Chap 6 Diagrammes UML (vue dynamique)

1. Introduction

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) et l'autre sur une collection d'objets qui coopèrent (diagrammes d'interaction).
Les diagrammes d'interactions permettent d'établir un lien entre les diagrammes de cas d'utilisation et les diagrammes de classes : ils montrent comment des objets (i.e. des instances de classes) communiquent pour réaliser une certaine fonctionnalité. Ils apportent ainsi un aspect dynamique à la modélisation du système.
Pour produire un diagramme d'interaction, il faut focaliser son attention sur un sous-ensemble d'éléments du système et étudier leur façon d'interagir pour décrire un comportement particulier.

2. Représentation générale d'un digramme d'interaction

Un diagramme d'interaction se représente par un rectangle contenant, dans le coin supérieur gauche, un pentagone accompagné du mot-clef sd lorsqu'il s'agit d'un diagramme de séquence et com lorsqu'il s'agit d'un diagramme de communication. Le mot-clef est suivi du nom de l'interaction.

3. Diagramme de communication (Communication diagram)

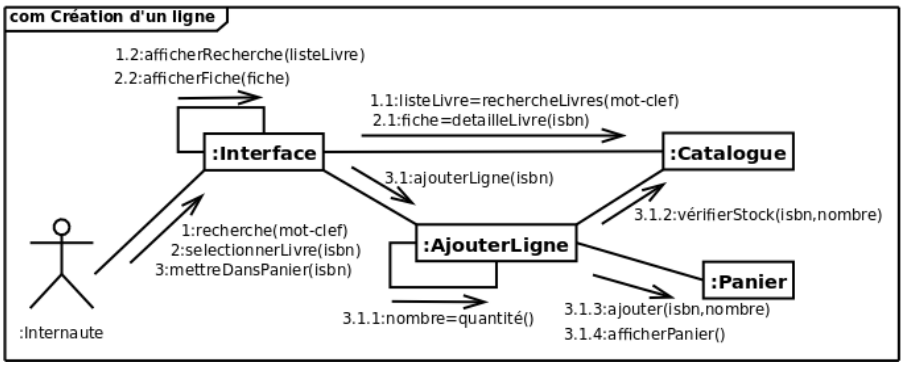

Figure 1 : Diagramme de communication illustrant la recherche puis l'ajout, dans -son panier virtuel, d'un livre lors d'une commande sur Internet.

Le diagramme de communication aide à valider les associations du diagramme de classe en les utilisant comme support de transmission des messages.

Remarque : Un diagramme de communication est un diagramme d'interactions UML 2.0, (appelé diagramme de collaboration en UML 1)

a. Représentation des lignes de vie

Les lignes de vie sont représentées par des rectangles contenant une étiquette (nom de objet).
Remarque : Le nom de la classe dont l'objet est une instance est précédé d'un << : >> et est souligné. Pour différencier les objets d'une même classe, leur identifiant peut être ajouté devant le nom de la classe. Exemple p1 :personne ou :personne désigne un objet de type la classe personne.

b. Représentation des connecteurs

Les relations entre les lignes de vie sont appelées connecteurs et se représentent par un trait plein reliant deux lignes de vie et dont les extrémités peuvent être ornées de multiplicités.

c. Représentation des messages

Dans un diagramme de communication, les messages sont généralement ordonnés selon un numéro de séquence croissant.

4. Diagramme de séquence

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique. Ainsi, contrairement au diagramme de communication, le temps y est représenté explicitement par une dimension (la dimension verticale) et s'écoule de haut en bas.

a. Représentation des lignes de vie

Une ligne de vie se représente par un rectangle, auquel est accrochée une ligne verticale pointillée, contenant une étiquette (nom de l'objet).

b. Représentation des messages

Un message définit une communication particulière entre des lignes de vie. Plusieurs types de messages existent, les plus communs sont :

- l'envoi d'un signal ;
- l'invocation d'une opération ;
- la création ou la destruction d'une instance.

b.1 Messages asynchrones

Figure 2: Représentation d'un message asynchrone.

