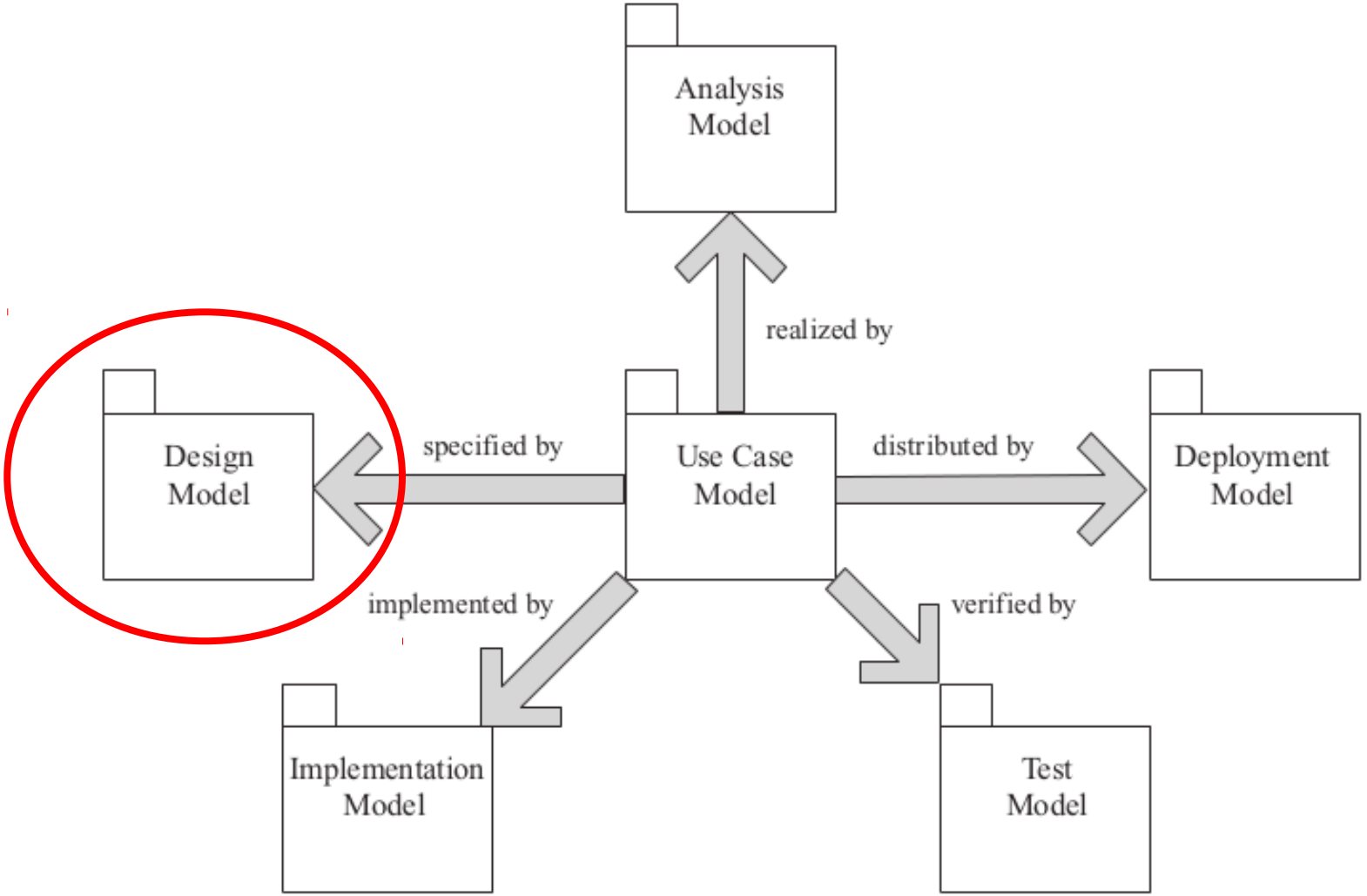
Les modèles dans UP



Modèle de conception

Dans UP, le modèle de conception affine la description du système :

- d'un point de vue structurel : on complète les diagrammes de classe et de paquetage.

- d'un point de vue fonctionnel : on détaille le fonctionnement du système avec des diagrammes d'interaction (séquences, communication) et des diagrammes d'états et d'activités.

Il est construit à partir des modèles de cas d'utilisation et d'analyse.

Le modèle de conception va compléter le modèle d'analyse en spécifiant les détails structurels et comportementaux.

Diagramme d'état

Un diagramme d'état permet de décrire sous forme d'un automate déterministe à états finis toute la vie d'un objet (pas seulement un scénario partiel).

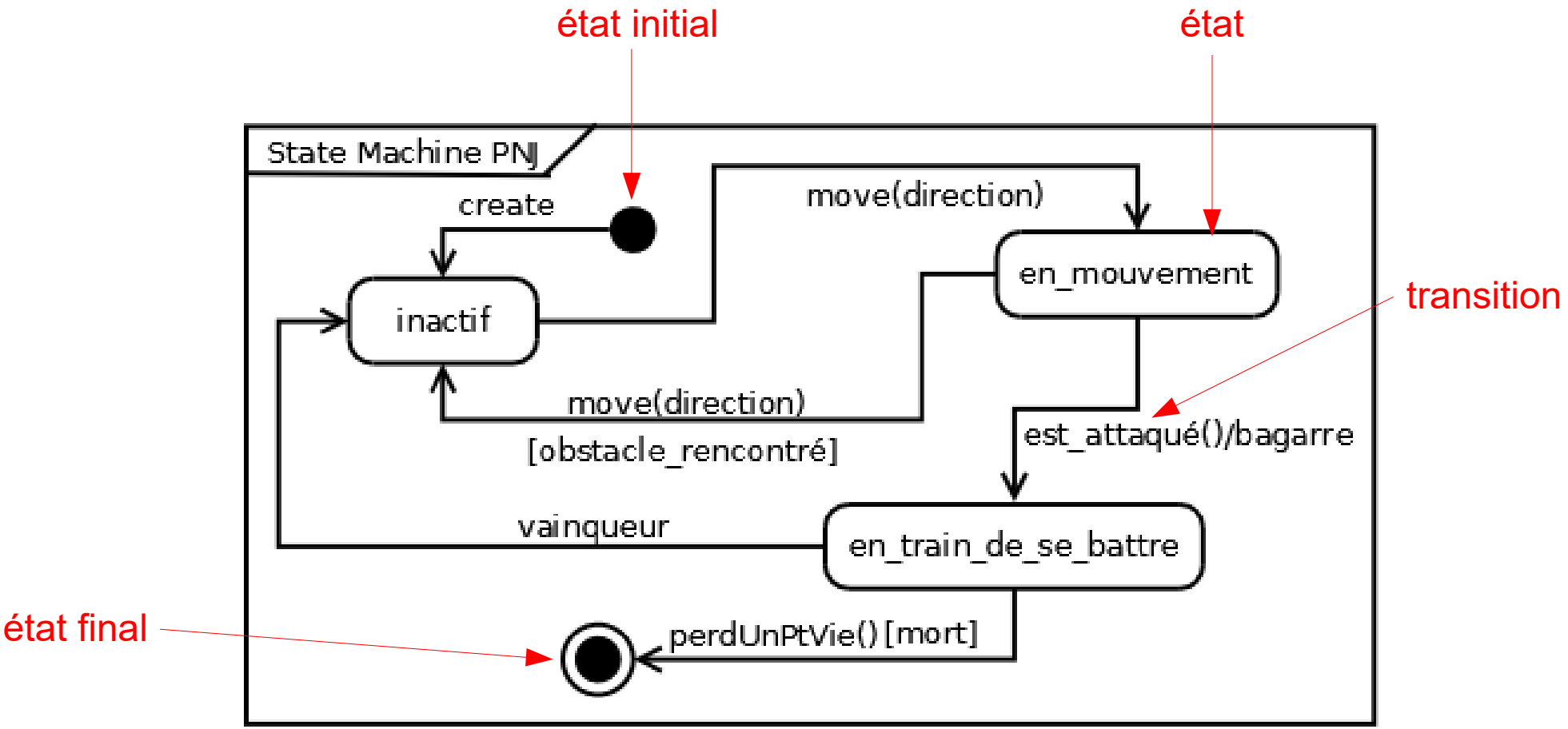


Diagramme d'état : transition

Syntaxe d'une transition : [trigger ["," trigger]] ["[" garde "]"] ["/" action]

Le trigger déclenche le franchissement de la transition si la garde est vérifiée et l'action est déclenchée.

Le trigger peut être un appel de méthode (synchrone ou non) ou un événement (temps écoulé, valeurs d'attributs atteintes, …).

Exemples :

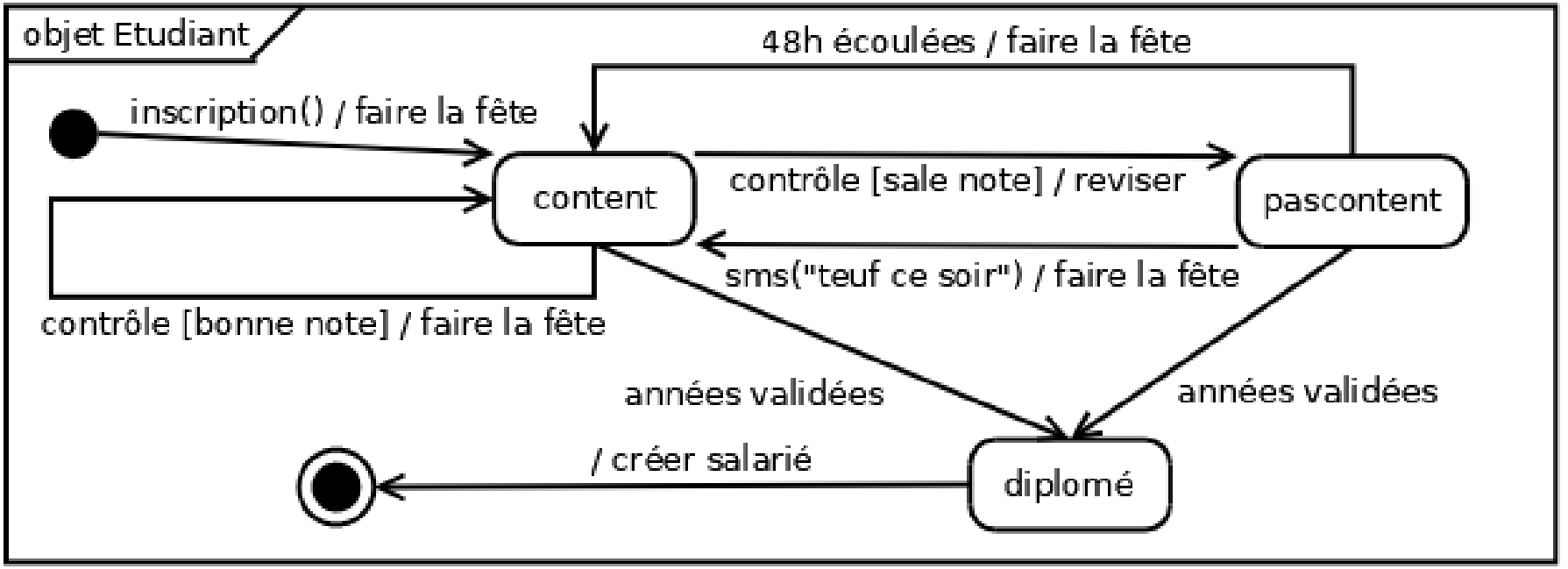


Diagramme d'état : choix

Il est possible de regrouper les transitions déclenchées par un même événement :

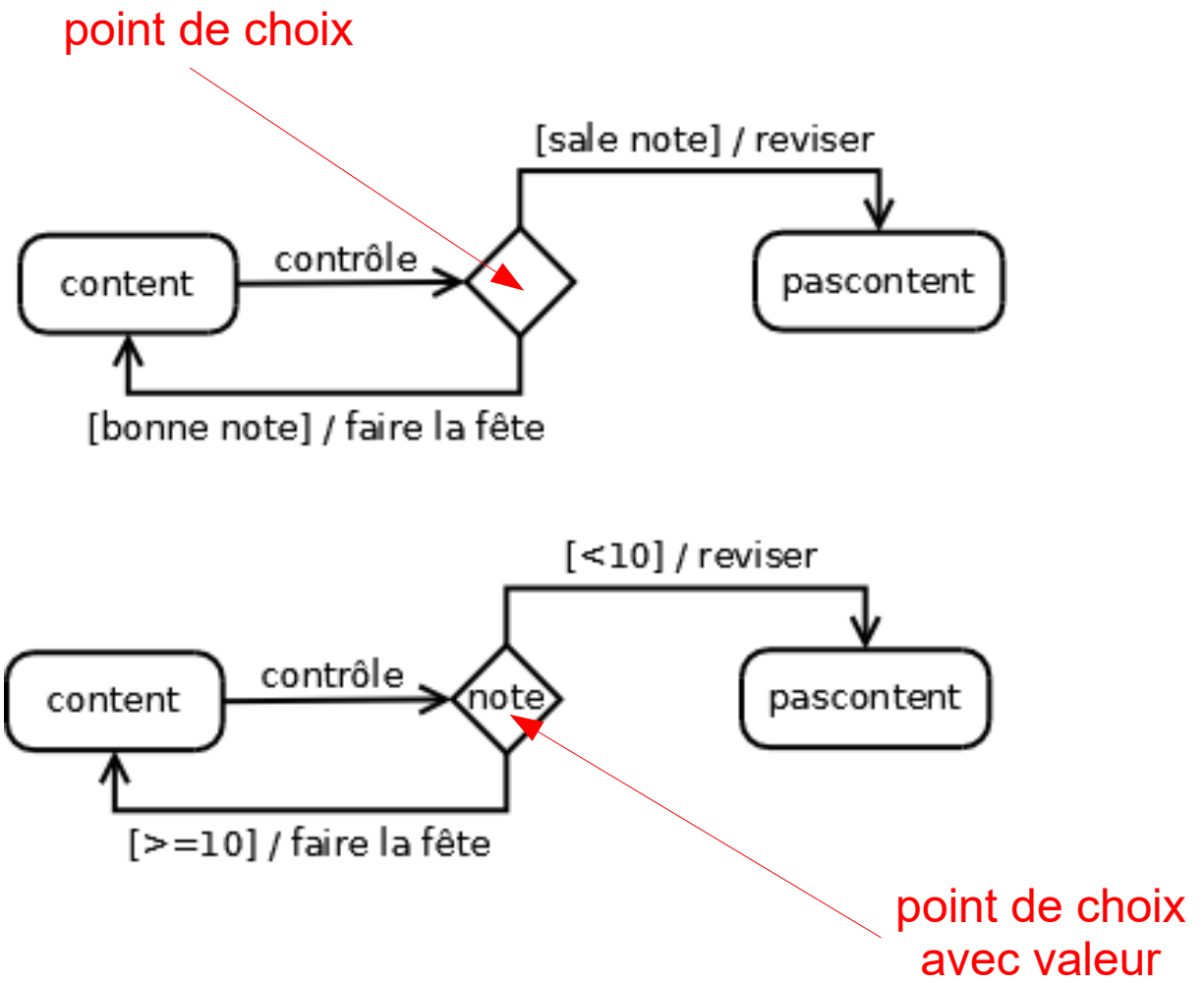


Diagramme d'état : activités (1/2)

Il est possible de spécifier des activités internes à l'état.

