conception et prog objet avancés

DEV.3.4– CPOO

Florent Madelaine

IUT de Sénart Fontainebleau
Département Informatique

Année 2021-2022
Cours 2

Les notions pour l'évaluation

1. Usage : DCU, synopsis.
2. Structure : DC sans les méthodes, DO.
3. Comportement : DC avec méthodes, Diagramme de Séquence.

But du diagramme de séquence

Permet d'ajouter le temps qui était absent du DCU dans le cadre d'exemples.
Permet de réfléchir aux interactions et de réfléchir aux méthodes nécessaires.

Diagrammes de séquences

Il y a en a en fait deux sortes.
Dans les deux cas, on part d'un synopsis qu'on illustre et précise avec un diagramme de séquence.

Diagramme de séquence système

On ignore la structure interne du système qu'on voit en boîte noire comme dans le DCU (on ne dispose pas ou on ne regarde pas le diagramme de classe). On ne modélise que les interactions entre les acteurs et le système.

Diagramme de séquence détaillé

On ouvre la structure interne du système qu'on voit en boîte blanche. On regarde le diagramme de classe. On modélise les interactions entre les acteurs (souvent des objets d'une vue) et les objets du système (modèle ou contrôleur).

Diagrammes de séquences

Clé de lecture avant de passer à des exemples détaillés.

- Chaque acteur / objet a une ligne de vie
- Le temps s'écoule de haut en bas
- Les interactions sont numérotées (d'où le nom de séquence)
- Ces interactions se font par le biais de messages
- En gros il y en a deux : les messages synchrones (en gros un appels de méthode, qui sont bloquants pour l'appelant qui attend la valeur de retour de l'appelé) et les messages (asynchrones).

Digression sur l'aspect asynchrone

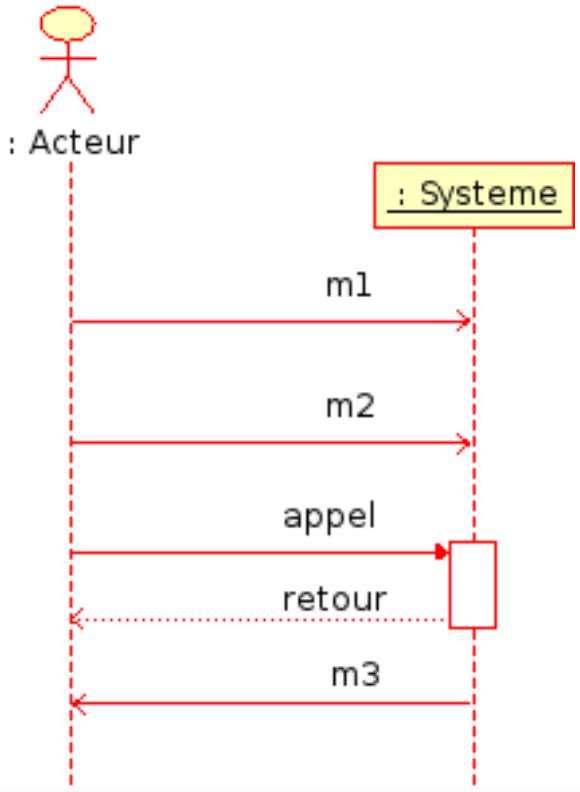
- Dans le cas de messages asynchrones, le comportement du système peut devenir complexe et peu facile à appréhender.
- Exemples : réseau, threads, accès concurrents à une base de données.
- On peut alors soit ignorer le problème (OK si peu fréquent et pas trop grave) soit utiliser des mécanismes de gestion de la concurrence.
- Exemple : mécanismes de verrous et de sémaphores en programmation, transactions et isolation en bases de données (ACID).

La concurrence

message asynchrone vs appel

- Lors d'un message, on ne se préoccupe pas de savoir si son interlocuteur est prêt. Pour un appel, il faut que l'interlocuteur soit disponible (un seul appel à la fois).
- Après l'envoi d'un message, je ne suspends pas mon exécution, pour un appel, je ne fais rien tant que je n'ai pas de réponse.
- Avec des messages, je n'ai pas de garantie sur l'ordre des événements : un message envoyé après peut arriver avant.

La concurrence illustrée



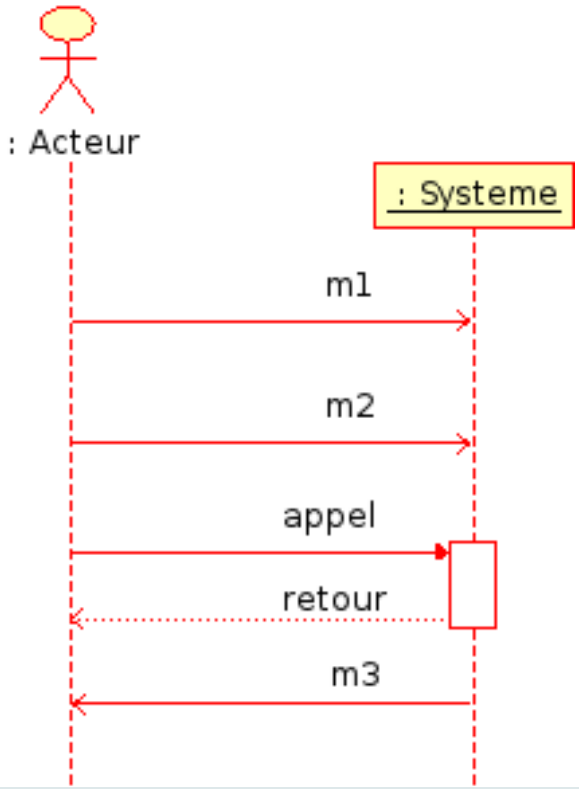
Ordres possibles (‘!’= envoi, ‘?’=réception)

- $m_1!\ m_1?\ m_2!\ m_2?\ appel\ retour\ m_3!\ m_3?$
- $m_1!\ m_2!\ m_1?\ m_2?\ appel\ retour\ m_3!\ m_3?$
- $m_1!\ m_2!\ m_2?\ m_1?\ appel\ retour\ m_3!\ m_3?$
- $m_1!\ m_2!\ appel\ retour\ m_1?\ m_3!\ m_3?\ m_2?$
- ...

On sait seulement que

- L’envoi se fait avant la réception
- Sur une même ligne de vie, l’ordre des messages sortants est respecté.

La concurrence illustrée



Ordres possibles (‘!’= envoi, ‘?’=réception)

- $m_1!\ m_1?\ m_2!\ m_2?\ appel\ retour\ m_3!\ m_3?$
- $m_1!\ m_2!\ m_1?\ m_2?\ appel\ retour\ m_3!\ m_3?$
- $m_1!\ m_2!\ m_2?\ m_1?\ appel\ retour\ m_3!\ m_3?$
- $m_1!\ m_2!\ appel\ retour\ m_1?\ m_3!\ m_3?\ m_2?$
- ...

