Sciences industrielles

CPGE 1ere année

Systèmes à événements discrets

Sommaire

(1) Systèmes du premier ou du 2ème ordre

(2) Echelon, rampe ou sinusoïde.

On a étudié plus tôt dans l'année la façon dont se comportaient les systèmes continus usuels(1) lorsqu'ils étaient soumis à des entrées, elles aussi usuelles(2). Dans ce cours, on s'intéresse au comportement des systèmes à évènement discrets (SED).

Les systèmes à événements discrets (SED) se définissent par opposition aux systèmes continus dont l'évolution est continue dans le temps et peut être décrite par des équations différentielles.

Dans un SED, le passage d'un état à un autre est déclenché par des événements ponctuels.

Contrairement aux systèmes continus où les informations traitées sont de nature analogique, les systèmes à événements discrets manipulent des informations logiques ou numériques.

Les SED sont le plus souvent séquentiels, c'est-à-dire que la ou les sorties dépendent de la combinaison des entrées et de l'état précédent des sorties

Exemple : télécommande de téléviseur

Un appui sur le bouton  entraine soit l'allumage ou soit l'extinction du téléviseur.

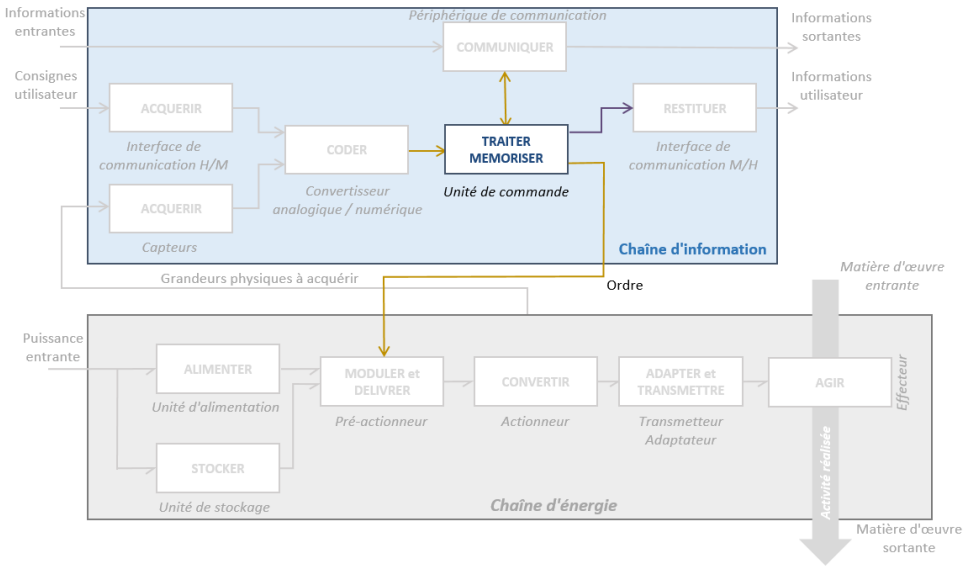
Ainsi, dans la cas d'un SED séquentiels :

- *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.*

Carte Arduino

L'unité de commande de ces systèmes à événements discrets est très souvent réalisée à partir de circuits logiques et de calculateurs.

Elle est programmée à partir de l'analyse d'un cahier des charges avec comme objectif, d'aboutir à un fonctionnement du système conforme aux attentes de ses utilisateurs.



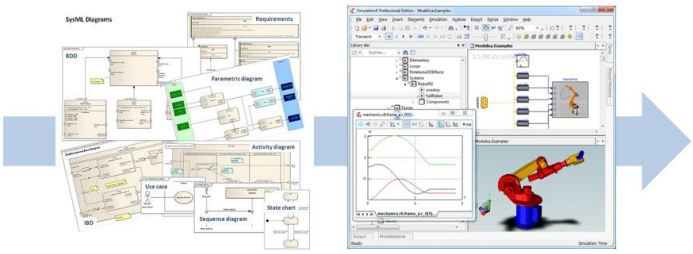
Unité de commande : fonction, entrée et sorties

Les **entrées** des SED sont des informations en provenance de l'**IHM** et des **capteurs**. Les **sorties** sont des **commandes** vers les **IMH** et les **préactionneurs** des différentes chaînes fonctionnelles.

Comme pour toutes les autres étapes de conception d'un nouveau système, il se révèle plus efficace techniquement et économiquement de modéliser la structure du programme et de la simuler, avant tout choix de langage de programmation et réalisation du programme.