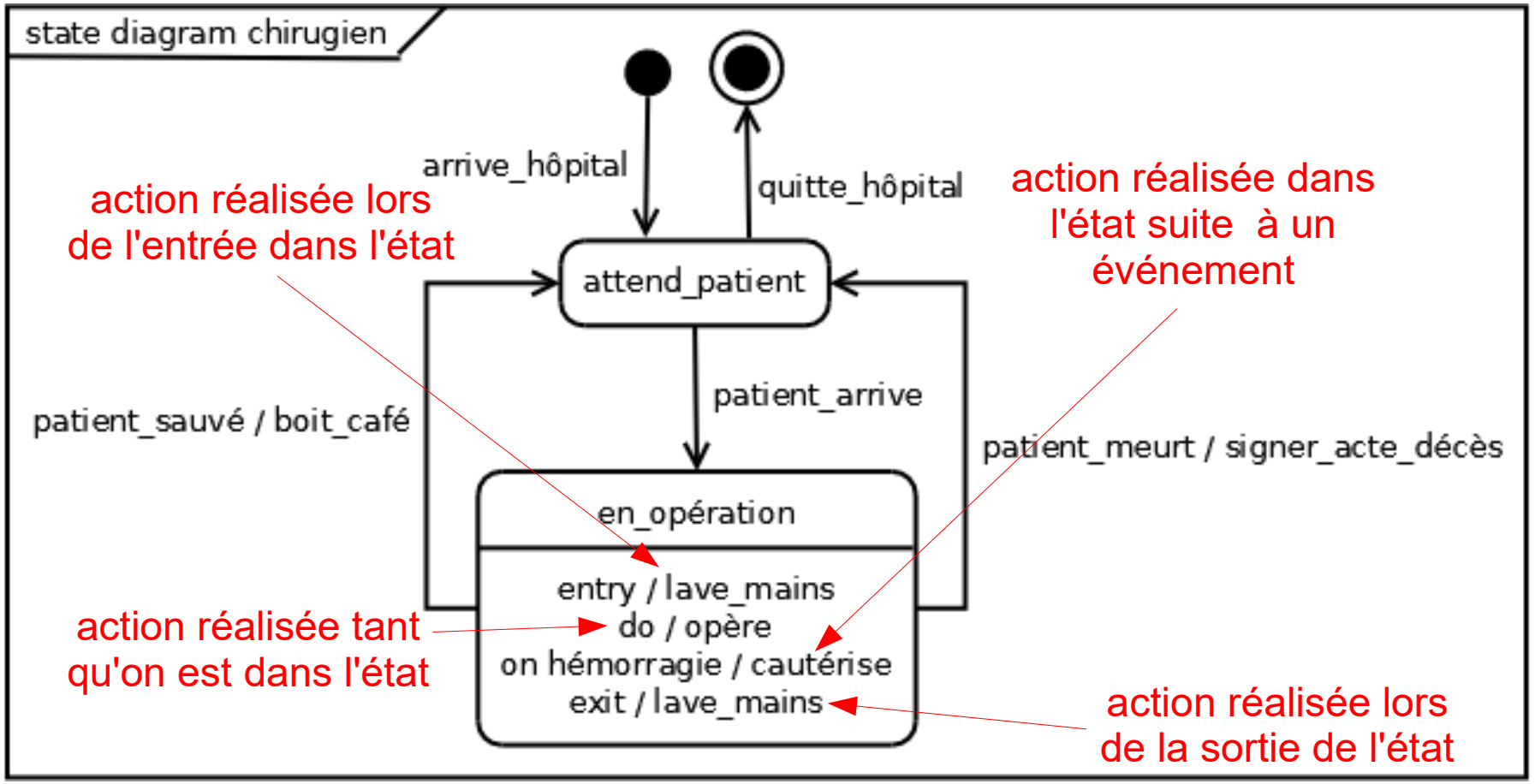


Diagramme d'état : activités (2/2)

Une activité interne à un état peut être :

- réalisée systématiquement à l'entrée dans l'état : *entry / <activité>*
- réalisée systématiquement à la sortie de l'état : *exit / <activité>*
- réalisée tout au long de l'état : *do / <activité>*
- réalisée sur événement : *on* <événement> */ <activité>*

Une activité est décrite comme une transition avec trigger, garde et action. Contrairement à une action sur transition, une activité s'interrompt si une transition sortante est franchie.

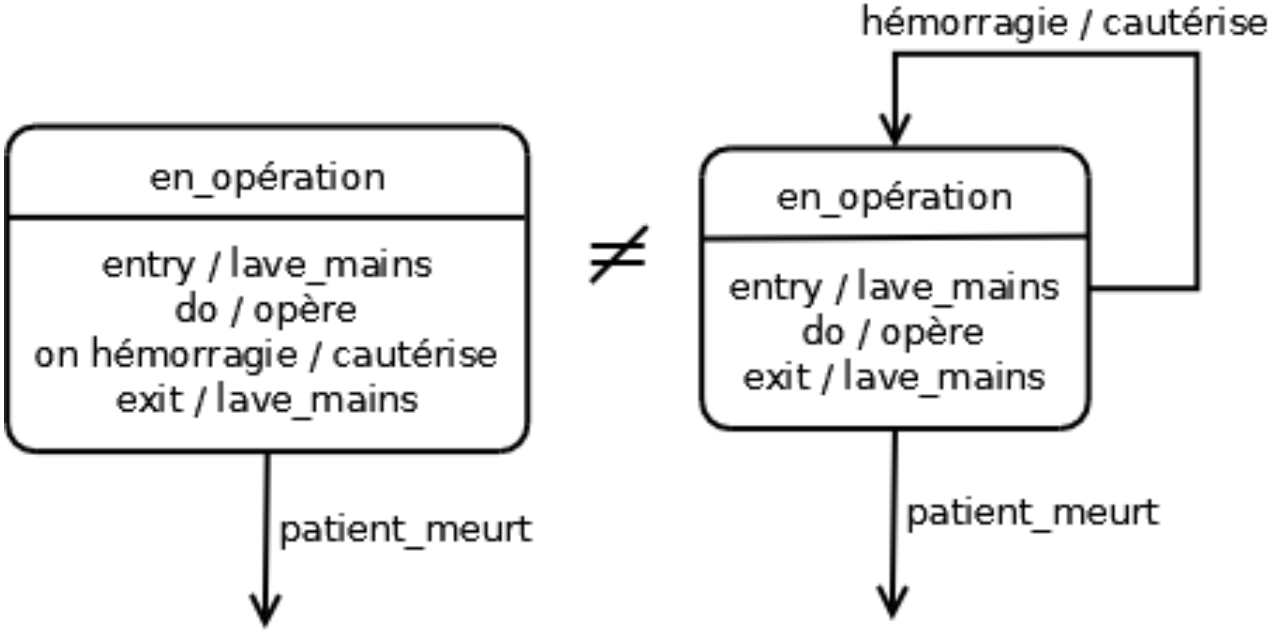


Diagramme d'état : états composites (1/5)

Il est possible d'imbriquer un diagramme d'états dans un état composite pour en détailler l'évolution interne.

l'entrée dans l'état composite fait passer dans l'état initial de l'état interne

la sortie de l'état composite fait quitter l'état interne dans lequel se trouve l'objet à ce moment

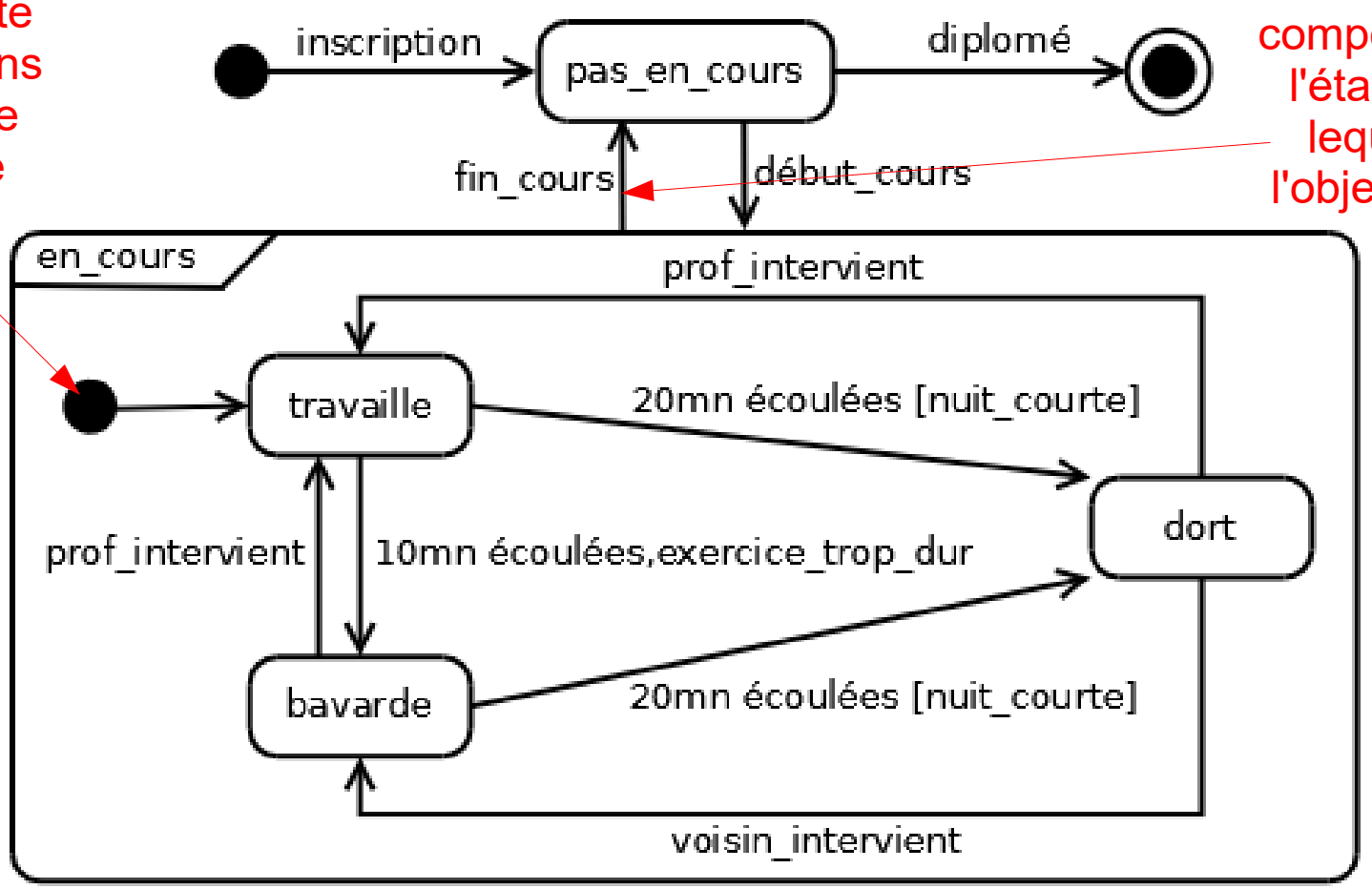


Diagramme d'état : états composites (2/5)

Il peut y avoir des transitions entre états internes et externes d'un état composite.

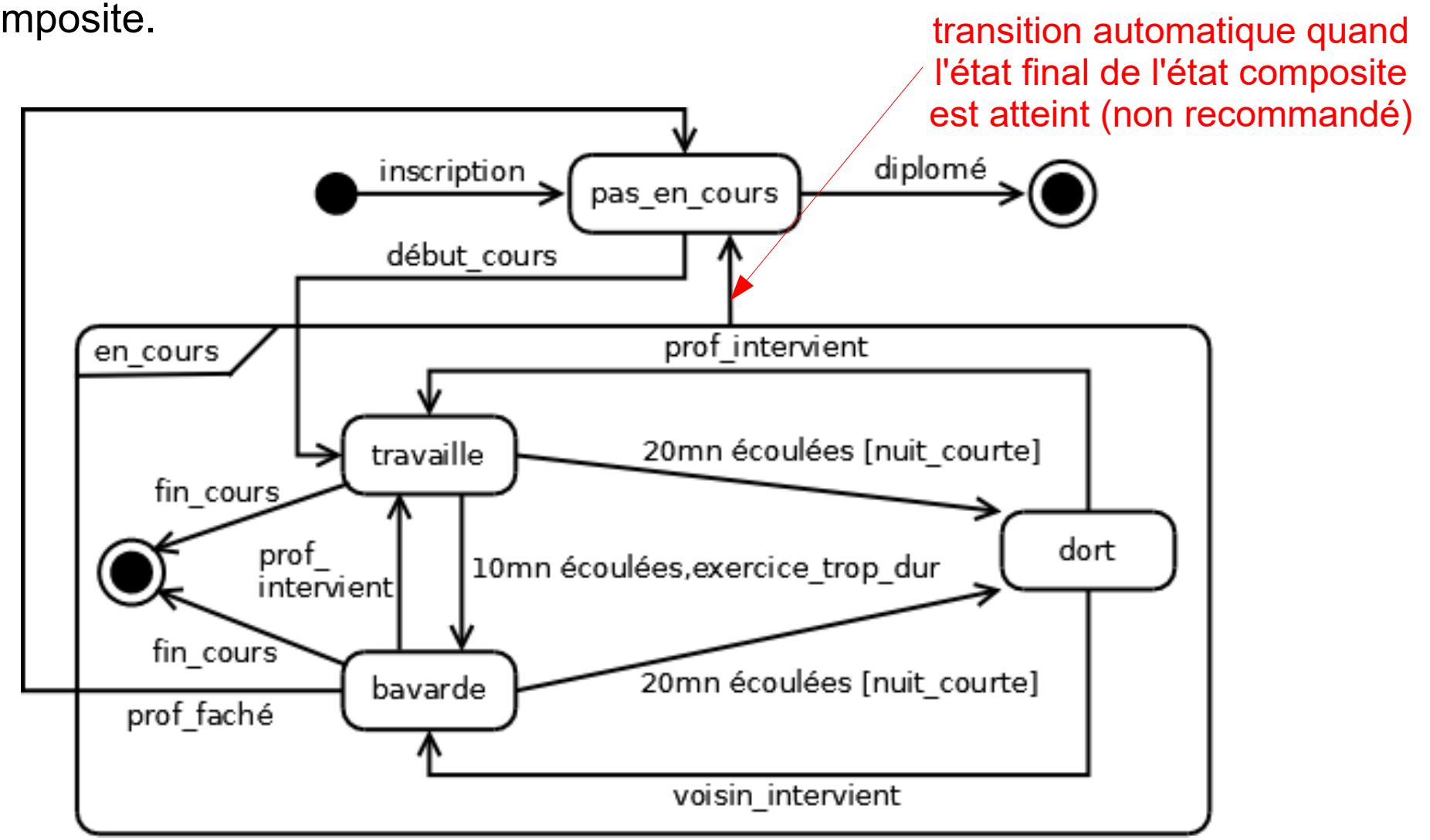


Diagramme d'état : états composites (3/5)

On peut faire apparaître explicitement les points d'entrée et de sortie des états composites.

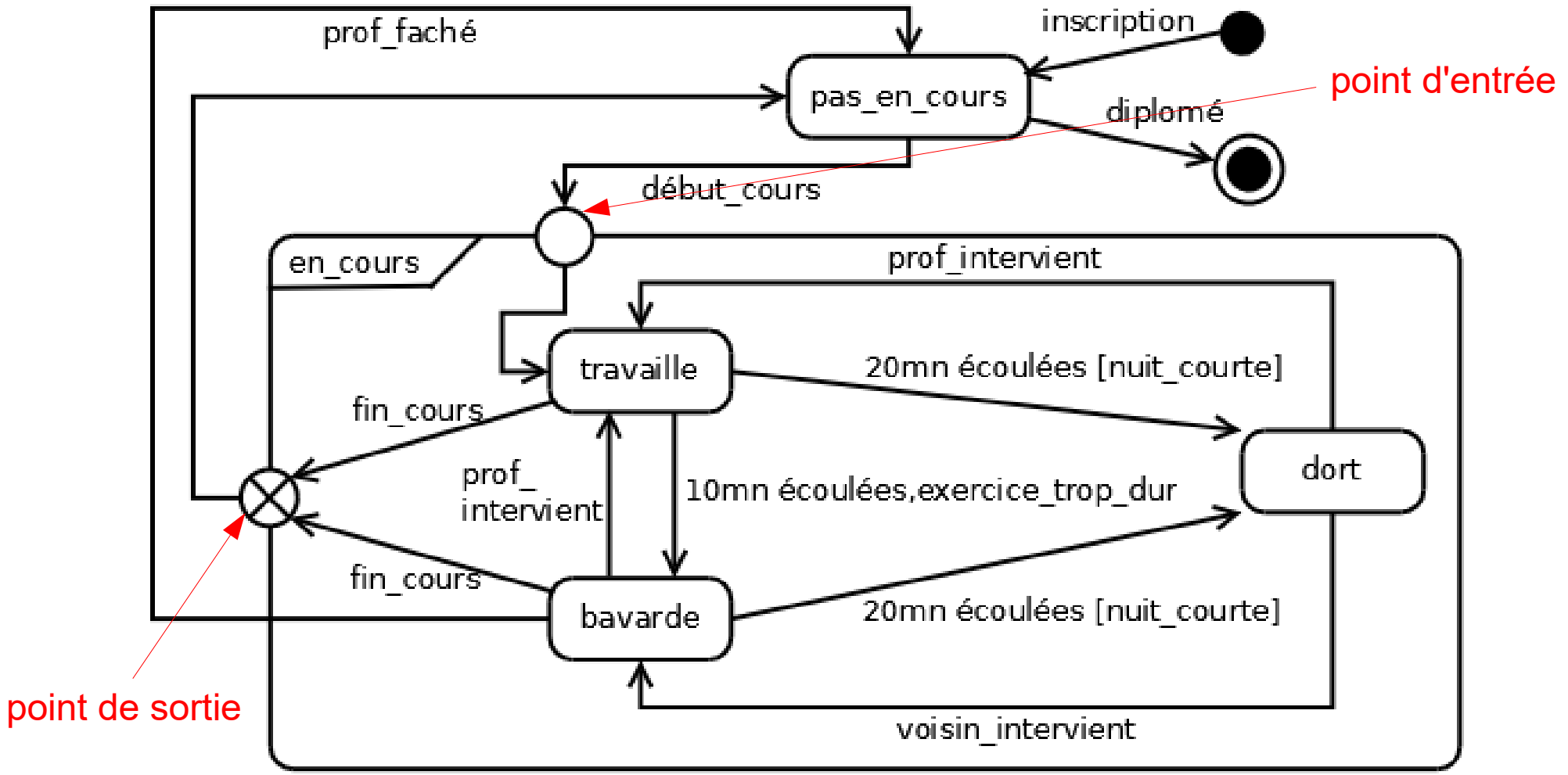


Diagramme d'état : états composites (4/5)

L'imbrication des états composites peut se faire sur plus de deux niveaux.

