La modélisation multiphysique

PHILIPPE FICHOU [1]

Au côté de SysML, la modélisation multiphysique a fait son apparition dans les programmes. Tour d'horizon des concepts à connaître sur les modèles, les langages et les logiciels permettant une démarche d'ingénierie système.

mots-clés
logiciel, modélisation, SysML

La réécriture des programmes de sciences industrielles de l'ingénieur en classes préparatoires aux grandes écoles donne l'opportunité d'une actualisation des outils à employer, en lien étroit avec les évolutions des enseignements au lycée, en écoles d'ingénieurs et dans le domaine industriel. En 2013, de nouveaux outils de description et de modélisation des systèmes sont introduits, remplaçant ou complétant ceux mis en œuvre jusque-là. À coté du schéma-bloc pour modéliser les systèmes linéaires continus et invariants, la description des systèmes à l'aide du langage graphique SysML et la modélisation multiphysique font leur apparition. C'est le moment de préciser les notions de modélisations causale et acausale, et de souligner l'intérêt de coupler l'utilisation de SysML et celle de Modelica dans une démarche d'ingénierie système guidée par les modèles, à laquelle les langages « bond graph » et ModelicaML s'intègrent totalement, notamment en tant que profils de SysML.

La modélisation des systèmes en CPGE

Schéma-bloc et modélisation multiphysique

Le schéma-bloc est une représentation largement utilisée dans le cadre de l'enseignement des sciences industrielles de l'ingénieur en classes préparatoires aux grandes écoles pour décrire le comportement d'un système linéaire continu et invariant. Une autre représentation, qui n'est pas au programme de ces classes, est également largement utilisée, à savoir la représentation d'état. Un bloc de la représentation par schéma-bloc possède la forme d'état suivante

$$\begin{cases} \dot{x} = f(t, x, u) \\ y = g(t, x, u) \end{cases}$$

u : vecteur des signaux d'entrée

x : vecteur de l'état interne

y : vecteur des signaux de sortie

Quand les blocs sont connectés, le modèle obtenu représente et permet la simulation des systèmes d'équations différentielles qui traduisent le comportement du système étudié.

Le schéma-bloc est un outil pour décrire et comprendre le comportement dynamique d'un système décrit par des équations différentielles traduites en équations algébriques par la transformation de Laplace. Il est ainsi très aisé d'en déduire la fonction de transfert du système, ses zéros, ses pôles, ses comportements temporel et fréquentiel dans le cas des systèmes linéarisés. Notons également l'intérêt d'un schéma-bloc pour étudier la commande d'un système.

Les liens entre les blocs d'un schéma-bloc « transportent » une seule information et ne peuvent donc traduire que de façon incomplète les transferts d'énergie dans un système.

D'autres outils, comme les bond graphs, peuvent être utilisés pour obtenir une modélisation plus fidèle de la structure du système et des échanges énergétiques entre les composants. En effet, dans la modélisation par bond graph des systèmes, les liens entre ceux-ci « transportent » deux informations dont le produit est un flux d'énergie, c'est-à-dire une puissance. Basée sur l'analogie avec les domaines physiques, la modélisation par bond graph permet « naturellement » la description multiphysique à partir d'éléments « invariants » d'un domaine à l'autre (éléments actifs, éléments passifs, jonctions) [1] [2][2]. Des liens d'information interviennent dans la modélisation de la mesure et de la commande des systèmes.

L'outil bond graph a fait son apparition en CPGE en 2003 – en classe de physique-technologie –, mais son formalisme a quelque peu rebuté les enseignants : il ne fera pas partie des outils inscrits aux programmes qui seront mis en place à la rentrée 2013. La modélisation multiphysique, par contre, a fait son apparition dans les programmes du lycée à la rentrée 2011, que ce soit en STI2D ou dans la filière scientifique du lycée général. Bien sûr, à ce niveau, il n'est pas question de construire des modèles complexes ou même d'entrer dans le détail des modèles mis en œuvre ; il s'agit de présenter à la lecture une modélisation « proche » du système étudié par la description structurelle de ce

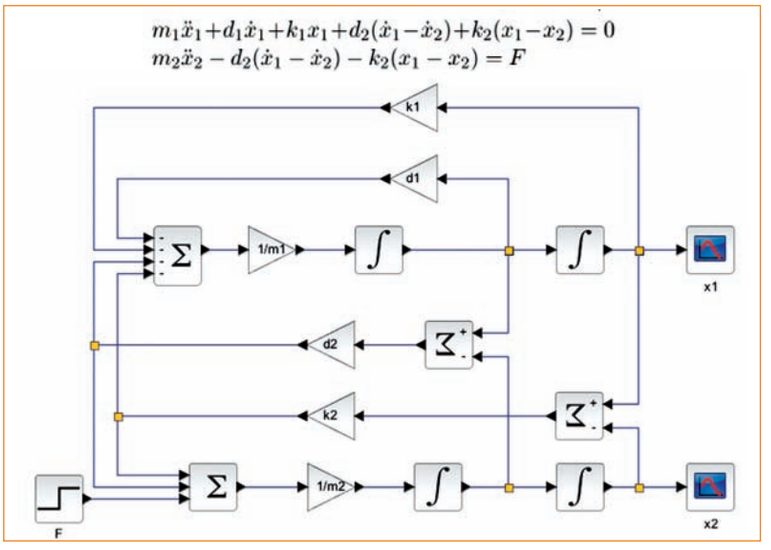
[1] Professeur en CPGE au lycée Chateaubriand de Rennes (35).

[2] Les chiffres gris entre crochets renvoient aux références en encadré.

dernier. Les simulations numériques raisonnées sont alors un outil pédagogique pertinent, notamment pour apprécier les écarts entre le réel et le modèle.

Il est donc naturel que les programmes de CPGE prennent également en compte cette évolution.

