Analyse et Conception Objet de Logiciels

UML
Unified Modeling Language
Partie 2

UML : aspects dynamiques

Modèles dynamiques :

spécifient le *comportement* du système logiciel au cours du temps

- diagrammes de séquence
- diagrammes de collaboration
- diagrammes d’états-transitions

Diagrammes de séquence

Fonctionnalités du système

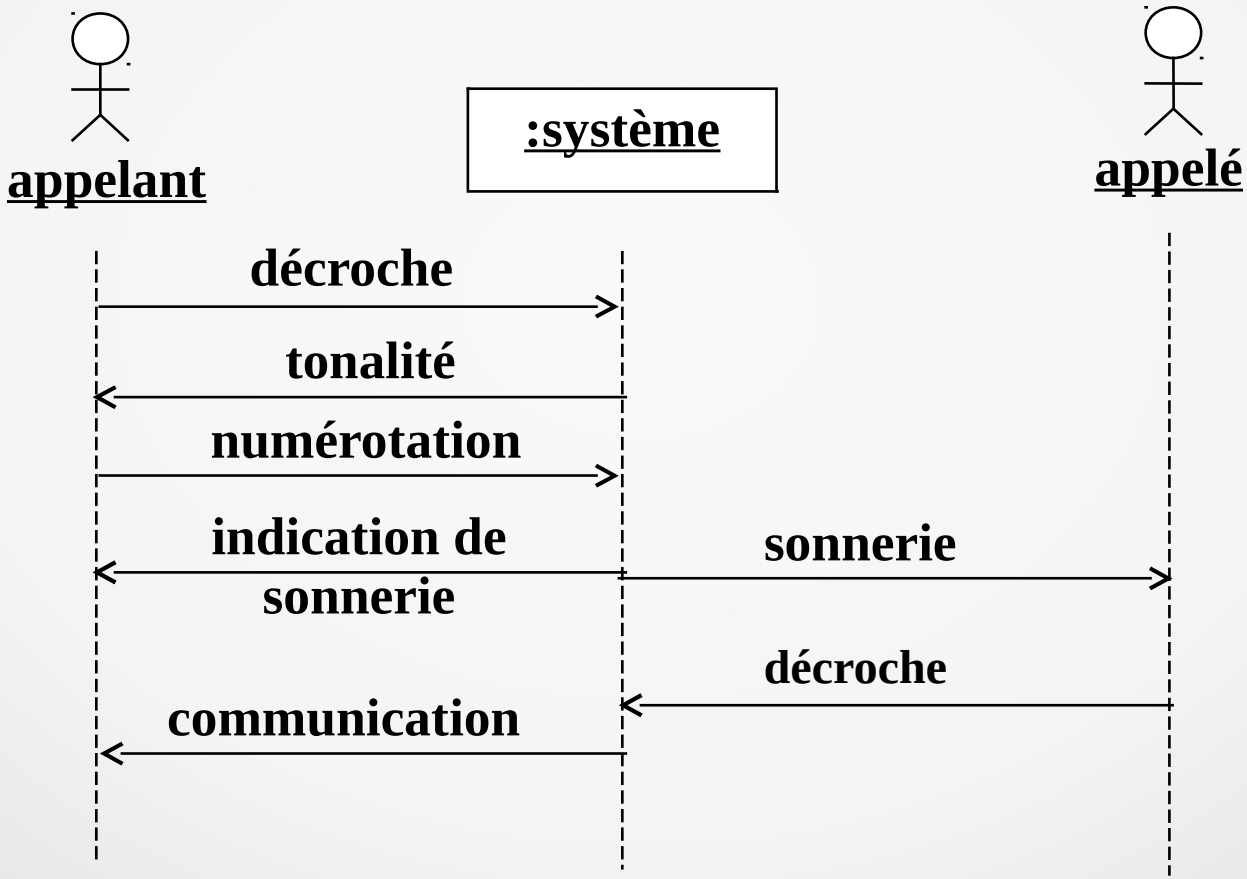
Documentation des cas d'utilisation : représentation des interactions entre les acteurs et le système

Comportement interne du système

Représentation des interactions entre objets d'un point de vue chronologique.

Exemple : communication téléphonique

Documentation d'un cas d'utilisation



Interactions entre objets du système

Représentation précise des interactions entre objets qui composent le système.

Les objets interagissent par l'envoi de messages.

Les messages correspondent à :

- des appels d'opération,
- des retours d'opération,
- des signaux.

Messages

Trois sortes de messages :

a) *Appel d'opération ou flot de contrôle imbriqué.*

La séquence imbriquée est entièrement faite avant que l'appelant ne continue : l'appelant est bloqué jusqu'à ce que l'appelé termine.

Notation : 

b) *Envoi de message "à plat", non imbriqué.*

L'appelant n'est pas bloqué jusqu'à ce que l'appelé termine (message asynchrones).

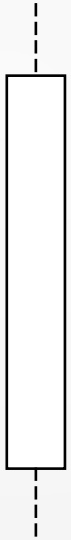
Notation : 

c) *Retour d'opération.*

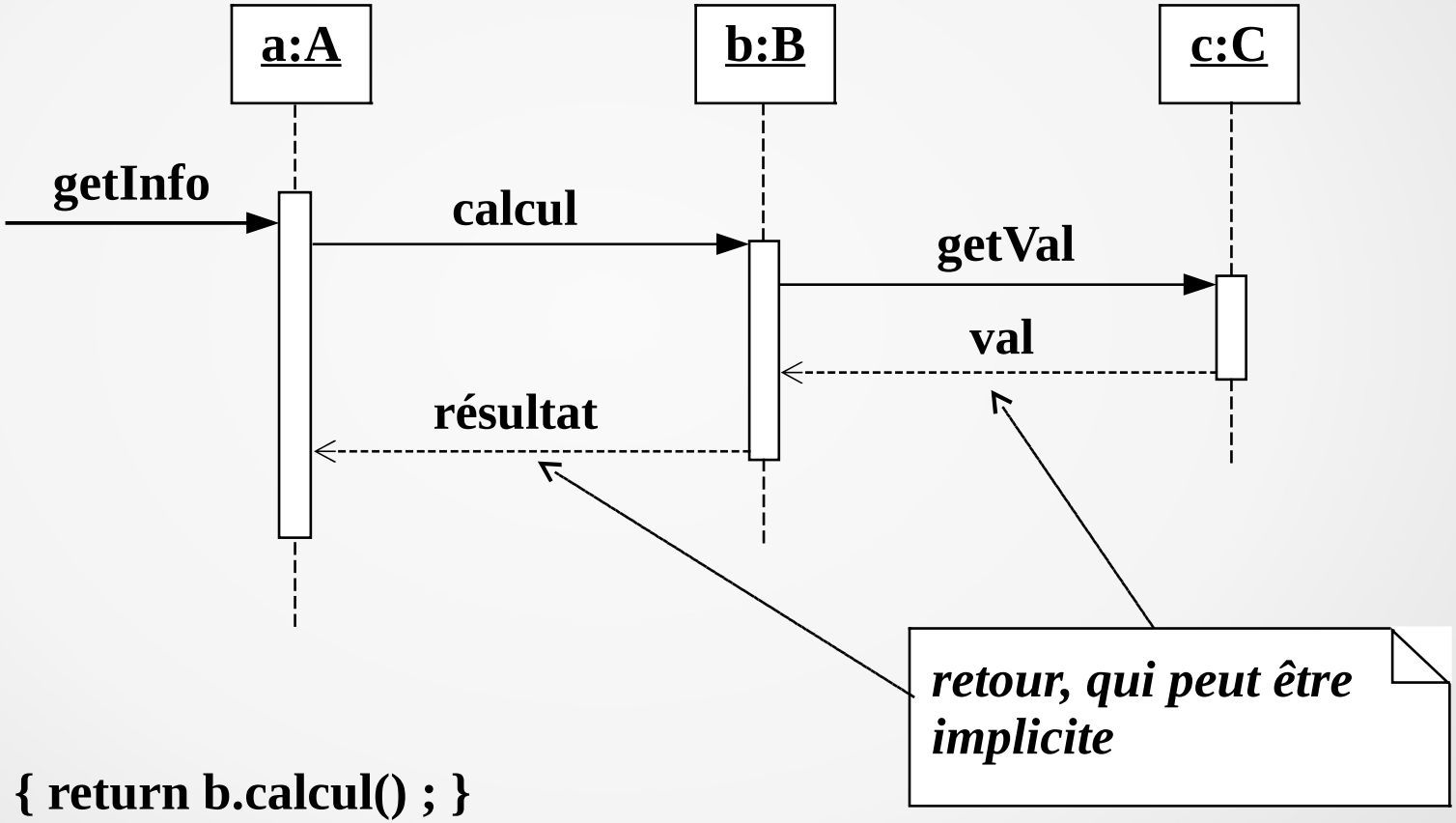
Notation : 

