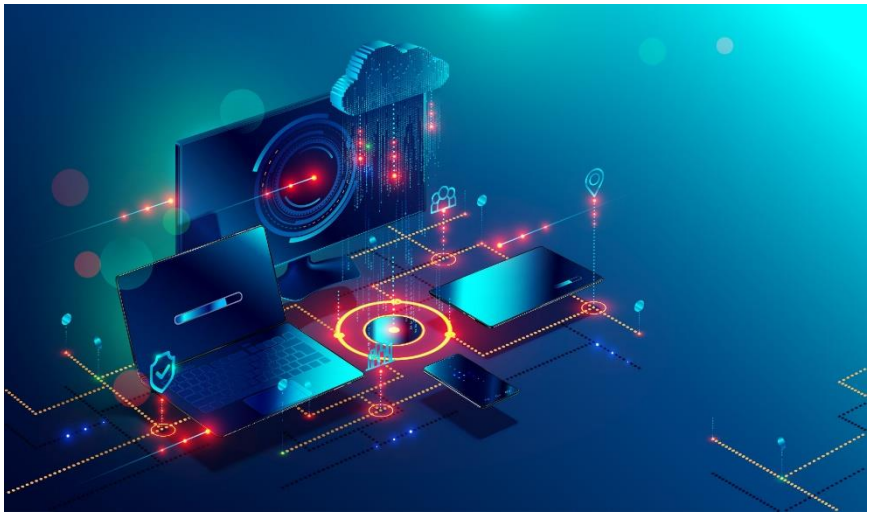
Analyse Orientée Objet

I. Introduction

II. Approche objet et système d'information.

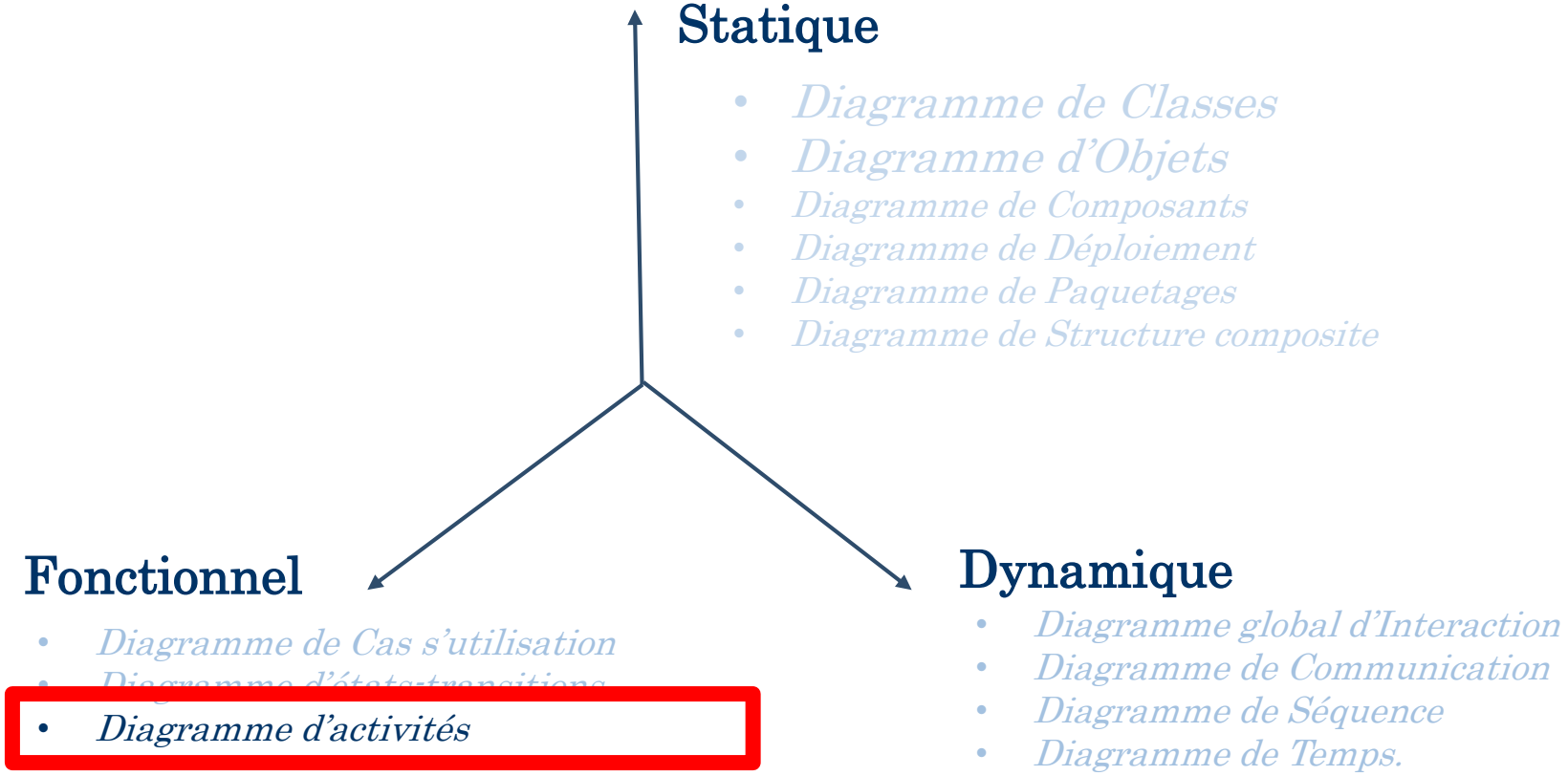
III. Principes Objet

IV. UML



IV. UML

3. Les diagrammes UML.



IV. UML

Diagramme d'activités

1. Introduction.
2. Action.
3. Activité.
4. Nœuds d'activité.
5. Partitions.

IV. UML

Diagramme d'activités

1. Introduction.
2. Action.
3. Activité.
4. Nœuds d'activité.
5. Partitions.

IV. UML
Diagramme d'activités

1. Introduction.

Les diagrammes d'activités d'UML offrent une manière graphique et non ambiguë pour modéliser les traitements.

Permettent alors de représenter graphiquement:

- Le comportement d'une méthode.
- Le déroulement d'un cas d'utilisation.

Ces diagrammes sont semblables aux diagrammes d'états-transitions mais avec une interprétation différente.

1. Introduction.

Ces diagrammes sont semblables aux diagrammes d'états-transitions mais avec une interprétation différente.

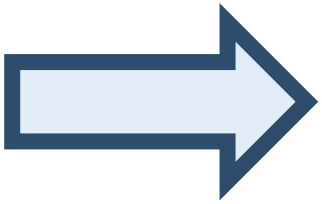
Diagrammes d'états-transitions:
Sont définis pour chaque classeur(un seul), et peuvent être complétés par exemple par des diagrammes de séquences.

Diagrammes d'activités:
Ne sont pas spécifiquement rattachés à un classeur particulier. Ils peuvent être rattachés à n'importe quel élément de modélisation afin de visualiser, construire ou documenter le comportement de cet élément.

1. Introduction.

- Les **diagrammes d'activités** permettent d'illustrer les **descriptions textuelles des cas d'utilisation** par une représentation sous forme d'organigrammes (facile et plus accessible).

- Un **diagramme des activités** donne une représentation des **activités** telles que **les voient les acteurs qui collaborent avec le système** dans le cadre d'un processus métier.



Permet de spécifier des **traitements séquentiels** donc donne une **vision très proche** de celle des **langages de programmation** comme le C++ ou Java.

IV. UML

Diagramme d'activités

1. Introduction.
2. Action.
3. Activité.
4. Nœuds d'activité.
5. Partitions.

2. Action.

1. Définition.

2. Les différents types d'actions.

2. Action.

1. Définition.

Définition:

Une **action** est le plus **petit traitement** qui puisse être exprimé en UML.
La notion d'action = la notion d'instruction élémentaire d'un langage de programmation (C++ ou Java).

2. Action.

1. Définition.

Exemple d'**action:**

- *Une affectation de valeur à des attributs,*
- *Création d'un nouvel objet,*
- *Un calcul arithmétique simple,*
- *L'émission d'un signal,*
- *La réception d'un signal,*
- *Etc.*

2. Action.

1. Définition.

2. Les différents types d'actions.

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- *Action accepter événement (eccept event),*
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- *Action accepter événement (eccept event),*
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- ***Action appeler (call operation),***

*Correspond à l'**invocation d'une opération** sur un objet de manière **synchrone** ou **asynchrone**.*
Lorsque l'action est exécutée, les paramètres sont transmis à l'objet cible.

* *Si l'appel est asynchrone, l'action est terminée et les valeurs de retour seront ignorées.*
* *Si l'appel est synchrone, l'appelant est bloqué pendant l'exécution de l'opération et les valeurs de retours (s'il y en a) seront réceptionnées.*

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- ***Action comportement (call behavior),***
- *Action envoyer (send),*
- *Action accepter événement (eccept event),*
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- ***Action comportement (call behavior),***

C'est une variante de l'action appeler (call operation), par contre elle invoque directement une ***activité*** *plutôt qu'une opération.*

Cette action offre la ***possibilité****, dans les diagrammes d'activités, de* ***directement se référer*** *à d'autres* ***types de diagrammes.***

Exemple: *Utiliser un diagramme de séquence imbriqué dans un diagramme d'activités pour illustrer le comportement d'une activité, et inversement.*

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- ***Action envoyer (send),***
- *Action accepter évènement (eccept event),*
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- ***Action envoyer (send),***

Cette action permet de transmettre un message ***asynchrone*** *à un objet cible où elle peut* ***déclencher un comportement****.*
L'objet appelant ***poursuit son exécution*** *sans attendre que l'entité cible ait bien reçu le message.*

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- ***Action accepter événement (eccept event),***
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- ***Action accepter événement (accept event),***

L'exécution de cette action interrompt l'exécution en cours jusqu'à la réception du type d'événement spécifié, qui est généralement un signal.
Cette action est utilisée pour la réception de signaux asynchrones.

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- *Action accepter événement (accept event),*
- ***Action accepter appel (accept call),***
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- ***Action accepter appel (accept call),***

Cette action est une variante de l'action accepter événement (accept event) pour les appels synchrones.

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- *Action accepter événement (eccept event),*
- *Action accepter appel (accept call),*
- ***Action répondre (reply),***
- *Action créer (create),*
- *Action détruire (destory),*
- *Action lever exception (raise exception).*
- *Exemple.*

2. Action.

2. Les différents types d'actions.

- ***Action répondre (reply)***

Cette action permet de transmettre un message en réponse à la réception d'une action de type accepter appel (accept call).
Reply *correspond précisément au* ***return*** *des langages de programmation.*

2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- *Action accepter événement (eccept event),*
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- ***Action créer (create),***
- ***Action détruire (destory),***
- ***Action lever exception (raise exception)***
- *Exemple:*

2. Action.

2. Les différents types d'actions.

- ***Action créer (create)***

Cette action permet d'instancier un objet.

- ***Action détruire (destory)***

Cette action permet de détruire un objet.

- ***Action lever exception (raise exception)***

Cette action permet de lever explicitement une exception.

