4. UML - Unified Modeling Language
Diagrammes dynamiques

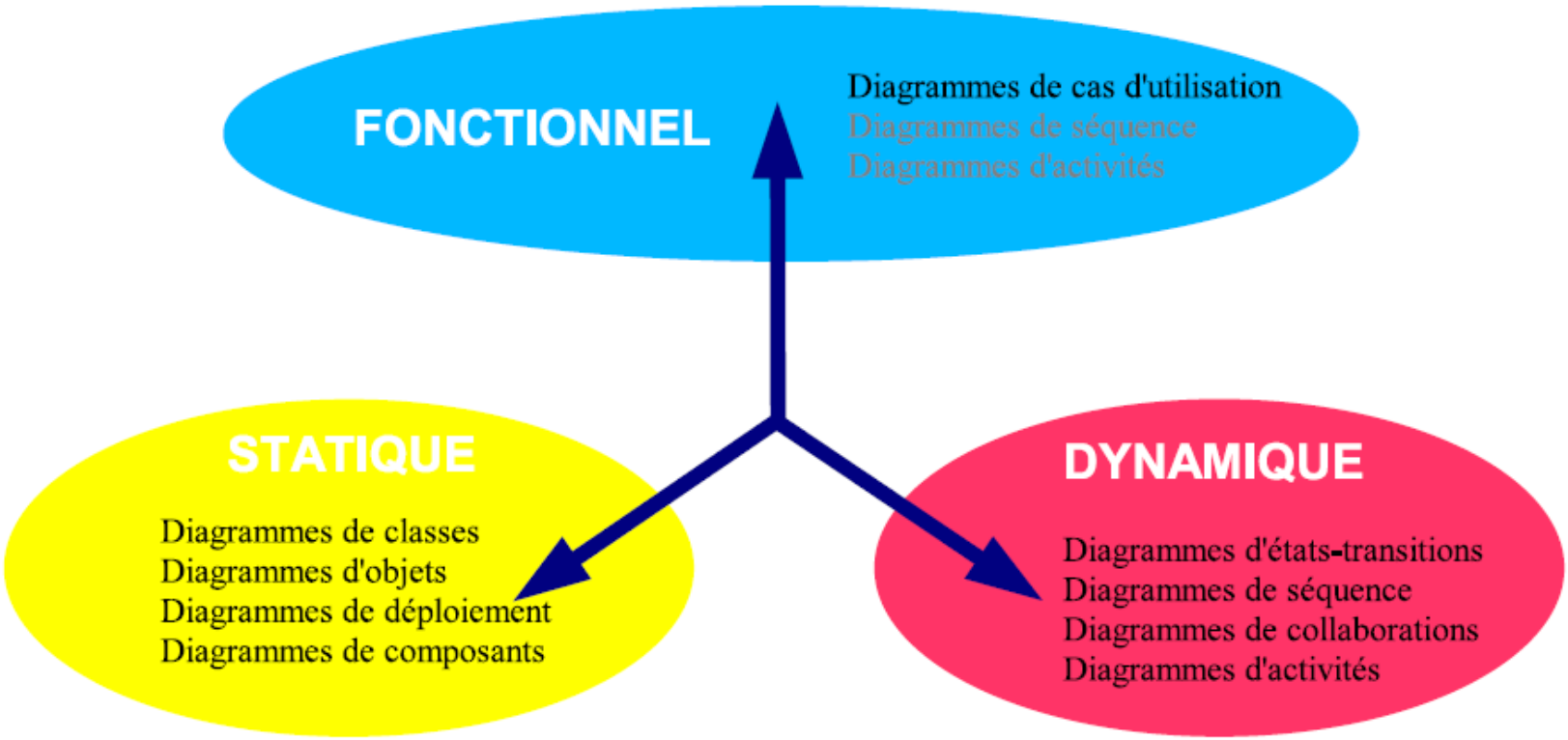
Laëtitia Matignon
laetitia.matignon@univ-lyon1.fr

Département Informatique - Polytech Lyon
Université Claude Bernard Lyon 1
2012 - 2013

Plan du cours

Plan

Diagrammes UML et axes de modélisation



Modèle dynamique

Intérêt

Le modèle dynamique montre le comportement du système, les interactions des objets et leur évolution dans le temps. Accent mis sur la chronologie.

Décrire le comportement du système

- Comment interagissent les objets pour atteindre un but ?

  → Diagrammes d'interaction : point de vue temporel sur les interactions

  - Diagrammes de communication : organisation structurelle des objets qui interagissent
  - Diagrammes de séquence : représentation temporelle/séquentielle du déroulement des interactions entre objets

Décrire l'évolution du système dans le temps

- Comment évoluent les états des objets ?

  → Diagrammes d'états-transitions : représentation du comportement d'un objet sous la forme d'un automate à états

- Modélisation de flux

  → Diagramme d'activité : variante des diagrammes d'états-transitions ; représentation du comportement d'une opération

Plan

Diagrammes d'interaction

Point de vue temporel sur les interactions

Pendant la capture des besoins

Montrer comment des objets (instances de classes) communiquent pour réaliser une certaine fonctionnalité : décrire les scénarios des cas d'utilisation

Pendant la conception

Montrer les interactions entre un jeu d'objets.

- Réfléchir à l'affectation de responsabilités aux objets
  - Qui crée les objets ?
  - Qui permet d'accéder à un objet ?
  - Quel objet reçoit un message provenant de l'IHM ?
  - ...
- Elaboration en parallèle avec les diagrammes de classes
  - Chaque événement reçu par un objet devra se traduire par une opération pour le gérer dans le diagramme de classes.

$\rightarrow$ Contrôler la cohérence des diagrammes de classes et dynamiques !

Diagrammes d'interaction
Présentation d'ensemble

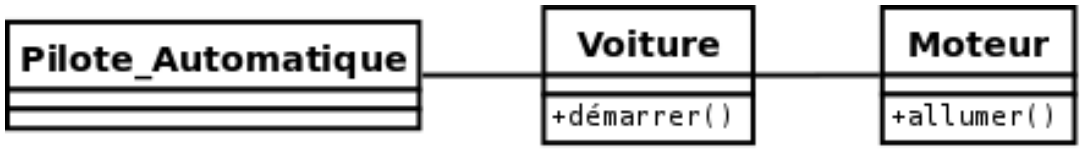


Diagramme de classes

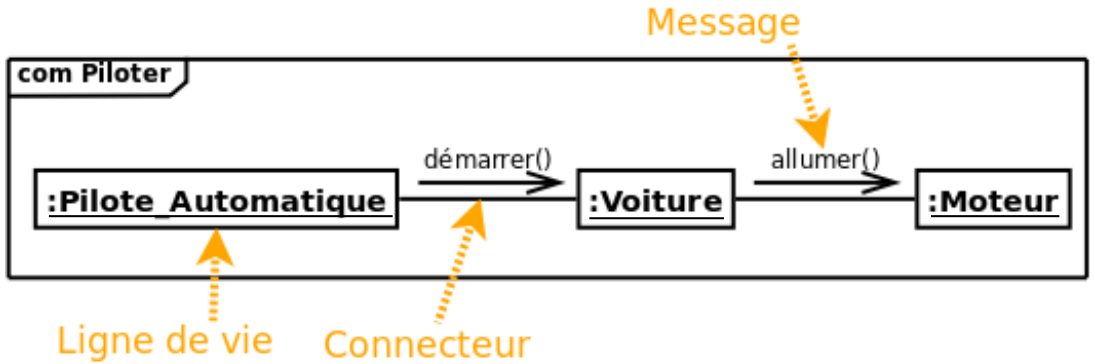


Diagramme de communication : focus sur **organisation structurelle** des objets qui communiquent.

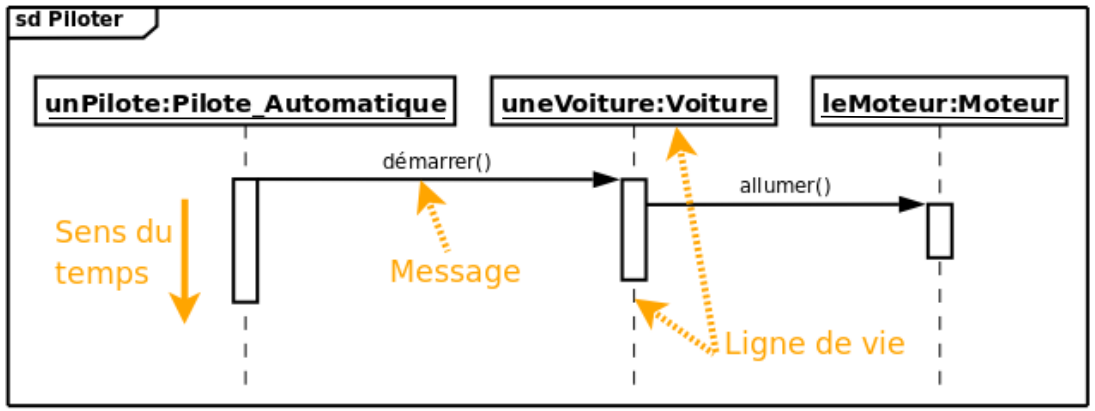


Diagramme de séquence : focus sur **chronologie d'envoi** des messages.

- Une interaction se focalise sur l'échange d'informations entre les participants.
- Elle est composée d'un jeu de lignes de vie.
- Elle spécifie les messages échangés entre participants.
- Cadres d'interaction : `sd` pour un diagramme de séquence, `com` pour un diagramme de communication

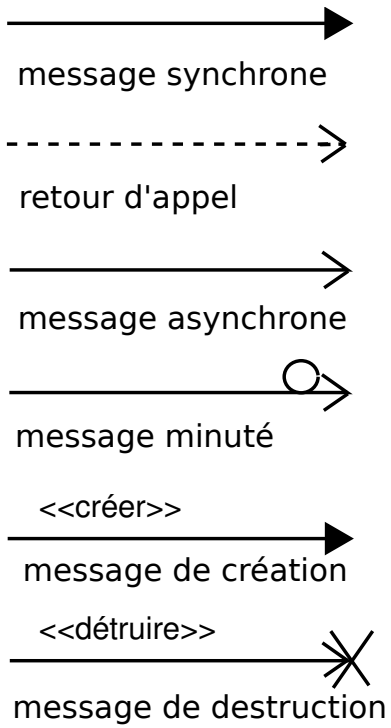
Diagrammes d'interaction
Les types de messages

- Matérialisation d'une communication, avec transmission d'information entre :
  - un émetteur (source)
  - un récepteur (destination)
- Un message peut :
  - déclencher une opération
    rôle le plus utilisé en programmation objet. Le déclenchement de l'opération peut être synchrone (cas le plus fréquent : l'émetteur reste bloqué le temps que dure l'invocation de l'opération) ou asynchrone (création d'un thread)
  - représenter l'émission d'un signal
    l'envoi d'un signal (interruption, évènement) déclenche une réaction chez le récepteur, de façon asynchrone et sans réponse : l'émetteur du signal ne reste pas bloqué le temps que le signal parvienne au récepteur. Il ne sait pas quand, ni même si le message sera traité par le destinataire
  - entrainer la création/destruction d'un objet

Diagrammes d'interaction
Les types de messages

Plusieurs types de messages (actions) peuvent transiter entre les objets :

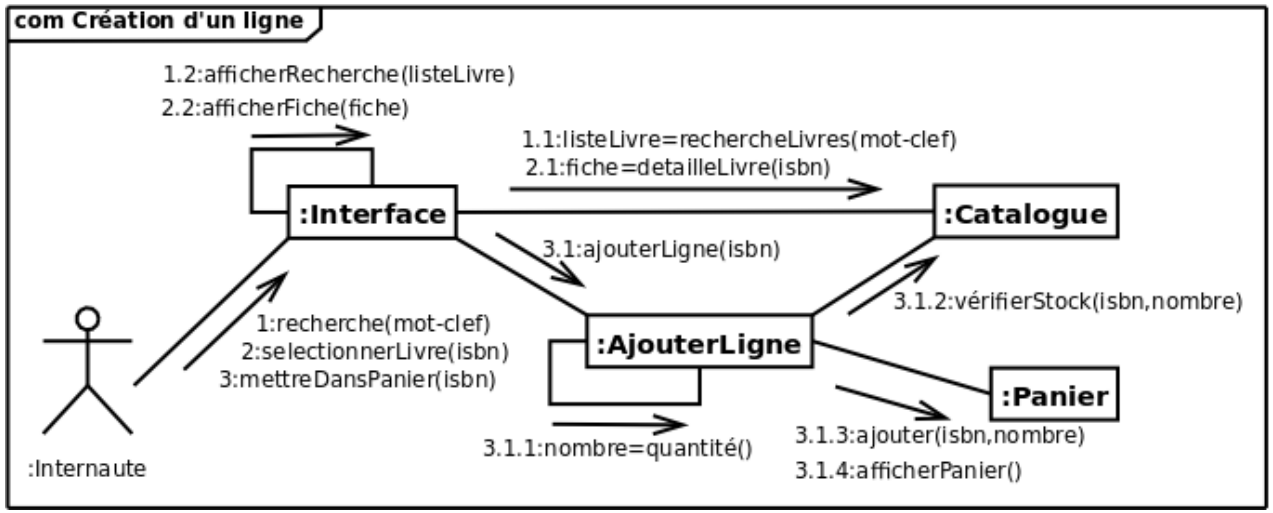
- message synchrone (standard) : l'expéditeur est bloqué pendant le traitement du message par le destinataire (appel de procédure, flôt de contrôle emboité)
- retour d'appel (optionnel)
- message asynchrone/signal : l'expéditeur continue son exécution pendant le traitement du message
- message minuté : l'expéditeur est bloqué pendant un certain temps et reprend ses activités si aucune réponse n'a lieu dans un délai prévu
- message de création d'instance
- message de destruction d'instance



Plan

Diagrammes de communication

- Diagrammes d'interaction UML2.0 (appelé diagramme de collaboration en UML1)
- Extension des diagrammes d'objets :
  - Description de la structure statique par les **liens** entre objets
  - Les liens sont les supports aux envois de messages : **ajout de la représentation des messages** échangés
- L'envoi de message déclenche une opération définie dans le récepteur avec la visibilité appropriée
- Acteurs : utilisateurs et objets



Recherche puis ajout, dans son panier virtuel, d'un livre lors d'une commande sur Internet.

