CPGE 1re année

SCIENCES INDUSTRIELLES DE L'INGENIEUR

Modéliser le comportement dynamique des SED par diagramme d'état

CI 11

2019-2020

Objectifs

Connaître et savoir modéliser le comportement dynamique des systèmes à évènements discrets. Savoir interpréter un diagramme d'état.

Sommaire

I Système à événements discrets (SED)

Les **systèmes à événements discrets** (SED)(1) se caractérisent par une **évolution temporelle non continue de l'information** traitée (variables d'entrée / sortie, messages…) et essentiellement **logique**.

(1) Comme pour les SLCI, le terme « système » doit se comprendre comme « modèle ».

Le **comportement dynamique d'un SED** modélise la manière dont les **activités réalisées par un système sont coordonnées temporellement** et comment elles dépendent **d'évènements ponctuels** (en temps), extérieurs, internes ou d'actions d'un utilisateur à travers une IHM. Le SED peut être un :

- **Système logique combinatoire**
- **Système séquentiel**.

Une activité est souvent associée à une chaîne fonctionnelle (chaîne d'information et chaîne de puissance), mais peut aussi être un traitement purement numérique d'informations. Il s'agit ici d'une définition très large.

Un système **logique combinatoire** est un système dont la PC traite des informations sous forme logique et pour lesquelles les états en sortie sont exclusivement définis à partir de leurs états en entrée.
De tels systèmes n'utilisent aucune solution de mémorisation (mémoire).

Dans un système **séquentiel**, la ou les sorties dépendent de la combinaison des entrées et de l'état précédent des sorties.
Ainsi :

− une même cause (même combinaison des entrées) peut produire des effets différents ; − le temps peut être une cause déclenchante ; − l'effet peut persister si la cause disparaît.

Télécommande Smart Touch de Samsung

Cette modélisation est utilisée d'un point de vue fonctionnel ou en phase de conception. Mais, les outils permettent aussi :

- la **simulation numérique** ;
- la **programmation** de **l'unité de commande** du système étudié.

Les constructeurs d'automates programmables proposent des interfaces de programmation basés sur ces modélisations et des compilateurs dédiés.

En SysML(2), le comportement dynamique d'un SED est représenté par :

- le **diagramme d'état** (*state machine diagramme* ou *stm*),
- le **diagramme de séquence** (*sequence diagram*, ou *seq*).

(2) Il existe d'autres outils pour modéliser le programme d'un système à événement discret ou décrire son comportement séquentiel

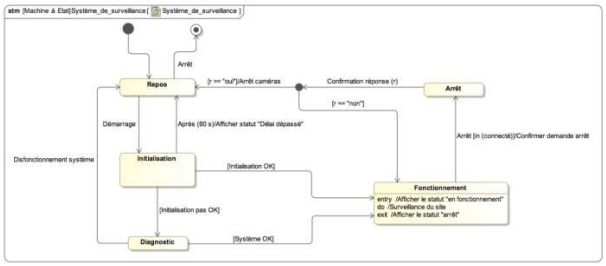


Diagramme d'état

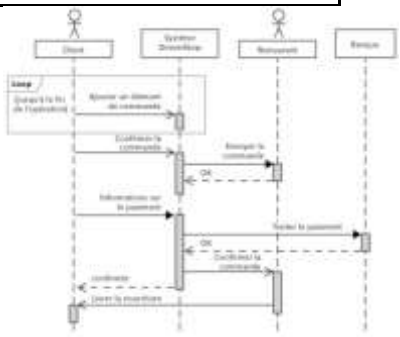
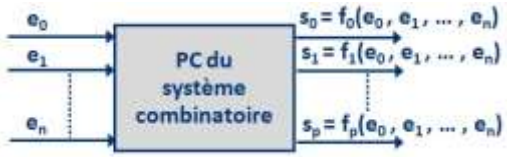


Diagramme de séquence

II Logique combinatoire et système logique combinatoire

II.1 Système logique combinatoire

Dans un système logique combinatoire, les états des sorties s'expriment comme une combinaison des états des entrées.



Les états des sorties (s_0, s_1, … s_n) sont contraints par des **fonctions logiques ne dépendant que des états des entrées** (e_0, e_1, … e_n).

