

Chapitre 4 :

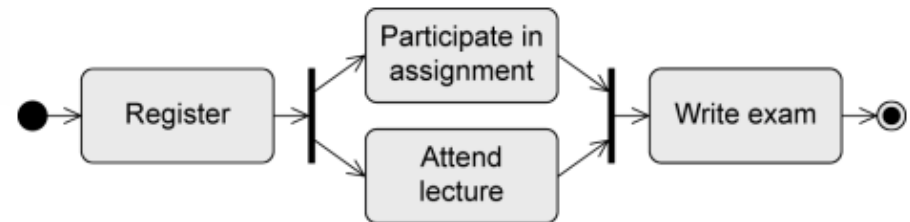
Diagramme d'activité



Plan

2

- Introduction
- Activités
- Actions
- Transitions
 - Flot de contrôle
 - Flot d'objets
- Nœud initial, nœud de fin d'activité, nœud de fin de flot
- Chemins alternatifs et chemins concurrents
- Nœuds d'objet
- Actions prédéfinis
- Partitions
- Gestion des exceptions



Introduction

3

- Les **diagrammes d'activité** offrent une manière graphique et non ambiguë pour modéliser les traitements.
 - Comportement d'une méthode
 - Déroulement d'un cas d'utilisation
 - Déroulement d'un processus métiers
- Ces diagrammes sont assez semblables aux états-transitions mais avec une interprétation différente.

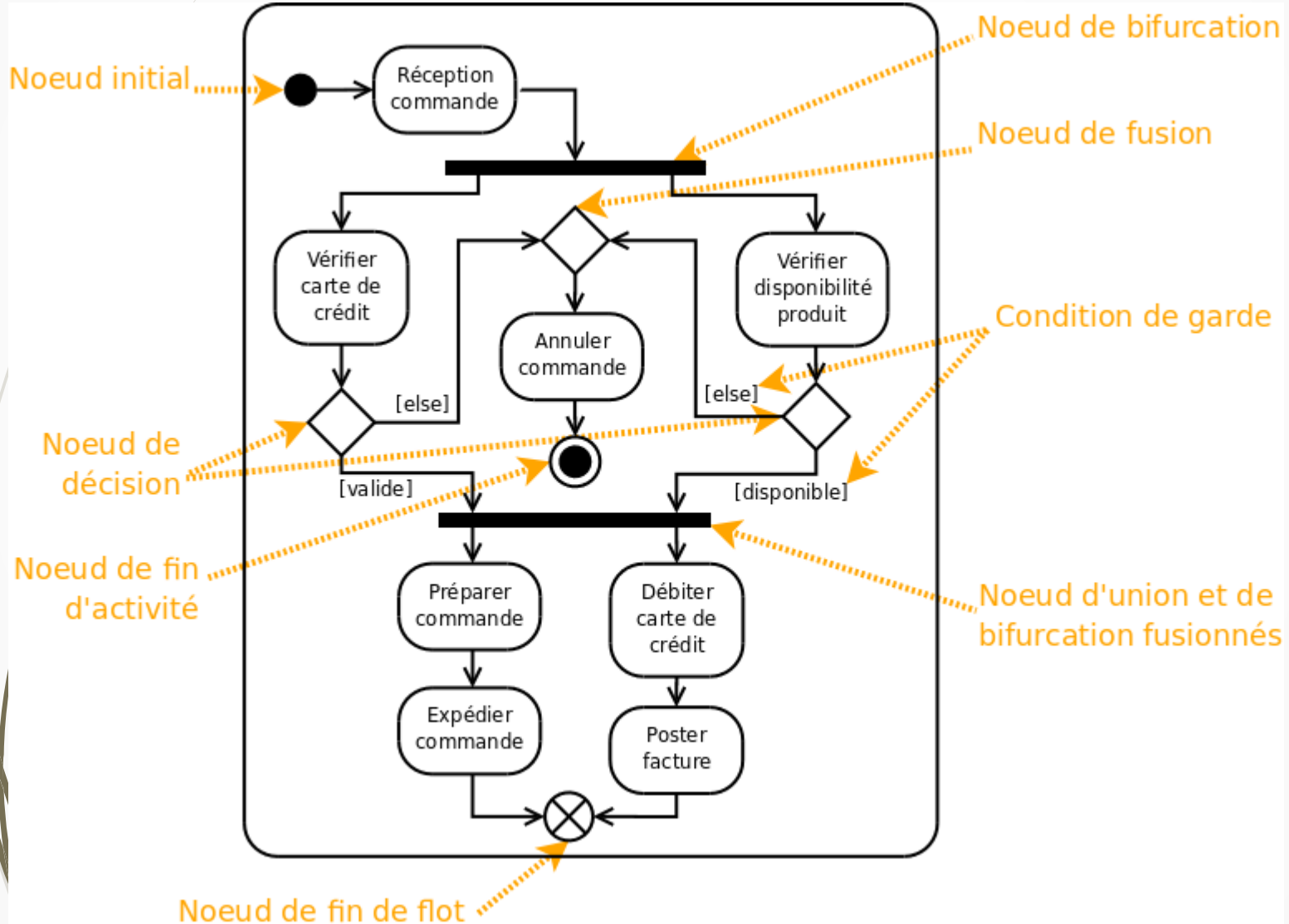
Une vision transversale des traitements

4

- Les diagrammes d'états-transitions sont définis pour un classeur et n'en font pas intervenir plusieurs.
- A l'inverse, les diagrammes d'activité permettent une description s'affranchissant (partiellement) de la structuration de l'application en classeurs.
- La vision des diagrammes d'activités se rapproche des langages de programmation impérative (C, C++, Java)
 - Les états représentent des calculs
 - Il n'y a pas d'événements externes mais des attentes de fins de calculs
 - Il peut y avoir de la concurrence entre activités

Exemple d'un diagramme d'activité

5

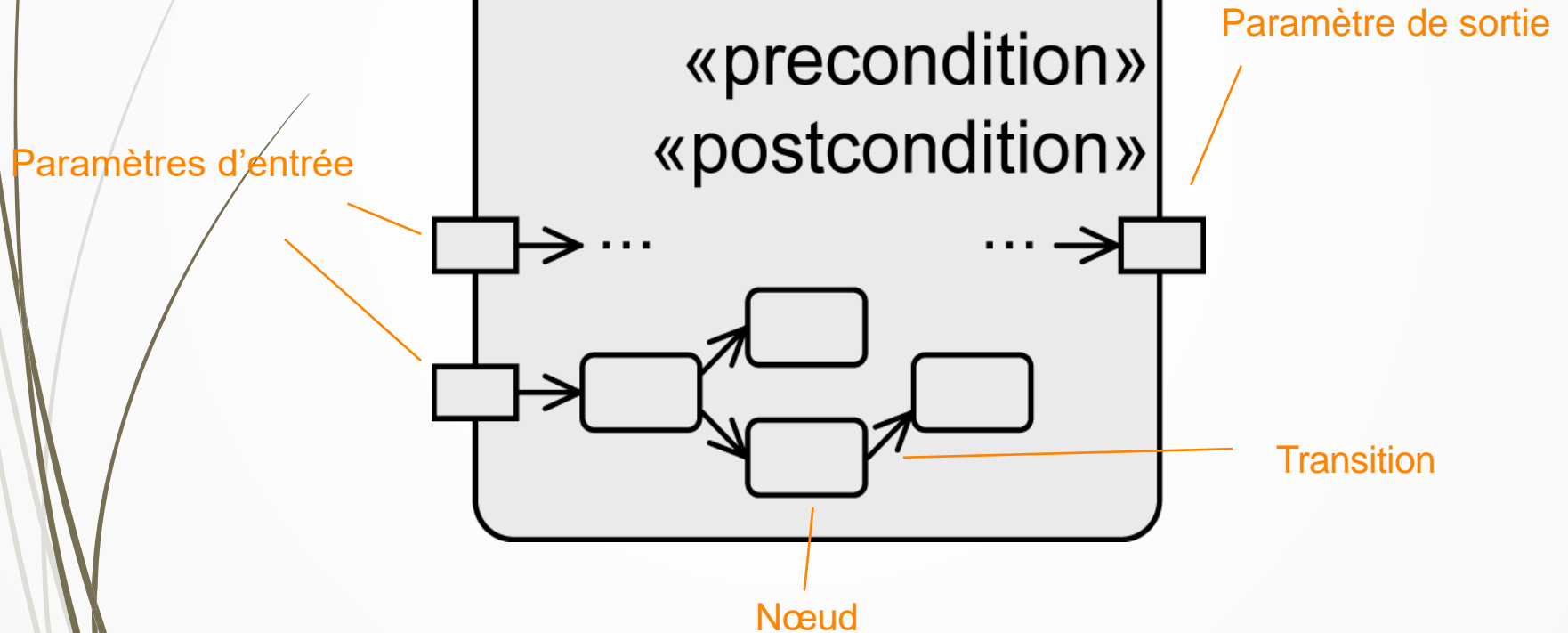


6

- Spécification du comportement défini par l'utilisateur à différents niveaux de granularité
 - Pour une méthode
 - Pour un cas d'utilisation
- Une activité est un graphe orienté
 - Nœuds : actions et activités
 - Transitions : flots de contrôle et flots d'objets
- Le flot de contrôle et le flot d'objets définissent l'exécution
- Facultatif :
 - Paramètre
 - pré et post conditions

Activité

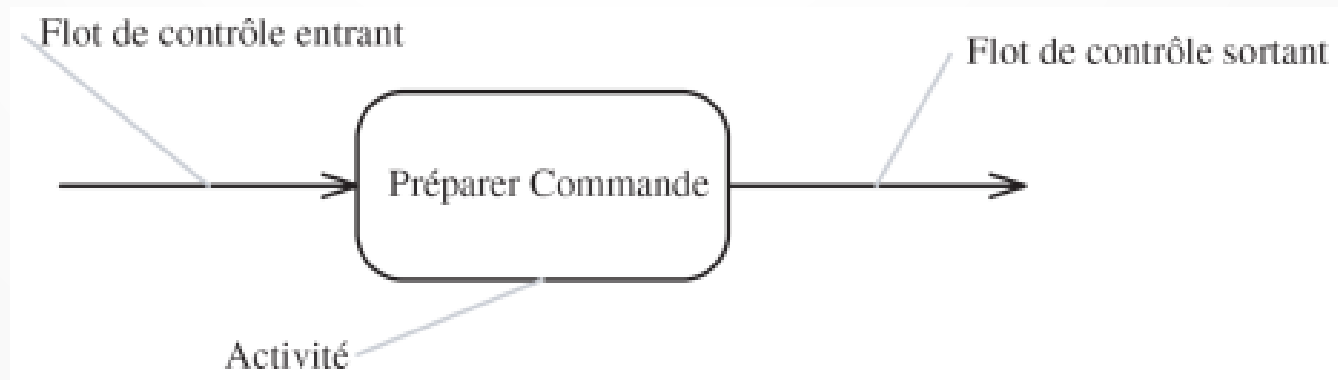
7



Activités

8

- Les activités décrivent un traitement.
- Le flot de contrôle reste dans l'activité jusqu'à ce que les traitements soient terminés.
- On peut définir des variables locales à une activité et manipuler les variables accessibles depuis le contexte de l'activité ,
- Les activités peuvent être imbriquées hiérarchiquement : on parle alors d'activités composites.



