques de Gantt: ils se concentrent sur les objets (ressources) qui exécutent une activité dans le temps.

Diagrammes d’activités (suite)

- D'autre part, on peut modéliser ces aspects dynamiques en utilisant les diagrammes d'activités, qui se concentrent d'abord sur les activités entre les objets. A cet égard, les diagrammes d'activités ressemblent aux graphiques de Pert.
- Un diagramme d'activités est essentiellement un organigramme qui met en évidence l'activité qui a lieu dans le temps.
- On peut considérer un diagramme d'activités comme un diagramme d'interaction qui aurait été inversé.
  - Un diagramme d'interaction s'intéresse aux objets qui transmettent des messages,
  - un diagramme d'activités aux opérations transmises entre les objets.
- La différence sémantique est subtile, mais cela donne deux visions du monde très différentes.

Construire sa maison



Résumons

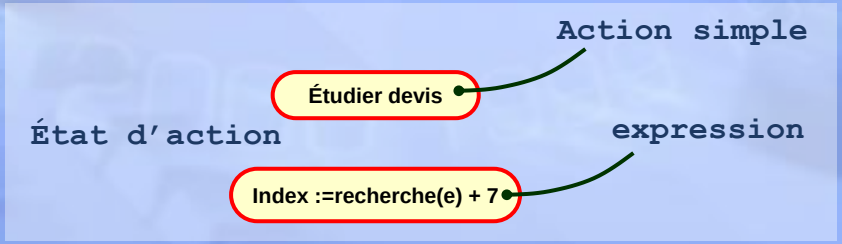
- Un diagramme d'activité représente le flot d'une activité à l'autre.
- Une activité est une exécution non atomique en cours à l'intérieur d'un automate à états finis.
- Les activité finissent par exécuter une *action,* composée de calculs atomiques exécutables qui aboutissent à un changement de l'état du système ou au retour d'une valeur.
- Les actions contiennent
  - l'appel d'une autre opération,
  - l'envoi d'un signal de création
  - ou le simple calcul, tel que l'évaluation d'une expression
- Un diagramme d'activités se composent d'un ensemble de sommets et d'arcs.

Propriétés communes

- **Un diagramme d'activités est un diagramme un peu particulier qui a néanmoins les mêmes propriétés que tous les autres diagrammes, à savoir un nom et un contenu graphique qui consistent en une projection dans un modèle.**
- **Mais il se distingue des autres diagrammes par son contenu spécifique.**
  - **des états d'activité et des états d'action,**
  - **des transitions,**
  - **des objets.**
- **Remarque: Un diagramme d'activités est un cas particulier d'automate à états finis dans lequel tous les états ou presque sont des états d'activité et dans lequel toutes les transitions ou presque sont déclenchées par l'achèvement des activités dans l'état source. Toutes les caractéristiques des automates à états finis s'appliquent à lui. Cela signifie que les diagrammes d'activités peuvent contenir des états simples et composés, des branches, des fourches des jonctions.**

