PTSI	AUTOMATIQUE CHAP.3.1 : SYSML DIAGRAMME D'ETATS	COURS

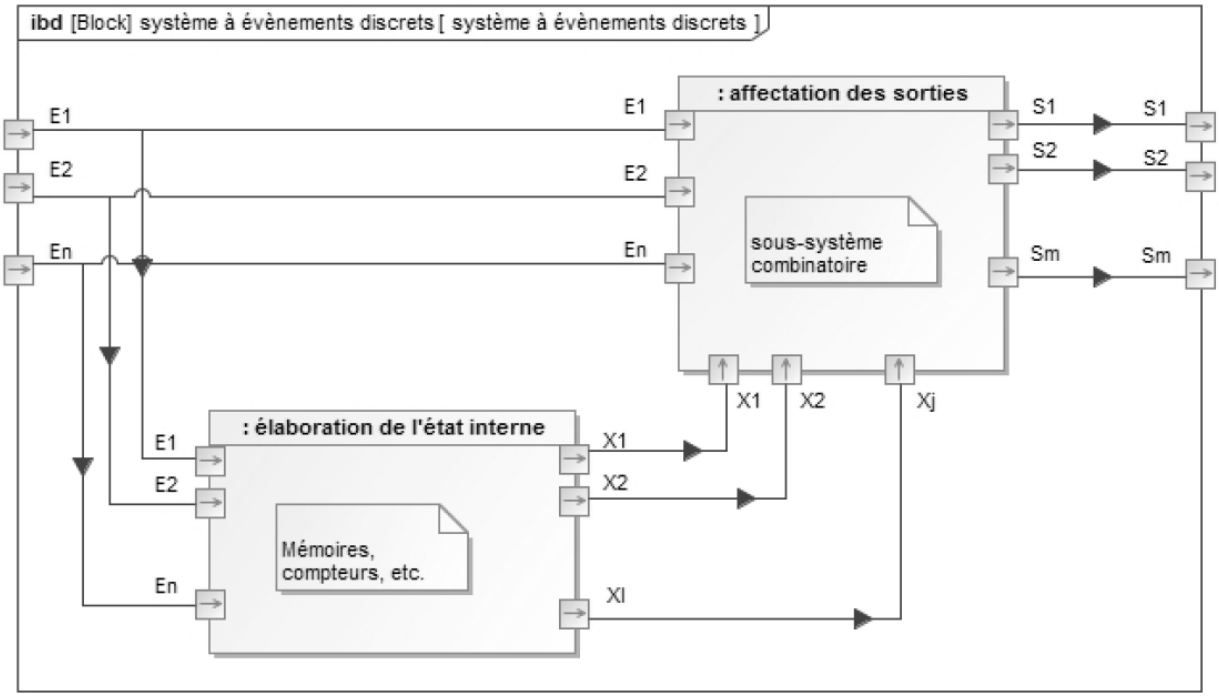
I. INTRODUCTION

L'étude et de la réalisation des diagrammes d'états concerne les ***Systèmes à Evènements Discrets*** (SED).
Dans un système « continu » ou « dynamique », l'état du système change en permanence au cours du temps.
Dans un ***système à événements discrets***, l'état change seulement à certains instants lors de l'occurrence d'évènements ponctuels (transitions).
Les SED apparaissent de façon naturelle dans la modélisation des systèmes informatiques, des réseaux de télécommunications, des réseaux de transport ou des systèmes de production (lignes d'assemblage, ateliers flexibles).

1. Les systèmes à contrôle logique

Dans ce type de système, les signaux à contrôler sont exclusivement des **signaux logiques**. Le système obéit à un processus pré-établi qui envisage toutes les possibilités d'évolution.
Le signal de sortie est élaboré à partir d'un signal ou d'une combinaison de signaux d'entrée logique et prend en compte une chronologie qui porte sur un nombre fini d'opérations. Ce sont les **automatismes séquentiels**. *Une même cause peut produire des effets différents. Un effet peut rester maintenu alors même que la cause a disparu*.
Par opposition aux **systèmes à logique combinatoire** (voir cours automatique Chap.2) *La même cause produit toujours le même effet, l'effet disparait quand la cause disparait.*

Le mode de fonctionnement est décrit sur l'ibd suivant :



Sur le schéma ci-dessus, 3 types de variables sont utilisées :

- Les entrées notées E
- Les sorties notées S
- Les variables internes notées X (pour l'élaboration de l'état interne, les états des sorties sont aussi pris en compte et permettent une étude simplifiée)

Pour exprimer l'état des sorties, il faut disposer d'informations sur l'historique de l'état des variables d'entrée et/ou de sortie.