Cette démarche structurée s'inscrit dans un contexte plus large d'*ingénierie système*.

Elle peut être mise en œuvre via le langage SysML et plus particulièrement grâce à l'utilisation des diagrammes d'état.



Ingénierie système : diagramme du langage SysML et logiciels de simulation associés

I Diagramme d'état (stm)

I.1 Présentation

En langage SysML, un diagramme d'état (stm) est nécessairement associé à un bloc(1) du diagramme de définitions de blocs (bdd) ou du diagramme de blocs internes (ibd). Ce bloc peut être le système, un sous-système ou un composant.

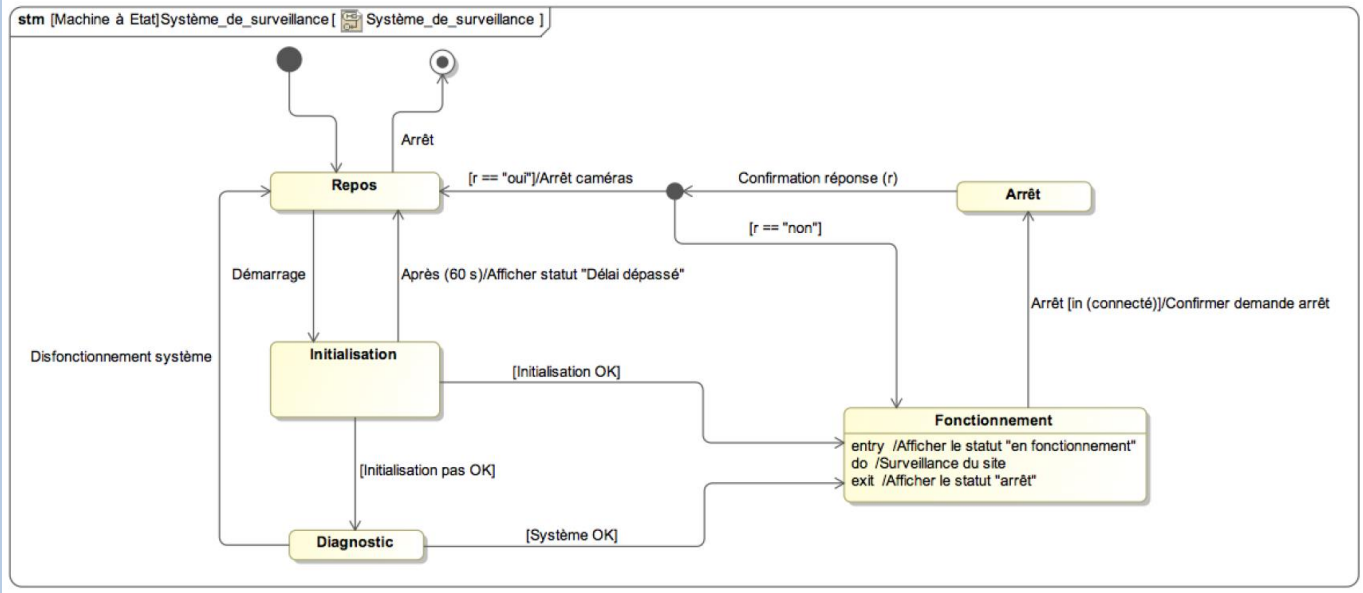
(1) Chaque bloc d'un bdd ou d'un ibd ne conduit pas nécessairement à un diagramme d'état.

> Le **diagramme d'état** représente le **comportement** du système et ses changements d'état en fonction des interactions.

Exemple : dispositif de vidéo surveillance

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

- 5 états : *Repos* , *Initialisation*, *Diagnostic*, *Arrêt* et *Fonctionnement* ;
- des transitions entre les états, représentées par des flèches, et qui précisent sous quelles conditions le système passe d'un état à un autre.



On peut remarquer, avec cet exemple, que la représentation du **comportement** est en générale fonctionnelle dans les diagrammes d'état. Aucune information technique sur la manière dont sont transmises les informations, ni sur la façon dont sont réalisées les activités, n'est précisée.

I.2 État et ses activités associées

Un **état**(1) modélise une **phase du fonctionnement** du système.

Pendant cette période, l'état est dit **actif** et le système accomplit :

- une simple **activité** ;
- OU une séquence d'activités ;
- OU est **en attente**.