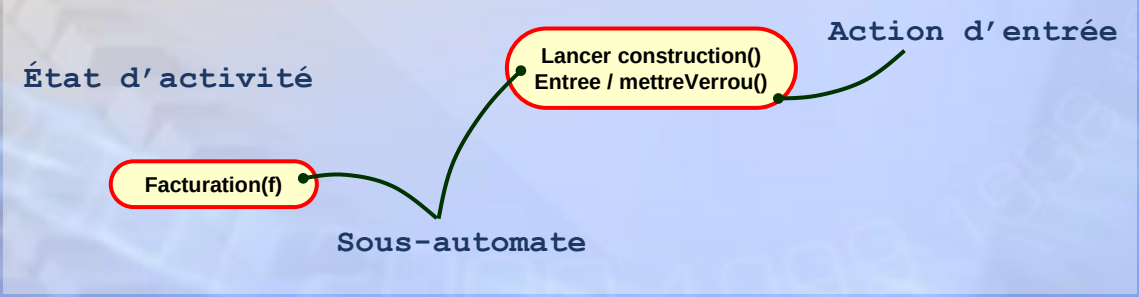
États d'actions et états d'activité

- **États d'actions**
  - Dans le flot de contrôle modélisé par un diagramme d'activités, certaines choses se produisent.
    - On peut évaluer une expression qui détermine la valeur d'un attribut ou qui envoie une valeur en retour
    - ou encore appeler une opération sur un objet, envoyer un signal à un objet voire créer ou détruire un objet.
  - Ces calculs exécutables atomiques sont appelés états d'action car ce sont des états du système qui représentent chacun l'exécution d'une action.
  - Un état d'action est représenté sous la forme d'un cartouche (deux lignes horizontales, en haut et en bas, et des côtés arrondis) dans lequel on peut écrire n'importe quelle expression.
  - Les états d'action ne peuvent pas être décomposés. De plus, ils sont atomiques, ce qui signifie que des événements peuvent se produire sans que le travail de l'état d'action soit interrompu.
  - Enfin, la durée d'exécution du travail d'un état d'action est considérée comme insignifiante.



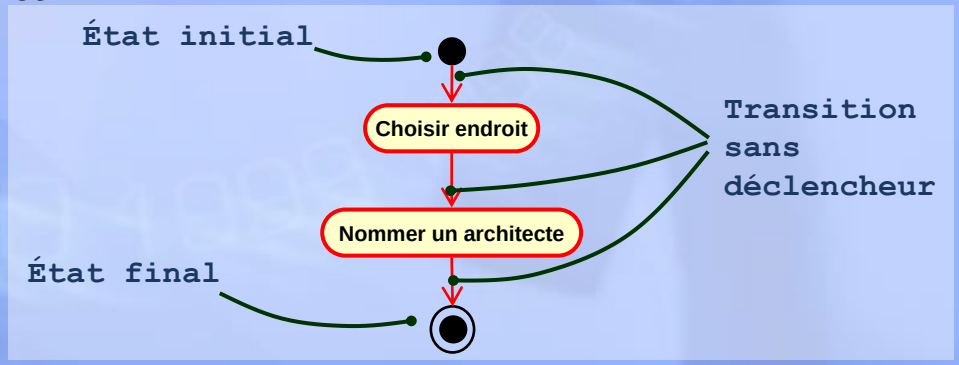
États d'actions et états d'activité

- **États activités**
  - En revanche, les états d'activité peuvent être décomposés de façon plus poussée car il est possible de représenter leur activité par d'autres diagrammes d'activité.
  - De plus, les états d'activité ne sont pas atomiques, ce qui signifie qu'ils peuvent être interrompus et qu'en règle générale leur temps d'exécution est relativement long.
    - On peut considérer qu'un état d'action est un cas spécial d'état d'activité.
    - Un état d'action est un état d'activité qui ne peut plus être décomposé.
    - De même, on peut comparer l'état d'activité à un composé dont le flot de contrôle est constitué d'autres états d'activité et d'états d'action.
    - Si l'on entre dans les détails d'un état d'activité, on trouve un autre diagramme d'activité.



Transitions

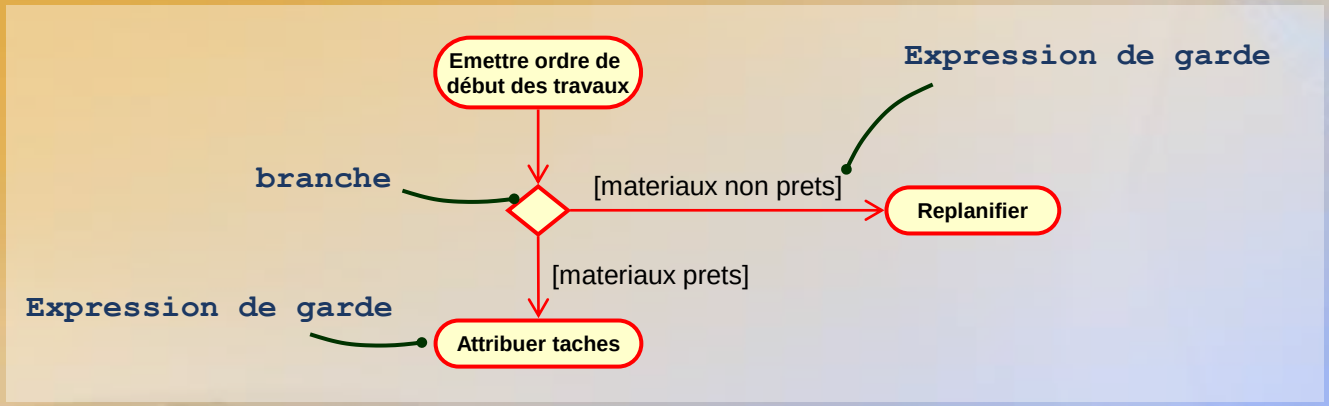
- **Lorsque l'action ou l'activité d'un état se termine, le flot de contrôle passe immédiatement à l'état d'action ou d'activité suivant.**
- **Ce flot est précisé à l'aide des transitions qui permettent d'indiquer le chemin d'un état d'action ou d'activité jusqu'à l'état d'action ou d'activité suivant.**
- **Remarque:**
  - **Du point de vue sémantique, on les appelle transitions sans déclencheur ou transitions de terminaison car le contrôle passe immédiatement une fois le travail de l'état source terminé. Le contrôle fait immédiatement suite à la transition et passe à l'état d'action ou d'activités suivant.**
  - **On déclenche alors (si nécessaire) l'action d'entrée dans cet état, puis on exécute l'action ou l'activité de l'état cible, en enchaînant sur la transition suivante une fois le travail de l'état terminé.**
  - **Ce flot de contrôle se poursuit indéfiniment dans le cas d'une activité sans fin ou s'arrête si on rencontre un état d'arrêt.**