II.2 Variable binaire

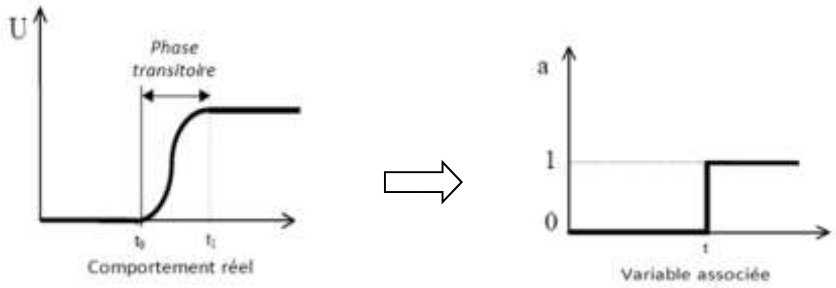
Une **variable binaire** (TOR = Tout Ou Rien) ne peut prendre que **deux états notés**(1) en général 0 ou 1.

(1) On dira aussi « vrai ou faux », « oui ou non » …

En pratique, cela correspond aux deux états que peut prendre un composant dans un système automatisé : interrupteur ouvert ou fermé, lampe allumée ou éteinte, bouton poussoir actionné ou relâché, moteur en marche ou à l'arrêt, tige de vérin sortie ou rentrée…

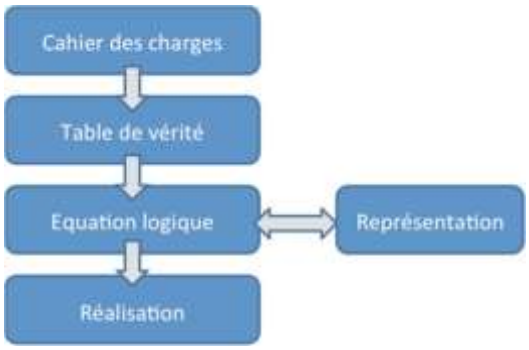
L'association d'une variable binaire à un composant est un modèle simple du comportement réel.

En effet, le passage d'un état à un autre ne peut se faire sans une phase transitoire qui est alors négligée.



II.3 Fonction logique

A partir du cahier des charges qui décrit de manière littérale ce qui est attendu du système combinatoire jusqu'à sa réalisation matérielle câblée et/ou programmée, la démarche peut être résumée ainsi :



Expression d'une fonction logique

La fonction est explicitée sous la forme d'une phrase qui décrit le fonctionnement attendu du système logique combinatoire.

Table de vérité

Elle permet de faire correspondre, sous la forme d'un tableau, l'état des sorties en fonction de l'état des entrées et cela pour toutes les combinaisons(2) possibles des états des entrées.

(2) Pour un système à n entrées, il y aura 2^n combinai-sons possibles.

Equation logique

L'équation d'une sortie peut être élaborée à partir de la table de vérité traduisant le cahier des charges. Cette équation qui fait intervenir l'état des entrées et l'état de la sortie utilise des opérateurs logiques selon les règles de ***l'algèbre de BOOLE***.

Ces 4 opérateurs logiques sont : **OUI**, **NON**, **OU**, **ET**.

Les 4 opérations de base sont :

Entrées : **a** et **b** Sortie : **S**			
OUI	notée $S = a$	*qui donne à S la valeur 1 si et seulement si*	a=1
NON (appelé aussi « complément »)	notée $S = \bar{a}$		a=0
OU (appelé aussi « somme logique »)	notée $S = a+b$		a=1 OU b=1
ET (appelé aussi « produit logique »)	notée $S = a \cdot b$		a=1 ET b=1

➢ *le symbole = ne traduit pas une égalité mais une identité d'état*

Les règles de l'algèbre de Boole sont :

Propriétés de la **somme** logique : **OU**	
$0 + 0 = 0$	$a + 1 = 1$
$0 + 1 = 1$	$a + 0 = a$
$1 + 0 = 1$	$a + a = a$
$1 + 1 = 1$	$a + \bar{a} = 1$

Propriétés du **produit** logique : **ET**	
$0 \cdot 0 = 0$	$a \cdot 1 = a$
$0 \cdot 1 = 0$	$a \cdot 0 = 0$
$1 \cdot 0 = 0$	$a \cdot a = a$
$1 \cdot 1 = 1$	$a \cdot \bar{a} = 0$

➢ ***L'opérateur ET est prioritaire par rapport à l'opérateur OU.***

involution	absorption
$\bar{0} = 1$ $\bar{1} = 0$ $\bar{\bar{a}} = a$	$a + a.b = a$ $a \cdot (a + b) = a$

commutativité	associativité	distributivité
$a \cdot b = b \cdot a$ $a+b = b+a$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c$ $a+(b+c) = (a+b)+c = a+b+c$	$a \cdot (b+c) = a \cdot b + a \cdot c$ $a+(b \cdot c) = (a+b) \cdot (a+c)$

