Les diagrammes d'état - stm (State Machine Diagrams)

1- Diagramme d'état

1.1- Diagramme d'état

Le diagramme d'état (State Machine Diagram ou stm) est un diagramme normalisé SysML. Il décrit le comportement séquentiel d'un système ou d'une de ses parties. C'est un graphe, dont les nœuds (sommets) représentent les états de la machine, et les arcs orientés, les transitions, c'est un automate à nombre fini (≥1) d'états (ou machine à états finis). Il permet de modéliser le comportement dynamique d'un bloc (de la machine) selon des scénarios définis au préalable. Il contient :

- ☞ un état initial
- ☞ un ensemble d'états possibles du bloc
- ☞ un ensemble d'activités et/ou d'actions
- ☞ un ensemble de déclencheurs (des évènements) permettant l'évolution d'un état à l'autre.

Des règles de comportement permettent de modéliser l'évolution d'état en état à partir d'apparition d'évènements ainsi que de décrire les actions et activités réalisées par le système.

1.2- Evènement

Phénomène considéré comme localisé et instantané (discret), c'est-à-dire survenant en un point de l'espace et à un instant bien déterminé (ex : appui sur un bouton). lorsque cet évènement est décrit par une variable "a", l'évènement correspond à un front montant de "a" (passage de 0 à 1)

1.3- Action

Représente un comportement élémentaire du système. Elle ne peut pas être interrompue. (ex : Affectation de variable, Incrémentation d'un compteur, Envoi d'un signal…).

1.4- Activité

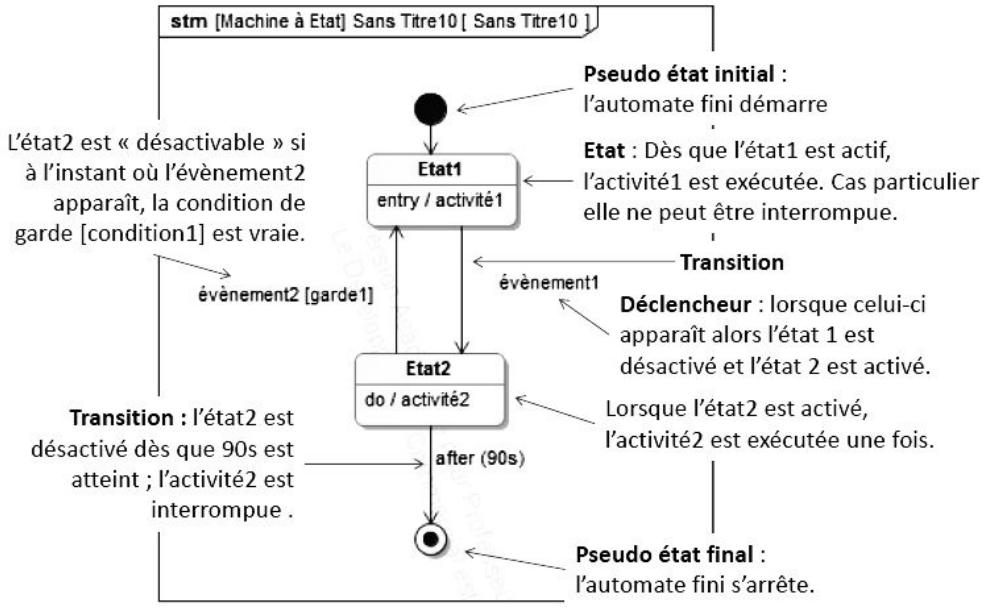
Séquence d'actions ou action maintenue dans le temps. Elle doit être interrompue si c'est une action maintenue. Exemple : Allumer un voyant, Mettre en rotation un moteur, Alimenter un vérin….