À ce niveau, une réflexion plus approfondie peut être menée, d'autant plus que la modélisation multiphysique s'inscrit dans le prolongement de la description SysML, également introduite dans les programmes de 2013. Le diagramme de blocs internes, le diagramme paramétrique, le diagramme d'activité ou le diagramme d'état de ce langage peuvent, par exemple, être construits et simulés à l'aide des logiciels disponibles aujourd'hui. La description ainsi proposée est proche de la structure réelle des systèmes, ce qui n'est pas toujours le cas avec les schémas-blocs. Nous pouvons d'ailleurs relever qu'un schéma-bloc n'est pas forcément plus parlant que le système d'équations différentielles qu'il représente 1. De plus, les modèles multiphysiques sont aisément réutilisables, grâce à la création de bibliothèques. Les logiciels mis en œuvre dans la modélisation multiphysique permettent de construire des modèles acausaux, contrairement à ceux qui simulent les schémas-blocs, qui sont quant à eux des modèles causaux. Nous allons, entre autres choses, tenter de préciser les deux approches.



1 **Un système différentiel et le schéma-bloc associé**

Modélisation causale et modélisation acausale

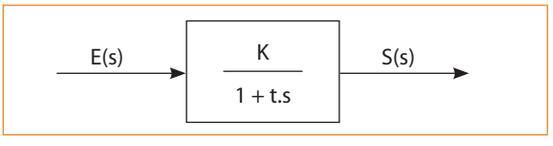
La modélisation par schéma-bloc est typiquement une modélisation causale. En effet, un bloc représentant une fonction de transfert possède une entrée et une sortie et est donc « orienté », c'est-à-dire ici qu'il a une cause – son entrée E(s) – et un effet – sa sortie S(s). La figure 2 représente un bloc pour une fonction de transfert du premier ordre. On a donc

$$S(s) = \frac{K}{(1 + \tau\, s)} E(s)$$

et l'on sait que la réponse temporelle à un échelon $e(t)$ = E0 × $u(t)$ est de la forme

$$s(t) = K\, E_0[1 - \exp(-\frac{t}{\tau})]u(t)$$

L'équation ci-dessus est orientée dans le sens où c'est l'entrée $e(t)$ qui impose la sortie $s(t)$ et non l'inverse, et ce, par la construction même du schéma-bloc. L'équation différentielle traduisant le comportement du modèle est :



2 **Bloc modélisant une fonction de transfert du premier ordre**

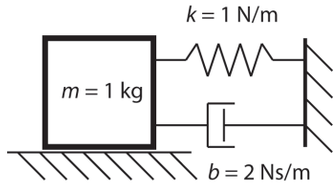
$$\tau \frac{ds(t)}{dt} + s(t) = K\, e(t)$$

Elle donne ainsi la forme causale du modèle. Cette équation est donc également orientée (relevons ici que, pour un système réel, l'ordre de dérivation de la sortie est nécessairement supérieur ou égal à l'ordre de dérivation de l'entrée). Nous pourrions écrire l'équation dans le domaine de Laplace sous la forme

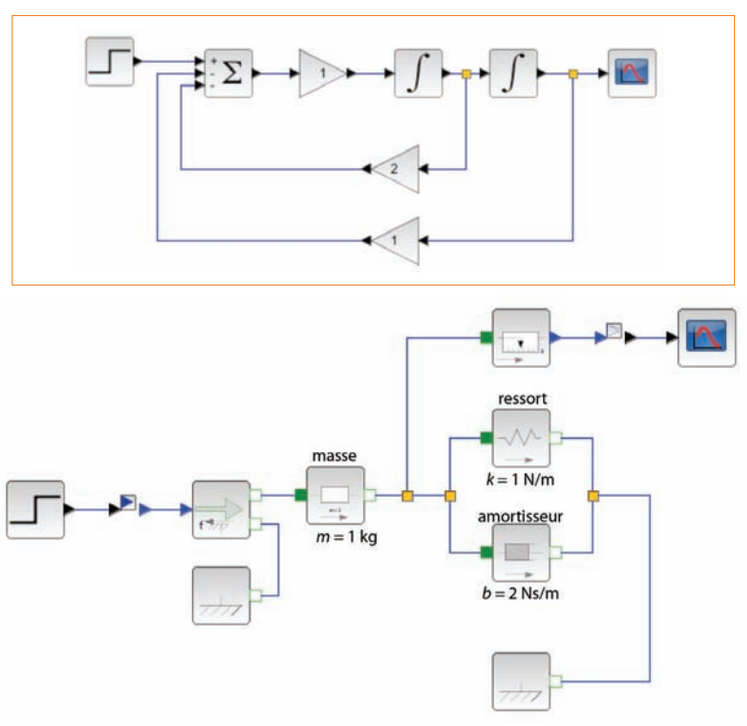
$$S(s) := \frac{K}{(1 + \tau\, s)} E(s)$$

ce qui indiquerait que la sortie est calculée à partir de l'entrée. Un schéma-bloc traduit ainsi l'algorithme de calcul du comportement du système plus que sa composition structurelle. Il est en quelque sorte proche d'un langage informatique procédural. Le flux d'information indiqué par le sens des flèches du schéma-bloc est explicite : le modèle causal sera simulé en utilisant la propagation des valeurs avant et après intégration.

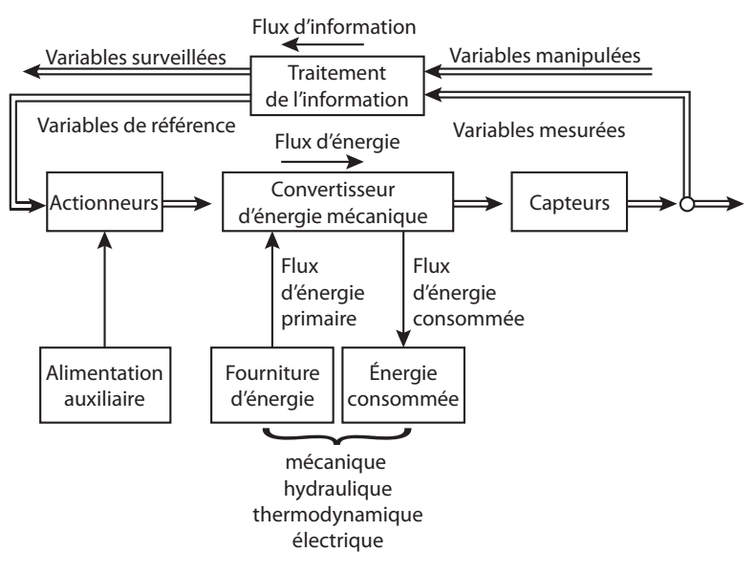
La modélisation acausale, quant à elle, ne procède pas de cette approche. Comme son nom l'indique, un modèle acausal ne présente pas de lien de cause à



3 Le modèle élémentaire masse-ressort-amortisseur



4 Les modèles causal et acausal du système masse-ressort-amortisseur



5 L'architecture d'un système

effet entre une entrée et une sortie potentielles du modèle. La modélisation acausale recourt à des équations implicites sans ordre entre des variables, donc sans spécification d'entrées et de sorties. Un modèle acausal comporte des variables et des relations entre ces variables qui agissent comme des contraintes entre les valeurs des variables à chaque instant. Un modèle acausal contient des objets qui représentent les constituants du modèle réalisant sa structure interne et son comportement. Il relève de la programmation orientée objet couramment utilisée aujourd'hui en informatique.