Diagrammes de communication
Représentation des messages

- Dimension temporelle avec **numérotation** des messages

Syntaxe des messages

pré / [garde] num-seq iter : val-retour := msg (param)

- `pré` : prédécesseurs (synchronisation)
- `garde` : expression booléenne (condition)
- `num_seq` : numéro de séquence du message (rang)
  - Emboitement : l'envoi du message 1.3.5 suit immédiatement celui du message 1.3.4 et ces deux messages font partie du flot 1.3
  - Simultanéité : l'envoi des messages 1.3.a et 1.3.b est simultané
- `iter` : envoi séquentiel avec * ou parallèle avec $*||$, peut être suivi d'une clause d'itération entre crochets (récurrence)
- `val-retour` : valeur de retour du message
- `msg` : nom du message
- `param` : paramètres du message

Exemple : `[heure = midi] 1 : manger()`
Exemple : `1.3.6 * : ouvrir()`
Exemple : `*||[i := 1..5] 4: fermer()`

Diagrammes de communication

Représentation des messages

Syntaxe des messages

pré / [garde] num-seq iter : val-retour := msg (param)

- `pré` : prédécesseurs (synchronisation)
- `garde` : expression booléenne (condition)
- `num_seq` : numéro de séquence du message (rang)
  - Emboitement : l'envoi du message 1.3.5 suit immédiatement celui du message 1.3.4 et ces deux messages font partie du flot 1.3
  - Simultanéité : l'envoi des messages 1.3.a et 1.3.b est simultané
- `iter` : envoi séquentiel avec * ou parallèle avec $*||$, peut être suivi d'une clause d'itération entre crochets (récurrence)
- `val-retour` : valeur de retour du message
- `msg` : nom du message
- `param` : paramètres du message

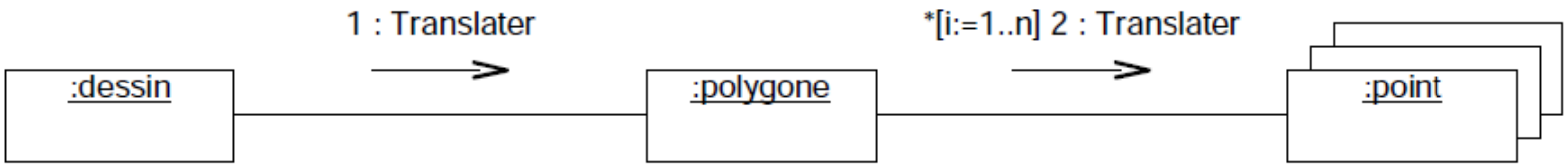
Exemple : `[t < 10s] 2.5 : age := demanderAge(nom,prenom)`

Exemple : `[disk full] 1.7.a * : deleteTempFiles()`
`[disk full] 1.7.b : reduceSwapFile(20%)`

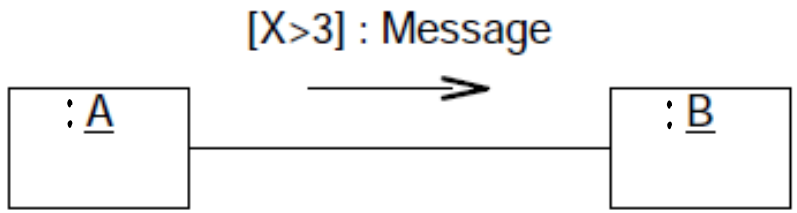
Diagrammes de communication

Représentation des messages

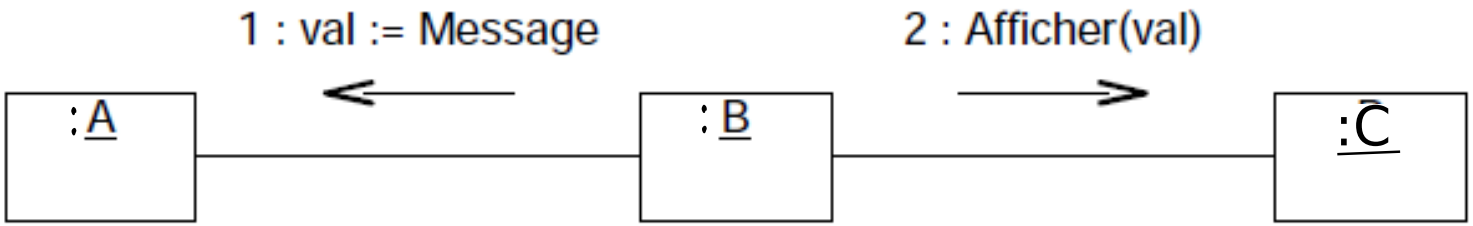
- Itérations



- Messages conditionnels



- Résultats et paramètres (ou arguments)



Diagrammes de communication

- Représentation des interactions (échanges de messages) entre objets selon un point de vue spatial
- Centré sur l'organisation structurelle des objets qui communiquent
- Visualisation claire d'un nombre limité d'information
- Vues généralement partielles et focalisées sur une collaboration de quelques objets
- Souvent utilisé pour illustrer un scénario de cas d'utilisation ou pour décrire une opération

Exercice

Exercice 1

Décrire à l'aide d'un diagramme de communication l'enchaînement suivant :

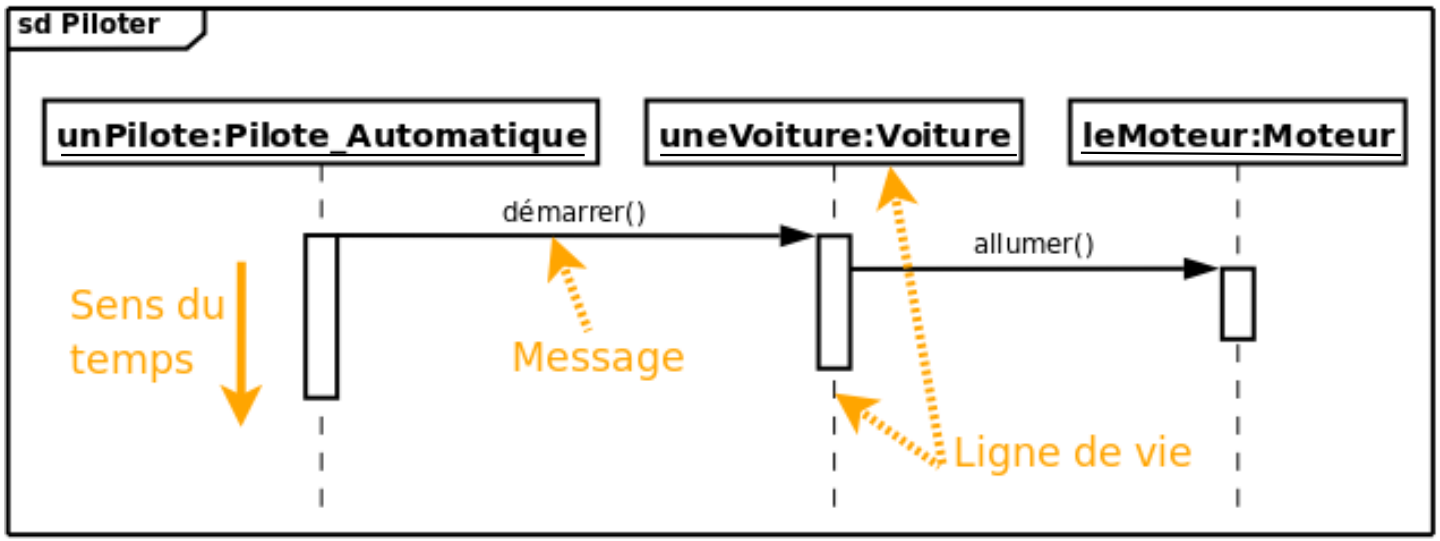
- L'avion Boeing pa87, en état de détresse, envoie plusieurs messages de détresse (*alerte*) les uns à la suite des autres à la tour de contrôle.
- La tour de contrôle décide de faire atterir cet avion sur une piste *pisteA*. Elle spécifie simultanément aux avions situées sur cette piste de se déplacer vers une voie de garage.
- Puis elle donne l'autorisation d'atterrir sur cette piste à l'avion en détresse et envoie les pompiers sur cette piste (cela se fait en même temps).

Exercice 1

Plan

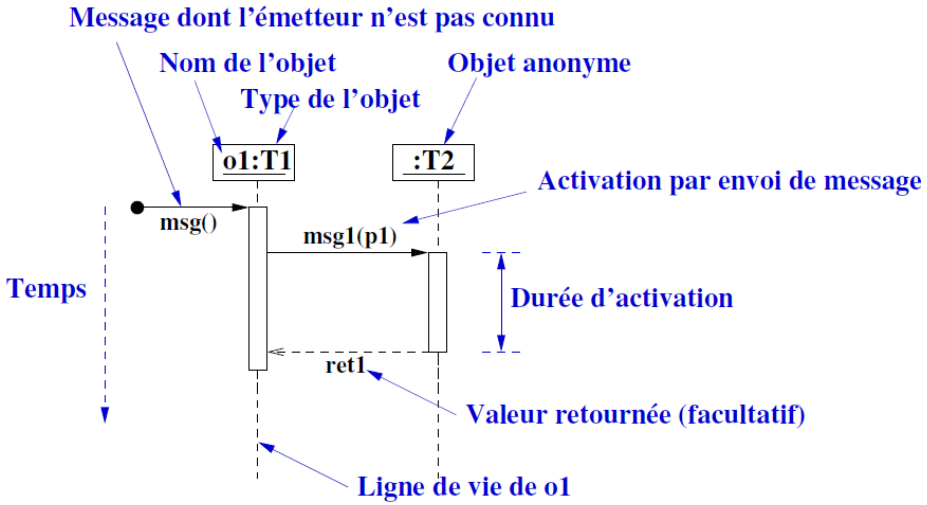
Diagrammes de séquences

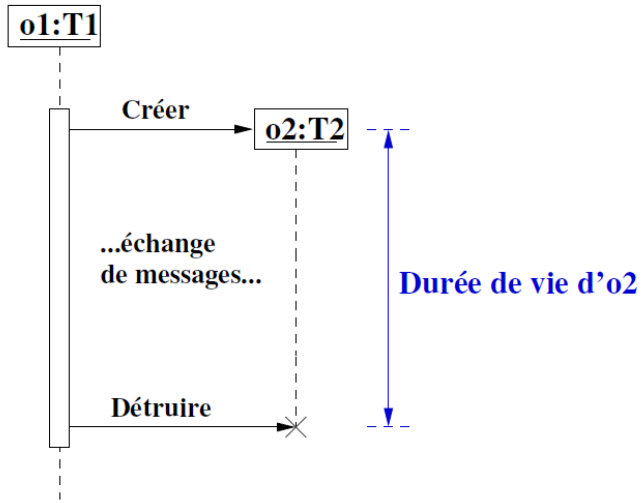
- Représentent les interactions (échanges de messages) entre objets selon un point de vue temporel, un ordre chronologique
- Représentation des **scénarios** des cas d'utilisation
- Diagrammes de séquence système : documentation des cas d'utilisation



Diagrammes de séquences

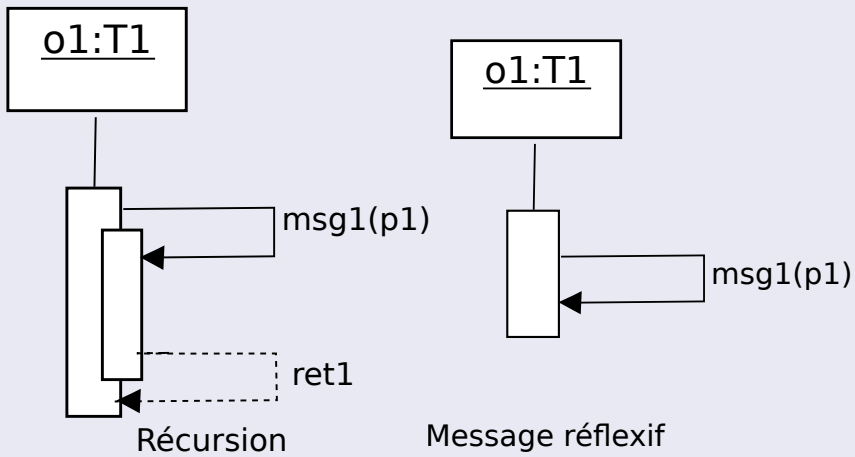
- Axe vertical : représentation implicite du temps, non gradué
- Emplacement de l'acteur : Si l'acteur existe dès le début du scénario, en haut ; Sinon, à l'extrémité du message donnant lieu à sa création
- Fin de l'acteur : Aucune si l'acteur n'est pas détruit ; Sinon, une croix là ou l'objet est détruit
- Durant sa vie, un acteur peut être actif ou inactif : Un acteur actif effectue une opération ou attend le retour d'un envoi de message en mode synchrone (ex : appel de fonction)