Branchement conditionnel

- **Les transitions simples et séquentielles sont courantes, mais elles ne sont pas la seule sorte de chemin utilisé pour modéliser un flot de contrôle.**
- **Comme dans un organigramme, on peut inclure une branche, qui spécifie les différents chemins que l'on prend en se basant sur une expression booléenne.**
- **Une branche est représentée par un losange.**
- **Une branche peut avoir une transition entrante et deux transitions sortantes, voire plus.**
- **Sur chaque transition sortante, on place une expression booléenne, qui est évaluée seulement lorsqu'elle entre dans la branche.**
- **Pour toutes ces transitions, les dispositifs de garde doivent être mutuellement exclusifs (autrement le flot de contrôle serait ambigu), mais doivent couvrir toutes les possibilités (dans le cas contraire, le flot de contrôle serait gelé).**
- **Pour plus de facilité, on peut utiliser le mot clé *else* pour marquer une transition sortante, en représentant le chemin utilisé si aucune autre expression de garde n'est vraie.**

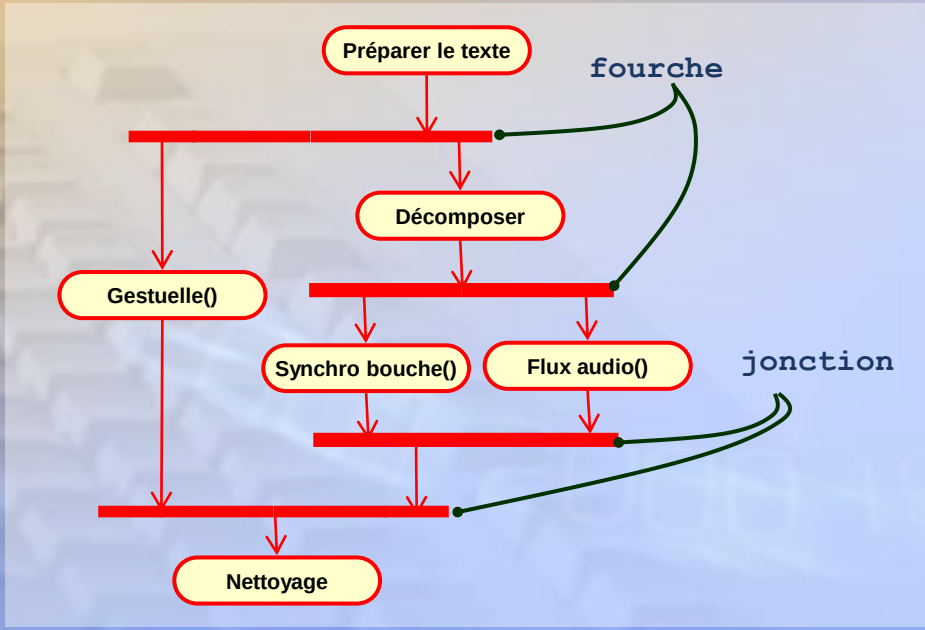
Branchement conditionnel (suite)



- On peut réaliser l'effet de l'itération en utilisant un état d'action qui initialise la valeur d'un itérateur, un autre état d'action qui incrémente l'itérateur et une branche chargée de juger si l'itération est terminée ou non.
- Remarque: UML n'impose pas de langage pour ces expressions. En théorie, il est possible d'utiliser simplement un texte structuré. En pratique, on utilise plutôt la syntaxe et la sémantique spécifiques à un langage de programmation donné.