Prenons pour premier exemple élémentaire une modélisation causale et une modélisation acausale d'un système mécanique masse-ressort-amortisseur, équivalant, par analogie, au circuit électrique inductance-condensateur-résistance 3 4.

Le modèle causal est conçu par l'ingénieur simultanément avec l'expérimentation et la simulation qu'il souhaite réaliser. Chacun des blocs de ce modèle représente une ou plusieurs opérations mathématiques (équations algébriques et différentielles).

Le modèle acausal explicite quant à lui la structure du système étudié sans envisager *a priori* l'expérimentation et la simulation du modèle. Dans notre exemple, une action mécanique est appliquée à la masse, et un capteur mesure sa position. C'est à ce moment que l'« expérimentation » est conduite, et donc que la simulation est mise en œuvre.

On pourrait parfaitement imaginer une autre expérimentation avec le même modèle masse-ressort-amortisseur en modifiant l'actionneur et le capteur. Pour la modélisation causale, il faudrait reconstruire le modèle pour réaliser une nouvelle expérimentation numérique.

Prenons maintenant l'exemple heuristique du moteur à courant continu. Lorsque l'on construit le schéma-bloc à partir des équations issues de la connaissance du système, on obtient un modèle causal pour lequel l'entrée est la tension appliquée à ses bornes, et la sortie la fréquence de rotation du rotor du moteur. Nous avons ainsi construit le modèle et l'expérience virtuelle qui va avec, à savoir la tension à appliquer pour obtenir la fréquence de rotation souhaitée.

Si maintenant on cherche à connaître la loi de commande à appliquer pour obtenir une gamme de vitesse angulaire du rotor, on est contraint de redéfinir le modèle, donc le schéma-bloc correspondant aux équations inversées du moteur, car il s'agit d'une nouvelle expérimentation virtuelle. Avec une modélisation acausale, le même modèle pourra être utilisé à la fois en direct et en inverse, par exemple. C'est évidemment très important au niveau industriel, puisque le temps gagné peut être employé à bien autre chose. Une sensibilisation à cette notion de modèle réutilisable est donc la bienvenue en CPGE.

La modélisation d'un système multiphysique

L'architecture globale d'un système mécatronique est rappelée en 5. En nous appuyant sur la direction assistée électrique (DAE) de la Twingo équipant certains laboratoires de sciences industrielles de l'ingénieur en CPGE 6, nous allons nous familiariser avec quelques langages de modélisation multiphysique.

Les paramètres localisés

Modéliser, c'est toujours simplifier, mais c'est indispensable, car on ne raisonne que sur des modèles. Cette simplification doit évidemment être adaptée à l'objectif de cette modélisation. Le niveau de détail doit être suffisant pour faire apparaître les paramètres souhaités dans la gamme d'étude envisagée. Une solution couramment pratiquée est la modélisation à paramètres localisés traduisant les équations différentielles issues de principes physiques. Il s'agit ici de localiser dans la chaîne d'énergie les éléments caractéristiques du système, et d'écrire les équations entre ceux-ci. La figure 7 présente un modèle à paramètres localisés de la DAE. Il est alors aisé, en appliquant les principes physiques, d'en déduire le système d'équations traduisant les mouvements du système :

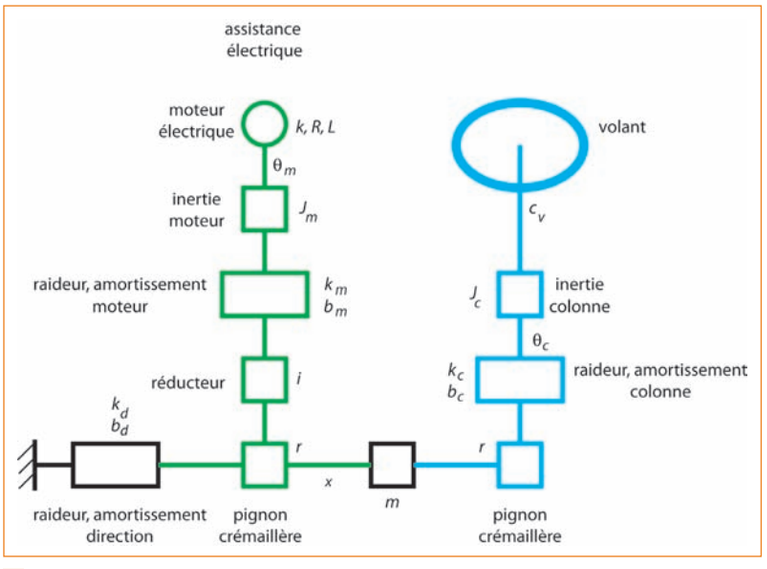
$$\begin{cases} L\dfrac{di}{dt} + Ri + k\theta_m = u \\ J_c\ddot{\theta}_c + b_c\dot{\theta}_c - k_c\left(\theta_c - \dfrac{X}{r}\right) = c_v \\ J_m\ddot{\theta}_m + b_m\dot{\theta}_m - k_m\left(\theta_m - \dfrac{Xi}{r}\right) = ki \\ m\ddot{X} + b_d\dot{X} + k_dX = \dfrac{k_c}{r}\left(\theta_c - \dfrac{X}{r}\right) + \dfrac{k_m i}{r}\left(\theta_m - \dfrac{Xi}{r}\right) \end{cases}$$

Pour d'autres types de modélisation, le choix de paramètres localisés n'est pas adapté à l'objectif visé (par exemple la finesse du modèle). Il est dans ce cas indispensable de choisir des paramètres répartis (dans l'espace et/ou dans le temps), et le modèle sera alors régi par des systèmes d'équations aux dérivées partielles. Leur simulation pourra être obtenue par des méthodes numériques (éléments finis, différences finies, équations intégrales, etc.). C'est évidemment le rôle de l'ingénieur d'opter, au moment opportun de la conception, pour tel ou tel type de modélisation.