Diagrammes de séquences

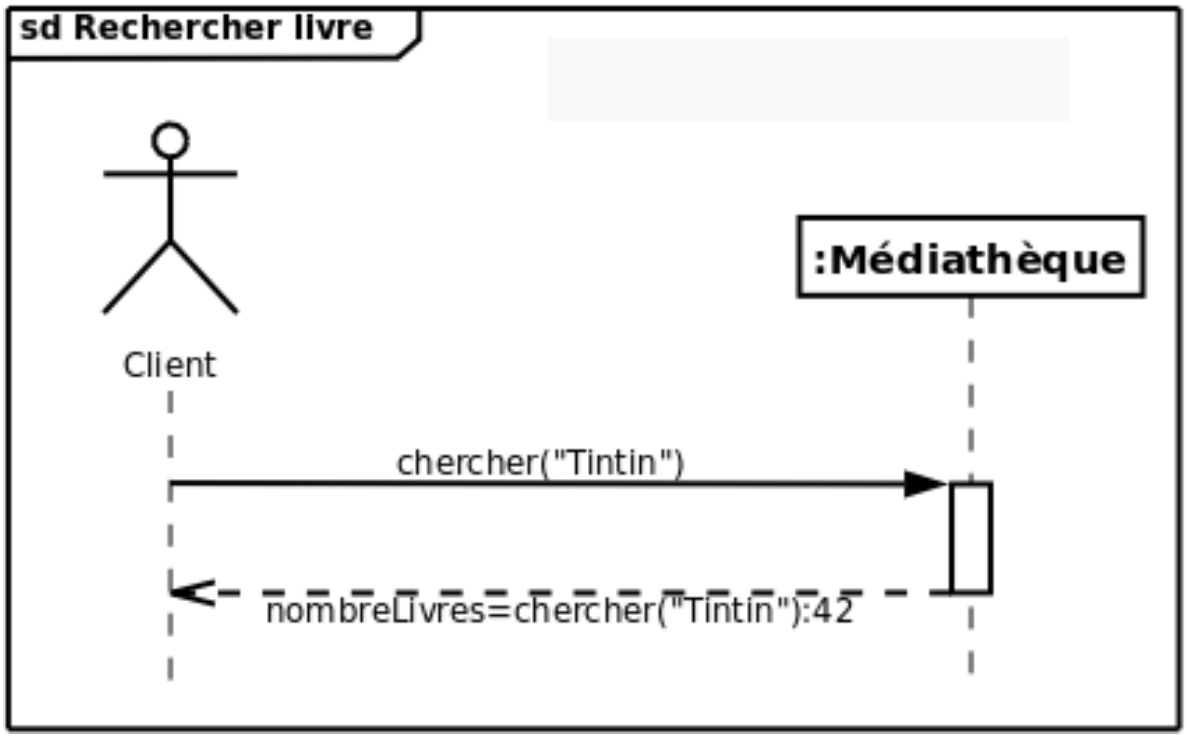
Messages réflexifs et récursifs



Diagrammes de séquences

Syntaxe des messages

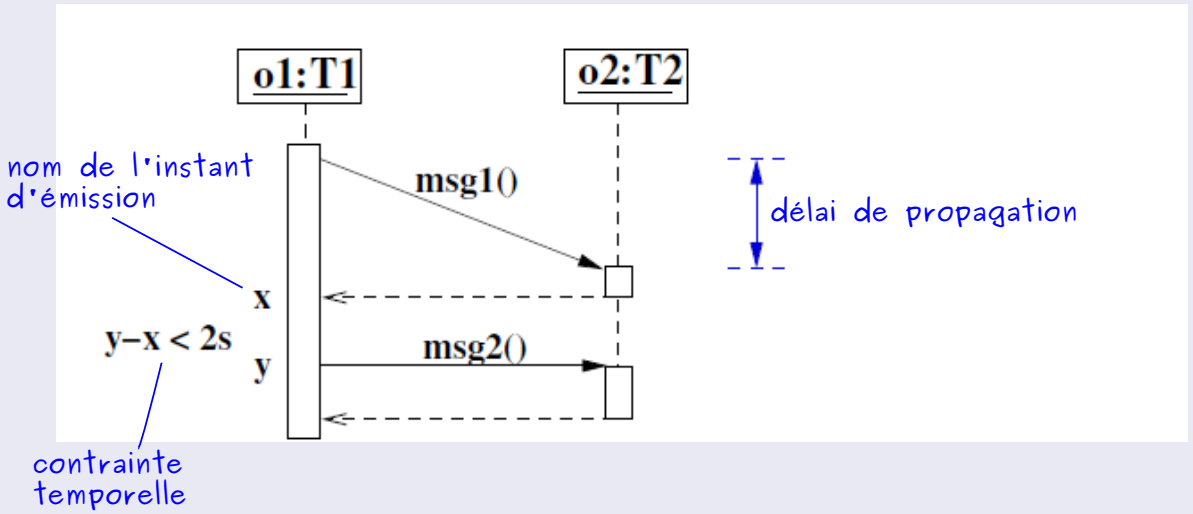
- Identique à celle des diagrammes de communication (numéro de séquence omis)
- Réponse à message : `attributRetourné = MessageDEnvoi : ValeurRetour`



Diagrammes de séquences

- Flèches horizontales = l'envoi d'un message est considéré comme instantané (le temps de transfert n'est pas pris en compte)
- Les messages asynchrones peuvent être reçus dans un ordre différent de l'ordre d'envoi

Contraintes temporelles

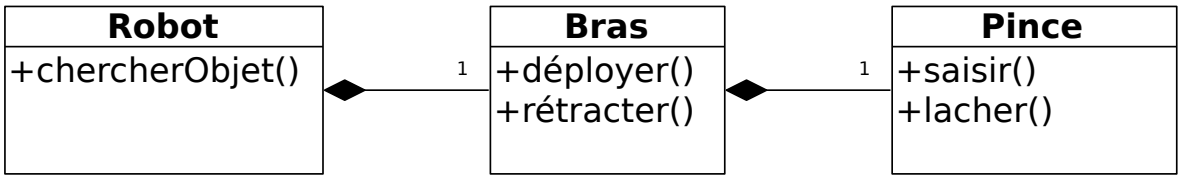


Exercice

Nono le petit robot

La situation est celle d'un bras articulé (ou petit robot) capable de déployer ou de rétracter son bras et d'ouvrir ou de fermer sa pince pour aller chercher des objets lorsque l'ordre lui en est donné. Le diagramme de classe ci-dessous "modélise" ce robot.

- Illustrer par un diagramme de communication le scénario suivant :
  - L'ordre est envoyé au robot d'aller chercher un objet
  - le robot déploie son bras
  - le robot saisit l'objet avec sa pince
  - le robot rétracte son bras
  - le robot lâche l'objet
- Proposer maintenant un diagramme de séquence équivalent.



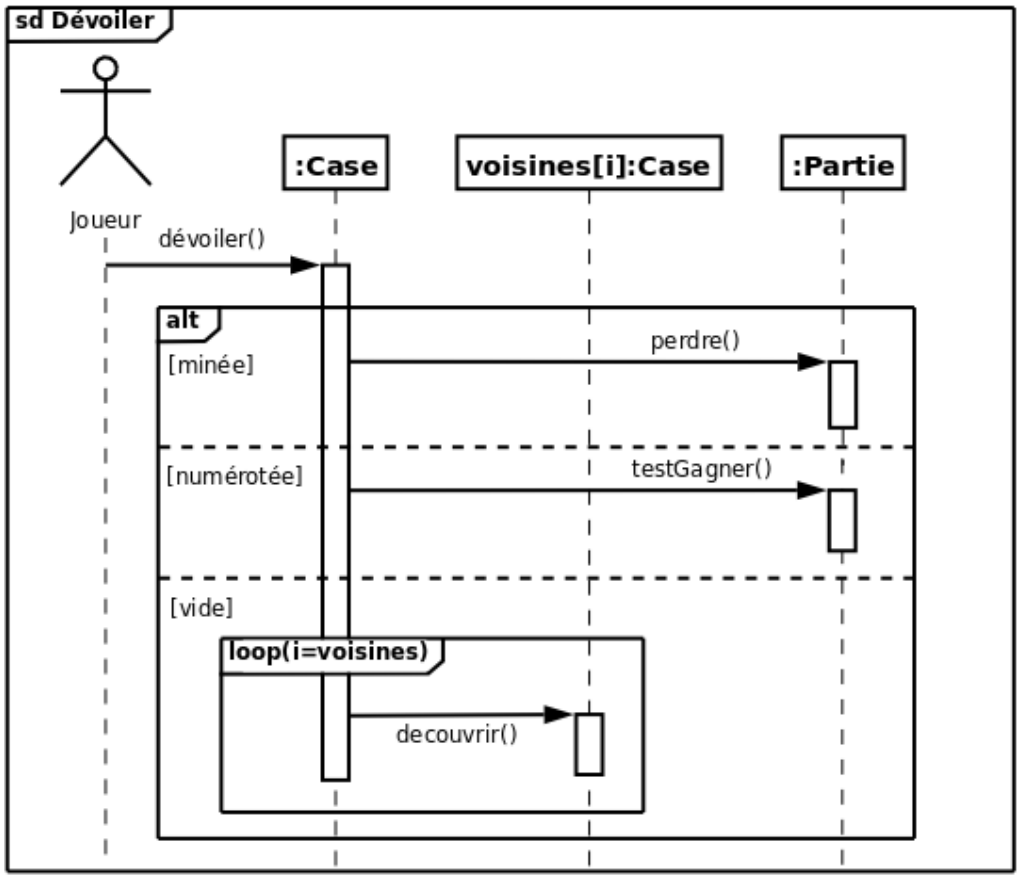
Fragments d'interaction combinés

- Permettent de décomposer une interaction en fragments simples : articulation d'interactions
- Entre autres, représentation graphique des tests et des boucles des langages de programmation
- Font intervenir tous ou un sous-ensemble des participants au scénario
- Représentation :
  - Boîte, englobant TOUTES les lignes de vies des acteurs concernés
  - Etiquette dans le coin supérieur gauche, contenant le type de combinaison : **opérateur d'interaction**
  - Eventuellement, opérandes avec conditions de choix entre [] et séparés par une ligne horizontale pointillée
- Regroupement des opérateurs d'interaction par fonctions :
  - choix et boucle : `alt, option, break, loop`
  - contrôle de l'envoi de messages en parallèle : `parallel, critical region`
  - contrôle de l'envoi de messages : `ignore, consider, assert, negative`
  - ordre d'envoi des messages : `weak sequencing, strict sequencing`

Fragments d'interaction combinés

Choix

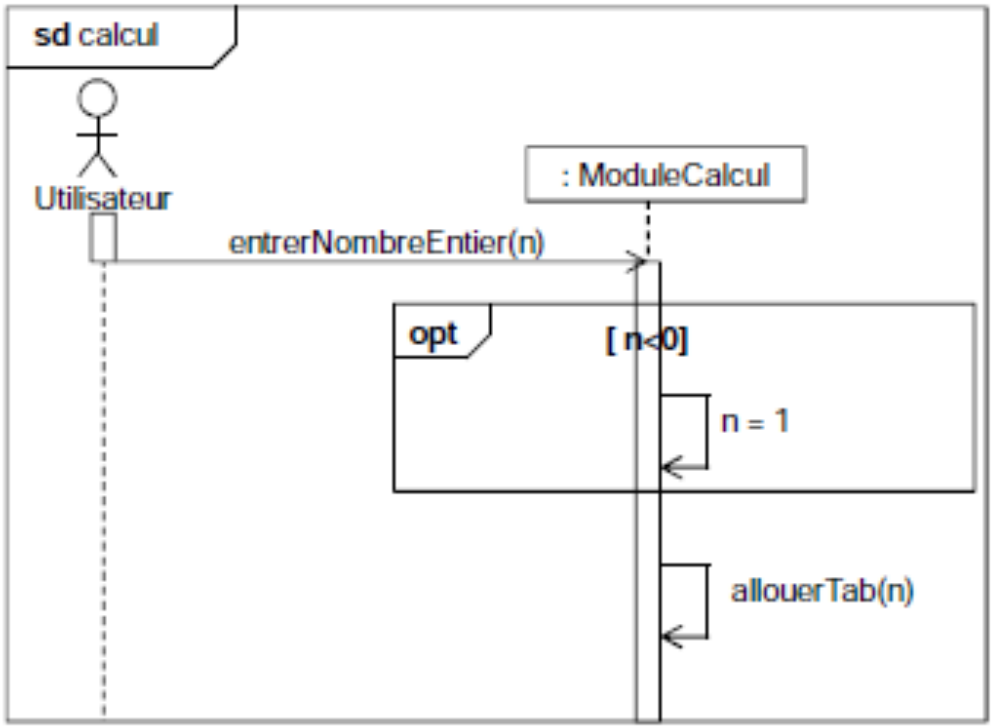
- Alternative : `alt`
  - Equivalent à un `switch`
  - Condition booléenne entre crochets [ ]
  - Si 2 opérandes, équivalent à `if (...) bloc1 else bloc2` (2nd opérande `[else]`)



Fragments d'interaction combinés

Choix

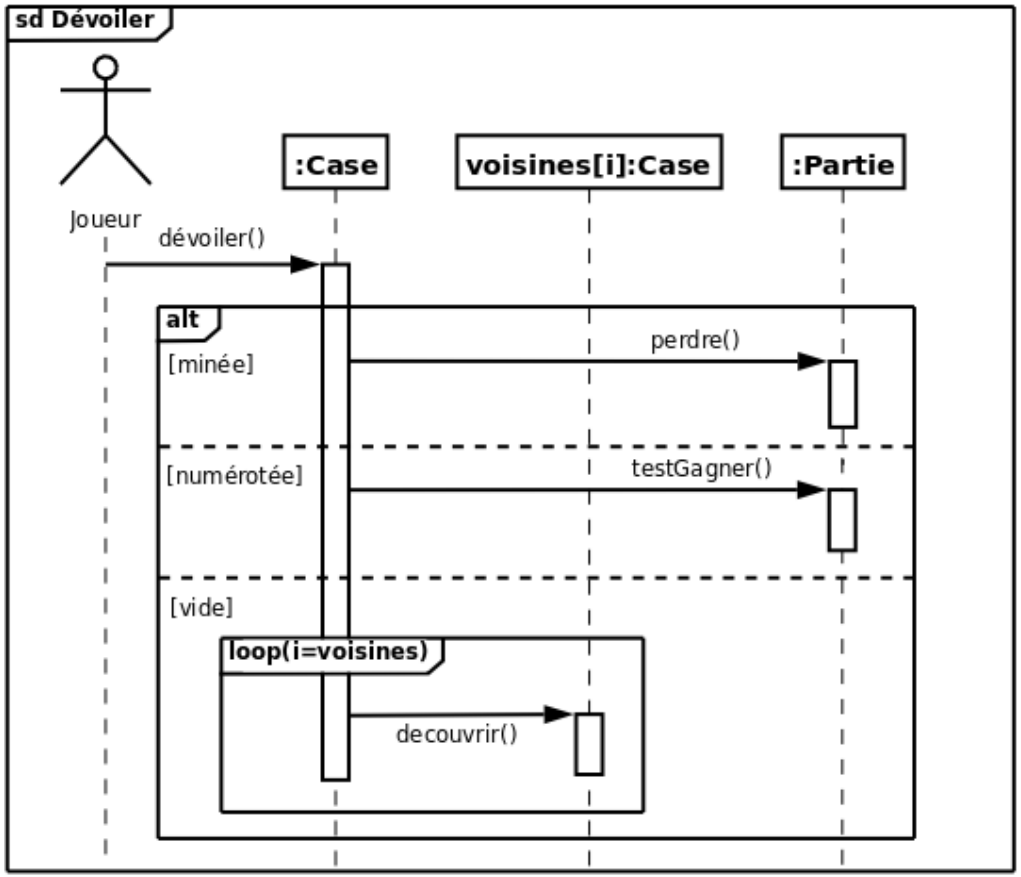
- Option : `opt`
  - Equivalent à un `if (...) bloc1`
  - Condition booléenne entre crochets [ ]
  - Un seul opérande