Synchronisation de flots concurrents

- Dans les diagrammes d'activités, les transitions séquentielles simples et arborescentes sont les chemins les plus courants.
- Cependant, il arrive que l'on rencontre des flots concurrents, surtout lorsqu'on modélise des systèmes de workflows de processus métier.
- En UML, on utilise une barre de synchronisation pour spécifier la division et le regroupement de ces flots de contrôle parallèles. En règle générale une barre de synchronisation est représentée par une ligne épaisse horizontale ou verticale.

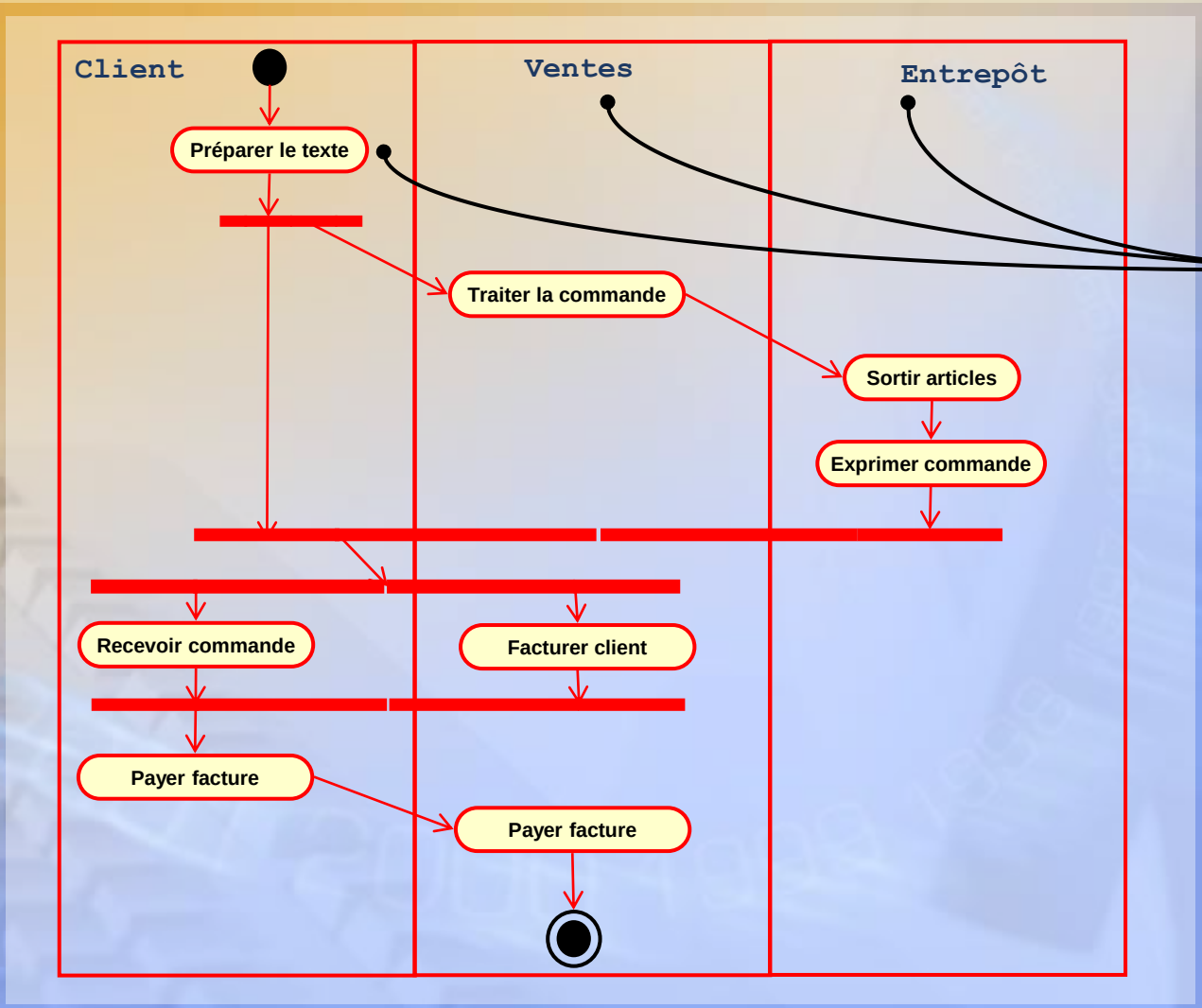


Remarque:
Les jonctions et les fourches doivent s'équilibrer, c'est-à-dire que le nombre de flots qui sortent d'une fourche doit correspondre au nombre de flots qui entrent dans la jonction correspondante.
De plus, les activités des flots de contrôle parallèles peuvent communiquer entre elles en s'envoyant des signaux.

Travées

