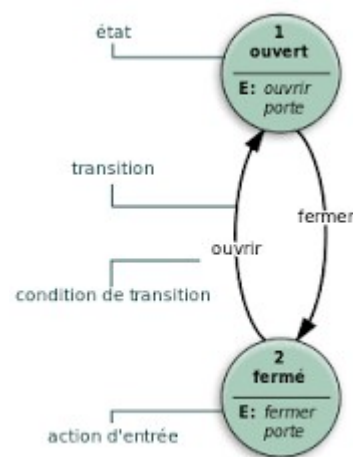


Diagramme états-transitions

Table des matières

1. Introduction.....	2
2. Définition.....	2
3. Composants.....	2
4. Automates à états finis.....	3
4. Types d'état.....	3
5. Événements.....	4
5.1. Événement d'appel (call).....	4
5.2. Événement de changement (change).....	4
5.3. Événement temporel (after ou when).....	4
5.4. Événement paramétré.....	4
6. Transition.....	5
7. Condition de garde.....	5
8. Effet d'une transition.....	5
8.1. Transition externe.....	6
8.2. Transition d'achèvement.....	6
8.3. Transition interne.....	6
8.4. Point de jonction.....	7
8.5. Point de décision.....	7
9. États composites.....	8
10.1. Points de connexion.....	9
10.2. Concurrence.....	10
11. Exercice : publiphone.....	10

Un diagramme états-transitions est un schéma utilisé en génie logiciel pour représenter des automates déterministes. Il fait partie du modèle UML et s'inspire principalement du formalisme des statecharts et rappelle les grafquets des automates. S'ils ne permettent pas de comprendre globalement le fonctionnement du système, ils sont directement transposables en algorithme. Tous les automates d'un système s'exécutent parallèlement et peuvent donc changer d'état de façon indépendante.



1. Introduction

Le diagramme d'états-transitions est le seul diagramme, de la norme UML, à offrir une vision complète et non ambiguë de l'ensemble des comportements de l'élément auquel il est attaché. En effet, un diagramme d'interaction n'offre qu'une vue partielle correspondant à un scénario sans spécifier comment les différents scénarii interagissent entre eux.

La vision globale du système n'apparaît pas sur ce type de diagramme puisqu'ils ne s'intéressent qu'à un seul élément du système indépendamment de son environnement.

Un automate à états finis est un automate dont le comportement des sorties ne dépend pas seulement de l'état de ses entrées, mais aussi d'un historique des sollicitations passées. Cet historique est caractérisé par un état global.

2. Définition

Un diagramme d'états-transitions présente un automate à états finis. Il permet ainsi de décrire les changements d'états d'un objet ou d'un composant.

- Un état se caractérise par sa durée et sa stabilité.
- Une transition représente le passage instantané d'un état vers un autre.

Une transition est déclenchée :





- soit par un événement.
- soit automatiquement lorsque aucun événement déclencheur est spécifié.

3. Composants

Un diagramme d'états-transitions est composé des éléments suivants :

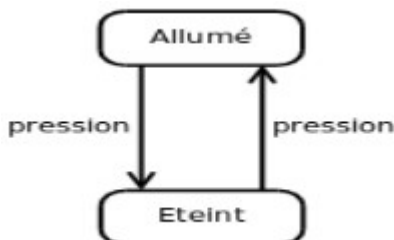
- **état** : représente la valeur des attributs d'un objet à un instant donné.
- **état initial** : représente l'état au démarrage du système.
- **état final** : représente l'état dans lequel se trouve le système à la fin du fonctionnement.
- **super-état** : permet de structurer le diagramme en indiquant plusieurs niveau de distinction entre les états.
- **historique** : représente le dernier état actif d'un super-état.
- **souche** : permet de symboliser les états contenus dans un super-état. Il est ainsi possible de relier ces états à d'autres états n'appartenant pas au super-état.
- **transition** : représente le passage d'un état à un autre.
- **paquetage** : divise et organise la représentation du diagramme (de la même manière que les répertoires organisent les fichiers).