Fragments d'interaction combinés

Boucle

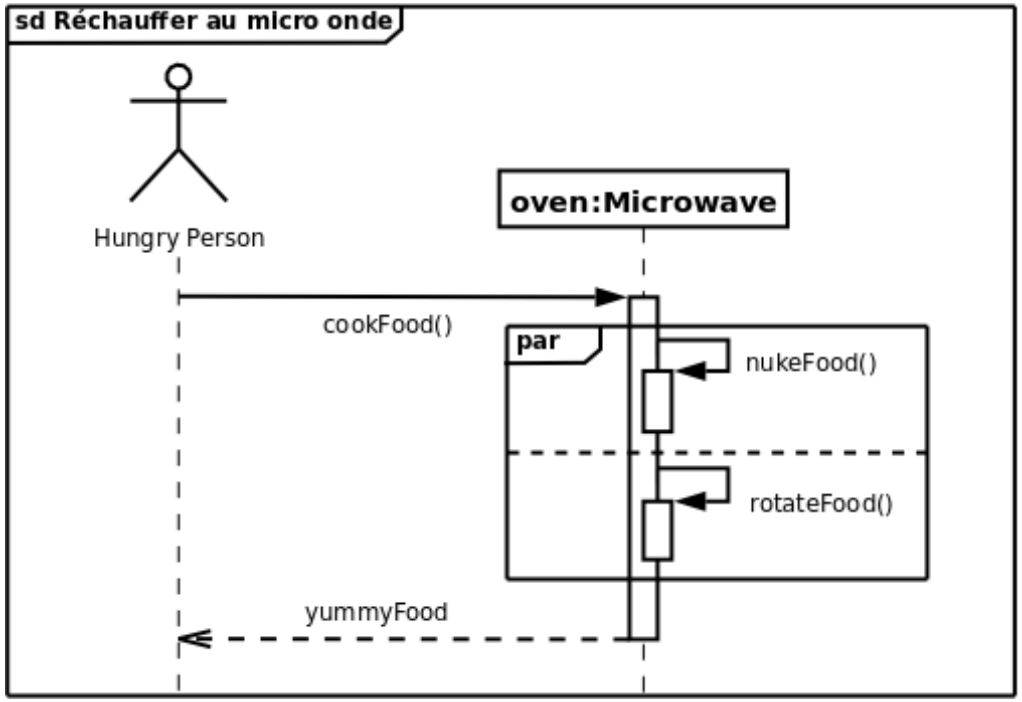
- Boucle : `loop`
  - `loop(condition)`



Fragments d'interaction combinés

Parallélisation

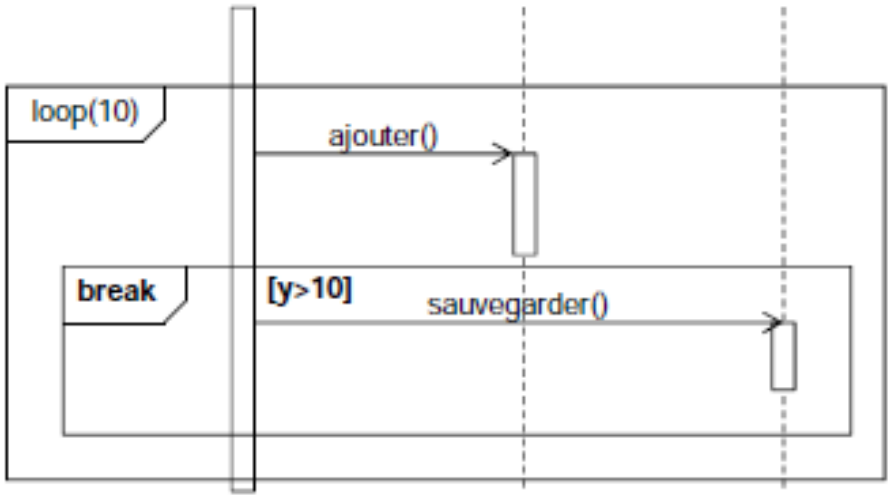
- Parallélisation : `par`
  - Envoi de messages en parallèle. Les opérandes du fragment se déroulent en parallèle.
  - Au moins deux sous-fragments exécutés simultanément



Fragments d'interaction combinés

Interruption

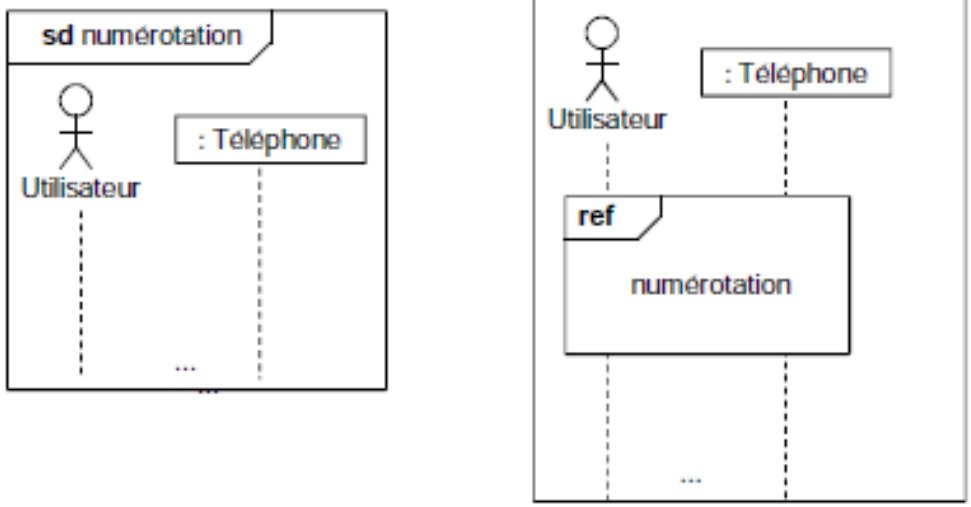
- Interruption : `break`
  - Avec condition : exécuté lorsque la condition est VRAIE. Le reste du fragment d'interaction contenant (ex : loop) est ignoré



Fragments d'interaction combinés

Référencement, réutilisation

- Référencement, réutilisation : `ref`
  - "Appel" à une interaction décrite dans un autre diagramme de séquence existant



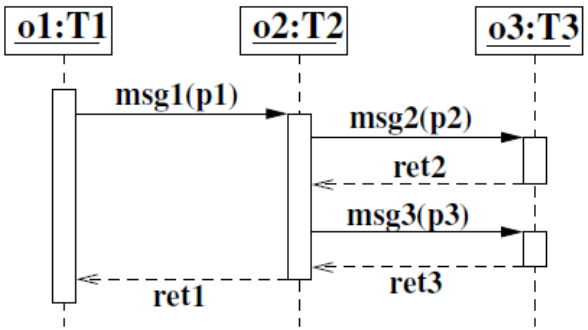
Diagrammes de séquence vs diagrammes de communication

Deux vues différentes mais logiquement équivalentes

Diagrammes de séquence

Structuration en termes de

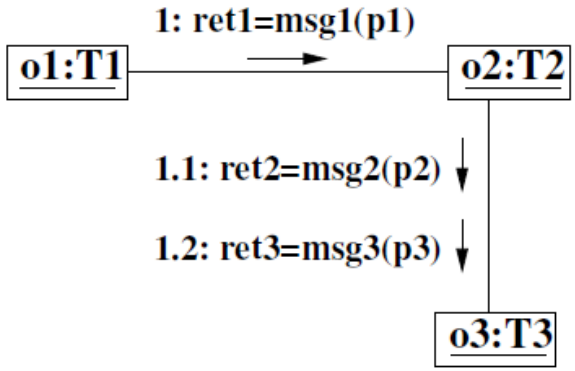
- temps : axe vertical
- objets : axe horizontal



Diagrammes de communication

Structuration en multigraphe

- Numérotation des arcs pour modéliser l'ordre des interactions



Exercice Distributeur Automatique de Billets

Réalisez un diagramme de séquence système (sd simplifié avec évènements et pas envoi de messages) qui décrit le scénario nominal du cas d'utilisation RETIRER DE L'ARGENT.

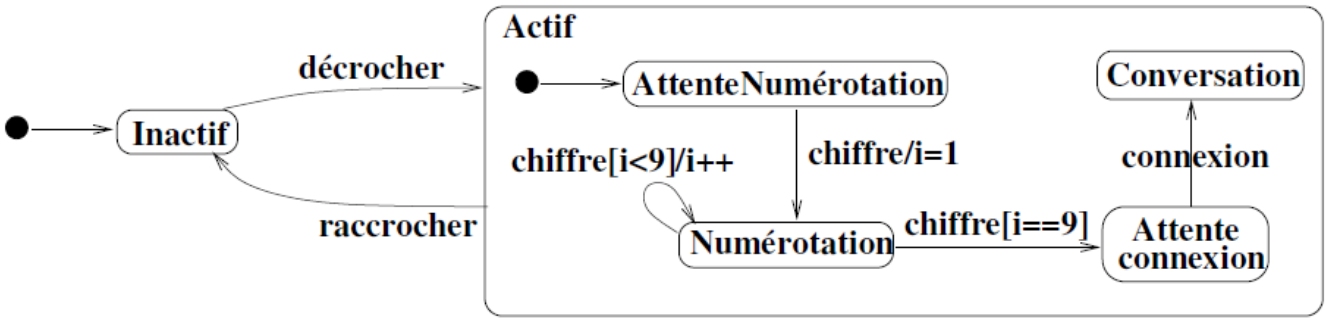
1. Le porteur de CB introduit sa carte dans le lecteur du DAB	2. Appel du cas 'S'authentifier' 3. Le DAB demande une autorisation au SI gestion CB.
4. Le SI gestion CB donne son accord et indique le solde hebdomadaire (réponse doit être < 2s).	5. Le DAB demande au porteur de CB de saisir le montant désiré du retrait.
6. Le porteur de CB saisit le montant.	7. Le DAB contrôle le montant par rapport au solde hebdomadaire. 8. Le DAB demande au porteur de carte s'il veut un ticket.
9. Le porteur de CB veut un ticket.	10. Le DAB éjecte la carte.
11. Le porteur de CB reprend sa carte.	12. Le DAB délivre le ticket et les billets.
13. Le porteur de CB prend son ticket et ses billets.	

Exercice Distributeur Automatique de Billets

Plan

Diagrammes d'états-transitions

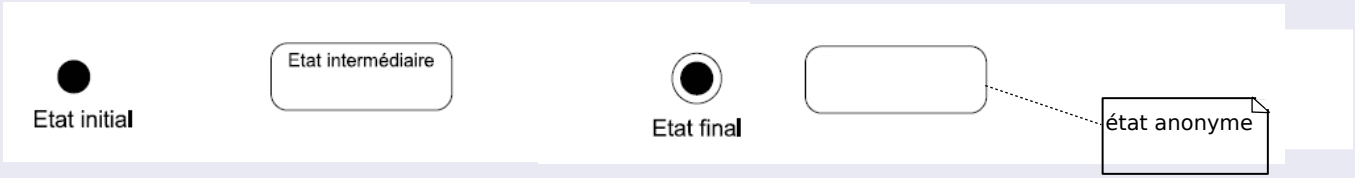
- Permettent de modéliser le **comportement interne** d'un objet à l'aide d'un automate à états finis déterministes
- Automates à états finis = graphes d'états, reliés par des arcs orientés qui décrivent les transitions.
- Vision complète et non-ambigüe de l'ensemble des comportements de l'élément auquel il est attaché
- Associé à un seul classeur
- Utilisés pour modéliser le cycle de vie des objets :
  - Représentation des changements d'états d'un objet
  - En réponse à des évènements (interactions avec d'autres objets ou avec des acteurs)



Les états

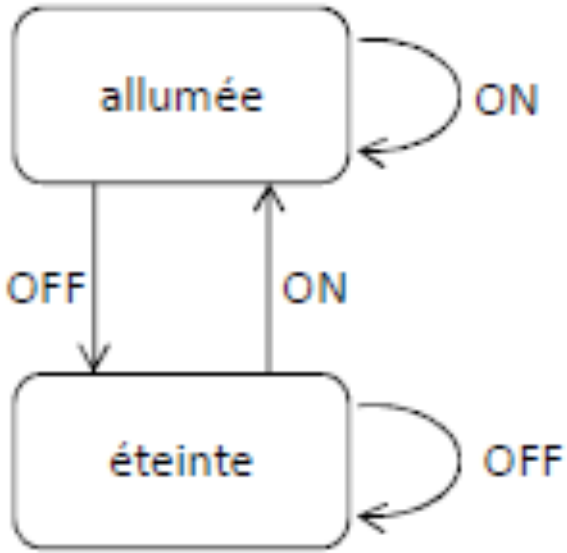
- Un état représente une période dans la vie d'un objet pendant laquelle ce dernier :
  - accomplit une certaine activité ;
  - ou attend un certain événement.
- Un état se caractérise par sa durée : finie, variable selon la vie de l'objet, en particulier en fonction des événements qui lui arrivent
- Un état représente une conjonction instantanée des valeurs des attributs d'un objet.
- Le cycle de vie d'un objet correspond à une **succession d'état**s
- Deux pseudo-états : état initial (création de l'instance) et état final (destruction de l'instance)
- → Identifier/Conserver les états significatifs

Représentation des états



Les états

- Exemple simple : *considérons une lampe munie de deux boutons-poussoirs ON et OFF. L'effet de l'appui sur les boutons dépend naturellement de l'état de la lampe.*



Les événements

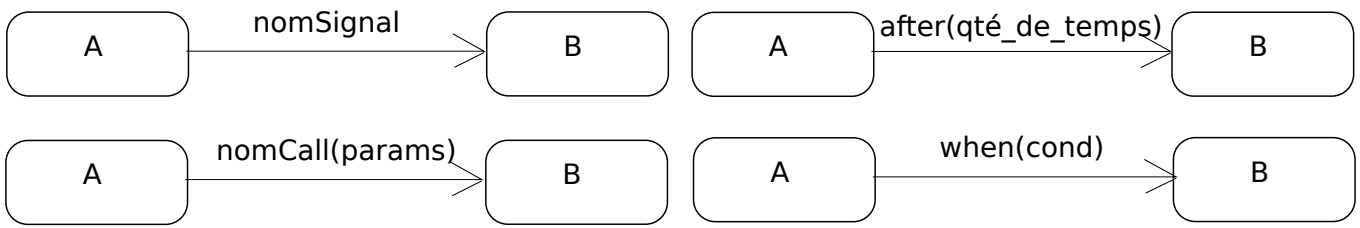
Un objet est considéré comme un élément passif contrôlé par les événements du système.