Les bond graphs

La traduction du modèle à paramètres localisés sous forme graphique est parfaitement décrite par le langage des bond graphs, dont la base est le transfert d'énergie entre les constituants d'un système. La description de l'état d'un système multiphysique nécessite l'utilisation de deux ensembles de variables fondamentales. Au nombre de quatre, elles sont reliées entre elles par les relations mathématiques décrites ci-après. Elles dépendent du temps *t*. En fonction du système étudié, ces variables peuvent avoir plus d'une dimension : elles sont alors exprimées à l'aide de vecteurs. Deux regroupements sont possibles :

6 **La maquette didactisée de la direction assistée électrique de la Twingo**



7 **La localisation des paramètres remarquables de la chaîne d'énergie de la DAE**

Groupement 1

variables de puissance, effort *e* et flux *f*

variables d'énergie, moment *p* et déplacement *q*

Groupement 2

variables cinématiques, déplacement *q* et flux *f*

variables cinétiques, moment *p* et effort *f*

Les termes *cinématique* et *cinétique* employés dans le second groupement pourraient être considérés comme un abus de langage, mais il faut comprendre les dénominations des variables dans un cadre plus général que le cadre classique dans lequel elles ont été définies (par exemple, le terme *effort* et la mécanique).

Le groupement 1 est issu de propriétés énergétiques caractéristiques des systèmes quel que soit le

Domaine	Effort *e*	Flux *f*	Moment *p*	Déplacement *q*
Mécanique translation	Force F	Vitesse *v*	Quantité de mouvement *p*	Position *x*
Mécanique rotation	Couple τ	Vitesse angulaire ω	Moment cinétique σ	Angle θ
Électrique	Tension *u*	Intensité *i*	Flux magnétique φ	Charge *q*
Hydraulique	Pression *p*	Débit volumique q_v	Moment de pression r	Volume V
Thermique	Température T	Flux d'entropie $\dot{S}$	Entropie S	
Chimique	Potentiel chimique μ	Flux molaire $\dot{n}$	Nombre de moles *n*	

8 **Les analogies multiphysiques**

domaine physique mis en jeu. En effet, le transfert de puissance P(*t*) entre les sous-systèmes ou les composants de systèmes est une notion générale indépendante du domaine physique. Ainsi, on montre qu'une puissance est toujours le produit de deux variables, et la convention prise par le langage des bond graphs correspond au produit d'un effort *e*(*t*) par un flux *f*(*t*) :

$$P(t) = e(t) \cdot f(t)$$

Nous reviendrons plus loin sur les notions mathématiques sous-jacentes. Si l'on note E(*t*) l'énergie accumulée entre t_0 et *t*, on a

$$E(t) = E(t_0) + \int_{t_0}^{t} e(\tau) \cdot f(\tau) d\tau$$

Par ailleurs, on a respectivement les relations suivantes entre, d'une part, effort et moment et, d'autre part, flux et déplacement :

$$p(t) = p(0) + \int_{t_0}^{t} e(\tau) d\tau$$
$$q(t) = q(0) + \int_{t_0}^{t} f(\tau) d\tau$$

En dérivant temporellement ces deux dernières relations, il vient

$$\frac{dp(t)}{dt} = e(t)$$
$$\frac{dq(t)}{dt} = f(t)$$

En remplaçant, dans la relation donnant l'énergie ci-dessus, respectivement l'effort et le flux précédemment calculés, nous obtenons

$$E(p) = E(p_0) + \int_{p_0}^{p} f(p) dp$$
$$E(q) = E(q_0) + \int_{q_0}^{q} e(q) dq$$

Ces dernières relations donnent une interprétation de l'appellation *variables d'énergie* attribuée au moment *p* et au déplacement *q*. Le tableau **8** présente les quatre variables ci-dessus pour cinq domaines physiques différents. Chaque domaine physique est décrit à l'aide d'éléments de base permettant d'exprimer les interactions au sein des systèmes. Ces éléments représentent les phénomènes liant les variables précédemment citées.

On distingue quatre catégories d'éléments :

Éléments actifs : Se et Sf

Éléments passifs : I, C et R

Détecteurs : De et Df

Éléments de jonctions : 0, 1, TF et GY

Les éléments actifs sont les sources d'effort et de flux. Ces éléments permettent de représenter les contraintes imposées au système étudié par son environnement, et de spécifier ainsi les conditions aux frontières du système.

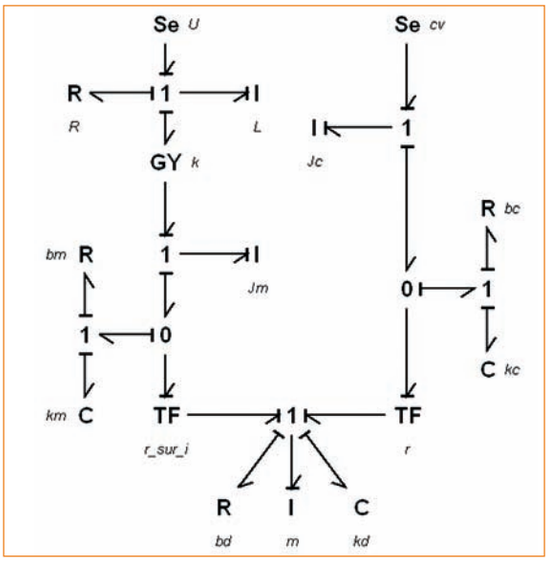
Les éléments passifs consomment la puissance en stockant l'énergie potentielle (élément C), l'énergie cinétique (élément I) ou en dissipant l'énergie en chaleur (élément R).

