ALGORITHMIQUE · CHAPITRE 27 · REPRÉSENTATIONS

Découvrir UML avec un diagramme d'activité

Comprenez le rond initial, les actions, gardes, décisions, fusions et cercle final d'une activité UML. Suivez les entrées et sorties pas à pas.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Reconnaître début, action, décision, fusion et fin d'activité.
- Lire les gardes des branches.
- Distinguer une fusion de chemins d'une synchronisation parallèle.

Prérequis. Savoir suivre un organigramme avec SI / SINON.

Pourquoi nommer la notation utilisée ?

Deux schémas peuvent décrire la même méthode avec des conventions différentes. **UML** est un langage de modélisation comportant plusieurs types de diagrammes. Nous utilisons ici seulement un petit diagramme d'activité : il montre le déroulement d'un traitement, sans classes, objets métier ou architecture logicielle.

Entrées, sorties et nature du diagramme UML

UML signifie *Unified Modeling Language*, un langage de modélisation. Une **activité** représente ici un traitement composé d'étapes. Un **nœud** est un élément du diagramme ; un **flux de contrôle** est une flèche qui indique la possibilité de passer au nœud suivant. Nous étudions un seul parcours séquentiel, sans tâches exécutées en parallèle.

L'**entrée** du problème est `quantity`, une quantité entière non négative déjà disponible au démarrage. La **sortie** observable est un message : `"Processing order"` si la quantité est positive, `"Empty order"` si elle vaut zéro. Le diagramme ne réalise ni paiement ni réservation. Le code se limite à rendre le choix visible.

Une entrée de problème et une flèche entrante ne sont pas synonymes. L'entrée de problème est une donnée ; une flèche entrante de contrôle indique d'où l'exécution arrive. De même, la flèche qui sort d'un losange est une possibilité de continuation, pas le résultat retourné par une fonction. Dans cet exemple, les valeurs sont données par le contrat et les libellés, sans dessiner de flux d'objets UML.

Les **flux d'objets** UML représentent le passage de données entre actions. Les **broches**, ou *pins*, sont de petits carrés qui peuvent représenter des entrées ou sorties d'une action. Nous ne les utilisons pas dans ce premier schéma : il serait trompeur de présenter les ronds de début et de fin comme des ports de données. La légende suivante explique chaque forme effectivement présente.

Sept éléments expliqués avant le premier parcours

Symboles et sens de lecture de notre activité UML

Élément	Signification	Exemple dans ce diagramme
● **Début**	**Nœud initial** : le rond noir marque le départ du flux de contrôle. Aucune flèche de contrôle ne doit y entrer.	Avec quantity = 2 ou 0, le parcours commence au même endroit. Le rond ne contient pas cette quantité.
▭ **Action**	**Action** : le rectangle arrondi décrit une étape du traitement. Son libellé dit ce qui est exécuté.	PRINT "Empty order" demande un affichage ; la donnée affichée est le texte, pas la forme du rectangle.
◇ **Décision**	**Décision** : le losange sélectionne un chemin selon des conditions, appelées gardes.	Le choix porte ici sur quantity > 0. Pour quantity = 2, le chemin correspondant peut être suivi.

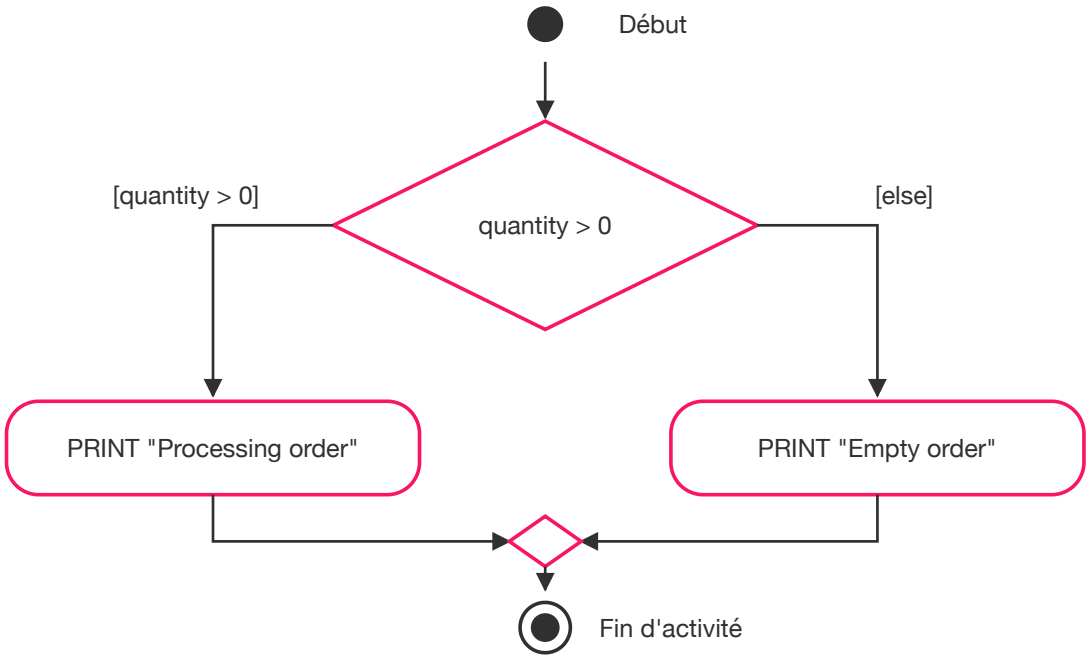
Élément	Signification	Exemple dans ce diagramme
[else] **Garde**	**Garde** : une condition entre crochets sur une flèche autorise le passage si elle est vraie.	[quantity > 0] autorise la première branche. [else] signifie qu'aucune autre garde du choix n'est vraie : ici quantity = 0 dans le domaine annoncé.
 **Fusion**	**Fusion** : le losange réunit des chemins alternatifs. Il ne teste pas une nouvelle condition et n'attend pas plusieurs branches simultanées.	Après un seul affichage, le parcours rejoint ce losange puis poursuit vers la fin.
 **Fin**	**Fin d'activité** : le disque entouré d'un cercle arrête l'activité représentée lorsqu'il est atteint.	Ce symbole n'est ni le nombre zéro, ni un port de sortie. Le résultat est le message déjà affiché.
 **Flèche**	**Flux de contrôle** : la flèche indique dans quel sens passer à la suite du traitement.	Une flèche entrante indique d'où arrive le parcours ; une flèche sortante indique où il peut continuer. Ce sont des directions, pas les données du problème.