FORMALISME

- Etat  Etat 1
- Transition  avec le nom d'événement écrit au-dessus, au-dessous, ou à côté.
- Etat initial 
- Etat final 

4. Automates à états finis

Un automate à états finis est graphiquement représenté par un graphe comportant des états, matérialisés par des rectangles aux coins arrondis, et des transitions, matérialisées par des arcs orientés liant les états entre eux.



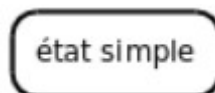
Un diagramme d'états-transitions simple.

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.

4. Types d'état

Dans un diagramme d'état-transition, l'état peut être simple ou composite :

- **État simple** : Il ne possède pas de sous-structure mais uniquement, le cas échéant, un jeu de transitions internes. Exemple d'état simple :



Dans le cas d'un diagramme d'états-transitions simple (sans transition concurrente), il ne peut y avoir qu'un seul état actif à la fois. Dans ce cas, les notions d'état actif et d'état global se rejoignent.

- **État Composites** : Il contient des sous-états.

Un état peut être partitionné en plusieurs compartiments séparés par une ligne horizontale. Le premier compartiment contient le nom de l'état et les autres peuvent recevoir des transitions internes, ou des sous-états, quand il s'agit d'un état composite. Dans le cas d'un état simple (sans transitions interne ou sous-état), on peut omettre toute barre de séparation.

5. Événements

Un événement est quelque chose qui se produit pendant l'exécution d'un système et qui mérite d'être modélisé. Les diagrammes d'états-transitions permettent justement de spécifier les réactions d'une partie du système à des événements discrets. 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 : appel, changement et temporel.

5.1. Événement d'appel (call)

Un événement d'appel représente la réception de l'appel d'une opération par un objet. Les paramètres de l'opération sont ceux de l'événement d'appel. La syntaxe d'un événement d'appel est la même que celle d'un signal. Par contre, les événements d'appel sont des méthodes déclarées au niveau du diagramme de classes.

5.2. Événement de changement (change)

Un événement de changement est généré par la satisfaction (passage de faux à vrai) d'une expression booléenne sur des valeurs d'attributs. Il s'agit d'une manière déclarative d'attendre qu'une condition soit satisfaite. La syntaxe d'un événement de changement est la suivante :

```
when ( <condition_booléenne> )
```

5.3. Événement temporel (after ou when)

Les événements temporels sont générés par le passage du temps. Ils sont spécifiés soit de manière absolue (date précise), soit de manière relative (temps écoulé). Par défaut, le temps commence à s'écouler dès l'entrée dans l'état courant.

La syntaxe d'un événement temporel spécifié de manière relative est la suivante :