2. Etude des systèmes à événements discrets (automatismes séquentiels)

La première étape de toute étude des SED comporte une analyse du comportement du système par la détermination de :

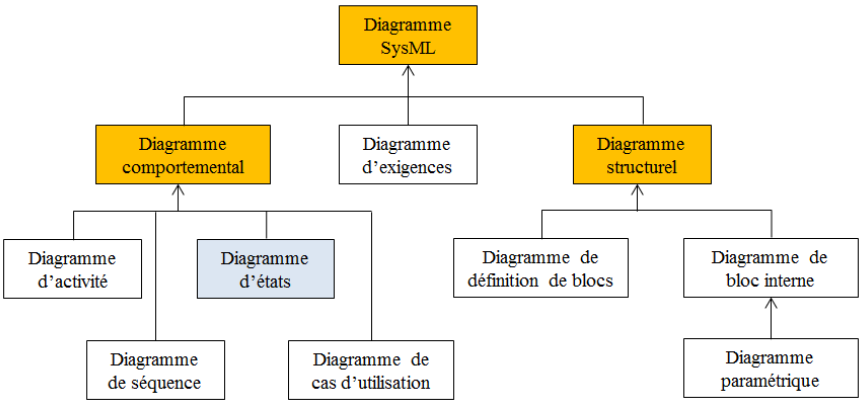
- L'**état initial**
- L'ensemble des **évènements**
- L'ensemble des **états**
- L'ensemble des **activités**
- L'ensemble des **règles**

Dans un état, un système peut être en attente ou avoir une activité. Les états d'un système se succèdent en fonction des évènements.

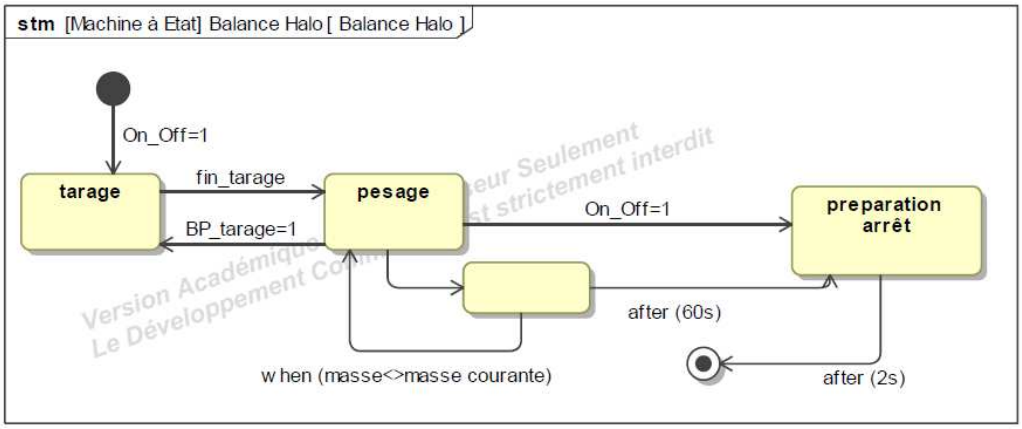
II. LE DIAGRAMME D'ETATS

1. Définition

Le diagramme d'états (state machine diagram : stm) est un diagramme comportemental normalisé SysML.
Il illustre les changements d'états d'un système ou d'un sous-système. Il décrit les transitions entre les états.
Il décrit les actions qu'un système réalise en réponse à des événements.



Exemple du diagramme d'états « description élémentaire » de la balance Halo :



III. ELEMENTS DU DIAGRAMME D'ETATS

1. Etat initial / état final

L'état initial, point d'entrée dans le diagramme, correspond à la création de l'instance de bloc pour lequel le diagramme d'états est spécifié.
L'état final, point de sortie, correspond à la désactivation du diagramme. Il peut y en avoir plusieurs dans un diagramme d'états ou aucun. Plusieurs scénarios peuvent être possibles pour mettre fin au comportement.
Ce sont des pseudo-états aucune action n'est réalisée en ces deux points.