- Un événement spécifie qu'il s'est passé quelque chose de **significatif**, localisé dans le temps et dans l'espace
- Un événement se produit à un instant précis et est dépourvu de durée
- Un événement peut déclencher une transition entre états et faire basculer l'objet dans un nouvel état

Les événements

Différentes sortes d'événements

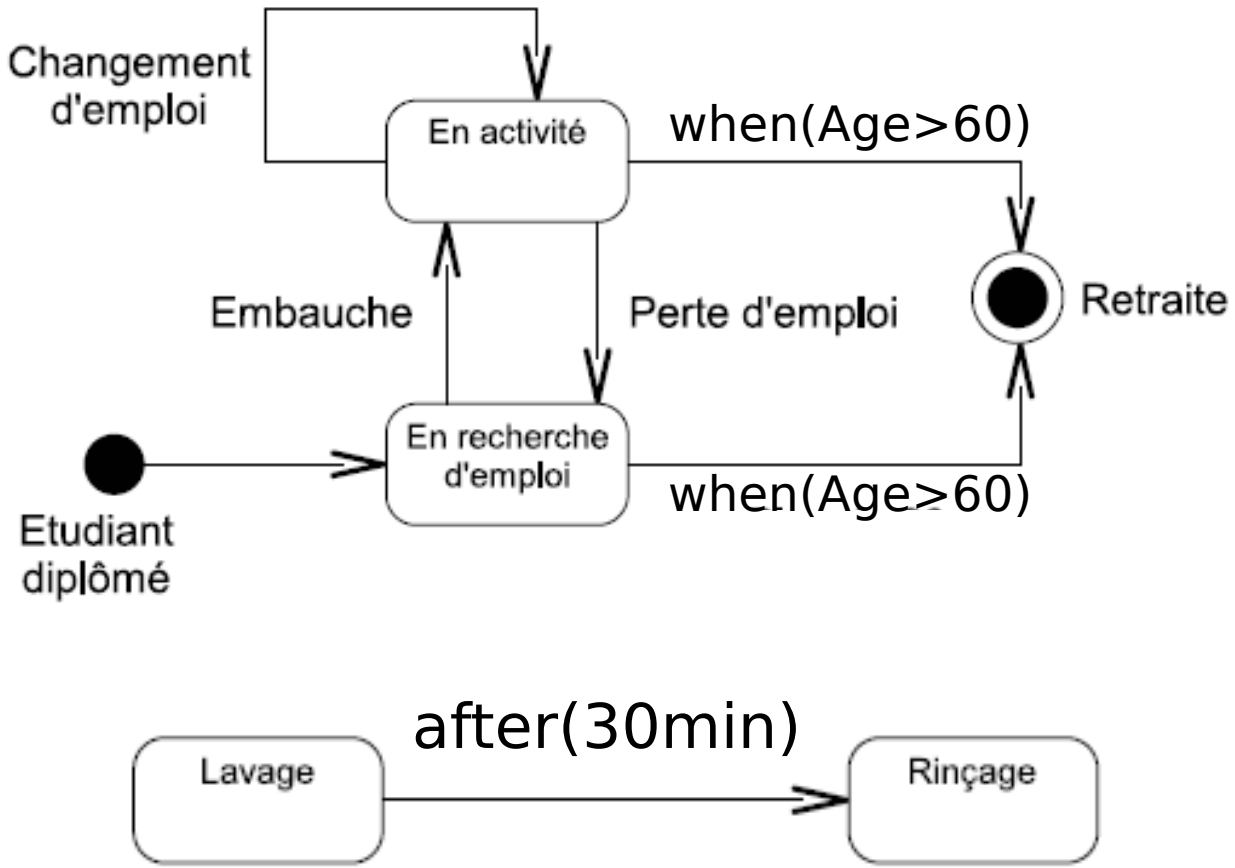
- *signal* réception d'un signal asynchrone, explicitement émis par un autre objet
- *call* appel de méthode sur l'objet courant (méthode déclarée dans le diagramme de classe).
- *after* causé par l'expiration d'une temporisation
- *change* causé par la satisfaction d'une condition booléenne
- *completion event* fin d'une activité liée à un état, de type do/ (déclenchement d'une transition "automatique", sans évènement déclencheur explicite)



Les transitions

- Une transition lie deux états (ou réflexif) et indique que l'objet change d'état
- Elle représente le passage instantané d'un état (source) vers un autre (cible)
- **Déclenchée par un événement** spécifié au-dessus de la transition (automatique si on ne spécifie pas l'événement qui la déclenche).
- Une transition n'a pas de durée

Les transitions : Exemple



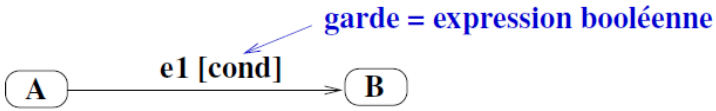
Transitions et événements

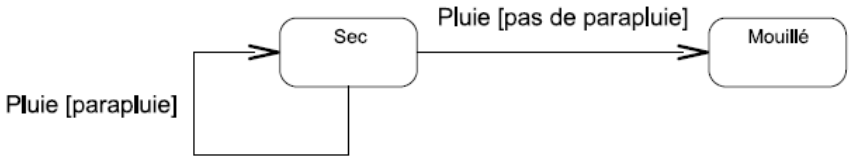
- En plus de spécifier un événement précis déclenchant la condition, il est aussi possible de conditionner une transition, à l'aide de gardes : conditions booléennes entre []
- La syntaxe d'une transition est donc la suivante :

- Lorsqu'une transition se déclenche, son effet s'exécute

Transitions et événements

Gardes (événements conditionnels)





Transition de A vers B réalisée si `cond` est vraie quand `e1` arrive
→ Si `cond` est fausse alors `e1` est perdu

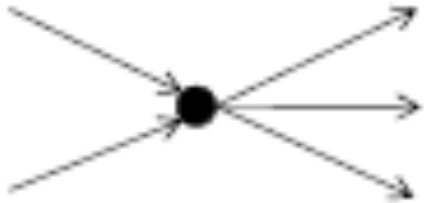
Condition de garde vs événement de changement

- Condition de garde : évaluée une fois que l'événement déclencheur de la transition a lieu et que le destinataire le traite. Si elle est fausse, la transition ne se déclenche pas et la condition n'est pas réévaluée.
- Évènement de changement : évalué continuellement jusqu'à ce qu'il devienne vrai, et c'est à ce moment-là que la transition se déclenche.

- Un événement est par convention instantané/atomique
  → Incorrect de tester sa durée ! Utiliser événement temporel ou activité finie

Transitions composites

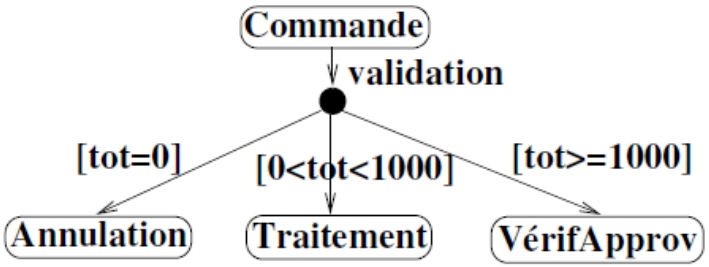
- Représenter des alternatives pour le franchissement d'une transition
- Utilisation de deux pseudo-états particuliers :
  - point de jonction

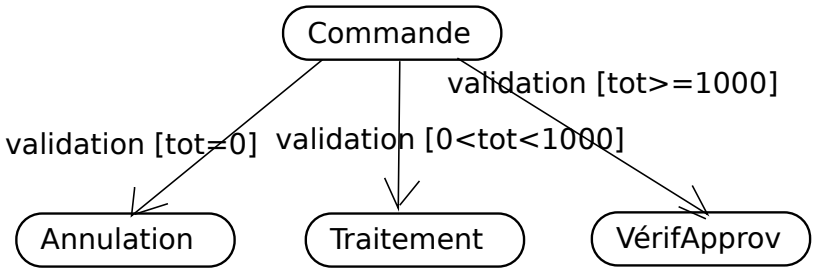


  - point de choix

Transitions composites
Point de jonction

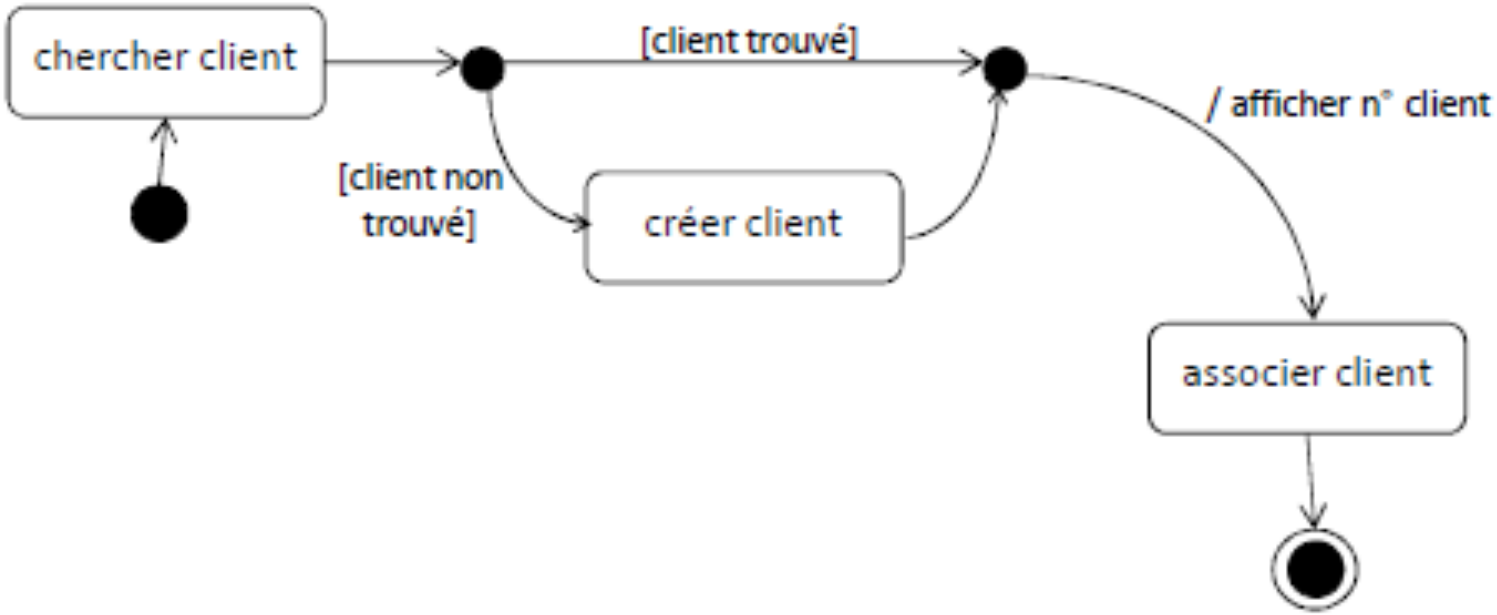
- Factorisation de l'événement déclencheur (ex. : `validation`)
- Les gardes doivent être mutuellement exclusives pour que l'automate soit déterministe
- **Point de jonction est statique** : gardes après le point de jonction évaluées avant que la transition soit empruntée
- Ces deux représentations sont équivalentes :





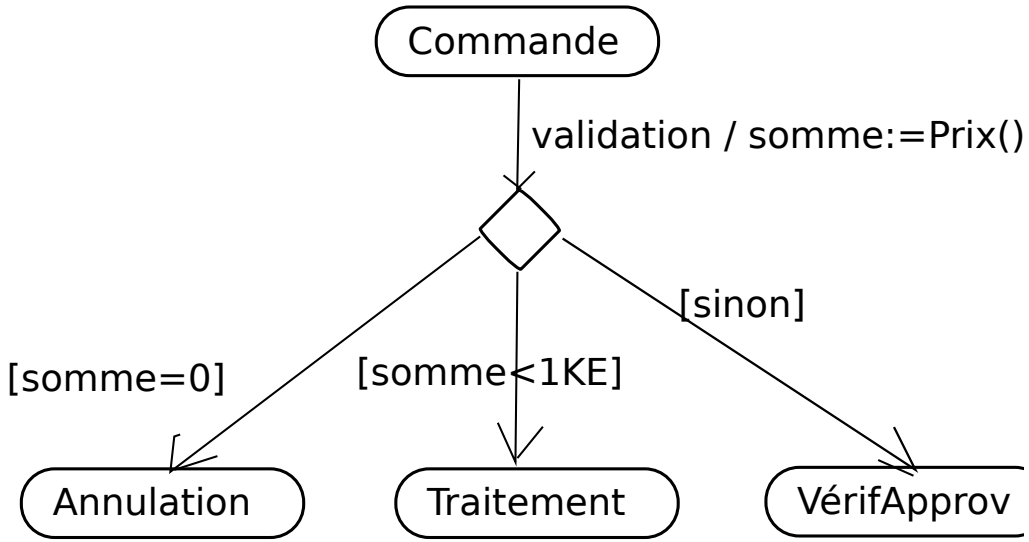
Transitions composites
Point de jonction

- Bien adapté pour représenter des chemins optionnels `if`



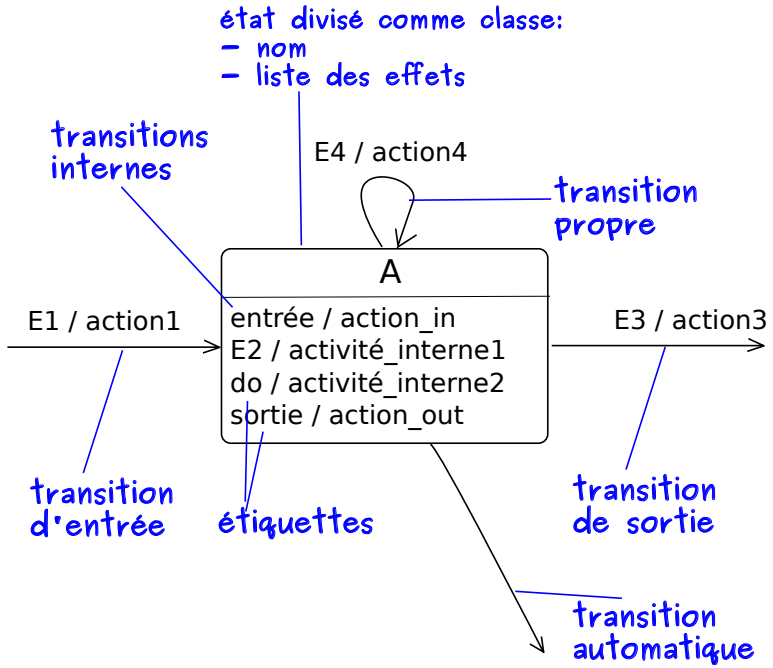
Transitions composites
Point de choix

- **Point de choix est dynamique** (ou point de décision) : gardes après le point de jonction évaluées au moment où le point de jonction est atteint



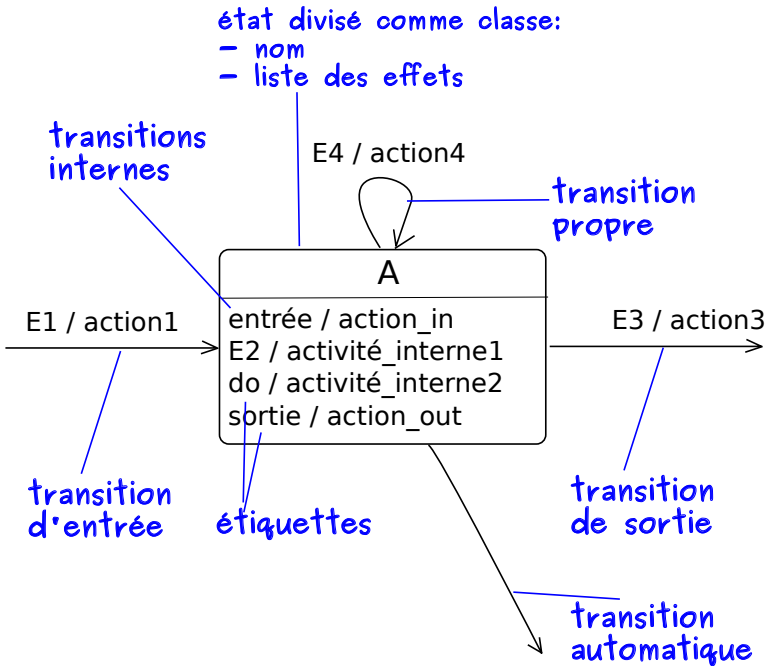
Activités internes à un état

- Un état peut être séparé en deux compartiments : nom de l'état & activités internes
- Quatre déclencheurs prédéfinis :
  - entry : activité à effectuer à chaque fois que l'objet rentre dans l'état. Cela permet de factoriser un même effet qui sera déclenché par toutes les transitions qui entrent dans l'état.
  - do : activité interne réalisée tant que l'objet est dans l'état courant, pouvant être interrompue.
  - evt-ext : activité interne instantanée, l'objet reste dans l'état courant.
  - exit : activité à effectuer à chaque fois que l'objet quitte l'état