On sait seulement que

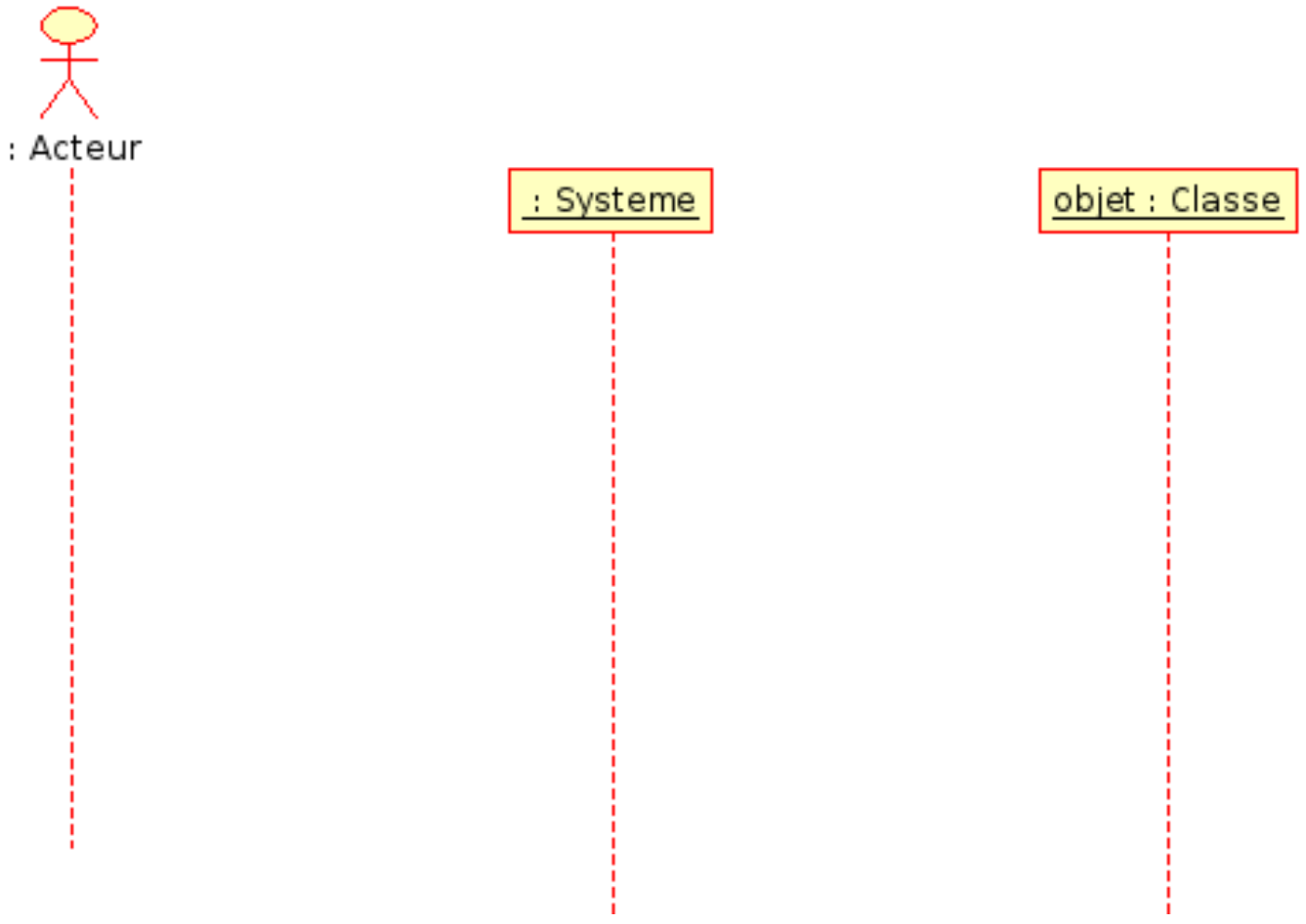
- L'envoi se fait avant la réception
- Sur une même ligne de vie, l'ordre des messages sortants est respecté.

Diagrammes de séquences

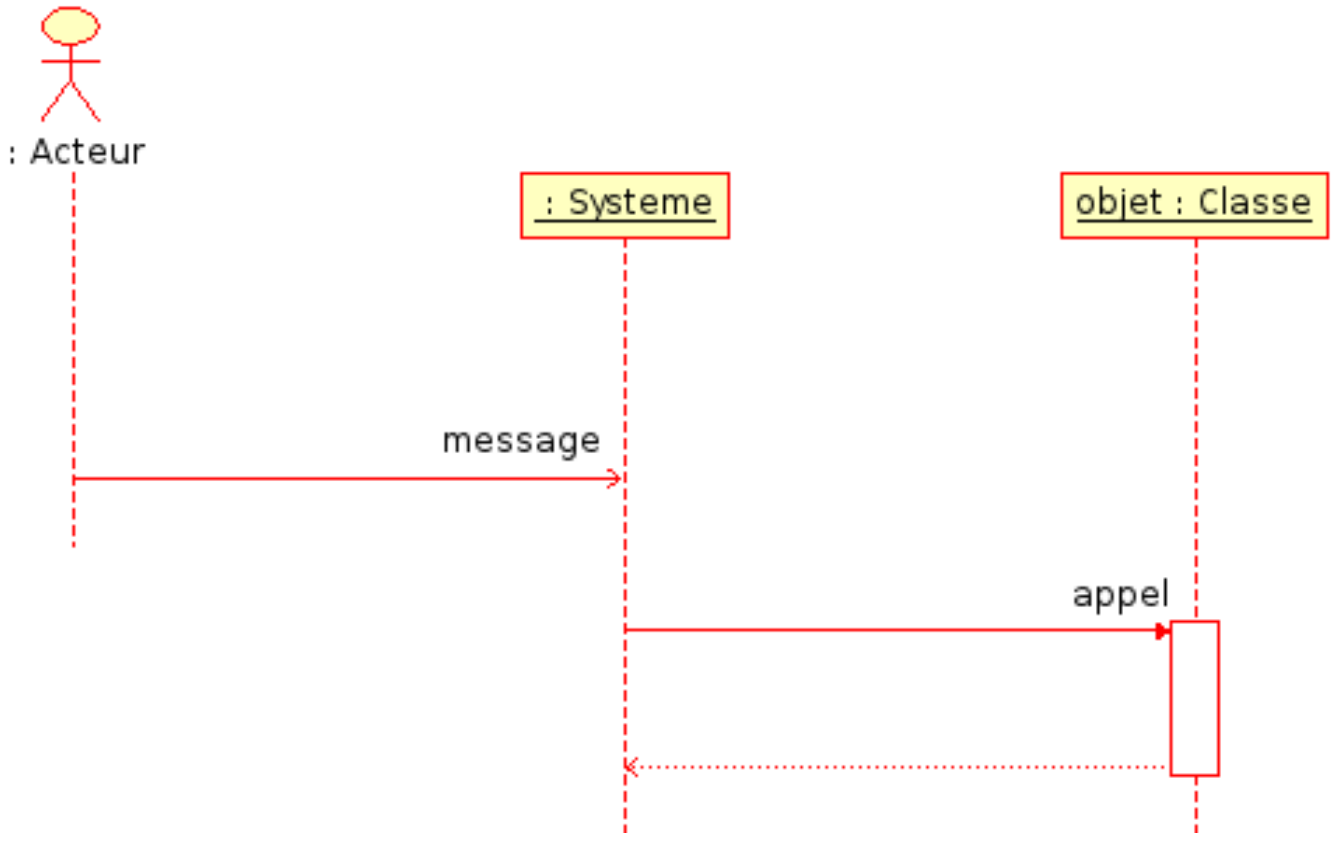
Clé de lecture avant de passer à des exemples détaillés.