1.5- Etat

Situation durant la vie d'un bloc pendant laquelle :

- ☞ Il satisfait une certaine condition
- ☞ Il exécute une certaine activité
- ☞ Il attend un certain évènement

1.6- Exemple

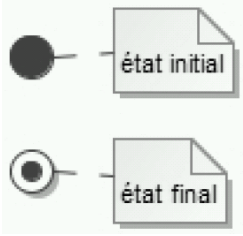


2- Les différents états

2.1- Etats initial et final

Ces états sont en fait des pseudo-états anonymes (sans nom).

Ils ne contiennent aucune action ou activité associée

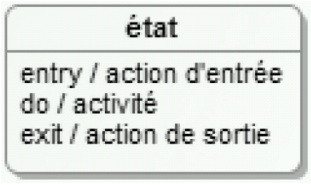


L'état initial correspond à l'état dans lequel on entre lors de la création de l'instance de bloc. Il n'a donc aucune transition entrante. On a un unique état initial par diagramme.

L'état final correspond à la destruction de cette même instance de bloc. Il n'a donc aucune transition sortante. Contrairement à l'état initial il peut y en avoir plusieurs, correspondant au déroulement de plusieurs scénarios possibles.

2.2- Etat : cas général

Un état modélise un comportement particulier du système. En général on lui associe une ou plusieurs activités ou actions. Il est représenté par un rectangle aux coins arrondis.



Un état possède un titre. Partie supérieure du rectangle. Ce titre est unique dans le diagramme.

Les activités et actions de cet état sont précisées dans la partie inférieure du rectangle.

Les actions ont une durée courte et ne sont donc pas interrompues. Elles sont effectuées à l'entrée et/ou à la sortie de l'état avec les mots clef suivant :

☞

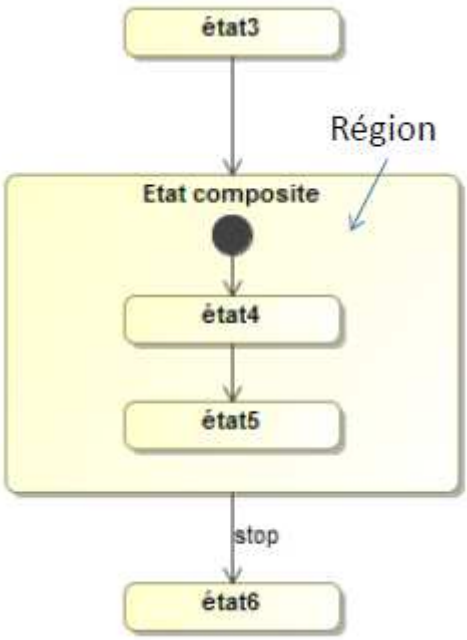
☞

Les activités durent tant que l'état est actif.

☞

2.3- Etat composite

C'est un état contenant un automate fini appelé région détaillant son fonctionnement séquentiel.



☞

☞

Un état composite peut contenir ou non des activités et/ou actions suivant les mots clef "entry" "do" et "exit".

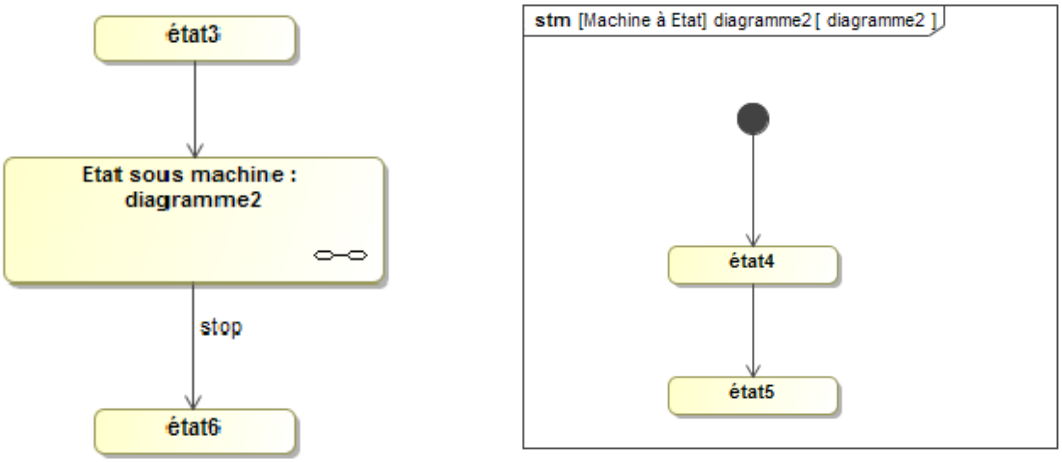
Chaque sous état peut également être un état composite.