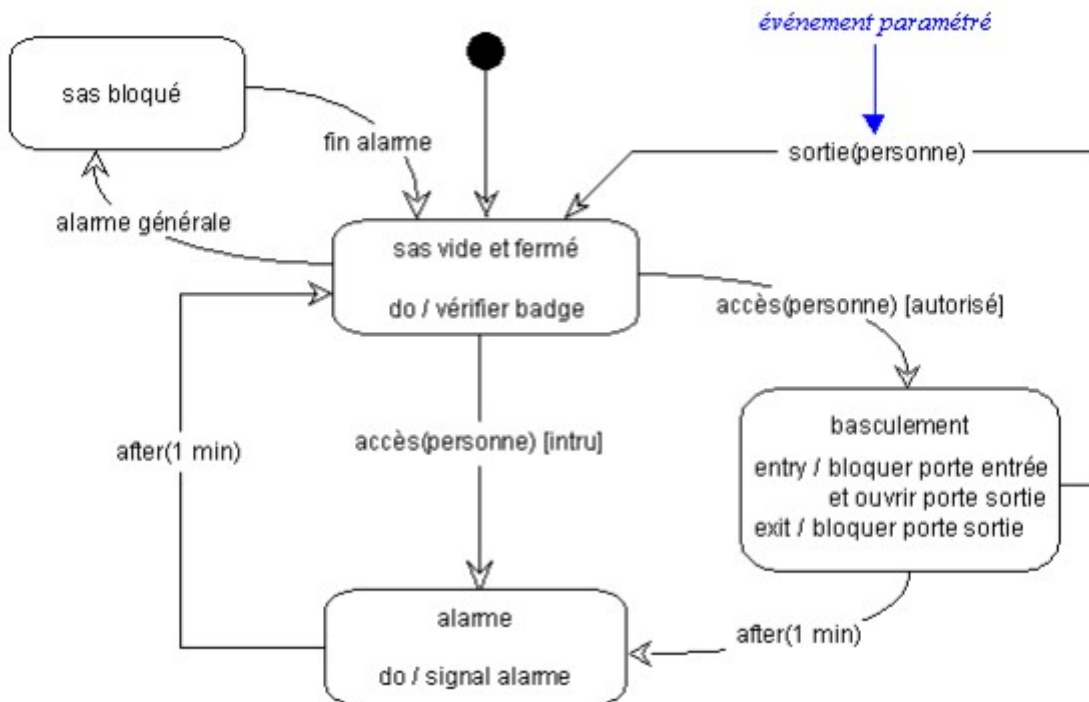
```
after ( <durée> )
```

Un événement temporel spécifié de manière absolue est défini en utilisant un événement de changement :

```
when ( date = <date> )
```

5.4. Événement paramétré

UML permet aussi de paramétrer les événements, comme dans l'exemple suivant :



6. 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. Elle peut être interne ou externe et sa syntaxe est :

[<événement>] ['[' <garde> ']'] ['/' <activité>]

Le même événement peut être le déclencheur de plusieurs transitions quittant un même état. Chaque transition avec le même événement doit avoir une condition de garde différente. En effet, une seule transition peut se déclencher dans un même flot d'exécution. Si deux transitions sont activées en même temps par un même événement, une seule se déclenche et le choix n'est pas prévisible.

7. Condition de garde

Une transition peut avoir une condition de garde (spécifiée par '[' <garde> ']' dans la syntaxe). Il s'agit d'une expression logique sur les attributs de l'objet, associé au diagramme d'états-transitions, ainsi que sur les paramètres de l'événement déclencheur. La condition de garde est évaluée uniquement lorsque l'événement déclencheur se produit. Si l'expression est fausse à ce moment là, la transition ne se déclenche pas, si elle est vraie, la transition se déclenche et ses effets se produisent.

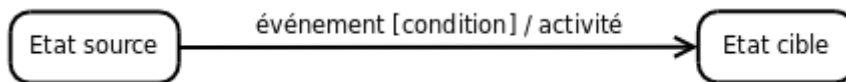
8. Effet d'une transition

Lorsqu'une transition se déclenche (on parle également de tir d'une transition), son effet (spécifié

par '/' <activité> dans la syntaxe) 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.

8.1. Transition externe



Une transition externe est une transition qui modifie l'état actif.

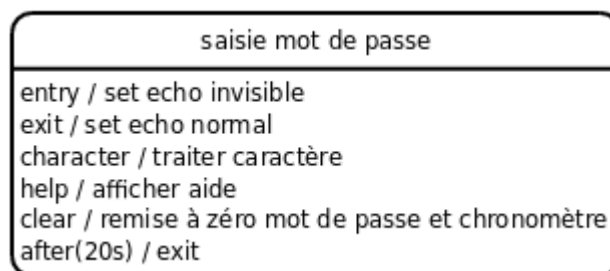
8.2. Transition d'achèvement

Une transition dépourvue d'événement déclencheur explicite se déclenche à la fin de l'activité contenue dans l'état source. Elle peut contenir une condition de garde qui est évaluée au moment où l'activité contenue dans l'état s'achève, et non pas ensuite.

Les transitions de garde sont, par exemple, utilisées pour connecter les états initiaux et les états historiques avec leur état successeurs puisque ces pseudo-états ne peuvent rester actifs.

8.3. Transition interne

Les règles de déclenchement d'une transition interne sont les mêmes que pour une transition externe excepté qu'une transition interne ne possède pas d'état cible et que l'état actif reste le même à la suite de son déclenchement. La syntaxe d'une transition interne reste la même que celle d'une transition classique. Par contre, les transitions internes ne sont pas représentées par des arcs mais sont spécifiées dans un compartiment de leur état associé.