Identité remarquables
$a + \bar{a}.b = a+b$ $(a+b) \cdot (\bar{a}+c) = a \cdot c + \bar{a} \cdot b$

Théorèmes de De Morgan(1) :

$$\overline{(a + b)} = a \cdot b \quad et \quad \overline{(a.b)} = a+b$$

En utilisant les opérateurs logiques et les règles de l'algèbre de Boole, **l'équation logique d'une sortie** est alors obtenue sous sa **forme canonique** de la façon suivante :

- pour **chaque ligne** de la table de vérité où la **sortie** vaut **1**, déterminer la combinaison d'entrées correspondante en utilisant l'opérateur **ET**,
- puis **sommer** ces combinaisons en utilisant l'opérateur **OU**.

Le **diagramme d'état** (2) est un diagramme SysML **associé à un bloc** d'un diagramme de définitions de blocs (bdd) ou d'un diagramme de blocs internes (ibd).

Il décrit les **différents états** pris par le bloc, les **activités réalisées** pendant ces états ainsi que les **évènements** et **conditions** nécessaires pour passer une **transition** d'un état à un autre.

(1) Ces deux théorèmes peuvent être généralisés à n variables : $\overline{\sum_{i=1}^{n} e_i} = \prod_{i=1}^{n} \bar{e}_i$ *et*

$$\overline{\prod_{i=1}^{n} e_i} = \sum_{i=1}^{n} \bar{e}_i =$$

(2) proche du concept informatique de machine d'état.

Chaque bloc d'un bdd ou d'un ibd ne conduit pas nécessairement à un diagramme d'état.

III Diagramme d'état : *state machine diagram* ou *stm*

III.1 Définition et exemple

Les **états** correspondent à des **situations de fonctionnement** différentes (marche, arrêt, en montée, en attente…).

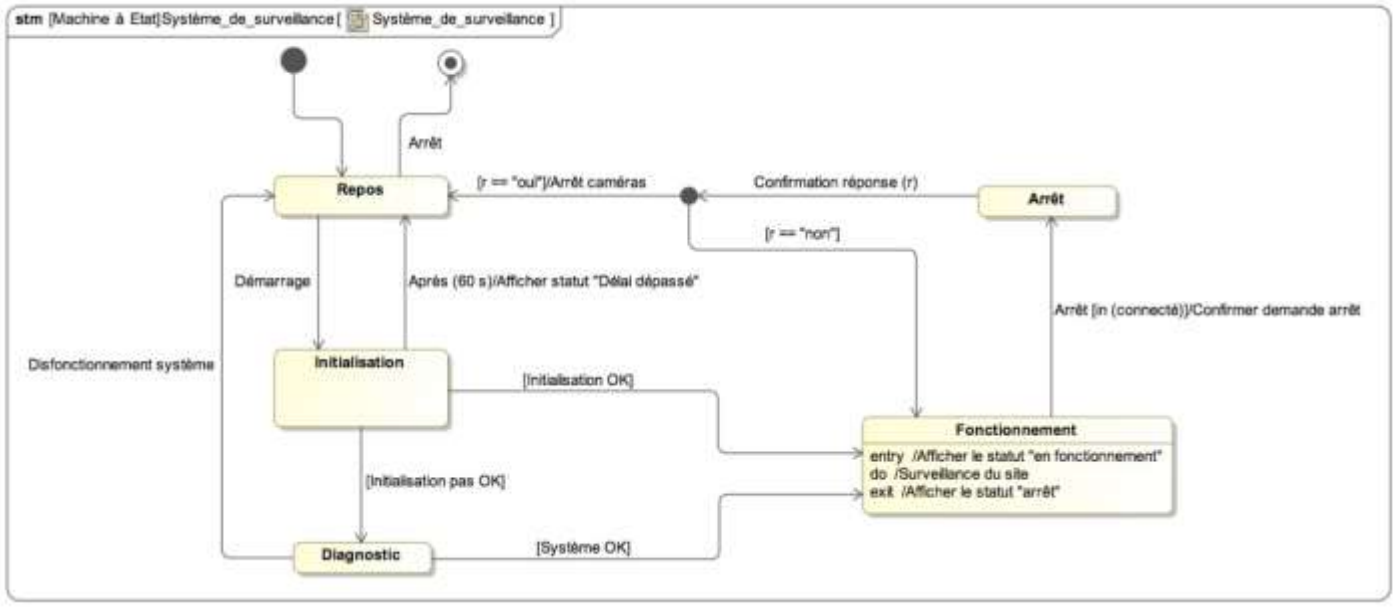
Les **évènements** sont des actions des utilisateurs, de l'environnement ou la réponse d'un capteur, conditionnant le passage (transition) d'un état à un autre et la mise en route et l'arrêt des activités.