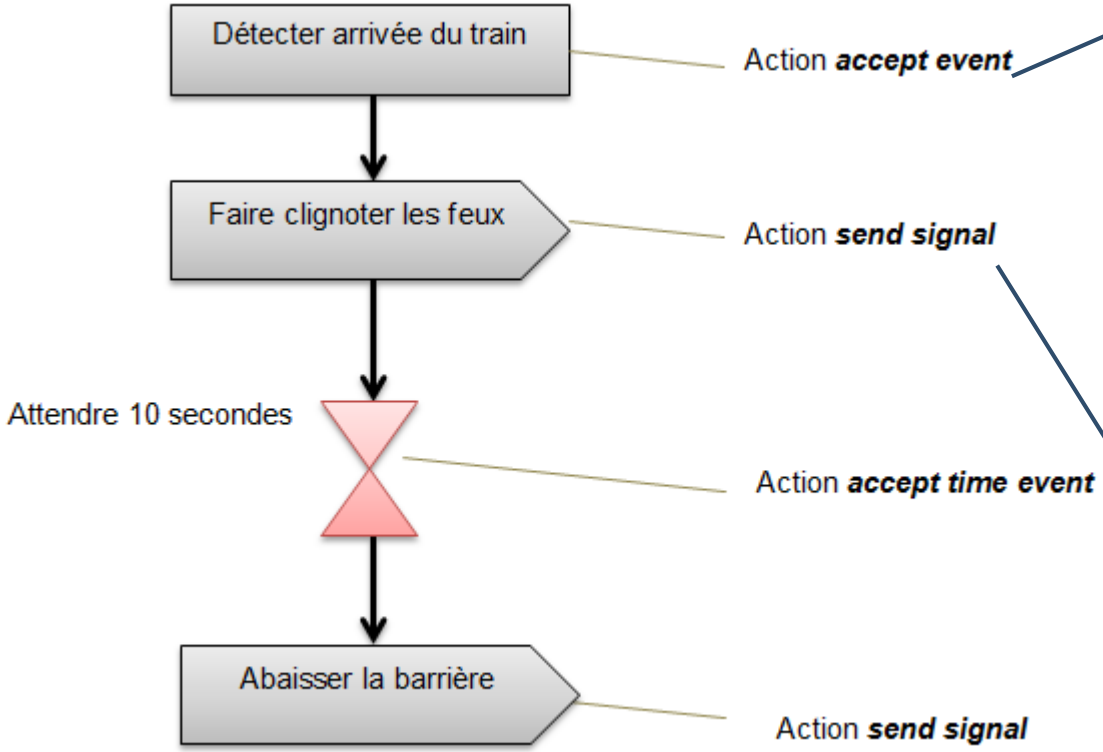
2. Action.

2. Les différents types d'actions.

- *Action appeler (call operation),*
- *Action comportement (call behavior),*
- *Action envoyer (send),*
- *Action accepter événement (eccept event),*
- *Action accepter appel (accept call),*
- *Action répondre (reply),*
- *Action créer (create),*
- *Action détruire (destory),*
- ***Action lever exception (raise exception).***
- ***Exemple.***

2. Action.

Exemple

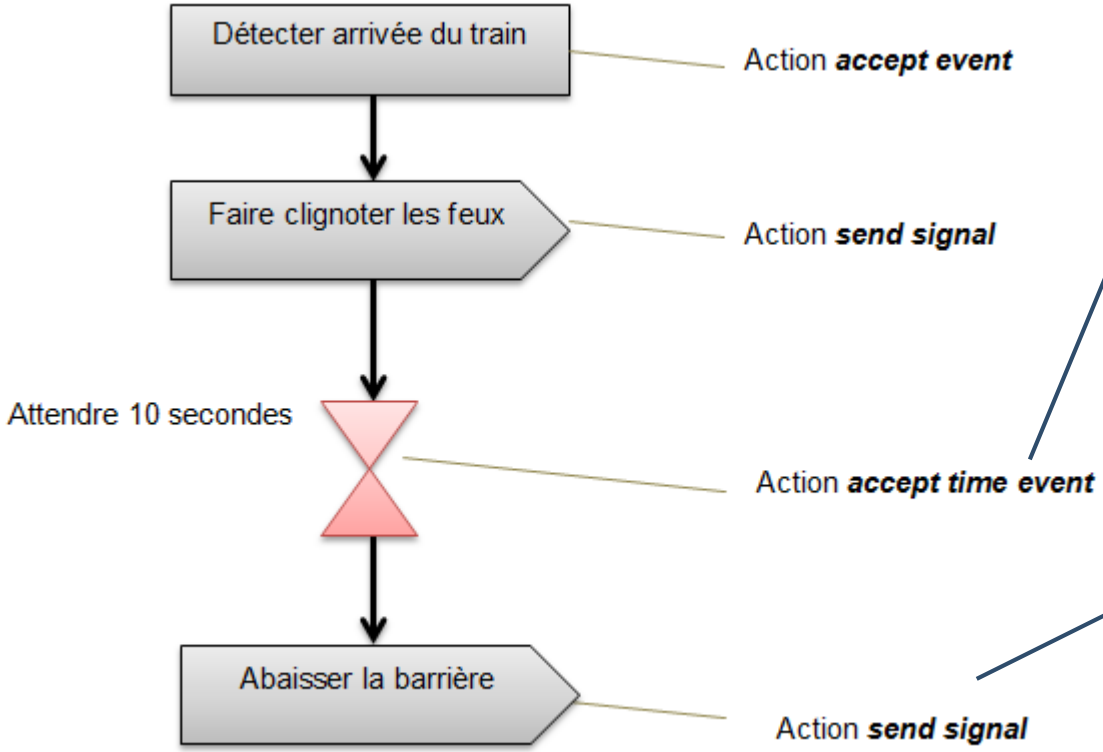


Accept event:
Détecte l'arrivée du train. On reçoit le signal de l'arrivée du train.

Send signal:
Envoi un signal « faire clignoter les feux », qui est transmis à un objet cible sans attendre que ce dernier ait bien reçu le signal..

2. Action.

Exemple



Accept time event:
Un événement temporel déclenché après l'écoulement d'une certaine durée (10 sec.)

Send signal:
Envoi un signal « Abaisser la barrière », un message envoyé et transmis à la cible.

IV. UML

Diagramme d'activités

1. Introduction.
2. Action.
3. Activité.
4. Nœuds d'activité.
5. Partitions.

3. Activité (activity).

- Une **activité** définit un comportement décrit par un séquencement organisé d'unités dont les **éléments** simples sont **les actions**.

- ***Une activité*** *est donc une série* ***d'actions.***

Représentation graphique:

3. Activité (activity).

Exemples d'activités:

- Les activités possibles pour le fonctionnement d'une **borne bancaire**.

Insérer la carte	Saisir code	Choisir opération
Saisir montant	Choisir compte	.....

IV. UML
Diagramme d'activités

1. Introduction.
2. Action.
3. Activité.
4. Nœuds d'activité.
5. Partitions.

4. Nœuds d'activité

1. Définition.
2. Les familles de nœuds d'activité.
3. Nœud d'activité structuré.

4. Nœuds d'activité

1. Définition.
2. Les familles de nœuds d'activité.
3. Nœud d'activité structuré.

4. Nœuds d'activité:

Définition:
Un nœud d'activité est un type d'élément abstrait permettant de représenter les étapes le long du flot d'une activité.

Trois familles de nœuds d'activité:

1. Les nœuds exécutable (executable node) et nœuds d'action (action node).
2. Les nœuds de contrôle (control node).
3. Les nœuds d'objet (object node).

4. Nœuds d'activité:

Représentations graphiques:

Nœud d'action

Nœud d'objet

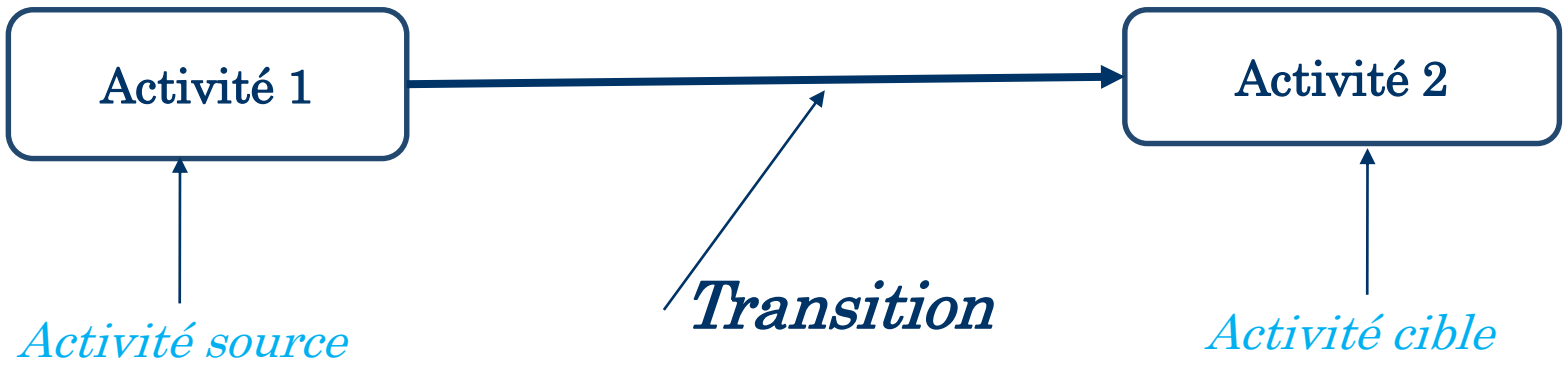
Représentations des nœuds de contrôle:

4. Nœuds d'activité:

Rappel: Transition

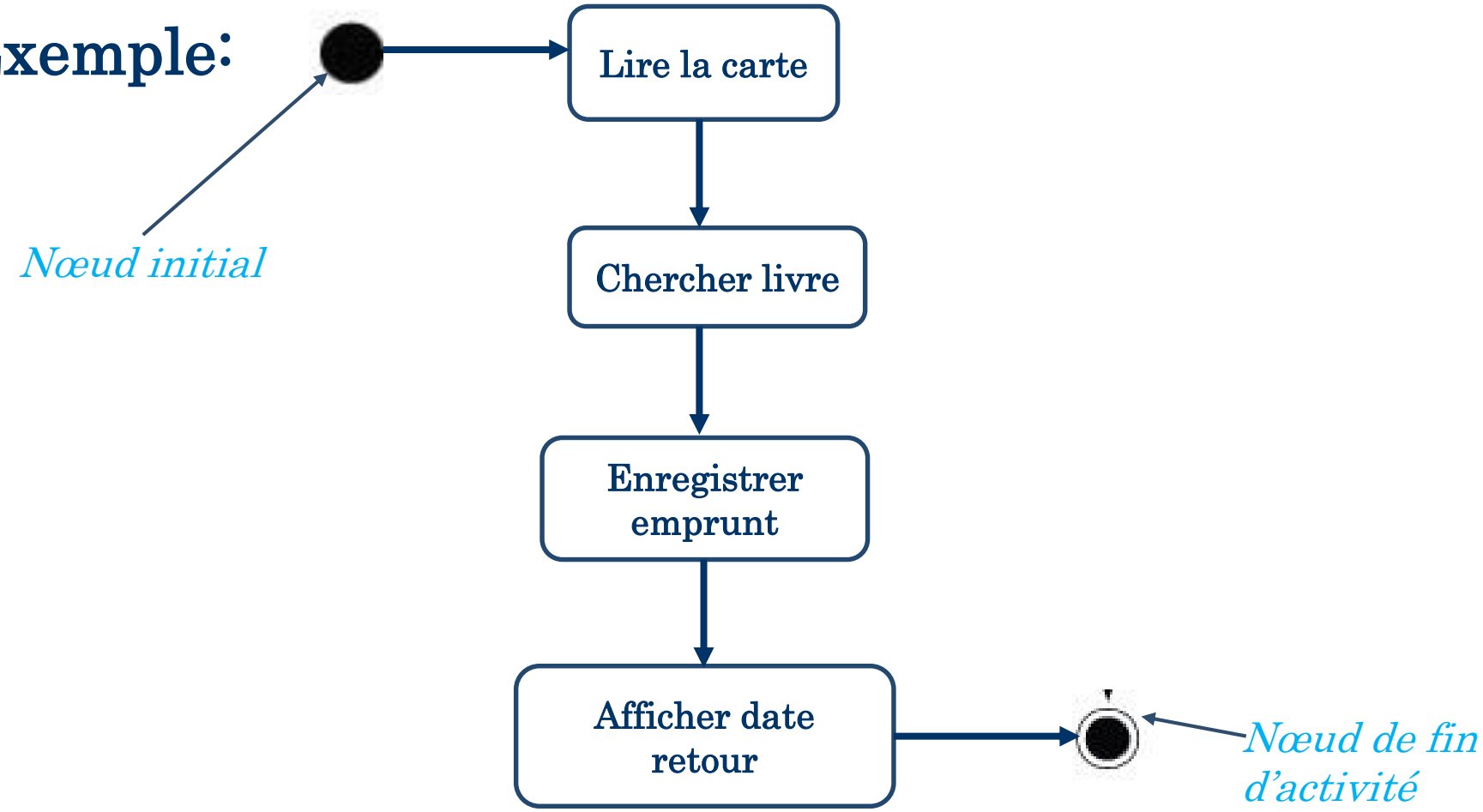
Le passage d'une **activité** vers une **autre** est matérialisé par une **transition**. Elles sont **déclenchées** dès que l'**activité source est terminée** et provoquent **automatiquement** et immédiatement le **début de la prochaine activité** à déclencher (l'activité cible). Contrairement aux activités, les **transitions** sont **franchies de manière atomique**, en principe sans durée perceptible.