En dehors de cette période, l'état est dit **inactif**.

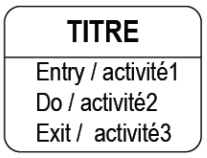
(1) Représenté graphiquement par un rectangle aux bords arrondis.

Par définition :

- il n'y a qu'**un seul état actif** à **chaque instant** ;
- un état possède un **titre unique** dans le diagramme.

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

entry	Activité ayant une **fin**(2), elle ne peut **pas** être **interrompue**. Elle est **exécutée** lors de l'**activation de l'état**.
do	Activités **interruptibles**. Elles sont **exécutées** dans l'**ordre** de leur **écriture**, à partir de l'instant où l'activité associée à *entry* est terminée.
exit	Activité ayant une **fin**(2), elle ne peut **pas** être **interrompue**. Elle est **exécutée** lors de la **désactivation de l'état**, à l'instant de la sortie de l'état.



(2) En général une activité instantanée : allumer un voyant, incrémenter un compteur, éteindre un voyant…

Remarques :

- 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) ;
- si **aucun mot** réservé n'est utilisé, cela correspond à un ***do*** ;
- un **état vide** (sans activité) indique un **état d'attente** ;

I.3 Franchissement des transitions

Une fois recensés tous les états d'un système, il faut ensuite en modéliser ses évolutions en identifiant les conditions de changement d'état.

Une **transition**(3) modélise la possibilité d'un **passage** instantané d'un **état** vers **un autre**.

On appelle **état source** l'état de départ d'une transition, et **état cible** l'état d'arrivée d'une transition

(3) Représentée graphiquement par une flèche.

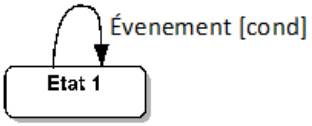
La **transition** :

- est **instantanée** ;
- n'est évaluée que si l'état source est actif.

Son **franchissement** est **conditionné** par des **événements déclencheurs** et des **conditions de garde**.

Ces événements et conditions de franchissement ainsi que l'éventuel effet associé à la transition sont indiqués le long de la flèche qui la symbolise suivant la notation :

événement [garde] / effet



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

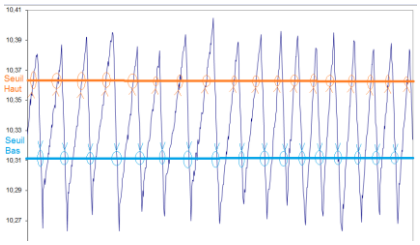
I.4 Evènement

Un **événement** correspond au changement d'état d'une variable observée. Il est **daté** dans le temps et il est **traité instantanément** lors de son **occurrence**(1) (apparition).

(1) Sur le front montant.

Exemples :

Appui sur un bouton, arrivée en fin de course d'un mécanisme, dépassement d'une valeur seuil…



Un **événement** n'est **jamais mémorisé** et est donc perdu s'il ne mène à aucune évolution du diagramme d'état.

Il est possible d'utiliser des variables internes (compteurs ou horloge) pour spécifier un événement déclencheur :

when	Se déclenche lors du changement d'état d'une valeur interne au diagramme d'état. Il permet par exemple d'utiliser un compteur : *when(N=3)*.
after (T)	se déclenche après une durée T passé dans l'état d'amont. Il permet de réaliser une temporisation.
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 une **transition** n'a **pas d'événement** spécifié, l'**événement** déclencheur **implicite** est la fin des **d'activités**(2) liées au ***do*** *de l'****état source***.

(2) Ces activités doivent donc avoir une fin.

I.5 Garde

La **garde** est une **condition de franchissement** de la transition. C'est une condition logique(3) évaluée à l'instant de l'évènement déclencheur.

(3) Vrai ou faux, généralement notée 1 ou 0.

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.

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.

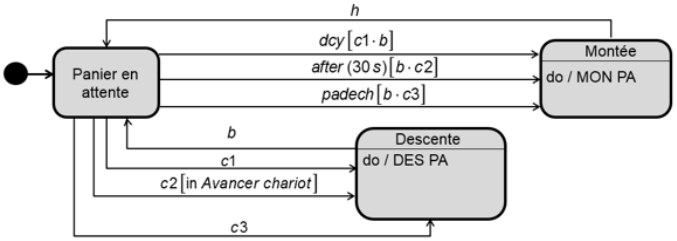
Si une **garde** n'est **pas présente** le long d'une transition, elle est **considérée** comme toujours **vraie**.