Une interruption ou un événement sont de bons exemples de signaux. Graphiquement, un message asynchrone se représente par une flèche en traits pleins et à l'extrémité ouverte partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible.

b.2 Messages synchrones

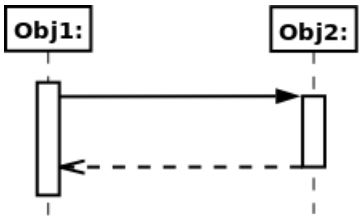

Figure 3 : Représentation d'un message synchrone.

L'invocation d'une opération est le type de message le plus utilisé en programmation objet. Graphiquement, un message synchrone se représente par une flèche en traits pleins et à l'extrémité pleine partant de la ligne de vie d'un objet expéditeur et allant vers celle de l'objet cible. Ce message peut être suivi d'une réponse qui se représente par une flèche en pointillé.

b.3 Messages de création et destruction d'instance

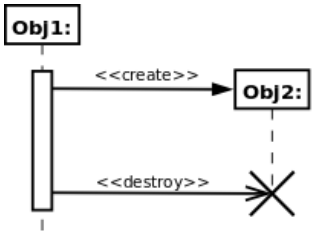

Figure 4 : Représentation d'un message de création et destruction d'instance.

La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.
La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.
La destruction d'un objet n'est pas nécessairement consécutive à la réception d'un message.

4.1. Événements et messages

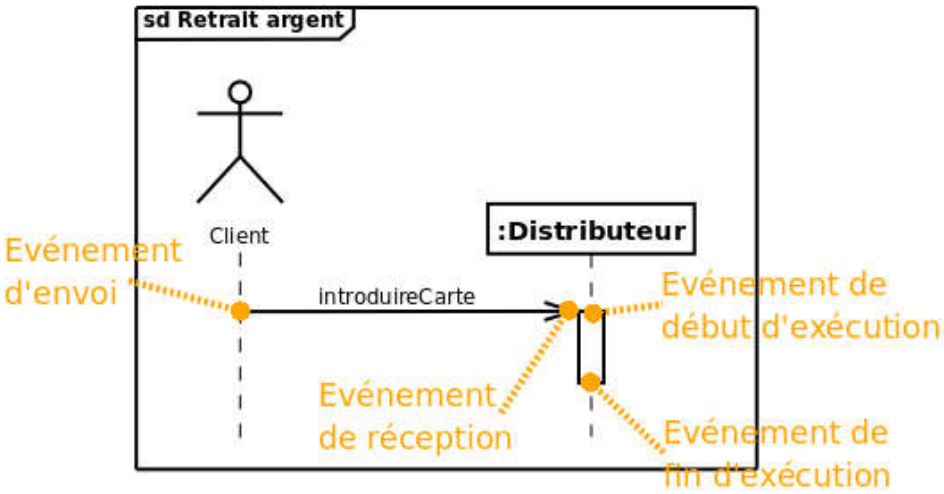

Figure 5 : Les différents évènements correspondant à un message asynchrone.

Dans la plupart des cas, la réception d'un message est suivie de l'exécution d'une méthode d'une classe. Cette méthode peut recevoir des arguments et la syntaxe des messages permet de transmettre ces arguments. La syntaxe de ces messages est la même que pour un diagramme de communication excepté deux points :

- la direction du message est directement spécifiée par la direction de la flèche qui matérialise le message
- les numéros de séquence sont généralement omis puisque l'ordre relatif des messages est déjà matérialisé par l'axe vertical qui représente l'écoulement du temps.

4.2. Fragments d'interaction combinés

Un fragment combiné se représente de la même façon qu'une interaction. Il est représenté dans un rectangle dont le coin supérieur gauche contient un pentagone. Dans le pentagone figure le type de la combinaison, appelé opérateur d'interaction.