Système de vidéo-surveillance

Exemple : le diagramme ci-dessous décrit le fonctionnement d'un système de vidéo-surveillance. On y trouve :

- 5 états : *Repos* , *Initialisation*, *Diagnostic*, *Arrêt* et *Fonctionnement* ;
- 3 pseudo-états :
  - initial : ,
  - final : ,
  - point de jonction ;
- des **transitions** entre les états, représentées par des flèches, et qui précisent sous quelle **condition** le système passe d'un état à un autre.

Dans cet exemple, la représentation du comportement est fonctionnelle. Aucune information technique sur la manière dont sont transmises les informations, ou sur la façon dont sont réalisées les activités, n'est précisée.



III.2 Démarche de modélisation du comportement dynamique par un diagramme d'état

Une démarche usuelle de modélisation est la suivante :

Étape 1
- Définir la **frontière** du système, **recenser les variables d'entrée** (consignes de l'utilisateur, grandeurs physiques acquises par les capteurs) **et de sortie** (messages à destinations de l'utilisateur ou d'autres systèmes et ordres vers les préactionnneurs).

Étape 2
- **Recenser**, **nommer** et tracer **les différents états** dans lesquels peut se retrouver le système lors de son utilisation. **Identifier les activités associées** à chaque état.

Étape 3
- **Identifier** et tracer **les transitions** possibles entre les états en fonction du comportement séquentiel souhaité ou observé.

Étape 4
- **Définir les conditions** et **événements** déclencheurs associés à chaque transition et qui autorisent son franchissement.

IV État et activités associées

(1) ensemble des étapes de la vie du système, de son démarrage jusqu'à son arrêt.

Un état possède un titre, **unique dans le diagramme**.

> Un **état** modélise une **phase du fonctionnement**(1) du système.
>
> Pendant cette période, l'état est dit **actif** et le système accomplit **une activité simple, une séquence d'activités** ou **est en attente.** Une activité peut être **instantanée** ou être **interruptible**.
>
> En dehors de cette période, l'état est dit **inactif**.

> Par définition, il n'y a qu'**un seul état actif** à chaque instant.

Le lancement des activités à l'intérieur de l'état actif est organisé selon des mots réservés :

entry	L'entrée dans l'état déclenche l'exécution de l'activité associée. C'est impérativement une **activité instantanée**, c'est-à-dire qu'elle **ne peut pas être interrompue.**
do	Activité(s) à exécuter dans l'ordre de leur écriture, à partir de l'instant où l'activité associée à *entry* est terminée. Les activités associées au *do* sont **interruptibles**.
exit	La sortie de l'état qui va entrainer sa désactivation, entraine l'interruption des activités associées à *do* et l'exécution de celle associée à *exit.* C'est impérativement une **activité instantanée**, c'est-à-dire qu'elle **ne peut pas être interrompue.**

Remarques :

- si aucun mot réservé n'est utilisé, cela correspond à un *do* ;
- les trois comportements *entry*, *do* et *exit* ne peuvent être **utilisés qu'une seule fois par état**, mais il est également possible de n'en utiliser qu'une partie (seulement *entry* par exemple) ;
- un **état « vide »** (sans activité) indique un **état d'attente** ;
- il est possible de placer dans un état des activités associées à un événement ou à une garde (voir plus loin).

V Transition : événement déclencheur, condition de garde et effet associé

V.1Transition

> Une **transition** modélise la possibilité d'un **passage instantané d'un état source vers un état cible**. Elle est :
>
> − orientée et sans durée ;
>
> − **évaluée** seulement **si l'état source est actif** ;
>
> − associée à des **conditions de franchissement** (optionnelles) : **évènements** et **conditions de garde** ;
>
> − associée à une activité instantanée (optionnelle) appelée **effet**.

événement [garde] / effet

V.2 Événement

> Un **événement** est un évènement (au sens des SED) **daté** dans le temps.
> Il est le **déclencheur** du **franchissement** de la transition.