Représentation graphique:



4. Nœuds d'activité:

Exemple:



4. Nœuds d'activité

1. Définition.
2. Les familles de nœuds d'activité.
3. Nœud d'activité structuré.

4. Les familles de nœuds d'activité

1. Les nœuds exécutable (executable node) et nœuds d'action (action node).
2. Les nœuds de contrôle (control node).
3. Les nœuds d'objet (object node).

IV. UML

Diagramme d'activités

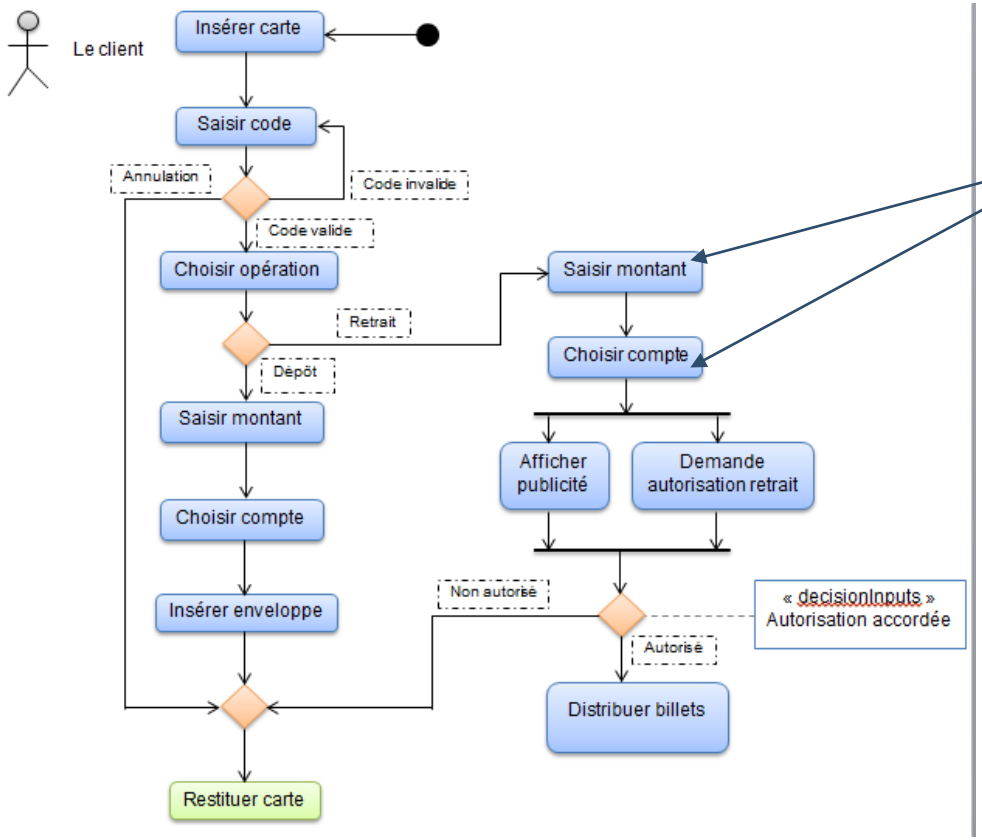
1. Les nœuds exécutable (executable node) et nœuds d'action (action node).

- Un **nœud exécutable** est un nœud d'activité qui donne lieu à une **exécution d'actions**.
- Un **nœud d'action** est un nœud d'activité exécutable qui constitue l'unité fondamentale de fonctionnalité exécutable dans une activité.

IV. UML

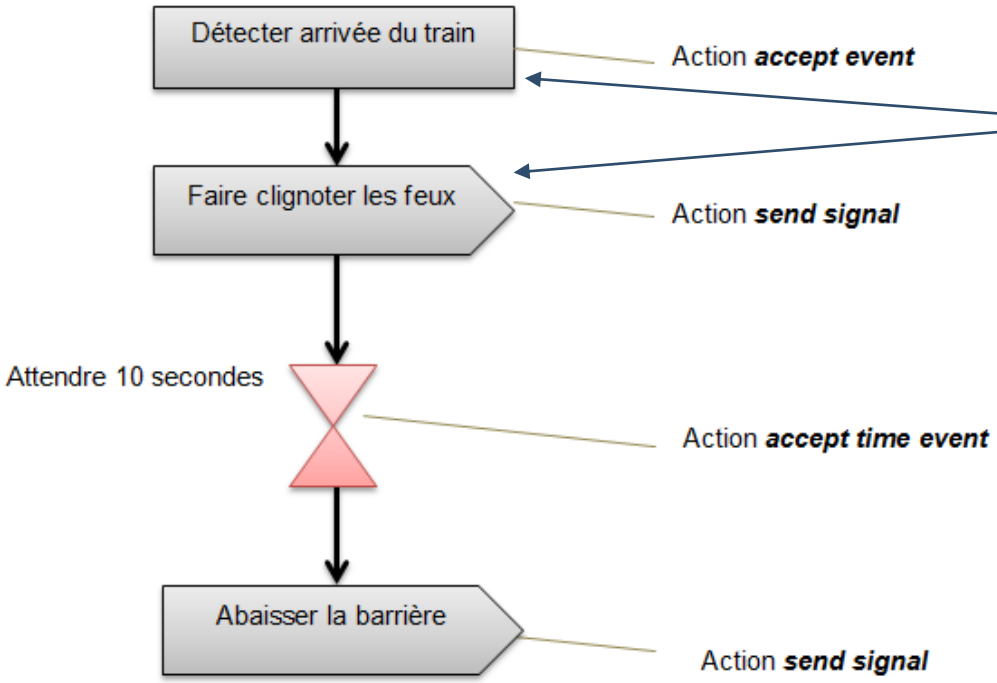
Diagramme d'activités

1. Les nœuds exécutable (executable node) et nœuds d'action (action node).



Nœud d'action:
Rectangle au coins arrondis avec une description textuelle (nom ou suite d'actions réalisées par l'activité ou une syntaxe d'un langage de programmation).

1. Les nœuds exécutable (executable node) et nœuds d'action (action node).



Nœud d'action

Certaines actions ont une représentation graphique spécifique.

Voir le point 2. Action

4. Les familles de nœuds d'activité

1. Les nœuds exécutable (executable node) et nœuds d'action (action node).
2. Les nœuds de contrôle (control node).
3. Les nœuds d'objet (object node).

2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

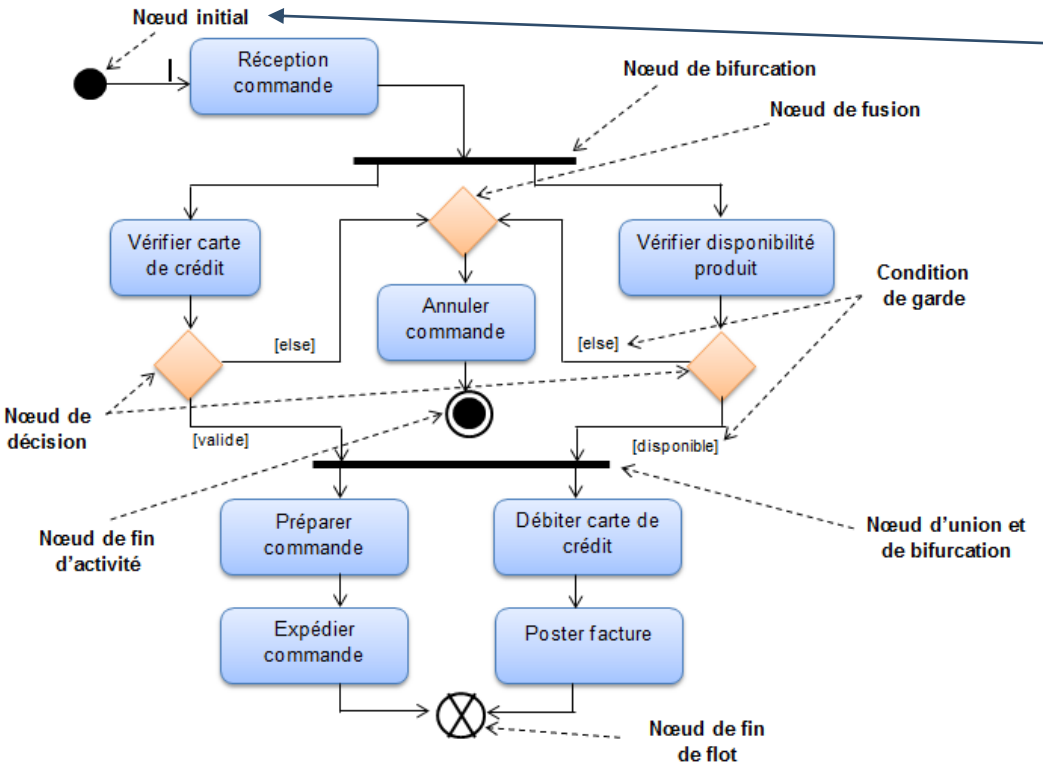
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

2. Les nœuds de contrôle (control node).



Nœud initial

- *C'est un **nœud de contrôle** à partir duquel le **flot débute** lorsque l'activité est invoquée.*
- *Une **activité** peut avoir **plusieurs nœuds** initiaux.*
- *Un **nœud initial** possède un **arc sortant** et pas d'arc entrant.*

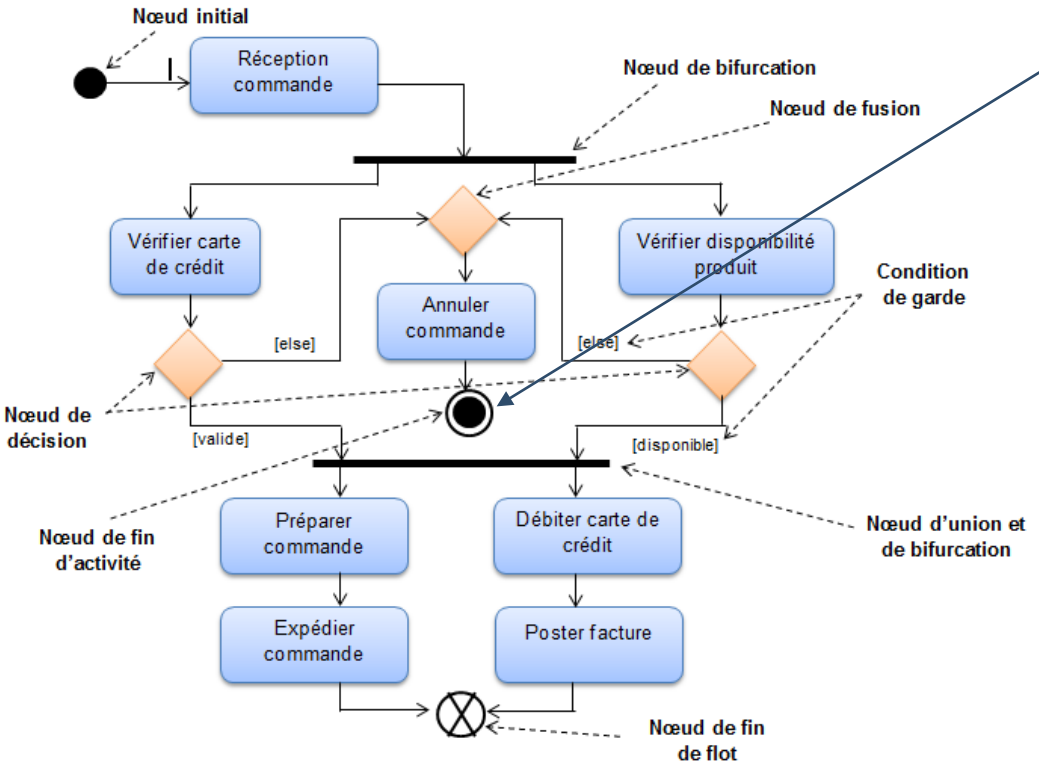
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