A l'intérieur de cet état composite il peut y avoir ou non un état final.

2.4- Sous machine à état

C'est également un état contenant un automate fini détaillant son fonctionnement séquentiel. La différence avec l'état composite étant que sa description n'est pas faite à l'intérieur du diagramme mais fait l'objet d'un autre diagramme d'état.

Pour le signaler on utilise un pictogramme représentant 2 états et une transition.

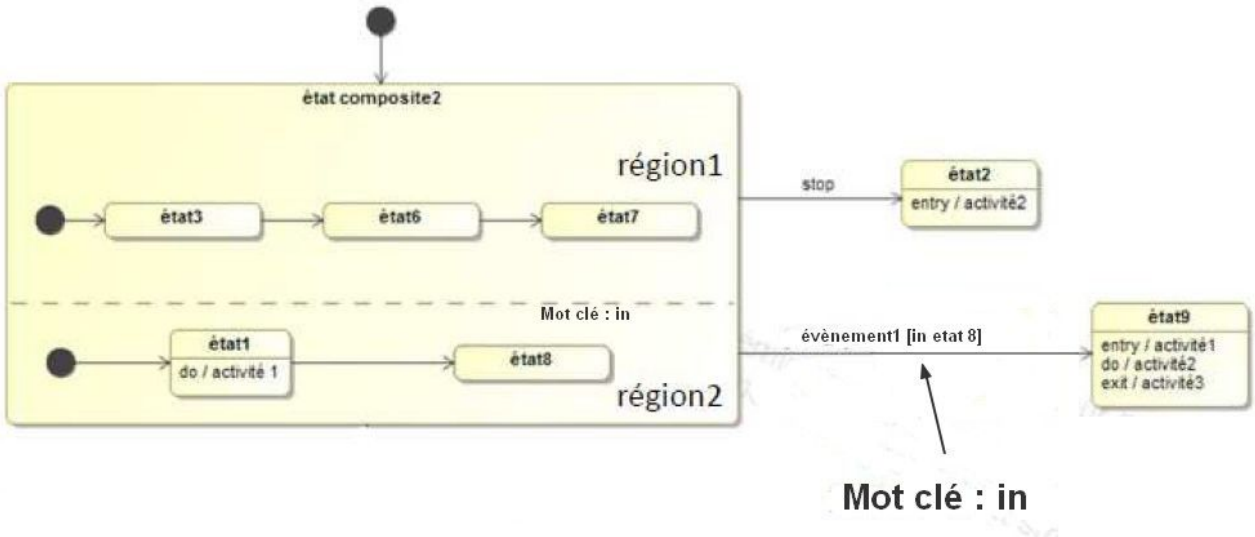


Le nom de la sous machine a état est suivit du nom du diagramme qui lui est associé.

2.5- Etats orthogonaux

☞

☞



☞ Si l'évènement « stop » survient, l'état composite2 se désactive ; tous les états (dans les deux régions) le composant se désactivent et l'état2 s'active.
Si des activités sont en exit, elles sont exécutées dans l'ordre inverse hiérarchique c'est-à-dire en premier celles des états des régions puis celle de l'état composite, l'état2 est activé.

☞ Si l'évènement « évèvement1 » survient et que l'état8 est actif, [in état8] l'état composite2 se désactive (et les activités exit des état8 et de l'état courant de la région 1 s'exécutent) puis l'état9 s'active.