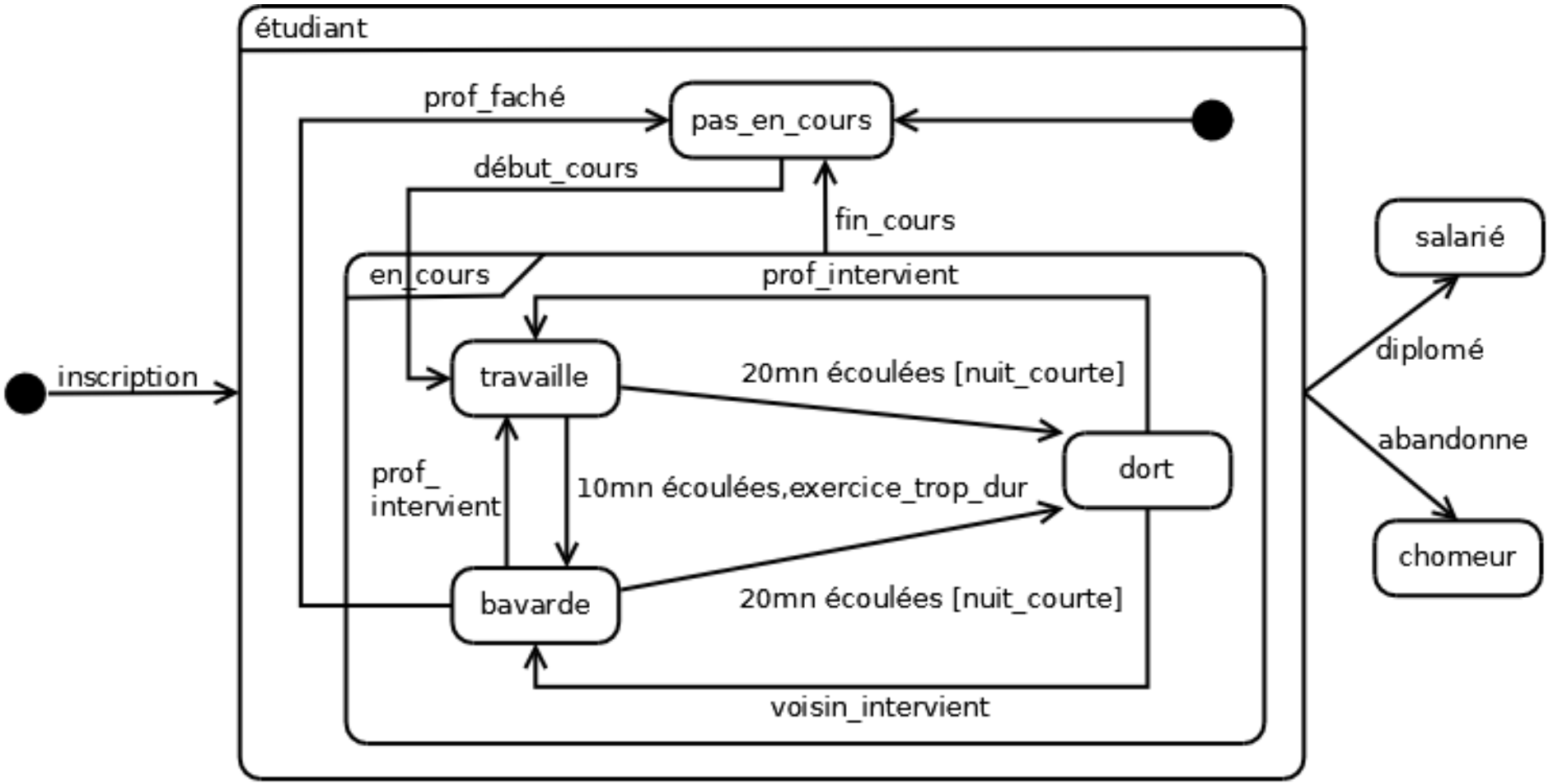


Diagramme d'état : états composites (5/5)

Il est possible d'indiquer qu'un état est composite, sans détailler la composition (ou en la détaillant dans un autre diagramme).

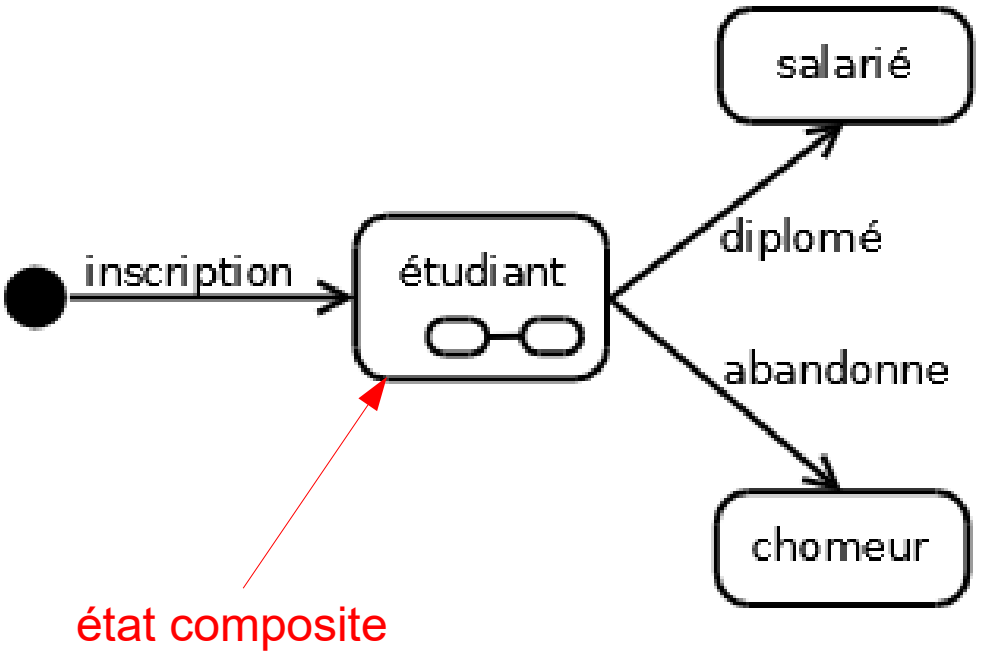


Diagramme d'état : états concurrents (1/4)

Des états internes peuvent être simultanés, l'état composite est alors décrit par des sous-automates concurrents.

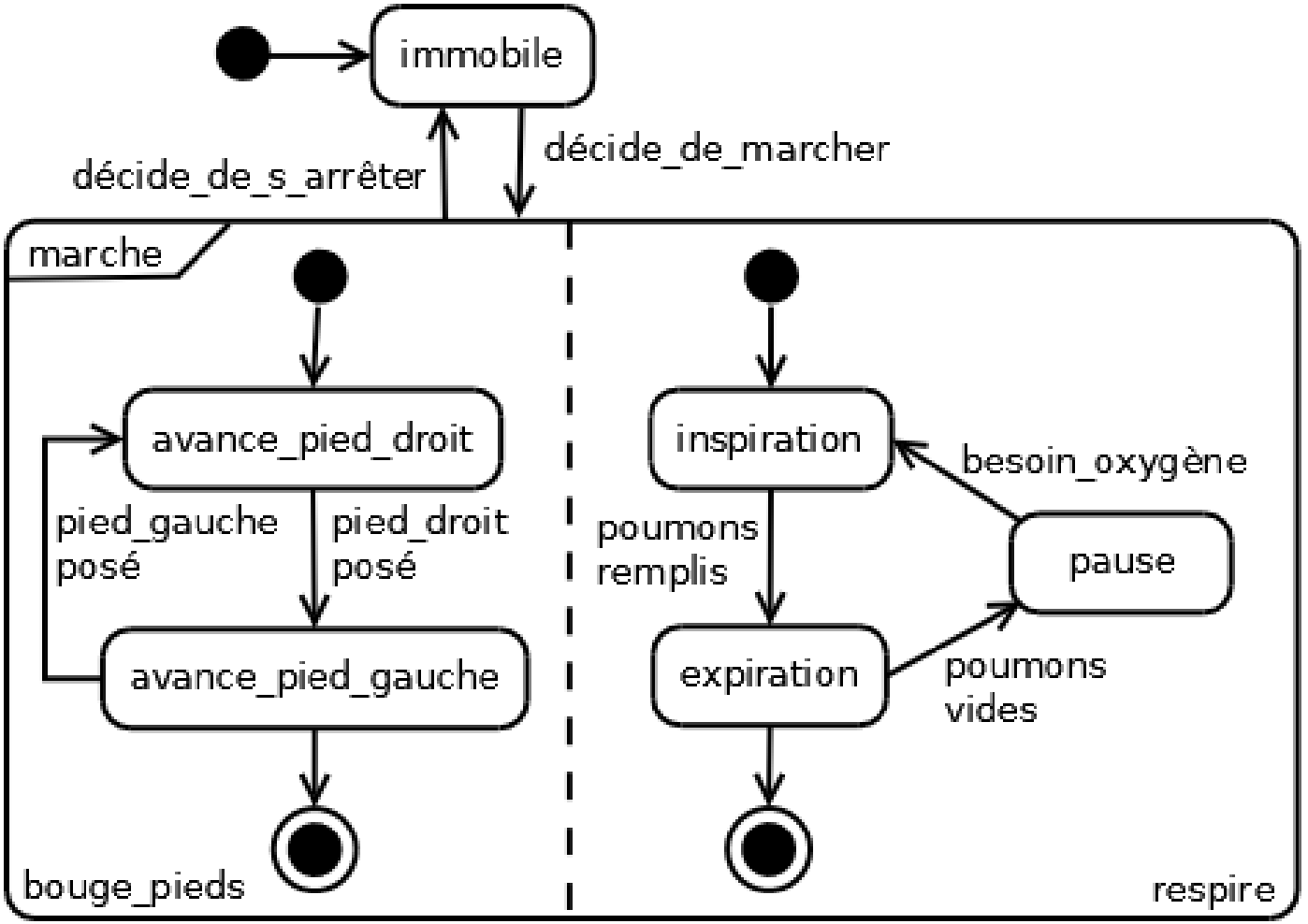


Diagramme d'état : états concurrents (2/4)

Des transitions peuvent lier un état interne concurrent à un état externe.

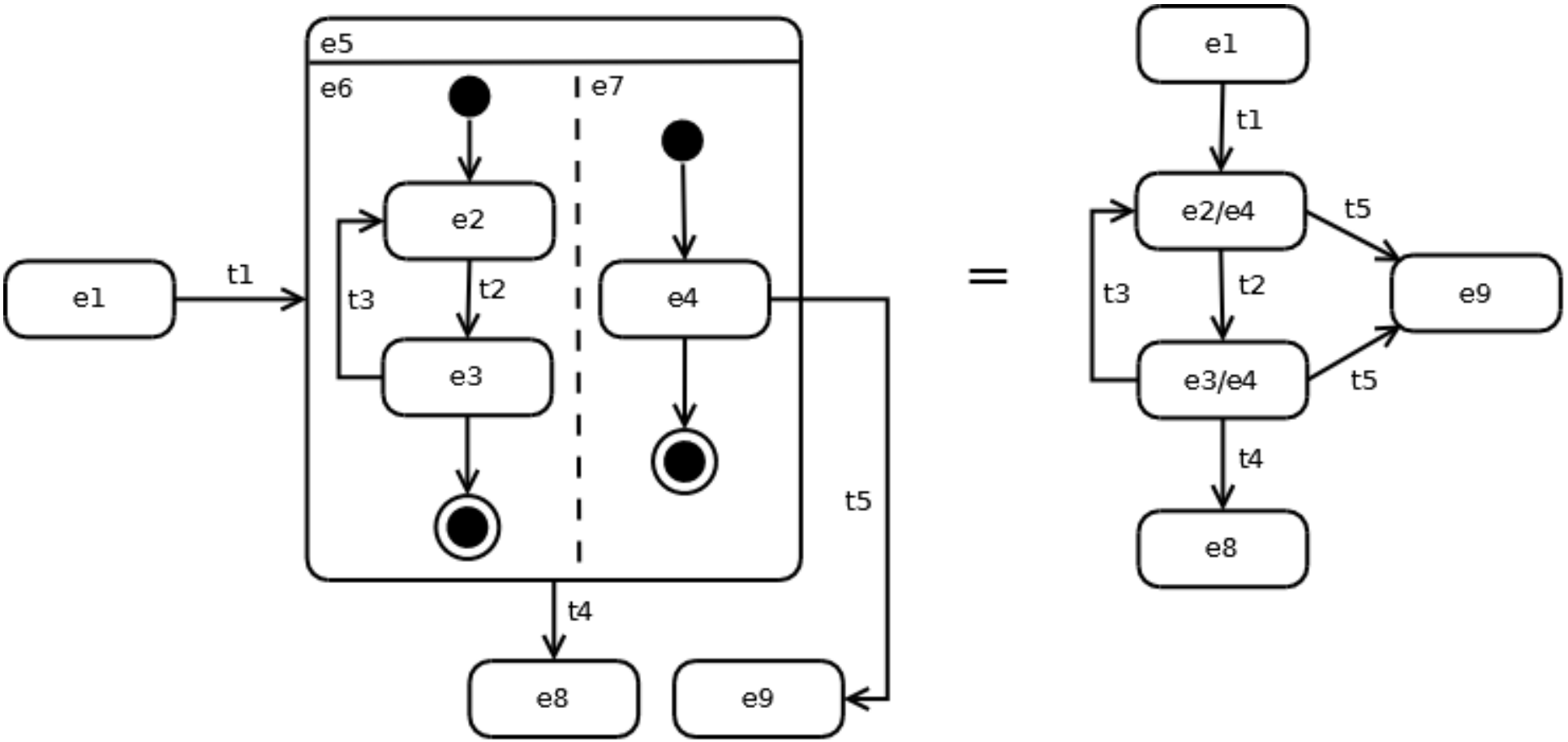


Diagramme d'état : états concurrents (3/4)

Si la transition de sortie est automatique, elle est franchie quand tous les états finaux internes sont atteints.

transition automatique quand les états finaux de l'état composite sont atteints

PROBLEME : une page peut ne pas être imprimée !

