Diagrammes SysML

Table des matières

I Diagramme de contexte

Diagramme de contexte SysML

Fondamental

(non normalisé)

Le **diagramme de contexte** SysML permet de définir les frontières de l'étude, et en particulier de préciser la phase du cycle de vie dans laquelle on situe l'étude (généralement la phase d'utilisation). Il répond à la question "**quels sont les acteurs et éléments environnants au système ?**".

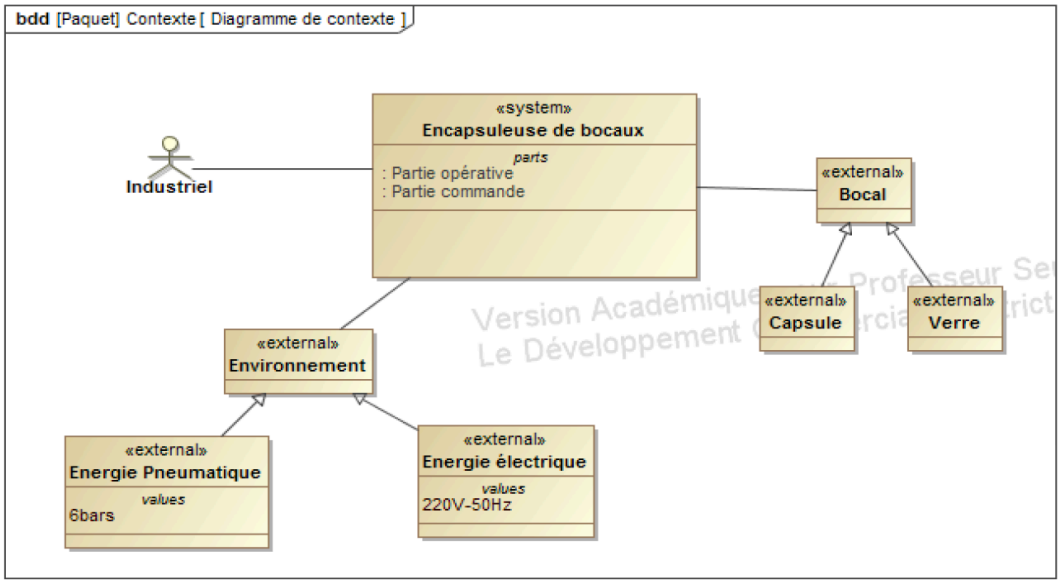


Diagramme de contexte de la capsuleuse de bocaux

II Point de vue fonctionnel

1. Cas d'utilisation

Afin de ne pas perdre de vue les **fonctionnalités offertes par le système complet**, le langage SysML décrit ce qui est appelé les "**cas d'utilisation**".

C'est la réponse à la question : "**quels services rend le système ?**"

Fonctionnalité

Az Définition

C'est un service rendu en **autonomie**, d'un bout à l'autre, par le système. La fonctionnalité est visible par l'entité extérieure en interaction avec le système.

Elle possède un point de départ, une succession d'étapes (cf. *diagramme de séquence* (cf. p.7)) , et se termine.

Remarque

Le recyclage ou le nettoyage, par exemple, ne sont pas des cas d'utilisation puisque ce ne sont pas des services rendus en **autonomie** par le système.

Diagramme des cas d'utilisation en SysML

Fondamental

(*Use Case Diagram*)

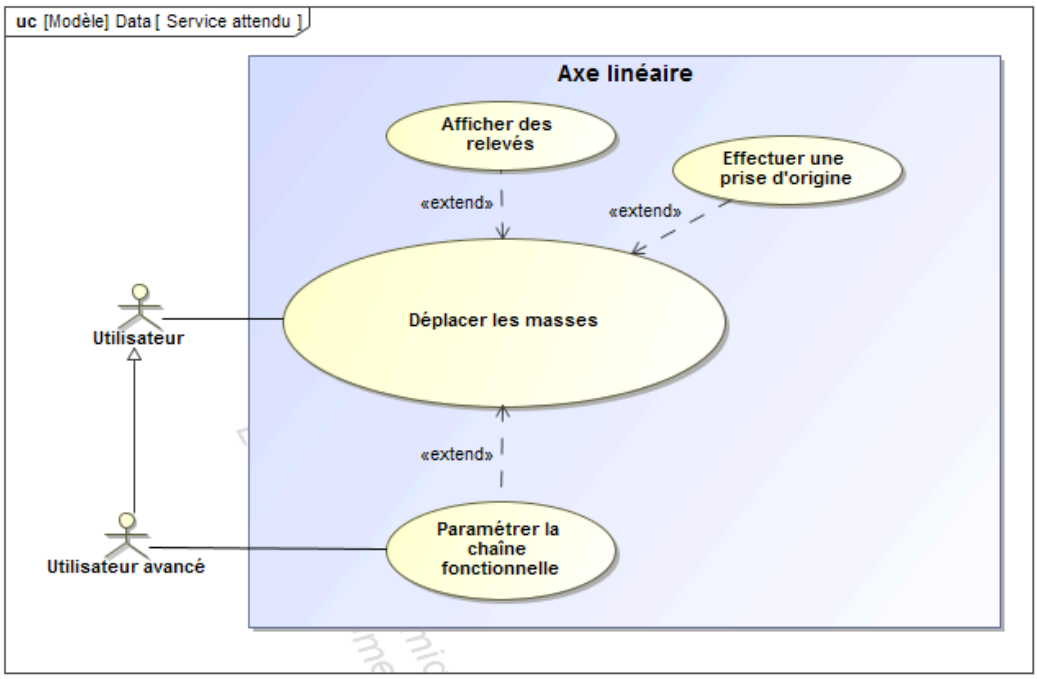


Diagramme des cas d'utilisation de l'axe linéaire Control'X

On retrouve sur ce diagramme les éléments suivants :

- acteur(s)
- relation(s) entre acteur(s) et le système
- frontière du système
- fonction globale du système

Remarque

Il doit y avoir autant de diagrammes de cas d'utilisation qu'il y a de phases de vie du système identifiées lors de l'analyse du besoin.