La liste suivante regroupe les opérateurs d'interaction par fonctions :

- les opérateurs de choix et de boucle : **alternative**, **option**, **break** et **loop** ;
- les opérateurs contrôlant l'envoi en parallèle de messages : **parallel** et **critical region** ;
- les opérateurs contrôlant l'envoi de messages : **ignore**, **consider**, **assertion** et **negative** ;
- les opérateurs fixant l'ordre d'envoi des messages : **weak sequencing** , **strict sequencing**.
-

Exemple Opérateur alt

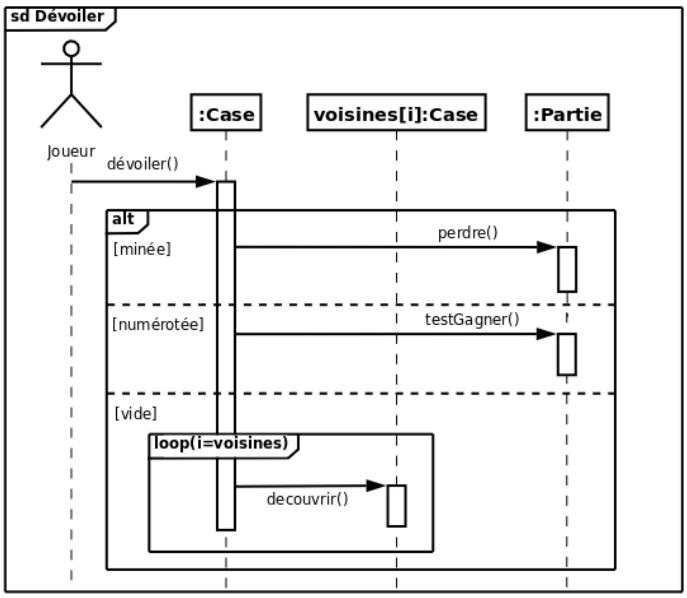

Figure 6 : Représentation d'un choix dans un diagramme de séquence illustrant le dé couvrement d'une case au jeu du démineur.

L'opérateur alternative, ou alt, est un opérateur conditionnel possédant plusieurs opérandes. C'est un peu l'équivalent d'une exécution à choix multiple. Chaque opérande détient une condition de garde. L'absence de condition de garde implique une condition vraie (true). La condition else est vraie si aucune autre condition n'est vraie. Exactement un opérande dont la condition est vraie est exécuté. Si plusieurs opérandes prennent la valeur vraie, le choix est non déterministe.

5. Diagramme d'activité

Les diagrammes d'activités permettent de mettre l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.

Dans la phase de conception, les diagrammes d'activités sont particulièrement adaptés à la description des cas d'utilisation. Plus précisément, ils viennent illustrer et consolider la description textuelle des cas d'utilisation. De plus, leur représentation sous forme d'organigrammes les rend facilement intelligibles.

5.1 Activité et Transition

a. Action (action)

Une action est le plus petit traitement qui puisse être exprimé en UML. Une action peut être, par exemple :

- une affectation de valeur à des attributs ;
- un accès à la valeur d'une propriété structurelle (attribut ou terminaison d'association) ;

- la création d'un nouvel objet ou lien ;
- un calcul arithmétique simple ;
- l'émission d'un signal ;
- la réception d'un signal ;

b. Activité (activity)

Une activité définit un comportement décrit par un séquencement organisé d'unités dont les éléments simples sont les actions. Le flot d'exécution est modélisé par des nœuds reliés par des arcs (transitions). Le flot de contrôle reste dans l'activité jusqu'à ce que les traitements soient terminés.

Une activité est un comportement (behavior en anglais) et à ce titre peut être associée à des paramètres.

c. Nœud d'activité

Un nœud d'activité modélise une action, permettant de représenter les étapes le long du flot d'une activité. Il existe trois familles de nœuds d'activités :

- les nœuds d'exécutions
- les nœuds objets
- les nœuds de contrôle

La figure suivante représente le nœud représentant une action, qui est une variété de nœud exécutable, un nœud objet, un nœud de décision ou de fusion, un nœud de bifurcation ou d'union, un nœud initial, un nœud final et un nœud final de flot.