2. Les nœuds de contrôle (control node).



Nœud final

- *C'est un **nœud de contrôle** à partir duquel l'exécution de l'activité enveloppante s'achève.*
- *Un **nœud final** possède un ou plusieurs **arcs entrants** et pas d'arc sortant.*

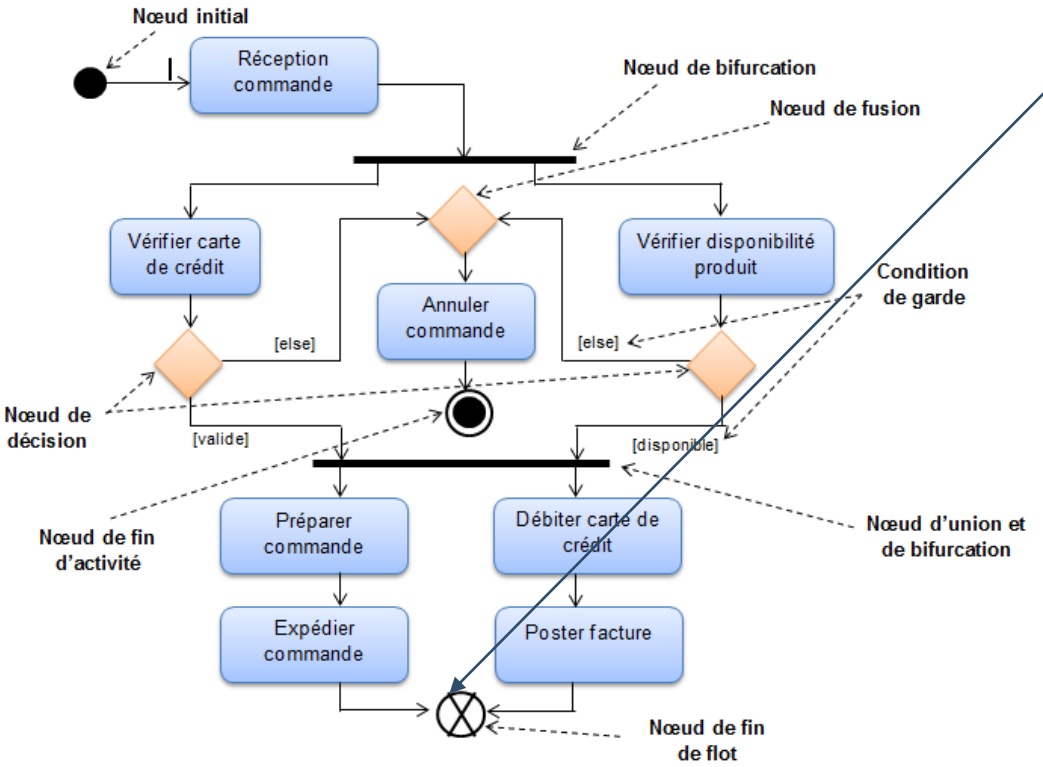
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

2. Les nœuds de contrôle (control node).



Nœud de fin de flot

- *C'est un **nœud de contrôle** à partir duquel le flot en question est terminé.*
- *Cette fin de flot n'a aucune incidence sur les autres flots actifs de l'activité enveloppante.*

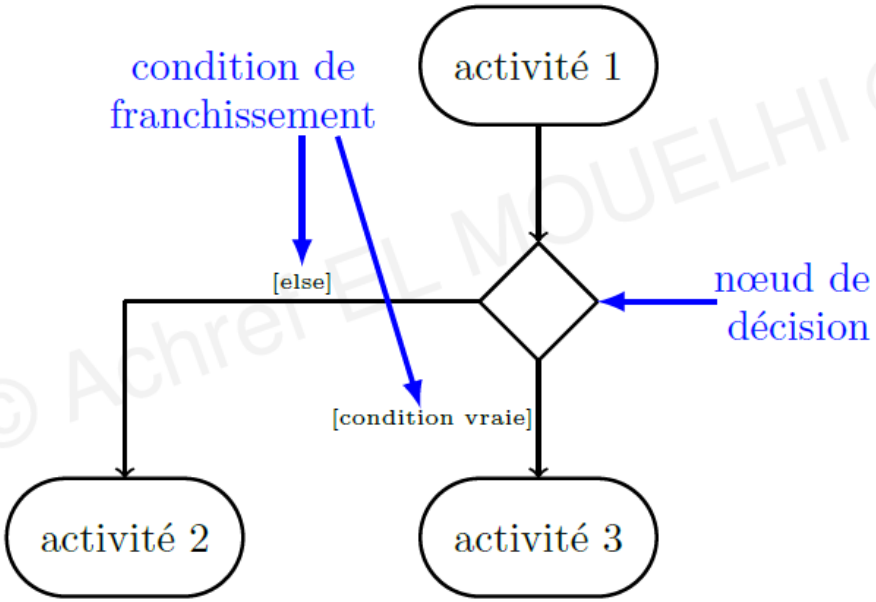
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node)*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

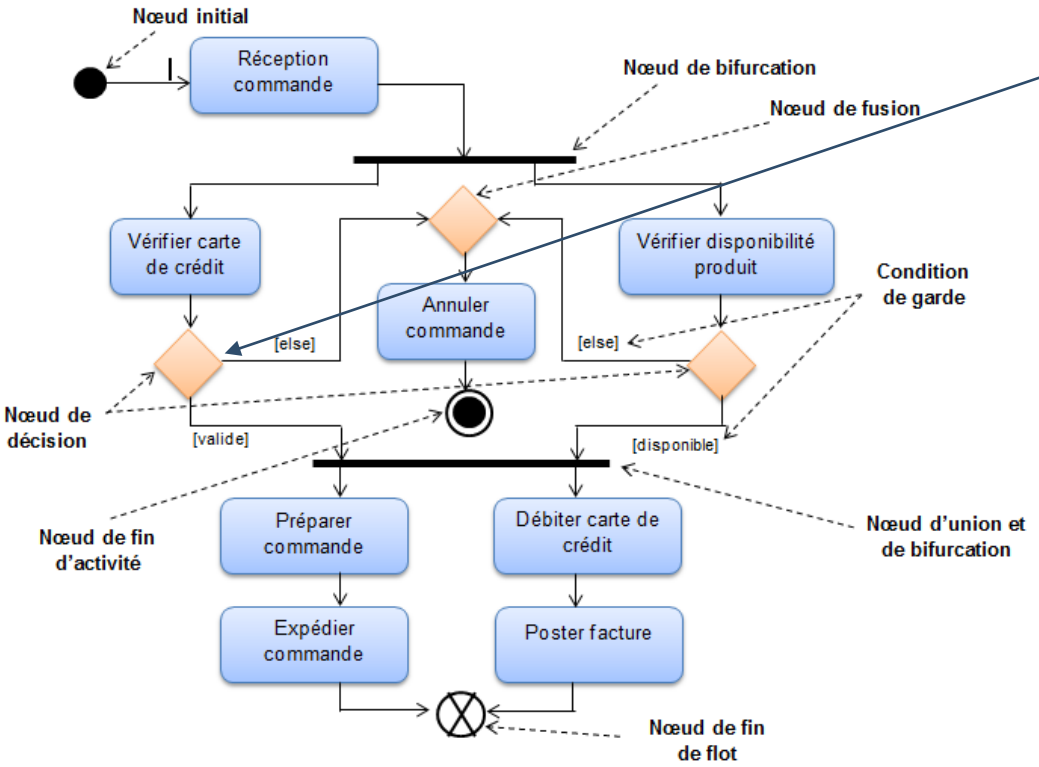
2. Les nœuds de contrôle (control node).



Nœud de décision

- *C'est un **nœud de contrôle** qui permet de **faire un choix** entre plusieurs **flots sortants.***
- *Ce nœud possède **un arc entrant** et **plusieurs arcs sortants.***
- *Les arcs sortants sont accompagnés de conditions de garde pour conditionner le choix.*

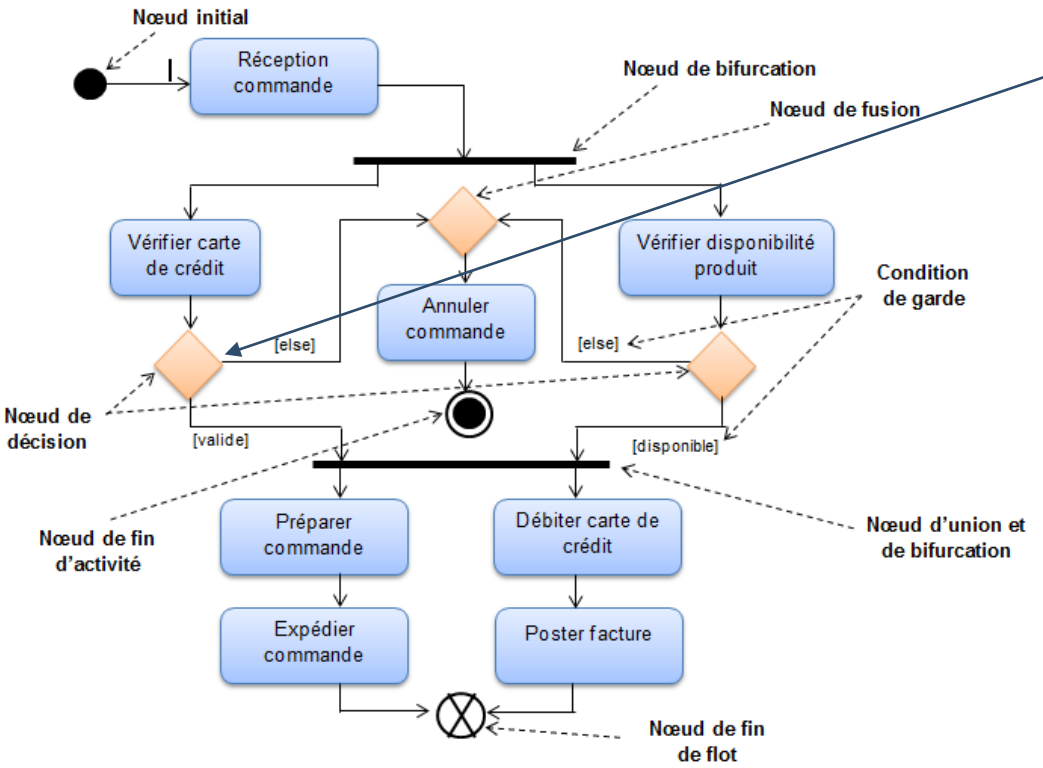
2. Les nœuds de contrôle (control node).