Les détecteurs idéaux (ne consommant pas d'énergie) mesurent les variables d'effort (De) et de flux (Df) de la chaîne d'énergie.

Les éléments de jonction permettent le couplage des éléments précédents : les jonctions 0 (iso-effort) et 1 (isoflux) traduisent les lois de la physique, et les jonctions TF (transformateur) et GY (gyrateur) transforment respectivement les variables d'un type en variables de même type (effort-effort et flux-flux) et les variables d'un type en variables de l'autre type (effort-flux et flux-effort). Le tableau **9** présente les éléments du langage bond graph en leur associant des exemples. La figure **10** propose une modélisation par bond graph de la DAE de la Twingo. Nous retrouvons évidemment les paramètres localisés identifiés à la section précédente. Les demi-flèches traduisent un flux d'énergie entre les éléments. Elles « transportent » deux variables dont le produit est une puissance. Le bond graph est acausal par construction, mais devient causal lors de la résolution par un logiciel. Cette causalité est traduite par un trait perpendiculaire à la demi-flèche du côté de l'élément de jonction où l'effort est imposé.

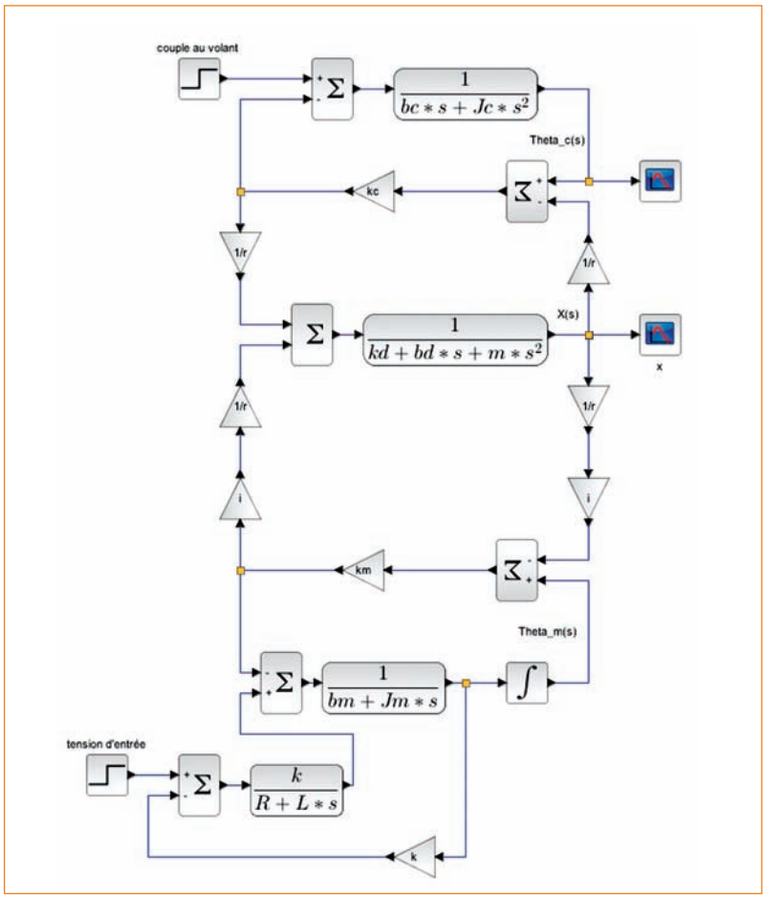
Élément	Symbole	Loi générique	Exemples
Éléments actifs	Se	*e* indépendant de *f*	Générateur de tension, générateur de pression, pesanteur
	Sf	*f* indépendant de *e*	Générateur de courant, générateur de débit
Éléments passifs	I	$\Phi_I(p, f) = 0$	Inductance, inertie, masse
	C	$\Phi_C(e, q)$	Condensateur, ressort, réservoir
	R	$\Phi_R(e, f)$	Résistances électrique et thermique, amortisseur, restriction hydraulique
Détecteurs	De		Voltmètre, manomètre
	Df		Ampèremètre, tachymètre
Jonctions	0	$e_1 = e_2 = \ldots = e_n$	Couplage en parallèle en électrique et hydraulique, couplage en série mécanique
	1	$f_1 = f_2 = \ldots = f_n$	Couplage en série en électrique et hydraulique, couplage en parallèle mécanique
	TF	$e_1 = m \cdot e_2$ $f_2 = m \cdot f_1$	Transformateur électrique, levier, réducteur mécanique
	GY	$e_1 = r \cdot f_2$ $e_2 = r \cdot f_1$	Conversion électromécanique

9 Éléments du langage bond graph



10 Le modèle bond graph de la DAE

Les équations de la DAE données ci-dessus peuvent se traduire sous la forme d'un schéma-bloc (une procédure systématique de passage d'un modèle bond graph causal à un modèle schéma-bloc est donnée par exemple par Geneviève Dauphin-Tanguy [3]). Comme déjà indiqué, celui-ci ne représente pas la structure du système, mais la forme causale du calcul qui sera mené par le logiciel qui va suivre l'algorithme du calcul. La figure 11 est réalisée avec le logiciel libre Scilab/Xcos, qui peut simuler ce type de modélisation.



11 Le schéma-bloc sous Scilab/Xcos de la DAE

Modèle acausal et langage Modelica

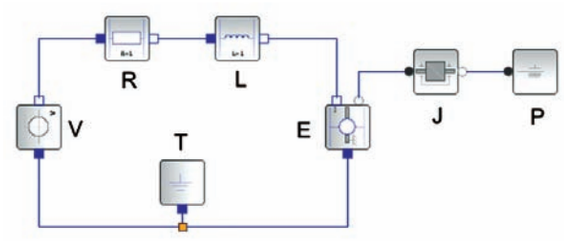
La modélisation acausale devient de plus en plus courante, pour les raisons que nous avons vues. Parmi les langages utilisés, Modelica, qui est avant tout un langage de programmation libre, s'est enrichi au fil du temps de nombreuses bibliothèques graphiques permettant au concepteur de créer des modèles multiphysiques pertinents. Les composants peuvent être assemblés et connectés comme le fait un concepteur qui réalise un prototype. Modelica se prête bien à la conception par décomposition, et de nombreux logiciels libres ou payants se basent sur ce langage. Langage acausal, Modelica est fondé sur les équations liant les variables et les paramètres des modèles sans assigner de causalité. Langage orienté objet, il est extrêmement intéressant pour la création de bibliothèques de modèles réutilisables, et possède les caractéristiques propres à ce type de langages :

- **Objet** : abstraction d'un élément réel qui possède une identité, un état (attributs) et un comportement (méthodes).
- **Classe** : caractéristiques communes à plusieurs objets et mécanismes permettant de créer des objets ayant ces propriétés.
- **Instance** : objet d'une classe.
- **Méthode** : suite d'instructions manipulant caractéristiques et état d'un objet.
- **Encapsulation** : protection des informations contenues dans un objet.
- **Héritage** : un objet « fils » hérite des propriétés de l'objet « père ».
- **Polymorphisme** : faculté d'une opération de s'appliquer à des objets de classes différentes.