Exemples : appui sur un bouton, arrivée en fin de course d'un mécanisme, dépassement d'une valeur seuil

> Un évènement est **traité instantanément lors de son occurrence** (apparition).
> Sauf indication contraire, c'est le **front montant** (passage de 0 à 1) qui est considéré.

Un évènement n'est pas caractérisé par une durée : il a lieu ou non. **Il n'est jamais mémorisé** et est **donc « perdu » s'il ne mène à aucune évolution du diagramme d'état**.

On distingue trois types d'évènement :

- l'événement de **signal** prend en compte l'envoi ou la réception d'un signal : appui sur un bouton, arrivée en fin de course d'un mécanisme ;
- l'événement de **changement** prend en compte le changement d'une valeur interne du modèle. Il se modélise avec le mot clé `when` suivi d'une expression booléenne concernant une variable interne : *compteur when (N=3)* ;
- l'événement **temporel** :
  - relatif : `after (T)` se déclenche après une durée T passé dans l'état d'amont. Il s'agit d'une temporisation ;
  - absolu : `at(D)` se déclenche à la date D dans un référentiel de temps dont l'origine correspond généralement au démarrage du fonctionnement du système.

Si plusieurs événements au choix sont écrits, ils sont séparés par une virgule. Une transition peut ne pas avoir d'événement.

V.3 Garde

> La garde est une **condition de franchissement de la transition.** C'est une condition **booléenne** supplémentaire à l'événement déclencheur.

Contrairement à l'événement qui, lui, est localisé dans le temps, la **garde traduit une condition qui dure dans le temps** et **qui doit persister** : vitesse non nulle, température >20°….

Exemple : pour un bouton :

- l'événement est associé à l'instant où le bouton est enfoncé ; -
la garde est associé à l'état du bouton : enfoncé ou non.

> La syntaxe d'une condition de garde vérifiant l'activité de l'état TOTO est : `[in TOTO]`.

V.4 Les étapes du franchissement

Lors du franchissement d'une transition d'un état source à un état cible : l'éventuelle activité *do* de l'état source est **interrompue** ;

↓

l'éventuelle activité *exit* de l'état source est exécutée ;

↓

l'état source devient inactif ;

↓

l'éventuelle activité *effet* est exécutée ;

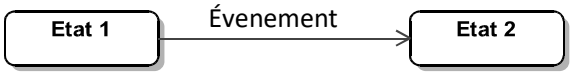
↓

l'état cible devient actif.

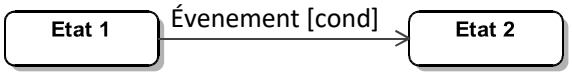
↓

l'éventuelle activité *entry* de l'état cible est exécutée ;

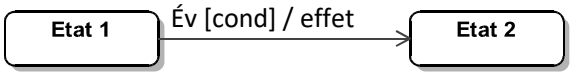
V.5 Principaux cas de franchissement d'une transition



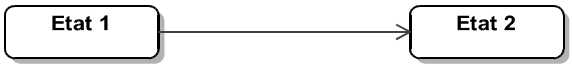
À l'occurrence (apparition) de l'événement, la transition est franchie, sans condition.



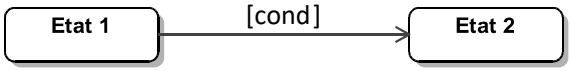
À l'occurrence de l'événement, si la garde « cond » est vraie, la transition est franchie.
Sinon, **l'événement est « perdu »** et **il faut attendre une deuxième occurrence** pour éventuellement franchir la transition.



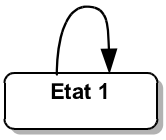
À l'occurrence de l'événement, si la garde « cond » est vraie, la transition est franchie :
les activités associées à l'effet sont effectuées après la désactivation de l'état source 1 (voir différentes étapes ci-dessus).



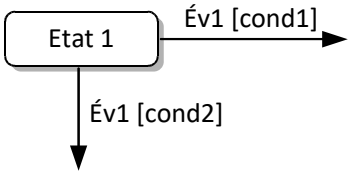
La transition est dite automatique. Elle est :
- immédiatement franchie s'il n'y a pas d'activité associée à l'état 1 ;
- franchie dès la fin de l'éventuelle activité associée au comportement do de l'état 1 afin de provoquer l'événement déclencheur implicite. **Il faut donc que cette activité do ait une fin**.