2. Exigences d'un système

La référence, tout au long de la conception d'un système, est constituée par l'ensemble de ce qui est exigé **par** et **pour** le système. Cela correspond donc aussi bien à son **but général**, mais également à tout **ce qui doit être respecté** vis-à-vis de l'environnement, de l'énergie disponible, des techniques utilisées, etc.

Il y a donc différents "niveaux de détail" des exigences, et en faire une liste exhaustive pour un système est très difficile.

Tondeuse à gazon Exemple

Le but général, donc une exigence pour le système, serait par exemple *« tondre le gazon »*. A un niveau beaucoup plus fin, il pourrait exister une exigence comme *« la couleur des poignées doit être jaune »*.

Diagramme des exigences en SysML Fondamental

(*Requirement Diagram*).

Objectifs principaux vis-à-vis des exigences :

- lister et spécifier
- hiérarchiser
- documenter (ajout d'éléments d'autres diagrammes, par ex. un élément architectural)

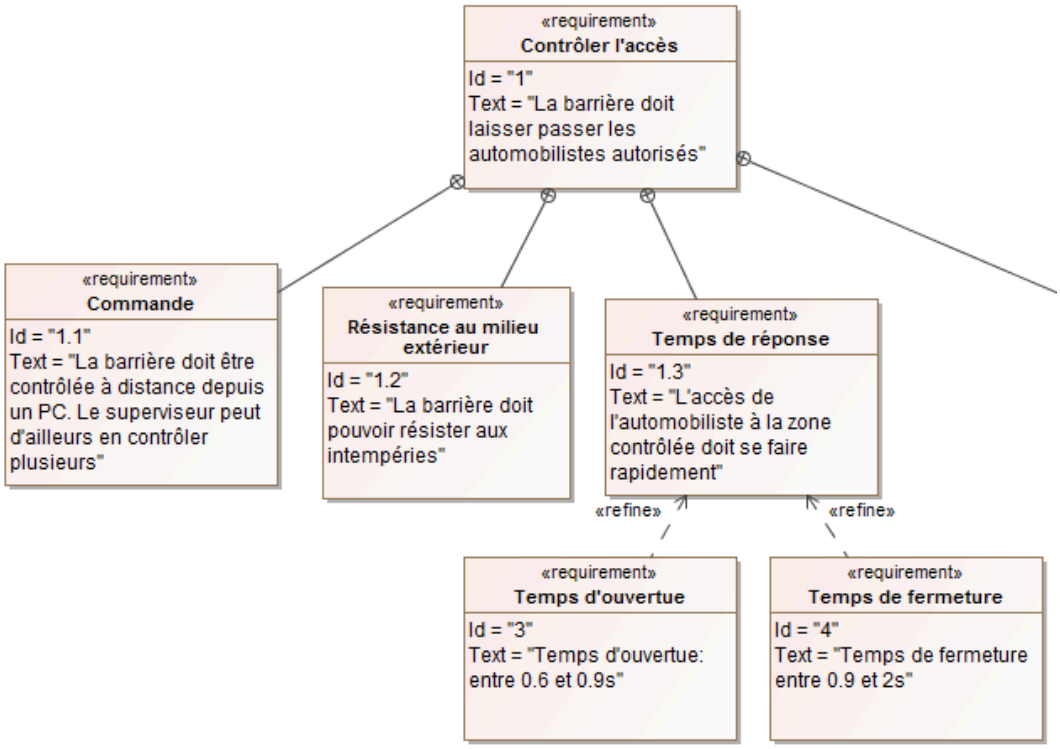


Image 1 Diagramme des exigences (partiel) de la barrière Sympact

Rectangles

- **exigence** (verbe à l'infinitif)
- **sous-système** : satisfaisant une exigence
- **commentaire**

Liens

- **contenance** ⊕——: une exigence est déclinée en d'autres "sous-exigences"
- **précision** <<refine>>: des précisions sont données à propos d'une exigence
- **dérivation** <<derive>> : une exigence, non imaginée au départ, est créée à partir d'une autre
- **satisfaction** <<satisfy>> : un bloc (cf. *diagramme de définition de blocs* (cf. p.17)) satisfait une exigence
- **vérification** <<verify>> : un scénario de test vérifie une exigence

> 💬 Remarque
>
> Ce diagramme devient vite illisible si l'on veut lister toutes les exigences. On peut éventuellement réaliser plusieurs diagrammes, chacun correspondant à un type d'exigences, par exemple.