Période d’activation

Période de temps pendant laquelle un objet effectue une action, soit directement, soit par l’intermédiaire d’un autre objet qui lui sert de sous-traitant



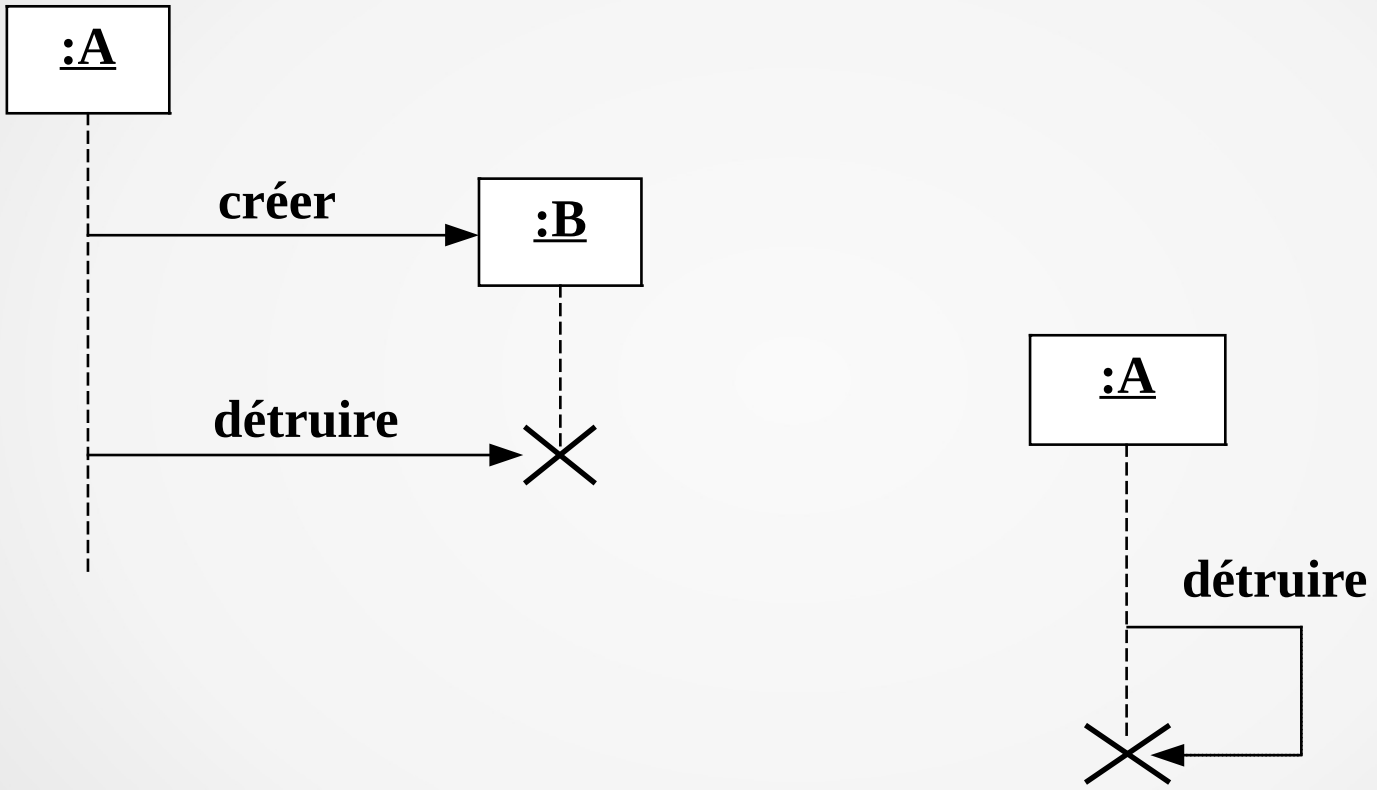
Exemple : appel d'opération



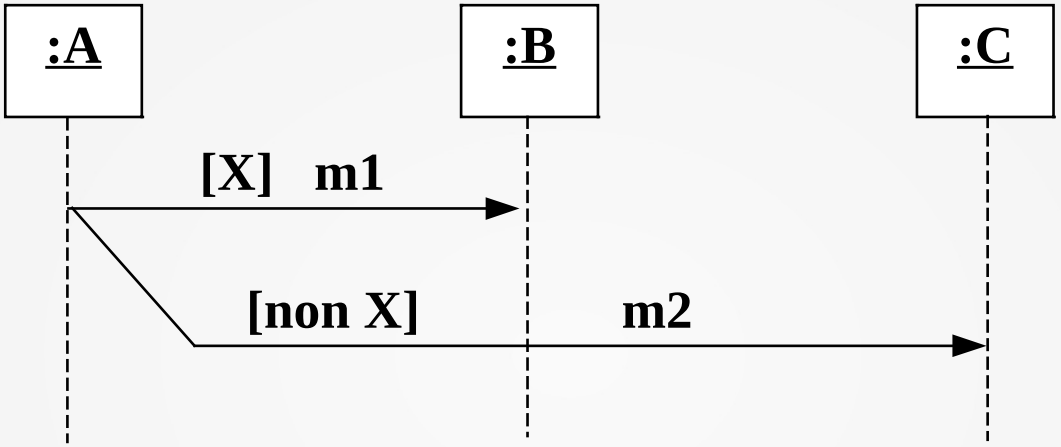
a.getInfo()