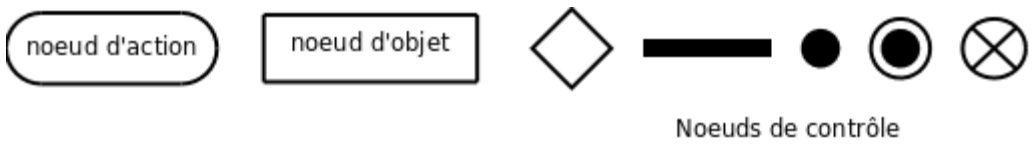

Figure 7 : Représentation graphique des nœuds d'activité.

d. Transition

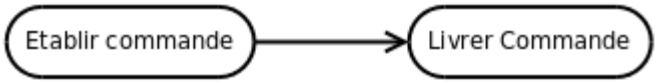

Figure 8 : Représentation graphique d'une transition.

Graphiquement les transitions sont représentées par des flèches en traits pleins qui connectent les activités entre elles. Les transitions spécifient l'enchaînement des traitements et définissent le flot de contrôle.

5.2 Exemple de diagramme d'activité illustrant l'utilisation de nœuds de contrôle décrit la prise en compte d'une commande.

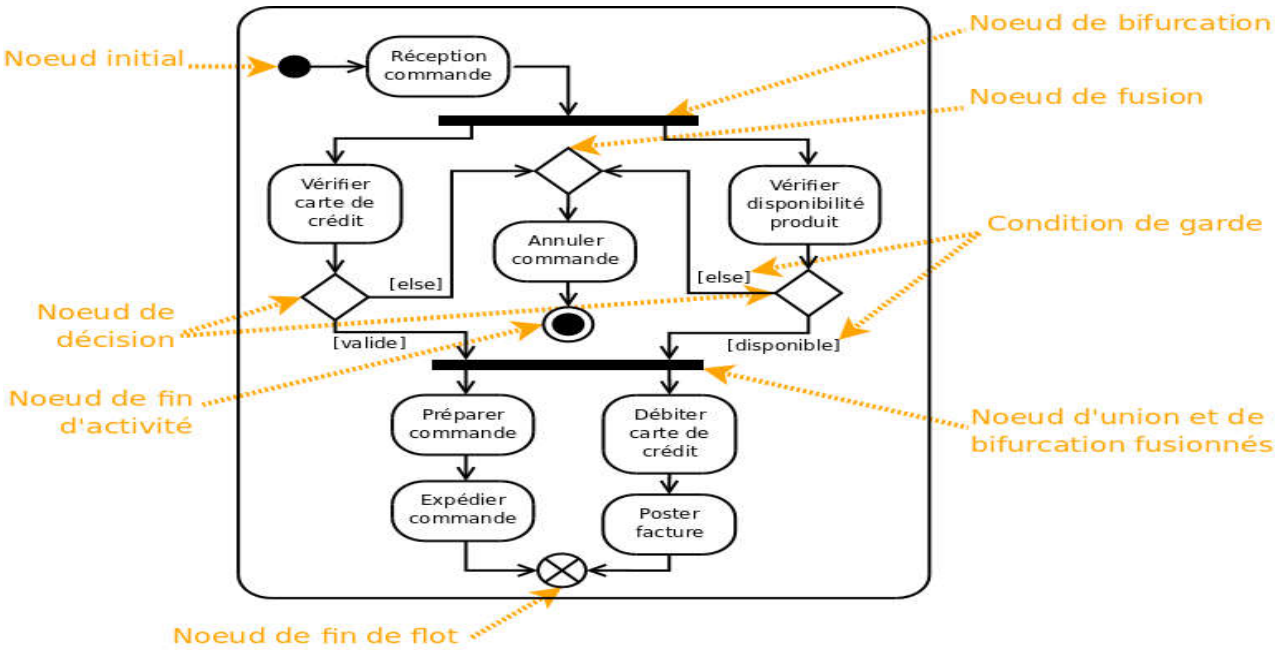

Figure 9 : Exemple de diagramme d'activité illustrant l'utilisation de nœuds de contrôle. Ce diagramme décrit la prise en compte d'une commande.

6. Diagramme d’état

6.1. Introduction

Les diagrammes d'états-transitions d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets.

6.2. Diagrammes d'états-transitions

Un diagramme d'états-transitions ne peut être associé qu'à un seul classeur (dans notre cas une classe). Tous les automates à états finis des diagrammes d'états-transitions d'un système s'exécutent concurremment et peuvent donc changer d'état de façon indépendante.