L'objet est un concept qui a été créé pour regrouper dans une même entité un ensemble de procédures portant sur un ensemble de données.

La programmation avec Modelica est organisée autour d'un ensemble de classes détaillant les éléments utilisés pour décrire les modèles. Une classe est une famille d'objets (ses instances) qui partagent des propriétés communes. Les modèles sont essentiellement basés sur des connecteurs, des variables (de types flux et potentielle, voir ci-après) et des contraintes (équations différentielles et algébriques, algorithmes). Le langage textuel a permis de générer des bibliothèques de composants décrits sous forme graphique qui affranchissent l'ingénieur de l'écriture fastidieuse des éléments de programmation sous-jacents. L'intérêt d'utiliser la modélisation acausale avec des élèves et des étudiants devient évident, puisque le langage textuel est alors transparent. De plus, l'architecture des constituants du modèle d'un système est proche du réel, au sens où il est possible d'observer, de comprendre et d'agir sur les paramètres d'un modèle représentatif d'un système comme lors d'une expérimentation sur le réel. Cette proximité peut d'ailleurs permettre de montrer les différences entre modèle et réel et les limites d'une modélisation.

Modelica devient progressivement un standard pour la modélisation acausale, c'est sans aucun doute pour cela que son utilisation dans l'enseignement trouve son sens pour présenter les modèles multiphysiques aux élèves, d'autant plus que l'iconographie utilisée permet de bien s'approprier les constituants modélisés. La modification d'un modèle ou le changement d'un paramètre sont également très aisés. Les logiciels payants utilisant Modelica ont évidemment développé des bibliothèques de composants très riches, bien documentées, qui permettent aux élèves d'utiliser, voire de construire, en fonction de leur niveau, une grande variété de modèles avec des détails d'une grande




```
model MCC
Resistor R(R=6.8) ;
Inductor L(L=2e-3) ;
VsourceDC V(f=10) ; Ground T ;
ElectroMechanicalElement E(k=0.434) ;
Inertia J(J=3e–5) ;
Free P ;
equation
connect(V.p,R.n) ; connect(R.p,L.n) ;
connect(L.p,E.n) ; connect(E.p,V.n) ;
connect(V.n,T.p) ;
connect(E.flange,J.flange_a) ;
connect(J_flange_b,Free.flange_a)
end MCC
```

12 Un exemple de modèle de MCC utilisant le langage Modelica

```
partial model TwoFlanges
"Base class for a component withh two translational
1D flanges" Interfaces.Flange_a flange_a;
Interfaces.Flange_b flange_b;
end TwoFlanges
model Ideal Spring "Linear 1D translational spring"
extends Interfaces.Compliant; parameter SI.Distance
s_rel0=0
"unstretched spring lenght"; parameter
Real c(
final unit="N/m", final min=0) = 1 "Raideur";
equation
f = c*(s_rel - s_rel_0);
end Spring;
```

13 Le modèle du ressort linéaire en langage Modelica

Modelica	SysML – IBD	SysML – act	Bond graph
Class model	Block	Action node	Nœud bg
Connector	Port	Pin class	Energy port
Connection	Connector	Activity edge	Bond
Variable (connector)	Port.Portname Port.Typename	Pin property	Effort, flux
Variable (class model)	Block property	Action property	Node property

14 L'analogie entre les éléments de différents langages (d'après [4])

précision. Quant aux logiciels libres, ils autorisent très largement la réalisation de modèles suffisamment pertinents pour un emploi dans nos classes. La figure 12 présente la forme textuelle d'un modèle du moteur à courant continu en langage Modelica ainsi qu'une représentation graphique de ce même modèle. On voit bien apparaître sur cette forme graphique le circuit électrique (V, E, R, L) et l'inertie du rotor (J), ainsi que le couplage électrique-mécanique dont le paramètre entré dans le modèle est la constante k telle que E = kΩ. La figure 13 montre le programme du composant « ressort ».

Notons, pour conclure ce paragraphe, que certains éléments de Modelica ont une analogie avec ceux d'autres langages, comme le présente le tableau 14.

Schéma-bloc et langage Scilab/Xcos

Un schéma-bloc est construit à partir des fonctions de transfert des constituants d'un système et éventuellement d'éléments non linéaires permettant de compléter et d'approcher au mieux le réel.

Pour construire un schéma-bloc, il est indispensable d'identifier les variables connues et les variables à calculer. À partir des équations régissant le comportement du modèle, la transformation de Laplace permet d'écrire les fonctions de transfert et donc de rendre causal le modèle, comme nous l'avons vu. Le schéma-bloc décrit l'algorithme de résolution qui fournit la sortie étudiée à partir de l'entrée considérée. Le schéma-bloc représente donc non pas un agencement de constituants comme c'est le cas pour un modèle acausal, mais bien le schéma de calcul. La modélisation acausale est, à ce titre, plus pertinente car plus proche du réel. Notons que de nombreux logiciels permettent la simulation des schémas-blocs pour visualiser les évolutions des sorties en fonction des entrées du schéma.