- Chaque acteur / objet a une ligne de vie
- Le temps s'écoule de haut en bas
- Les interactions sont numérotées (d'où le nom de séquence)
- Ces interactions se font par le biais de messages
- En gros il y en a deux : les messages synchrones (en gros un appels de méthode, qui sont bloquants pour l'appelant qui attend la valeur de retour de l'appelé) et les messages (asynchrones).

Acteurs et ligne de Vie



Messages

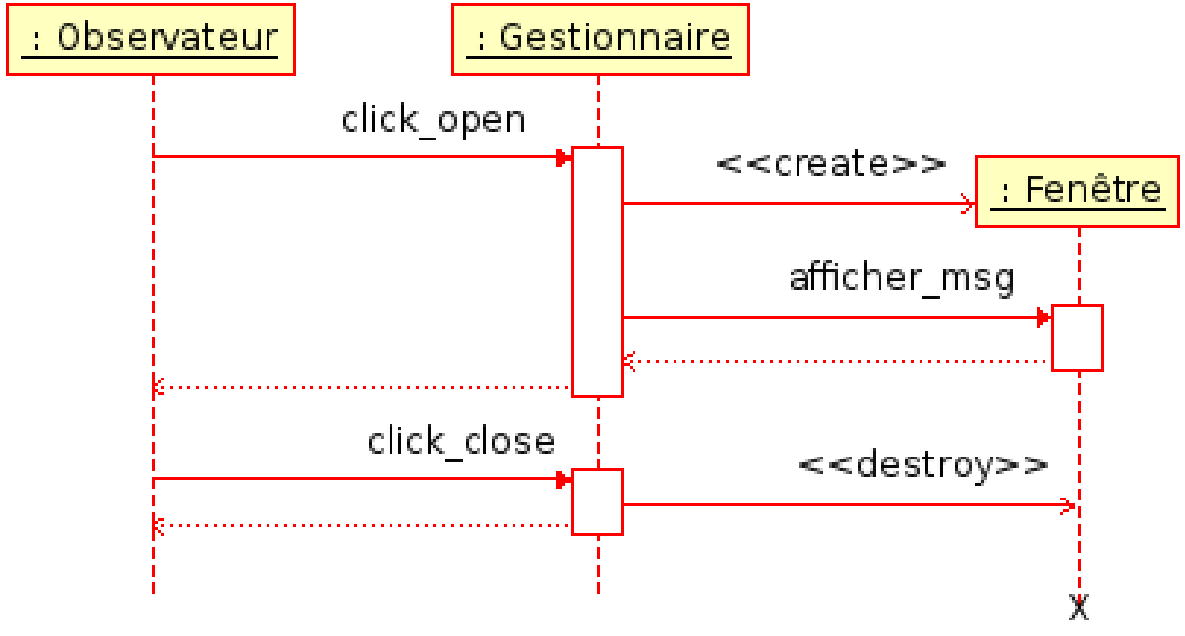


Création, destruction

- Messages qui créent ou détruisent des objets.

↪ On ne crée ni ne détruit un acteur ou un système !

- La création pointe (en général) sur l'objet et non sur sa ligne de vie.
- La destruction termine la ligne de vie de l'objet : plus d'interaction ultérieure.



Diagrammes de séquences

Il y a en a en fait deux sortes.
Dans les deux cas, on part d'un synopsis qu'on illustre et précise avec un diagramme de séquence.

Diagramme de séquence système

On ignore la structure interne du système qu'on voit en boîte noire comme dans le DCU (on ne dispose pas ou on ne regarde pas le diagramme de classe). On ne modélise que les interactions entre les acteurs et le système.

Diagramme de séquence détaillé

On ouvre la structure interne du système qu'on voit en boîte blanche. On regarde le diagramme de classe. On modélise les interactions entre les acteurs (souvent des objets d'une vue) et les objets du système (modèle ou contrôleur).

Diagrammes de séquence

Diagramme de séquence système

- boîte noire
- DCU + synopsis ⇝ DS Système

Diagramme de séquence détaillé

- boîte blanche
- DCU + synopsis + DC ⇝ DS détaillé

Diagrammes de séquence système

On part du synopsis.
Nous sommes en boîte noire. On ne regarde pas le fonctionnement interne du système.

Ligne de vie

Les acteurs du synopsis et le système ont une ligne de vie

Interactions

Dans notre cas se sera des appels de méthode. On ne fera pas de concurrence.
On réfléchit au cas d'usages et aux étapes du synopsis en terme d'appels de méthodes.

Bilan

Pour le DC, on déduit quelles genre de méthodes il faut ajouter et ce qu'elles doivent faire.

Remarque

Il faut itérer et construire des exemples (synopsis) adaptés pour alimenter la réflexion et trouver le diagramme de classe avec les bonnes méthodes.

Diagramme de séquence détaillé

- On procède comme avant mais en boîte blanche.
- On regarde le diagramme de classe.
- On zoome la partie système qu'on éclate en de multiples objets.
- Chaque objet a sa ligne de vie.
- Des objets peuvent être détruits ou créés.
- On en déduit le placement des méthodes.

Cadres d’interaction

Pour l’instant les diagrammes de séquences système ne permettent que de faire des exemples.
Si on veut abstraire et donner une idée du calcul distribué organisé par les différents objets, on peut intégrer des algorithmes dans les DS avec les cadres d’interaction, permettant de préciser les opérantes d’un ensemble de messages.

Les fragments d'interaction pour le flot de contrôle

Notation

Représenté par une boîte englobant une partie des interactions.

Ils permettent d'exprimer des flots de contrôle sur le diagramme :

opt Fragment optionnel, équivalent à un `if...then`.