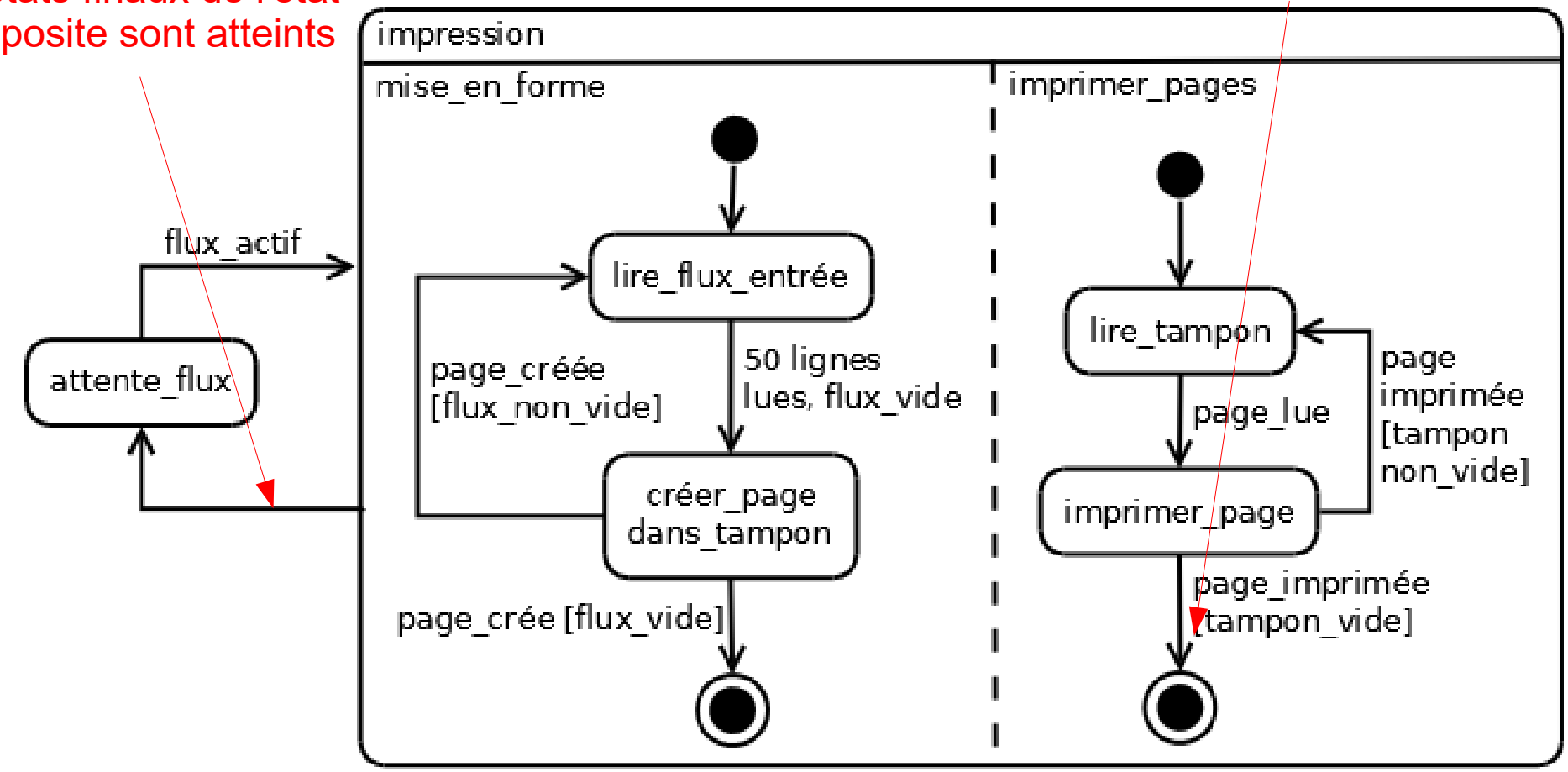


Diagramme d'état : états concurrents (4/4)

Synchronisation des entrées et sorties d'états concurrents : les fork et join ne sont franchis que lorsque toutes les transitions entrantes ont été franchies.

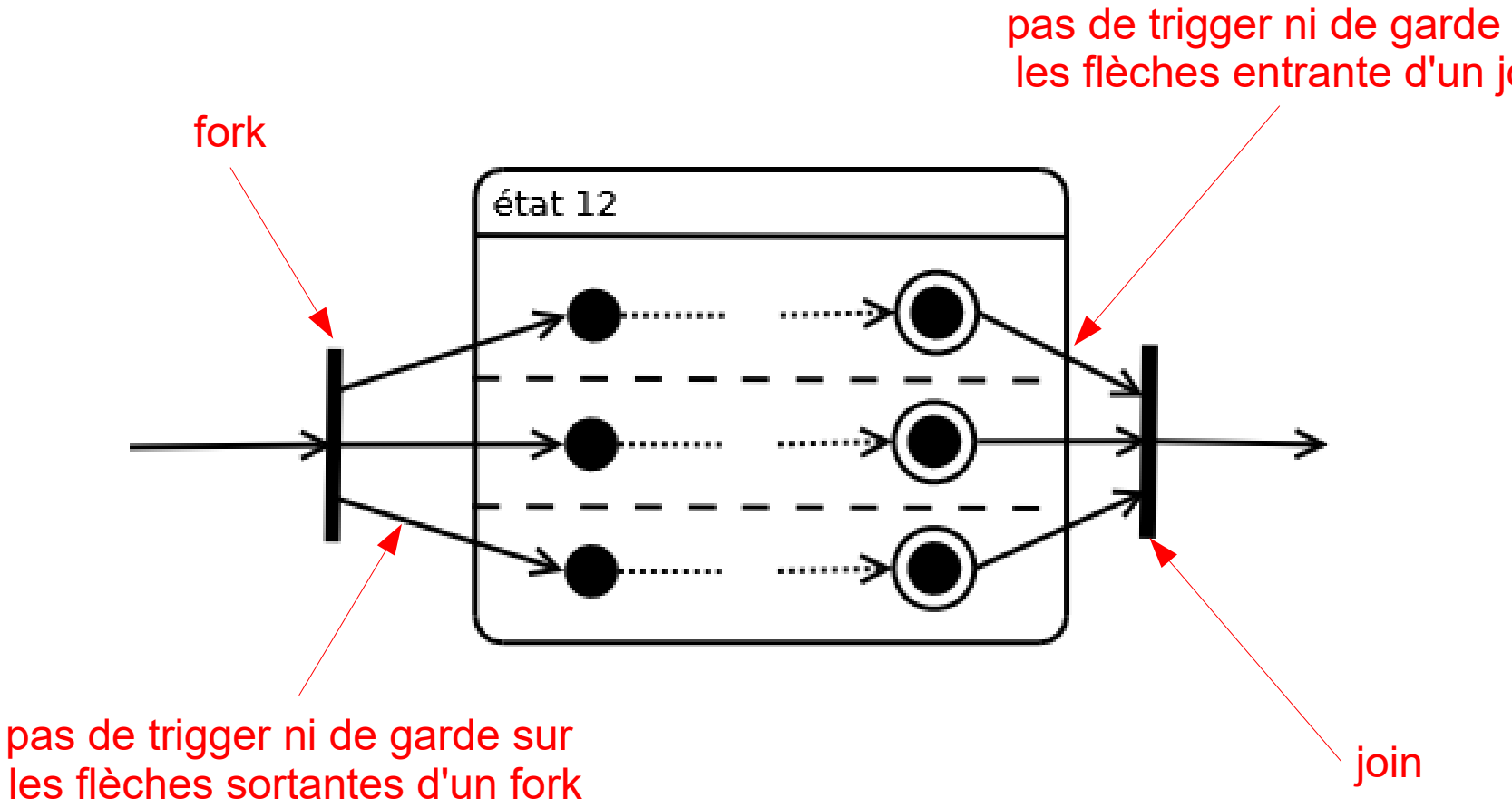
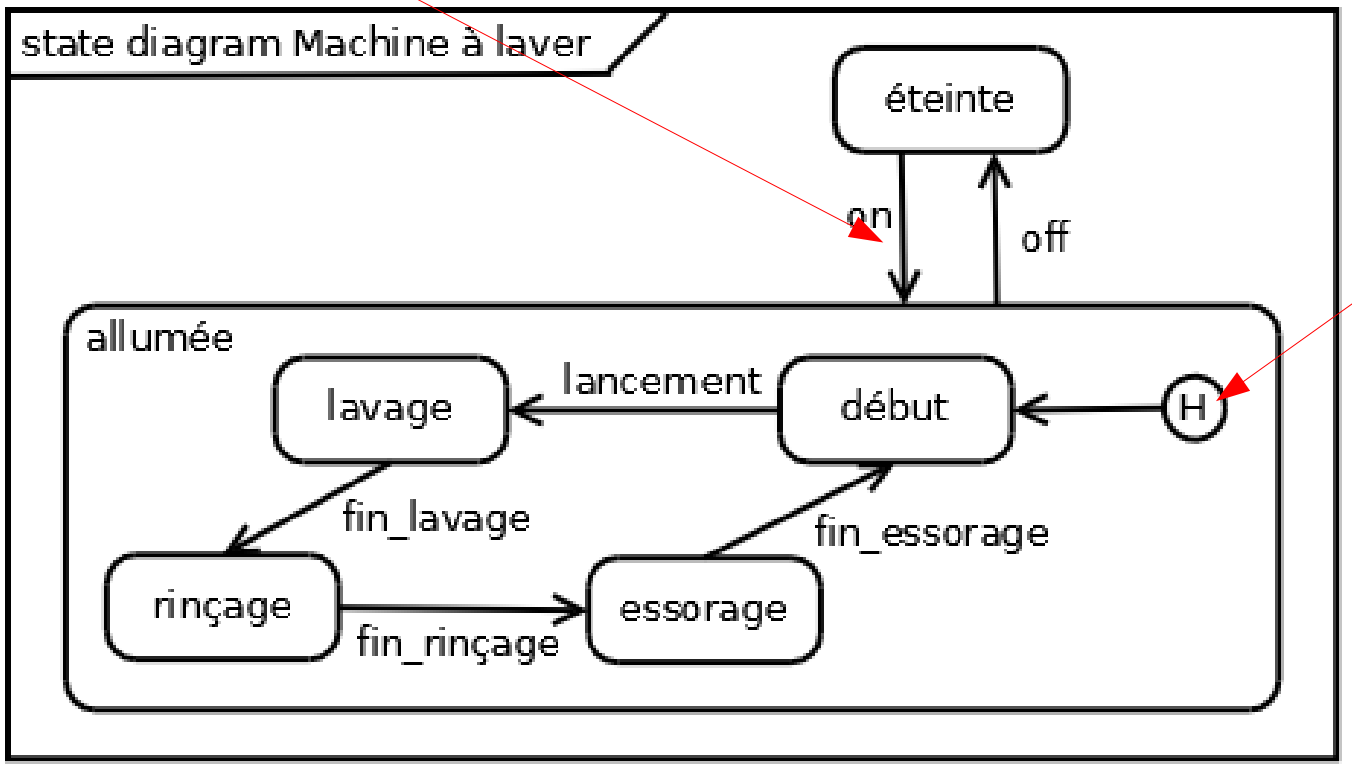


Diagramme d'état : états avec historique

Un état composite avec historique conserve la mémoire du dernier état interne où il s'est trouvé.

la première entrée dans l'état se fait sur l'état initial,
les suivantes sur le dernier état mémorisé

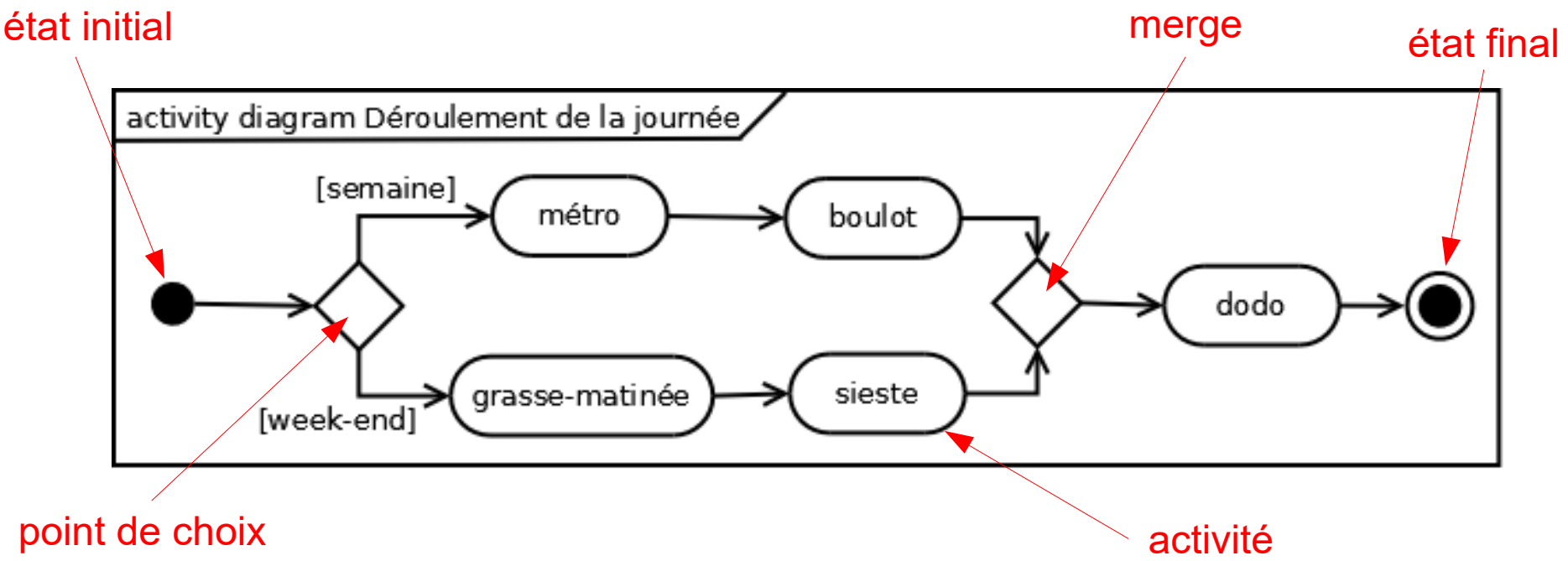


état initial marqué d'un H

Si l'état initial est marqué d'un H*, il y a sauvegarde de toute la configuration interne de l'état composite (par exemple dans le cas d'un état concurrrent).

Diagramme d'activités

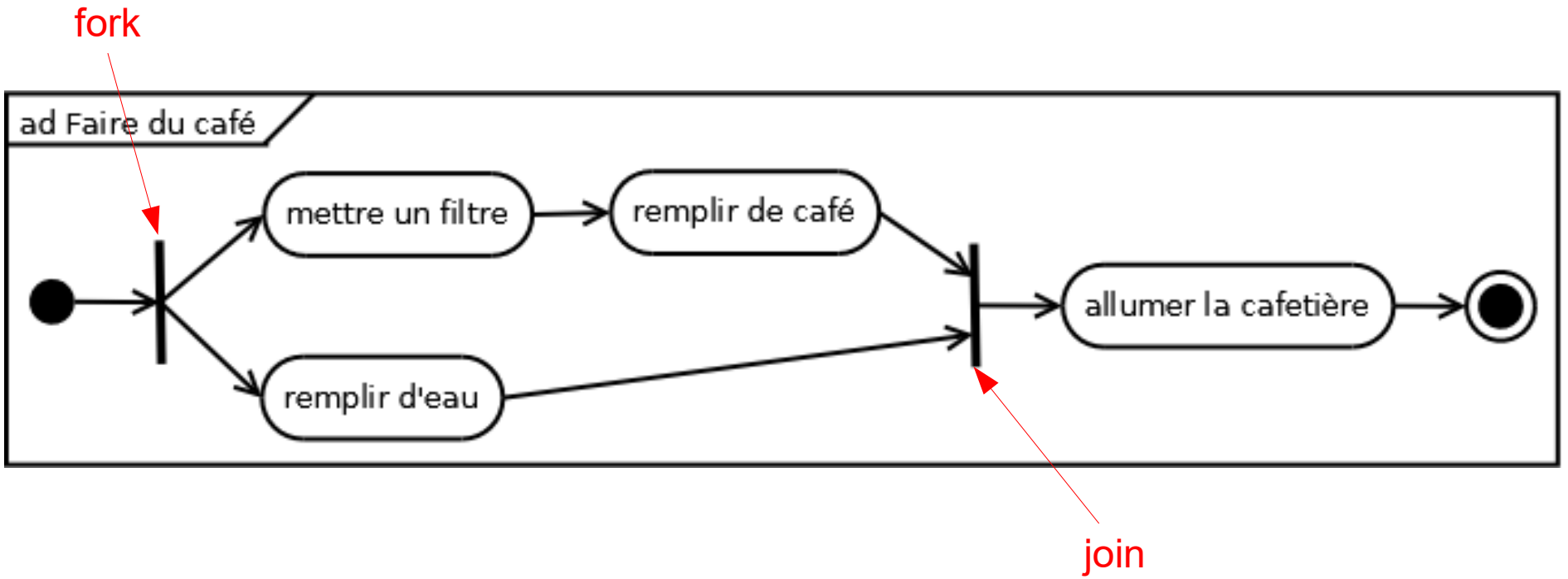
Les diagrammes d'activités permettent de représenter le fonctionnement complet du système (ou d'une partie du système), décomposé en activités (actions).



Les diagrammes d'activités permettent aussi de décrire les cas d'utilisation avant que les classes et la structure du programme soit fixée.

Diagramme d'activités : synchronisation

Synchronisation des flux : les fork et join ne sont franchis que lorsque toutes les activités entrantes sont terminées.



Il est possible qu'une activité n'aboutisse pas sans arrêter pour autant le workflow.

Diagramme d'activités : fin de flux

Il est possible d'indiquer la fin d'un flux d'activités, qui n'est pas bloquante pour les autres branches de flux.