Représentation de la saisie d'un mot de passe

Les transitions internes possèdent des noms d'événement prédéfinis correspondant à des déclencheurs particuliers : **entry**, **exit**, **do** et **include**. Ces mots clefs réservés viennent prendre la place du nom de l'événement dans la syntaxe d'une transition interne.

- **entry** – permet de spécifier une activité qui s'accomplit quand on entre dans l'état.
- **exit** – permet de spécifier une activité qui s'accomplit quand on sort de l'état.
- **do** – Une activité do commence dès que l'activité entry est terminée. Lorsque cette activité est terminée, une transition d'achèvement peut être déclenchée, après l'exécution de l'activité exit bien entendu. Si une transition se déclenche pendant que l'activité do est en cours, cette dernière est interrompue et l'activité exit de l'état s'exécute.

- **include** – permet d’invoquer un sous-diagramme d’états-transitions.

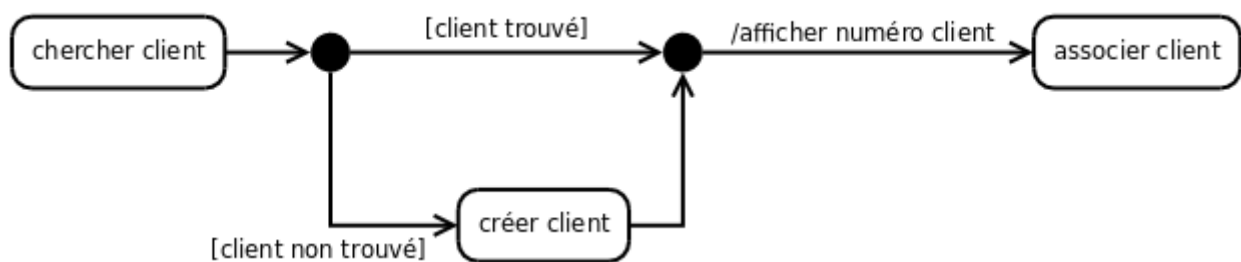
Les activités entry servent souvent à effectuer la configuration nécessaire dans un état. Comme il n’est pas possible de l’éluder, toute action interne à l’état peut supposer que la configuration est effectuée indépendamment de la manière dont on entre dans l’état.

De manière analogue, une activité exit est une occasion de procéder à un nettoyage. Cela peut s’avérer particulièrement utile lorsqu’il existe des transitions de haut niveau qui représentent des conditions d’erreur qui abandonnent les états imbriqués. Le déclenchement d’une transition interne ne modifie pas l’état actif et n’entraîne donc pas l’activation des activités entry et exit.

8.4. Point de jonction

Un point de jonction peut avoir plusieurs segments de transition entrante et plusieurs segments de transition sortante. Par contre, il ne peut avoir d’activité interne ni des transitions sortantes dotées de déclencheurs d’événements.

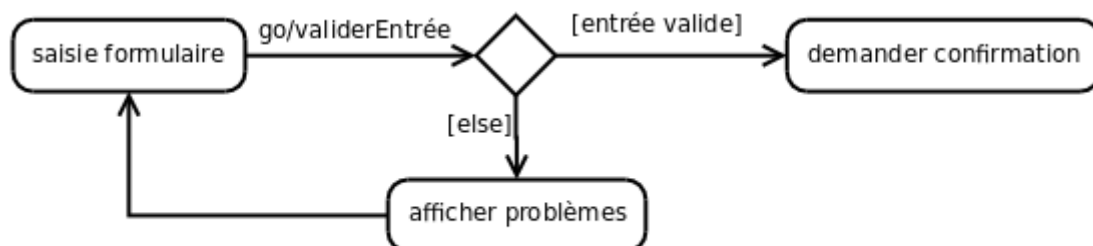
Il ne s’agit pas d’un état qui peut être actif au cours d’un laps de temps fini. Lorsqu’un chemin passant par un point de jonction est emprunté (donc lorsque la transition associée est déclenchée) toutes les gardes le long de ce chemin doivent s’évaluer à vrai dès le franchissement du premier segment.