- **Les travées *(swimlanes)* sont très utiles, lors de la modélisation des workflows de processus métier afin de diviser les états d'activités en groupes sur un diagramme d'activités, chaque groupe représentant le département responsable des ces activités.**
- **En UML, chaque groupe s'appelle une travée, il est visuellement séparé de son voisin par une ligne verticale pleine. Une travée définit un lieu géométrique d'activités.**
- **Chaque travée possède un nom, unique dans le diagramme. Une travée n'a pas une sémantique vraiment poussée, mais elle peut représenter une entité qui existe dans la réalité.**
- **Elle représente une responsabilité de haut niveau pour une partie de l'activité globale d'un diagramme d'activités et peut être implémentée par la suite par une ou plusieurs classes.**
- **Dans un diagramme d'activités divisé en travées, chaque activité appartient à une seule travée, même si les transitions peuvent passer d'une travée à l'autre.**

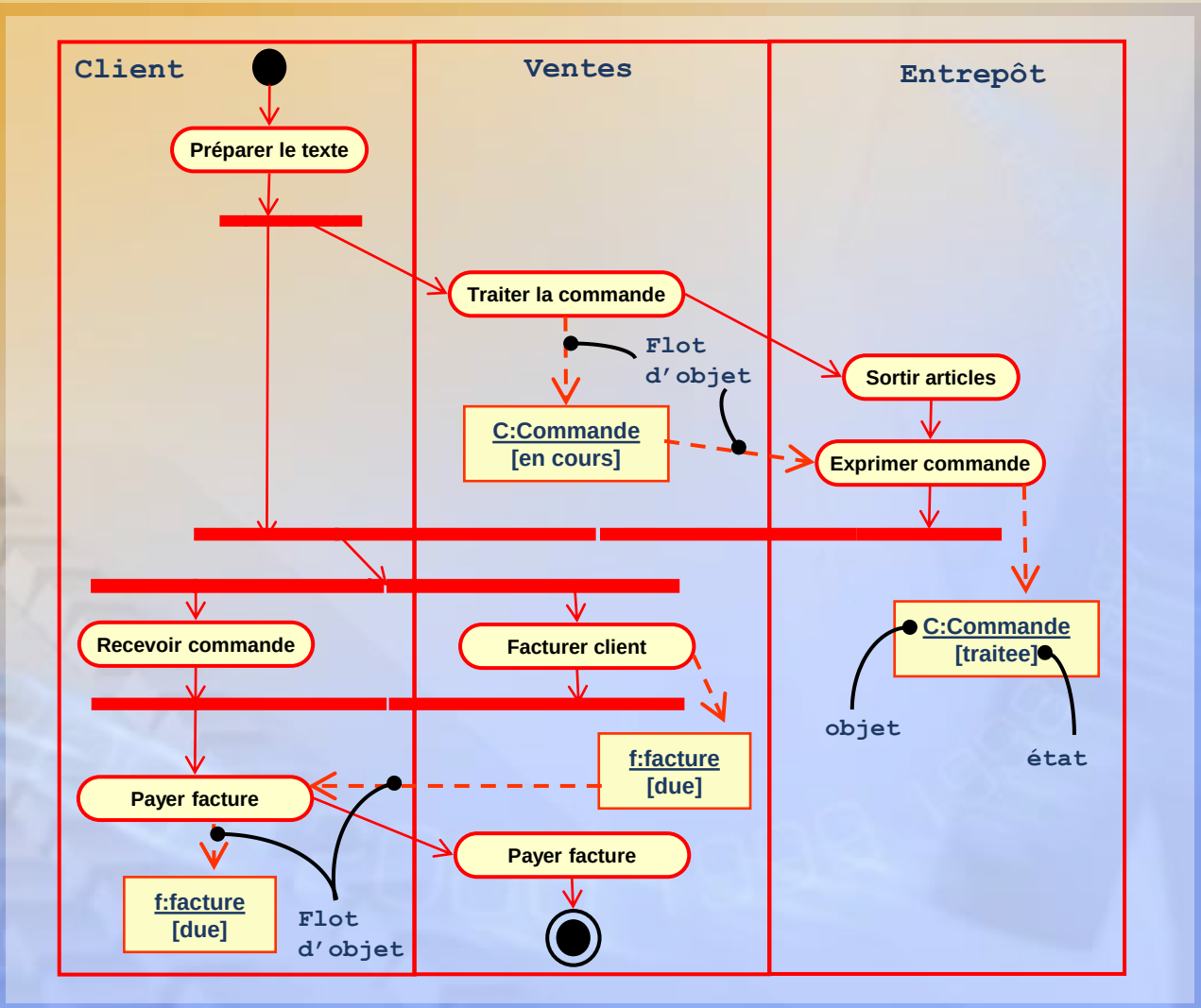
Travées (représentation)