Nœud de décision

- *Si la carte de crédit est valide on peut passer au nœud suivant sinon on annule la commande.*

2. Les nœuds de contrôle (control node).



Remarque importante:

- *Si **aucune condition de garde** n'est **vraie**, c'est que le modèle est mal formé.*
- *L'utilisation d'une garde **[else]** est recommandée après un nœud de décision.*
- *La condition de garde **[else]** est validée si et seulement si **toutes** les autres gardes des transitions ayant la même source **sont fausses**.*

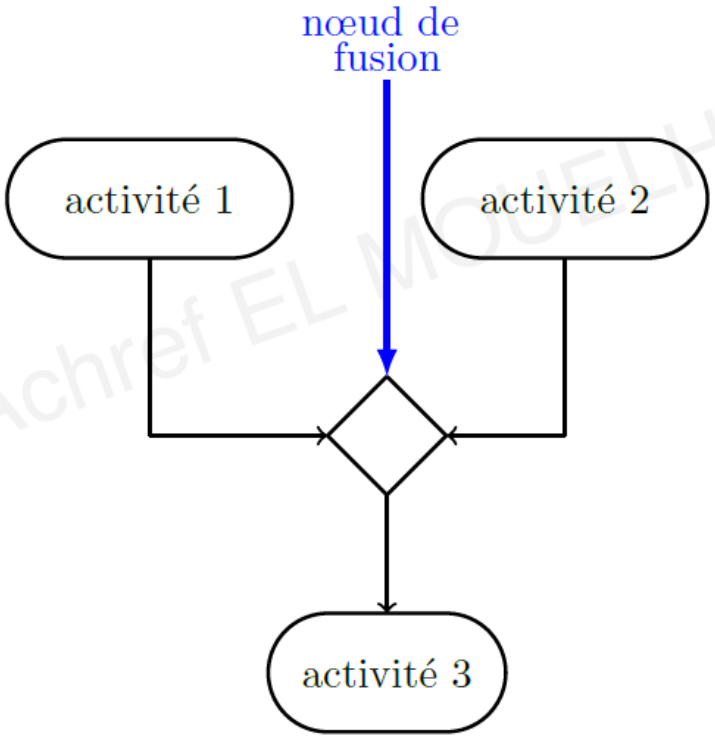
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

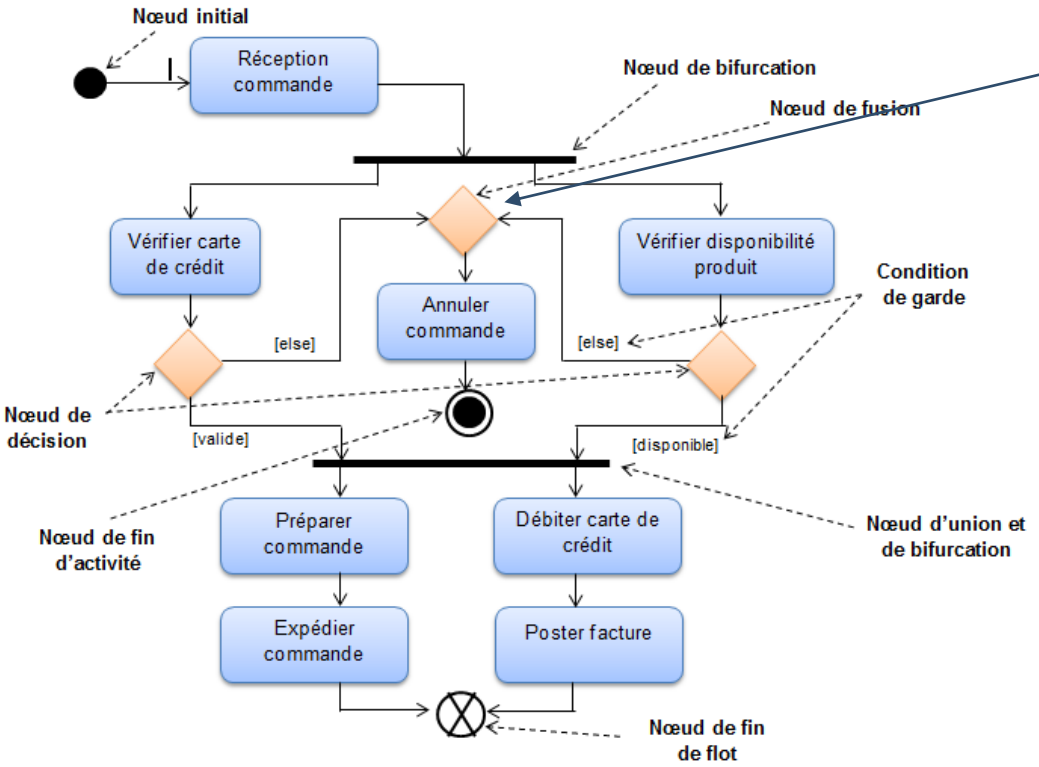
2. Les nœuds de contrôle (control node).



Nœud de fusion

- *C'est un **nœud de contrôle** qui permet de **rassembler plusieurs** flots **alternatifs entrants** en un seul **flot sortant**.*
- *Il est utilisé seulement pour accepter un flot parmi plusieurs.*

2. Les nœuds de contrôle (control node).



Nœud de fusion

- *Si la carte de crédit n'est pas valide ou le produit n'est pas disponible alors on peut annuler la commande.*

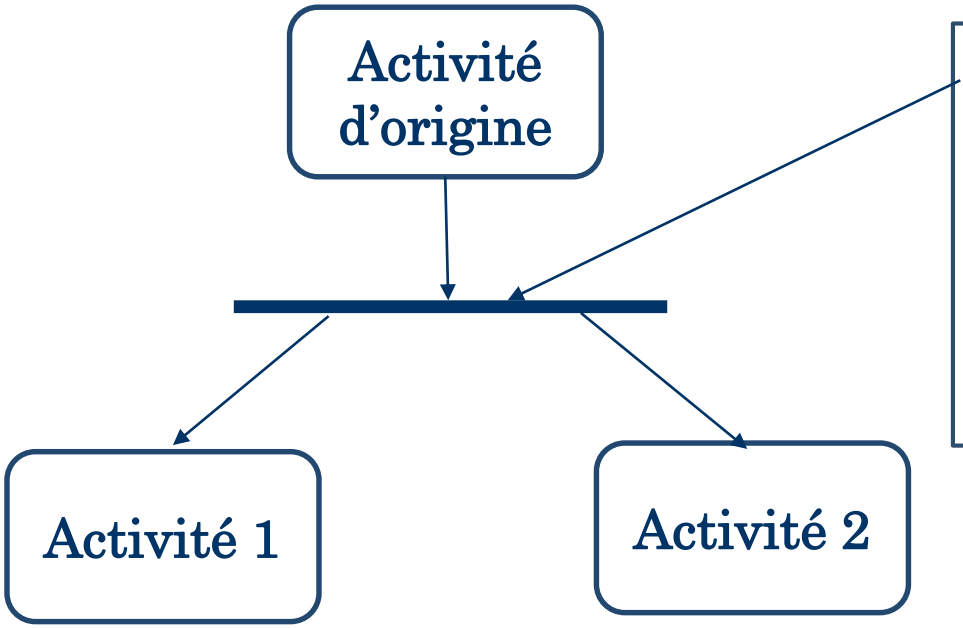
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

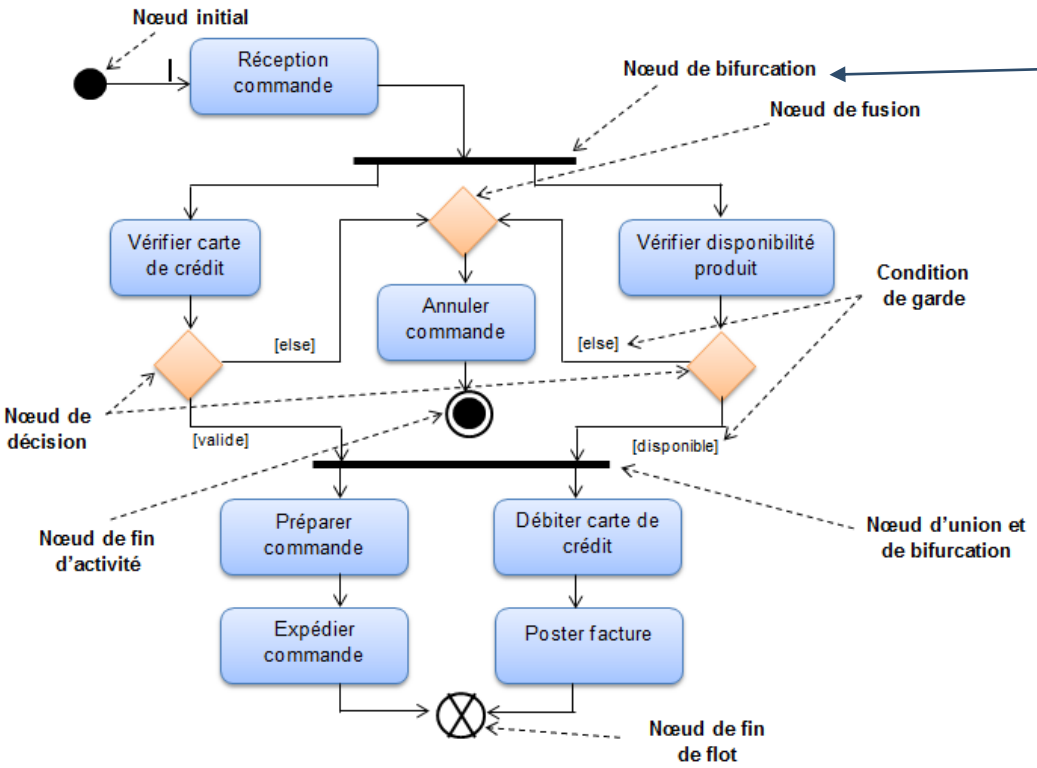
2. Les nœuds de contrôle (control node).



Nœud de bifurcation

- *C'est un* ***nœud de contrôle*** *qui* ***sépare un flot*** *en* ***plusieurs flots concurrents.***
- *Ce nœud possède* ***un arc entrant*** *et* ***plusieurs arcs sortants.***

2. Les nœuds de contrôle (control node).



Nœud de bifurcation

- *Dès réception de la commande, on va en parallèle vérifier la validité de la carte de crédit et vérifier la disponibilité du produit.*

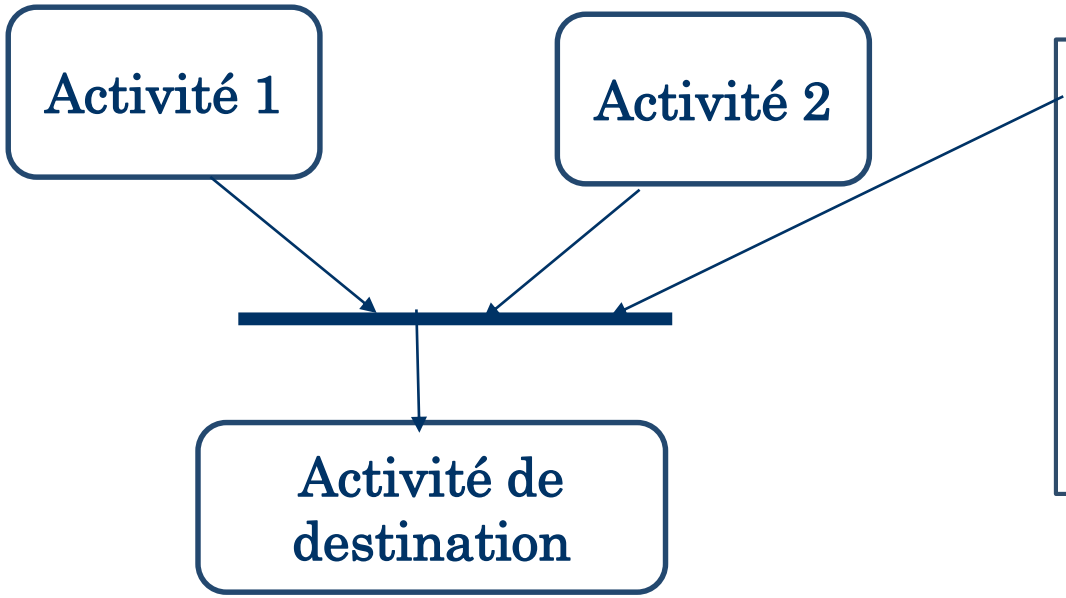
2. Les nœuds de contrôle (control node).