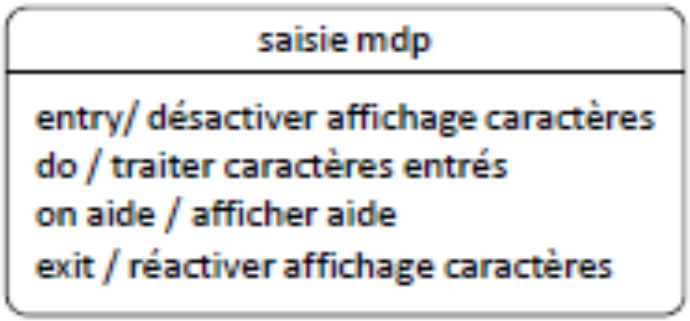
Transitions propres et internes

- Transition propre : l'objet quitte son état de départ pour le retrouver.
  - effets secondaires non négligeables : interruption puis redémarrage d'une activité durable, réalisation d'effets en entrée ou en sortie de l'état, retour à l'état initial si super-état, ...
- Transition interne : aucune influence sur l'état courant

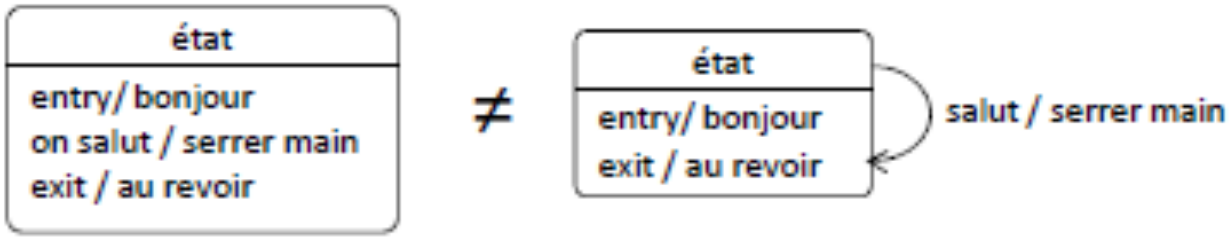


Transitions propres et internes

- Exemple : *lorsque le terminal entre en mode "saisie de mot de passe", l'affichage des caractères est désactivé, puis réactivé ensuite*



- Les activités liées aux déclencheurs entry et exit ne sont pas exécutées si l'événement spécifié par on survient. Pour indiquer qu'elles peuvent être exécutées plusieurs fois à l'arrivée d'un événement, on utilise une transition réflexive :



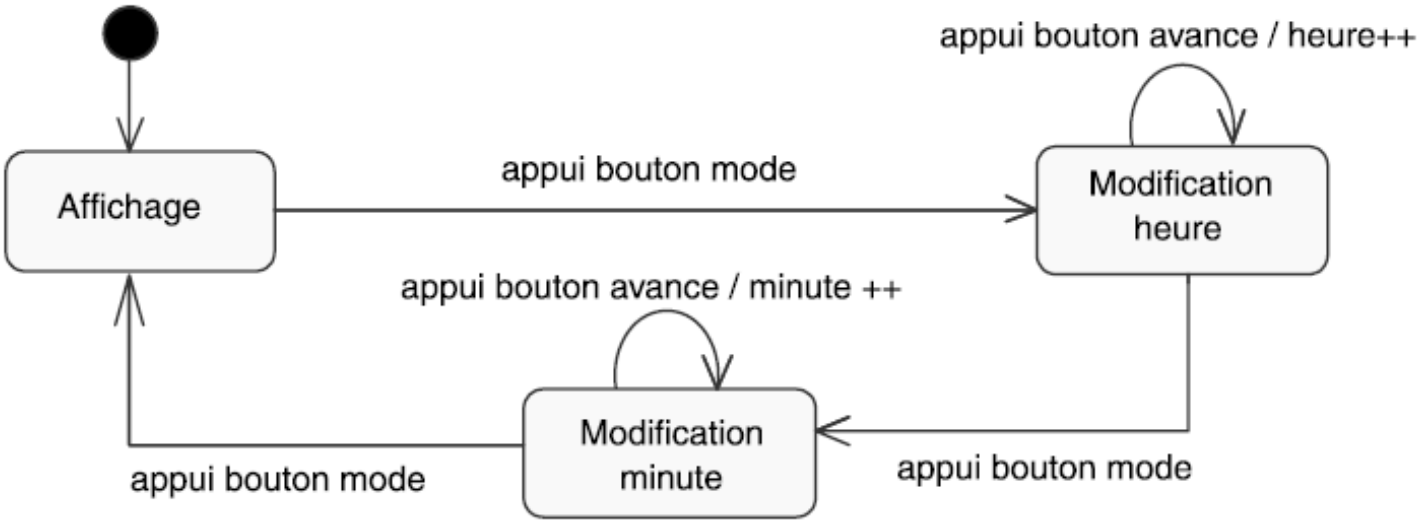
Exemple

Considérons une montre à cadran numérique simplifiée :

- Le mode courant est le mode ≪ Affichage ≫.Quand on appuie une fois sur le bouton mode, la montre passe en ≪ modification heure ≫. Chaque pression sur le bouton avance incrémente l'heure d'une unité.

  Quand on appuie une nouvelle fois sur le bouton mode, la montre passe en ≪ modification minute ≫. Chaque pression sur le bouton avance incrémente les minutes d'une unité.

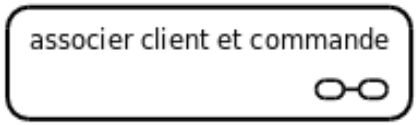
  Quand on appuie une nouvelle fois sur le bouton mode, la montre repasse en mode ≪ Affichage ≫.



Etats composites/englobants

Généralisation d'états

- Un état peut contenir des sous-états (décomposition)
- Facilite la représentation et permet d'occulter les détails
- Permet de factoriser les transitions de sortie du composite
  - Sous-états disjoints (décomposition disjonctive)
    → L'objet doit être dans un seul sous-état à la fois
  - Sous-états concurrents (décomposition conjonctive)
    → L'objet doit être simultanément dans plusieurs sous-état

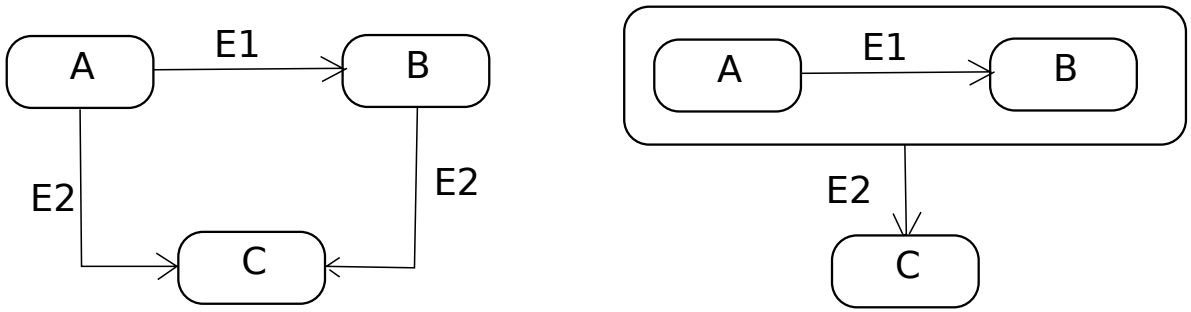


Notation abrégée d'un état composite (masquage des sous-états)

Etats composites/englobants

Etats disjoints

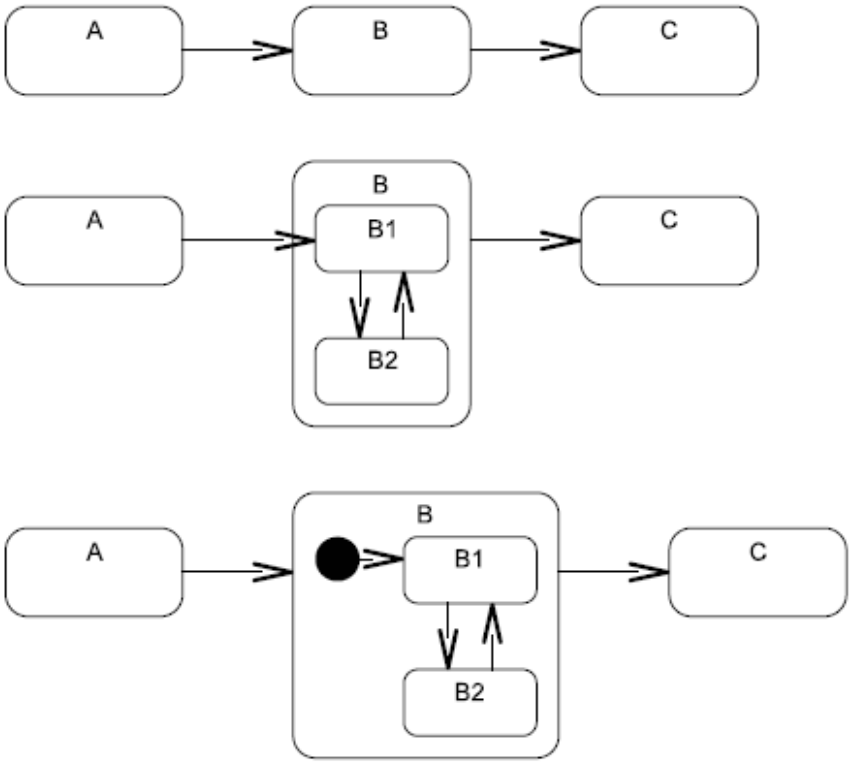
- Simplification par décomposition hiérarchique et factorisation

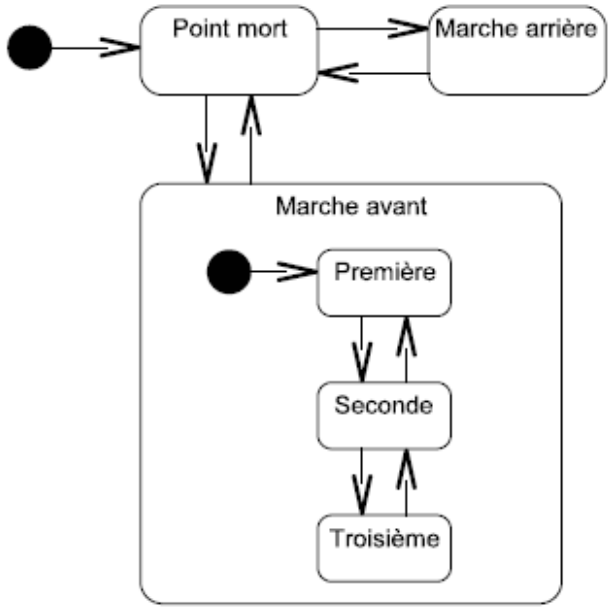


Etats composites/englobants

Etats disjoints

- Chaque transition de sortie s'applique à tous les sous-états
- Une seule transition d'entrée qui concerne un seul état (directement reportée dans état composite ou avec état initial emboité)





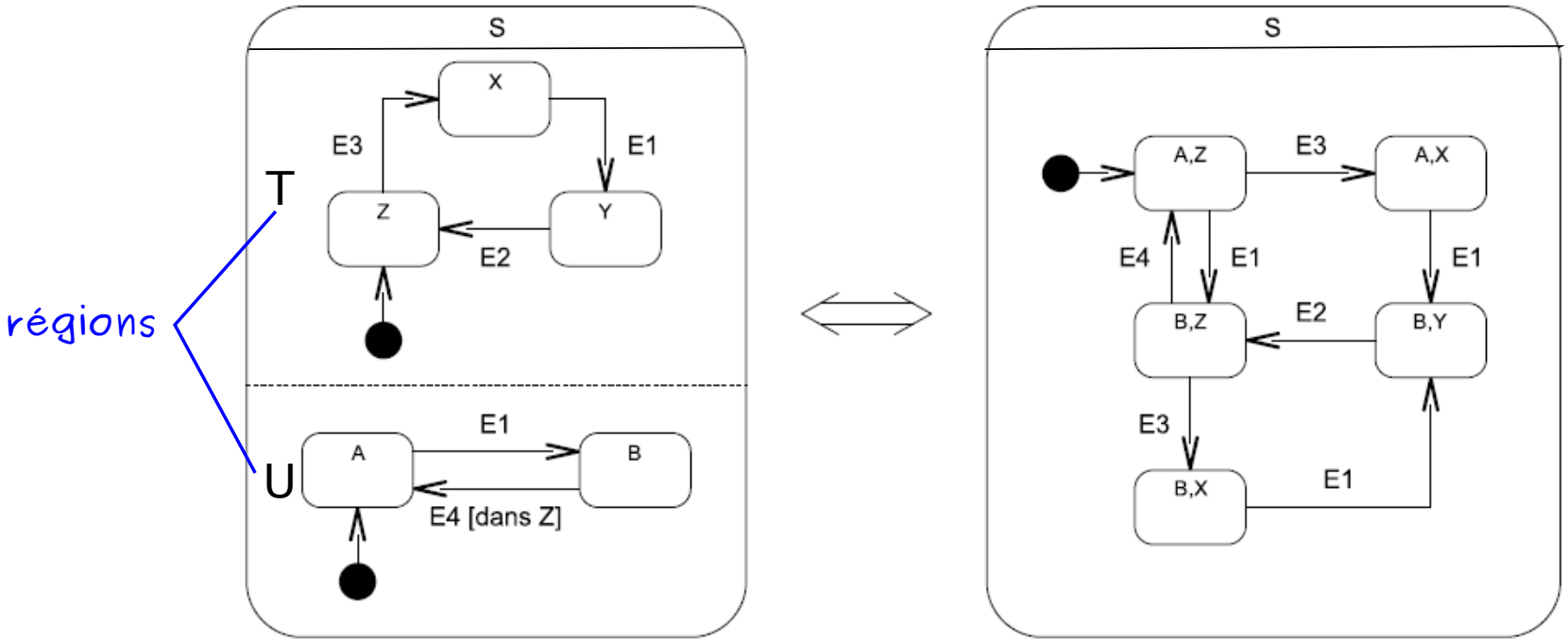
Etats composites/englobants

Etats concurrents

- Un état composite peut comporter plus d'une région : régions concurrentes
- Régions séparées par une ligne pointillée
- Chaque région représente un flot d'exécution : elles sont exécutées en parallèle
- Agrégation d'états : composition d'un état à partir de plusieurs sous-états concurrents appelés régions
- Permet de décrire des automates parallèles : plusieurs régions, chacune représente un flot d'exécution
- Un événement peut déclencher une transition dans plusieurs sous-automates
- Sortie possible quand tous les sous-automates sont dans un état final