C'est utile pour la synchronisation de flux parallèles.

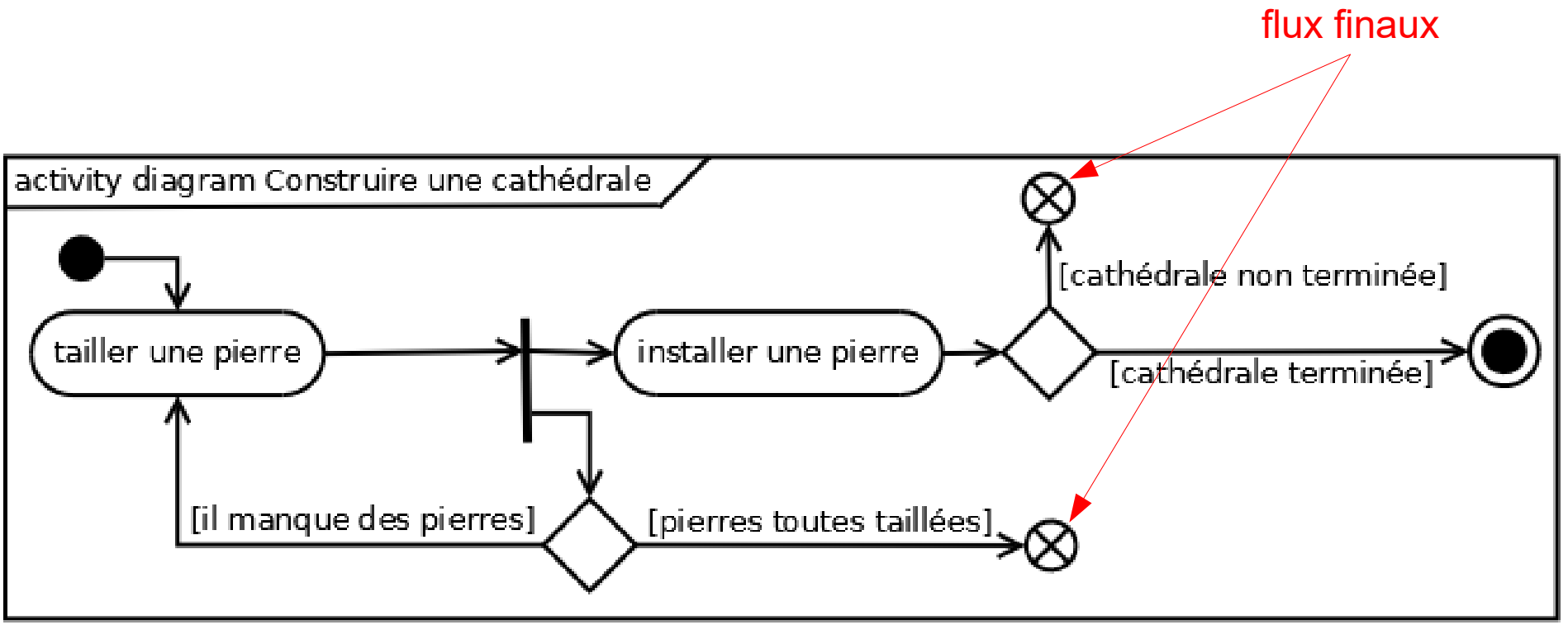


Diagramme d'activités : acteurs et activités

Il peut être utile de découper un diagramme d'activités selon les acteurs responsables de chaque activité, ou selon les classes impliquées.

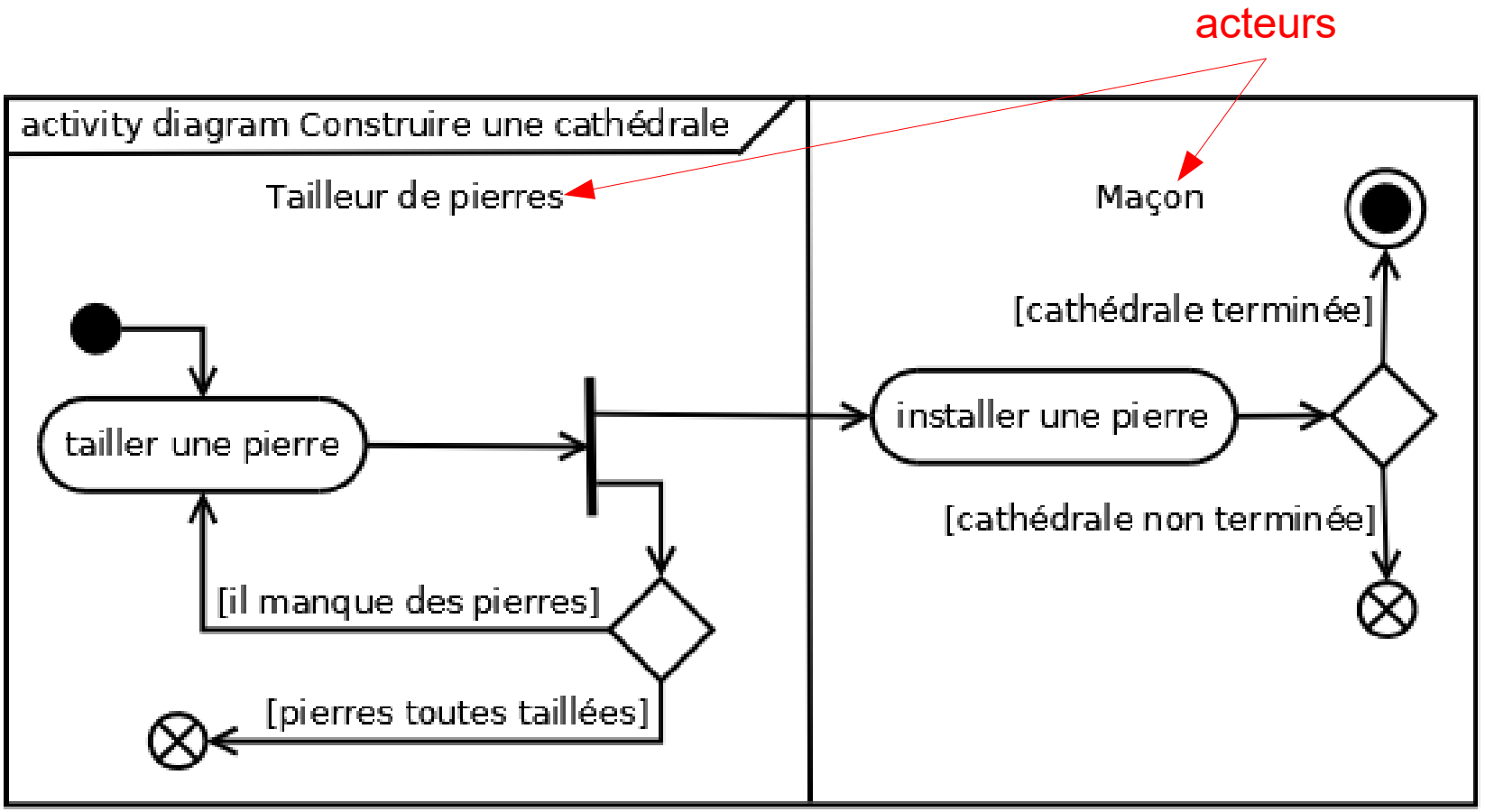


Diagramme d'activités : connecteur

Il est possible de donner des noms aux flux pour faciliter la lecture des diagrammes.

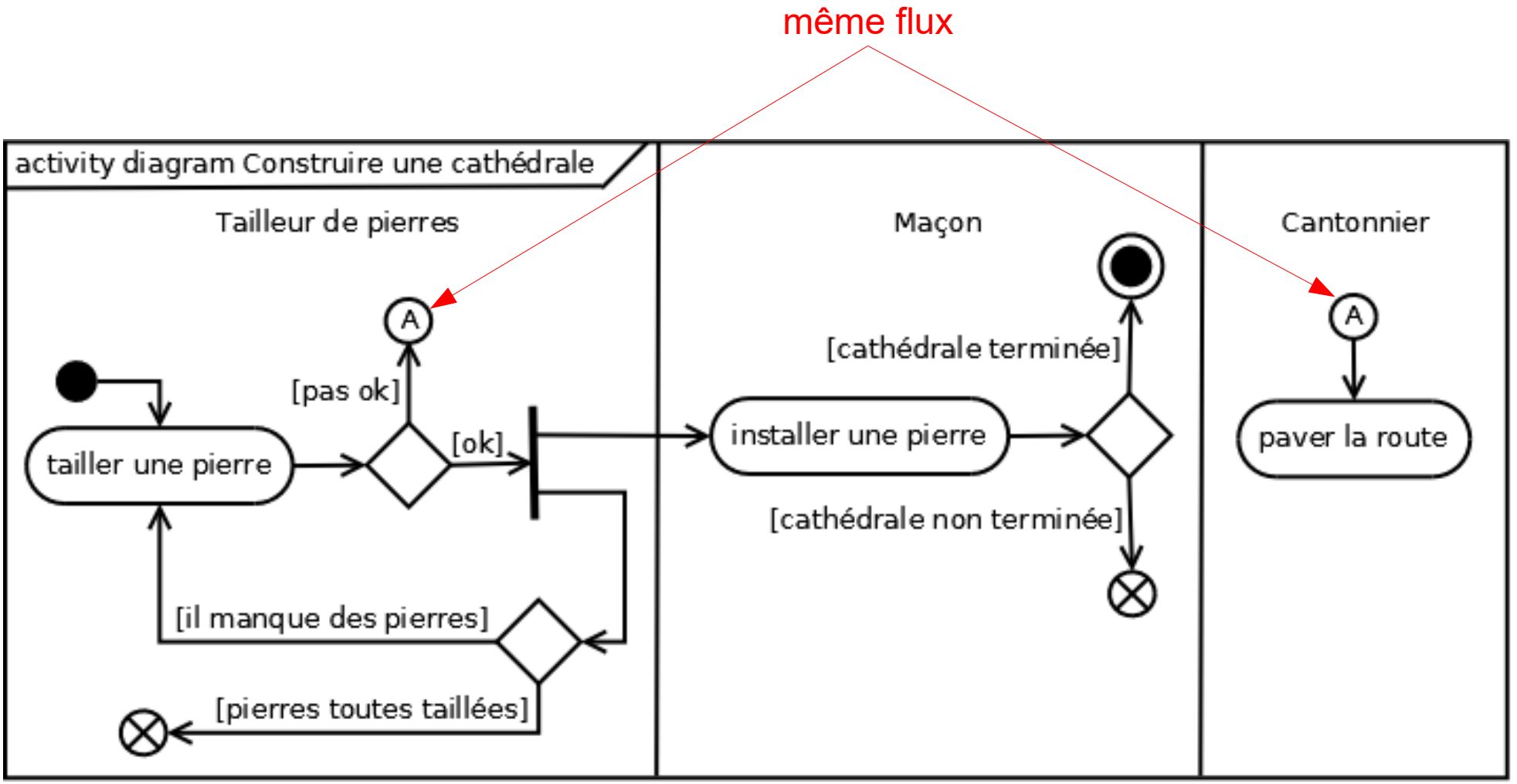


Diagramme d'activités : objets

Des objets peuvent apparaître dans les diagrammes d'activités.

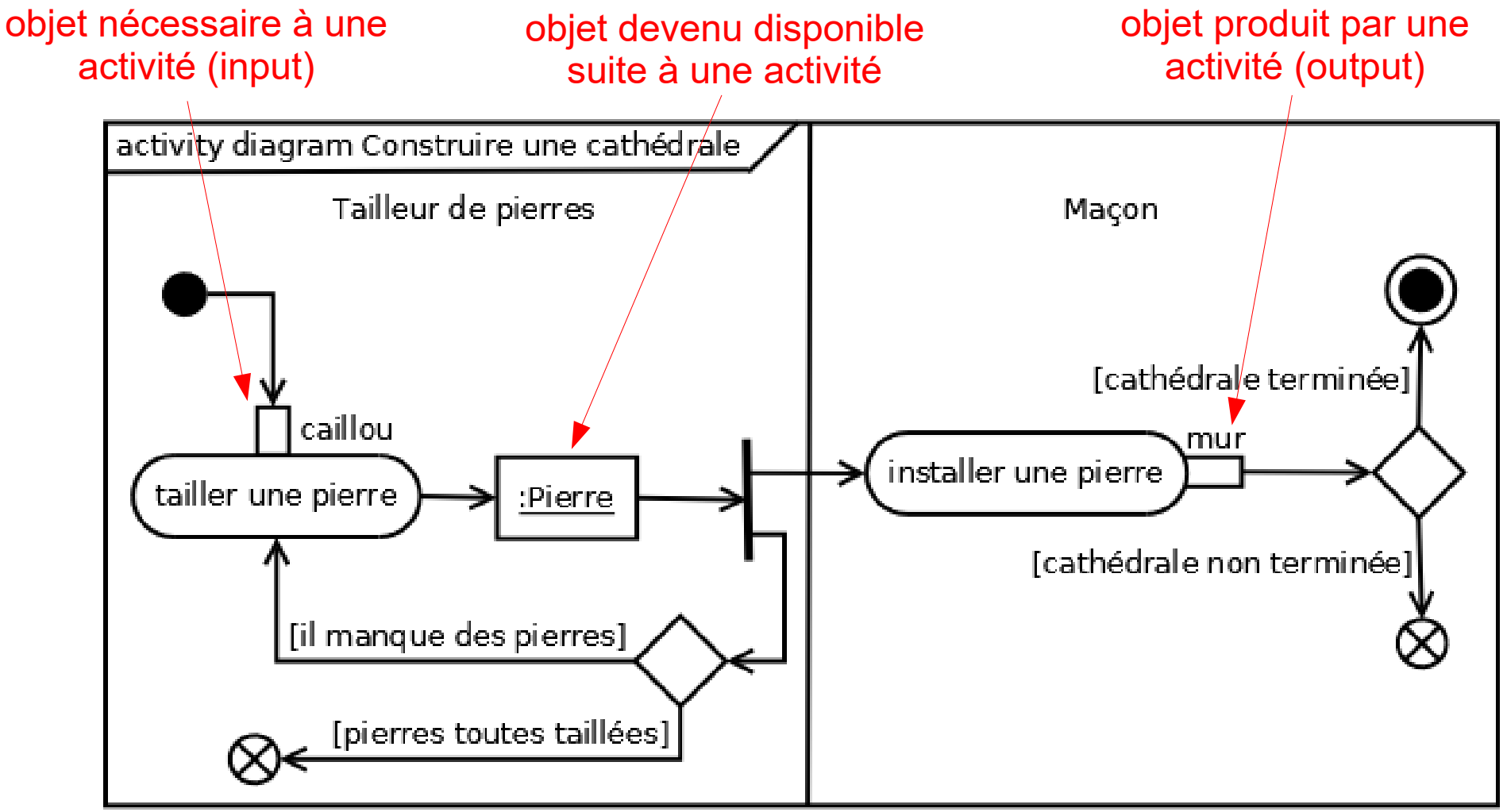


Diagramme d'activités : paramètres d'activités

Les paramètres d'une activité peuvent être spécifiés.