Si la garde « cond » est vraie, la transition est :
- immédiatement franchie s'il n'y a pas d'activité associée à l'état 1 ;
- franchie dès la fin de l'éventuelle activité associée au comportement do de l'état 1 afin de provoquer l'événement déclencheur implicite. **Il faut donc que cette activité do ait une fin**.

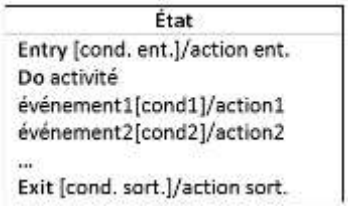


Évenement [cond] Une transition réflexive entraîne une sortie de l'état puis un retour dans ce même état, avec appel des éventuelles activités exit et entry.



Plusieurs transitions peuvent quitter un même état.
Une seule d'entre elles doit être déclenchée ; **les événements et/ou les conditions de garde doivent donc être exclusives**.

Remarque : il est possible de placer, dans un état, des activités associées à un événement ou/et à une garde.



Notons cependant qu'il est déconseillé d'utiliser ces transitions que l'on peut qualifier d'internes. Il est plutôt recommandé de changer d'état lors de l'apparition des événements.

VI Pseudos-états

Un *pseudo-état* est un état ne pouvant avoir d'activités.

Ils servent essentiellement comme éléments de liaisons et pour indiquer l'état final ou l'arrêt du diagramme d'état.

VI.1 Pseudo-état initial

: unique et obligatoire, il est activé au lancement de la machine à états et marque le début de l'exécution du diagramme d'état. Il n'a aucune transition entrante

VI.2 Pseudo-état final

: optionnel, signe la fin de l'exécution du diagramme d'état.

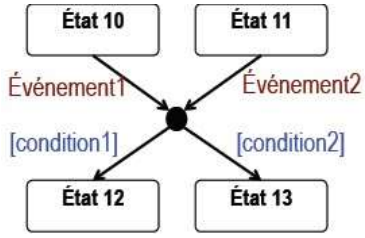
Lorsque cet état est activé, la machine à états s'arrête. Il n'a donc aucune transition sortante. Il peut y en avoir plusieurs car différents scénarios peuvent être possibles pour mettre fin à un comportement.

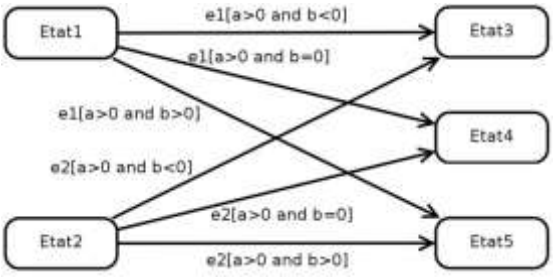
VI.3 Pseudo-état de jonction

: utilisé pour regrouper (« factoriser ») des conditions de franchissement de transitions, en particulier des gardes communes à un événement.

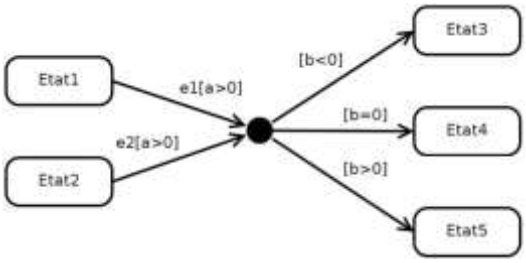
Cela permet de partager des segments de transition et d'aboutir à une notation plus lisible des chemins alternatifs. L'évaluation des conditions de garde en aval du pseudo-état est réalisée avant qu'il ne soit atteint.

Exemple ci-contre, on passe de l'état 10 à l'état 13 si l'événement 1 apparait et que la condition 2 est vrai.