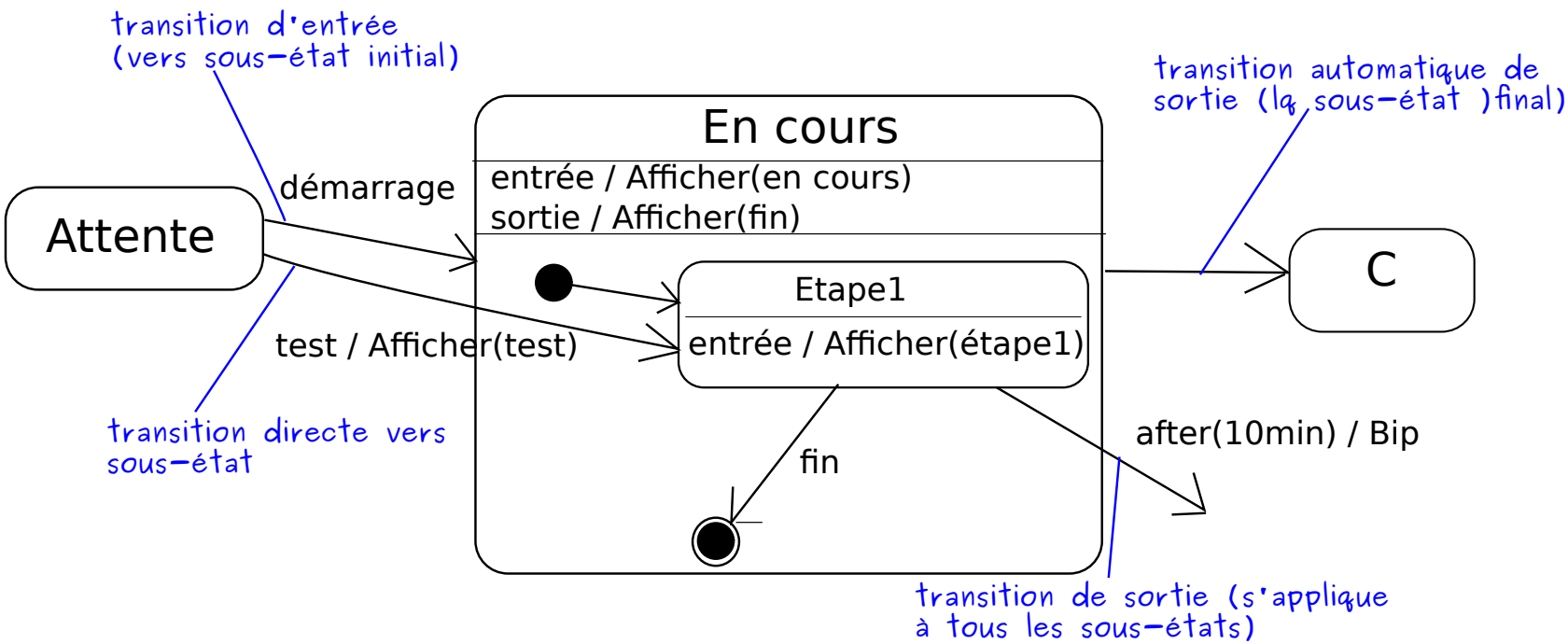
Etats composites/englobants

Etats concurrents



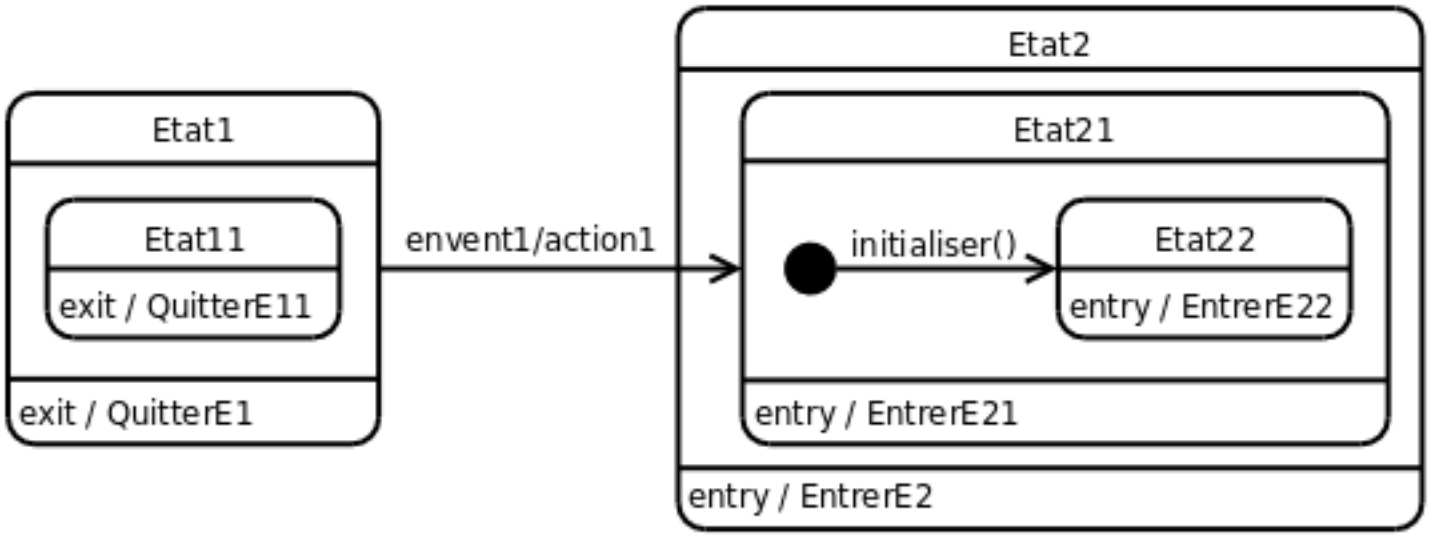
Etat `S` composite formé de 2 sous-états concurrents `T` et `U`

Transitions entre états composites

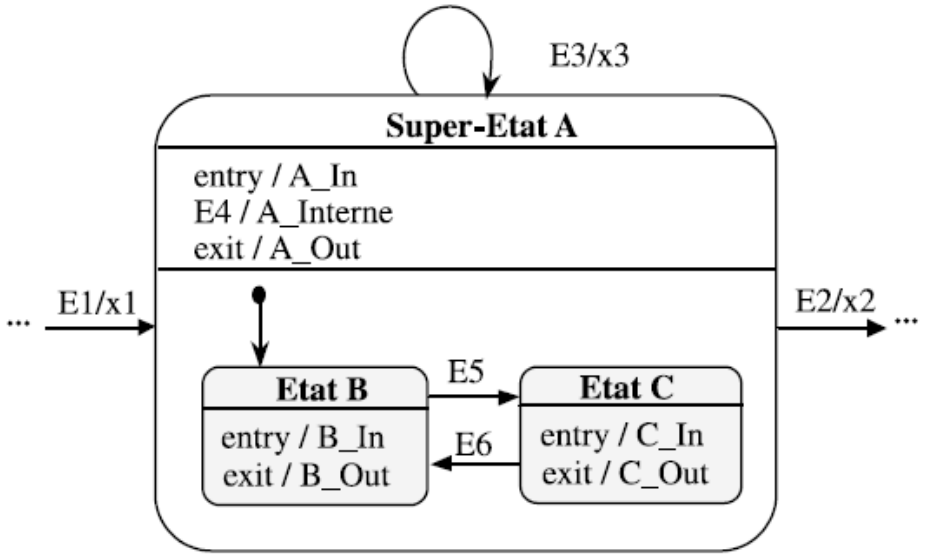


Transitions entre états composites

Exemple : depuis l'état `Etat11`, la réception de l'évènement `event1` provoque la séquence d'activités `QuitterE11`, `QuitterE1`, `action1`, `EntreeE2`, `Entree21`, `initialiser`, `EntreeE22` et place le système dans l'état `Etat22`



Exercice : Diagramme d'états hiérarchique

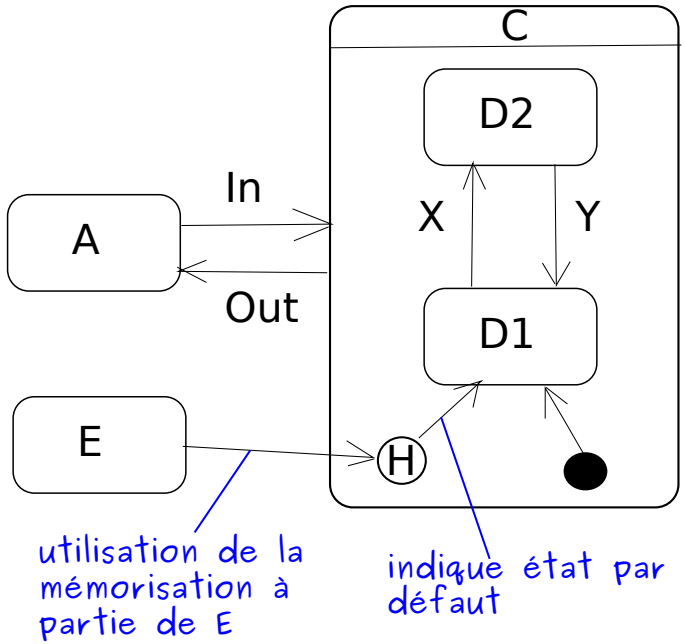


- Etudier l'ordre temporel d'exécution des effets en complétant le tableau suivant.

Etat de départ	Evénement	Effets	Etat d'arrivée
...	E1	?	?
?	E5	?	?
?	E4	?	?
?	E6	?	?
?	E3	?	?
?	E5	?	?
?	E3	?	?
?	E2	?	?

États historiques

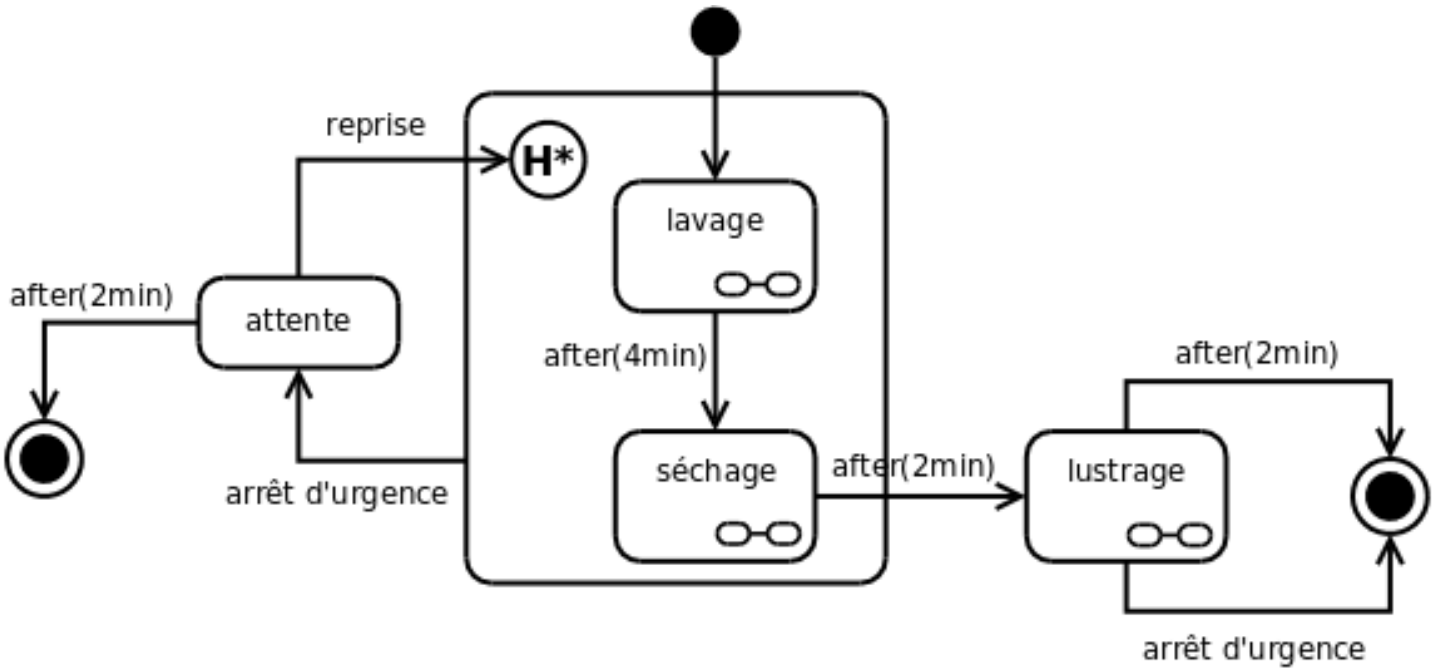
- Pseudo-état qui mémorise le dernier sous-état visité d'un état composite
- Une transition ayant pour cible l'état historique est équivalente à une transition qui a pour cible le dernier état visité de l'état englobant
- Indicateur spécial H placé dans l'état composite, H* mémorise quel que soit le niveau d'emboîtement



L'état C mémorise le dernier sous-état actif

États historiques

Exemple : lavage automatique d'une voiture. En phase de lavage ou de séchage, le client peut appuyer sur le bouton d'arrêt d'urgence (la machine se met en attente). Il a alors 2 min pour reprendre le lavage ou le lustrage, exactement où le programme à été interrompu, c'est-à-dire au niveau du dernier sous-état actif des états de lavage ou de lustrage (état historique profond). Si l'état avait été un état historique plat, c'est toute la séquence de lavage ou de lustrage qui aurait recommencée. En phase de lustrage, le client peut aussi interrompre la machine. Mais dans ce cas, la machine s'arrêtera



Quelques conseils

- Pas de transition sans événement
- L'automate doit être déterministe
- Si plusieurs transitions partant d'un état ont le même événement, alors il doit y avoir des gardes qui garantissent le déterminisme
- Tous les états doivent être accessibles depuis l'état initial
- S'il y a des états terminaux alors, pour chaque état non terminal, il doit exister un chemin de cet état vers un état terminal (...en général)