2.6- Pseudo état de jonction

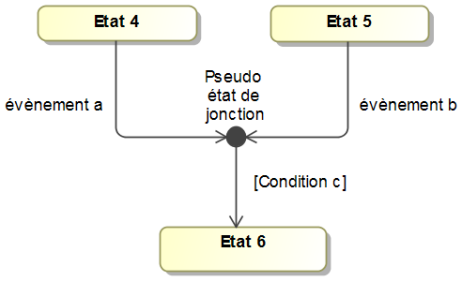
Il s'agit d'un pseudo état donc il ne lui est associé aucune activité ou action.

Le pseudo état de jonction permet de regrouper des conditions de garde.

Exemple :

A la survenue de l' "évènement a" et lorsque l'état 4 est actif (ou de l' "évènement b" et lorsque l'état 5 est actif), le pseudo état de jonction est activé.

Ensuite, si on a la garde "Condition c" l'état 6 est activé et le pseudo état de jonction ainsi que l'état 4 (ou l'état 5) sont désactivé.



2.7- Pseudo état de choix

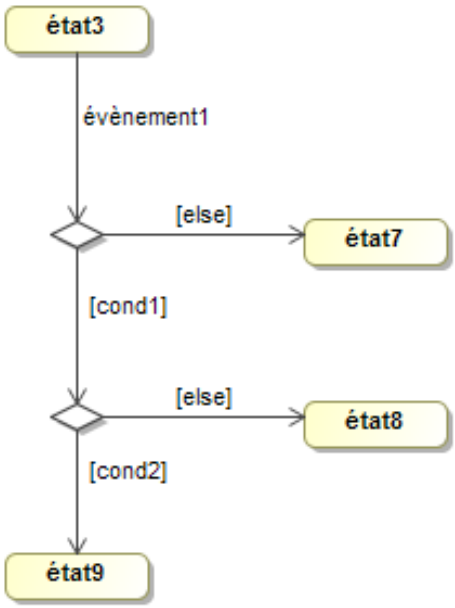
Il s'agit d'un pseudo état donc il ne lui est associé aucune activité ou action.

Le pseudo état de choix permet d'activer des états en fonction des conditions de garde.

Exemple :

A la survenue de l' "évènement 1" le premier pseudo état est atteint. Puis les gardes sont évaluées pour atteindre les états qui lui sont liés (état 7 ou deuxième état de choix). Et ainsi de suite.

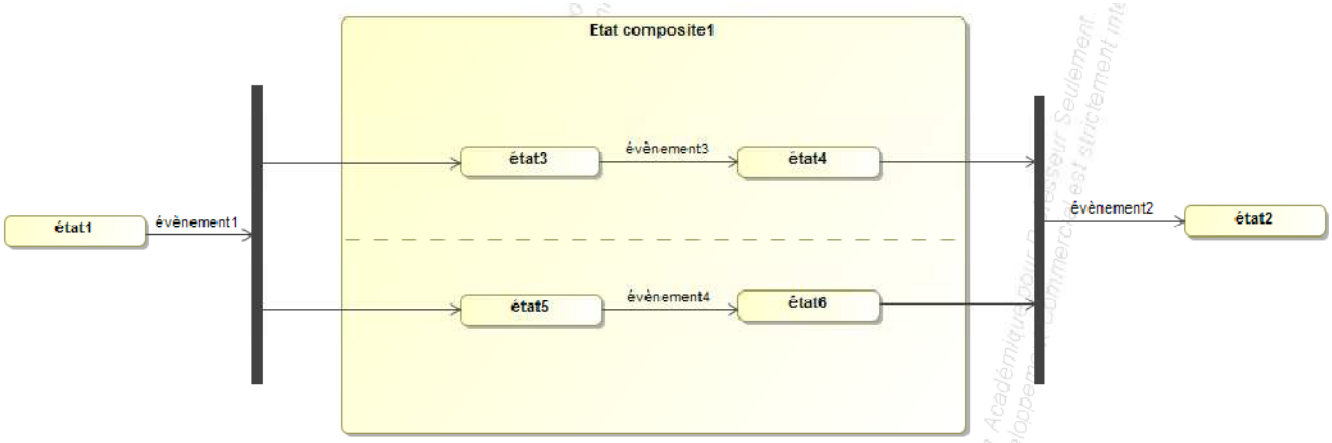
SysML/UML impose un choix exclusif pour obtenir un unique état actif et ne pas rester bloqué sur ce pseudo état.