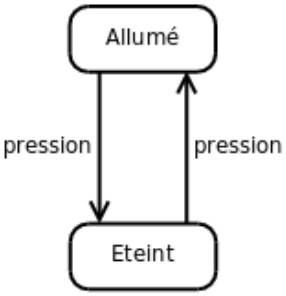

Figure 10 : Un diagramme d'états-transitions simple.

6.3 État dans un diagramme d'états-transitions Un état se représente graphiquement dans un diagramme d'états-transitions par un rectangle aux coins arrondis.
Certains états, dits composites, peuvent contenir des sous-états.
Le nom de l'état peut être spécifié dans le rectangle et doit être unique dans le diagramme d'états-transitions, ou dans l'état enveloppant. Un objet peut passer par une série d'états pendant sa durée de vie. Un état représente une période dans la vie d'un objet pendant laquelle ce dernier attend un événement ou accomplit une activité. Dans le cas d'un diagramme d'états-transitions simple, il ne peut y avoir qu'un seul état actif à la fois.

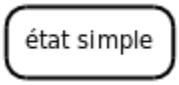

Figure 11 : Exemple d'état simple.

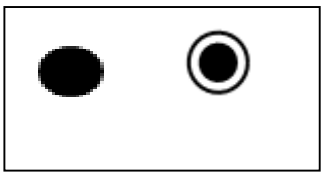

Figure 12 : Représentation graphique de l'état initial et final.

6.4. Événement

Un événement se produit à un instant précis et est dépourvu de durée. Quand un événement est reçu, une transition peut être déclenchée et faire basculer l'objet dans un nouvel état. On peut diviser les événements en plusieurs types explicites et implicites : signal, appel, changement et temporel.

1. Les événements de type signal est déclenché par La réception d'un signal qui est un événement pour l'objet destinataire. Un même objet peut être à la fois expéditeur et destinataire.
2. Les événements d'appel **(call)** sont les événements déclenchés par des méthodes déclarées au niveau du diagramme de classes.
3. Les événements de changement **(change)** sont d'une manière déclarative d'attendre qu'une condition soit satisfaite.
4. Les événements temporels (**after ou when**) Le temps commence à s'écouler dès l'entrée dans l'état courant.

6.5. Transition

Une transition définit la réponse d'un objet à l'occurrence d'un événement. Elle lie, généralement, deux états E1 et E2 et indique qu'un objet dans un état E1 peut entrer dans l'état E2 et exécuter certaines activités, si un événement déclencheur se produit et que la condition de garde est vérifiée.

Lorsqu'une transition se déclenche, son effet s'exécute. Il s'agit généralement d'une activité qui peut être

- une opération primitive comme une instruction d'assignation ;
- l'envoi d'un signal ;
- l'appel d'une opération ;
- une liste d'activités, etc.

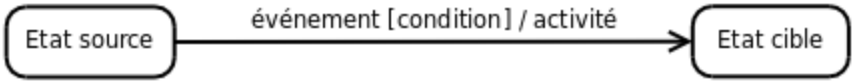

Figure 13 : Représentation graphique d'une transition entre deux états.

6.6. Point de choix

Il est possible de représenter des alternatives telles que les points de jonction (représentés par un petit cercle plein) et les points de décision (représentés par un losange).