2. Etats

Chaque état est représenté sous la forme d'un rectangle aux coins arrondis contenant son **nom**.
L'état est soit actif soit inactif.
Le lancement des actions à l'intérieur d'un état actif est organisé selon les mots clés :

- Entry/ est suivi des actions exécutées lorsque l'état devient actif ;
- Do/ est suivi d'une ou plusieurs activités exécutées dans l'ordre de leur écriture ;
- Exit/ est suivi des actions qui se déroulent lorsque l'état se désactive.

Pendant que l'état est actif, un évènement peut lancer une action avec la syntaxe : évènement[condition de garde]/action. Cette action est lancée chaque fois que l'évènement survient tant que l'état est actif.

3. Activité et action

Une activité est considérée comme une unité de comportement. Elle prend du temps et peut être interrompue. Elle suit le mot clé « do ».
Une action ne prend pas de temps et ne peut pas être interrompue. *Exemple* : incrémenter ou décrémenter un compteur, émission d'un ordre pour un préactionneur. On peut la trouver dans les transitions (effet) ou dans les états : mots clé « entry » ou « exit ».
Dans l'outil SysML, les activités et actions sont directement liées aux fonctions (opérations) définies dans les propriétés des blocs.

4. Transition : évènement, condition de garde, effet

Une transition représente le passage instantané d'un état source à un état destination. Elle n'est évaluée que si l'état source est actif.

Une transition peut être associée à un **évènement** (trigger), à une **condition de garde** [guard], à un **effet** (/action).
Une transition peut être « vide » -pas de texte associé -, c'est une transition automatique quand l'activité de l'état actif est finie (appelée aussi transition de complétion).
Une transition réflexive entraîne une sortie d'état puis un retour dans le même état.

Il existe 4 types d'évènements associés à une transition :

- Le **message** (signal event) : un message (asynchrone) est arrivé ;
- L'**évènement temporel** (time event) : une temporisation (mot clé « after ») ou un temps absolu est atteint (mot clé « at ») ;
- L'**évènement de changement** (change event) : une valeur a changé (mot clé « when ») ;
- L'**évènement d'appel** (call event) : une demande a été faite, la réponse est attendue (requête).

Remarque : un évènement peut être émis par un autre diagramme d'états.

En plus de spécifier un évènement précis, on peut conditionner la transition à une « **condition de garde** ». C'est une expression booléenne faisant intervenir les entrées et les variables internes. Elle est évaluée lorsque l'état précédant la transition est actif et que l'évènement déclencheur se produit. Si la condition de garde est vraie, la transition est franchie, sinon elle ne l'est pas et l'évènement est perdu.

Un **effet** est une **action.** *Rem : l'action peut être inscrite dans la transition, être l'action liée au mot clé « exit » de l'état précédant la transition ou être l'action liée au mot clé « entry » de l'état suivant.*

IV. STRUCTURES DU DIAGRAMME D'ETATS

1. Choix d'un état cible à partir d'un état source (divergence en OU)

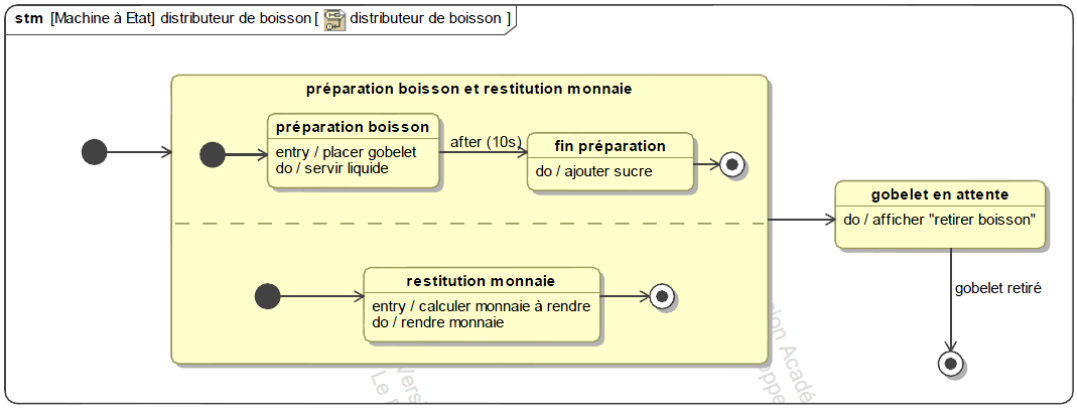
Plusieurs transitions peuvent quitter un même état. Une seule d'entre elles doit être déclenchée ; les évènements et/ou les gardes doivent être exclusives.