Exemple d’utilisation de deux points de jonction pour représenter une alternative.

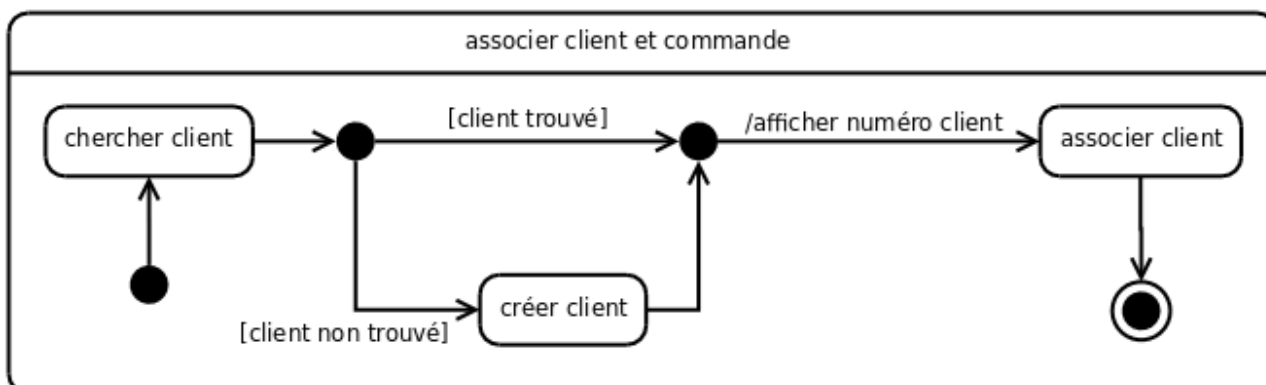
8.5. Point de décision

Un point de décision possède une entrée et au moins deux sorties. Contrairement à un point de jonction, les gardes situées après le point de décision sont évaluées au moment où il est atteint. Cela permet de baser le choix sur des résultats obtenus en franchissant le segment avant le point de choix. Si, quand le point de décision est atteint, aucun segment en aval n’est franchissable, c’est que le modèle est mal formé. Il est possible d’utiliser une garde particulière, notée [else], sur un des segments en aval d’un point de choix. 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.

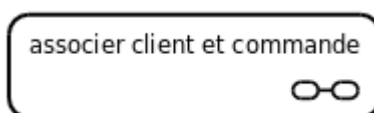


9. États composites

Composés de sous-états, les états composites sont présentés comme suit :

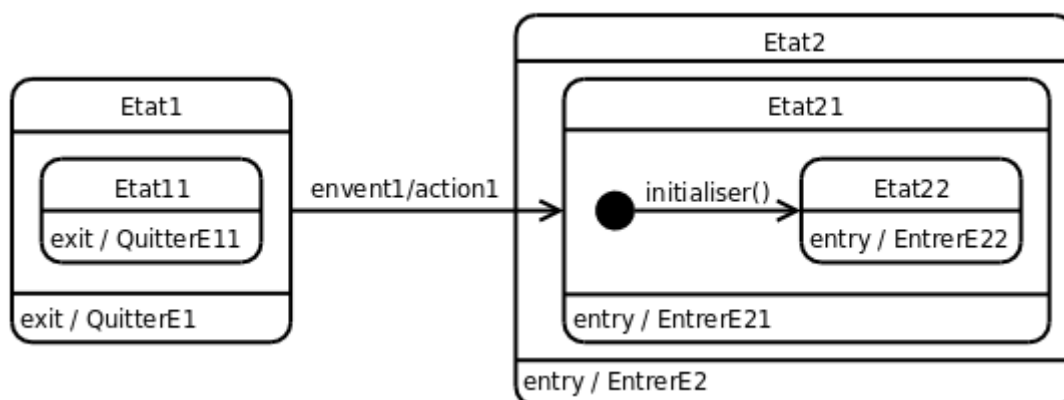


Lorsqu'un état orthogonal est actif, un sous-état direct de chaque région est simultanément actif, il y a donc concurrence. Implicitement, tout diagramme d'états-transitions est contenu dans un état externe qui n'est usuellement pas représenté. Cela apporte une plus grande homogénéité dans la description :



Notation abrégée d'un état composite

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. Une notation abrégée permet d'indiquer qu'un état est composite et que sa définition est donnée sur un autre diagramme.