int getInfo() { return b.calcul() ; }

Création et destruction d'objets

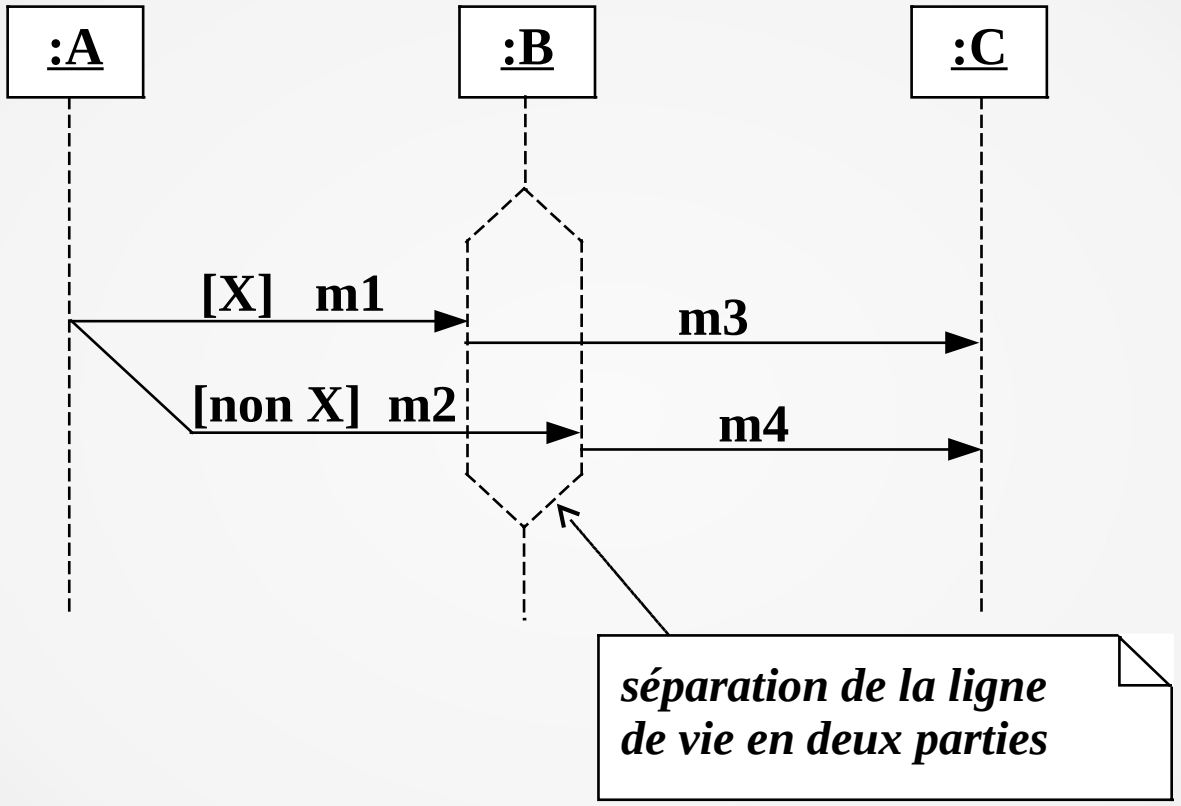


Conditions



Si la condition **X** est satisfaite,
le message **m1** est envoyé à **:B**,
sinon
le message **m2** est envoyé à **:C**

Conditions

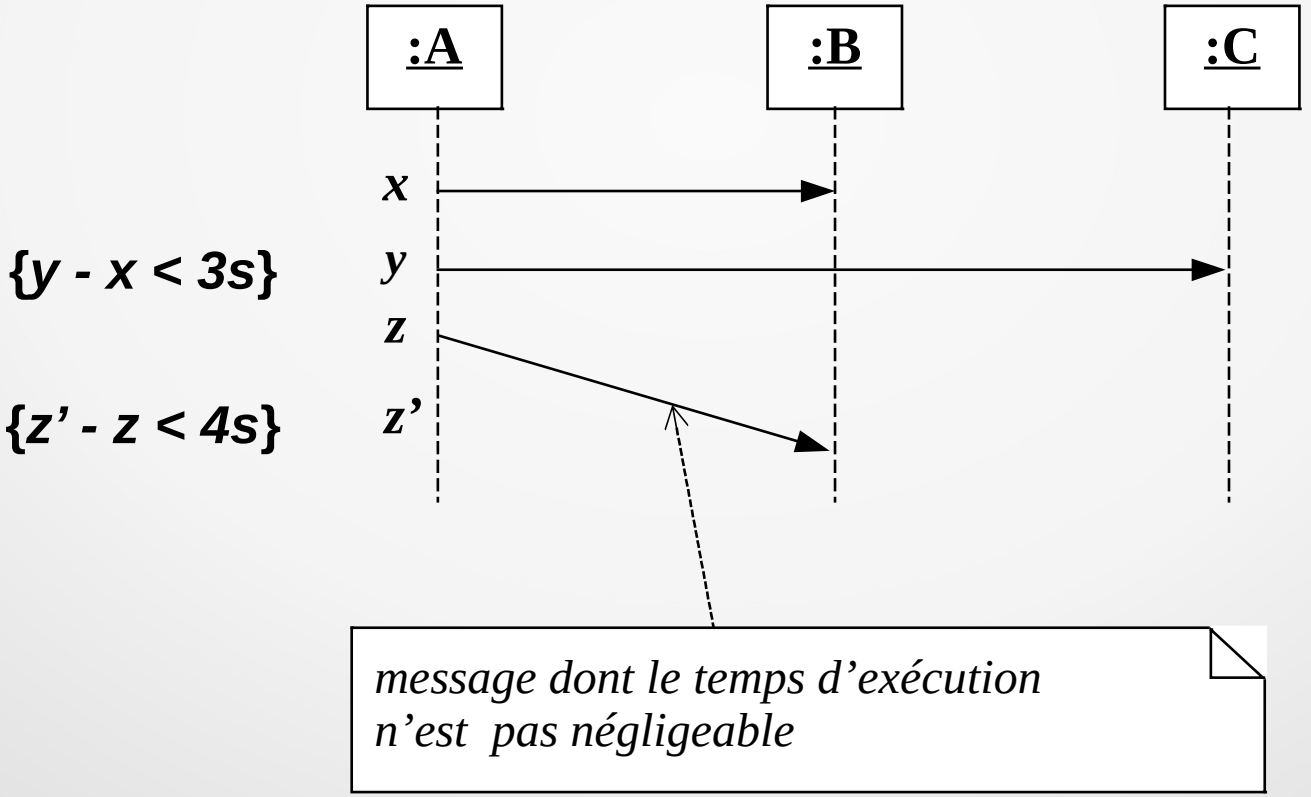


Si la condition **X** est satisfaite : **m1 ; m3**

Sinon : **m2 ; m4**.

Contraintes temporelles

- Nommage des instants d'émission et de réception des messages
- Contraintes de temps



Diagrammes de collaboration

Représente les interactions entre les objets (et éventuellement les acteurs) d'un point de vue spatial.

Par opposition aux diagrammes de séquence, les liens entre les différents objets sont explicitement représentés.

Pour mettre en évidence la dimension temporelle, les messages envoyés par les différents objets peuvent être numérotés.