Action

9

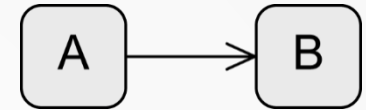
- Les actions sont des étapes discrètes à partir desquelles se construisent les comportements.
- Une action est atomique mais qui peut être interrompu
- Pas de règles spécifiques pour la description d'une action
 - Définition en langage naturel ou dans n'importe quel langage de programmation
- Notation spéciale pour les types **d'actions prédéfinis**, surtout les actions de communication
 - Actions à base des événements (**send signal, accept event, ...**)
 - Actions de comportement d'appel

Action

10

- Une action est le plus petit traitement qui puisse être exprimé en UML.
- La notion d'action est à rapprocher de la notion d'instruction élémentaire d'un langage de programmation (comme C++ ou Java). Une action peut être, par exemple :
 - une affectation de valeur à des attributs ;
 - la création d'un nouvel objet ou lien ;
 - un calcul arithmétique simple ;
 - l'émission d'un signal ;
 - la réception d'un signal ;
 - ...

Transitions



11

- Les transitions sont représentées par des flèches pleines qui connectent les activités entre elles.
 - Elles expriment l'ordre d'exécution
 - Elles sont déclenchées dès que l'activité source est terminée.
 - Elles provoquent automatiquement le début immédiat de la prochaine activité à déclencher (l'activité cible).
- Contrairement aux activités, les transitions sont franchies de manière atomique, en principe sans durée perceptible.

Donc, **les transitions spécifient l'enchaînement des traitements et définissent le flot de contrôle.**

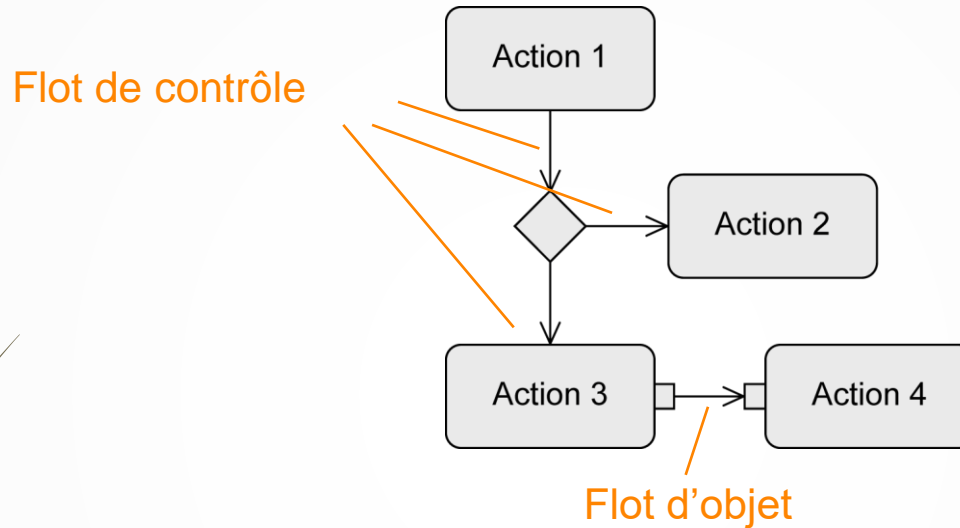
Transitions

12

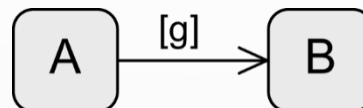
- Les types
 - Transition du flot de contrôle
 - Définir l'ordre entre les nœuds
 - Transition du flot d'objets
 - Utilisé pour échanger des données ou des objets
 - Exprime une dépendance de données/causale entre les nœuds

Transition

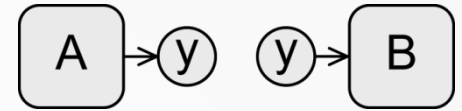
13



- ➡ Transition dotée de **garde** (condition)
- ➡ Les flots de contrôle et d'objets ne continuent que si la condition de garde entre crochets est évaluée à vrai



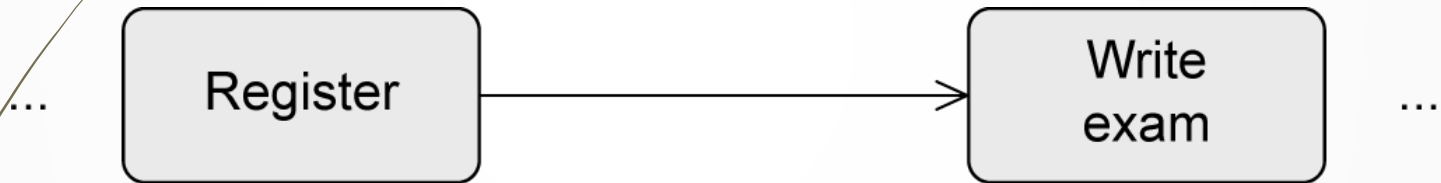
Connecteur



15

- Utilisé si deux actions consécutives sont éloignées dans le diagramme

- Sans connecteur :



- Avec connecteur



Début et fin des activités

16

● Nœud initial

➤ Démarre l'exécution d'une activité

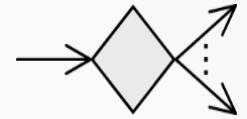
⦿ Nœud de fin d'activité

➤ Termine tous les flots d'une activité

⊗ Nœud de fin de flot

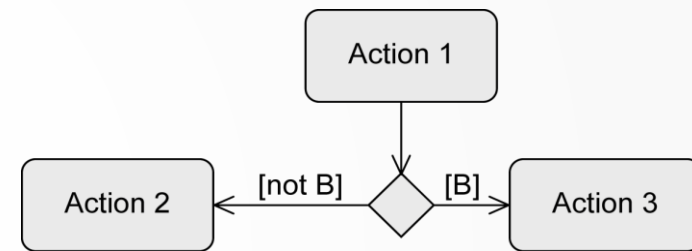
➤ Termine un chemin d'exécution d'une activité

Nœud de décision

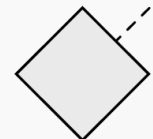


17

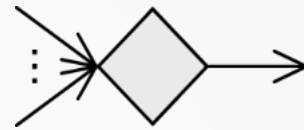
- Le nœud de décision est utilisé pour définir des chemins ou des flots alternatifs
- Les transitions sortantes ont des gardes
 - Syntaxe : [Expression booléenne]
 - Les gardes doivent s'exclure mutuellement
 - Un garde prédéfini : [else]
- Comportement décisionnel
 - Préciser le comportement nécessaire à l'évaluation des gardes



«decisionInput»
decision behavior

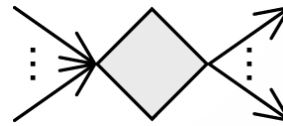


Noëud de fusion

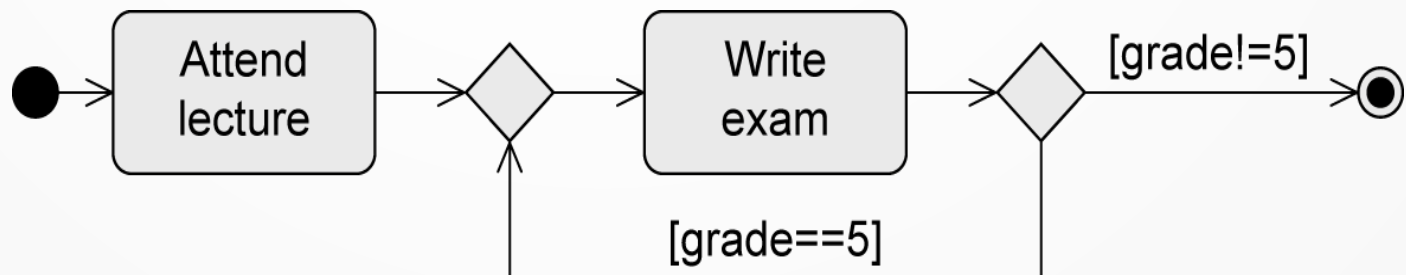


18

- Noëud de fusion
 - Pour rassembler des sous-chemins ou des flots alternatifs
 - A partir de plusieurs flots alternatifs, on peut déclencher un seul flot
- On peut utiliser un nœud qui combine les noeuds de décision et de fusion

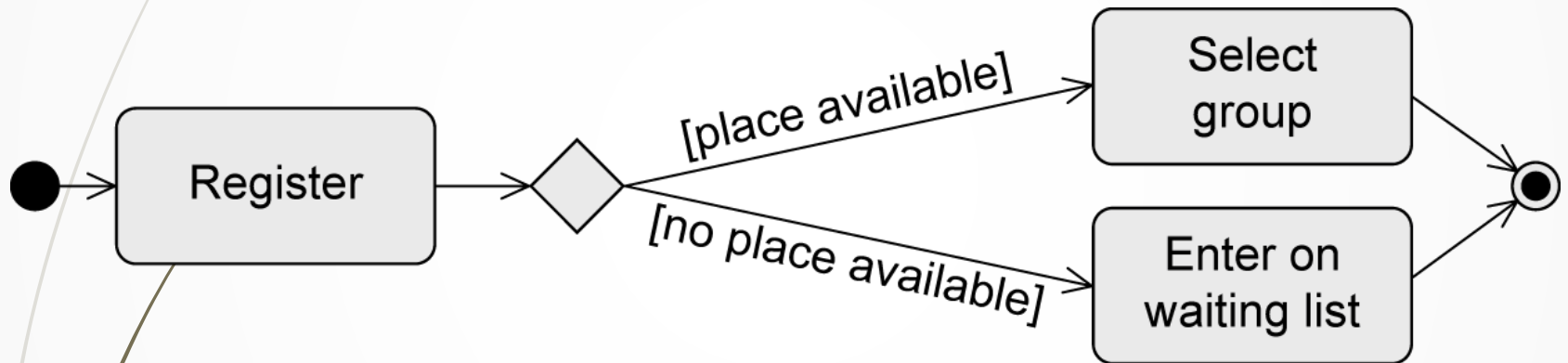


- Les nœuds de décision et de fusion peuvent également être utilisés pour modéliser des boucles :



Exemple : Nœud de décision

19



Activités structurées

20

- Les activités structurées utilisent les structures de contrôle usuelles (conditionnelles et boucle) à travers une syntaxe qui dépend de l'outil.
- La syntaxe précise de ces annotations pas définie dans la norme UML.
- Dans une activité structurée, on définit les arguments d'entrée et les sorties par des flèches encadrées.

```
total=0;  
for (int i=lignes.size()-1 ; i >= 0 ; i++)  
    total += lignes[i].prixUnitaire * lignes[i].quantité
```



lignes : vector<ligneCommande>

total:float

```
if (c==' ' || c=='\t' || c=='\n')  
    return true;  
else  
    return false;
```



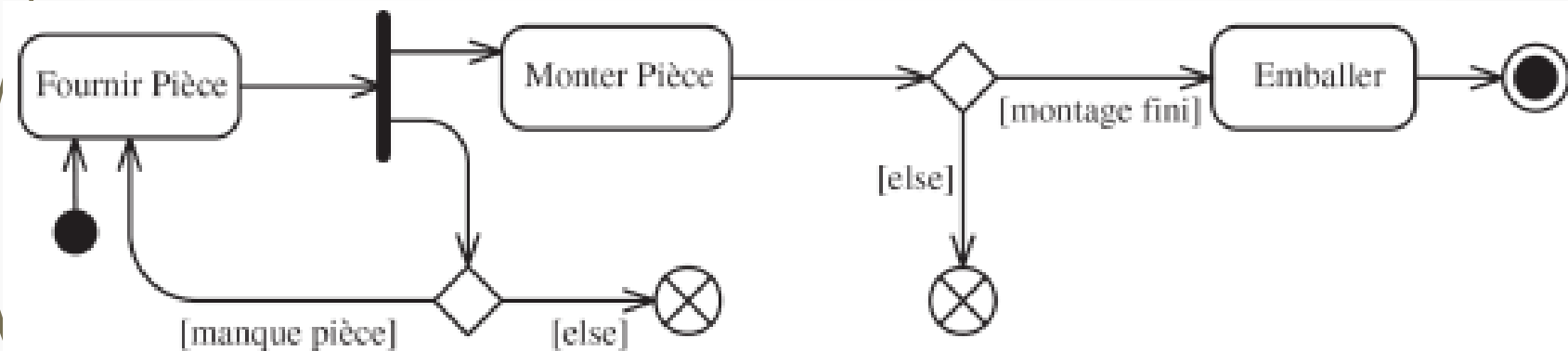
c : char

bool

Concurrence

21

- Les diagrammes d'activités permettent en principe de représenter des activités séquentielles.
- Néanmoins, on peut représenter des **activités concurrentes** avec :
 - Les barres de synchronisation,
 - Les nœuds de contrôle de type « flow final ».