III Point de vue comportemental

1. Séquences d'un système

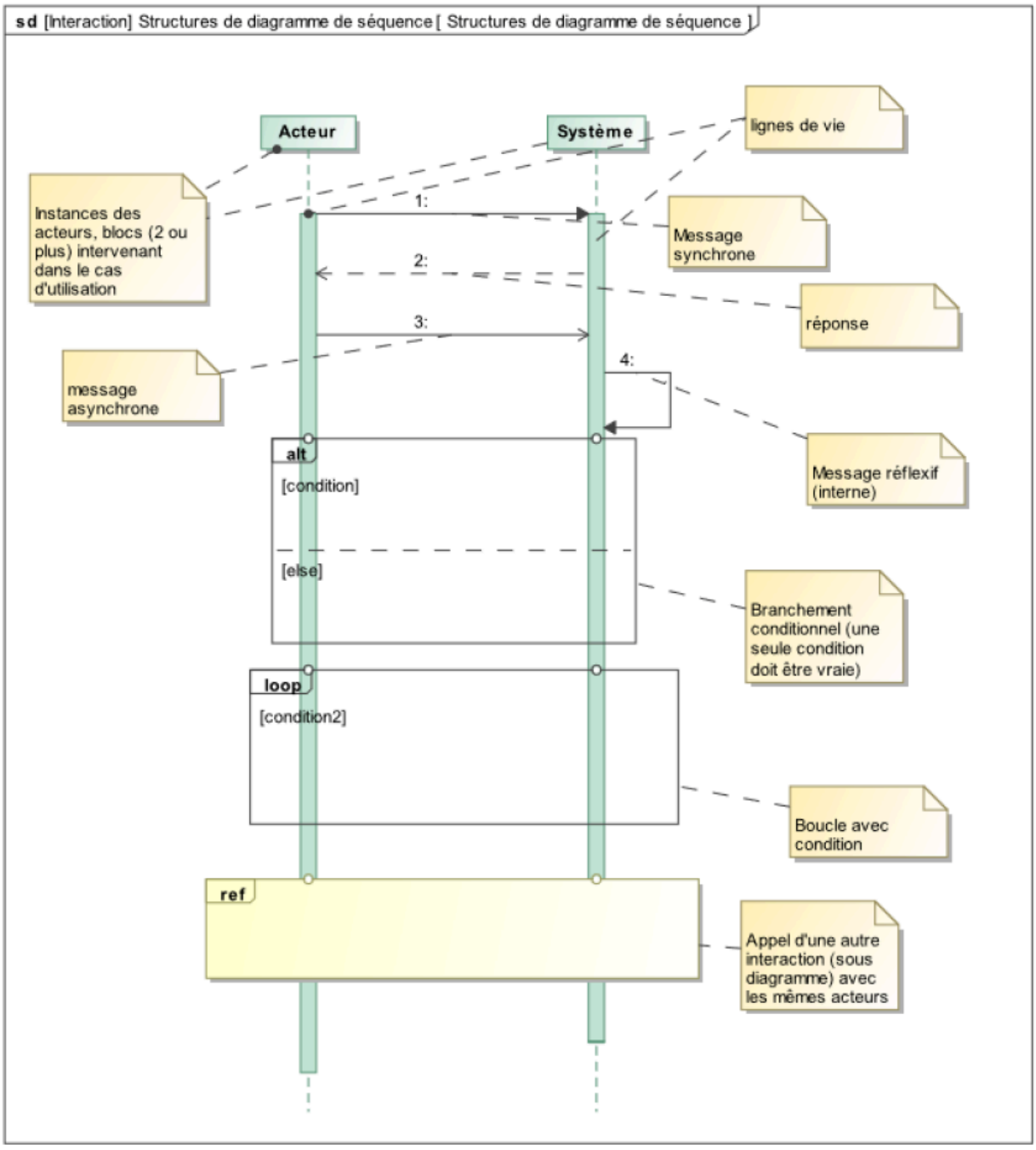
Pour chaque cas d'utilisation d'un système, on peut décrire plus précisément quelles sont les **étapes successives** (la "séquence") qui permettent de rendre le service prévu.

C'est donc un enchaînement d'échanges, des **interactions** dont la nature est précisée, entre les acteurs (utilisateurs) et le système.

Diagramme de séquence en SysML

Fondamental

(*Sequence Diagram*).



Syntaxe

Les "lignes de vie" représentent un élément du diagramme de cas d'utilisation.

Plusieurs types de messages peuvent être échangés :

- **synchrone** : l'émetteur attend une réponse (flèche retour en pointillé)
- **asynchrone** : pas de réponse est attendue
- **réflexif** : représente une activité au sein de l'émetteur

Exemple

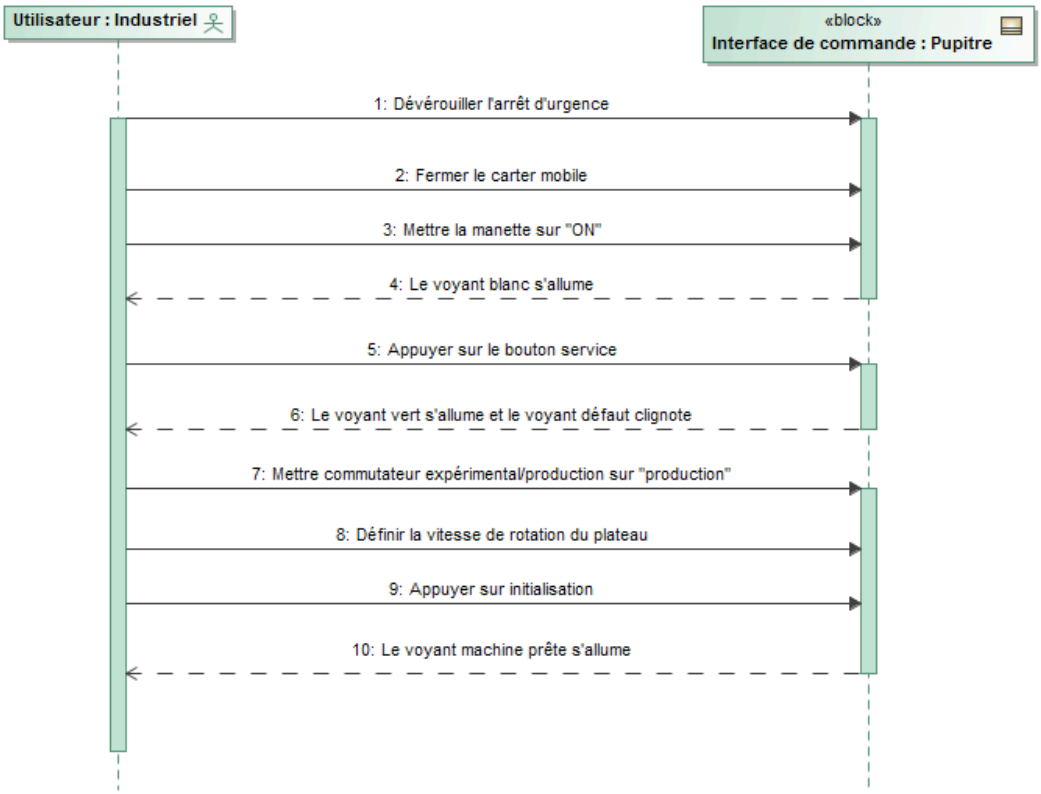


Diagramme de séquence de l'initialisation de la capsuleuse

2. États d'un système

2.1. Généralités

Les différents états possibles d'un système - dans son ensemble - peuvent être représentés sous forme d' "automate fini" (machine à états).