Une autre représentation est possible : le point de décision

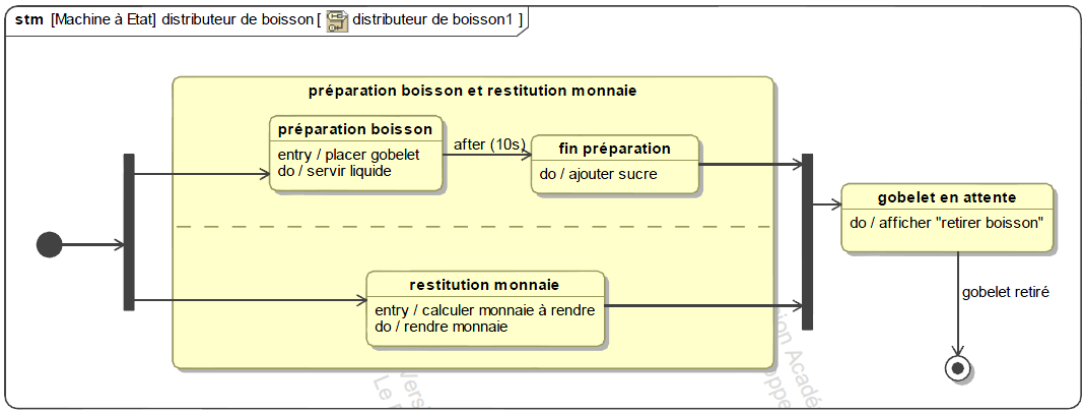
Il est possible d'utiliser une condition particulière avec le mot clé « else » (voir pseudo état choice - VI).

2. Activation de plusieurs états cibles à partir d'un état source (divergence en ET ou parallèle)

C'est un **état composite** avec plus d'une **région** appelé **état orthogonal**. Il est constitué de sous-états liés par des transitions.
Lorsqu'un état orthogonal est actif, un sous-état direct de chaque région est simultanément actif.



Une autre représentation possible utilise des barres de synchronisation « fork » et « join » :