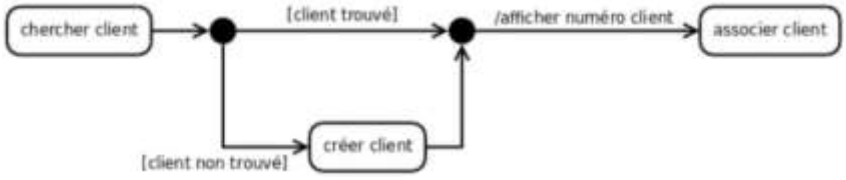




Exemple sans point de jonction



Exemple avec point de jonction



Exemple avec état conditionnel

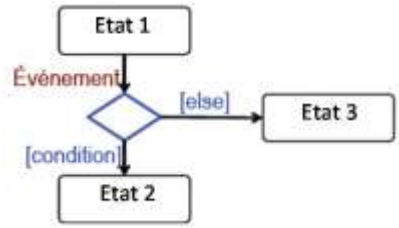
VI.4 Pseudo-état de choix

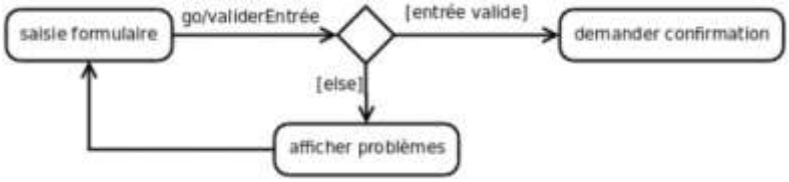
 : utilisé pour une sélection ou une convergence de séquences exclusives.

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, en particulier lorsqu'un effet (activité) est placée dans celui-ci.

Les conditions de gardes doivent être exclusives. L'utilisation d'une clause [else] est recommandée après un point de décision, car elle garantit un modèle correct en englobant tout ce qui n'est pas décrit dans les autres expressions booléennes et en assurant ainsi qu'un moins un segment en aval est franchissable.

Exemple : dès que l'événement apparaît, le pseudo-état de choix est atteint. Si la condition est vraie, c'est l'état 3 qui devient actif, sinon, c'est l'état 2





VII État composite

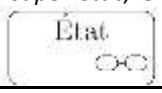
Un **état composite** (ou **super-état**) décrit les **évolutions internes d'un état** à l'aide **d'autres diagrammes d'états**.

Il permet de **hiérarchiser** les états et de modéliser des **évolutions simultanées** d'états.

VII.1 Etat composite de hiérarchisation et historique

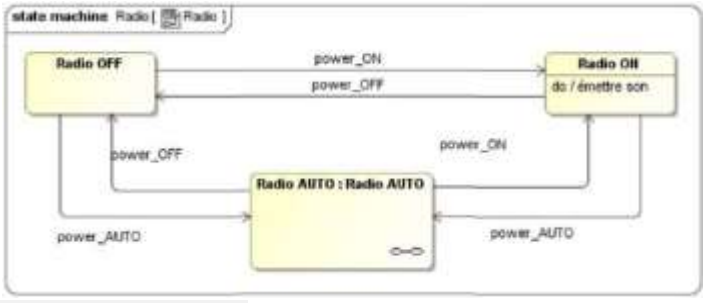
Pour repérer un état composite de hiérarchisation, un signe symbolisant des lunettes est apposé sur l'état

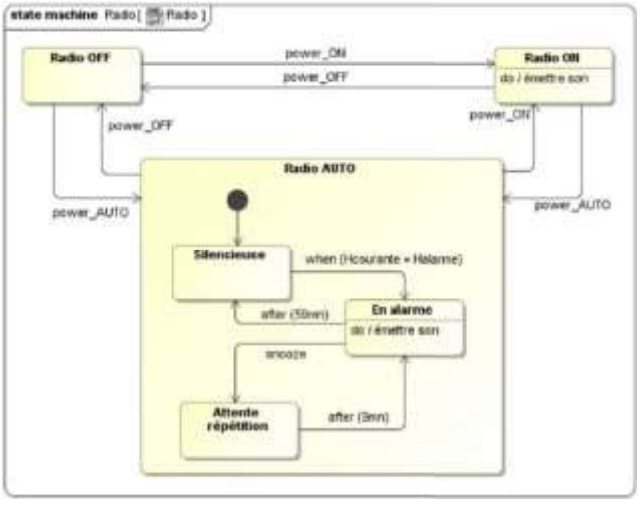
Un état **composite de hiérarchisation** , appelé parfois *super-état*, **englobe un sous-diagramme**.



Cette structure qui permet d'englober plusieurs sous-états exclusifs qui sont considérés comme *hiérarchiquement inférieur* au diagramme principal, permet de rendre ce dernier plus lisible en entrant séparément dans le détail des évolutions internes du système.