La syntaxe d'une condition de garde vérifiant si l'état TOTO est actif est : [in TOTO].

(1) On verra en pratique que cette règle n'est pas toujours respectée dans les diagrammes d'état rencontrés.

Equation logique

La condition **logique** évaluée pour la garde peut-être le résultat d'une combinaison de l'état de plusieurs grandeurs logiques, on parle alors du résultat d'une équation logique.



Dans ce type d'équation, on utilise des opérateurs logiques selon les règles de l'algèbre de BOOLE.

Ces 4 opérateurs logiques sont : **OUI**, **NON**, **OU**, **ET**. Ils permettent de réaliser les opérations de base :

(2) Remarque : le symbole = ne traduit pas une égalité mais une identité d'état (0 ou 1, Vrai ou Faux).

Entrées : **a** et **b** résultat : **S**(2)			
OUI	notée $S=a$	*qui donne à S la valeur 1 si et seulement si*	$a=1$
NON (appelé aussi « complément »)	notée $S=\overline{a}$		$a=0$
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 $b=1$

L'opérateur logique **OU** correspond à un montage en **parallèle**.

L'opérateur logique **ET** correspond à un montage en **série**.

I.6 Effet

Un **effet** est une **activité** accomplie lorsque la **transition** est **franchie**.

Les activités associées aux effets sont considérées instantanées, par exemple on peut définir une variable /N=1

Une transition peut ne pas avoir d'effet associé.

I.7 Pseudos-états

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

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

I.8 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.

I.9 Pseudo-état final

(3) Il peut y en avoir plusieurs car différents scénarios peuvent être possibles pour mettre fin à un diagramme d'état.

Optionnel(3), il signe la **fin de l'exécution du diagramme d'état**. Il n'a aucune transition sortante.

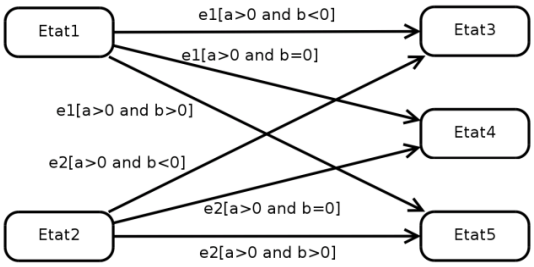
I.10 Pseudo-état jonction

Utilisé pour **regrouper** (« factoriser ») des **conditions de franchissement de transition**, en particulier des gardes communes à un événement.

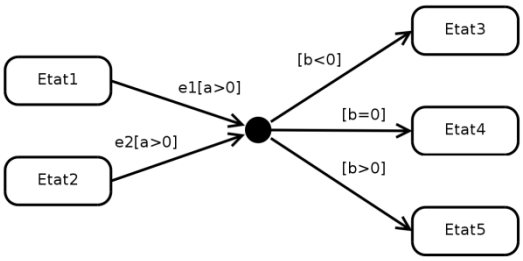
Il permet de partager des segments de transition et d'aboutir à une notation plus lisible des chemins alternatifs.

L'**évaluation** des **conditions de garde** en **aval** du pseudo-état est réalisée **avant** qu'il ne soit atteint.

Exemples : pseudo-état jonction



Exemple sans pseudo-état jonction



Exemple avec pseudo-état jonction

I.11 Pseudo-état décision

Utilisé pour une sélection ou une convergence de séquences exclusives.

(1) Contrairement au pseudo-état jonction.

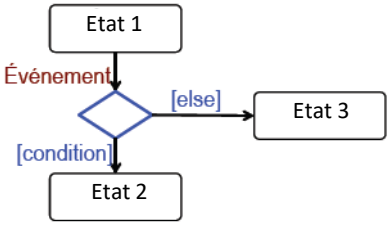
L'**évaluation** des **conditions de garde** en **aval** du pseudo-état est réalisée **au moment**(1) où il est atteint.

Les **conditions de gardes** doivent être **exclusives**.

L'utilisation d'une clause [*else*] est recommandée après un pseudo-état décision, car elle garantit un modèle correct en englobant tout ce qui n'est pas décrit dans les autres expressions logiques et en assurant ainsi qu'un moins un segment en aval est franchissable.

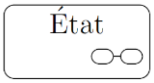
Exemple :

Ci-contre, dès que l'événement apparaît, le pseudo-état décision est atteint. Si la condition est vraie, c'est l'état 2 qui devient actif, sinon, c'est l'état 3.