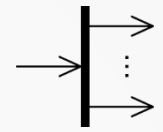


Barre de synchronisation

22

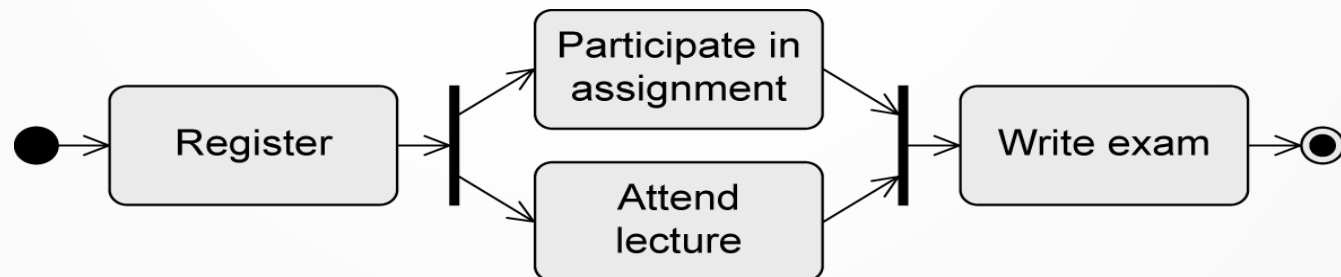
- Barre de synchronisation
 - Permet d'ouvrir et de fermer des branches parallèles au sein d'un flot d'exécution d'une méthode ou d'un cas d'utilisation
 - Les transitions au départ d'une barre de synchronisation sont déclenchées simultanément.
 - Une barre de synchronisation ne peut être franchie que lorsque toutes les transitions en entrée sur la barre ont été déclenchées

Nœud de parallélisation

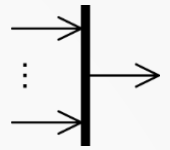


23

- Lorsque la barre de synchronisation a plusieurs transitions en sortie, on parle de transition de type **fork**
 - Le nœud est dit nœud de **parallélisation** (ou de **bifurcation**)
 - Il correspond à une duplication du flot de contrôle en plusieurs flots (sous-chemins) indépendants.
 - Ces flots sont alors des flots concurrents



Nœud de synchronisation

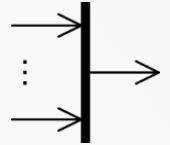


24

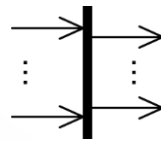
- Lorsque la barre de synchronisation a plusieurs transitions en entrée, on parle de transition de type **join** qui correspond à un rendez-vous entre des flots de contrôle.
- Il est utilisé pour fusionner des sous-chemins ou de flots concurrents
 - Il attend la fin de l'exécution de tous les flots concurrents (entrants) avant de déclencher la transition sortante

Nœud de synchronisation

25



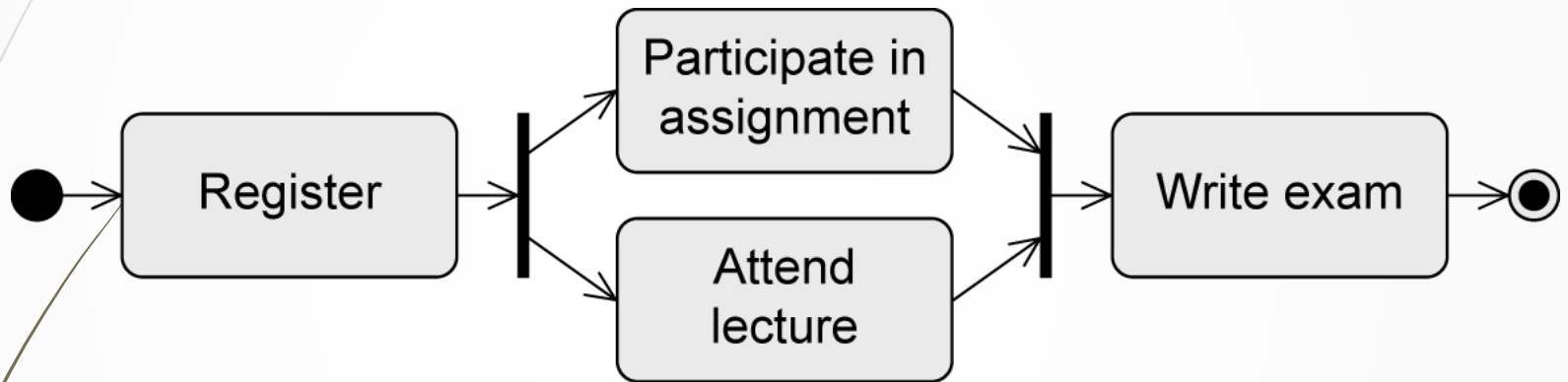
- Pour plus de commodité, il est possible de fusionner des barres de synchronisation de type **join** et **fork**.
- On a alors plusieurs transitions entrantes et sortantes sur une même barre.



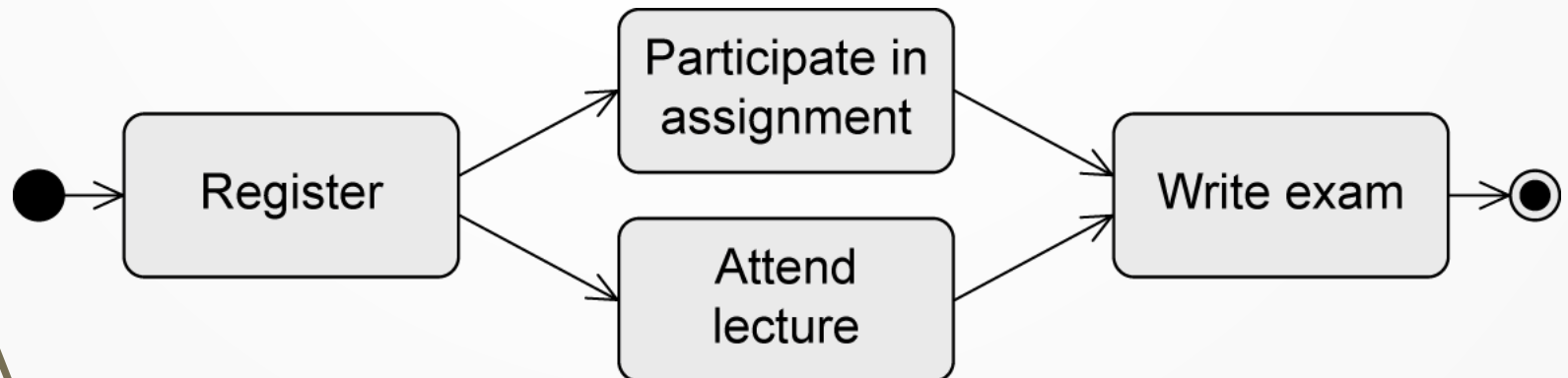
Exemple : Flots de contrôle équivalents

26

- Gérer un cours (point de vue de l'étudiant)

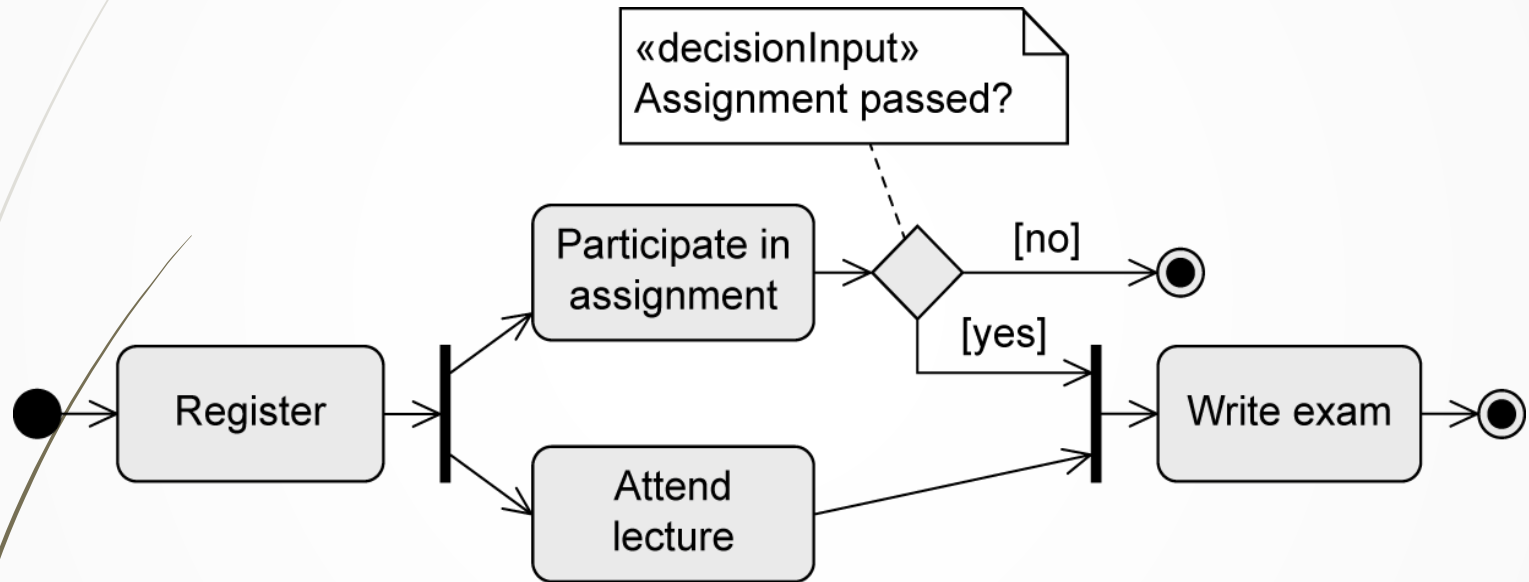


... équivalent à



Example :

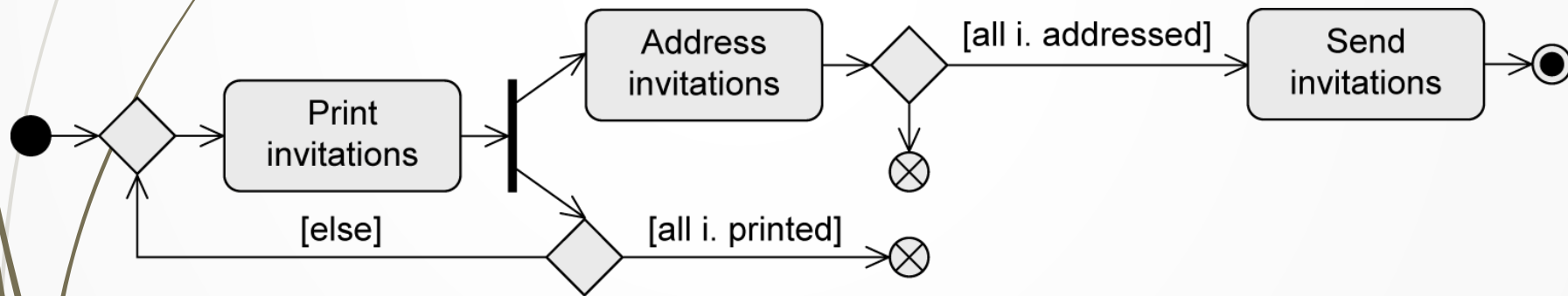
27



Exemple : Créer et envoyer des invitations à une réunion

28

- Pendant que les invitations sont imprimées, les invitations déjà imprimées sont adressées.
- Lorsque toutes les invitations sont adressées, les invitations sont envoyées.