2.8- Pseudo état de divergence (fork) et convergence (join)

Il s'agit de pseudo états donc il ne leur est associé aucune activité ou action.

Les pseudo états "fork" et "join" permettent de représenter graphiquement le parallélisme entre des régions d'un même état composite.



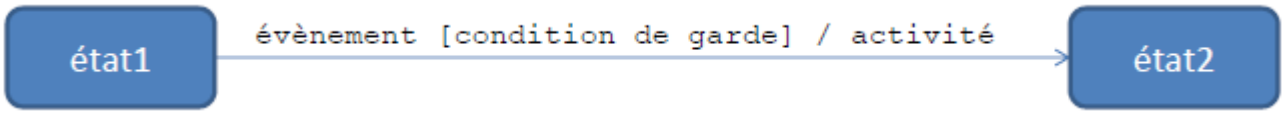
☞ A la survenu de l' "évènement 1" (et état 1 actif) les états "état3" et "état5" sont atteint simultanément.

☞ L'état 2 sera atteint uniquement avec l'évènement 2 et les deux états "état4" et "état6" actifs .

3- Les transitions

3.1- Principe

Transition : arc orienté qui relie deux états (source vers cible). Elle permet de passer d'un état (source) à l'autre (cible). L'état source est alors désactivé et l'état cible activé.



☞

☞

☞

☞

☞

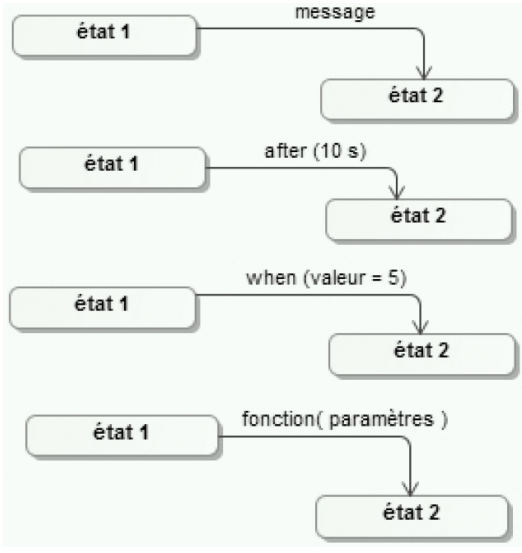
Une transition qui ne contient aucune condition de franchissement est dite automatique (utilisée de préférence uniquement après un pseudo état initial). L'évènement correspond alors à la fin des activités d'entrée de l'état ou de l'activation de l'état 1 si celui-ci n'a pas d'activité.

3.2- Transition réflexive

Une transition peut être réflexive (d'un état vers lui-même) cela entraine une interruption puis une reprise de l'activité liée ainsi qu'une nouvelle exécution des actions d'entrée et de sortie.

A chaque transition peut être associé un évènement qui déclenche cette transition et/ou un effet qui est réalisé lors du franchissement de la transition

3.3- Les évènements



Les évènements peuvent être de quatre types :

Un message asynchrone qui lorsqu'il est reçu déclenche la transition. Ce message peut être une expression booléenne d'entrée et/ou de variables internes. L'évènement est alors le front montant de ce booléen.

Un évènement temporel défini un intervalle de temps écoulé depuis l'entrée dans l'état (mot clef « after (10 s) ») ou une date absolue atteinte (mot clef « at »)

Un changement : Une valeur a changé entrainant le franchissement de la transition (mot clef « when »)

Une requête d'appel : Une requête de fonction du bloc à été émise avec si nécessaire des paramètres. Un retour sous forme de booléen est alors attendu.