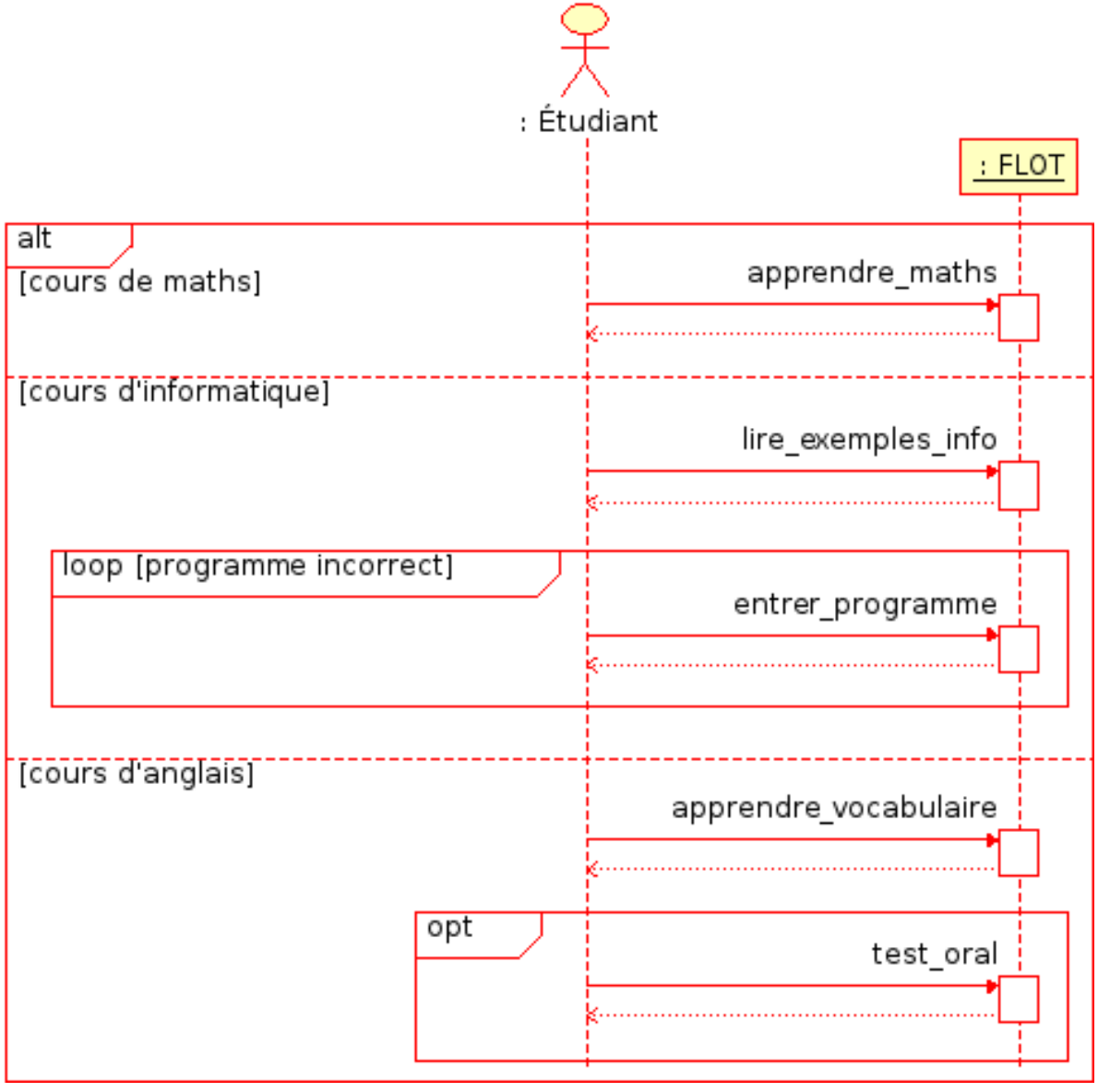
alt Fragment d'alternative, équivalent à un `if ...then ...else`, voire à un `switch ...case ...`.

loop Fragment de boucle, équivalent à un `while` ou un `for`.

Usage

Les conditions (plus ou moins précises) sont placées **en note, entre crochets** dans la partie idoine du fragment.

Les fragments d’interaction pour le flot de contrôle



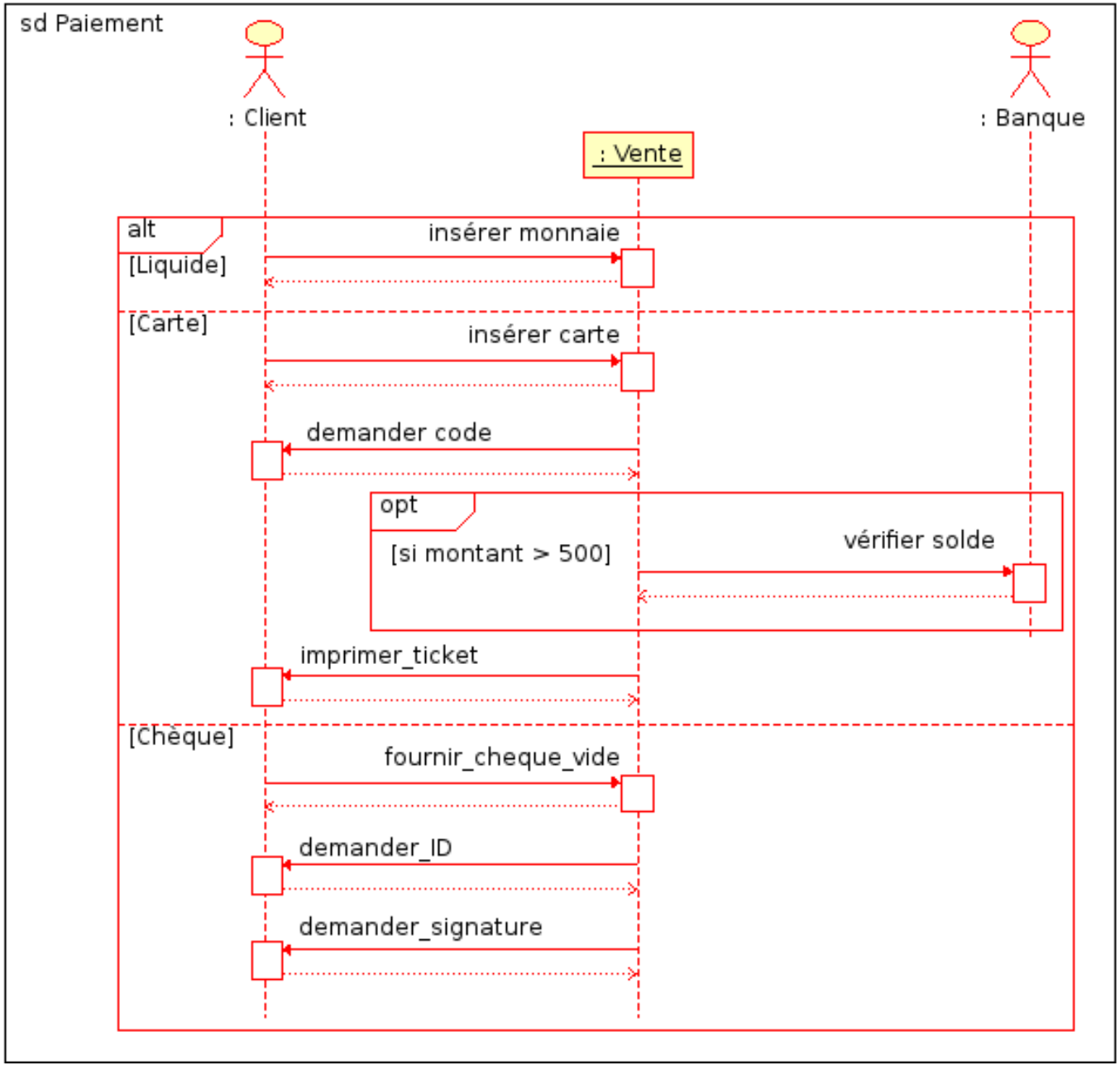
Les fragments d'interaction pour les références

Pour les références : utile lorsqu'un scénario **inclut** un autre, ou lorsqu'on a déjà fait le diagramme d'une fonction annexe appelée.