- Un état représente une **activité**, ou une **attente** d'un événement particulier.
- Des **transitions** permettent de passer d'un état à un autre.

Diagramme d'états en SysML

Fondamental

(*State Machine Diagram*)

Ce diagramme est rattaché à un bloc des diagrammes de blocs.

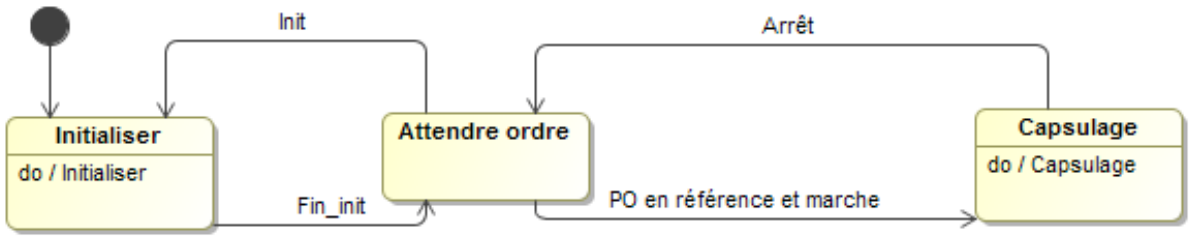


Diagramme d'états de la capsuleuse de bocaux

2.2. Détail

a) État

Les éléments graphiques utilisés dans ce diagramme sont principalement des rectangles aux coins arrondis pour représenter les états.

L'état initial est représenté par un rond plein, et les états finaux par des ronds "creux".

> 💬 Remarque
>
> Il peut y avoir plusieurs états finaux car plusieurs scénarios peuvent être possibles pour mettre fin à un comportement.

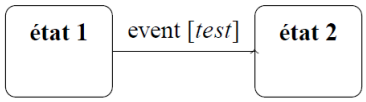
b) Transition

Une **transition** peut être associée à un **événement**, une **condition de garde** et / ou à un **effet** (action).

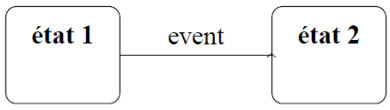
Elle s'écrit : "événement [condition de garde] / effet".

Quelques exemples :

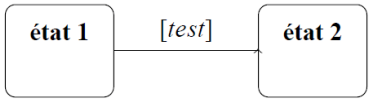
- A l'occurrence de **event**, **test** est évalué et la transition est franchie uniquement si **test** est vrai. L'éventuelle activité est interrompue. Si **test** n'est pas vrai, **event** est perdu et il faut attendre une seconde occurrence de **event** pour éventuellement franchir la transition si cette fois **test** est vrai.



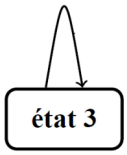
- A l'occurrence de **event**, la transition est franchie sans condition. L'éventuelle activité est interrompue.



- Si **test** est vrai, la transition est franchie uniquement dès la fin de l'éventuelle activité (qui doit donc être une activité finie). S'il n'y a pas d'activité associée à l'état 1, la transition est franchie immédiatement si **test** est vrai.



- Transition de complétion : est immédiatement franchie dès la fin de l'éventuelle activité. Équivaut à [1].



Une transition réflexive entraîne une sortie d'état puis un retour dans ce même état.

Cela n'est donc pas sans conséquences selon les cas.

Événement

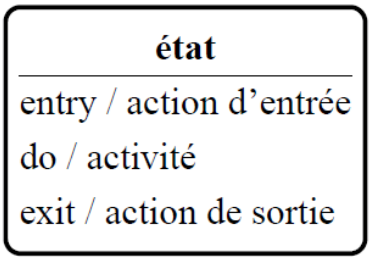
Il existe **quatre** types d'événements associés à une transition :

1. le **message** (*signal event*) : un message asynchrone est arrivé
2. l'**événement temporel** (*time event*) : un intervalle de temps s'est écoulé depuis l'entrée dans un état (mot clé *after*) ou un temps absolu a été atteint (mot clé *at*)
3. l'**événement de changement** (*change event*) : une valeur a changé de telle sorte que la transition est franchie (mot clé *when*)
4. l'**événement d'appel** (*call event*) : une requête de fonction du bloc a été effectuée et un retour est attendu ; des arguments (paramètres) de fonction peuvent être nécessaires.

Condition de garde

La **condition de garde** est une expression booléenne faisant intervenir des entrées et/ou des variables internes. Elle autorise le passage d'un état à un autre.

c) Activité, action



A un état, on peut ainsi principalement rattacher une activité, une action d'entrée et une action de sortie.

Une **activité** peut être considérée comme une unité de comportement. Elle prend du temps et peut être interrompue. On la trouve à l'intérieur des nœuds du diagramme (mot clé *do*).