Flot d'objets

29

- Les diagrammes d'activités présentés jusqu'ici montrent bien le comportement du flot de contrôle.
 - Sans prendre en compte les données échangées entre les actions ou les activités
- Un flot d'objets permet de passer des données d'une activité à une autre.
- Représente l'échange de données/objets
 - C'est un conteneur typé qui permet le transit des données.

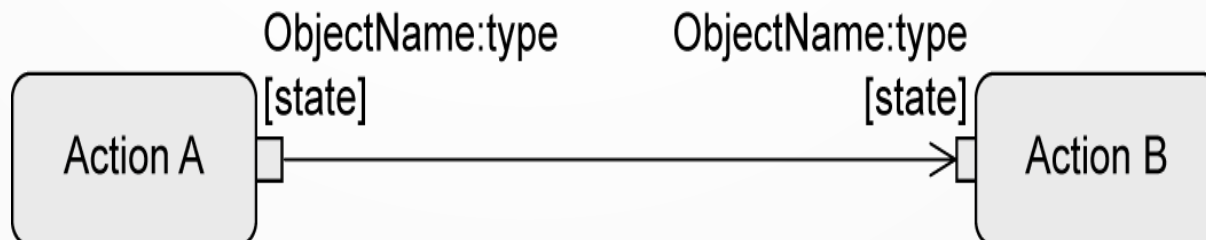
Flot d'objets

30

- Notations alternatives pour les flots d'objet :
 - Un **noeud d'objets** permettent de mieux mettre en valeur les données.



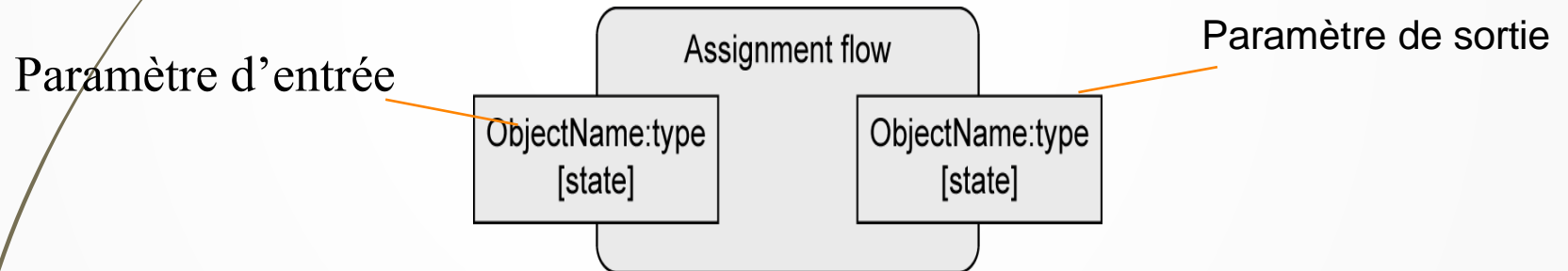
- Un arc qui a pour origine et destination un pin correspond à un flot de données.



Flot de données

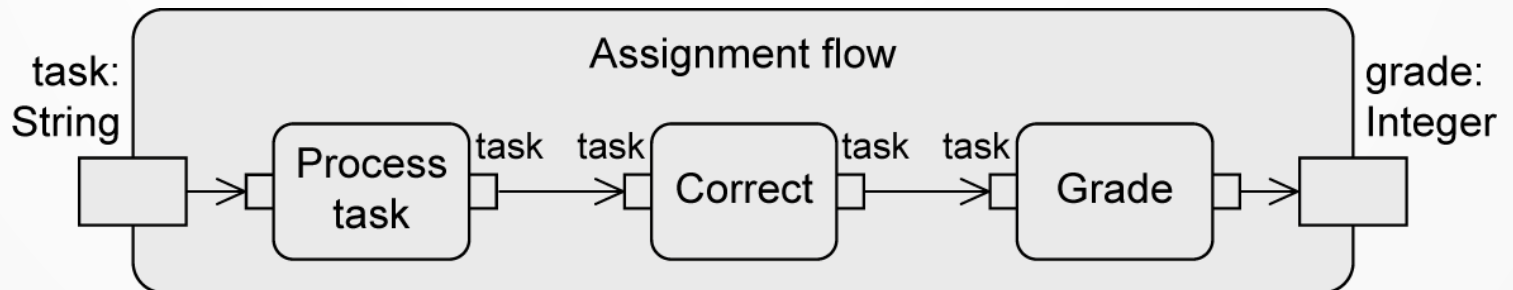
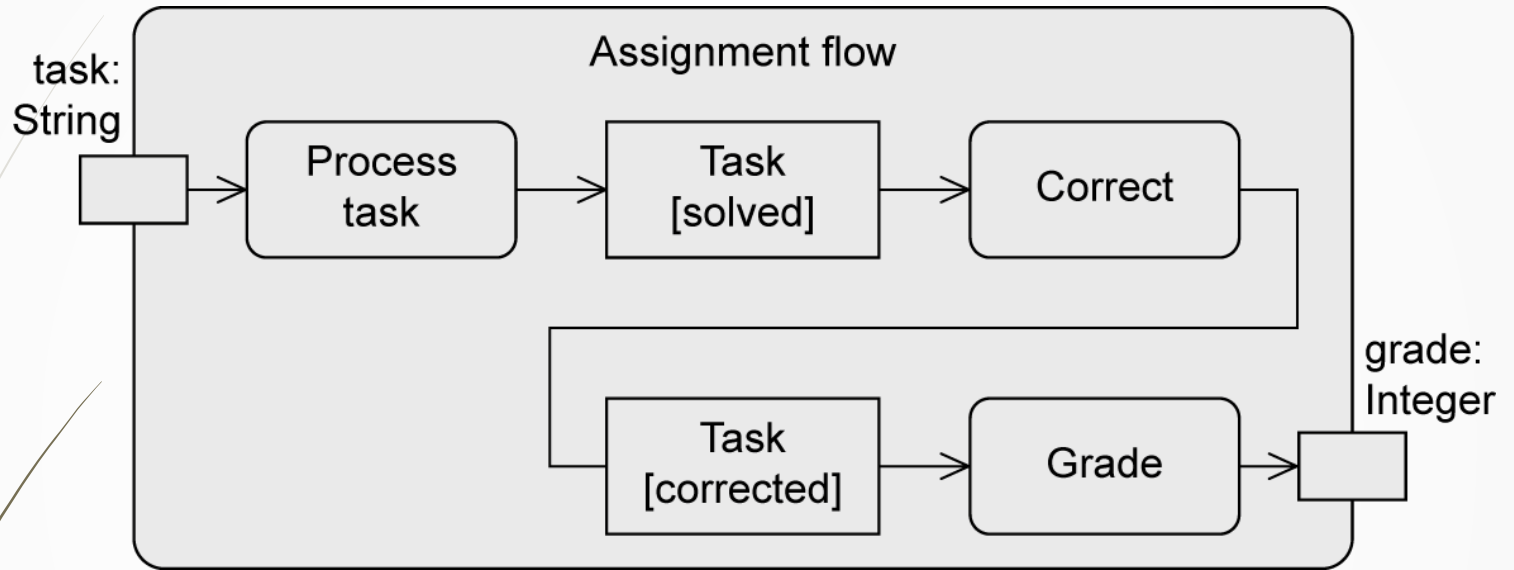
31

- Flots d'objet pour les activités



Exemple : flots d'objets

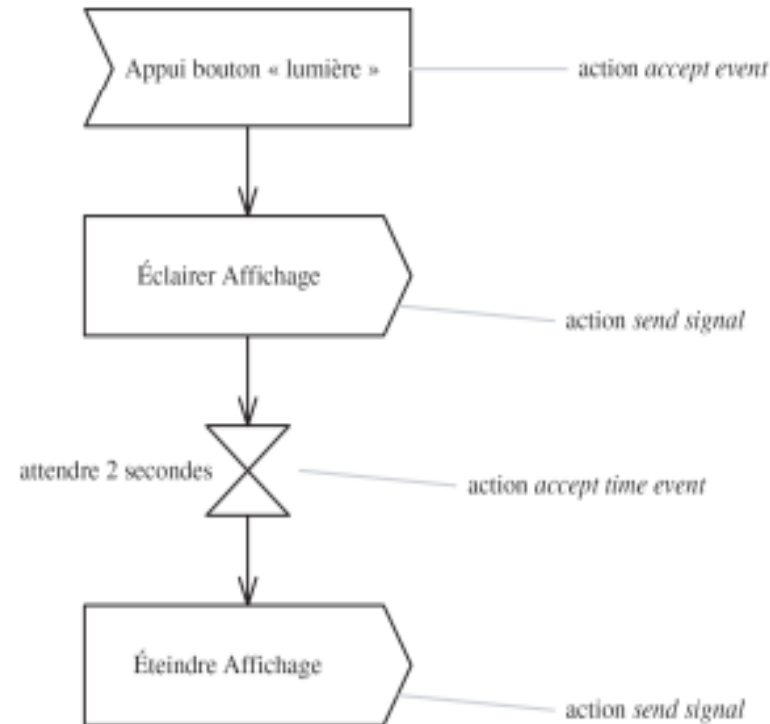
32



Actions de communication

33

- Les actions de communication gèrent le passage et le retour de paramètres, les mécanismes d'appels d'opérations synchrones et asynchrones.
- Les actions de communication peuvent être employés comme des activités dans les diagrammes d'activité.
- Plusieurs types d'actions prédéfinis



Actions de communication

34

- Les actions de type **call** correspondent à des appels de procédure ou de méthode.
 - **Call operation** correspond à l'appel d'une opération sur un classeur.
 - **Call behavior** correspond à l'invocation d'un comportement spécifié à l'aide d'un diagramme UML
 - Dans les deux cas, il est possible de spécifier des arguments et d'obtenir des valeurs en retour.