Un nœud de contrôle est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.

Plusieurs types de nœuds de contrôle:

- ***Nœud initial*** *(initial node).*
- ***Nœud de fin d'activité*** *(final node).*
- ***Nœud de fin de flot*** *(flow final)*
- ***Nœud de décision*** *(decision node).*
- ***Nœud de fusion*** *(merge node).*
- ***Nœud de bifurcation*** *(fork node).*
- ***Nœud d'union*** *(join node).*

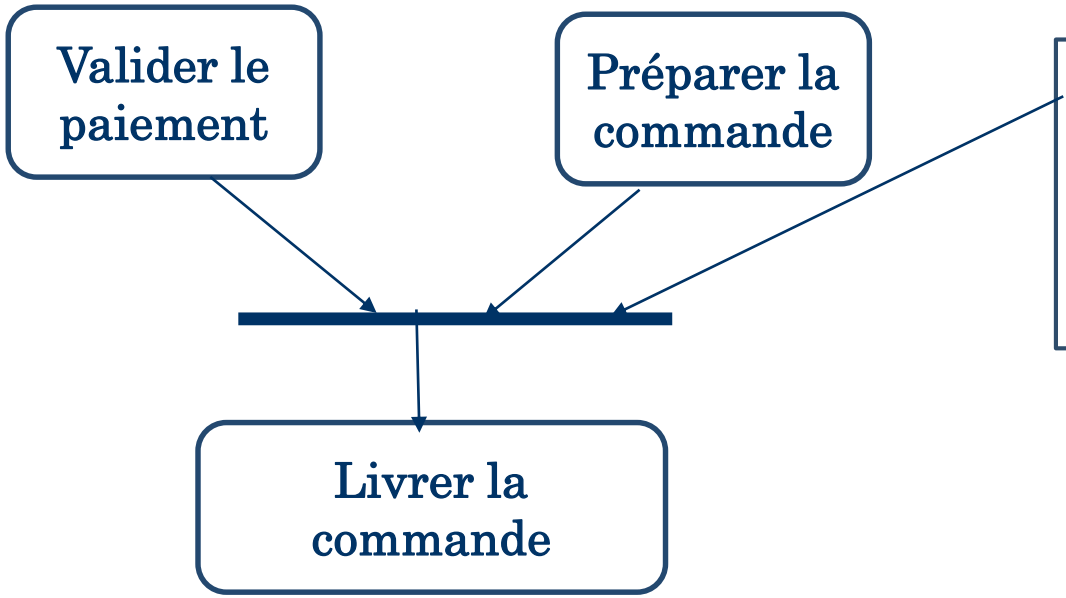
2. Les nœuds de contrôle (control node).



Nœud d'union ou de jointure

- *C'est un* ***nœud de contrôle*** *qui* ***synchronise des flots multiples.***
- *Ce nœud possède* ***plusieurs arcs entrants*** *et* ***un seul arc sortant.***
- *Lorsque tous les arcs entrants sont activés, l'arc sortant l'est également.*

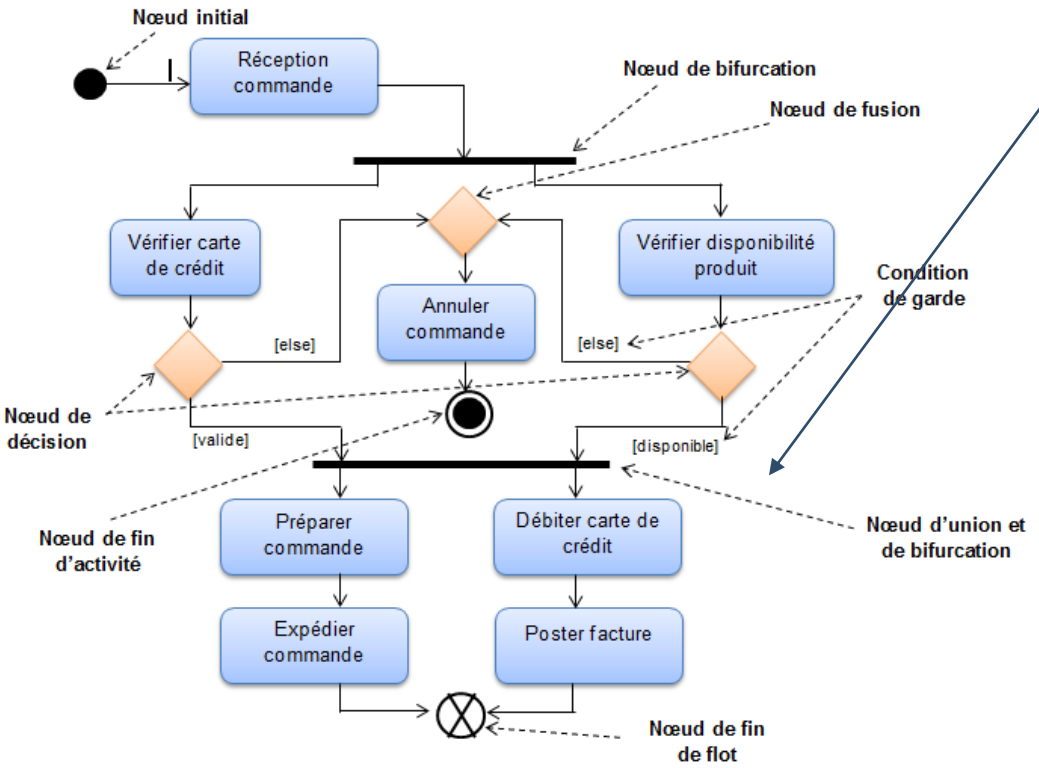
2. Les nœuds de contrôle (control node).



Nœud d'union ou de jointure

Il faut que le paiement soit validé et que la commande préparé pour passer à l'activité « livrer la commande ».

2. Les nœuds de contrôle (control node).



Nœud d'union et de bifurcation

- *Dès réception de la commande, on va en parallèle vérifier la validité de la carte de crédit et vérifier la disponibilité du produit.*

4. Les familles de nœuds d'activité

1. Les nœuds exécutable (executable node) et nœuds d'action (action node).
2. Les nœuds de contrôle (control node).
3. Les nœuds d'objet (object node).

3. Les nœuds d'objet (object node)

1. Introduction.
2. Pins d'entrée/sortie.
3. Flot d'objet.
4. Nœud de stockage de données (data store node).

3. Les nœuds d'objet (object node)

1. Introduction.
2. Pins d'entrée/sortie.
3. Flot d'objet.
4. Nœud de stockage de données (data store node).

1. Introduction:

- Les diagrammes d'activités présentés jusqu'ici montrent bien le **comportement du flot de contrôle**, le flot de données n'apparait pas clairement.

Or, le flot de données est un élément essentiel des traitements.

- Une activité est bien adaptée à la description d'une opération d'un classeur, il faut aussi un moyen de **spécifier les arguments** et **valeurs de retour** de l'opération.
  => C'est le rôle des ***pins***, des ***nœuds*** et des ***flots d'objets*** associés.

3. Les nœuds d'objet (object node)

1. Introduction.
2. Pins d'entrée/sortie.
3. Flot d'objet.
4. Nœud de stockage de données (data store node).

2. Pin d'entrée/sortie:

- On peut utiliser des **nœuds d'objets** appelés **pins d'entrée ou de sortie**, pour spécifier les **valeurs** passées en **argument** à une activité et les **valeurs de retour**.

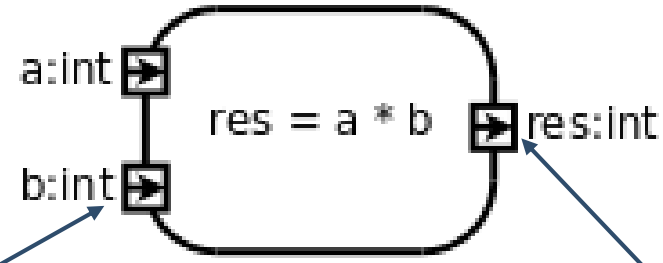
- L'activité ne peut **débuter** que si nous affectons **une valeur** à chacun de ses **pins d'entrée**.

- Quand l'activité se **termine**, **une valeur** doit être affectée à chacun de ses **pins de sortie**.

Remarque importante:
La sémantique associée est celle des langages de programmation usuels: les valeurs sont ***passées par copie***, une modification des valeurs d'entrée au cours du traitement de l'action n'est visible qu'à l'intérieur de l'activité.

2. Pin d'entrée/sortie:

Représentation des pins d'entrée et de sortie sur une activité:



Graphiquement, un pin est représenté par un petit carré attaché à la bordure d'une activité. Il est typé et éventuellement nommé.

Il peut contenir des flèches indiquant sa direction (entrée ou sortie) si l'activité ne permet pas de le déterminer de manière univoque.

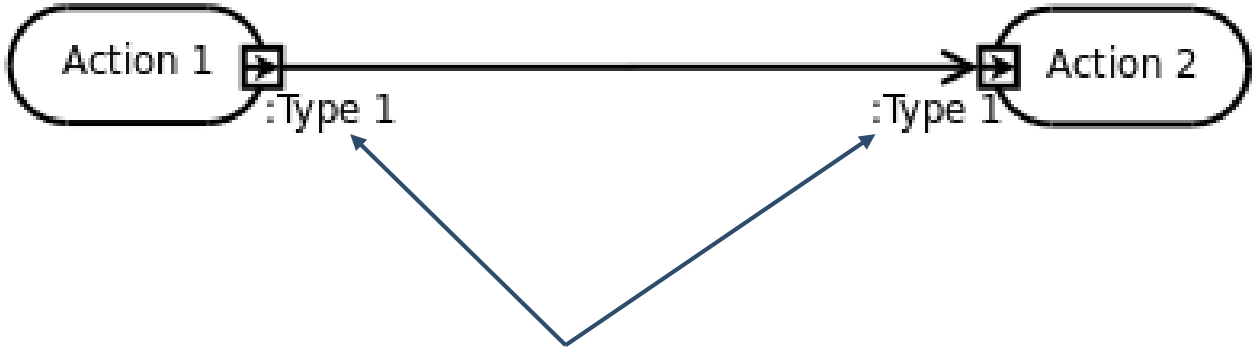
3. Les nœuds d'objet (object node)

1. Introduction.
2. Pins d'entrée/sortie.
3. Flot d'objet.
4. Nœud de stockage de données (data store node).

3. Flot d'objet:

- Un **flot d'objet** permet de **passer des données** d'une activité à une autre.

Première représentation:



Le type du pin récepteur doit être identique ou parent (au sens héritage) du type de pin émetteur.

3. Flot d'objet:

Deuxième représentation:

Cette représentation est **axée sur les données**, car elle fait intervenir un flot d'objet détaché d'une activité spécifique.
Ce **nœud** est représenté par un rectangle dans lequel est mentionné le **type d'objet** (souligné).
On peut aussi ajouter :

- le *nom d'un état* ou liste d'états [NomEtat]
- Des c*ontraintes* {contrainte}

3. Les nœuds d'objet (object node)

1. Introduction.
2. Pins d'entrée/sortie.
3. Flot d'objet.
4. Nœud de stockage de données (data store node).

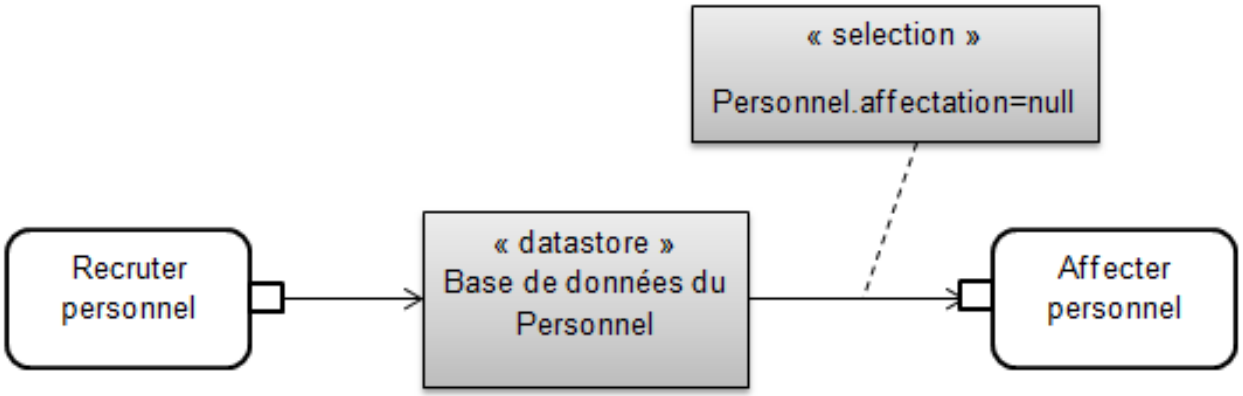
4. Nœud de stockage de données (data store node):

Un **nœud de stockage** des données est un **nœud tampon** central particulier qui assure la **persistance des données**.