Les variables conjuguées

Un concept important pour l'élaboration de modèles à paramètres localisés est celui de variables duales ou conjuguées. Elles apparaissent notamment à

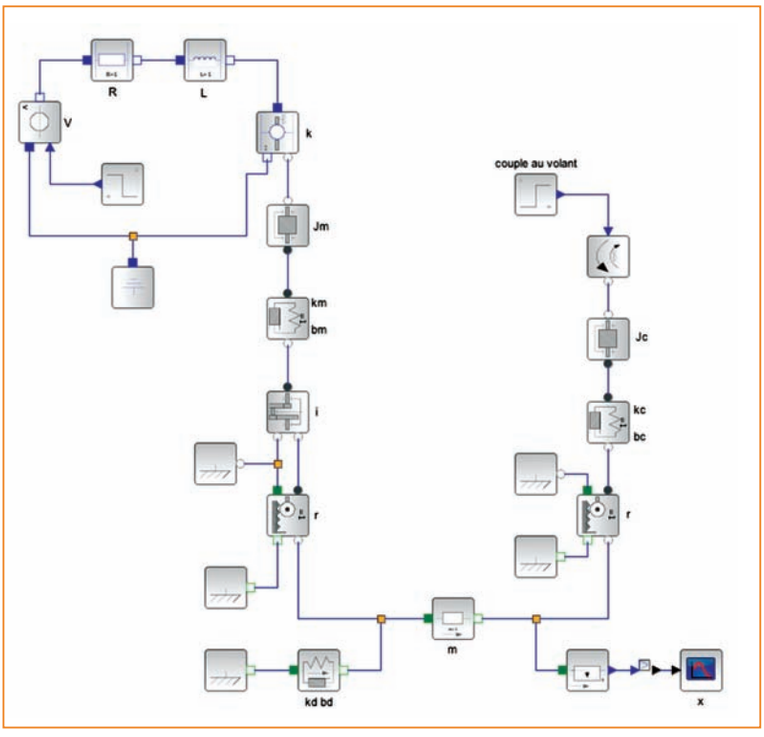
Domaine	Potentielle	Traversante
Mécanique translation	Vitesse	Force
Mécanique rotation	Vitesse angulaire	Couple
Électrique	Tension	Intensité
Hydraulique	Pression	Débit volumique
Thermique	Température	Flux d'entropie

15 Les variables potentielles et traversantes (Simscape)

Domaine	Potentielle	Flux
Mécanique translation	Position	Force
Mécanique rotation	Angle	Couple
Électrique	Tension	Intensité
Hydraulique	Pression	Débit volumique
Thermique	Température	Flux d'entropie

16 Les variables potentielles et flux (Modelica)

l'interface entre constituants du système. Cette notion de variables duales est fondée sur la notion de dualité au sens mathématique du terme. Rappelons qu'en algèbre linéaire un espace vectoriel dual F d'un espace vectoriel E est l'espace vectoriel des formes linéaires sur E, une forme linéaire sur E étant une application linéaire de E sur le corps à partir duquel est défini E (pour nous, il s'agira de l'ensemble des réels). Pour ce qui nous concerne, sans entrer dans les détails et en simplifiant à l'extrême, on peut définir un espace vectoriel de grandeurs notées *flux*. L'espace vectoriel dual de celui-ci est l'ensemble des grandeurs que l'on peut appeler efforts telles que leur produit par un flux soit un nombre réel (la puissance, dans de nombreux cas). On voit ainsi l'arbitraire dans le choix des variables flux et donc des variables effort. Suivant la théorie ou le logiciel utilisé, des couples différents sont utilisés. Nous l'avons vu, en langage bond graph, le couple utilisé est celui de variables effort et flux telles que



17 Le modèle acausal de la DAE réalisé avec Scilab/Xcos/Coselica

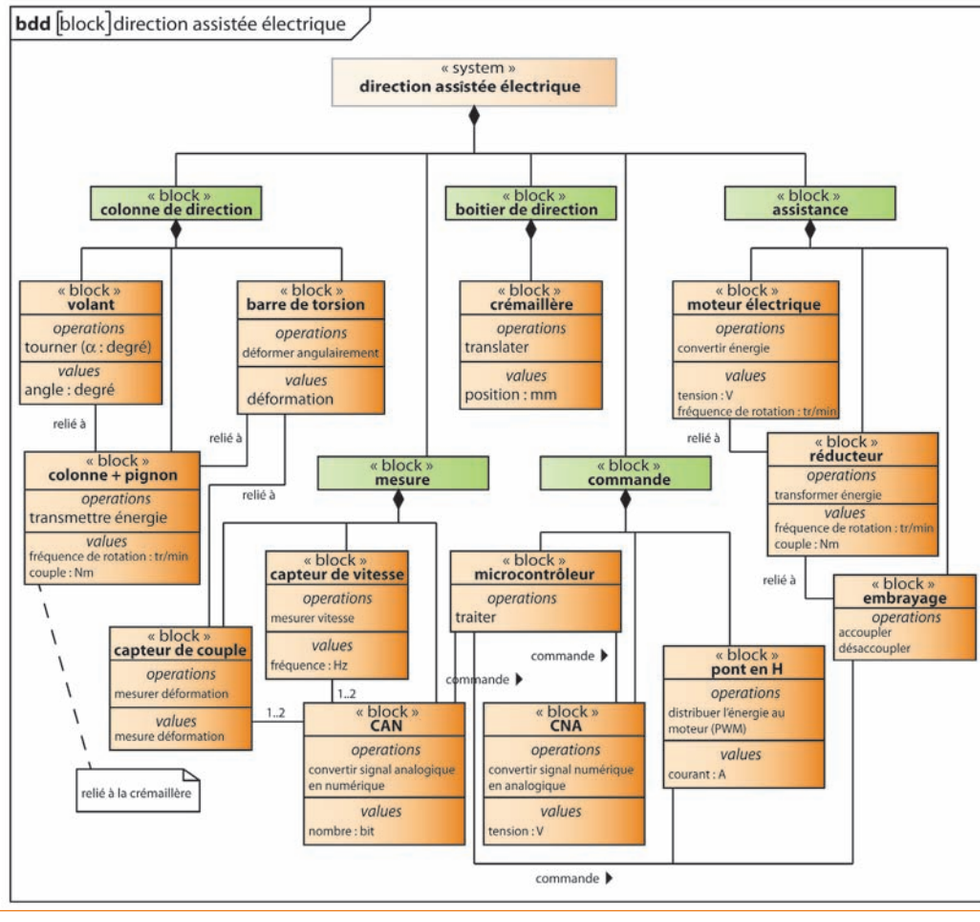
leur produit est une puissance. La suite Matlab/Simulink/Simscape utilise le concept de variables potentielle (*across*) et traversante (*through*) 15. Elles se définissent ainsi :

- Une variable potentielle est mesurée par un instrument en parallèle avec la chaîne d'énergie.
- Une variable traversante est mesurée par un instrument en série avec la chaîne d'énergie.