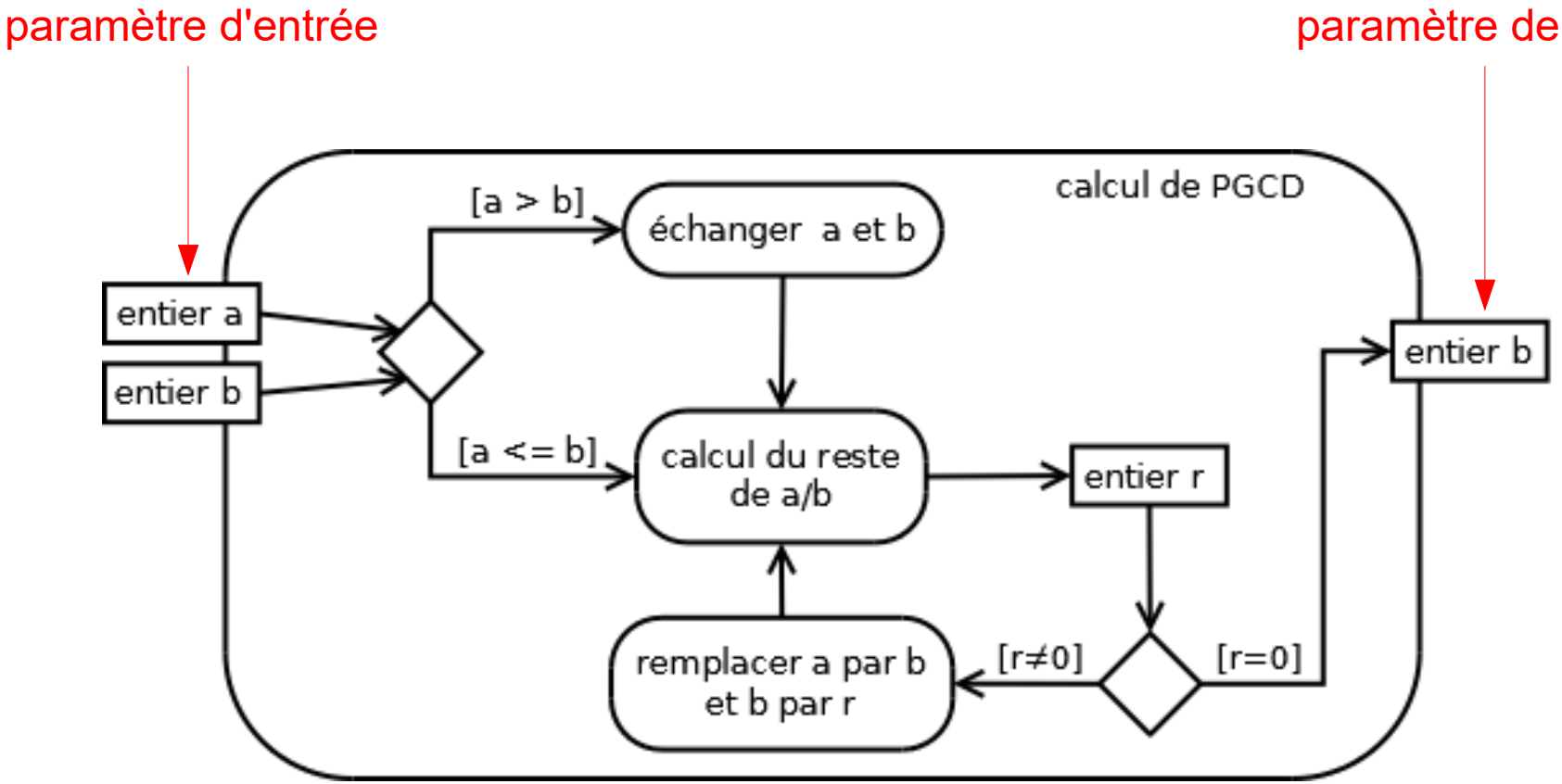


Diagramme d'activités : expansion

Une région d'expansion permet de faire répéter des activités sur des collections d'éléments. Les modes de répétition sont *parallel*, *iterative* ou *stream* (les objets en entrée et/ou sortie sont modifiés lors des activités).

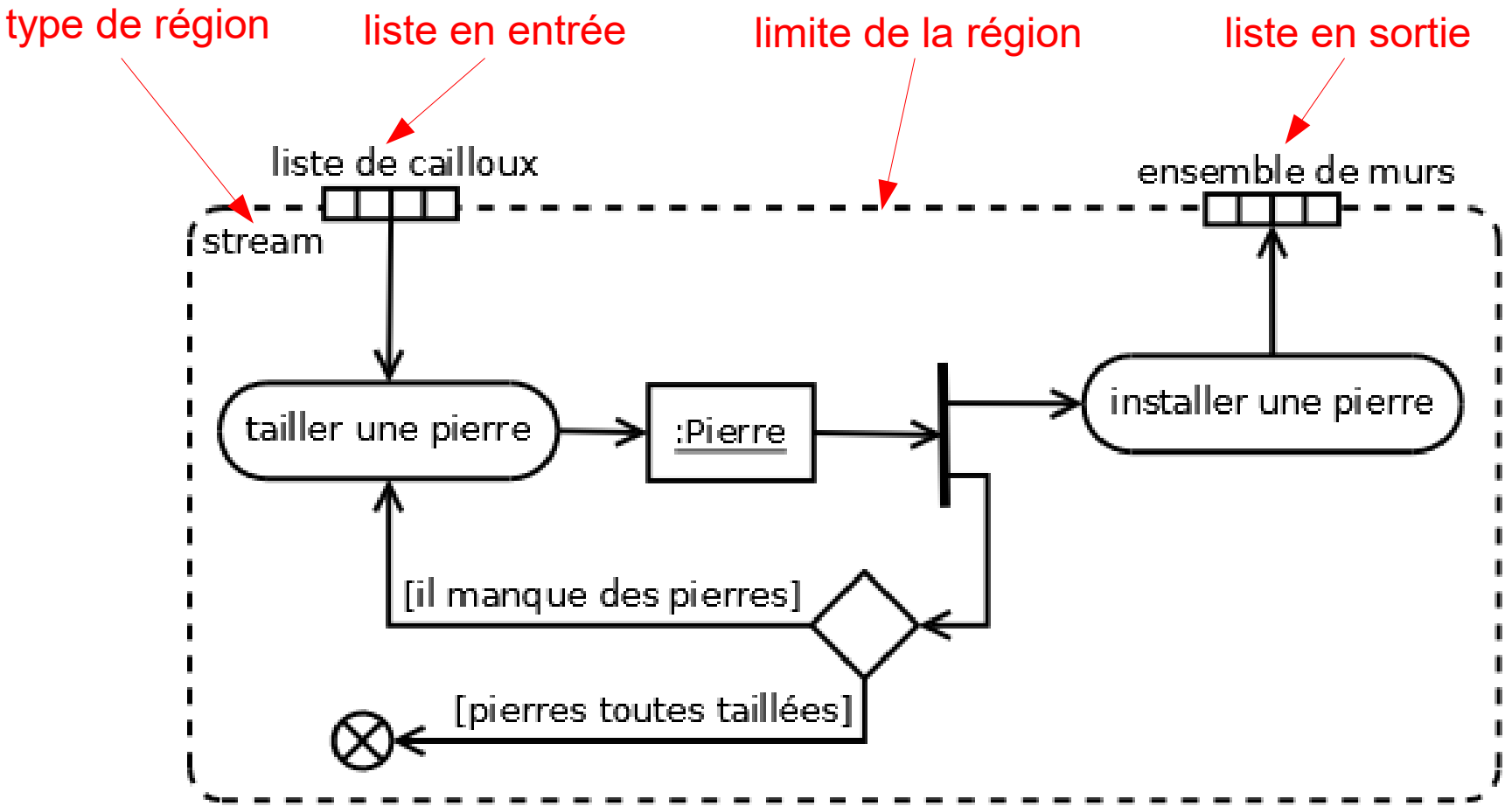


Diagramme d'activités : types d'activités

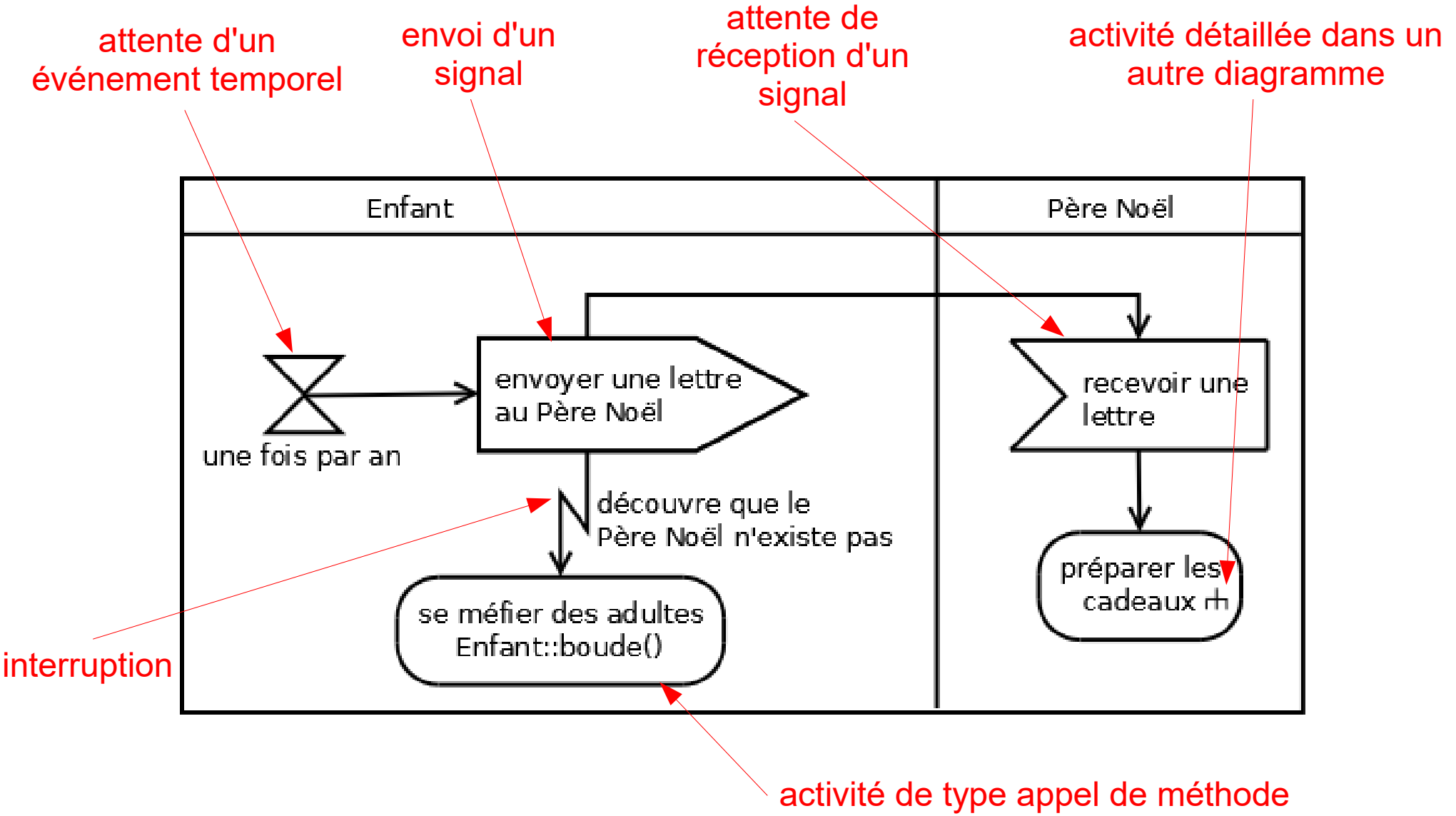
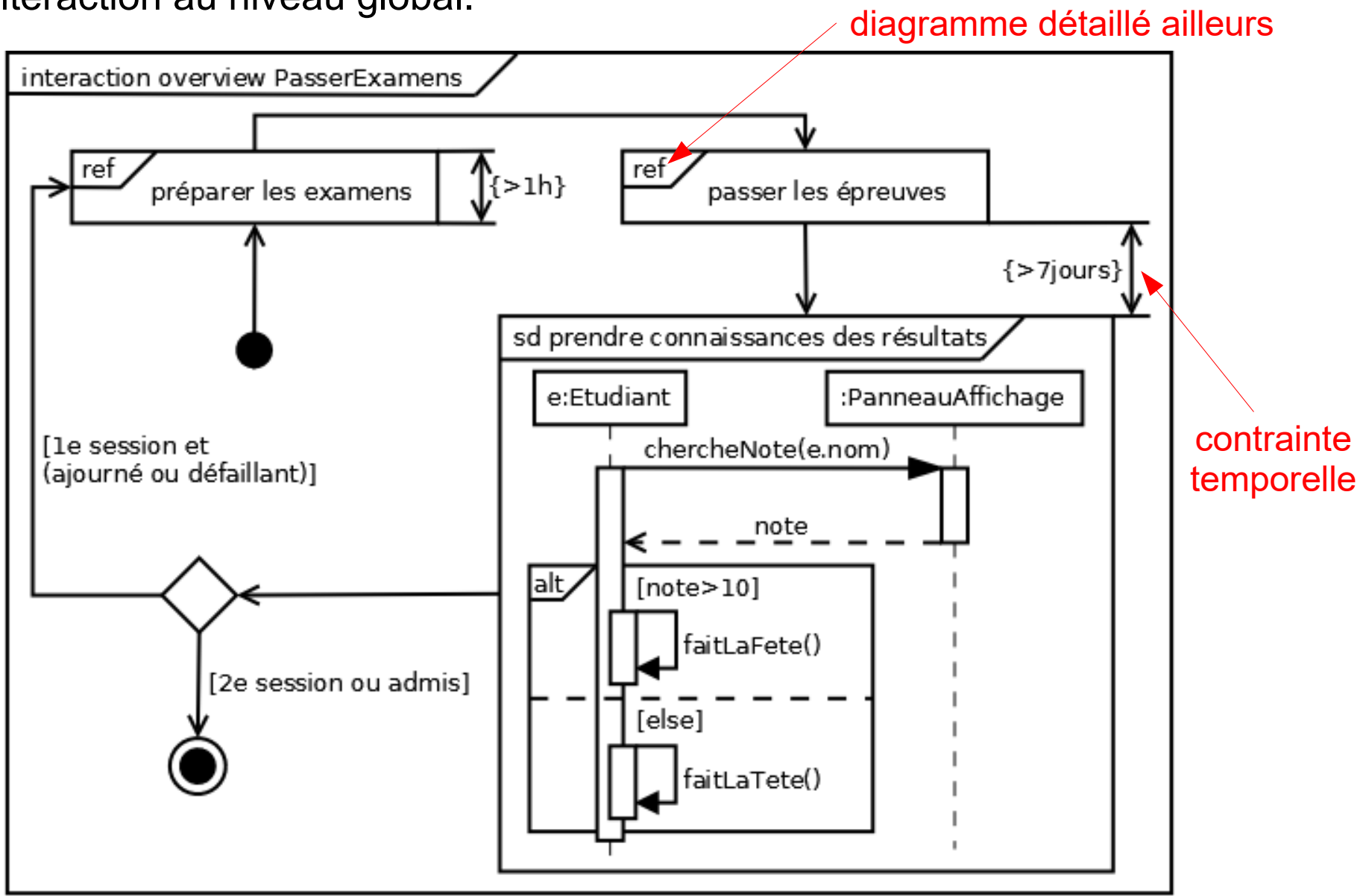
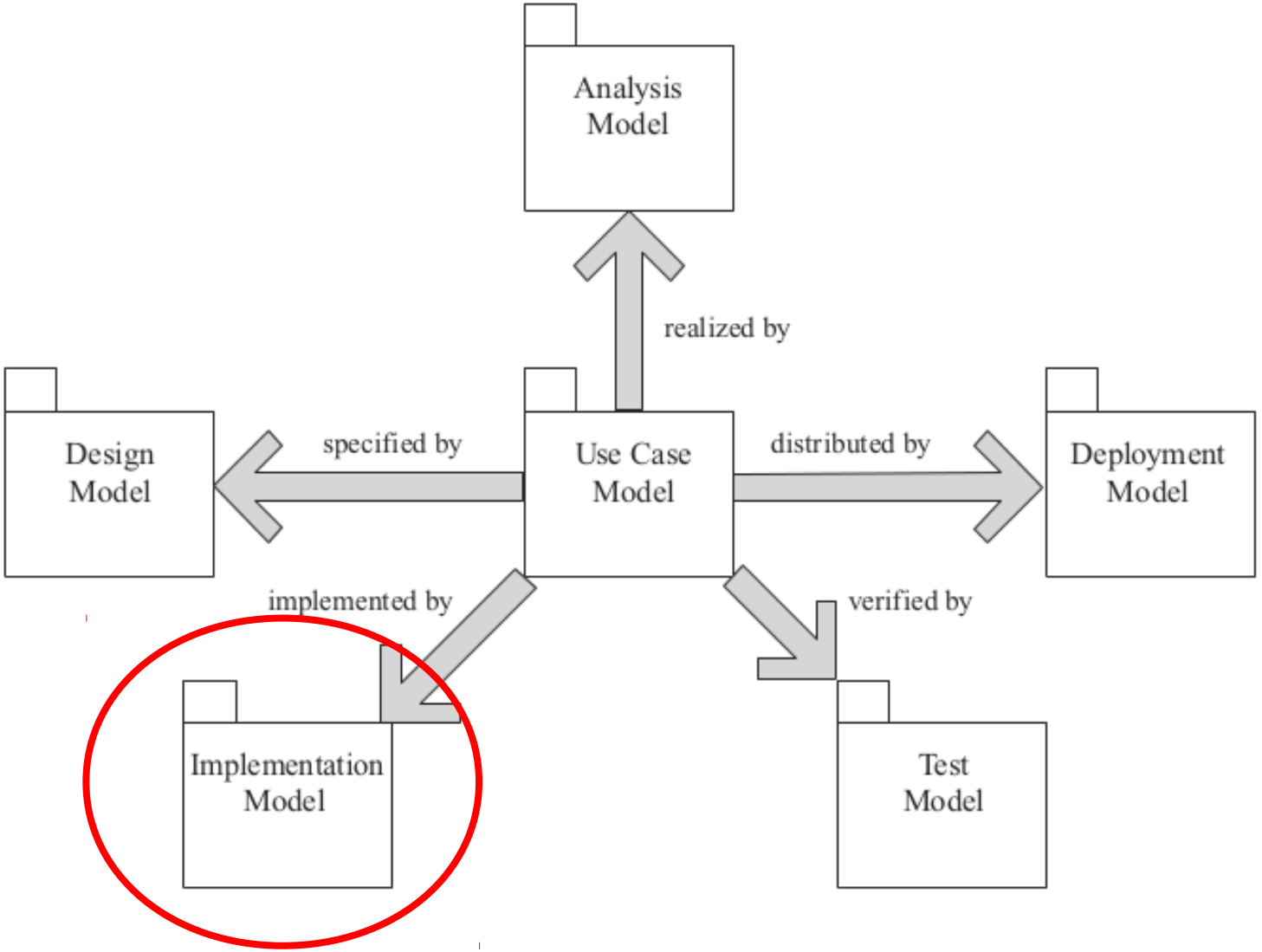


Diagramme général d'interactions

Le diagramme général d'interactions (interaction overview) décrit les flots d'interaction au niveau global.