Radio réveil

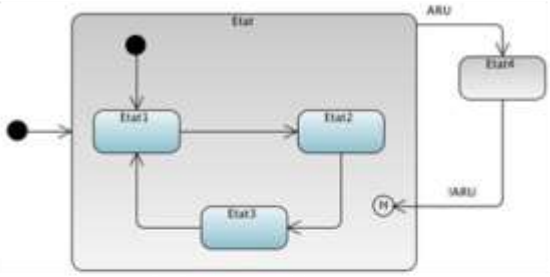




L'indication **historique** permet de **mémoriser l'état actif au moment de la sortie** état composite. (H)

Lors de la désactivation de l'état composite, **l'état actif situé au même niveau hiérarchique est mémorisé**. Lors de la réactivation de l'état composite, la machine d'état se réactive au niveau de l'état composite.

L'*historique* est particulièrement utilisé pour **permettre à un système de recommencer en cours de cycle lors du redémarrage** après un appui sur stop ou l'arrêt d'urgence.



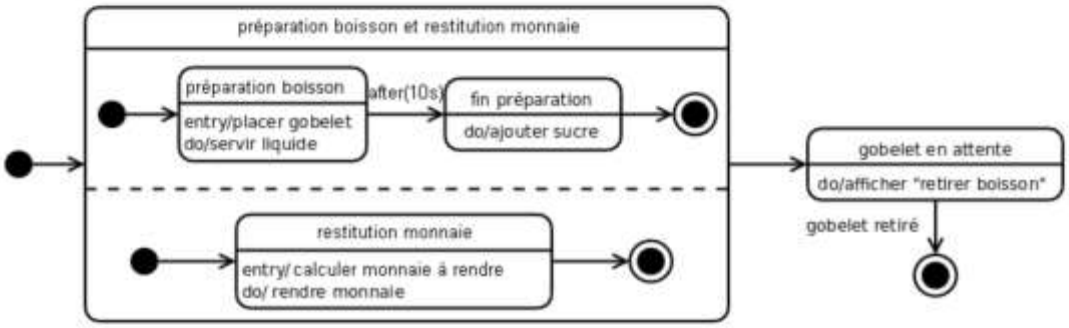
Historique et arrêt d'urgence (ARU)

VII.2 État composite orthogonal

Un **état composite orthogonal** permet de modéliser des systèmes réalisant **plusieurs activités simultanément**.

Il comprend plusieurs **diagrammes évoluant simultanément** (en parallèle) dans des **régions séparées**. **Plusieurs états sont actifs** simultanément.

Chaque région peut posséder un peudo-état initial, un pseudo-état final ou un historique.



Distributeur de boisson

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.

Toutes les régions d'un état composite orthogonal doivent atteindre leur état final pour que l'état composite soit considéré comme terminé. La synchronisation est alors automatique et la transition de sortie de l'état composite est déclenchée.

Il est également possible d'utiliser des **transitions constituées de barres de synchronisation *fork* et *join***.

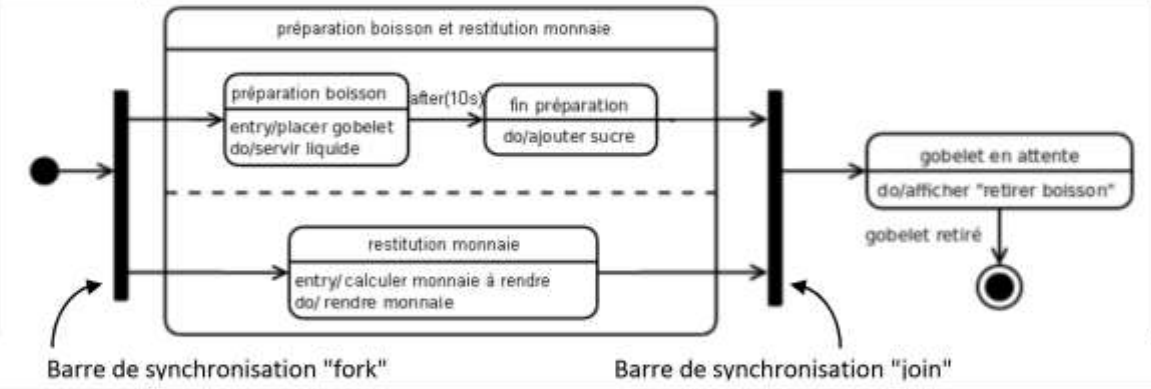


Diagramme d'état de la machine à café équivalente à celui au-dessus

Les transitions automatiques (sans événement ou condition de garde) qui partent d'une barre de synchronisation "fork" se **déclenchent en même temps**.

Une barre de synchronisation join n'est **franchie** qu'**après franchissement de toutes les transitions qui s'y rattachent**.