Flot d’objet

- **Les objets peuvent être impliqués dans le flot de contrôle associé à un diagramme d'activités.**
- **En UML, on peut non seulement montrer le flot d'un objet à travers un diagramme d'activités, mais en plus on peut montrer comment son rôle, son état et ses valeurs d'attributs sont modifiés.**
- **De même, la valeur des attributs d'un objet est représentée dans un compartiment placé sous le nom de l'objet.**

Flot d'objet (représentation)



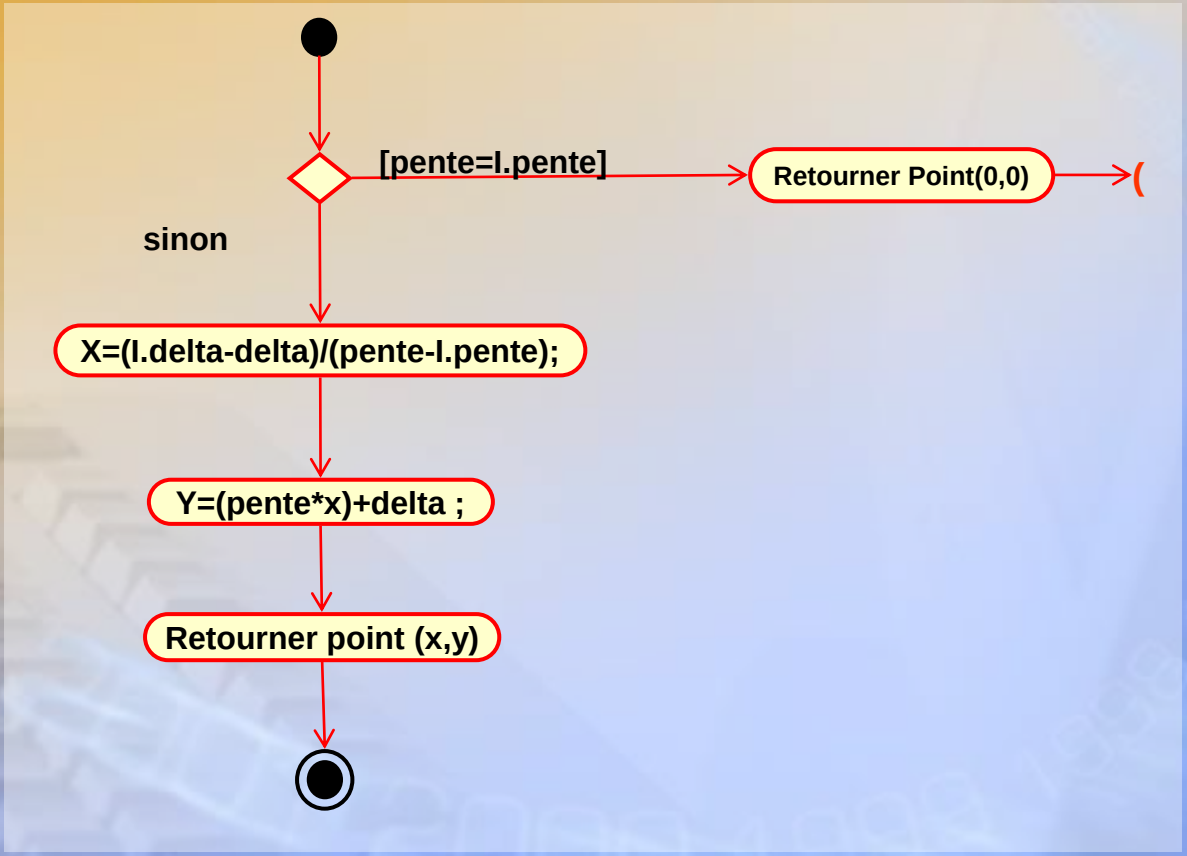
Modélisation d'un workflow

- **Pour modéliser un workflow, il faut:**
  - **déterminer un point central pour le workflow.**
    - **Pour les systèmes importants, il est impossible de représenter dans un seul diagramme tous les flots de travail intéressants;**
  - **sélectionner les objets métier qui ont des responsabilités de haut niveau pour certaines parties du workflow global.**
    - **Il peut s'agir d'objets réels qui font partie du vocabulaire du système ou au contraire d'objets abstraits.**
    - **Dans chaque cas, il faut créer une travée pour chaque objet métier important;**
  - **identifier les préconditions de l'état initial du workflow et les postconditions de son état final, cela afin d'aider à la modélisation de son périmètre;**
  - **en partant de l'état initial du workflow, préciser les activités et les actions qui ont lieu dans le temps et les représenter dans le diagramme d'activités soit comme des états d'activité, soit comme des états d'action;**
  - **réduire les actions compliquées ou les groupes d'actions qui apparaissent de façon répétée en les transformant en états d'activité et proposer un diagramme d'activités à part qui développe chacun d'eux;**
  - **représenter les transitions qui connectent ces états d'activité et d'action en commençant d'abord par les flots séquentiels du workflow, puis en continuant avec les branchements conditionnels et enfin avec la synchronisation des flots concurrents;**