I.12 État composite

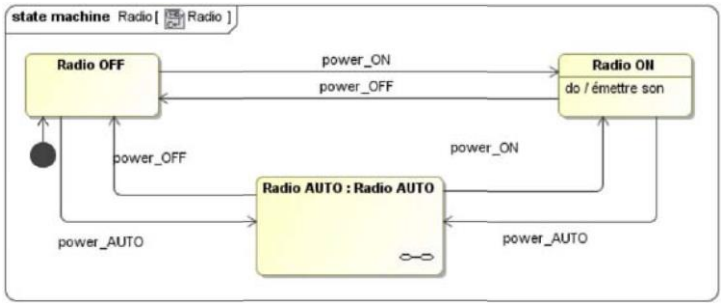
Un **état composite** décrit les **évolutions internes** d'un **état** à l'aide d'un **autre diagramme d'état**.

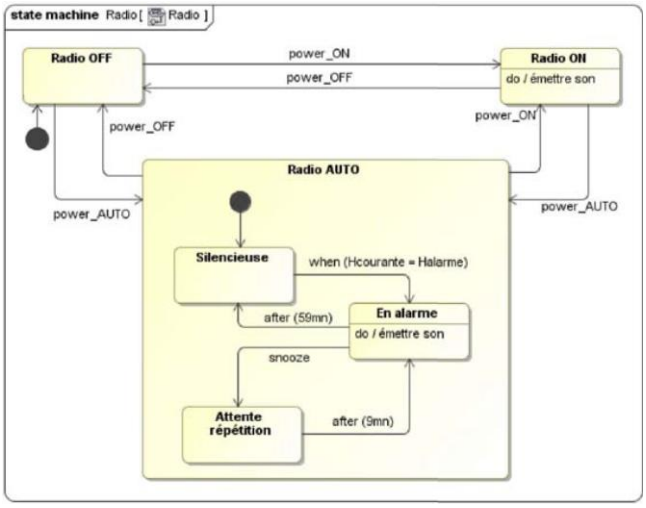


Pour repérer un état composite, un signe symbolisant des lunettes est apposé sur l'état.

Cette structure qui englobe plusieurs sous-états exclusifs 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.

Exemple : radio-réveil





Une transition qui atteint la bordure d'un état composite est équivalente à une transition qui atteint l'état initial de sa région interne.

Une transition qui sort de la bordure d'un état composite est équivalente à une transition qui sort de tous les états de sa région interne.

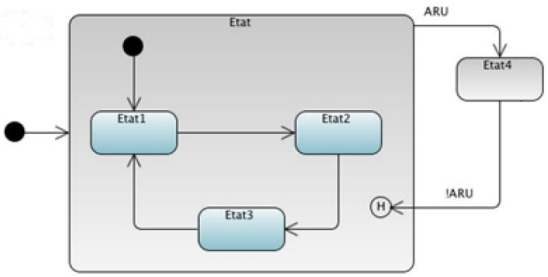
I.13 Historique d'un état composite

L'**état actif** au **moment** de la **sortie** d'un **état composite** peut être **mémorisé** par l'indication **historique**.

Lors de la **réactivation** de l'**état composite**, celui-ci se **réactive à cet état.**

Exemple :

L'*historique* est utilisé ici pour permettre à un système de recommencer en cours de cycle lors du redémarrage après un appui sur l'arrêt d'urgence (ARU).



I.14 État composite orthogonal

Dans un **état composite orthogonal**, **plusieurs diagrammes d'états** peuvent évoluer **simultanément** dans des **régions** séparées par des pointillés.

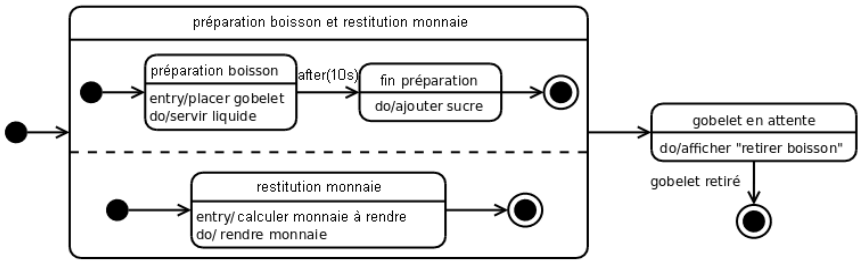
Les **différentes régions** de l'état orthogonal **fonctionnent** en **parallèle** sans **aucune influence** les unes sur les autres (1).