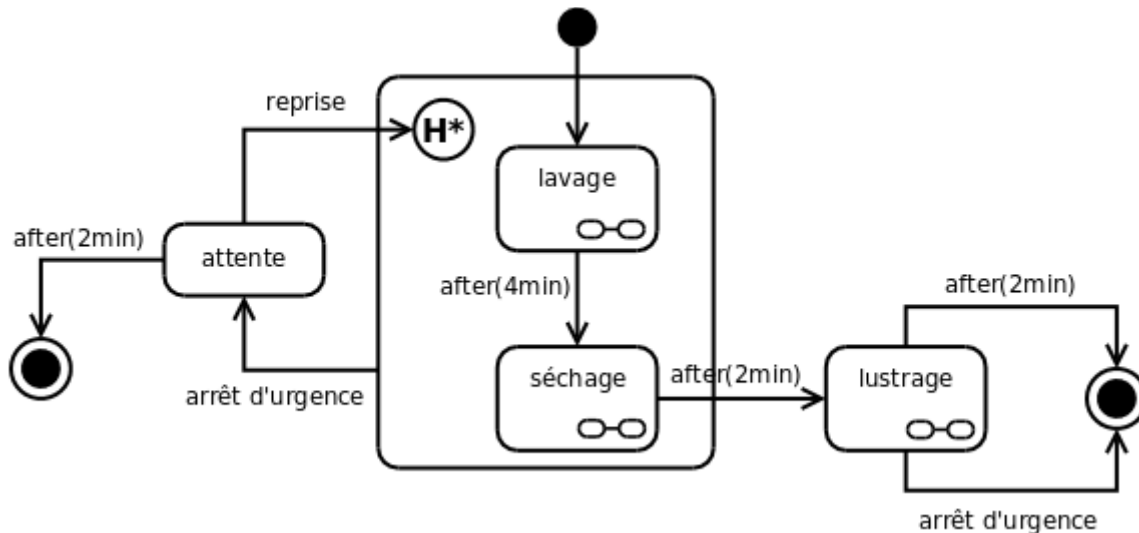


Les transitions peuvent également toucher des états de différents niveaux d'imbrication en traversant les frontières des états composites.

10. État historique

Un état historique, également qualifié d'état historique plat, est un pseudo-état qui mémorise le dernier sous-état actif d'un état composite. Graphiquement, il est représenté par un cercle contenant un H.

Il est également possible de définir un état historique profond représenté graphiquement par un cercle contenant un H*. Cet état historique profond permet d'atteindre le dernier état visité dans la région, quel que soit son niveau d'imbrication, alors que l'état historique plat limite l'accès aux états de son niveau d'imbrication.