Appel asynchrone

35

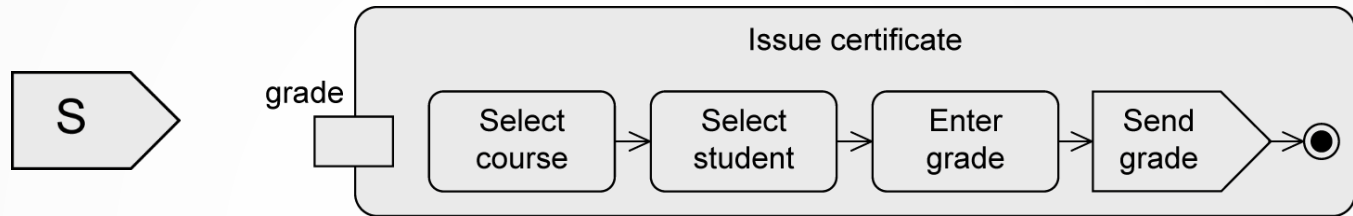
- Les appels asynchrones de type **send** correspondent à des envois de messages asynchrones.
- L'action symétrique côté récepteur est **accept event**, qui permet la réception d'un signal.
- Autres appels asynchrones

Actions à base d'événements

36

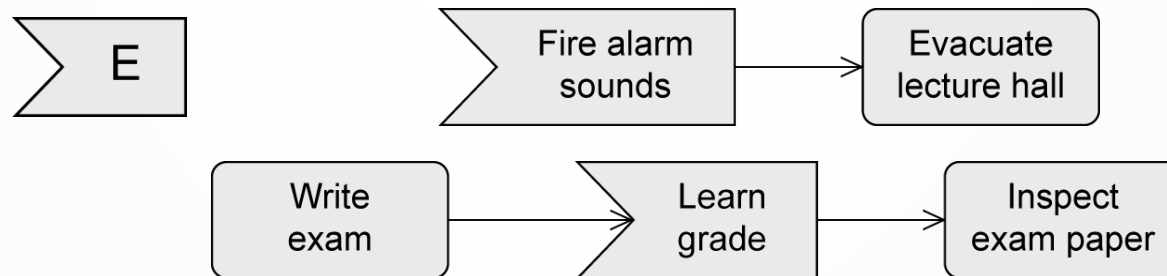
➤ Pour envoyer des signaux

➤ Action **send signal**

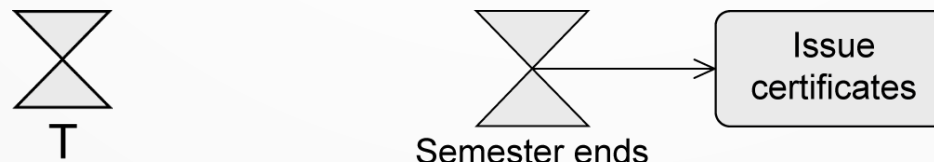


➤ Accepter les événements

➤ Action **accept event**

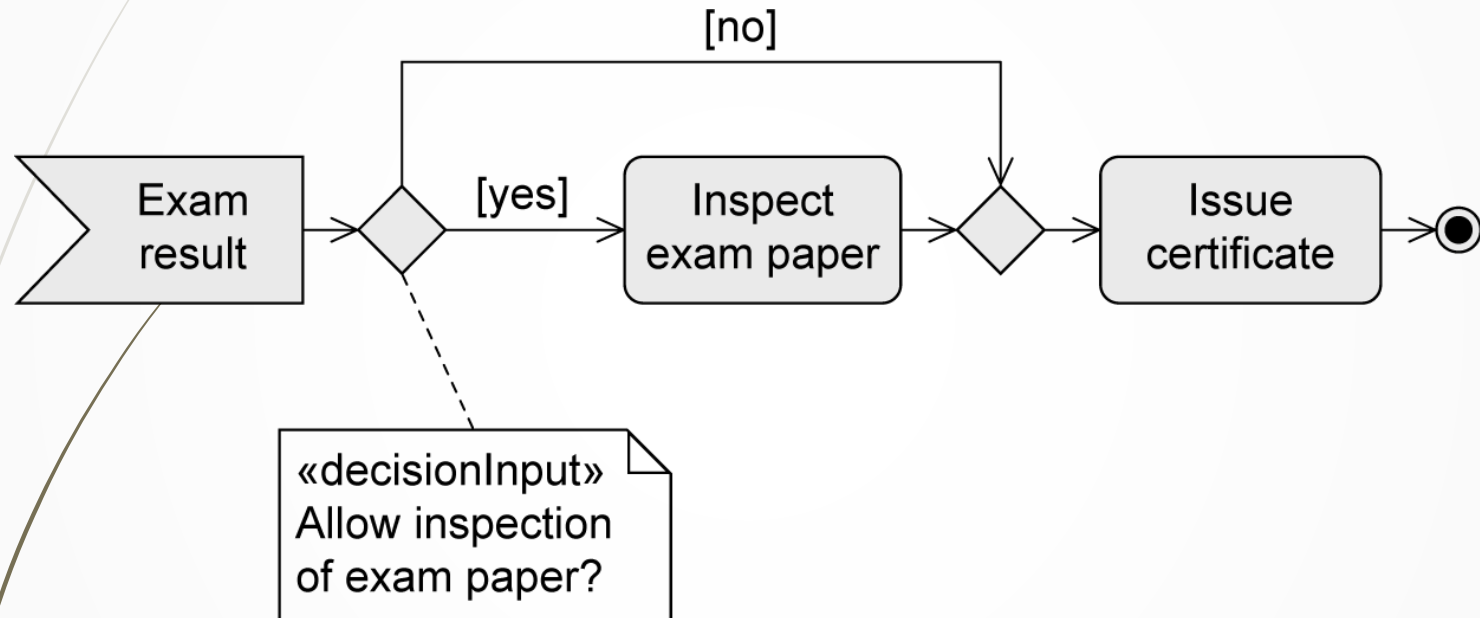


➤ Action **accept temporal event**

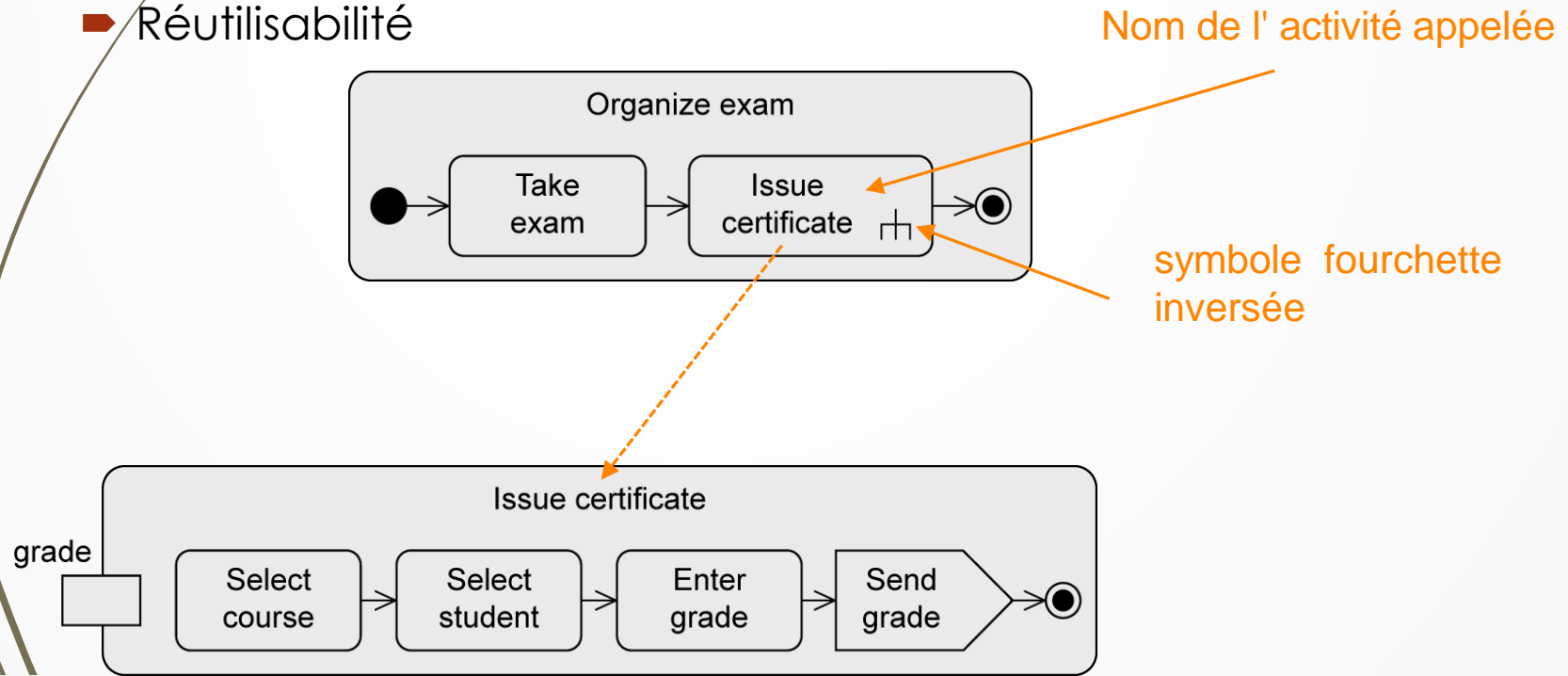


Exemple : Actions à base d'événements

37



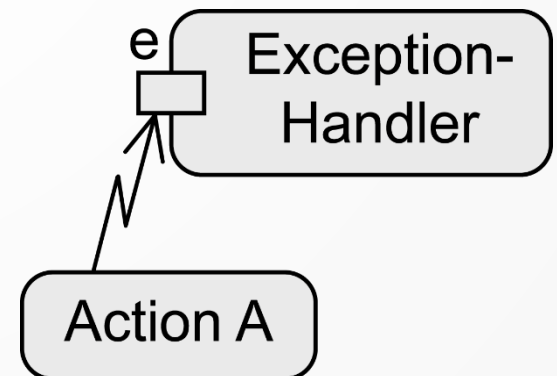
- L'exécution d'une action peut appeler une activité
- Le contenu de l'activité appelée peut être modélisé ailleurs
- Avantages :
 - Le modèle devient plus clair
 - Réutilisabilité



Exceptions

39

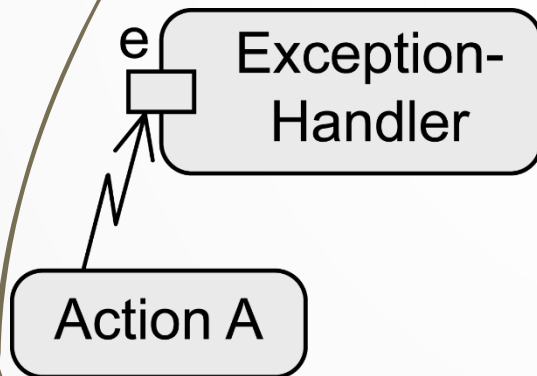
- Les **exceptions** permettent d'interrompre un traitement quand une situation qui dévie du traitement normal se produit.
 - Elles assurent une gestion plus propre des erreurs qui peuvent se produire au cours d'un traitement.
- Un flot de données correspondant à une exception est matérialisé par une flèche en « zigue zague ».