6.6.1. Point de jonction

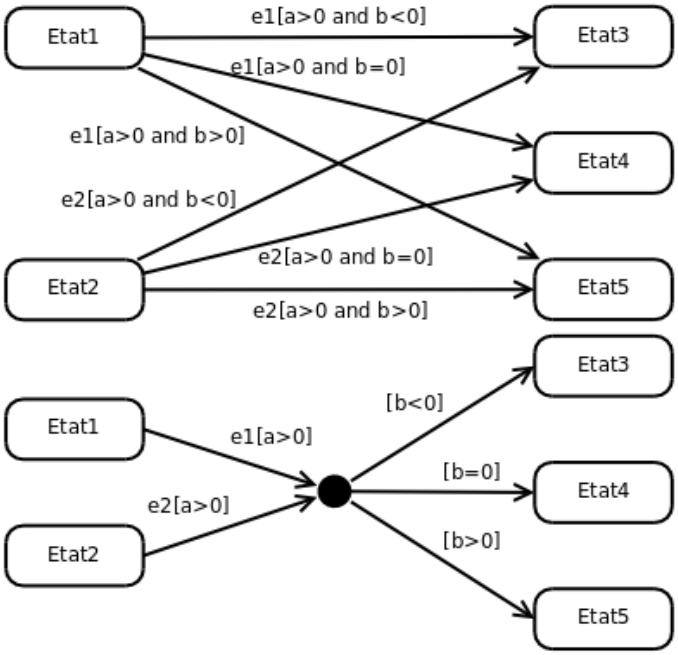

Figure 14 : Diagramme représentant un point de jonction.

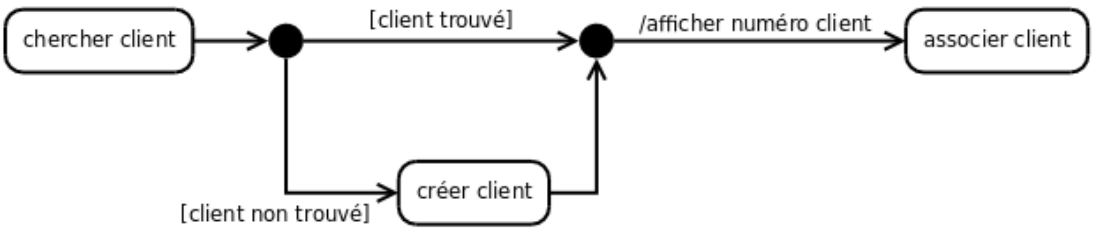

Figure 15 : Exemple d'utilisation de deux points de jonction pour représenter une alternative.

- Le point de jonction représente le branchement d'une clause conditionnelle.
- Les points de jonction sont un artefact graphique (**un pseudo état en l'occurrenc**e) qui permet de partager des segments de transition.

6.6.2. Point de décision

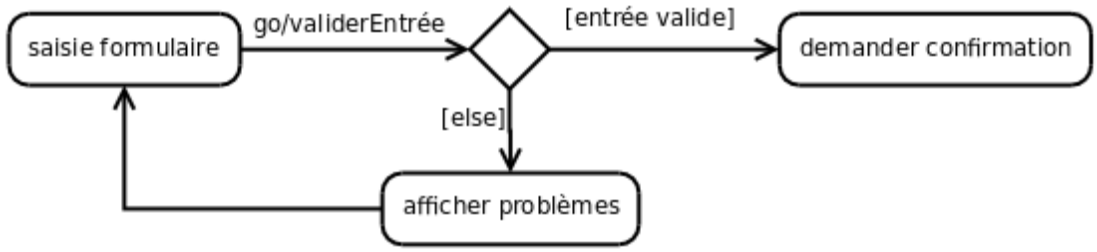

Figure 16 : Exemple d'utilisation d'un point de décision.

Un point de décision possède une entrée et au moins deux sorties. Ce segment n'est franchissable que si les gardes des autres segments sont toutes fausses. L'utilisation d'une clause [else] est recommandée après un point de décision, car elle garantit un modèle bien formé.

6.7. États composites

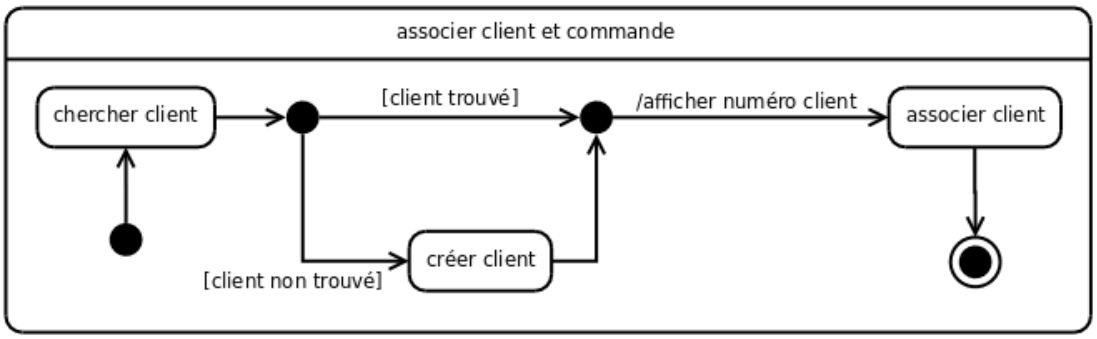

Figure 17 : Exemple d'état composite modélisant l'association d'une commande à un client.