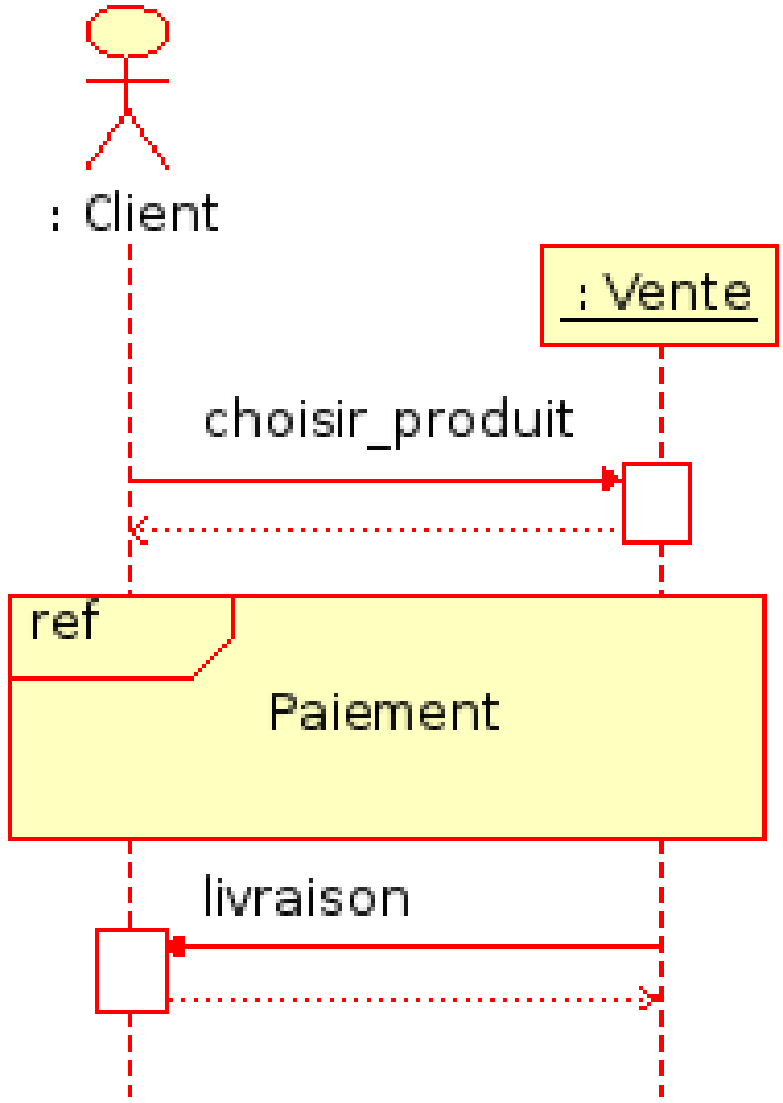
sd Pour **sequence diagrame**, englobe tout le diagramme afin de lui donner un nom.

ref Pour **reference**, agit au sein d'un diagramme comme si le diagramme référé était présent.

Les fragments d'interaction pour les références



Les fragments d'interaction pour les références



Les fragments d'interaction pour la concurrence

Pour la concurrence : pour limiter le nombre d'ordres possibles ou bien spécifier que plusieurs choses peuvent se faire en parallèle.

par L'ordre des événements sortants n'a plus d'importance : ils peuvent se faire en parallèle.

seq Chaque sous fragment doit être terminé avant de passer au suivant : ils se produisent en séquence.

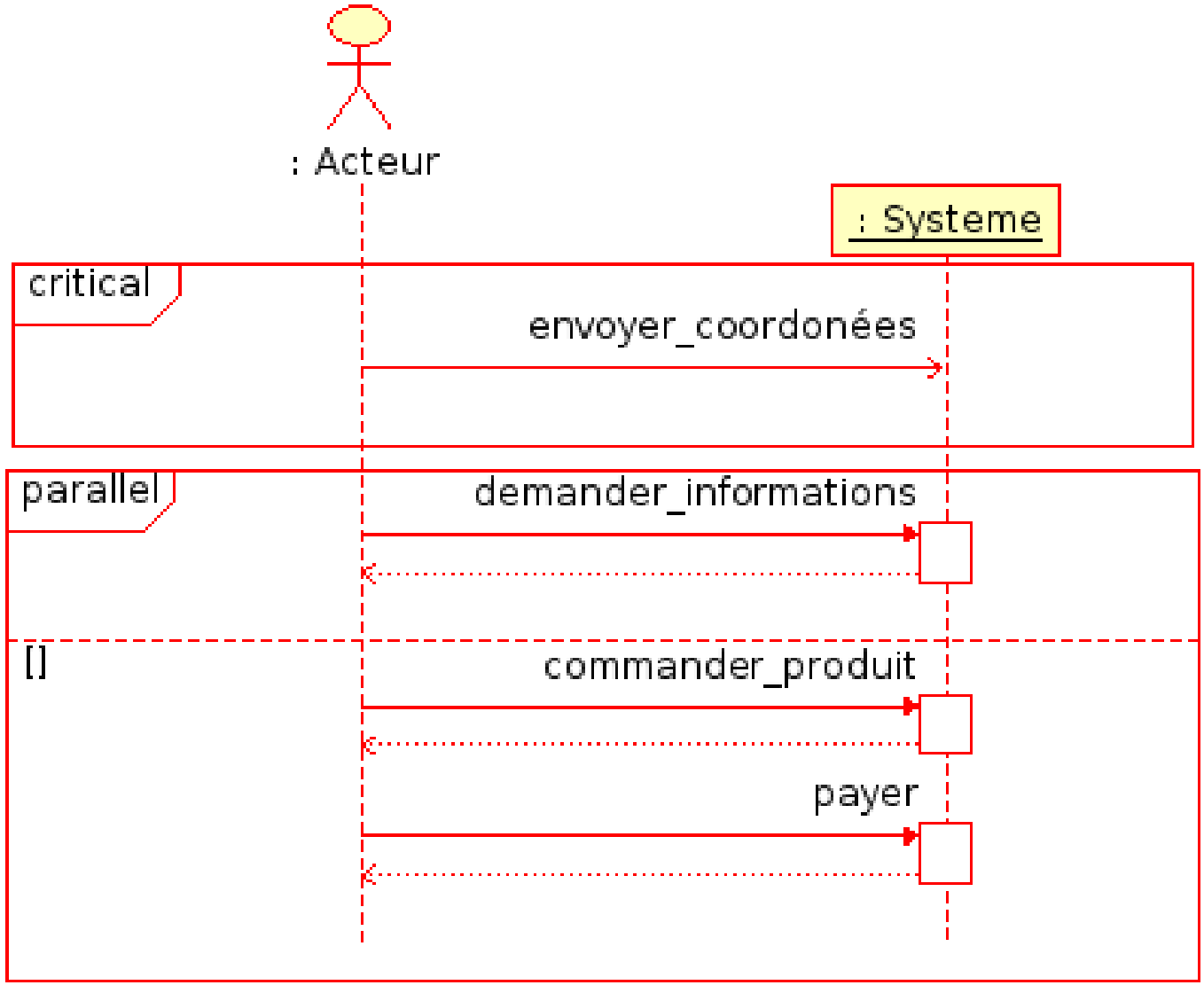
critical Cette partie doit être traitée comme une portion critique : tout d'un bloc. Une séquence est donc équivalente à une suite de portions critiques.