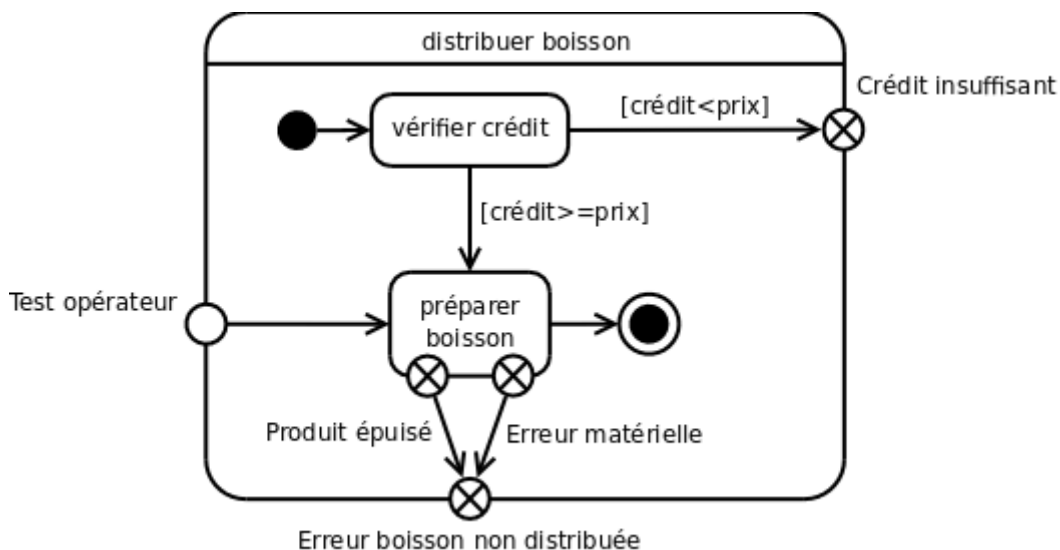
Exemple de diagramme possédant un état historique profond permettant de reprendre le programme de lavage ou de séchage d'une voiture à l'endroit où il était arrivé avant d'être interrompu.

Les états de lavage, séchage et lustrage sont des états composites définis sur trois autres diagrammes d'états-transitions non représentés ici. En phase de lavage ou de séchage, le client peut appuyer sur le bouton d'arrêt d'urgence. S'il appuie sur ce bouton, la machine se met en attente. Il a alors deux minutes pour reprendre le lavage ou le lustrage, exactement où le programme a 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 recommencé. En phase de lustrage, le client peut aussi interrompre la machine. Mais dans ce cas, la machine s'arrêtera définitivement.

10.1. Points de connexion

Il est possible de masquer les sous-états d'un état composite et de les définir dans un autre diagramme. Cette pratique nécessite parfois l'utilisation de pseudo-états appelés « *points de connexion* ».

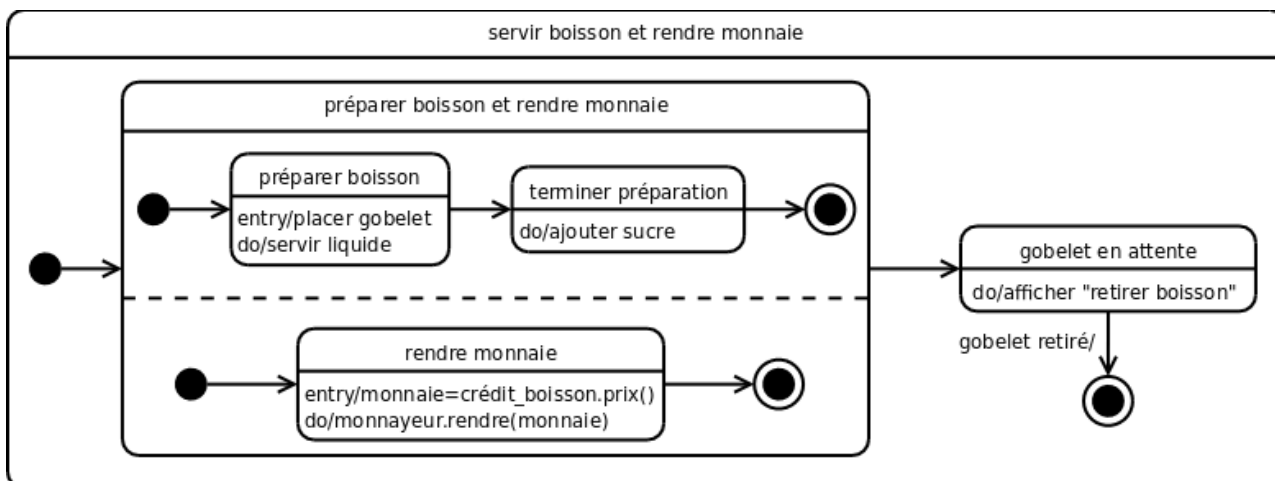
Les points de connexion sont des points d'entrée et de sortie portant un nom, et situés sur la frontière d'un état composite. Ils sont respectivement représentés par un cercle vide et un cercle barré d'une croix. Il ne s'agit que de références à un état défini dans l'état composite. Une unique transition d'achèvement, dépourvue de garde, relie le pseudo-état source à l'état référencé. Cette transition d'achèvement n'est que le prolongement de la transition qui vise le point déconnexion (il peut d'ailleurs y en avoir plusieurs).