En revanche, une **action** ne prend pas de temps et ne peut pas être interrompue. Son exécution peut par exemple provoquer un changement d'état, l'émission d'un ordre pour un préactionneur ou un retour de valeur. On peut les trouver dans les transitions (**effet**) ou dans les états (mots clé *entry* ou *exit*). Les actions sont les éléments de base permettant de spécifier les activités dans le diagramme d'activités.

d) État composite (super-état)

Un **état composite** est constitué de sous-états liés par des transitions. Cela permet d'introduire la notion d'état de niveau hiérarchique inférieur et supérieur.

Exemple

Dans l'exemple qui suit, l'état "Radio Auto" est composite :

- il apparaît de façon monobloc dans le diagramme de gauche
- le diagramme d'états de droite détaille son contenu.

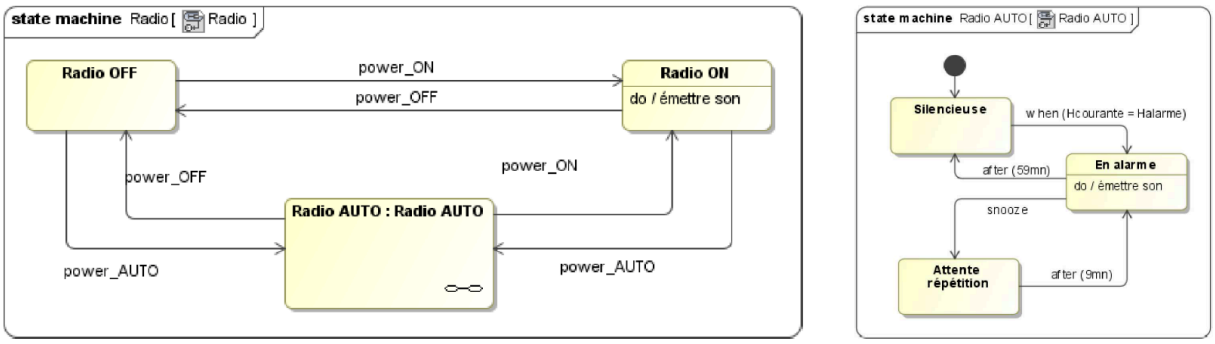


Image 2

3. Activités d'un système

3.1. Généralités

L'activité des **différents composants** d'un système peut être également être représentée sous forme d' "automate fini" (machine à états).

Des transitions permettent de passer d'un état à un autre.