Remark

Si vous voulez vraiment faire de la concurrence, les diagrammes d'activités sont plus adaptés.

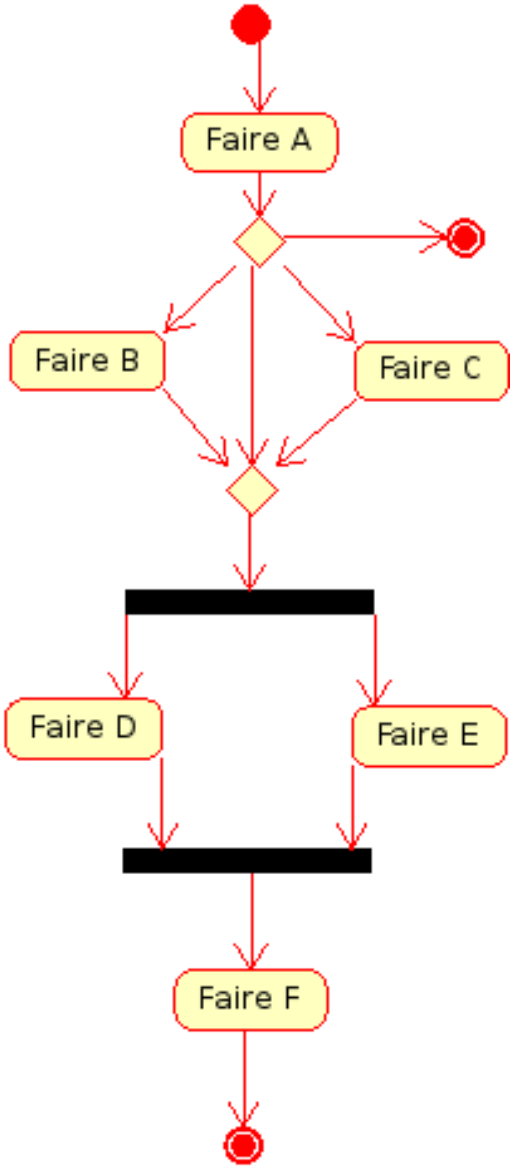
Les fragments d'interaction pour la concurrence

Exemple



Syntaxe de base

Révisions



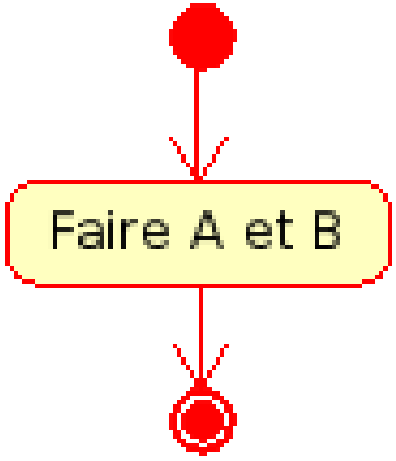
Ingrédients

- Départ du diagramme : unique.
- Activités : verbe à l'infinitif + complément.
- Choix (losange) : activé si une entrée a terminé ; une sortie est choisie : OU.
- Parallélisation/synchronisation (barre) : activé si toutes les entrées ont terminé ; toutes les sorties sont prises : ET.
- Fin du diagramme : potentiellement plusieurs.

Règle d'or

Attention

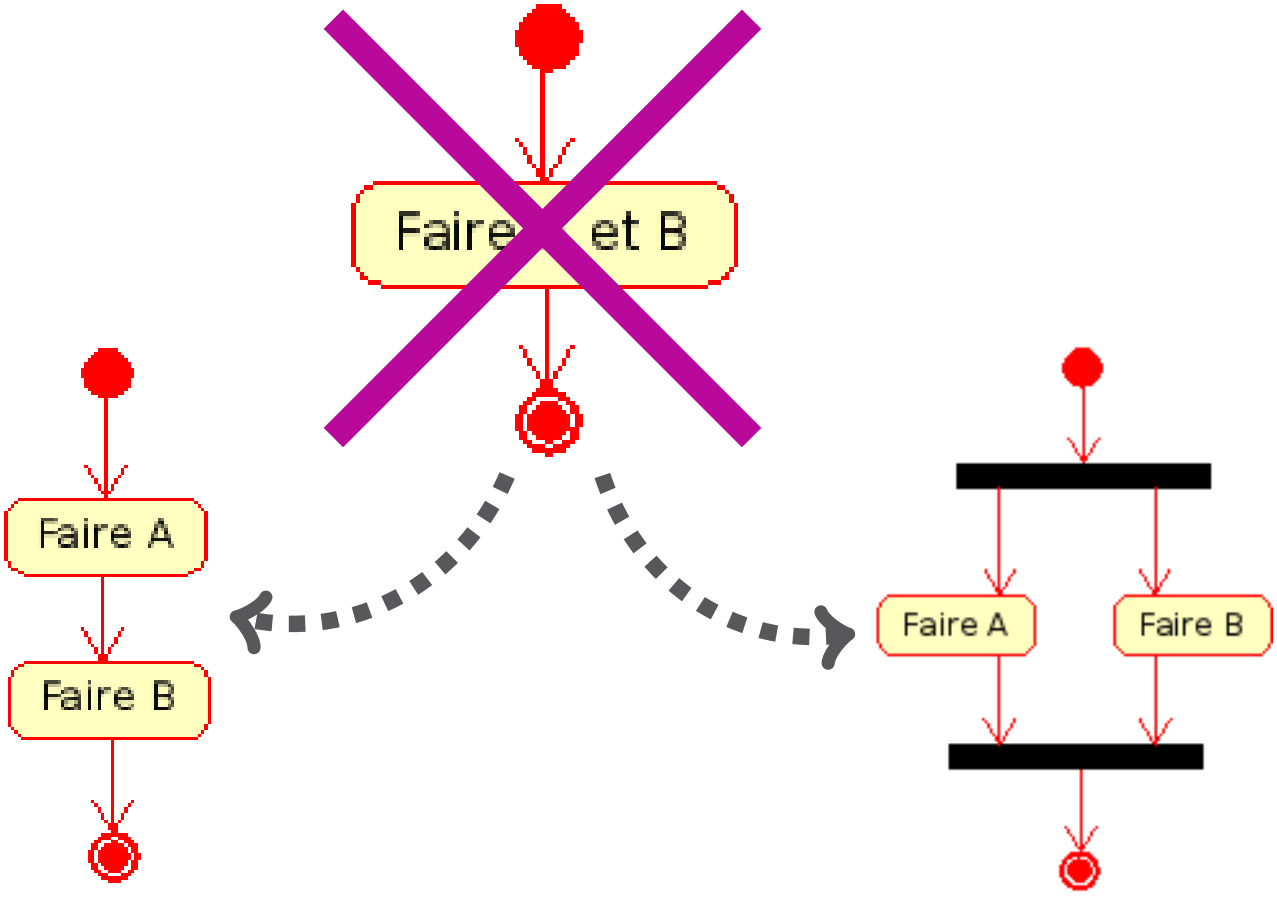
Pas de « et »dans les intitulés d'activités !