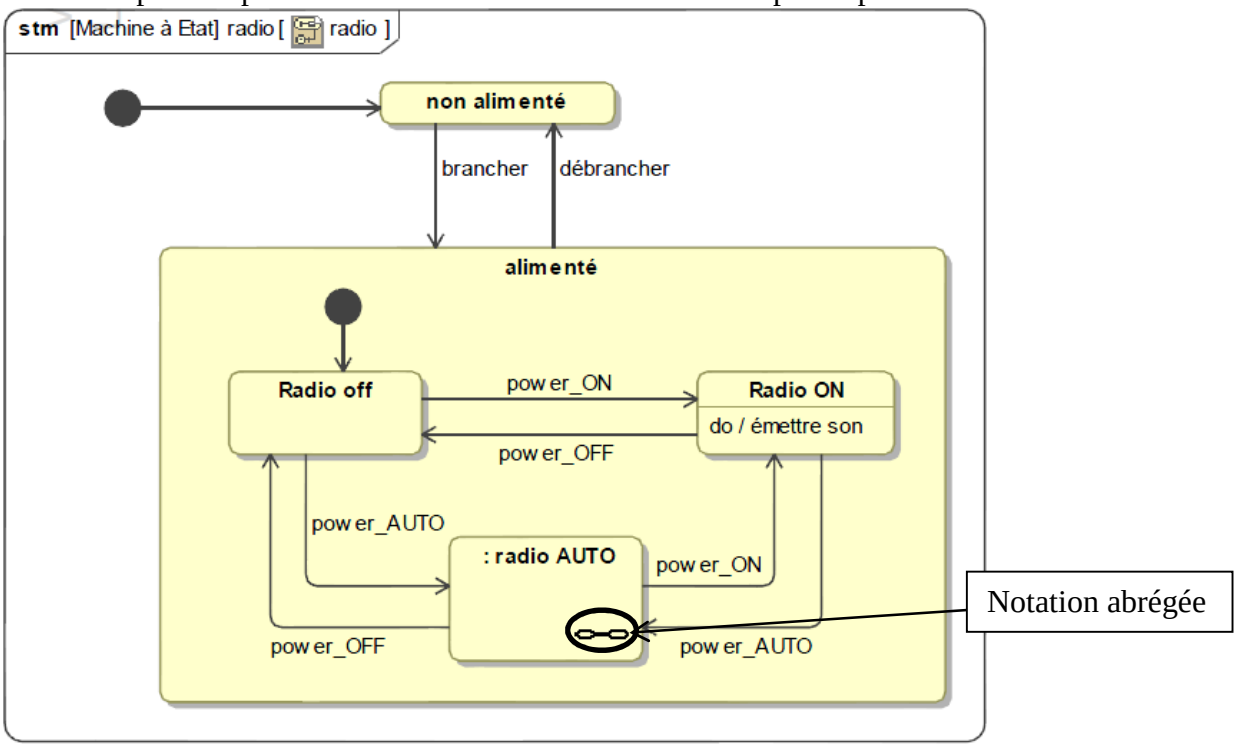


3. Hiérarchisation

On utilise, ici aussi, les **états composites** ou « **super états** » (submachine state).

Graphiquement les états composites peuvent être « ouverts » et le diagramme est lisible ou les états composites sont « fermés » (notation abrégée), le diagramme n'est plus lisible mais peut être développé dans un diagramme spécifique.

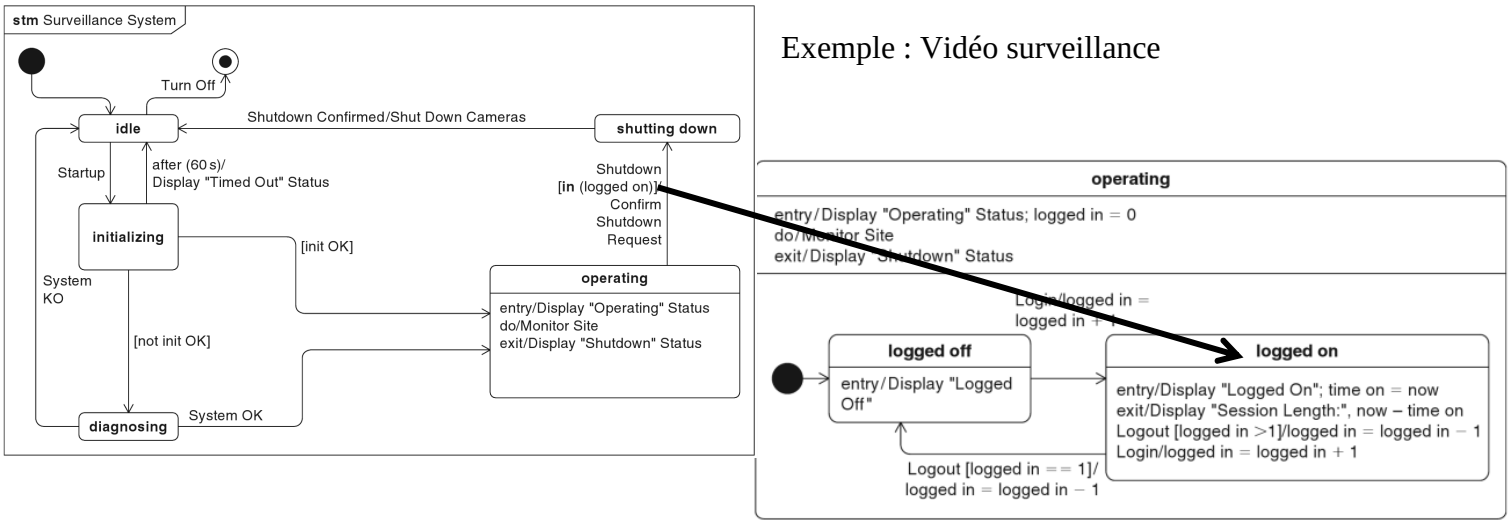
Un état composite est composé de plusieurs sous-états internes. Un état composite possède un état initial.