- Lorsqu'une **information** est sélectionnée par un **flux sortant**, l'information est **dupliquée** et ne **disparait pas du nœud** de stockage des données.
- Lorsqu'un **flux entrant** véhicule **une donnée déjà st**ockée par le nœud de stockage des données, cette dernière est **écrasée par la nouvelle**.

4. Nœud de stockage de données (data store node):

Exemple:



- Après avoir recruté le personnel, il est stocké dans le nœud de stockage des données « Base de données du Personnel » de façon permanente.
- Ceux qui n'ont pas été affectés sont disponibles pour être affectés par l'activité « Affecter Personnel ».
- L'étiquette « selection personnel.affectation = null » permet de sélectionner ceux qui n'ont pas été affectés.

IV. UML

Diagramme d'activités

1. Introduction.
2. Action.
3. Activité.
4. Nœuds d'activité.
5. Partitions.

5. Partitions ou couloirs d'activités (travées – swinlanes):

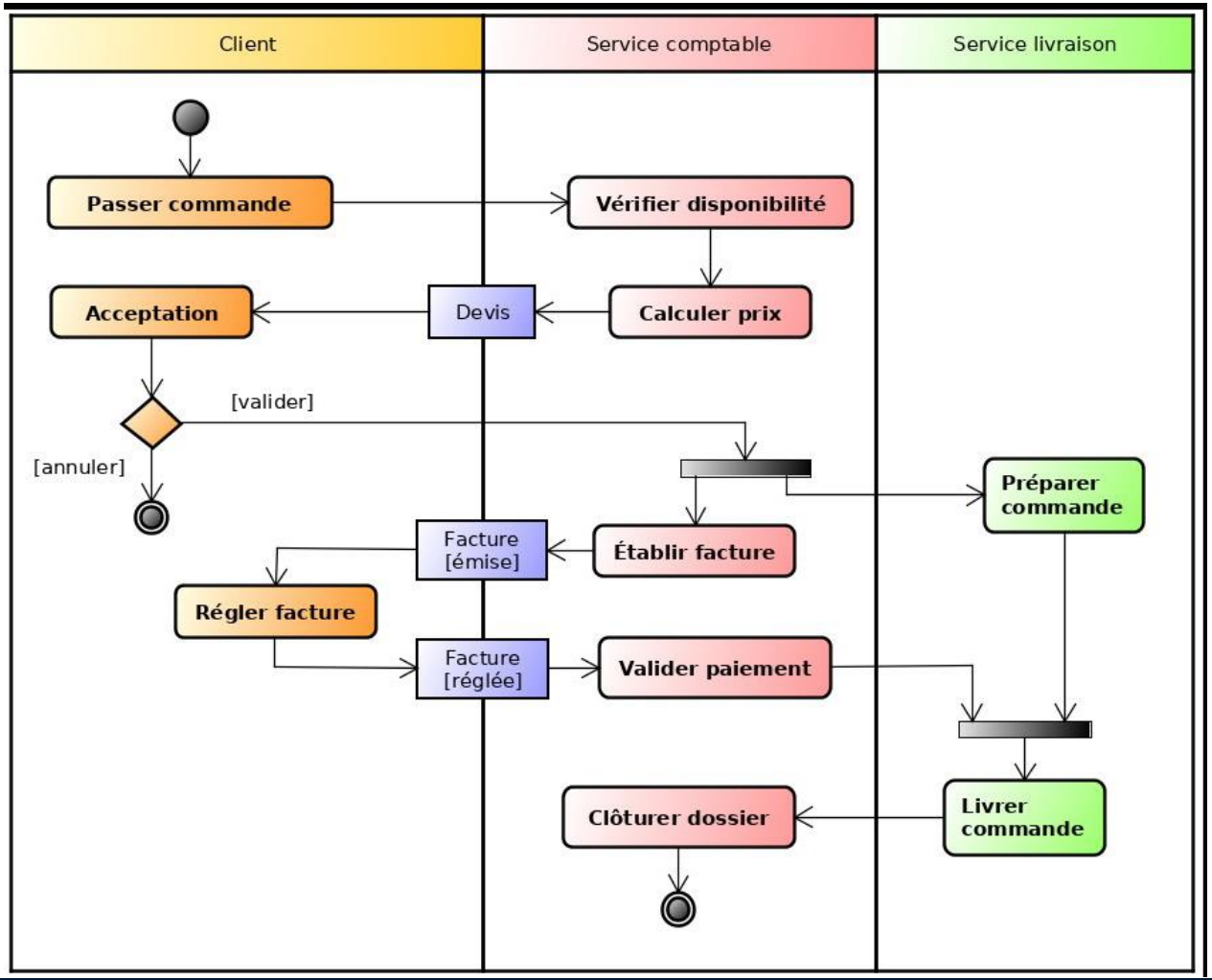
Les diagrammes d'activités indiquent ce qui se passe sans préciser qui fait quoi c'est-à-dire:

- En terme de programmation, ils ne précisent pas quelle classe est responsable.
- En terme de processus métier, ils ne précisent pas quelle partie de l'organisation exécute chaque action.

Il est possible de diviser un diagramme d'activités en partitions ou couloirs d'activités.
Chaque partition montre ainsi quelles actions sont exécutées par une classe ou par une unité organisationnelle.

5. Partitions ou couloirs d'activités (travées – swinlanes):

Exemple:



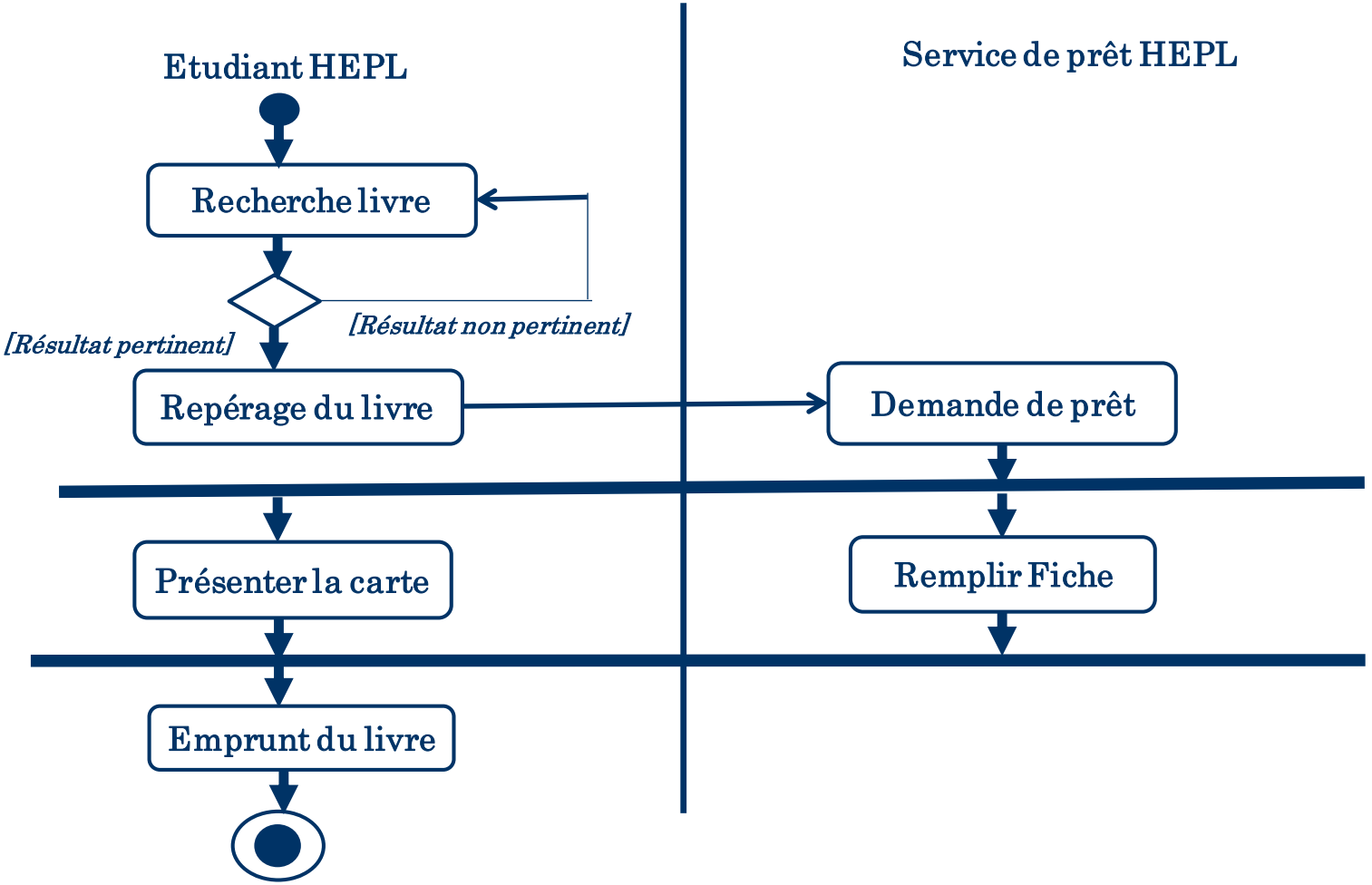
Exercices

Un étudiant de l'HEPL se dirige vers la bibliothèque pour effectuer une recherche.
En fonction des résultats et de la disponibilité des documents l'étudiant procédera à un emprunt.
Dans le service d'emprunt, cette opération nécessite la présence de la carte d'étudiant.

Représenter par un diagramme d'activités le workflow.