Exemple 1 : communication téléphonique

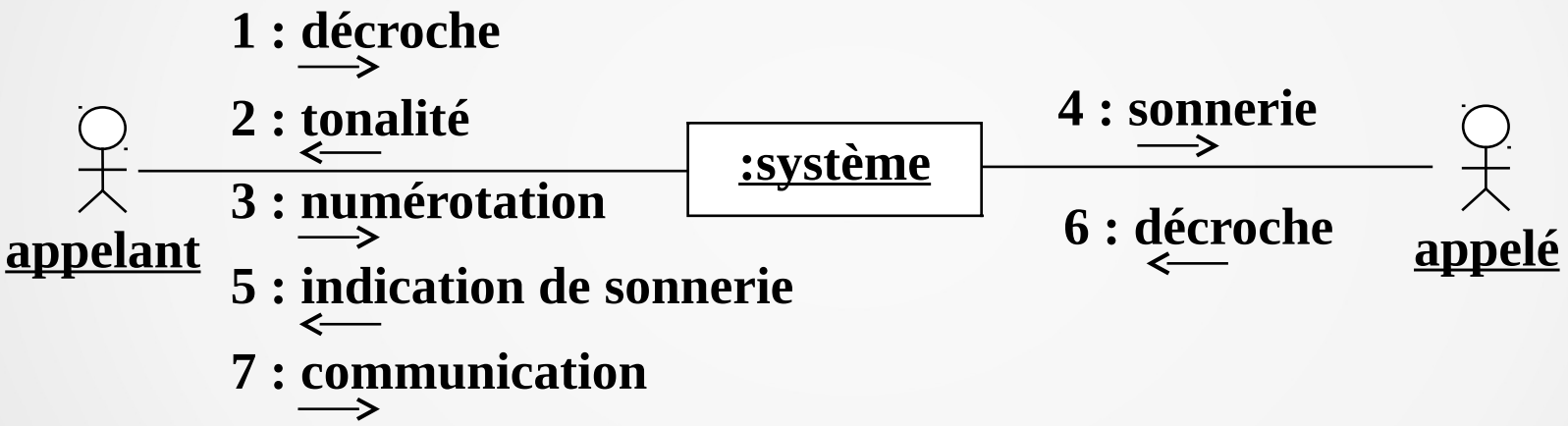
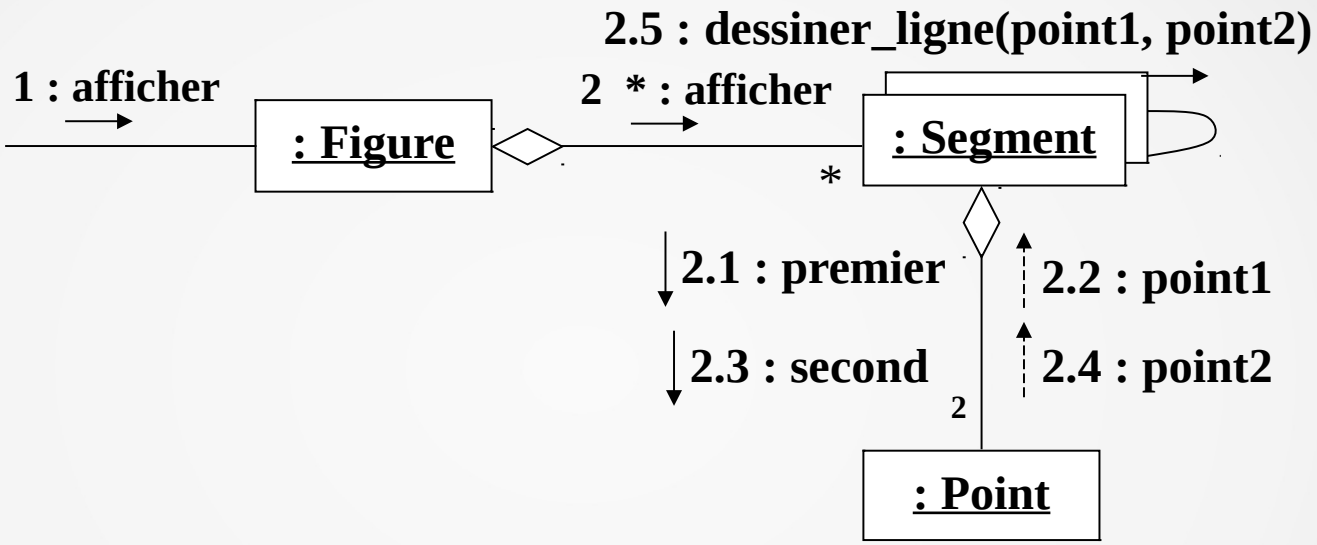


Diagramme de collaboration

Exemple 2 : affichage d'une figure



« Multi-objet », qui représente l'ensemble des segments dont la figure est composée

2 * : afficher →

On envoie le message à chaque instance en relation avec la figure.

Diagrammes d’états-transitions

- Inspirés des « state-charts » de David Harel
- Automates hiérarchiques, qui peuvent être mis en parallèle
- Permettent de décrire
  - les aspects dynamiques d’un cas d’utilisation, d’un acteur, ou d’un système,
  - le comportement d’un objet d’une classe.

Notion d’état

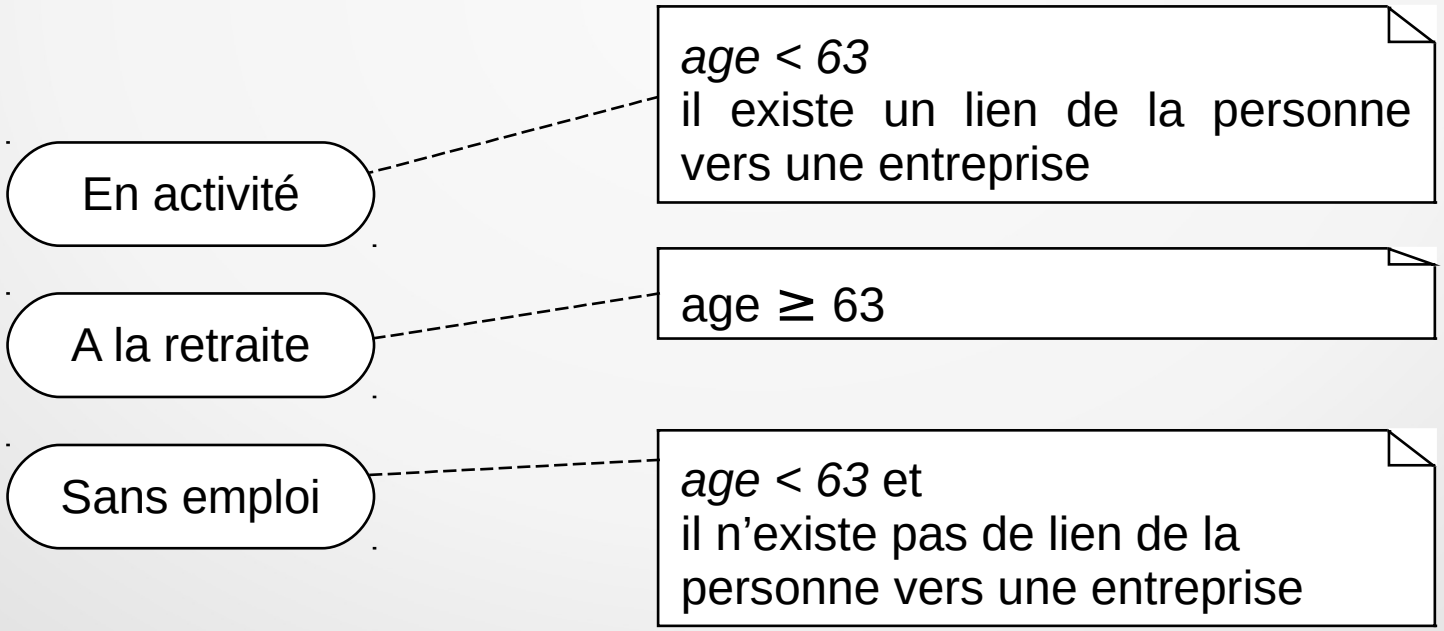
L’état d’un objet est caractérisé par :

- l’ensemble des valeurs de ses attributs,
- l’ensemble des liens qu’il entretient avec d’autres objets

État d’un automate : *abstraction* qui représente un ensemble d’états de l’objet

Exemple : état d'une personne / emploi

Trois états pour une personne : en activité, à la retraite et sans emploi



États initial et final

- État initial : « pseudo-état », dans lequel on se trouve avant que l'objet soit créé
- État final : « pseudo-état », dans lequel on se trouve une fois que l'objet est détruit

État initial

État final

Transitions

Permettent de passer d'un état à un autre

Si l'objet se trouve dans l'état A :

- si l'*événement* se produit, et si la *condition* est vraie, l'objet effectue l'*action* et se retrouve dans l'état B.

Le passage d'un état à un autre est considéré comme instantané, et l'exécution de l'action également .