4. Concurrence et synchronisation

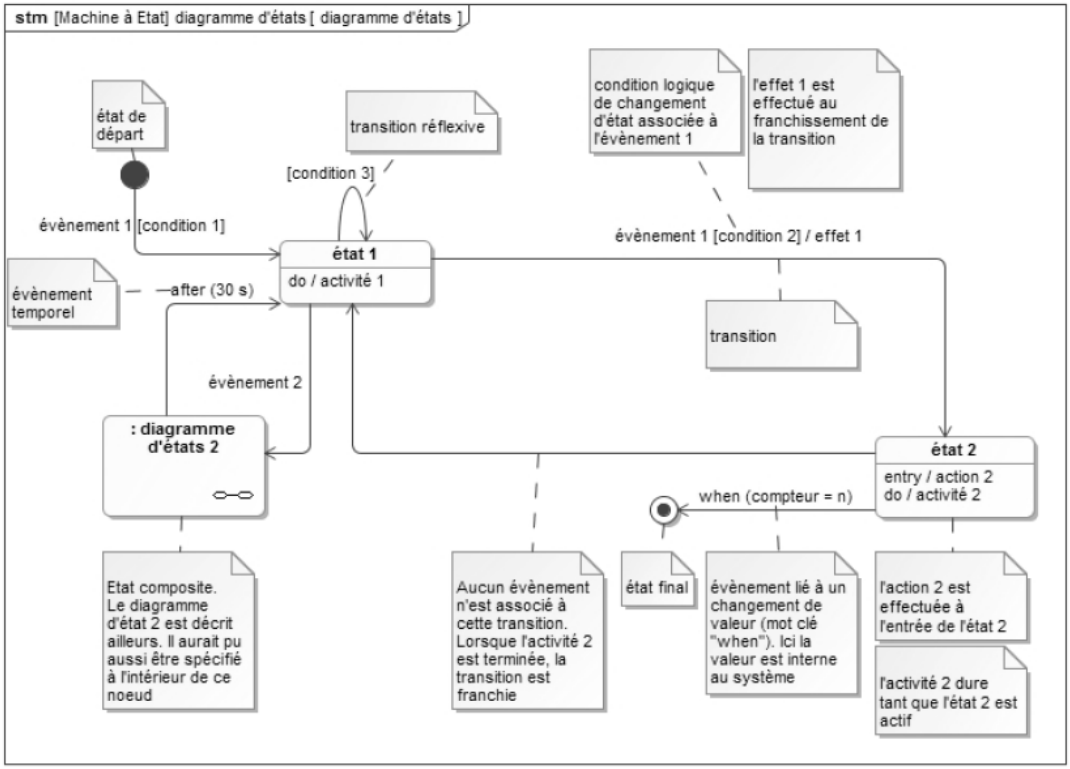
Dans un état composite, plusieurs diagrammes d'états peuvent évoluer simultanément (en parallèle). On dit qu'il y a concurrence de plusieurs états.

L’activation d’un état peut se produire si un état d’un autre diagramme est actif. Dans ce cas-là, la syntaxe de la condition de garde vérifiant l’activité (actif ou inactif) de ETAT est [in ETAT]



Exemple : Vidéo surveillance

Synthèse graphique des éléments d’un diagramme d’états :