Règle d'or

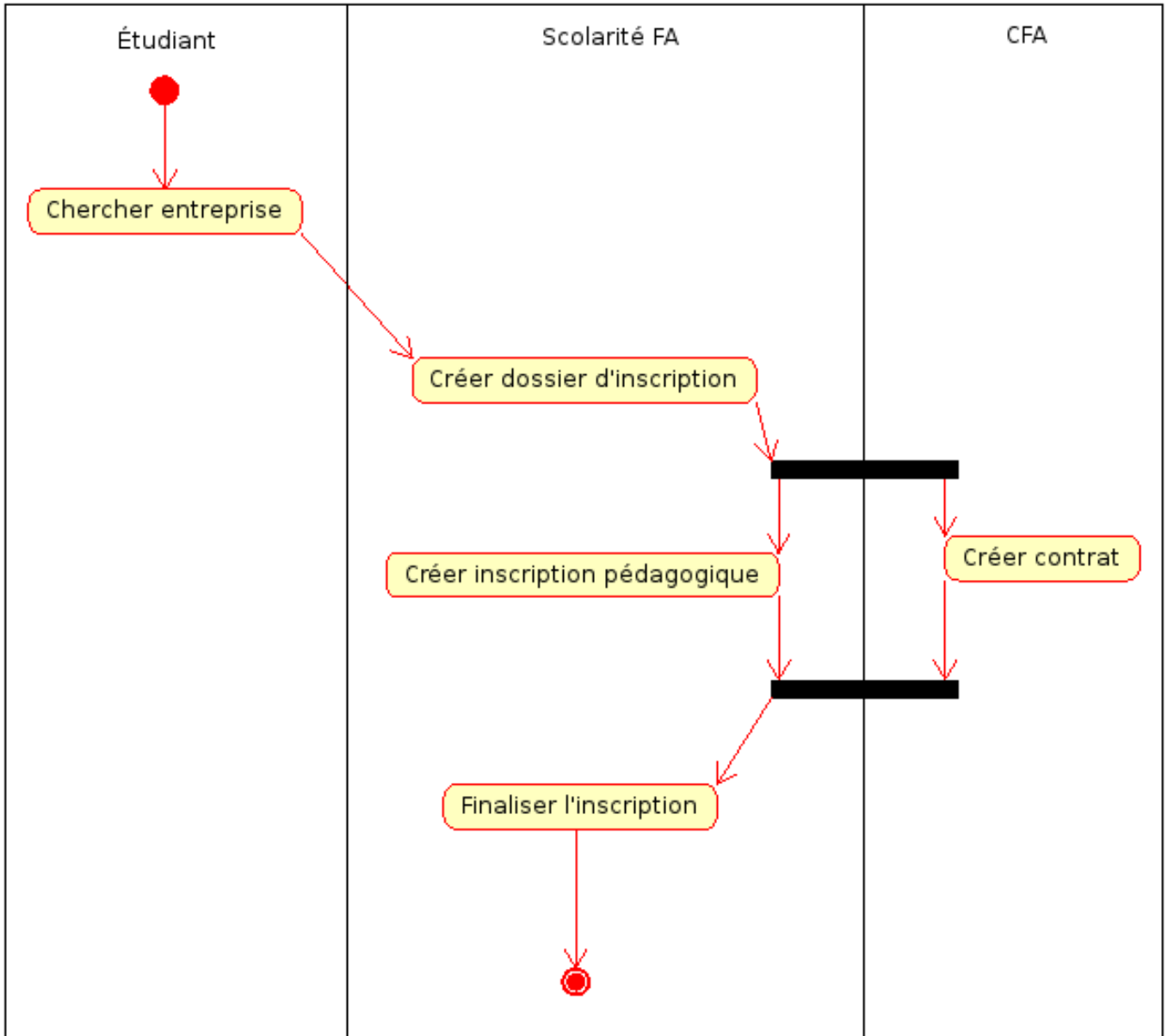
Attention

Pas de « et »dans les intitulés d'activités !



Qui fait quoi ?

Couloir de nage (Swimlanes)

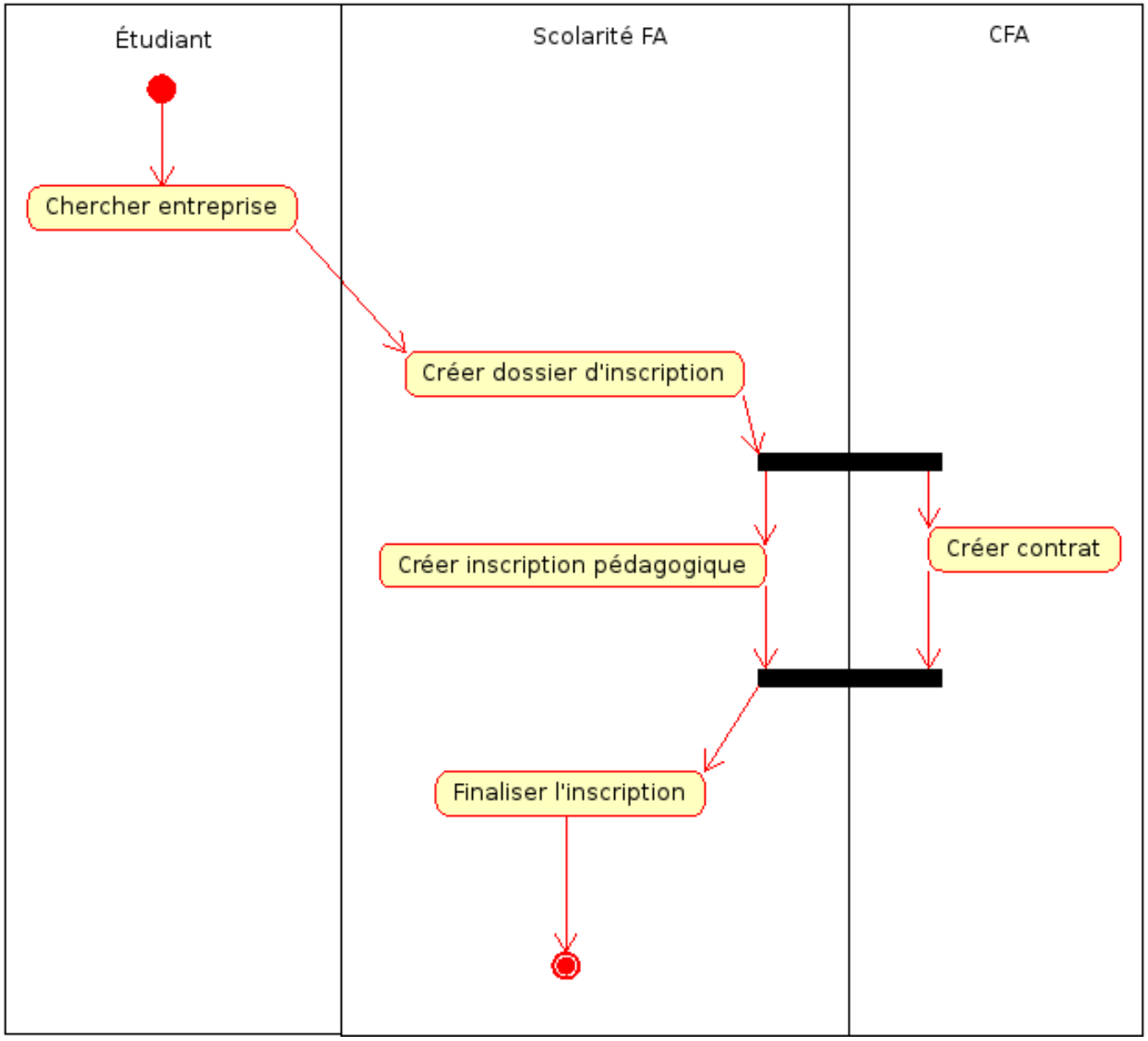


La hiérarchie

- Une activité peut elle-même être décomposée.
- En entrant dans l’activité, on entre dans le point d’entrée du diagramme d’activité correspondant.
- En passant par n’importe quel point de sortie, on a terminé l’activité.