IV. UML

Corrections

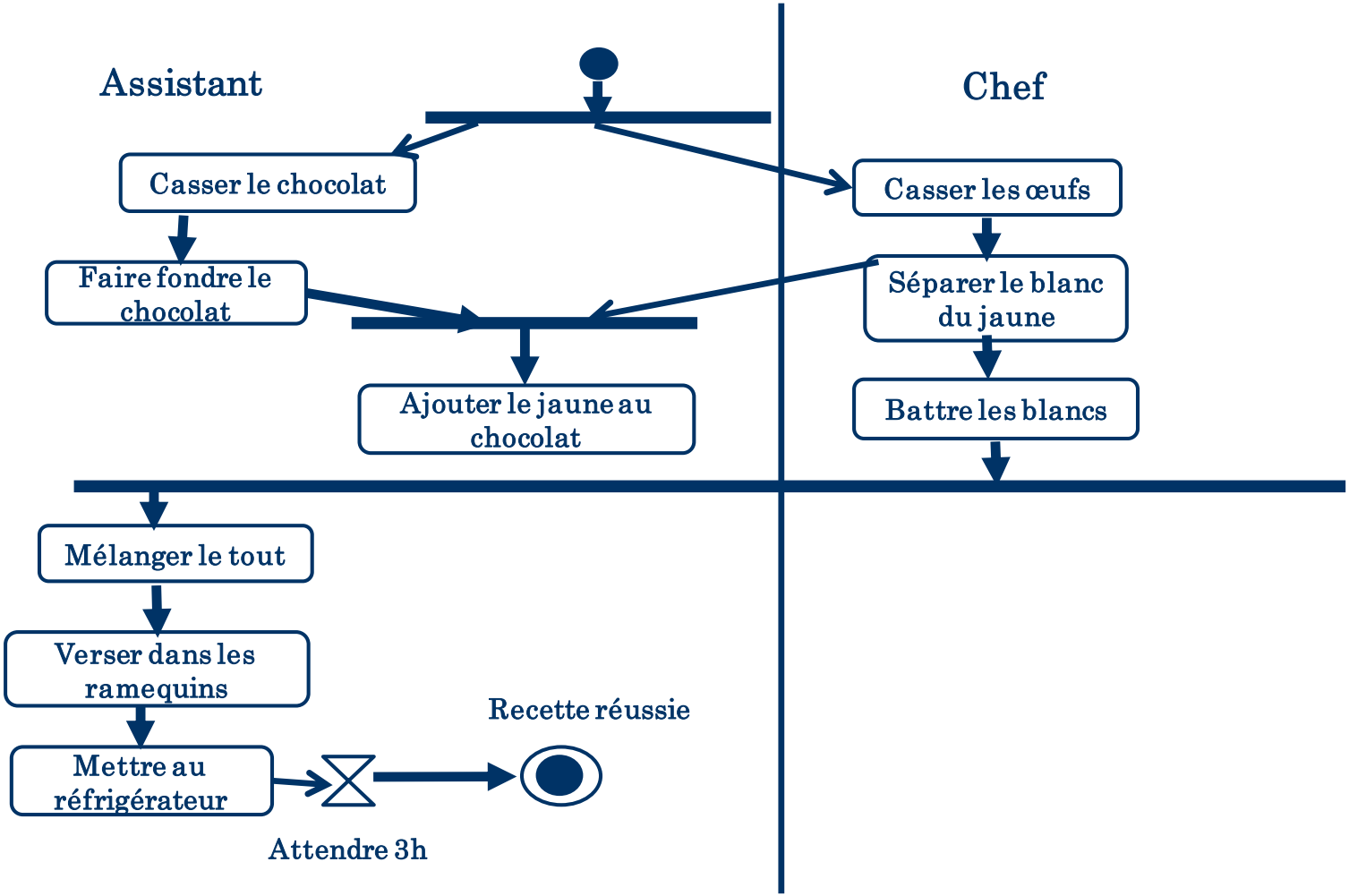


Exercices

Réaliser un diagramme d'activité permettant de préparer une recette « Mousse au chocolat ».
Le chef et son assistant préparent cette recette. Le chef s'occupe des œufs, sépare les jaunes et les blancs et bât les blancs en neige.
L'assistant s'occupe du chocolat, le fait fondre et le mélange avec les jaunes d'œufs.
Il verse ensuite la mousse dans les ramequins, les met au réfrigérateur, attend 3 heures puis la mousse est prête à être servie.

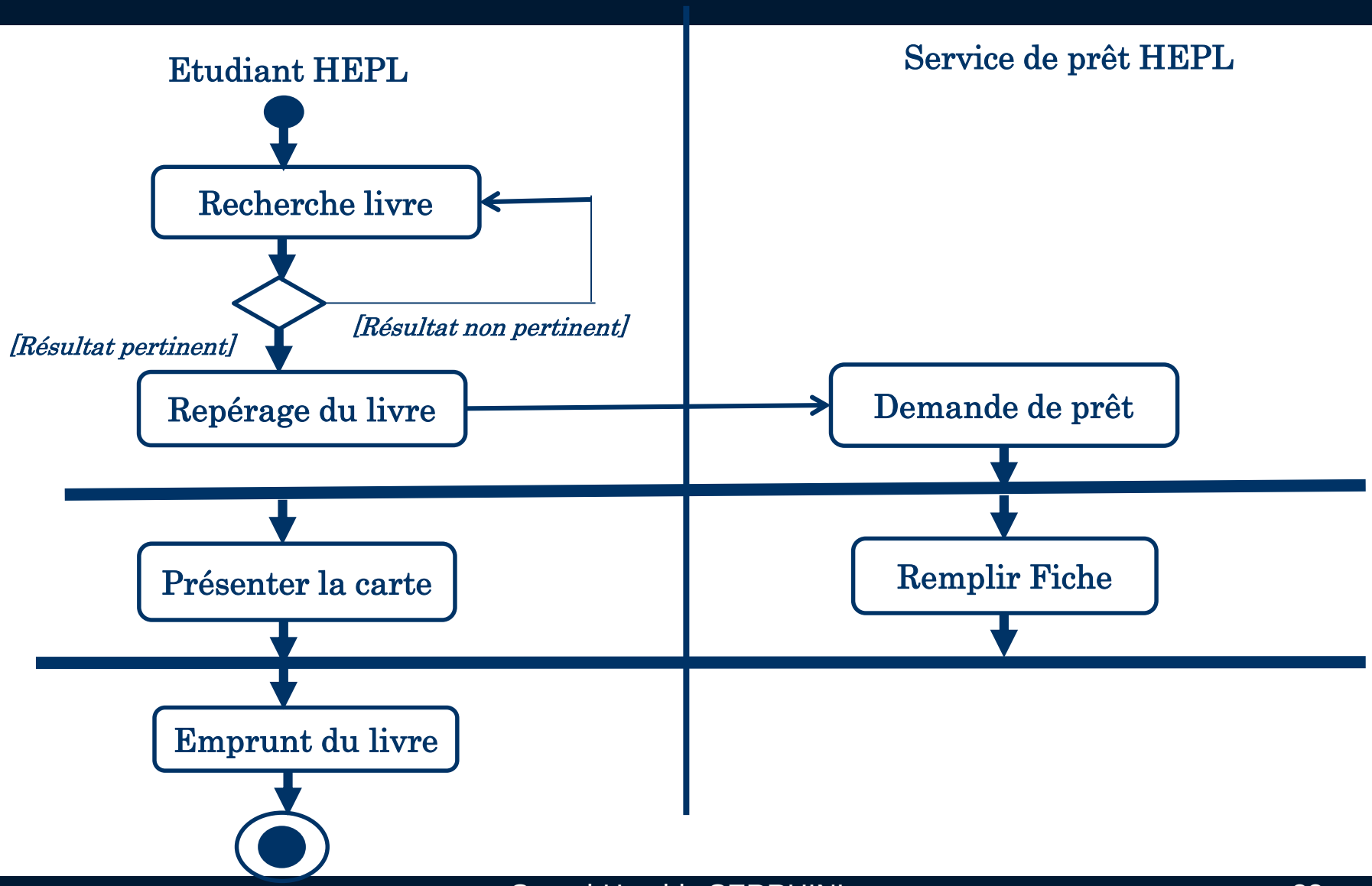
IV. UML

Corrections



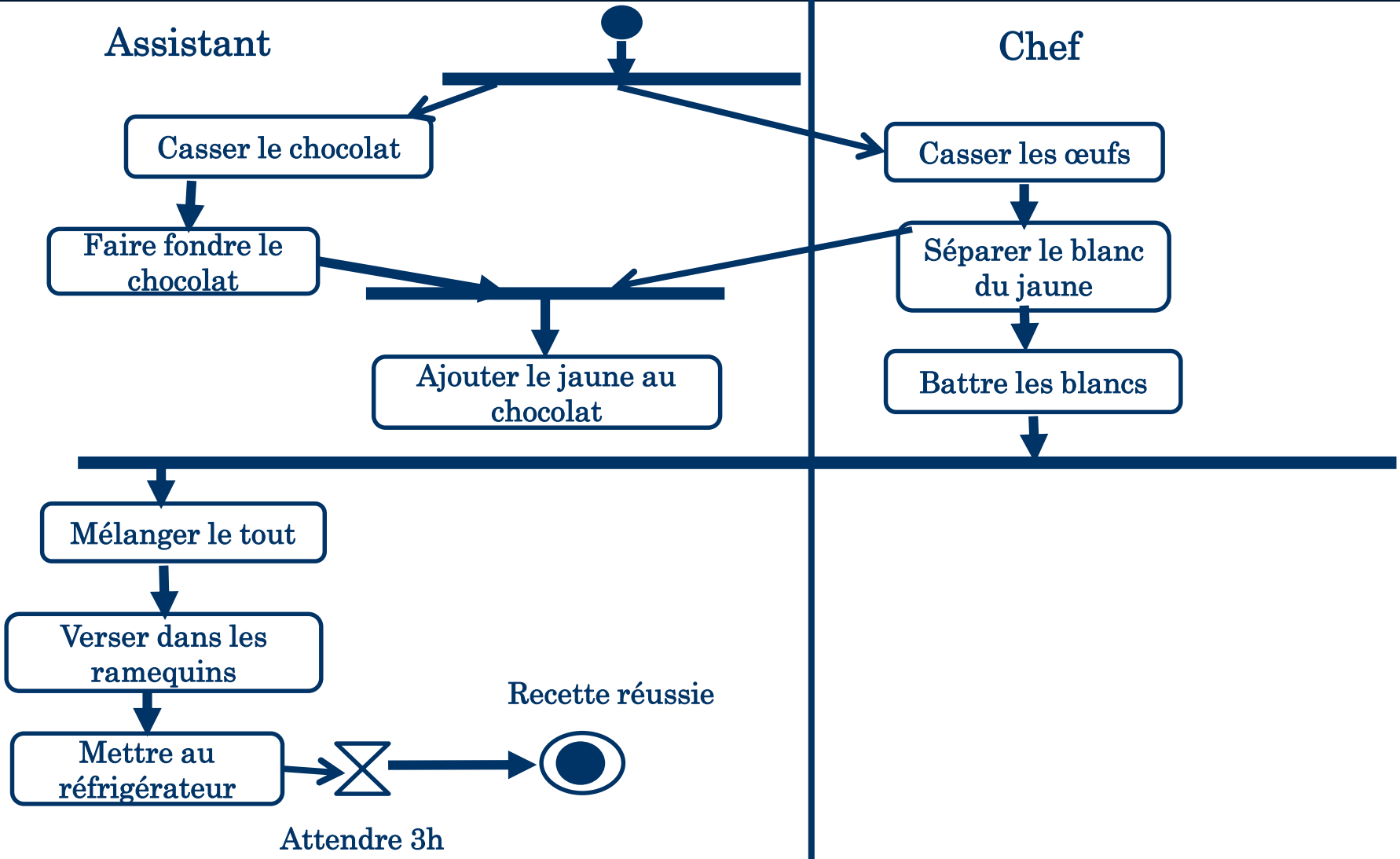
IV. UML

Corrections



IV. UML

Corrections



Bibliographie

1. Benoît CHARROUX, Aomar OSMANI, Yann THIERRY-MIEG. UML2 Pratique de la modélisation. 3ème édition. PEARSON.
2. Laurent AUDIBERT. UML2 de l'apprentissage à la pratique. 2ème édition. ELLIPSES.
3. Christian SOUTOU. Modélisation des bases de données (UML et les modèles entité-association). 3ème édition. EYROLLES.
4. Chantal MORLEY, Jean HUGUES, Bernard LEBLANC. 4ème édition. UML 2 pour l'analyse d'un système d'information. DUNOD.
5. Hugues BERSINI. L'orienté objet. 3ème édition. EYROLLES.
6. Laurent DEBRAUWER, Fien VAN DER HEYDE. UML 2.5. 4ème édition. ENI Editions.
7. Jean-Luc HAINAUT. Bases de données concepts, utilisation et développement. DUNOD.
8. Gilles ROY. Conception de bases de données avec UML. Presses de l'université du Québec.
9. Craig LARMAN. UML2 et les design patterns. 3ème édition. PEARSON Education.
10. Frank BARBIER. UML 2 et MDE. DUNOD.
11. Laurent DEBRAUWER, Naouel KARAM. UML 2 entraînez-vous à la modélisation. Seconde édition. ENI Editions.
12. Corine COSTA. Cours Projets et bureau d'études.