Les modèles dans UP



Modèle d'implémentation

Dans UP, le modèle d'implémentation décrit l'architecture du système sous forme de composants.

Un composant regroupe un ensemble de briques logicielles (classes en POO) présentant des fonctionnalités liées ou complémentaires et formant un module réutilisable.

Un composant est souvent une librairie exécutable (dll, jar, …) partagée entre différentes applications et décrite par des interfaces.

Diagramme de composants

Un diagramme de composants décrit des composants liés par des interfaces.

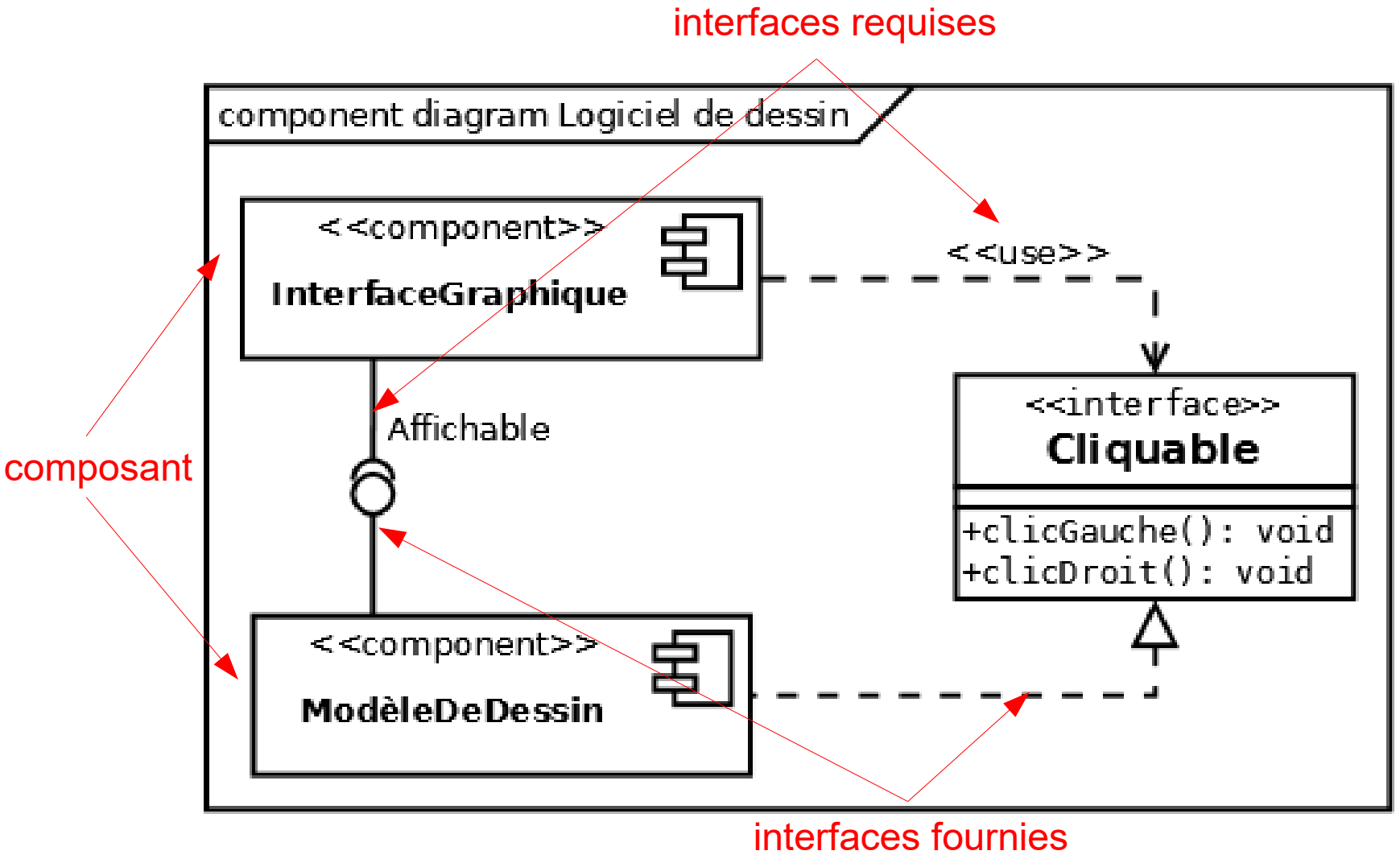
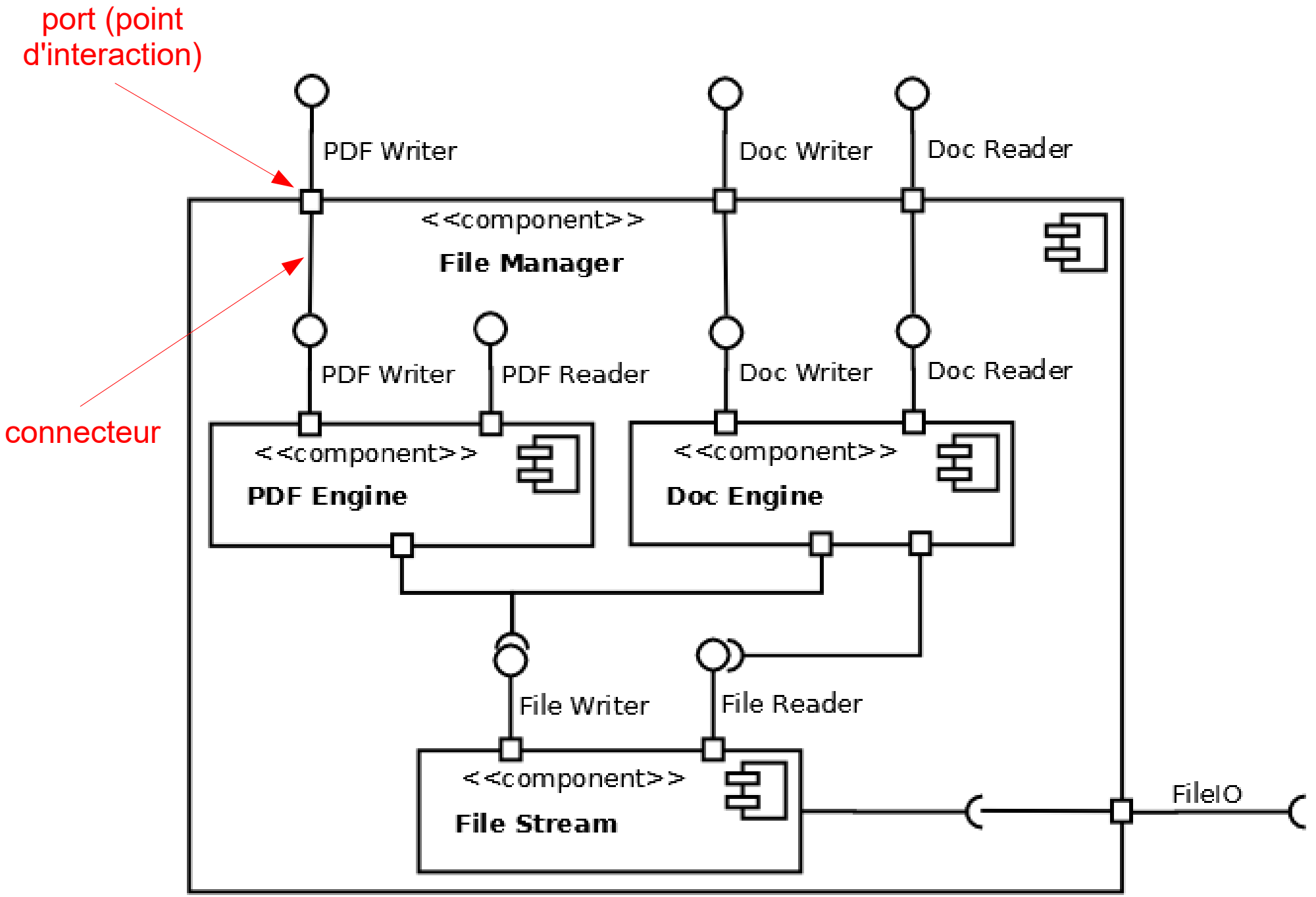
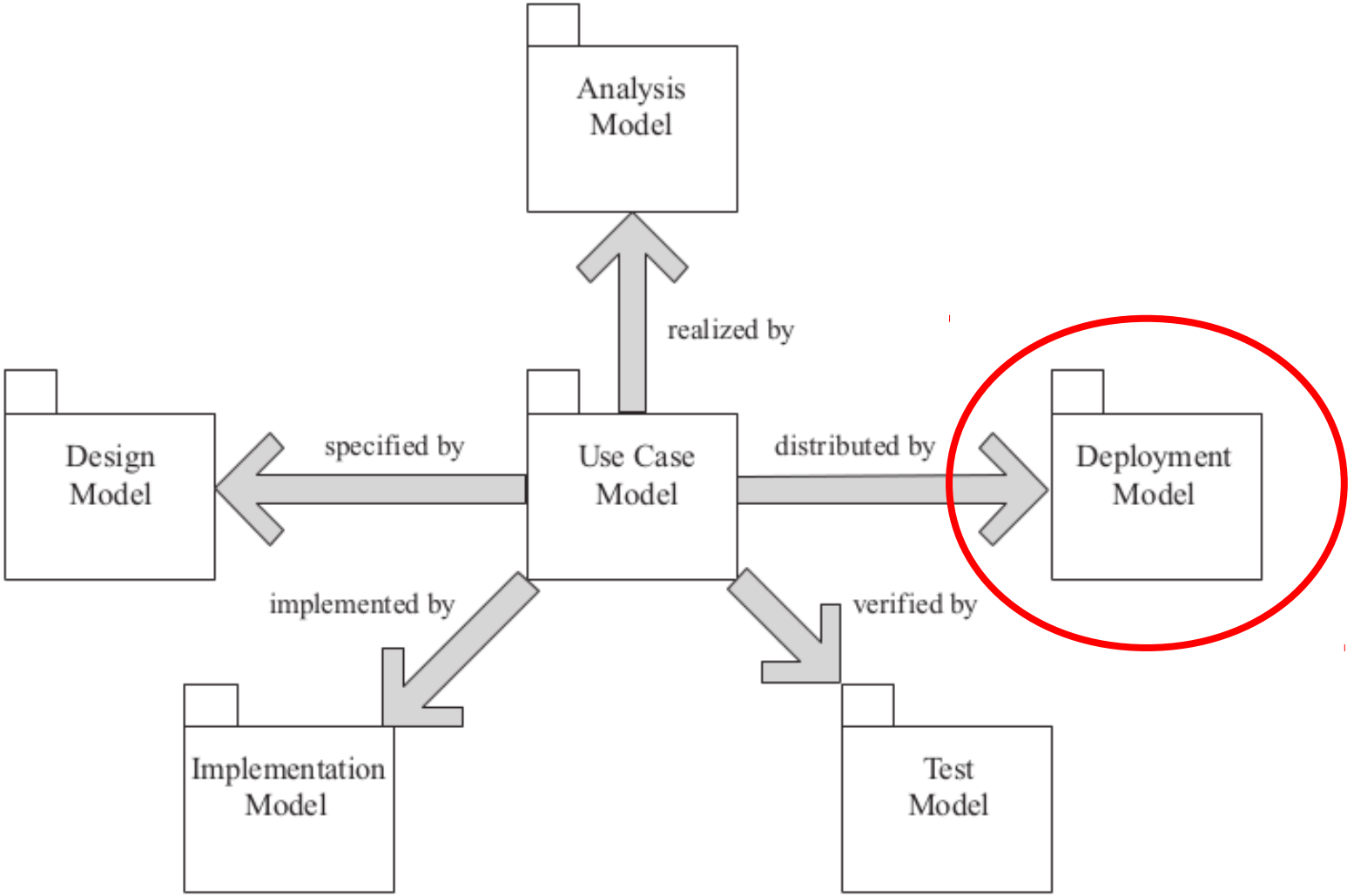


Diagramme de composants : composition



Les modèles dans UP



Modèle de déploiement

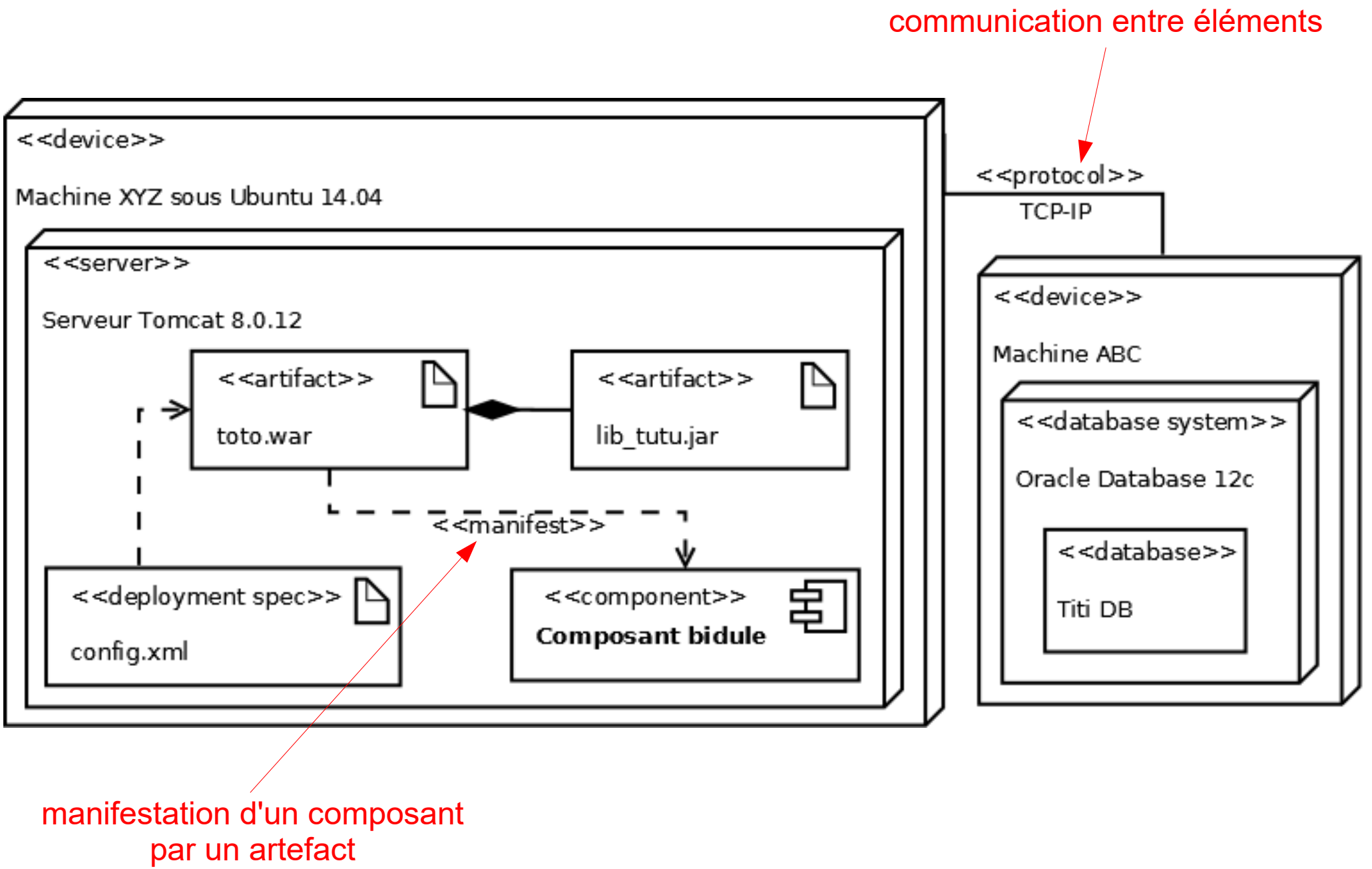
Dans UP, le modèle de déploiement décrit la façon dont le logiciel sera déployé sur une infrastructure informatique.