Diagramme d'activités en SysML Fondamental

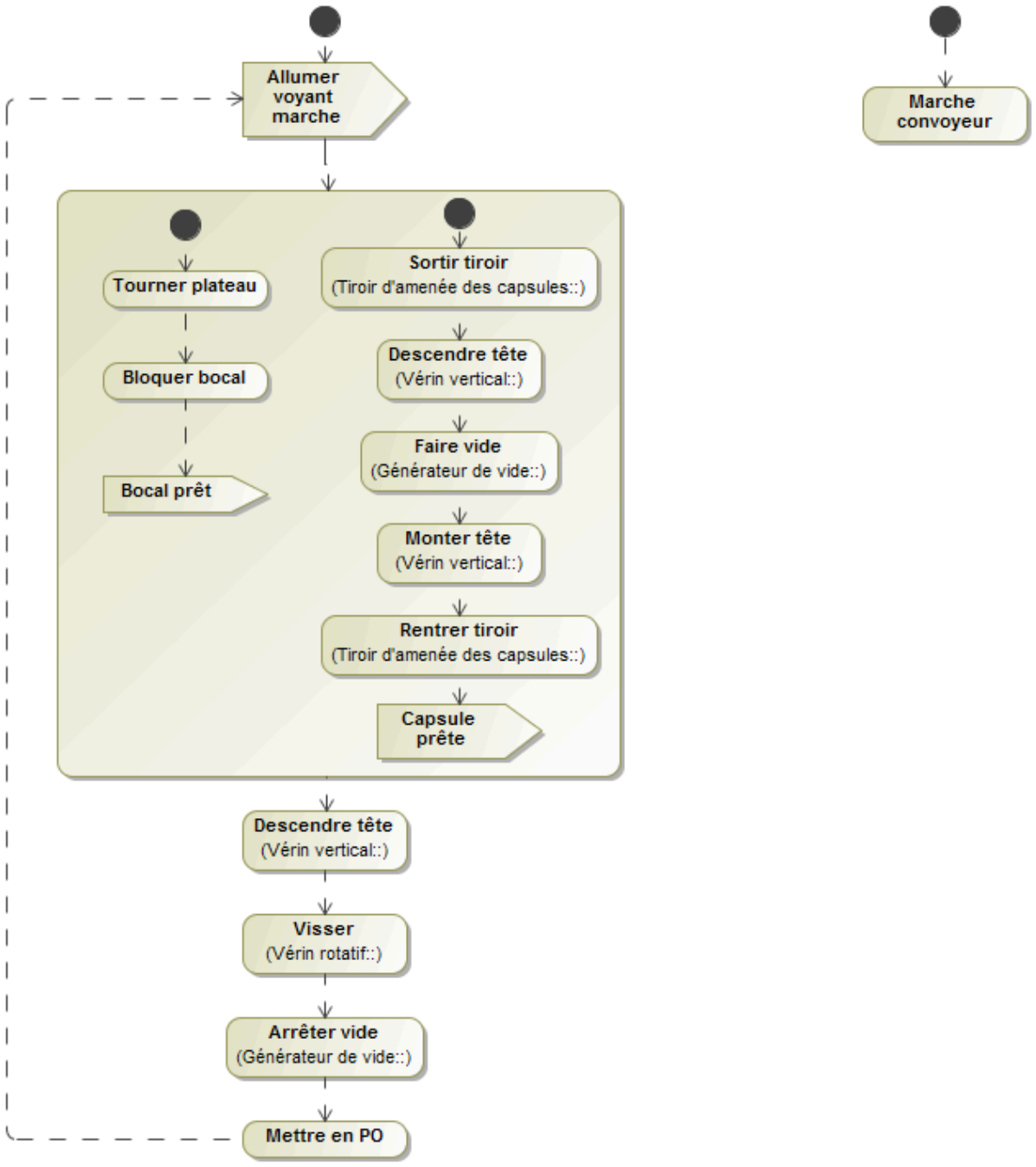


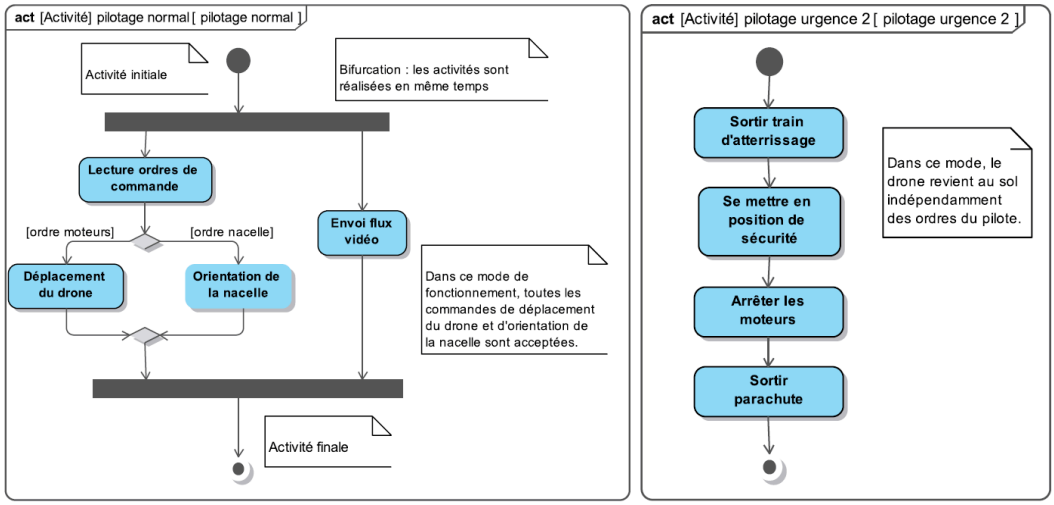
Diagramme d'activité du capsulage de la capsuleuse

3.2. Détail

Le diagramme d'activités permet de décrire la transformation des flux d'entrées en flux de sorties (matières, énergies, informations) par le biais de séquences d'actions ou d'activités déclenchées par des flux de contrôle.

Lorsqu'une tâche est terminée, la suivante commence : **il n'y a pas d'événement associé aux transitions** (au contraire du diagramme d'états).

Exemple



Signaux et événements

En plus de consommer et de produire des paramètres, une activité peut recevoir et émettre des signaux.

Les activités peuvent communiquer en incluant l'émission d'un signal et dans une autre la réception d'événements.

Il faut utiliser pour cela des types d'action particuliers, possédant chacun une représentation graphique spécifique :

- drapeau "entrant" : réception d'un événement
- drapeau "sortant" : envoi d'un signal
- "sablier" : événement temporel

4. Structures algorithmiques de base

4.1. Introduction

Suite à la description des diagrammes d'états et d'activités, on peut dégager quelques structures de base dans un algorithme, indépendamment de tout langage spécifique.

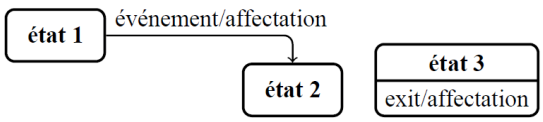
4.2. L'affectation

L'affectation consiste à donner une valeur particulière à une variable.

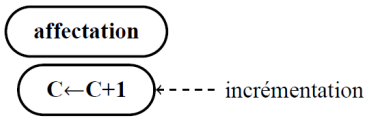
Cela peut se faire à l'aide d'une action et ne prend pas de temps significatif.

Avec le diagramme d'états

Exemple



Avec le diagramme d'activités Exemple



4.3. Suite d'instructions

Un groupe ou un bloc d'instructions peut être une séquence d'un diagramme d'activité. Cela correspond à une succession d'actions et / ou d'activités.

4.4. Fonctions et procédures

Scinder un algorithme en plusieurs fonctions et procédures permet :

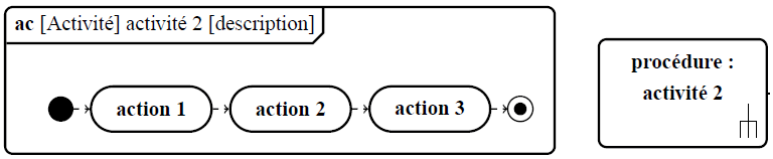
1. de décomposer un problème général en plusieurs problèmes élémentaires
2. de pouvoir réutiliser des programmes réalisant des tâches élémentaires.

Fonction ou procédure ? Attention

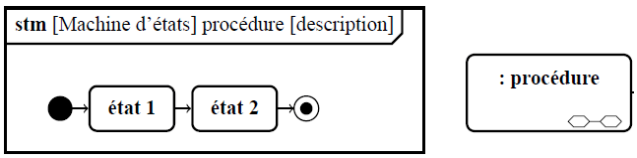
La **fonction** comporte une succession d'instructions et renvoie une valeur, une liste, un objet, etc...

La **procédure** comporte une succession d'instructions mais ne renvoie rien.

Avec le diagramme d'activités Exemple



Avec le diagramme d'états Exemple



4.5. Structures conditionnelles

Az Définition

Il y a une **alternative** :

"si ... alors faire ...., sinon faire ...."