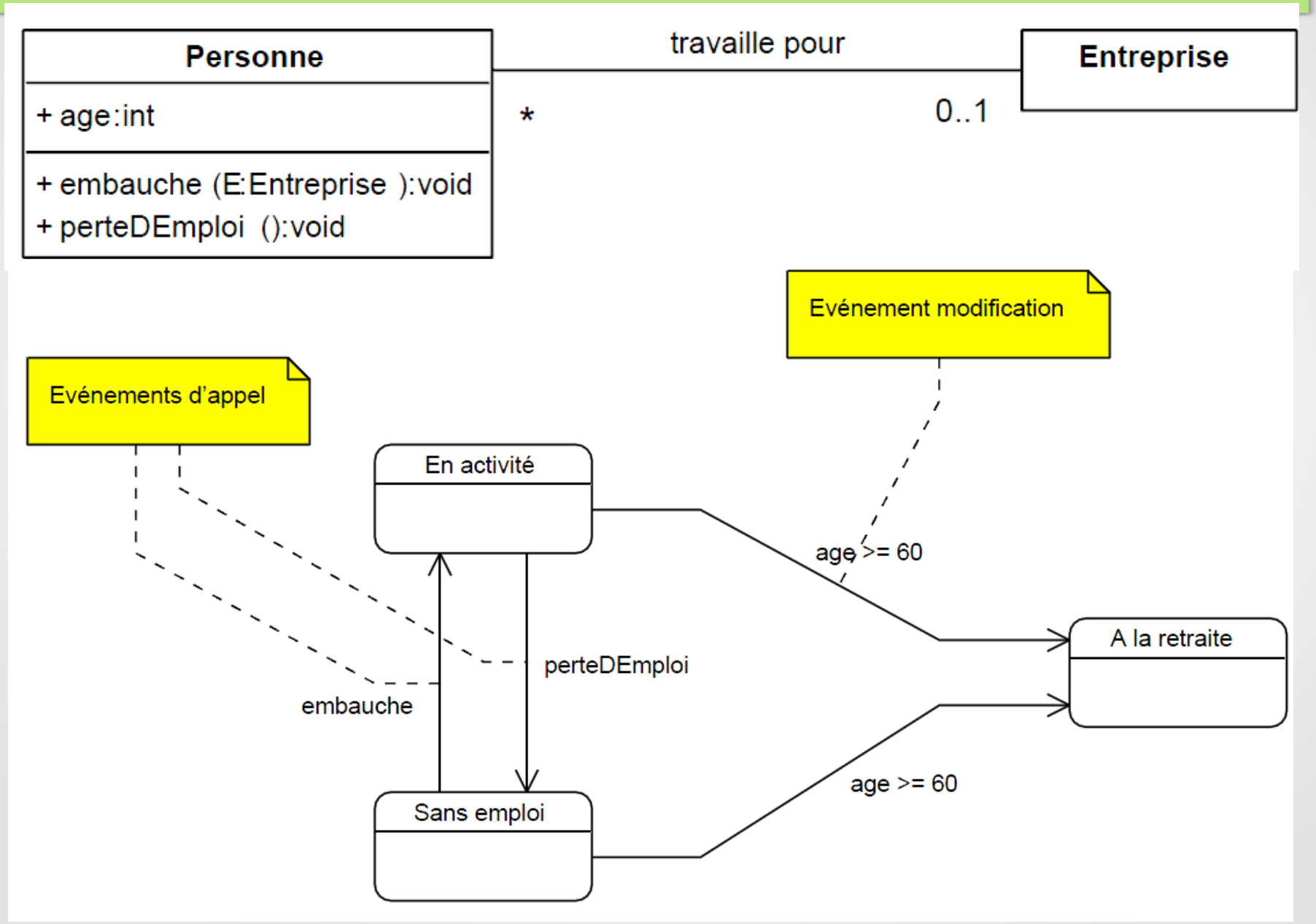
La condition (également appelée « garde ») est évaluée uniquement si l'événement se produit.

Evénements

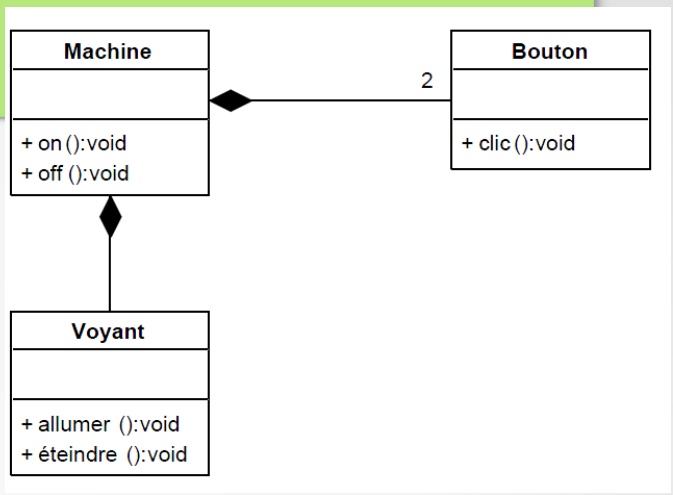
4 types d'événements en UML

- événement d'*appel* (« call event ») : causé par l'appel d'une opération de la classe
- événement de *modification* (« change event ») : causé par le passage d'une condition de la valeur faux à la valeur vrai, suite à un changement de valeur d'un attribut ou d'un lien
- événement *temporel* (« time event ») : survient suite à l'expiration d'une temporisation
- événement *signal* (« signal event ») : stimulus asynchrone entre deux objets (ex : clic de souris)

Exemple 1


Personne
+ age:int
+ embauche (E:Entreprise):void
+ perteDEmploi ():void
travaille pour
*
0..1
Entreprise
Evénement modification
Evénements d'appel
En activité
age >= 60
A la retraite
perteDEmploi
embauche
age >= 60
Sans emploi

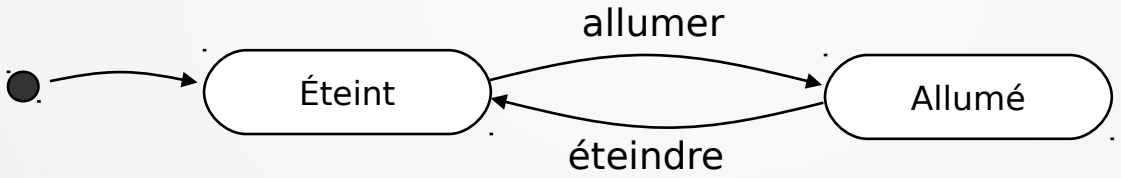
Exemple 2



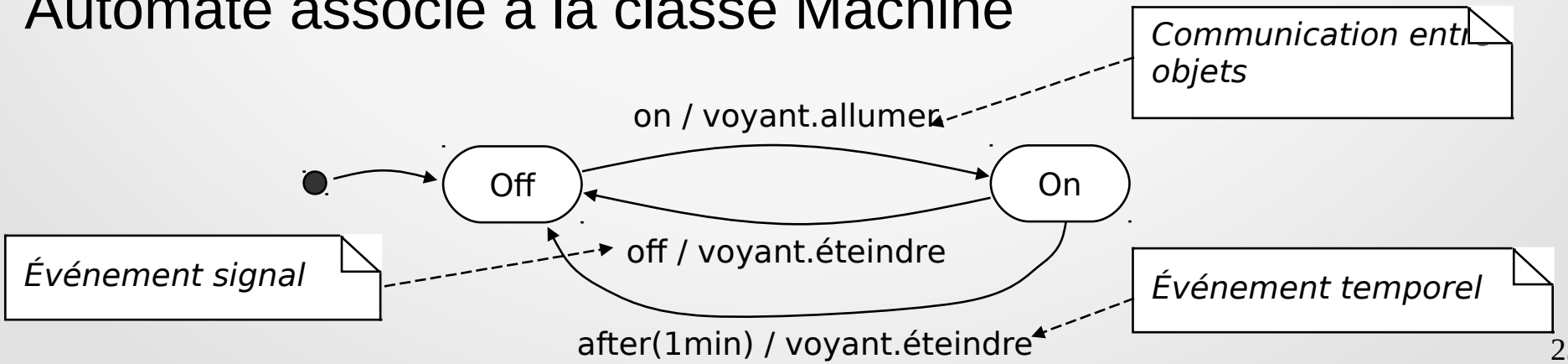
Machine avec boutons et voyant

La machine a deux boutons : un bouton pour la mettre sous tension (signal : on) et un bouton pour la mettre hors-tension (signal : off).