Gestionnaire d'exception

40

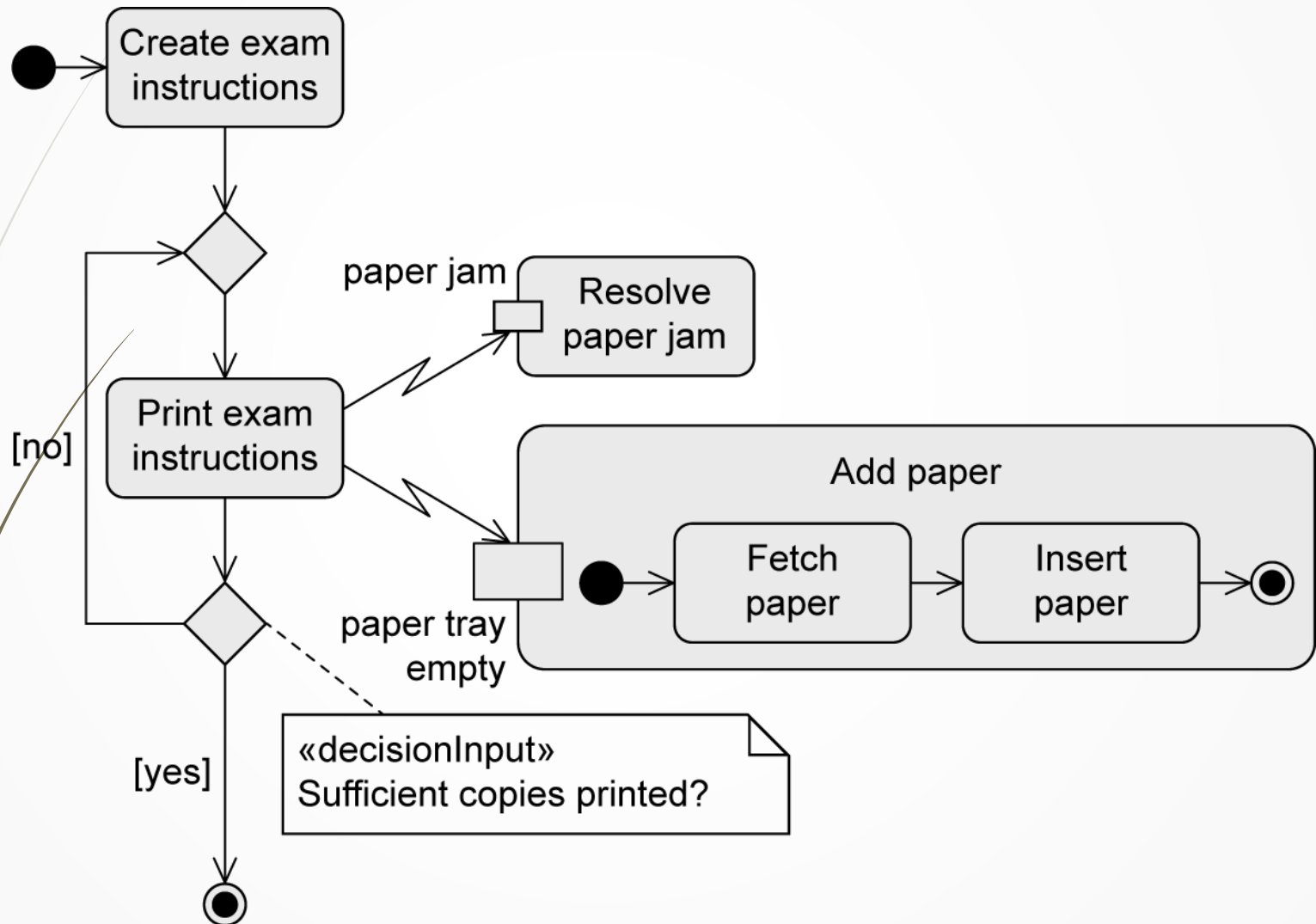
- Définir comment le système doit réagir dans une situation d'erreur spécifique
- Le gestionnaire d'exception remplace l'action où l'erreur s'est produite



- Si l'erreur **e** se produit...
 - Le gestionnaire d'exception est activé
 - Le gestionnaire d'exception est exécuté à la place de **l'action A**
 - L'exécution se poursuit ensuite régulièrement

Exemple : Gestion des exceptions

41



Région Interruptible

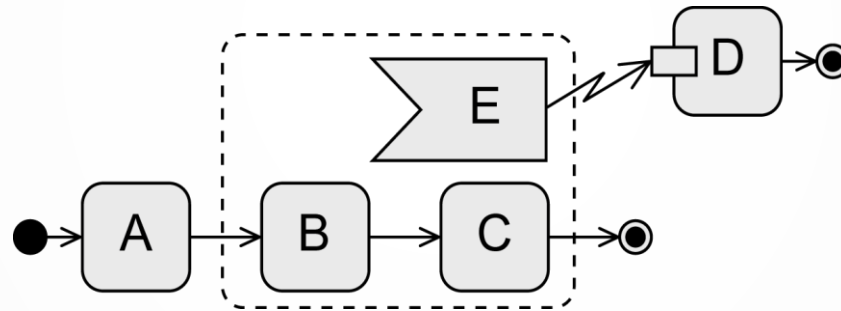
42

- Ce mécanisme est analogue à celui des interruptions, mais il est moins détaillé.
- Les **régions interruptibles** sont mieux adaptées aux phases de modélisation fonctionnelles.
- Définir un groupe d'actions dont l'exécution doit être terminée immédiatement si un événement spécifique se produit. Dans ce cas, un autre comportement est exécuté

Région Interruptible

43

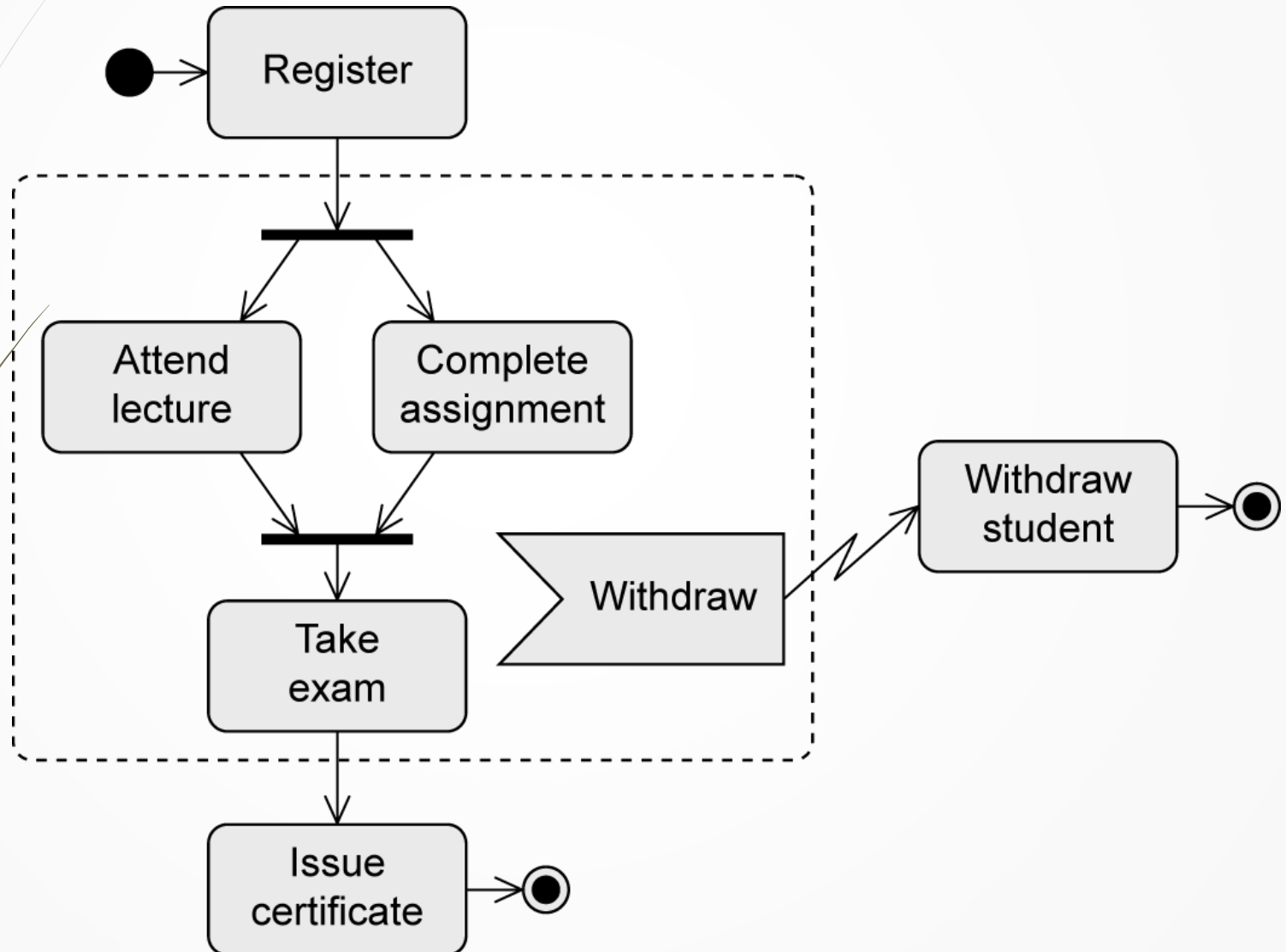
- Une région interruptible est représentée par un cadre arrondi en pointillés.
- Si l'événement d'interruption se produit, toutes les activités en cours dans la région interruptible sont stoppées
- Le flot de contrôle suit la flèche en zigzag qui quitte la région.



- Si **E** se produit pendant que **B** ou **C** sont exécutés
- La gestion des exceptions est activée
- **D** est activé et exécuté
- Pas de « retour en arrière » à l'exécution régulière !