  - **dans le diagramme d'activités, faire figurer également les objets importants impliqués dans le workflow.**
    - **Représenter leurs changements de valeurs et d'état de façon à communiquer le but du flot d'objet.**

Modélisation d'une opération

- On peut attacher des diagrammes d'activités à des classes, des interfaces, des composants, des nœuds, des cas d'utilisation et des collaborations, mais l'élément le plus courant auquel on peut attacher un diagramme d'activités est l'opération.
- Ainsi utilisé, un diagramme d'activités est simplement un organigramme des actions d'une opération.
- Le principal avantage d'un diagramme d'activités réside dans le fait que tous les éléments qu'il contient sont sémantiquement liés à un modèle sous-jacent riche.

Modélisation d'une opération



A vous de jouer

Exercice 1 : Cafetière

Exercice 2 : Cafetière

En bref pour construire un diagramme d'activité

- Identifiez la portée (« scope ») du diagramme d'activité
  Commencez en identifiant ce que vous allez modéliser. Un seul use case? Une partie d'un use case ? Un « workflow » qui inclut plusieurs use cases ? Une méthode de classe ?
- Ajouter l'état de *départ* et de *terminaison*
- Ajouter les activités
  Si vous modélisez un use case, introduisez une activité pour chaque use case principal. Si vous modélisez un « workflow », introduisez une activité pour chaque processus principal, souvent un use case. Enfin, si vous modélisez une méthode, il est souvent nécessaire d'avoir une activité pour chaque grand étape de la méthode.
- Ajouter des transitions (séquentielles), des transitions alternatives (conditionelles), des synchronisations entre des activités, des itérations.
- Identifier des swimlanes et répartir des activités identifiées dans ces swimlanes.

Corrections exercices 1

Plus d'information …

- http://www.rational.com/products/rup/
- http://www.therationaledge.com
- http://www.ambysoft.com/
- http://www.ronin-intl.com/publications/unifiedProcess.html

Dialogue