(1) Dans ce cas de figure plusieurs états sont actifs en même temps (un seul par région).

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**. Ce n'est qu'à cette condition que la **transition** de **sortie** de l'**état composite** devient **franchissable**.

Exemple : distributeur de boissons



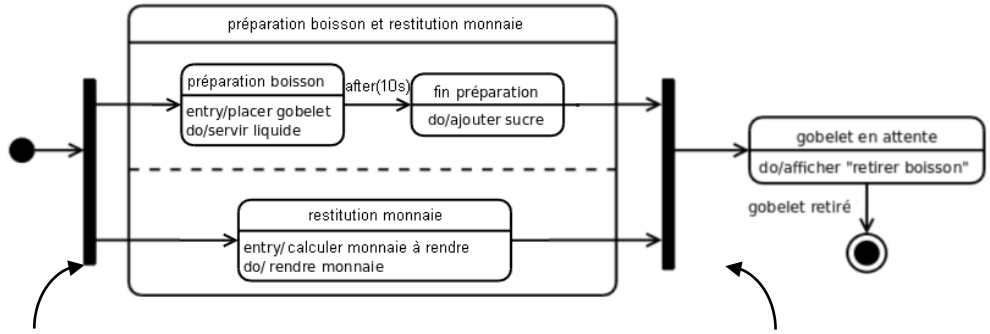
L'**activation** et la **sortie** d'un **état composite orthogonal** peuvent être également symbolisés par des **barres de synchronisation *fork*** et ***join*** qui fonctionnent par paire.

- Les **transitions,** nécessairement **automatiques**(1), qui **partent** d'une barre de synchronisation ***fork*** sont **franchies simultanément**.
- La **transition** qui **part** d'une barre de synchronisation ***join*** n'est **franchissable** qu'après le **franchissement** de **toutes** les **transitions,** nécessairement **automatiques**, qui **convergent** vers cette barre.

(1) Sans événement ni garde associée.

Exemple : distributeur de boissons

Le diagramme d'état ci-dessous décrit le même comportement que celui présenté précédemment.



II Diagramme de séquence (sd)

Un autre diagramme du langage SysML permet de décrire le **comportement** séquentiel d'un système. Il s'agit du diagramme de séquence.

II.1 Diagramme de séquence

Un diagramme de séquence est rattaché à un cas d'utilisation et décrit ce dernier en entier ou en partie, ce qui correspond à un scénario de fonctionnement possible, défini dans un cadre précis.

Il décrit, dans l'**ordre chronologique**, l'**enchaînement** des **interactions**(2) entre les **acteurs** du système ou entre des **composants** du système eux-mêmes.

(2) Appelés messages

Les principaux éléments sont :

Ligne de vie	Ligne verticale en pointillée. Une pour chaque élément dialoguant (acteur, système, sous-système ou composant).
Période d'activité	Bande verticale sur une ligne de vie. Optionnelles elles facilitent la lecture du diagramme.
Message	Flèche horizontale unidirectionnelle entre deux lignes de vie représentant un élément de communication. Elle déclenche une période d'activité (un comportement) chez le receveur du message, pour qui, l'arrivée d'un message est un événement déclencheur. Ce message peut être :

- synchrone : l'émetteur attend une réponse, son comportement est bloqué pendant l'attente ; 
- asynchrone : l'émetteur n'attend pas de réponse, son comportement continu ; 
- une réponse. 

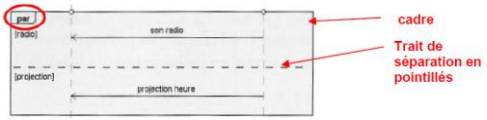
Il est possible d'avoir un message réflexif correspondant à une interaction interne au composant.

Fragment

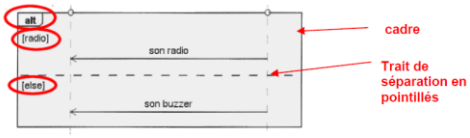
Cadre englobant une partie de la séquence

loop Boucle, le fragment est maintenu pendant une durée qui dépend de la condition de garde.