Diagramme d'états Exemple

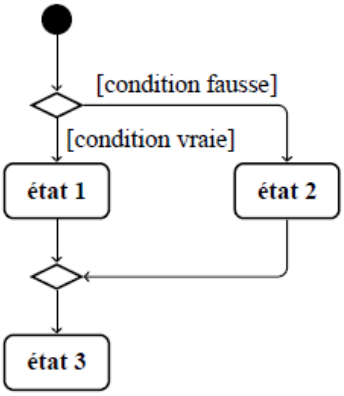
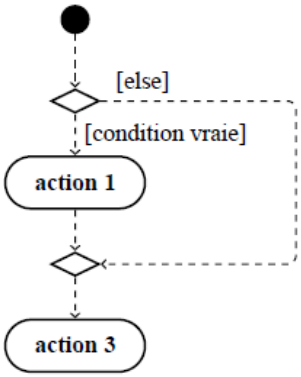


Diagramme d'activités Exemple



4.6. Structures répétitives

Le comportement est **itératif** :

- "tant que condition vraie, faire ..."
- "répéter ... jusqu'à condition vraie"
- "pour variable = valeur initiale, jusqu'à valeur maximale, faire..."

Diagramme d'états Exemple

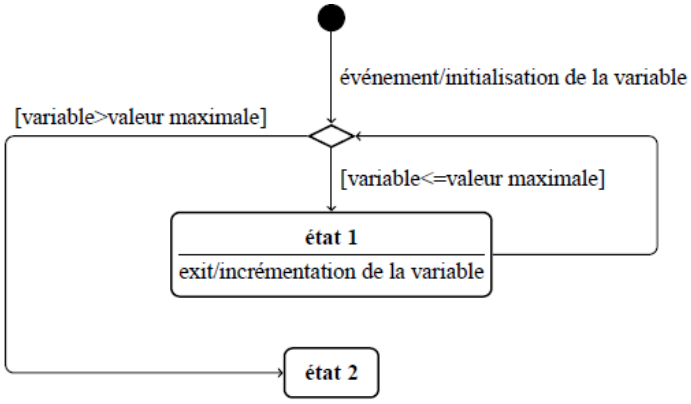
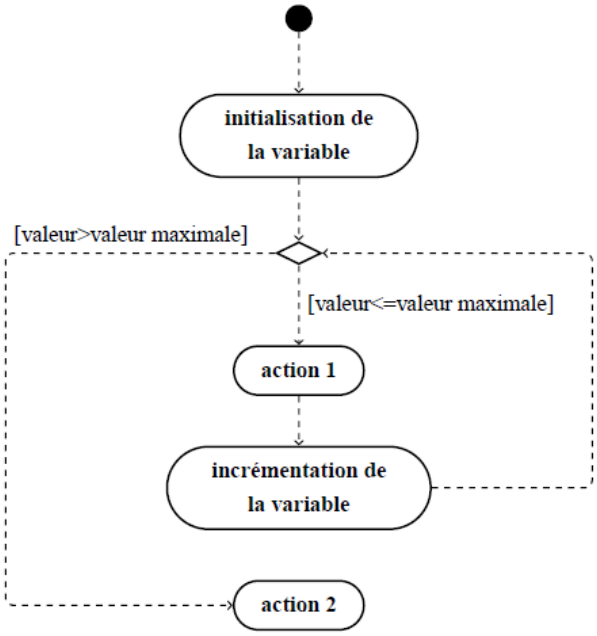


Diagramme d'activités

Exemple



IV Point de vue structurel

1. Diagramme de (définition de) blocs SysML

L'objectif est ici de savoir "qui contient quoi", à différents niveaux de précision.

Des "propriétés" et des "opérations" peuvent être ajoutées à chaque **bloc**, comme renseignements supplémentaires :

- grandeurs physiques caractérisant son comportement
- entrées et sorties de matière, d'énergie, d'information... sous forme de **ports**
- fichiers (plans, photos...) en pièce jointe

Les traits avec les extrémités en losange indiquent une appartenance d'un élément à un autre :

 : le losange est plein (composition), l'élément est obligatoire,

 : le losange est vide (agrégation), l'élément est facultatif.

Fondamental

(*Block Definition Diagram = BDD*)