Automate associé à la classe Voyant



Automate associé à la classe Machine



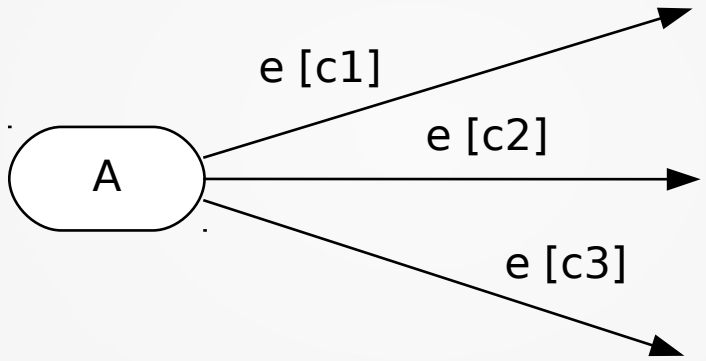
Non déterminisme

Les automates d'UML peuvent être non déterministes : il peut exister plusieurs transitions déclenchées par le même événement qui partent du même état.

Sauf si le comportement est vraiment non déterministe (par exemple si le hasard intervient), il est préférable d'utiliser des automates déterministes.

Non déterminisme et gardes

Garde : expression booléenne évaluée lorsqu'un événement se produit



Pour que l'automate soit déterministe, il faut que les conditions c1, c2 et c3 soient mutuellement exclusives.

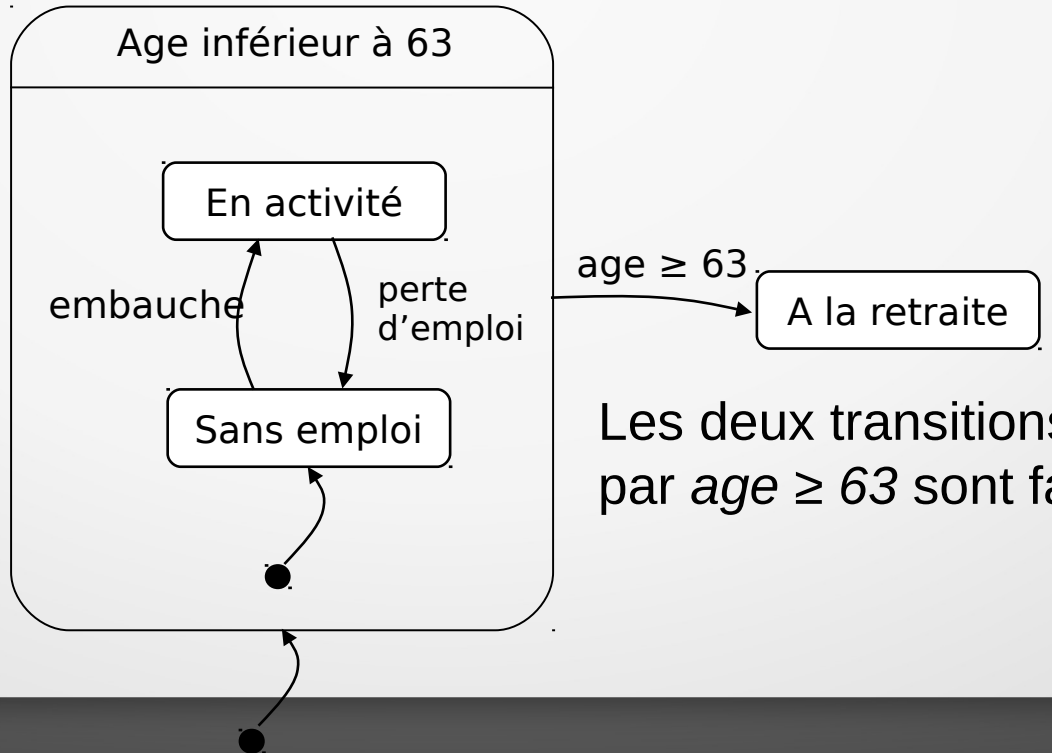
État composite

État qui se décompose en plusieurs *sous-états*

Les états composites permettent de :

- structurer les automates,
- rendre les automates plus lisibles,
- factoriser des transitions.

Exemple



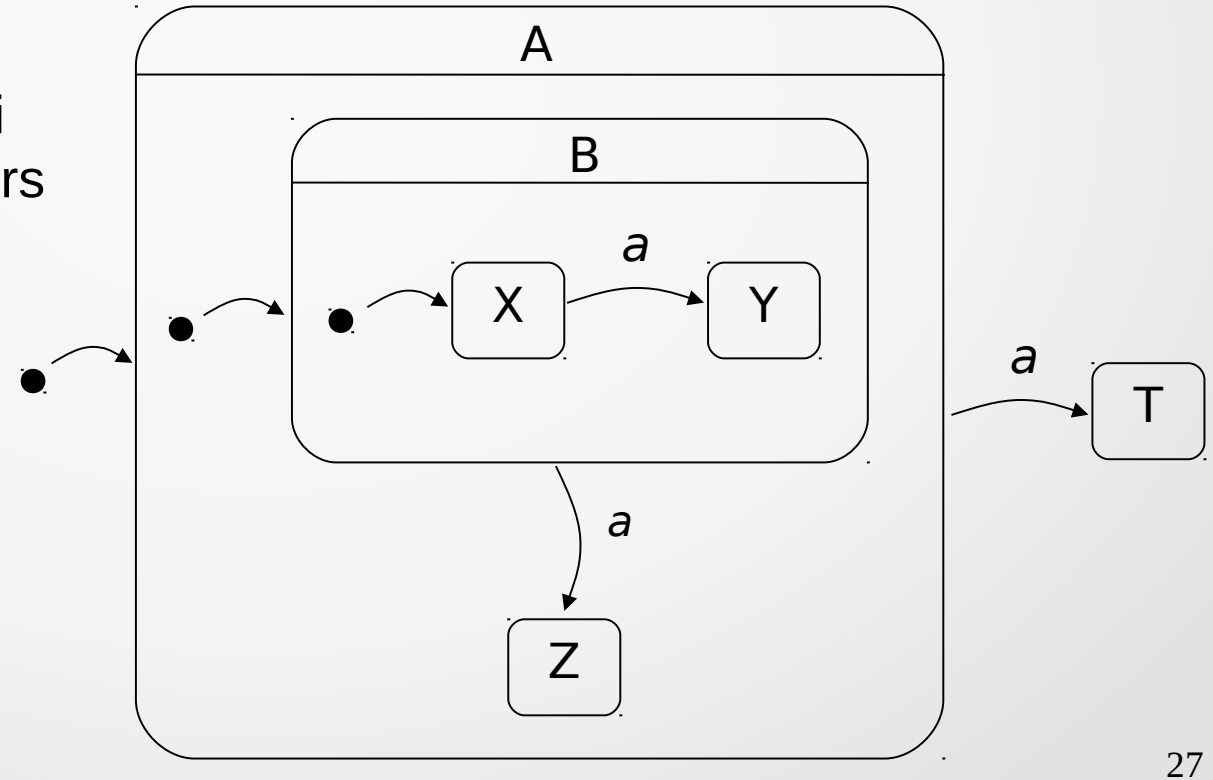
Les deux transitions étiquetées par *age ≥ 63* sont factorisées.