Un état composite est un état décomposé en régions contenant chacune un ou plusieurs sous-états.

L'utilisation d'états composites permet de développer une spécification par raffinements. Il n'est pas nécessaire de représenter les sous-états à chaque utilisation de l'état englobant.

Les transitions peuvent avoir pour cible la frontière d'un état composite et sont équivalentes à une transition ayant pour cible l'état initial de l'état composite.

Une transition ayant pour source la frontière d'un état composite est équivalente à une transition qui s'applique à tout sous-état de l'état composite source. Cette relation est transitive : la transition est franchissable depuis tout état imbriqué, quelle que soit sa profondeur.

Par contre, si la transition ayant pour source la frontière d'un état composite ne porte pas de déclencheur explicite, elle est franchissable quand l'état final de l'état composite est atteint.

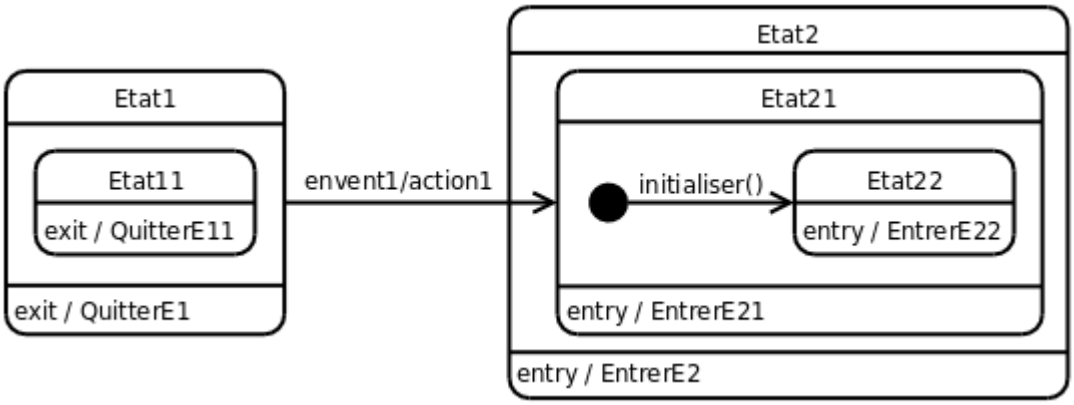

Figure 18 : Exemple de configuration complexe de transition. Depuis l'état État 1, la réception de l'événement event1 produit la séquence d'activités QuitterE11, QuitterE1, action1, EntrerE2, EntrerE21, initialiser(), EntrerE22, et place le système dans l'état État22.

Exercice

a) Modéliser le retrait d'argent avec une carte VISA avec un diagramme d'activités.
La carte peut être invalide. Si elle est valide, le client doit taper son code. La carte est avalée après trois essais infructueux. Le SA VISA autorise un certain montant ou refuse tout retrait. Une carte non récupérée est avalée. Les billets non récupérés par le client sont repris. Un ticket est toujours imprimé pendant que les billets sont proposés.
b) Modéliser le scénario nominal (succès) avec un diagramme de séquence.

Référence :

1. G. Booch, J. Rumbaugh, I. Jacobson, « The Unified Modeling Language (UML) user guide", Addison-Wesley, 1999.
2. Benoit charroux, aomar Osmani, Yann Therry-Mieg,"UML2 synthèse et exercices",Pearson édition france,ISBN2-7440-7124-2,2005.
3. G. Booch et al., « Object-Oriented Analysis and Design, with applications », Addison- Wesley, 2007.
4. Cours UML 2.0 de Laurent Audibert , site http://www.developpez.com