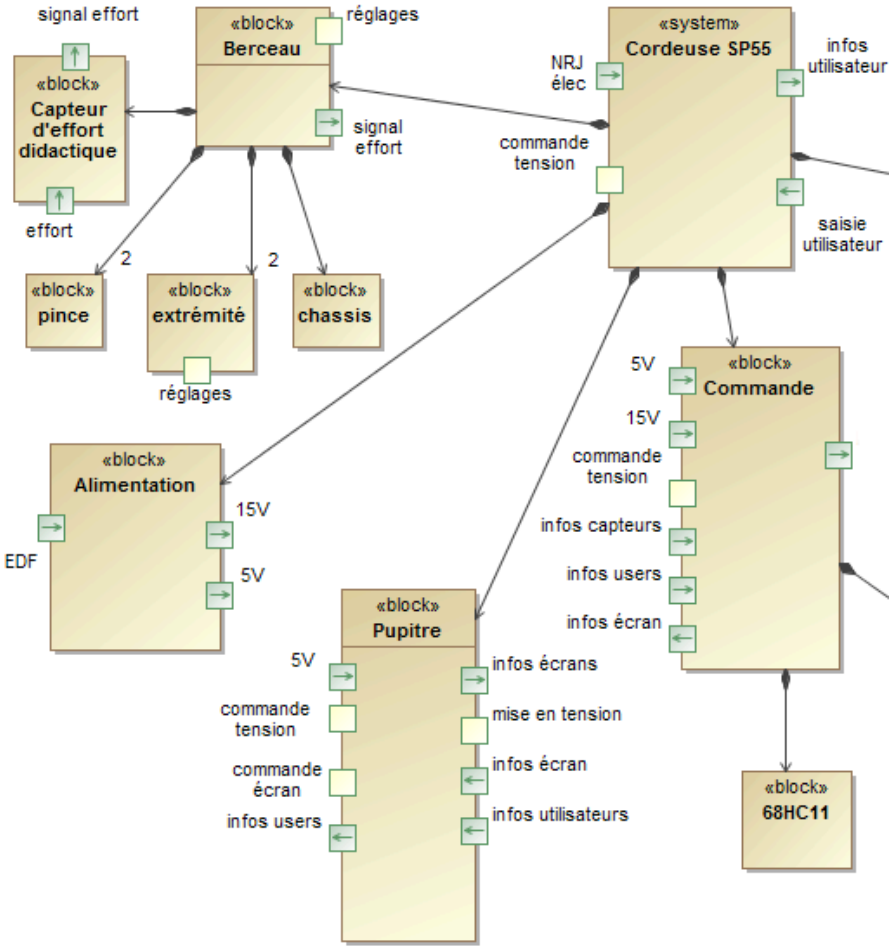


Diagramme de blocs (partiel) de la cordeuse de raquettes

Attention

On distingue l'**instance** du **bloc** : une instance est **une** des réalisations concrètes du bloc.

2. Diagramme de blocs internes SysML

Contrairement au diagramme précédent, celui-ci ne montre que des blocs **de même niveau**.

En revanche, il **détaille** les **échanges** qu'il peut y avoir entre les blocs. Ces échanges sont de type :

- **commande** : interface permettant d'invoquer un service ou une opération
- **flux** : canal (entrée ou de sortie) par lequel transite de la matière, de l'énergie ou de l'information.

La représentation graphique de ces échanges se fait au moyen de :

- **parties** (*parts*) : bloc muni ou non de ports
- **ports** : petits carrés à la frontière des parties, indiquant le sens des flux (les commandes n'étant pas orientées)
- **connecteurs** : véhiculant le contenu entre parties

Fondamental

(*Internal Block Diagram*)

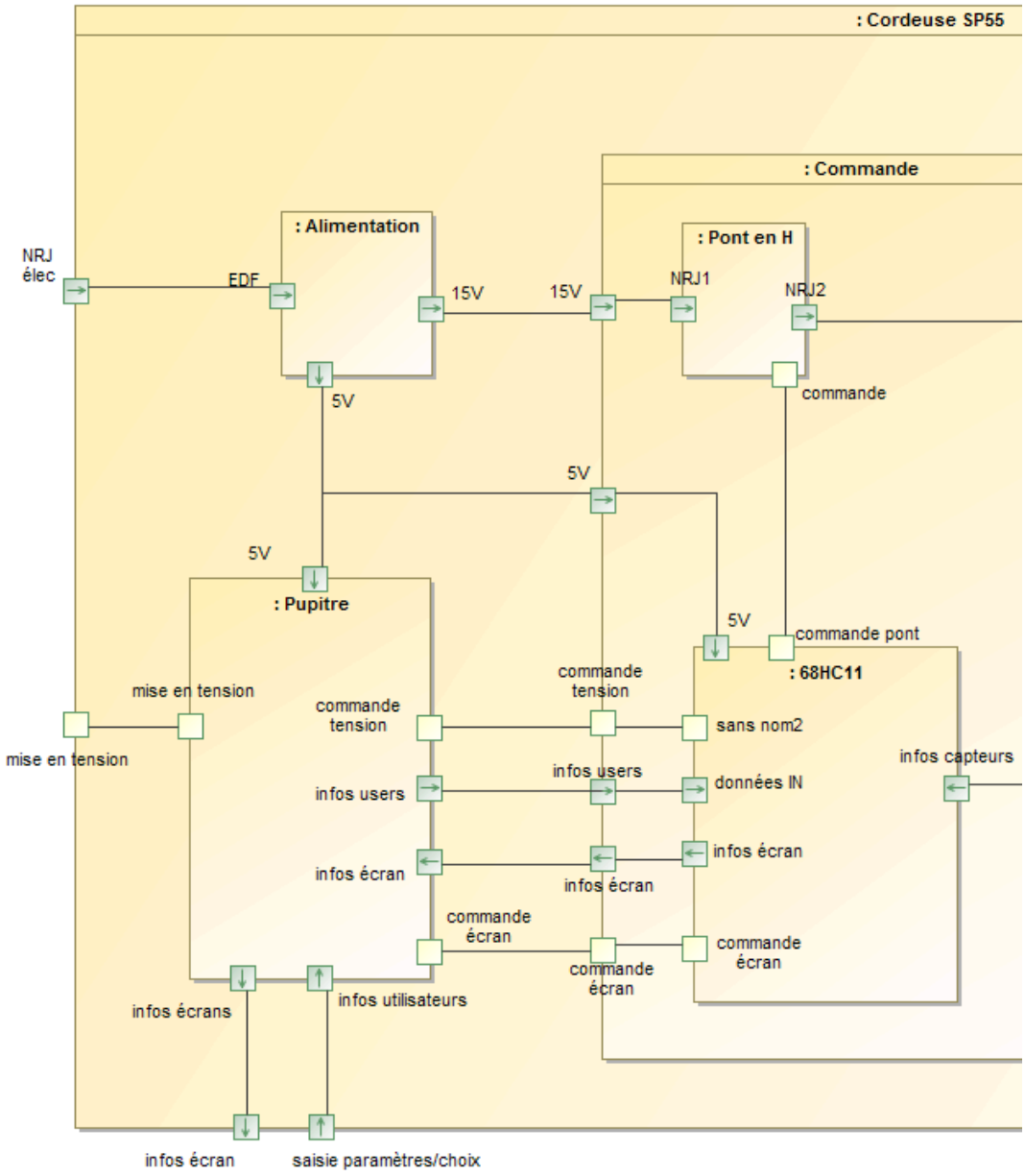


Diagramme de blocs internes (partiel) de la cordeuse de raquettes