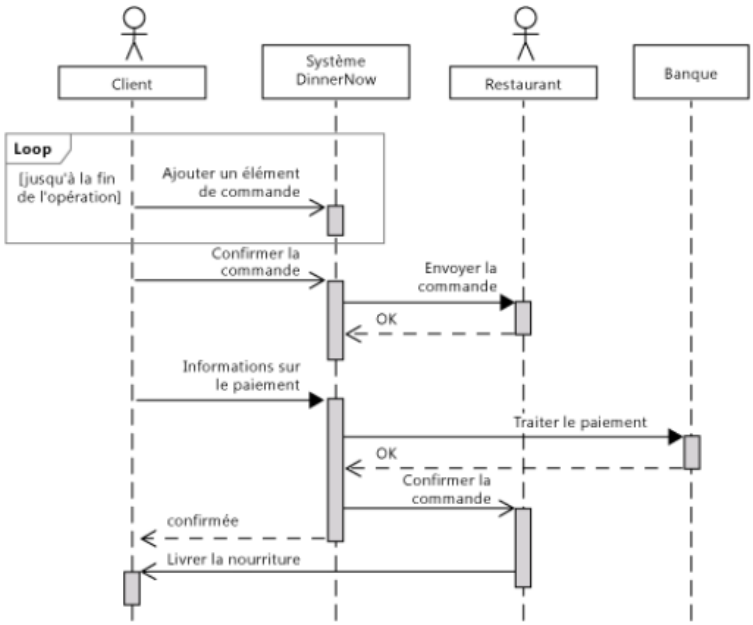
par Séquence en parallèle. Les régions peuvent être exécutées simultanément.



alt Séquences alternatives conditionnées. Seule la région dont la condition est vraie s'exécute.



Exemple : système de vente de repas en ligne



III Mesurer et coder une position angulaire

III.1 Familles de capteurs

Pour contrôler le bon fonctionnement d'une chaîne d'énergie, il est nécessaire de mesurer des grandeurs physiques. Les 3 types de capteurs qui permettent d'acquérir des grandeurs sont :

- **les capteurs :** délivre une information **analogique** (potentiomètre linéaire, rotatif, règle magnétique, cellule magnétorésistive, tachymètre , génératrice tachymétrique, accéléromètre, débitmètre, dynamomètre, jauges de déformation, cellules piézo-électriques, manomètre...)
- **les codeurs :** délivre une information **numérique** (codeur incrémental, absolu)
- **les détecteurs :** délivre une information **logique** (détecteur fin de course ILS, détecteur à effet hall, boutons...)

On distingue également ;

les capteurs actifs : Ils ont besoin dans la plupart des cas d'apport d'énergie extérieure pour fonctionner

les capteurs passifs : Le phénomène physique qui est utilisé pour la détermination du mesurande effectue directement la transformation en grandeur électrique

III.2 Vocabulaire de métrologie

(1) La beauté n'est pas une grandeur mais on peut lui associer des critères qui eux le sont.

Grandeur : Phénomène physique mesurable.(1)

Mesure : Évaluation d'une grandeur par comparaison avec une autre grandeur de même nature prise pour unité.

Mesurande : Grandeur à mesurer.

Résolution : Plus petite variation de grandeur mesurable par le capteur.

Ex : Le codeur incrémental à une résolution de 0,087°.

Sensibilité : Variation du signal de sortie par rapport à la variation du signal d'entrée.

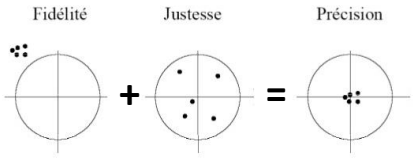
Ex : Le capteur de température LM35 a une sensibilité de 10mV/°C.

(2) La fidélité concerne le système, la justesse et la précision concernent la mesure.

Fidélité : Répétabilité de la mesure.(2)

Justesse : Réponse proche de la valeur vraie.

Précision : Écart entre la valeur vraie et la valeur mesurée.



III.3 Les codeurs

Le codeur est généralement placé en amont du réducteur, car pour un même mouvement, on obtient plus d'impulsion et donc une meilleure résolution.

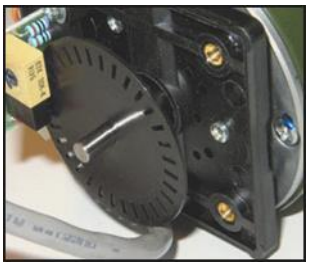
Codeur incrémental (ou roue codeuse)

Codeur incrémental (ou roue codeuse) :

Un codeur incrémental est un générateur d'**impulsions** qui fournit 2 voies en **quadrature** et un top zéro. Elles sont divisées en n secteurs angulaires égaux, alternativement opaques et transparents.