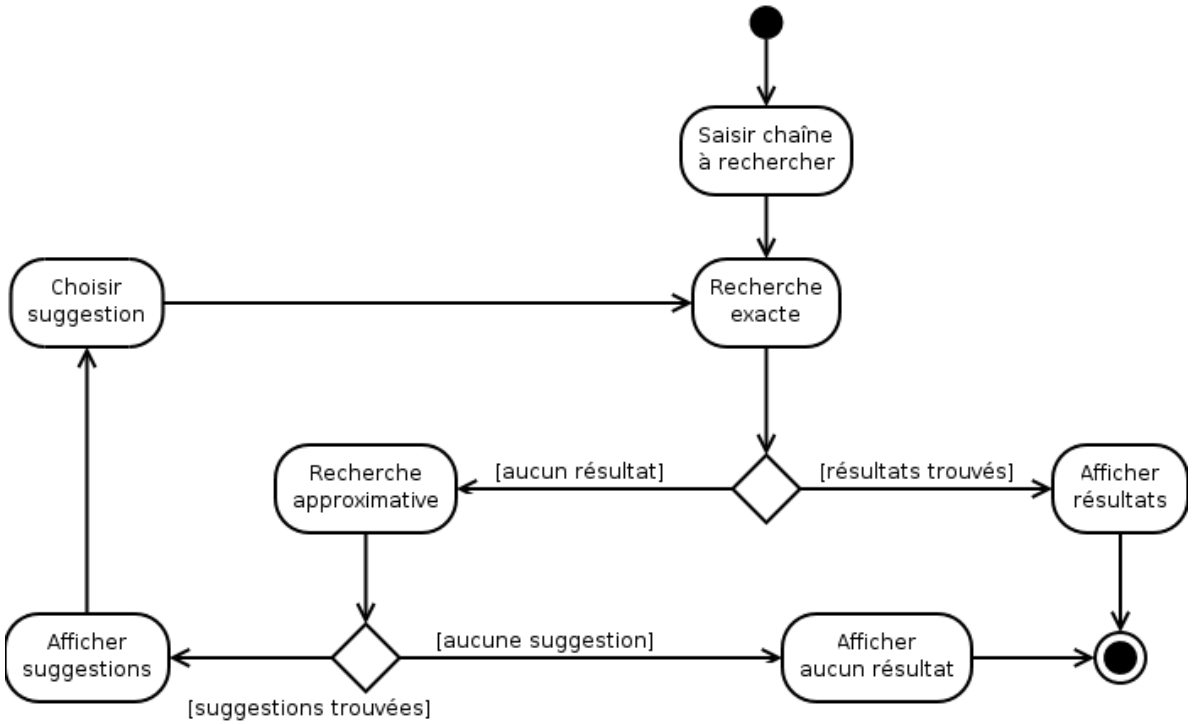
Comment le construire ?

- 1. Représentez tout d'abord la séquence d'états qui décrit le comportement nominal d'un objet, avec les transitions qui y sont associées
- 2. Ajoutez progressivement les transitions qui correspondent aux comportements alternatifs ou d'erreur
- 3. Complétez les effets sur les transitions et les activités dans les états
- 4. Structurez le diagramme en sous-états s'il devient trop complexe

Plan

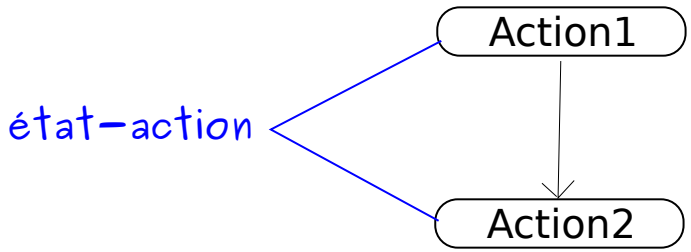
Diagrammes d'activités

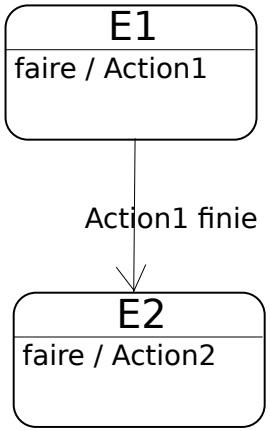
- Variante des diagrammes d'états-transitions
- Représentent le comportement interne d'une méthode ou d'un cas d'utilisation, réalisation d'une opération
- Mette l'accent sur les traitements (modélisation du cheminement de flots de contrôle et de flots de données)
- Nature procédurale du traitement des opérations (événement = fin de l'activité précédente), donc pas besoin de distinguer les états/activités/événement



Les états-action

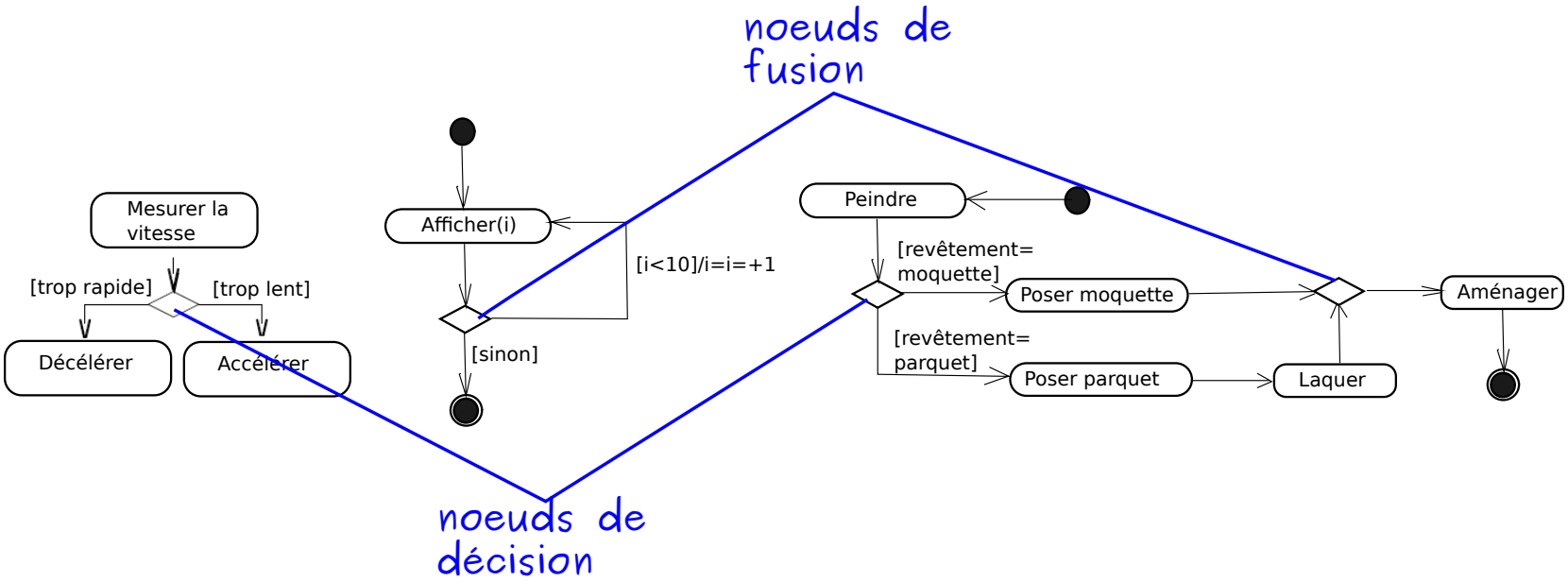
- Modélise une **étape** dans l'exécution d'un algorithme
- Etat simplifié
- Ex. : appel de procédure, création d'un objet, envoi d'un signal, ...





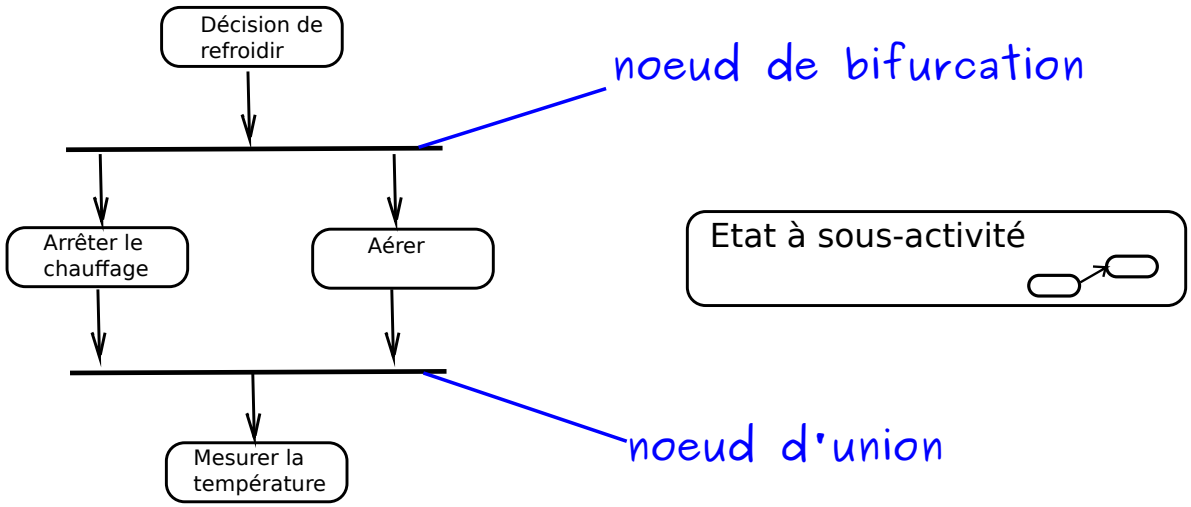
Les transitions

- En général, les états-action sont reliés par des transitions automatiques
- Possibilité d'inclure des gardes et effets
- Noeud de décision (*decision node*) : faire un choix entre plusieurs flots sortants (gardes)
- Noeud de fusion (*merge node*) : accepter un flot parmi plusieurs



Synchronisation

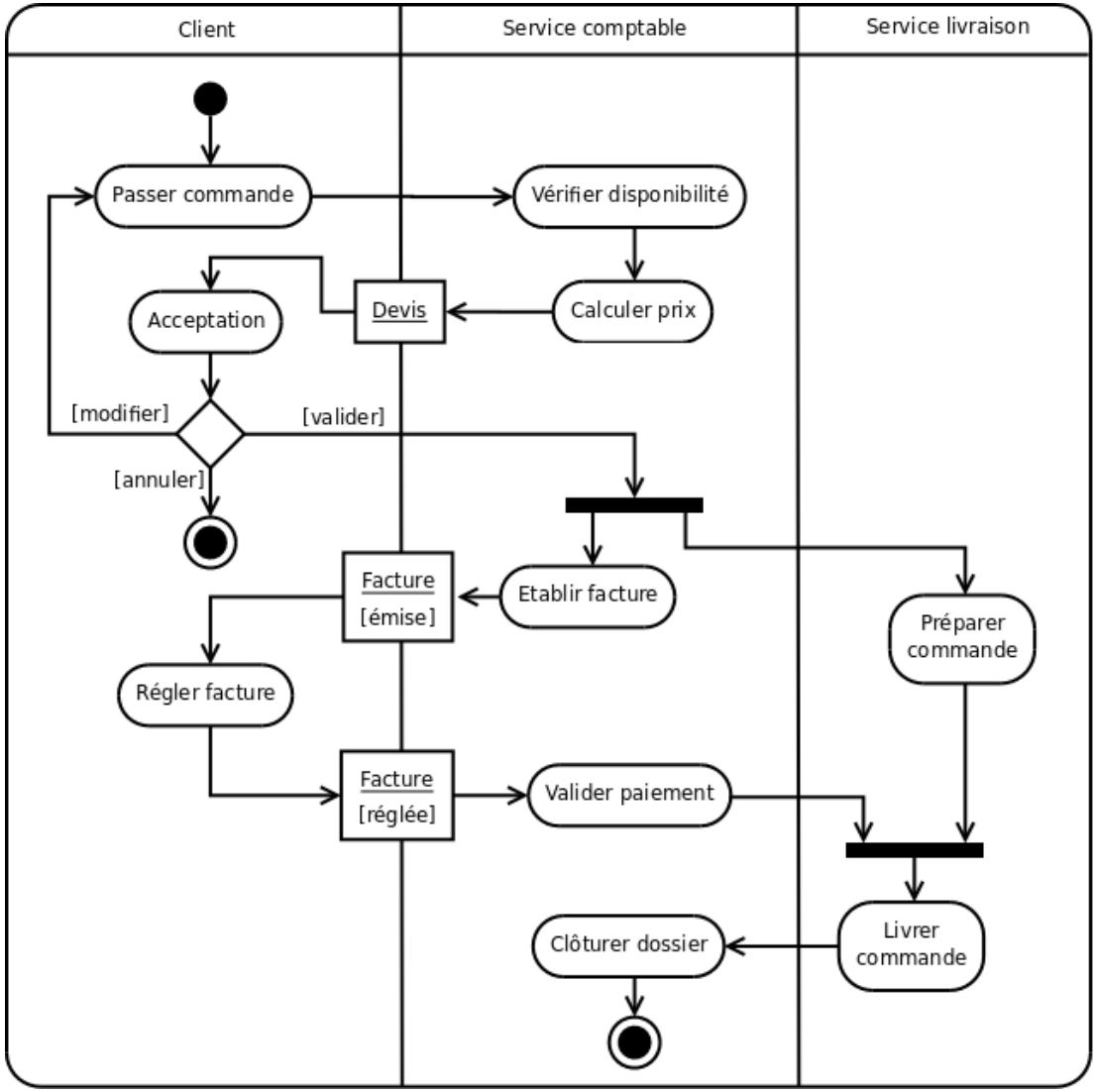
- Noeud d'union (*join node*) : franchi que lorsque toutes ses transitions en entrée sont déclenchées
- Noeud de bifurcation (*fork node*) : les transitions au départ sont déclenchées simultanément



Les travées

- Montrer les différentes responsabilités, organiser, regrouper
- Chaque responsabilité est assurée par un ou plusieurs objets
- Un état-action par travée, les transitions peuvent traverser les travées
- Faire apparaître les objets qui initient/sont utilisés par/sont modifiés par des actions

Les travées


Client
Service comptable
Service livraison
Passer commande
Vérifier disponibilité
Acceptation
Devis
Calculer prix
[modifier]
[valider]
[annuler]
Facture
[émise]
Etablir facture
Préparer
commande
Régler facture
Facture
[réglée]
Valider paiement
Clôturer dossier
Livrer
commande

Plan

Diagrammes dynamiques

Un objet interagit pour implémenter un comportement. On peut décrire cette interaction de deux manières complémentaires :

- l'une est centrée sur des objets individuels (diagramme d'états-transitions). On peut ainsi modéliser le comportement nominal d'un objet.
- l'autre sur une collection d'objets qui coopèrent (diagrammes d'interaction). On peut ainsi décrire un scénario de cas d'utilisation.

Le diagramme d'activité permet de représenter le comportement interne d'une opération et la dynamique d'un cas d'utilisation.