Le modèle de déploiement est spécifié essentiellement dans un diagramme de déploiement qui décrit :

- les éléments matériels et logiciels sur lesquels sont déployés les artefacts (éléments logiciels produits par le développement du système)

- les protocoles de communications entre éléments

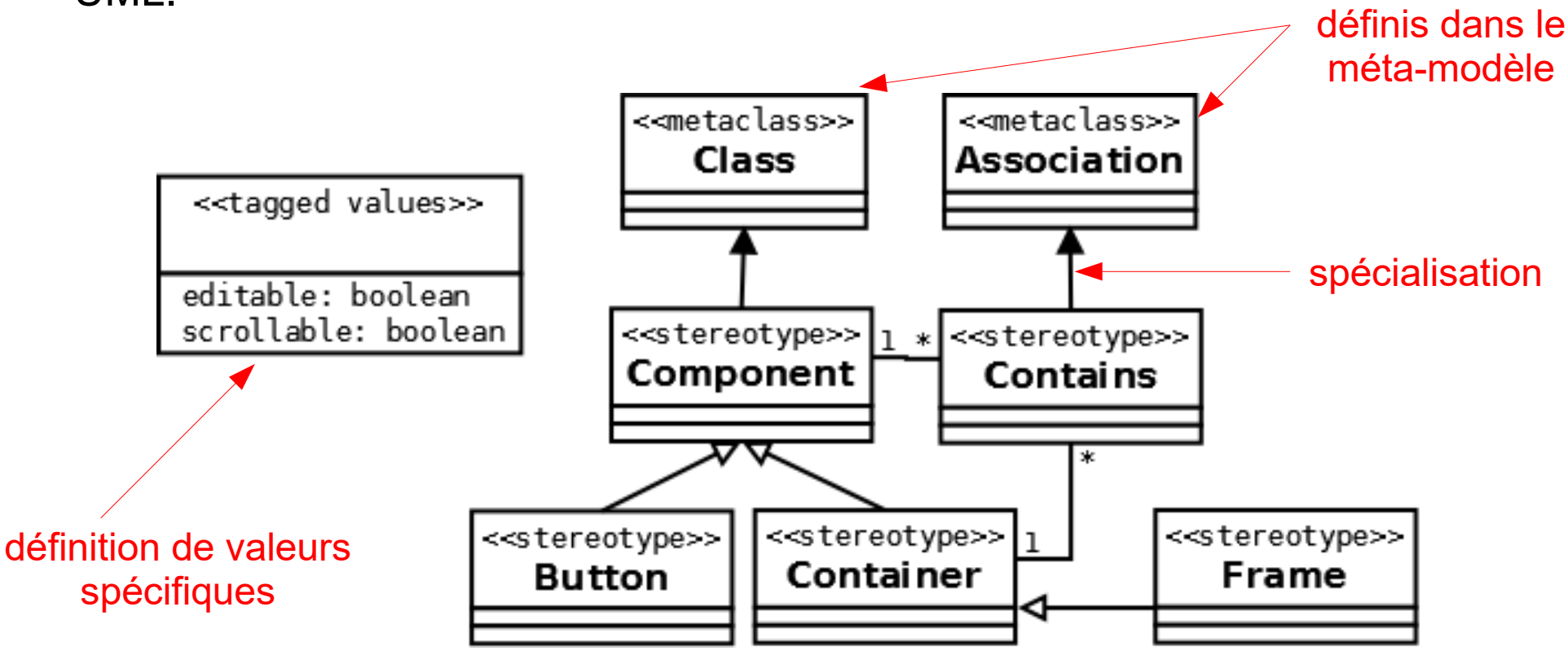
Modèle de déploiement (1/2)



Et ce n'est que le début ...

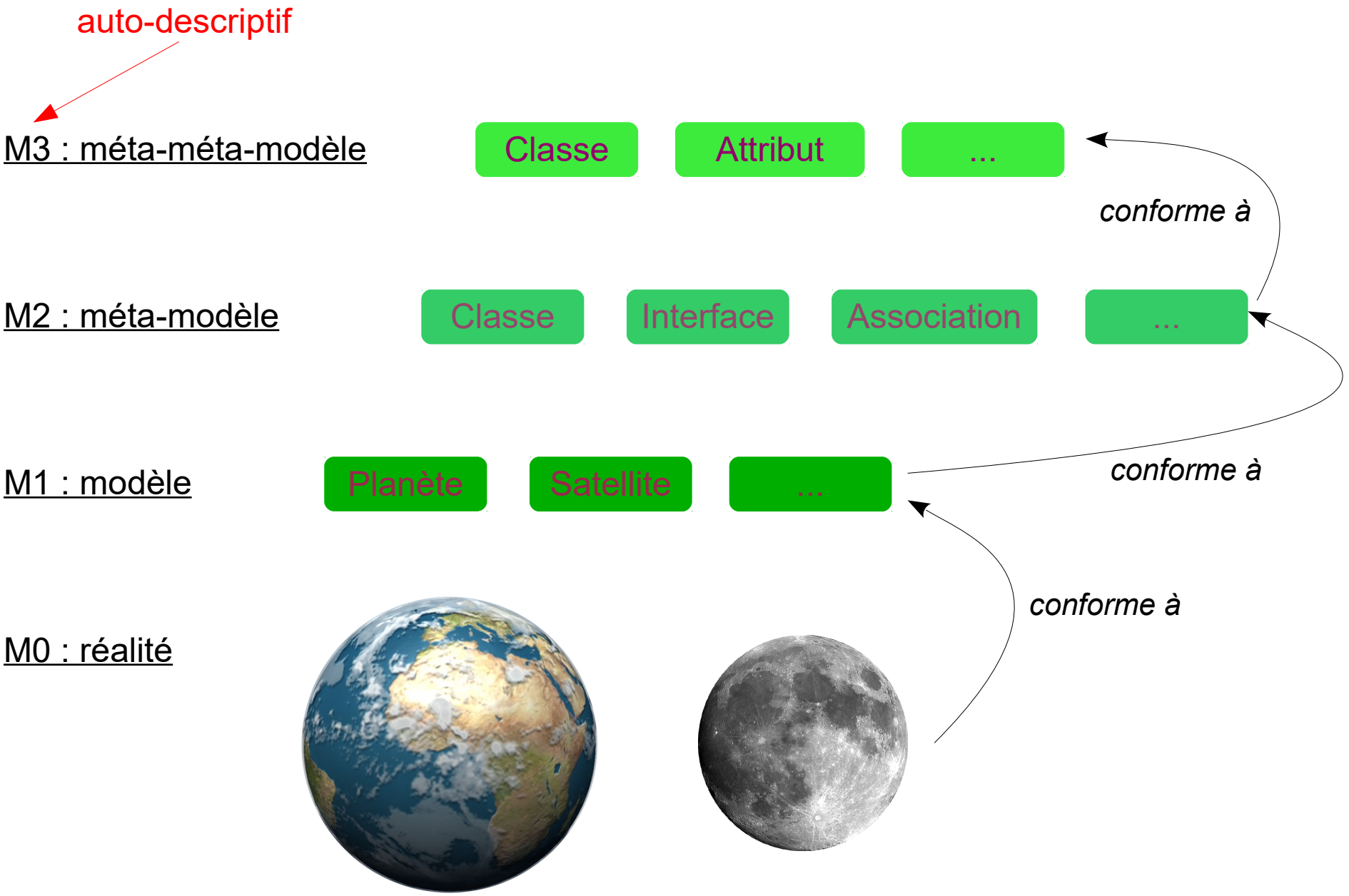
Diagramme de profil

Un diagramme de profil sert à définir des méta-modèles spécialisant UML.



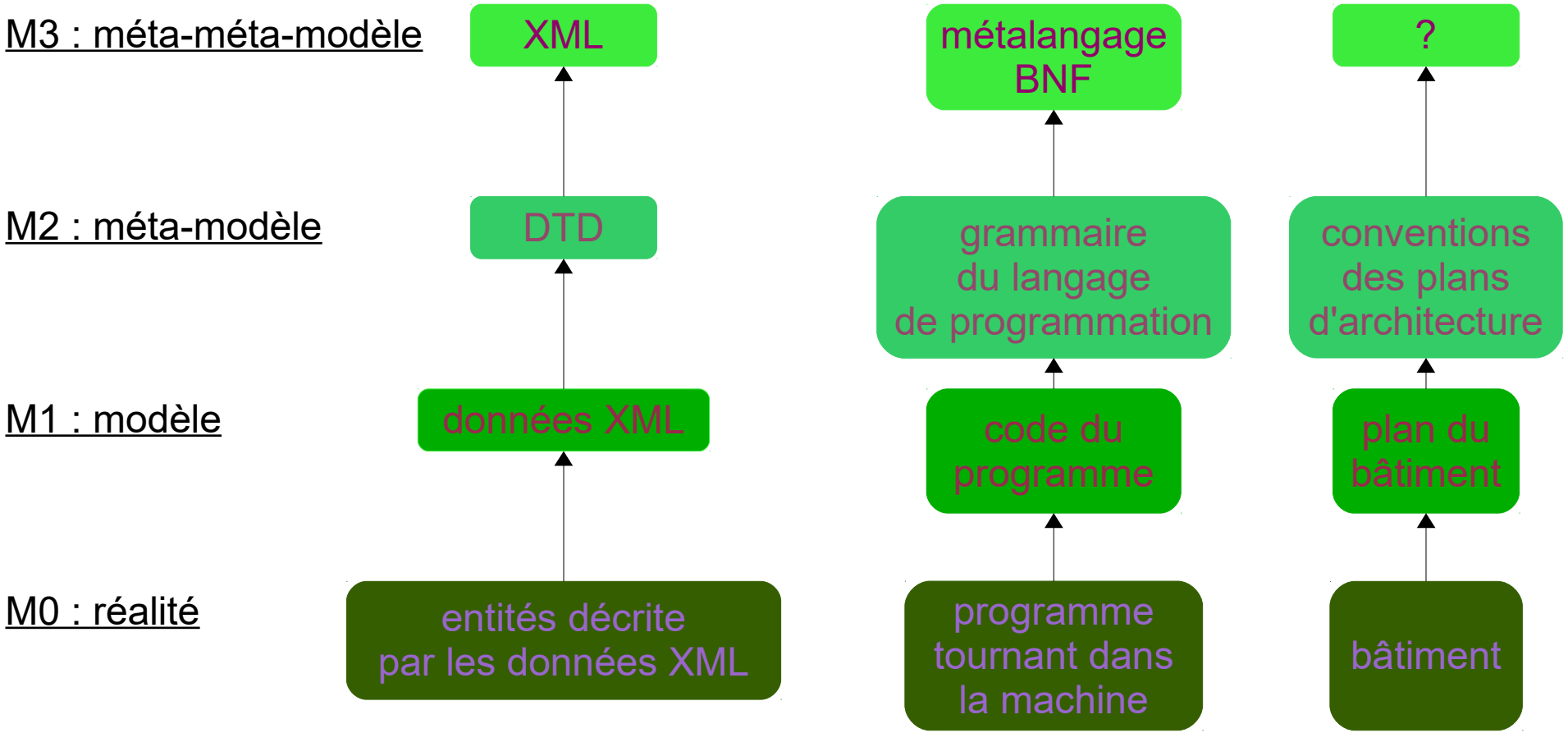
La spécification de contraintes à l'aide d'OCL (Object Constraint Language) permet d'obtenir des modèles vérifiables et formellement exploitables.

Métamodélisation (1/2)

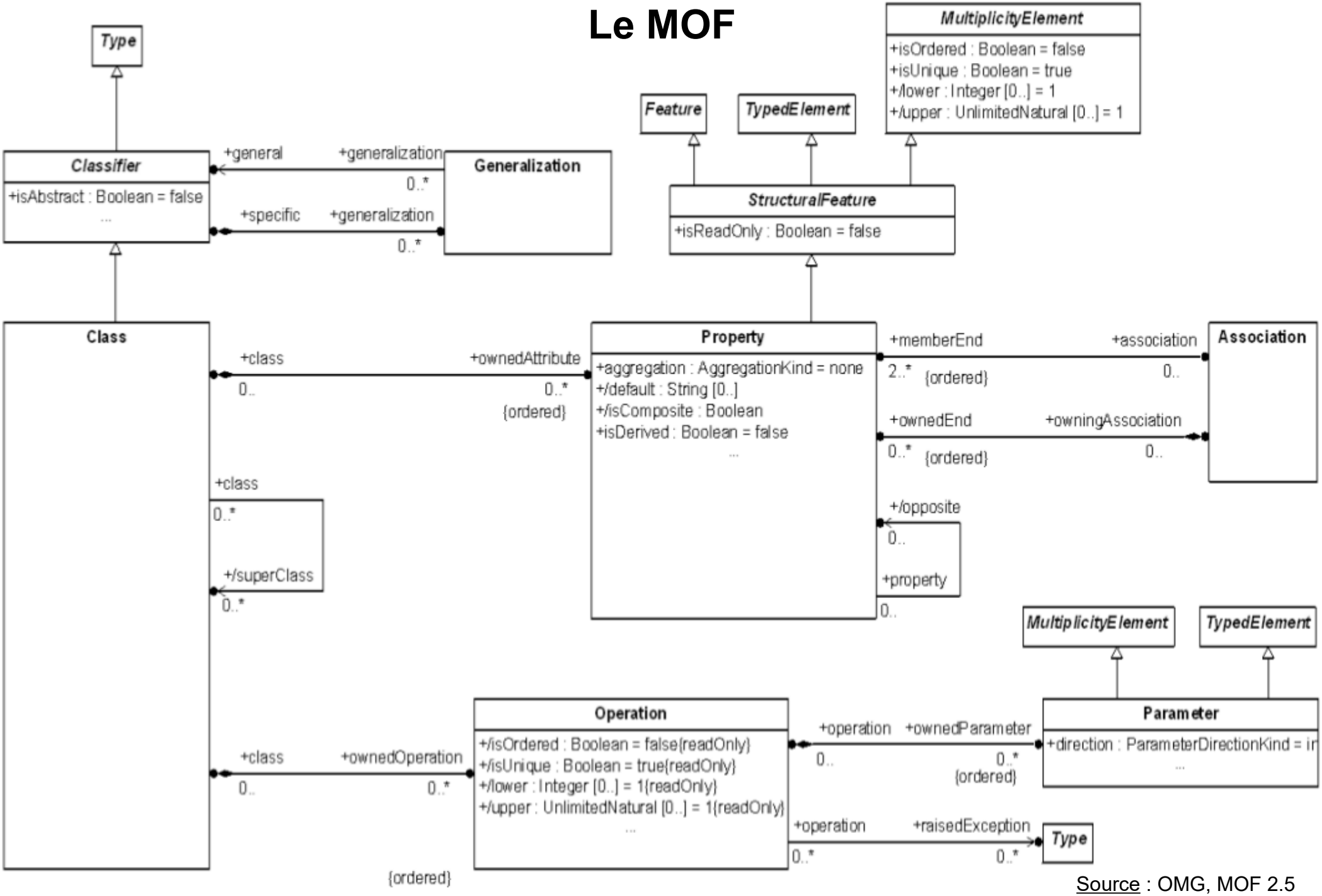


Métamodélisation (2/2)

Cette hiérarchie n'est pas propre au développement objet :



Le MOF



Source : OMG, MOF 2.5

Ingénierie des modèles

L'ingénierie des modèles définit des outils et méthodes pour :

- la modélisation et la méta-modélisation

- la transformation de modèles

- l'utilisation de modèles pour la conception et la production (automatique ??) de logiciels

→ les outils : UML, MOF, XMI (XML Metadata Interchange), OCL

→ la méthodologie : MDA (Model Driven Architecture)

L'ingénierie des modèles est appliquée dans l'ingénierie dirigée par les modèles (IDM), qui vise le développement d'applications par l'utilisation des outils et méthodes issues de l'ingénierie des modèles.