Priorité des transitions

Lorsque deux transitions sont étiquetées par le même événement, *la transition qui part du sous-état est prioritaire sur la transition qui part de l'état englobant.*

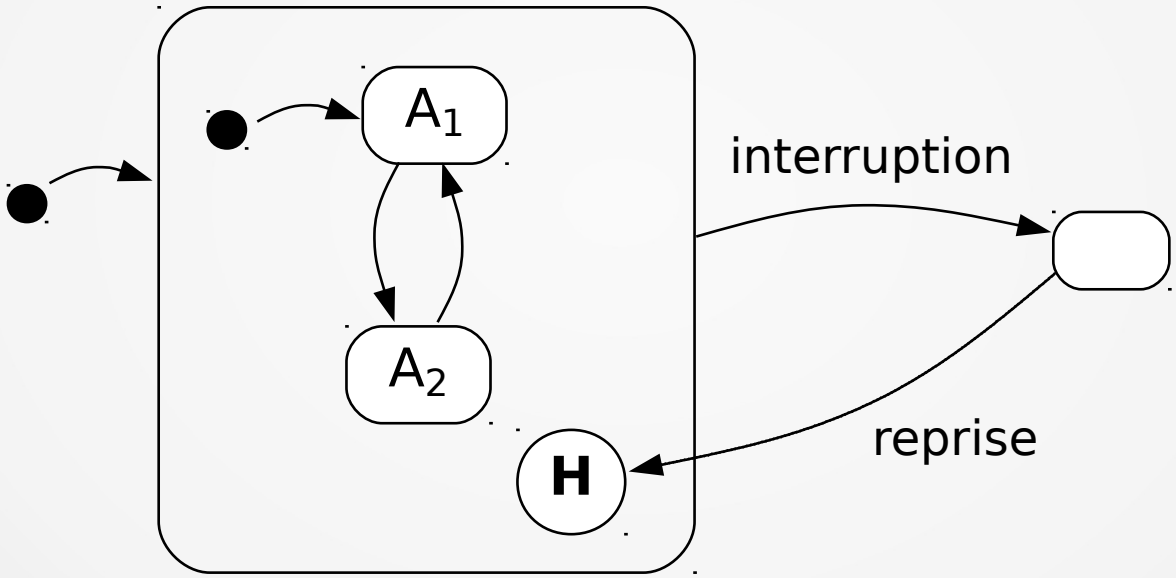
Exemple

Si on est dans l'état X, et si l'événement *a* est reçu, alors on passe dans l'état Y
(et non dans l'état Z ou T).



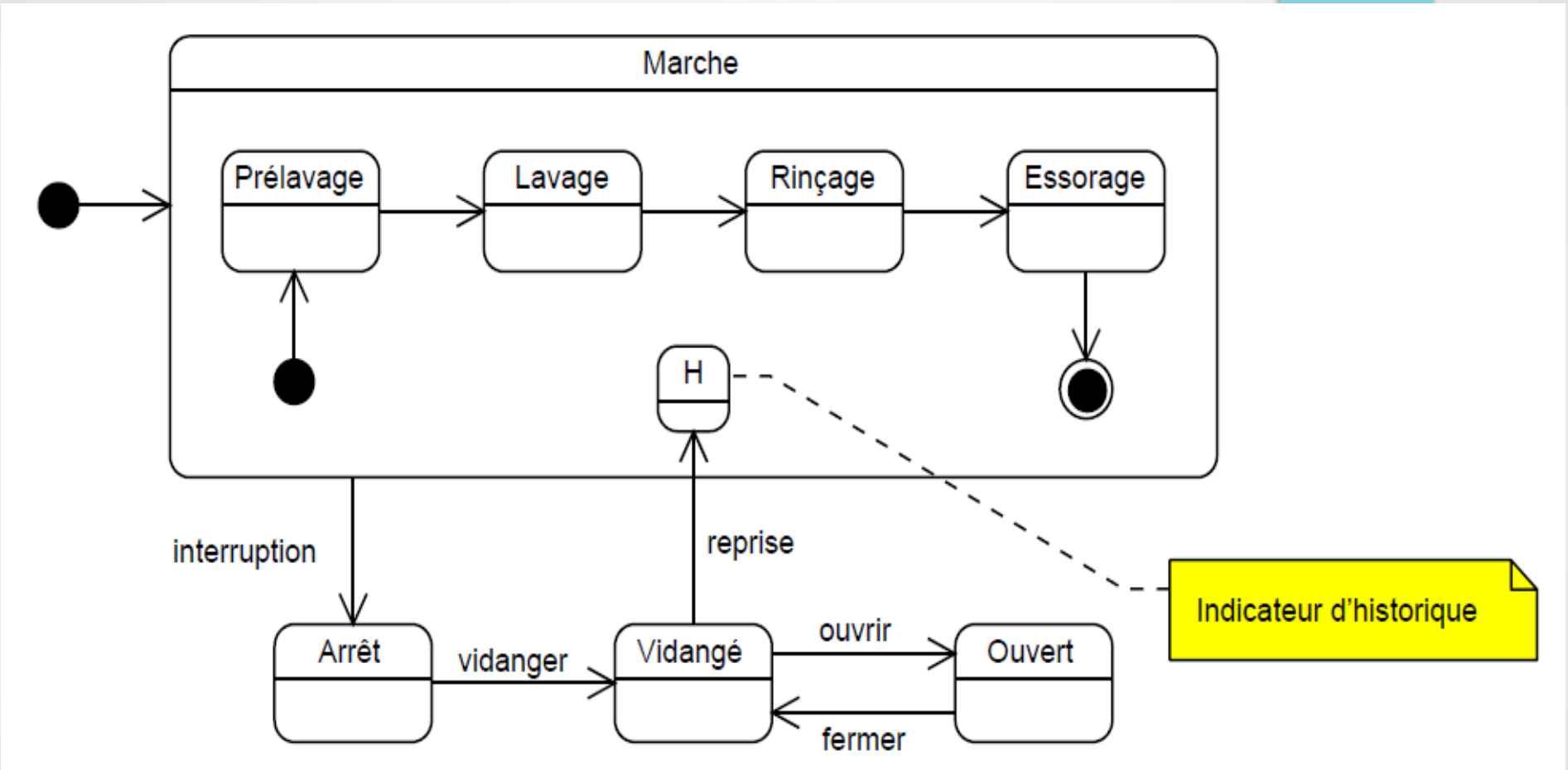
Indicateur d’historique

« Pseudo-état » qui permet de mémoriser le dernier état visité d’un sous-automate, pour y retourner ultérieurement



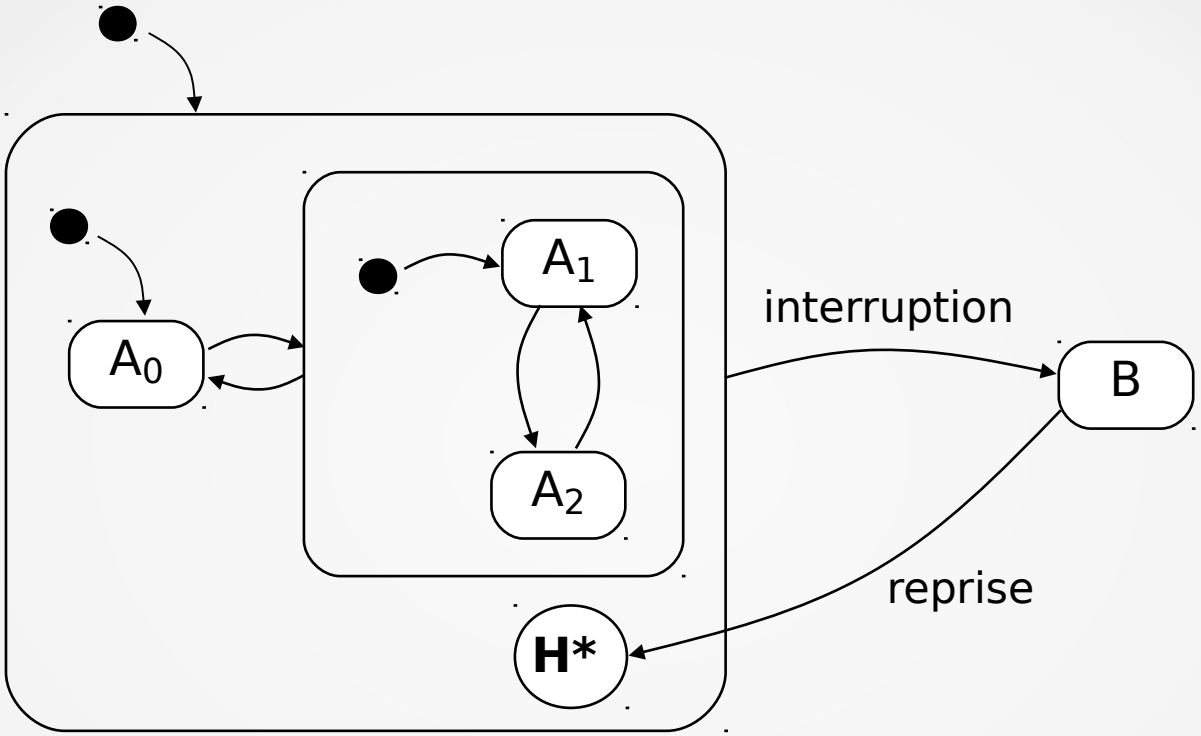
Lorsque la transition « reprise » est effectuée, l’automate reprend son exécution dans l’état où il se trouvait lorsque la transition « interruption » s’est produite, c’est-à-dire soit l’état A_1, soit l’état A_2