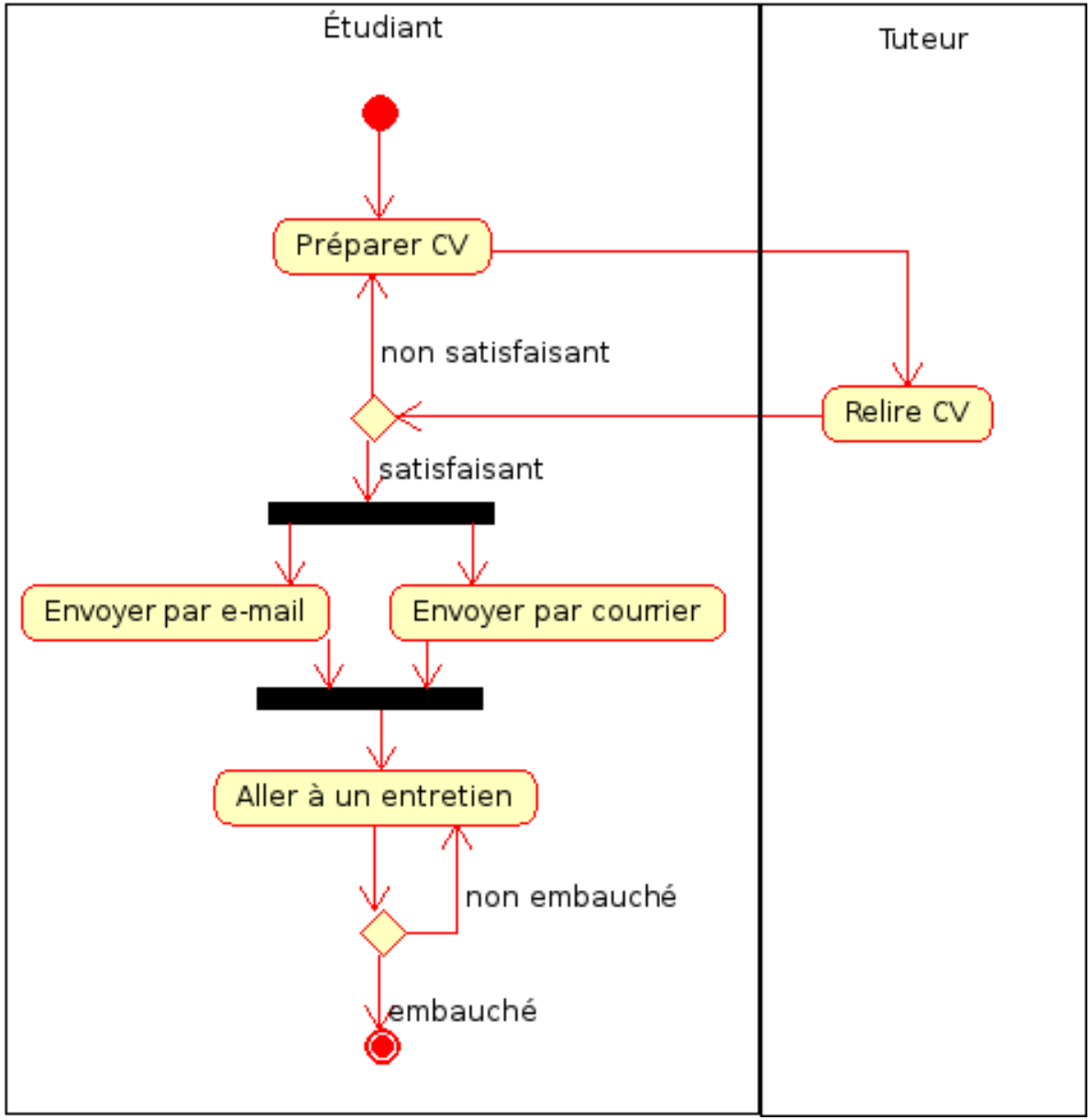
Qui fait quoi ?

Couloir de nage (Swimlanes)



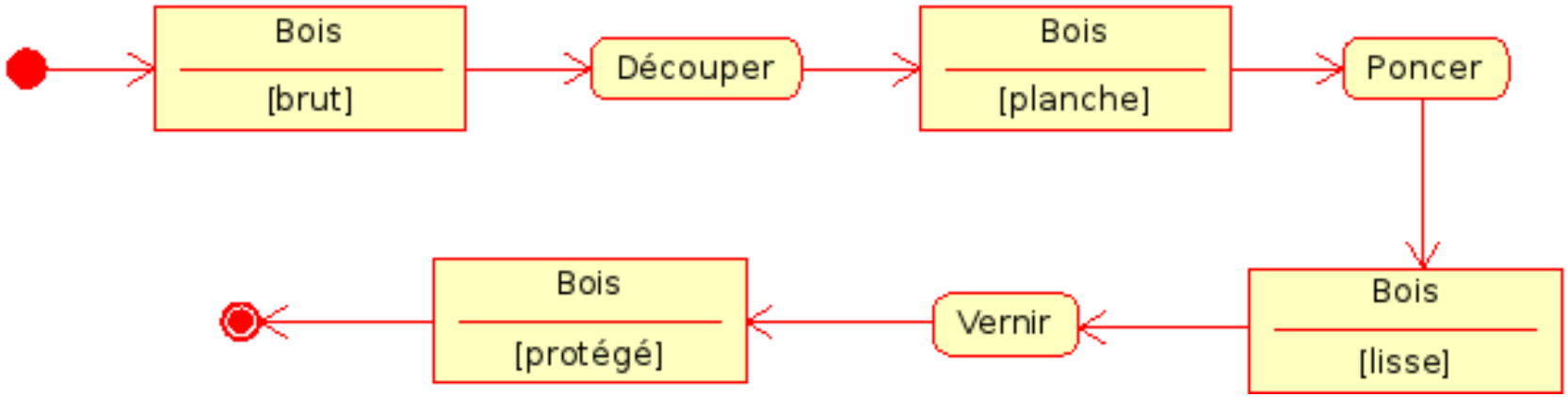
La hiérarchie — Exemple

Activité : « Chercher entreprise »



Les objets

- Pour indiquer lorsqu'il y a interaction à travers un même objet.
- Pour indiquer les changements d'états des objets induits par l'activité.
- Syntaxe : boîte entre deux activités ; l'état de l'objet est précisé.



Exemple

Activité : « Chercher entreprise »avec objet CV