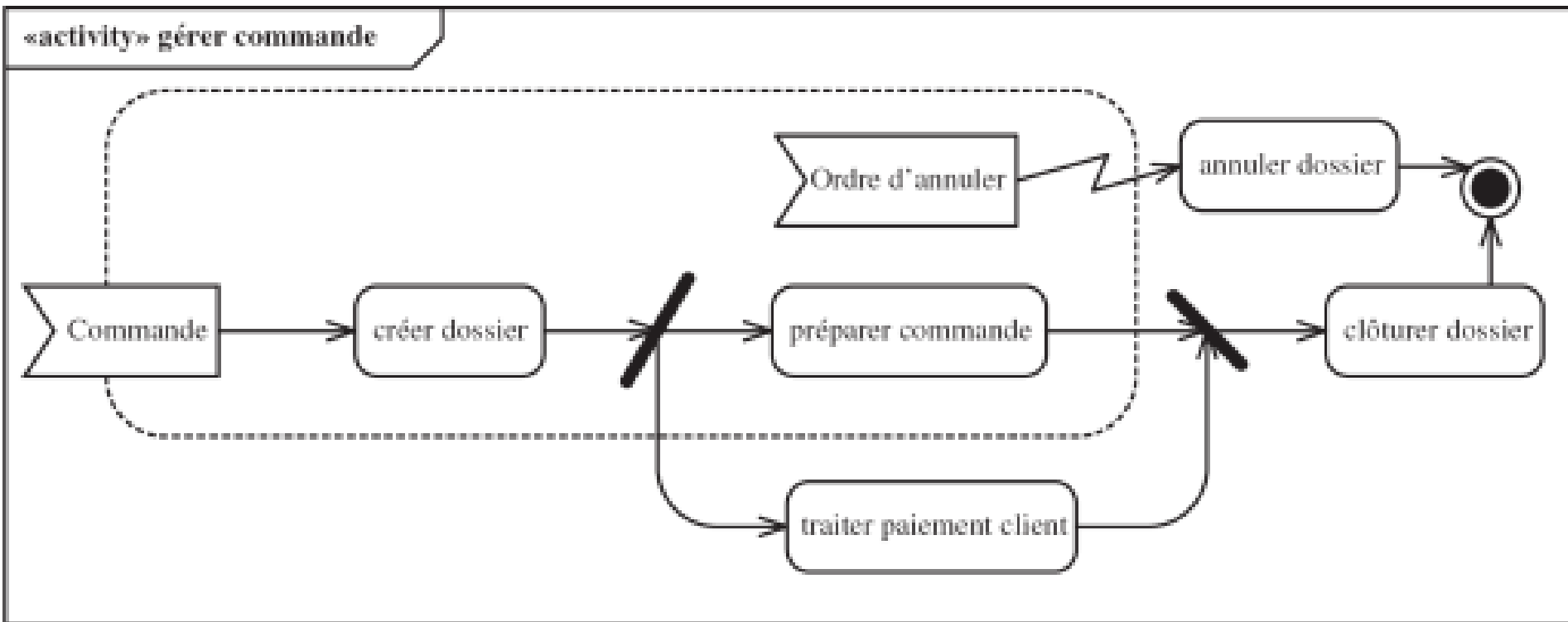
Exemple : Région Interruptible

44



Exemple : Région Interruptible

45

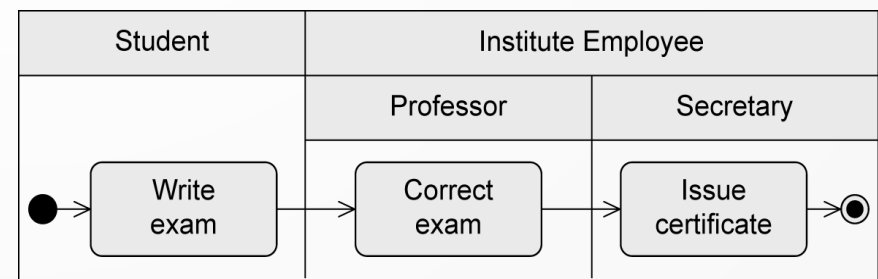


Partition ou Couloir

A	B
B	

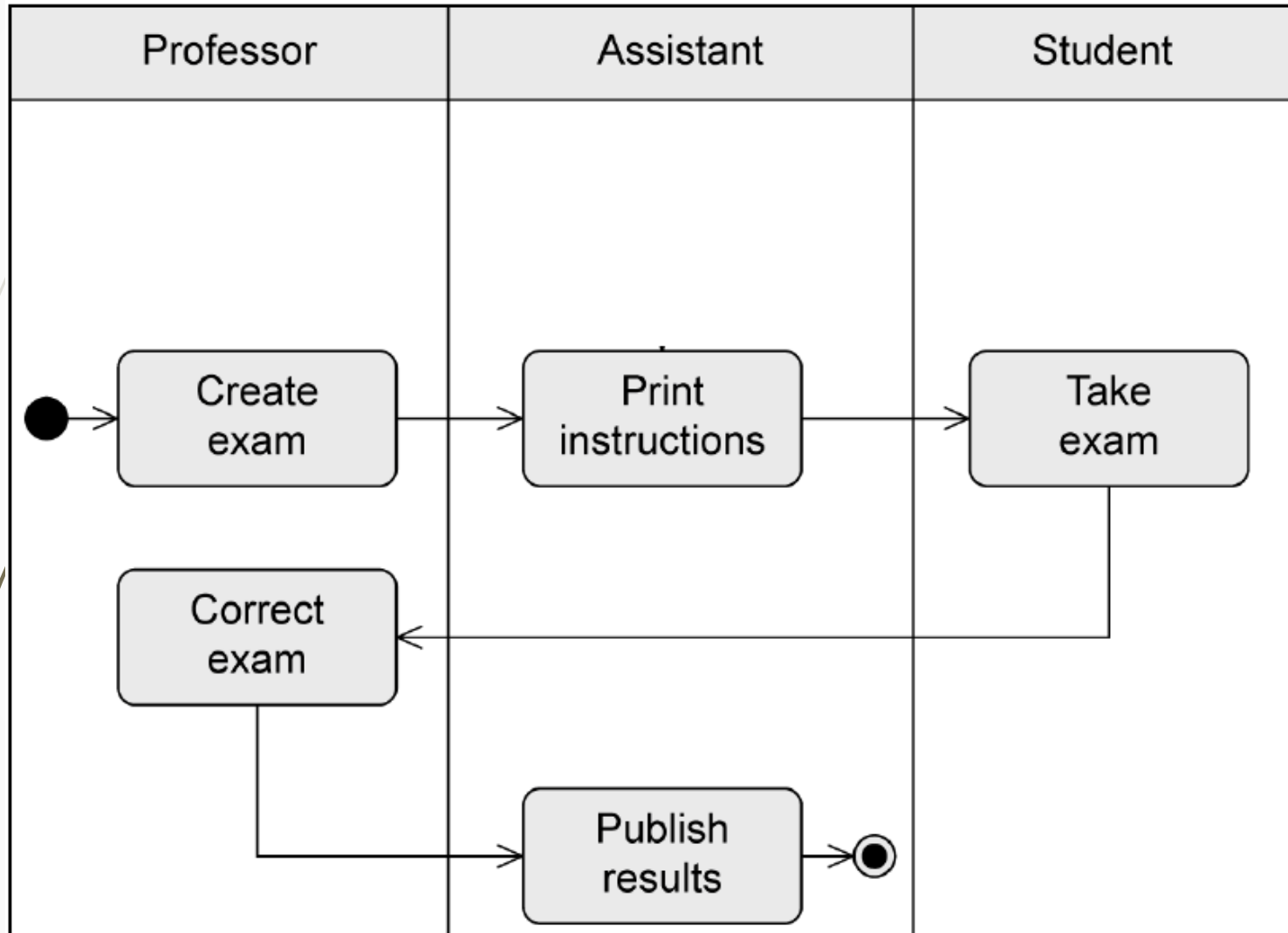
46

- Pour modéliser un traitement mettant en oeuvre plusieurs classeurs, on peut spécifier le classeur responsable de chaque activité.
- Les **partitions** permettent d'attribuer les activités à des classeurs particuliers du modèle.
- Une partition peut elle-même être décomposée en sous-partitions.
- Pour spécifier qu'une activité est effectuée par un classeur particulier, on la positionne dans la partition correspondante.
- Exemple : partitions **Étudiant** et **Employé** de l'Institut (avec sous-partitions **Professeur** et **Secrétaire**)