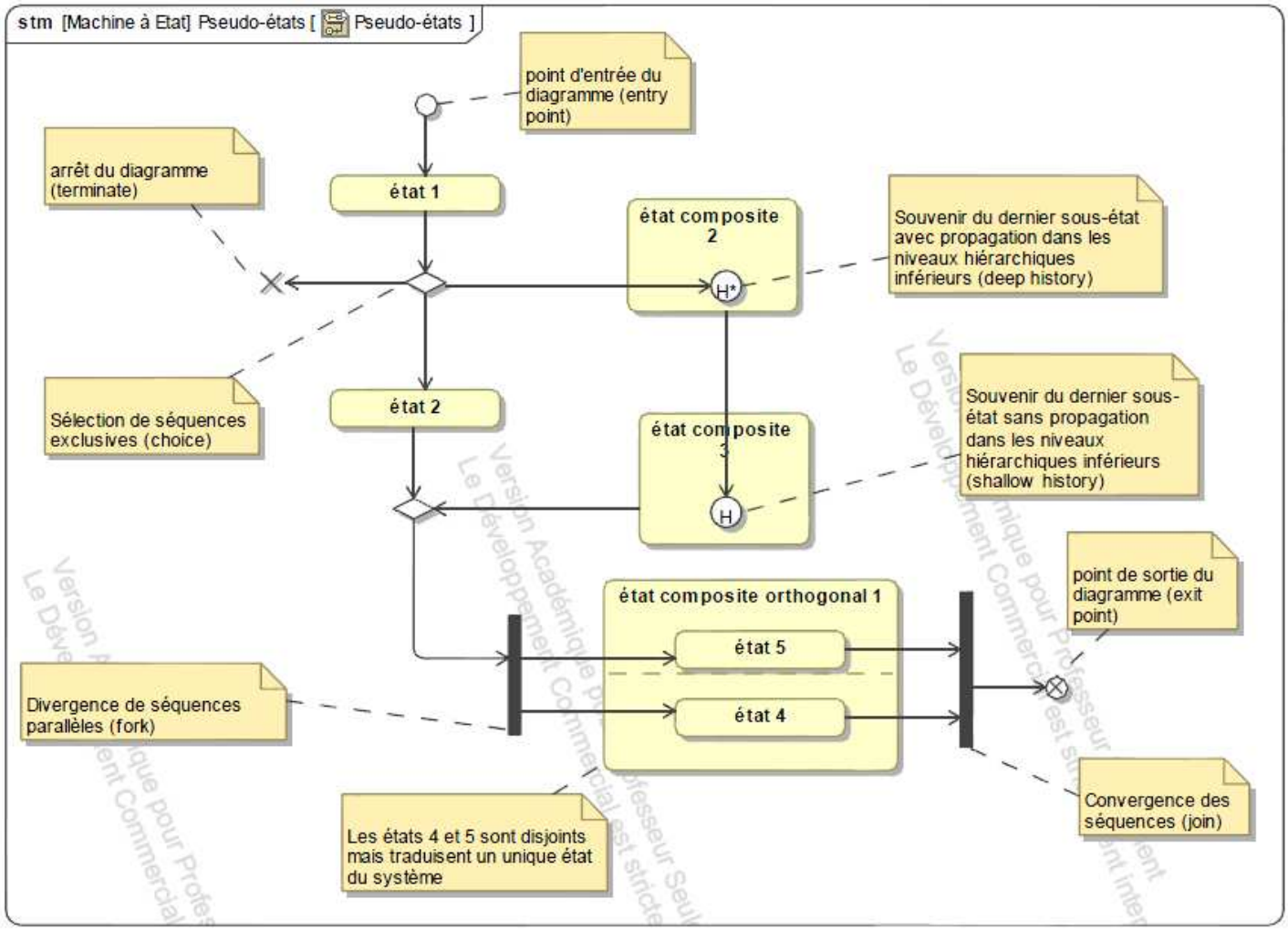


VI. POUR ALLER PLUS LOIN

Le formalisme SysML admet neuf pseudo-états :

- « shallow history » (H): permet à un état de niveau hiérarchique supérieur (état composite) de se souvenir du dernier sous-état, avant qu’il n’évolue vers un autre état,
- « deep history » (H*): idem que précédemment mais avec la propagation de l’historique à tous les sous-états composites de niveaux hiérarchiques inférieurs,
- « fork » et « join » : divergence et convergence de séquences parallèles,
- « choice » ◇: sélection et convergence de séquences exclusives. Il est nécessaire qu’une condition située en aval soit vraie pour que l’évolution du système se poursuive. Les conditions de gardes doivent être exclusives. Le mot clé « else » peut-être utilisé pour englober tout ce qui n’est pas décrit dans les autres expressions booléennes. Les conditions de garde situées en aval sont toutes évaluées une fois le pseudo-état atteint,
- « junction » •: idem au pseudo-état « choice », à la différence que pour qu’un chemin soit emprunté, toutes les conditions de garde situées en aval et en amont, doivent être vraies. L’évaluation des conditions avales est réalisée avant que le pseudo-état soit atteint,

- « entry point » ○ et « exit point » ⊗: permet de créer un point d'entrée du diagramme et un point de sortie vers un autre diagramme,
- « terminate » ⊗: permet de terminer une séquence sans destruction de l'instance de bloc.



Sources :
http://www.uml-sysml.org/
livre de Pascal Roques « **SysML par l'exemple** »
livre « **Sciences industrielles de l'ingénieur MPSI-PCSI-PTSI - Cours, synthèse & exercices corrigés** » dont les auteurs sont : Marc Derumaux, Patrick Kaszynski, Sébastien Roux, Christian Garreau, Vincent Crespel, Alain Caignot, Baudouin Martin
livre "**A Practical Guide to SysML : The Systems Modeling Language**" par Sanford Friedenthal, Alan Moore, Rick Steiner
http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_statediagram.html