Le losange de décision et le losange de fusion ont la même forme ; leurs connexions permettent de les distinguer. Ici, la décision reçoit une flèche et en propose deux, tandis que la fusion en reçoit deux et n'en propose qu'une. Les deux gardes de notre choix sont exclusives pour les entrées autorisées : on ne doit pas afficher les deux messages lors d'un même parcours.

Un **jeton de contrôle** est une façon de représenter la possibilité de poursuivre l'exécution. Pour ce premier exemple séquentiel, imaginez un repère qui avance le long d'un seul chemin. Ce repère n'est pas la valeur de quantity. Nous n'introduisons pas les règles des activités parallèles ; une barre de synchronisation, qui concerne des chemins simultanés, serait une autre notion.

Lire une activité simple

Une branche est choisie, puis les chemins alternatifs se rejoignent avant la fin de l'activité.



▼ Lire les parcours sans animation

quantity = 2

1. Début du parcours avec quantity = 2. Les données sont déjà disponibles.
2. Test 1 : quantity > 0 donne true (vrai).
3. Action choisie : PRINT "Processing order".
4. Les chemins alternatifs se rejoignent. Une seule branche a été exécutée ; il n'y a pas deux tâches à attendre.
5. Fin du parcours. Résultat observable : Processing order.

quantity = 0

1. Début du parcours avec quantity = 0. Les données sont déjà disponibles.
2. Test 1 : quantity > 0 donne false (faux).
3. Action choisie : PRINT "Empty order".
4. Les chemins alternatifs se rejoignent. Une seule branche a été exécutée ; il n'y a pas deux tâches à attendre.

5. Fin du parcours. Résultat observable : Empty order.

L'entrée est une quantité entière non négative. Si elle est positive, l'action représente le traitement choisi ; sinon, elle signale une commande vide. Le losange inférieur ne demande pas d'attendre deux branches : une seule branche a été prise. Il s'agit d'une **fusion**, pas d'une synchronisation parallèle.

Conserver le même choix

Pseudo-code

```
// example.pseudo
// process_quantity
quantity <- 2
IF quantity > 0 THEN
    PRINT "Processing order"
ELSE
    PRINT "Empty order"
END IF
```

Python

```python
# process_quantity.py
quantity = 2
if quantity > 0:
    message = "Processing order"
else:
    message = "Empty order"
print(message)
```

MATLAB

```
% process_quantity.m
quantity = 2;
if quantity > 0
    message = 'Processing order';
else
    message = 'Empty order';
end
disp(message);
```

Le code de démonstration remplace l'action métier par un message. Cela rend le chemin observable sans inventer un système complet de gestion de commandes. Le rôle du diagramme est d'expliquer l'ordre du traitement ; il ne transforme pas ce message en application métier.

Suivre deux entrées

Deux chemins d'une même activité

quantity	Garde vraie	Action observée
0	[else]	PRINT "Empty order"
2	[quantity > 0]	PRINT "Processing order"

Refaites la lecture avec une quantité de 1 : elle suit le même chemin que 2. Une quantité négative reste hors du contrat ; une implémentation plus robuste devrait traiter cette entrée avant de suivre ce diagramme.

Éviter une interprétation trompeuse

▼ Deux sorties [quantity ≥ 0] et [quantity ≤ 0] constituent-elles nos deux cas exclusifs ?

Non : pour zéro, les deux gardes seraient vraies. Pour représenter ce choix exclusif, il faut des conditions complémentaires, par exemple quantity > 0 et son contraire sur les entrées autorisées.

▼ Pourquoi ne pas dessiner deux tâches parallèles pour les deux affichages ?

Parce que le programme n'exécute qu'un seul affichage. Introduire du parallélisme raconterait un autre comportement. Un diagramme plus sophistiqué serait ici moins fidèle.

Garder un petit vocabulaire précis

Vous n'avez pas besoin de maîtriser tout UML pour lire cette activité. Conservez la légende et comparez-la à l'organigramme usuel du chapitre précédent. Les conventions complètes du langage de modélisation sont décrites dans la **spécification UML de l'OMG**.

À retenir - fiche mémo

Ronds

Le disque noir démarre le flux ; le disque entouré d'un cercle termine l'activité.

Choix

La décision sélectionne une branche selon une garde ; la fusion réunit des alternatives.

Données

Les ronds ne sont pas les ports d'entrée/sortie de la quantité ou du message.

Pseudo-code

```
// example.pseudo
// activity_guards
// First branch: [quantity > 0]
// Other branch: [else]
// Exactly one branch for the stated input domain.
```

Point de vigilance. La fusion de ce diagramme n’attend pas deux branches exécutées simultanément.

Consulter la cheatsheet : Organigrammes et UML · **Télécharger la fiche du thème en PDF**

Version web : https://www.clearcode.fr/courses/programming/algorithmic/representations/uml-activities