10.2. Concurrency

Les diagrammes d'états-transitions permettent de décrire efficacement les mécanismes concurrents grâce à l'utilisation d'états orthogonaux. Un état orthogonal est un état composite comportant plus d'une région, chaque région représentant un flot d'exécution. Graphiquement, dans un état orthogonal, les différentes régions sont séparées par un trait horizontal en pointillé allant du bord gauche au bord droit de l'état composite.

Chaque région peut posséder un état initial et final. Une transition qui atteint la bordure d'un état composite orthogonal est équivalente à une transition qui atteint les états initiaux de toutes ses régions concurrentes. Toutes les régions **concurrentes** d'un état composite orthogonal doivent atteindre leur état final pour que l'état composite soit considéré comme terminé.



Sur ce diagramme, l'état orthogonal préparer boisson et rendre monnaie peut éventuellement ne pas apparaître (tout en gardant la représentation de ses sous-états) pour alléger la représentation, car la notion de concurrence est clairement apparente de par l'utilisation des transitions complexes.

11. Exercice : publiphone

L'utilisation d'un publiphone doit respecter les éléments suivants :

1. Le prix minimal d'une communication interurbaine est de 1€.
2. Après l'introduction de la monnaie, l'utilisateur a 2 min pour composer son numéro (ce délai est décompté par le standard).
3. La ligne peut être libre ou occupée.
4. Le correspondant peut raccrocher le premier.
5. Le publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps (UT) générée par le standard.
6. On peut ajouter des pièces à tout moment.
7. Lors du raccrochage, le solde de monnaie est rendu.

Diagramme de séquences :

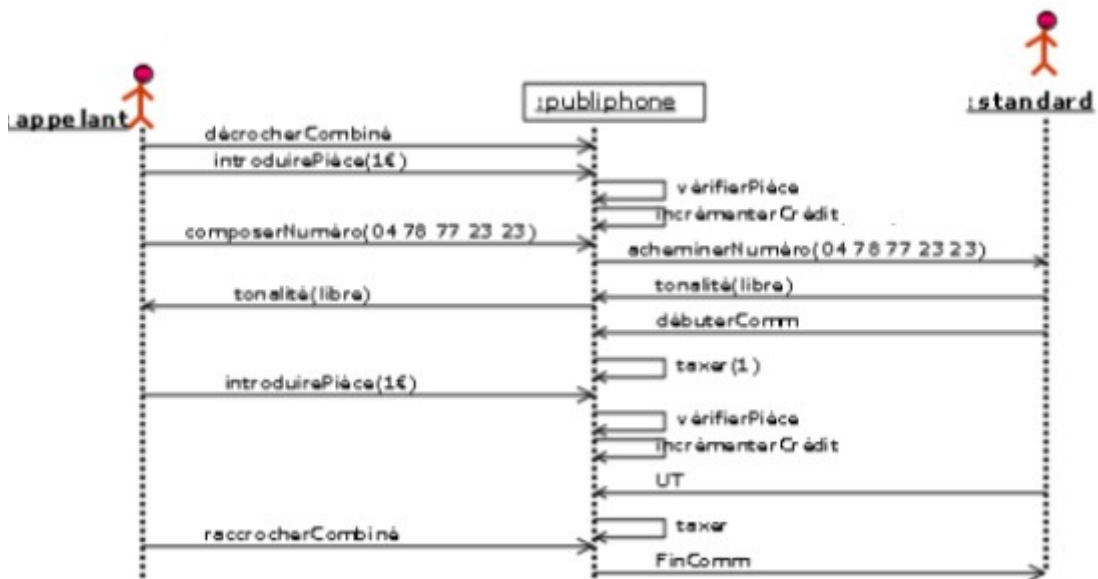


Diagramme d'états-transitions :