Ils fonctionnent sur le principe de **comptage** et **décomptage d'impulsions** et donne donc le déplacement relatif.

n s'appelle le nombre de périodes, c'est le nombre d'impulsions qui sont délivrées par le codeur pour un tour complet de son disque.



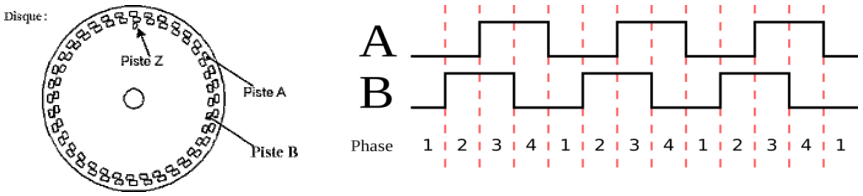
Codeur incrémental

Avantages :

- Mesure prise à coût raisonnable ;
- Entrées de comptage adaptées (voies A, B, Z) en standard sur les automates programmables récents ;
- Obtention aisée de la vitesse par intégration numérique.

Inconvénients :

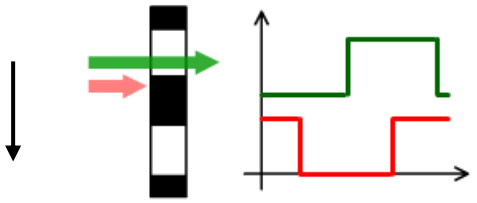
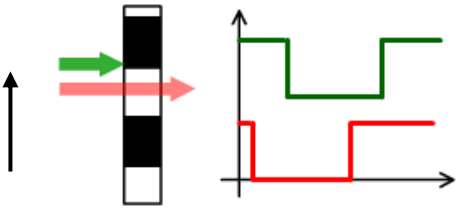
- Perte totale des informations en cas de coupure d'énergie ;
- Nécessite une procédure de prise d'origine.



Disque d'un codeur incrémental et pistes en quadrature de phase

Les pistes intérieures et extérieures sont en **quadrature de phase**, ce qui permet de :

- connaitre le **sens de rotation** du capteur
- **améliorer par 2 la résolution** du capteur.

Déterminer le sens de rotation	
	
Le front montant de la voie verte se présente avant celui de la voie rouge.	Le front montant de la voie rouge se présente avant celui de la voie verte.

Codeur absolu (ou numérique)

Codeur absolu (ou numérique) :

Délivre un signal image de la position à mesurer **sous forme d'un code numérique binaire** et donne le déplacement absolu.

Il dispose de N pistes, généralement agencées suivant le code **Gray**.

La piste intérieure correspond au bit de poids le plus fort.

Codeur absolu

Avantages :

- Chaque secteur possédant son code unique, il est inutile de déterminer le sens de rotation ;
- Pas de perte d'information en cas de coupure d'énergie ;
- Code connu en permanence, pas besoin de procéder à la Prise d'Origine Machine lors de la mise sous tension ;
- Pas d'erreur de lecture avec le code Gray.

Inconvénients :

- Relativement onéreux ;
- Interface avec la commande plus complexe (N entrées) ;
- Nécessite un transcodeur pour reconvertir le signal en binaire naturel.

Disque et pistes d'un codeur absolu

(1) Le code Gray n'est pas pondéré, on ne peut pas faire d'opérations arithmétiques avec.

Le **code Gray**(1) , également appelé binaire réfléchi, est un code **binaire** qui présente la particularité qu'**un seul bit change d'état entre deux combinaisons successives**.

Ce qui permet d'éviter des erreurs de lecture.

Code binaire

	2^2	2^1	2^0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Code Gray

	2^2	2^1	2^0
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

III.4 Systèmes de numération

Base binaire : Pour un nombre codé sur $n+1$ (signe) bits, chaque bit i vaut 2^i en décimal (Décimal Codé Binaire DCB).

Base octale : Peut compter de 0 à 7. Équivaut à un regroupement de 3 bits.

Base hexadécimale : Peut compter de 0 à 15, (avec 0, 1, ..., 8, 9, A, B, ..., E, F). Équivaut à un regroupement de 4 bits. Les octets sont souvent écrits sous la forme de deux chiffres hexadécimaux en informatique.