Exemple : machine à laver



Pour ouvrir la porte, il faut vidanger la machine. La porte refermée, la machine reprend son cycle dans l'état où elle s'était arrêtée lors de la vidange.

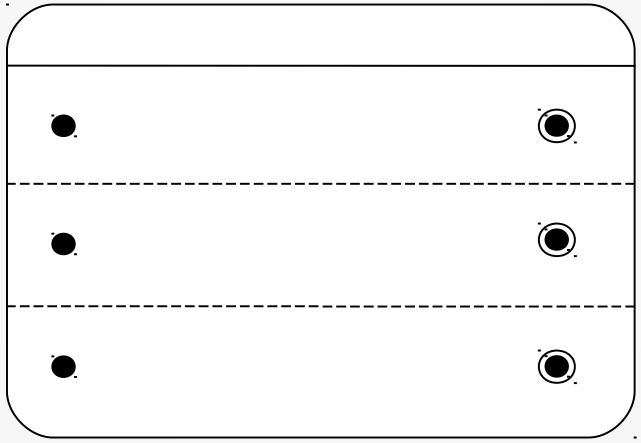
Indicateur d'historique à un niveau quelconque



Lorsque la transition « reprise » est effectuée, l'automate revient dans l'état où il était lors de la transition « interruption », *à un niveau quelconque*, autrement dit, dans l'état A_0, A_1 ou A_2.

Sous-automates en parallèle

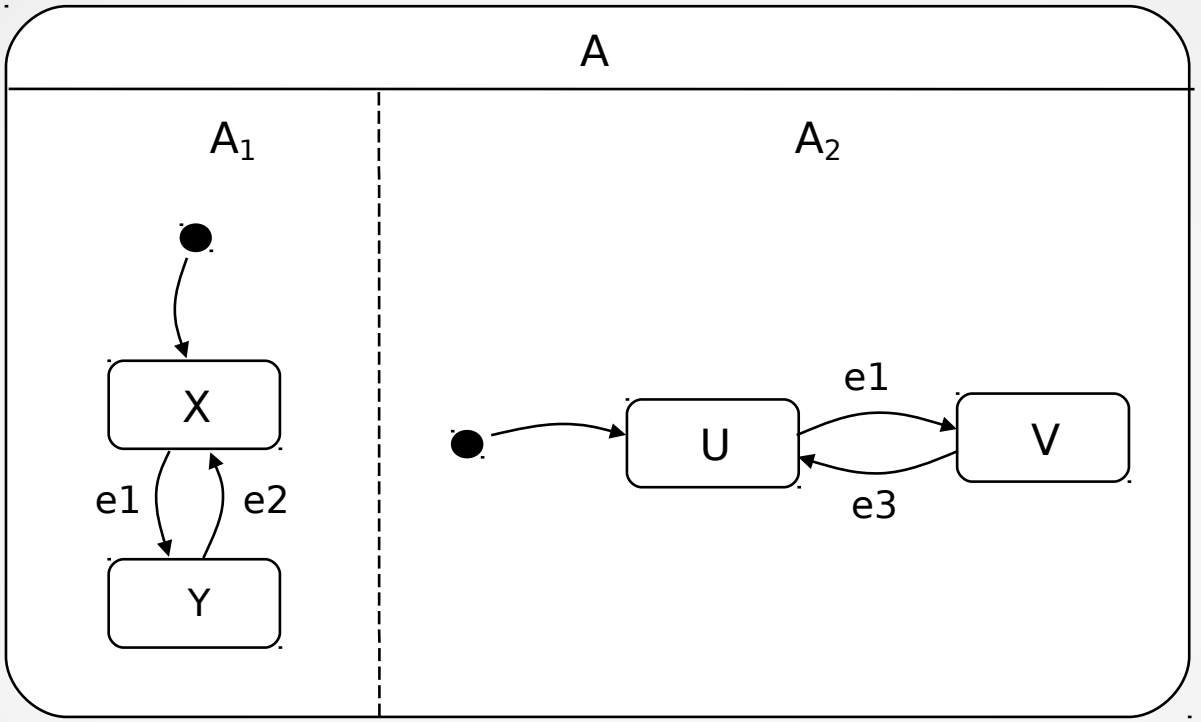
- A l’intérieur d’un état, plusieurs automates peuvent s’exécuter en parallèle. Chaque sous-automate a un état initial et peut avoir un état terminal.
- Notation



- L’activité de l’état termine lorsque tous les sous-automates parviennent à un état terminal.

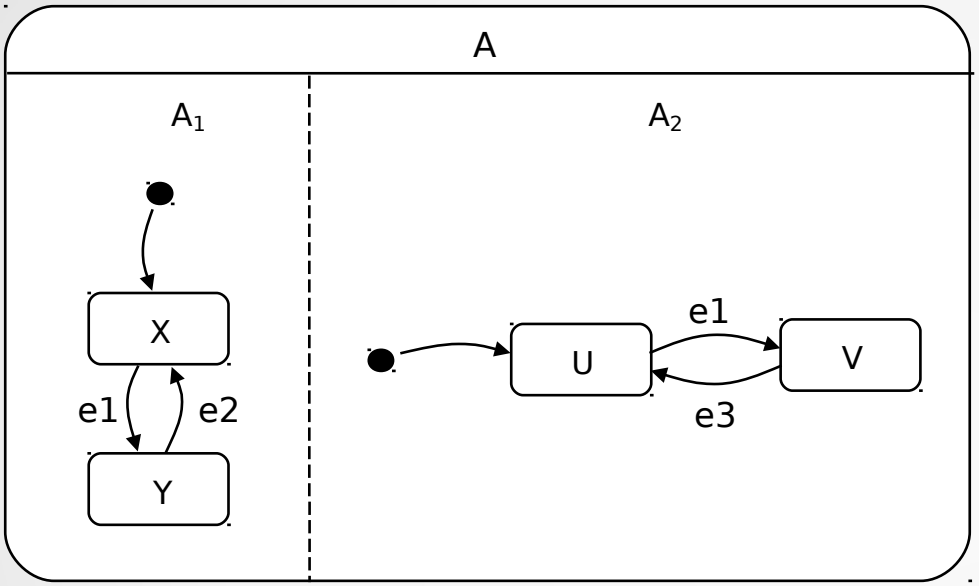
  Lorsqu’un événement se produit, toutes les transitions qui peuvent être effectuées sont effectuées.

Exemple



- Si A_1 est dans l'état X et A_2 est dans l'état U, et si l'événement e1 se produit, alors A_1 passe dans l'état Y et A_2 passe dans l'état V.
- Si A_1 est dans l'état Y et A_2 est dans l'état V, et si l'événement e2 se produit, alors A_1 passe dans l'état X et A_2 reste dans l'état V.

« Aplatissement » de sous-automates en parallèle



Automate équivalent