Partition : Exemple

47



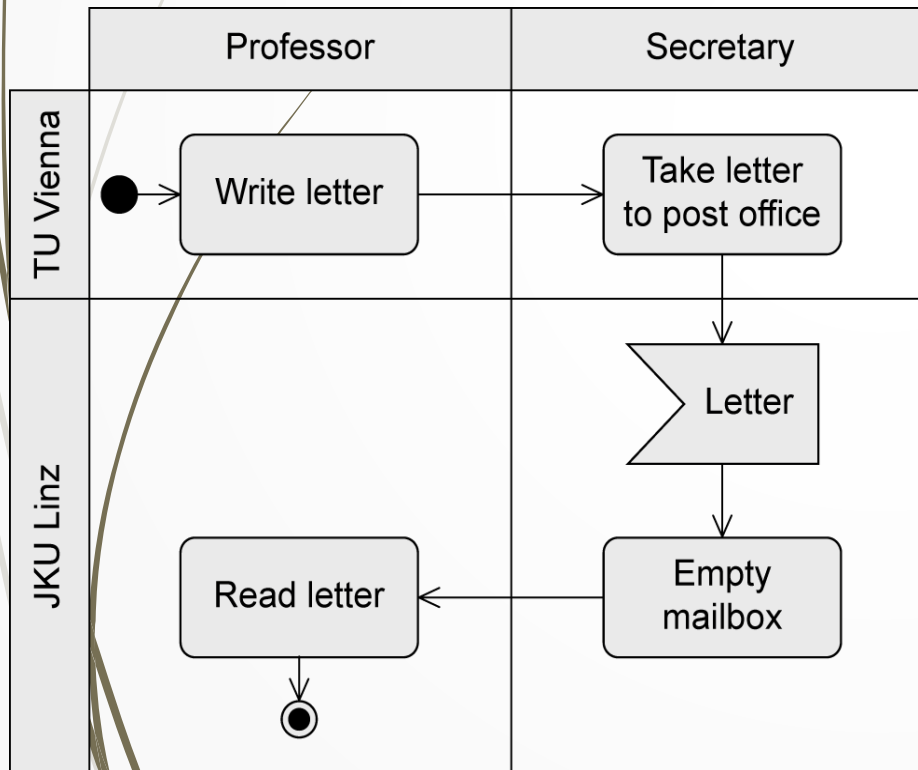
Partition multidimensionnelle

48

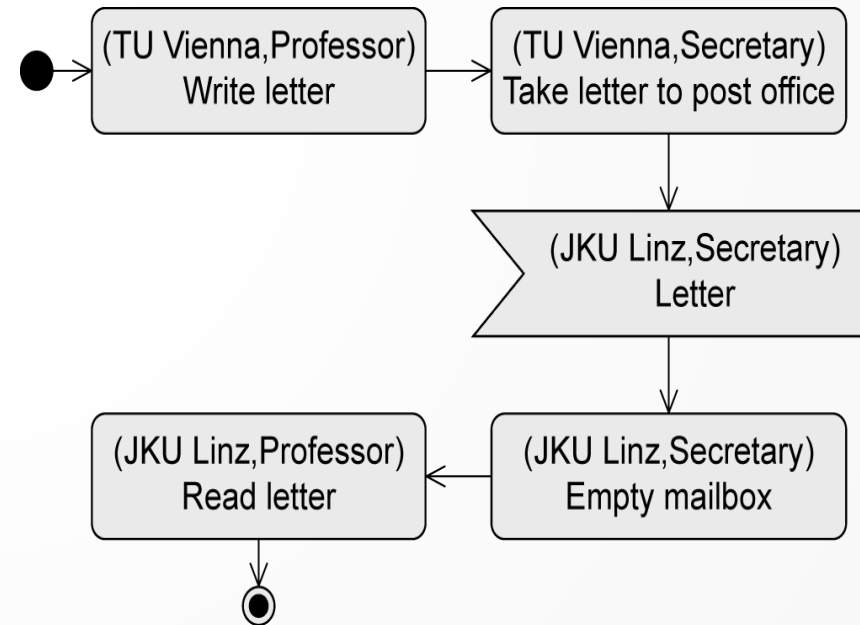
► Partition

- Il y a deux notations : graphique et textuelle

Notation graphique

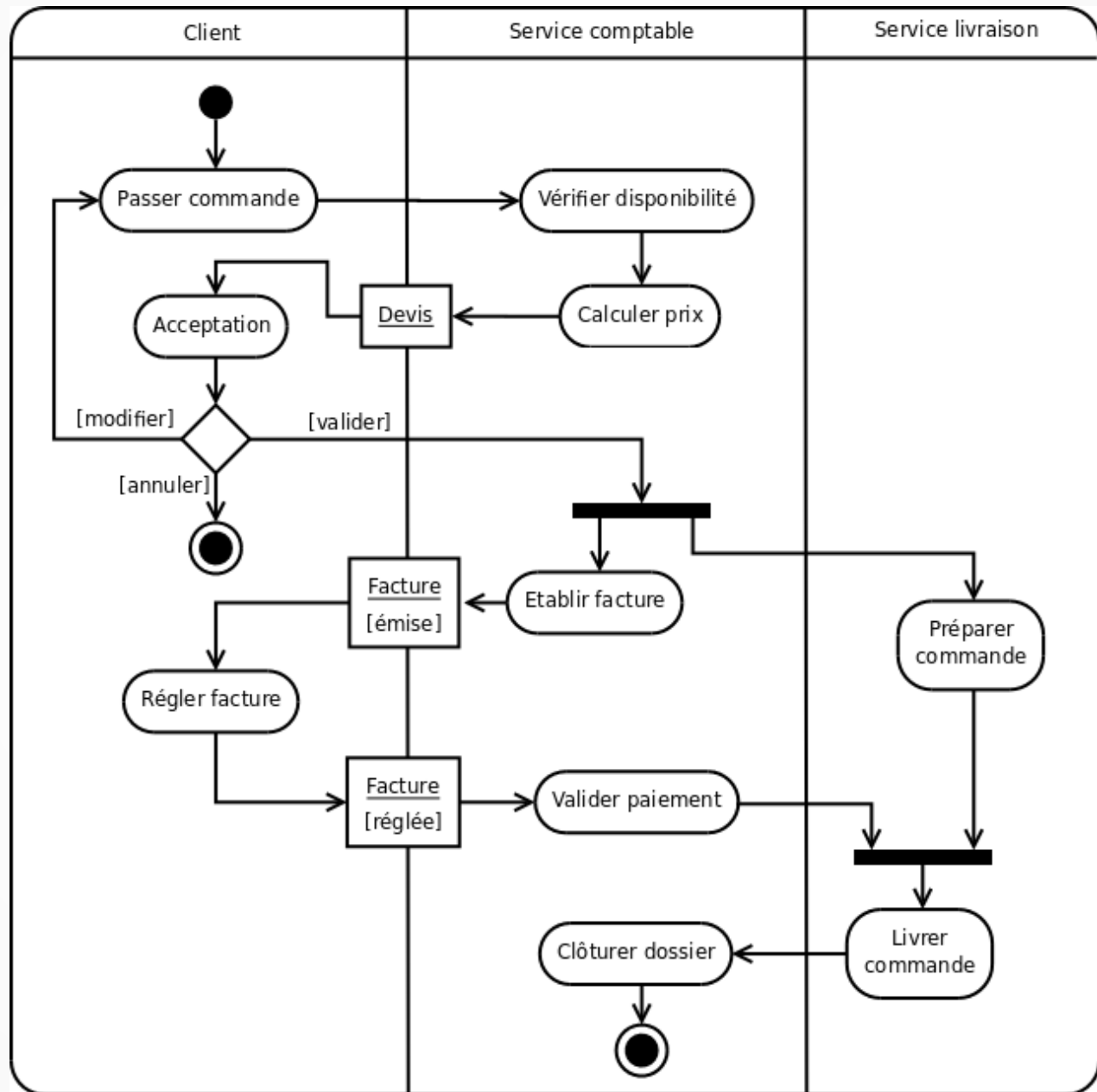


Notation textuelle



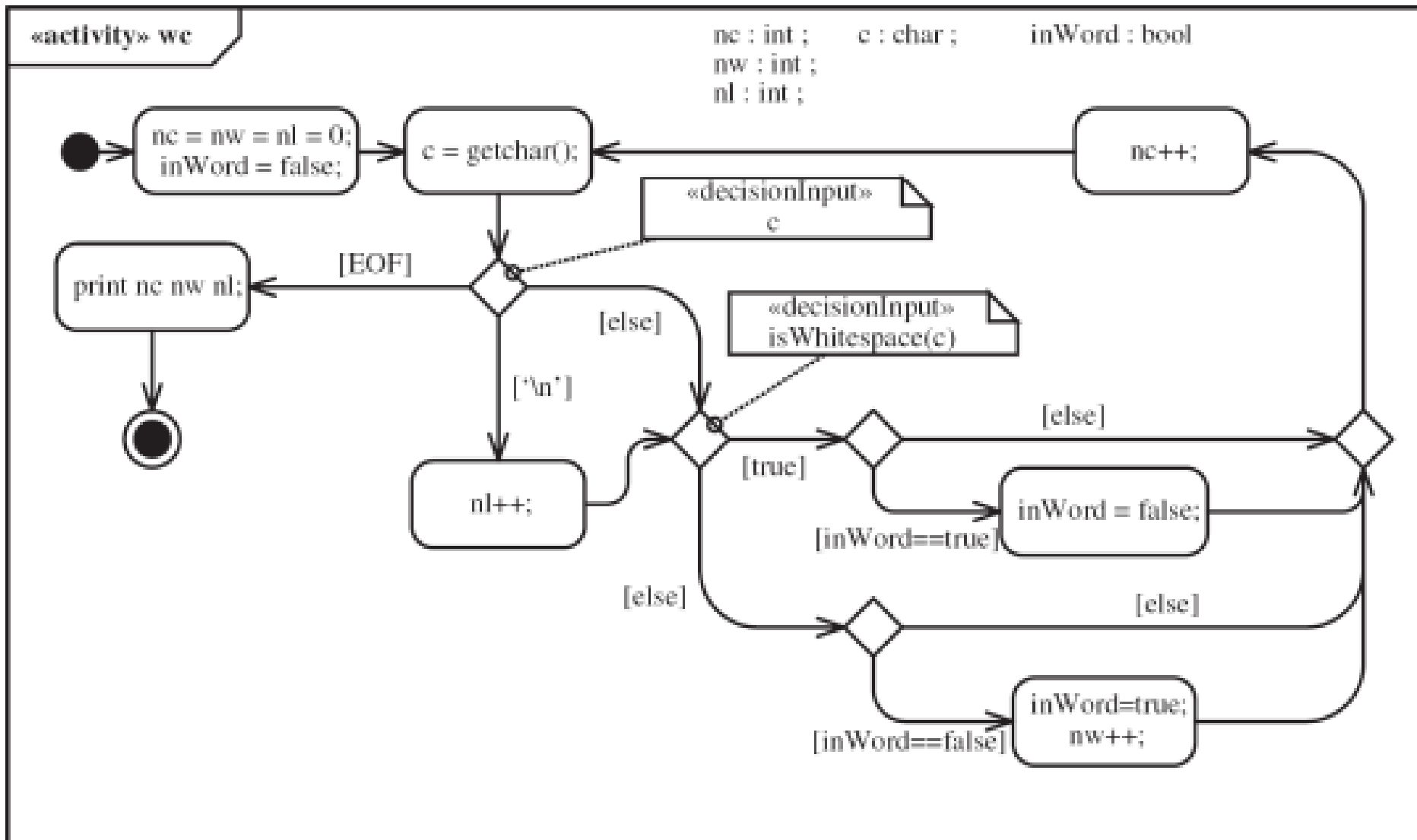
Partition : Traitement d'une commande

49



Exemple : DA pour une méthode : calcul de nc, nw, nl dans un fichier

50



Exemple : DA pour le cas d'utilisation





51

« Payer »

Cas d'utilisation	Payer
Acteur	Client
Événement déclencheur	Clic sur le bouton "payer"
Intérêt	Finaliser la commande en enregistrant le paiement.
Précondition	Avoir sélectionné au moins une vidéo
Scénario nominal	<ol style="list-style-type: none">1. Le système affiche le montant à payer.2. Le client est invité à saisir un code de réduction.3. Le système affiche le nouveau montant en tenant compte du code de réduction éventuel.4. Le client choisit son mode de paiement.5. Le système affiche les champs correspondants à remplir.6. Le client saisit les informations demandées et valide.7. Le système informe que le paiement est accepté.
Scénario alternatif	<ol style="list-style-type: none">2a. Le client ne saisit pas de code de réduction. Aller au point 4.3a. Le code de réduction n'est pas valide, le système en informe le client. Retourner au point 2.4a. Le client annule sa demande. Fin de traitement.6a. Le client annule la saisie. Retourner au point 4.7a. Le paiement est refusé, le système en informe le client. Retourner au point 4.

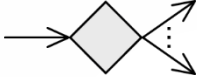
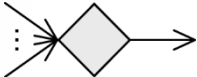
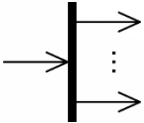
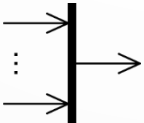
Notations

54

Nom	Notation	Description
Nœud d'action		Représente une action (atomique!)
Nœud d'activité		Représente une activité (peut être décomposée davantage)
Nœud Initial		le début d'exécution d'une activité
Nœud de fin d'activité		Fin de TOUS les chemins d'exécution d'une activité


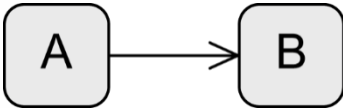
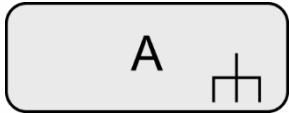
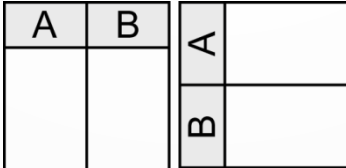
Notations

55

Nom	Notation	Description
Nœud de décision		Fractionnement d'un chemin d'exécution vers des voies d'exécution alternatives
Nœud de Fusion		Fusion d'exécution des chemins alternatifs en un seul chemin d'exécution
Nœud de bifurcation		Fractionnement d'un chemin d'exécution dans des chemins d'exécution concurrents
Nœud de Synchronisation		Fusion de l'exécution des chemins concurrents en un seul chemin d'exécution


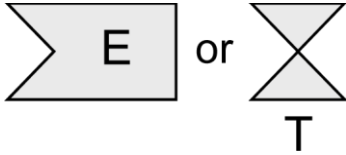
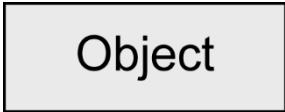


Notations

56

Nom	Notation	Description
Nœud de fin de flot		Fin d'exécution d'UN chemin d'une activité
Transition		liaison entre deux nœuds d'une activité
Action Appel comportementale		Action A fait référence à une activité du même nom
Couloir ou partition		Regroupement de nœuds et transitions dans une activité

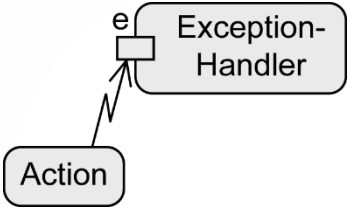
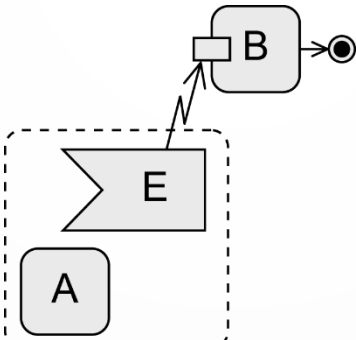
Notations

57

Nom	Notation	Description
Action send signal Envoyer un signal		Transmission d'un signal à un destinataire
Action accept event accepter événement (temporel) Asynchrone		Attendez pour un événement \mathbb{E} ou un événement temporel \mathbb{T}
Nœud Objet		Contient des données ou objets
Paramètre pour activités Paramètre pour Actions (épingles)	 	Contient des données et objets comme paramètres d'entrée et de sortie

Notations

58

Nom	Notation	Description
Gestionnaire d'exception	 <p>The diagram shows a rounded rectangle labeled 'Action' at the bottom. A jagged arrow points from the top of 'Action' to a small square on the left side of a larger rounded rectangle labeled 'Exception-Handler' at the top. The letter 'e' is placed above the arrowhead.</p>	Le gestionnaire d'exception est exécuté à la place de l'action lorsque l'événement e se produit
Région Interruptible	 <p>The diagram shows a dashed rectangular box containing two nodes: a rounded rectangle labeled 'A' at the bottom and a chevron-shaped node labeled 'E' at the top. A jagged arrow points from the top of 'E' to a small square on the left side of a rounded rectangle labeled 'B' at the top. Node 'B' has a solid arrow pointing to a final state symbol (a circle with a dot in the center).</p>	Flot continue sur un chemin différent si l'événement E est détecté