Quant au langage Modelica, il note son couple de variables conjuguées *potentielle* et *flux* 16 et les définit ainsi :

- Variables potentielles, les variables qui sont reliées au même port sont égales.
- Variables flux, les variables qui aboutissent au même port ont pour somme 0.

Ces distinctions peuvent créer une certaine confusion dans la modélisation des systèmes. Force est de constater que le produit des variables effort et flux du langage bond graph comme celui des variables potentielle et traversante utilisées par Simscape sont égaux à une puissance. Modelica utilise les déplacements au lieu des vitesses en mécanique, sans doute car il les considère comme plus adaptés à la modélisation de ce domaine physique.



18 Le diagramme de définition de blocs de la DAE

SysML et Modelica : des langages compatibles ?

Le langage de modélisation SysML (*System Modeling Language*) introduit neuf diagrammes répartis en diagrammes structurels, de comportement et d'exigences afin de modéliser les systèmes. Il s'agit d'un outil puissant pour décrire une abstraction des systèmes. Il est possible pour certains diagrammes et grâce à des logiciels dédiés de simuler le comportement du modèle. Il peut donc être utile dans le cadre de l'enseignement en CPGE d'utiliser ces logiciels.

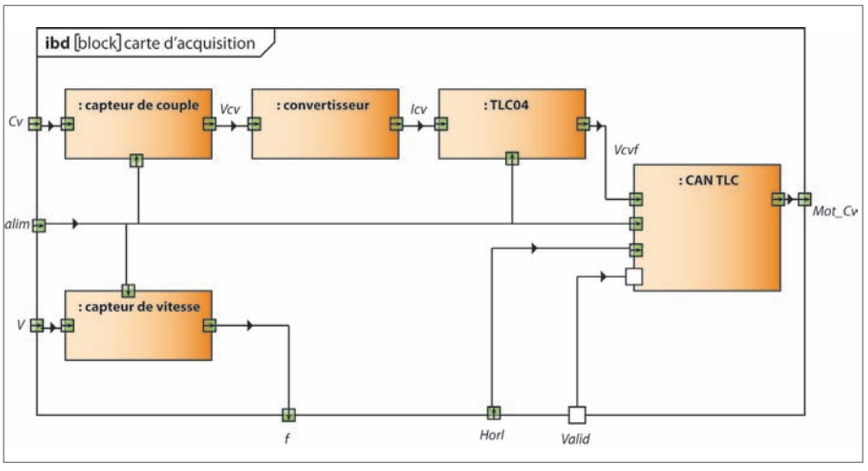
SysML est un profil UML (*Unified Modeling Language*), c'est-à-dire un ensemble de stéréotypes (processus d'extension du langage pour créer des éléments nouveaux) et de contraintes qui vient compléter la définition standard d'UML (langage utilisé dans le domaine informatique). Un profil UML réalise ainsi le rapprochement et la spécialisation d'UML à un domaine particulier. SysML bénéficie alors des mécanismes d'extension d'UML que sont les stéréotypes, les métapropriétés (*tagged values*) et les contraintes.

Comme nous l'avons vu, Modelica est un langage de programmation textuel enrichi par des bibliothèques graphiques pratiques pour une simplification de la modélisation des systèmes. Cette approche graphique illustre clairement le lien entre les deux langages SysML et Modelica. L'un et l'autre proposent une décomposition d'un système en sous-systèmes, constituants et composants, comme le montrent respectivement les figures 17 et 18. Ces éléments sont reliés par des connections. Dans le cas de Modelica, les connecteurs sont de deux types (acausal pour l'énergie et la matière, causal pour l'information). Ils permettent donc la circulation de quantités de types matière, énergie et information. Quand on connecte une sortie d'information d'un composant à une entrée d'information d'un autre composant, cette dernière se voit imposée la valeur de la sortie du premier bloc. Pour les connecteurs d'énergie et de matière, les connections sont régies par les lois de Kirchhoff, comme déjà mentionné plus haut : la somme des valeurs des variables flux (au sens de Modelica) qui aboutissent en un même point est nulle, et les variables qui ne sont pas des flux sont égales. Pour la suite, nous nous référerons à des diagrammes de SysML dont on trouvera une étude poussée dans de nombreuses publications anglo-saxonnes (je recommande notamment *SysML for System Engineering* de Holt et Perry [5]). L'ouvrage en français *SysML par l'exemple* de Pascal Roques [6] est une bonne introduction.

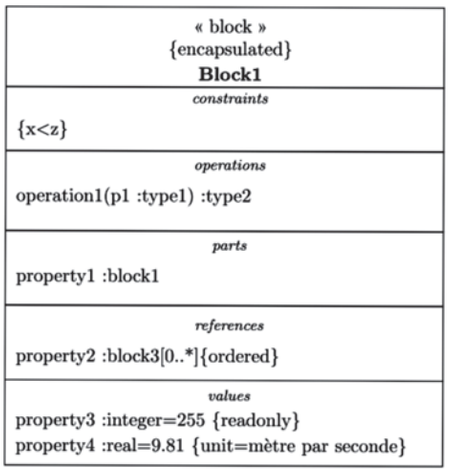
Trois diagrammes du langage SysML ont une structure similaire (blocs hiérarchiques reliés par des connecteurs) à celui de Modelica : le diagramme de blocs internes, le diagramme paramétrique et le diagramme d'activité. Le langage SysML utilise le terme de *port* pour traduire la connexion entre éléments. Notons également que le terme de *block* utilisé par SysML est une extension essentielle d'UML en tant que stéréotype de la classe de ce dernier. Il s'agit d'une unité modulaire qui peut contenir des éléments structurels et comportementaux. Les blocs sont définis à l'aide du diagramme de définition de blocs, qui permet de décrire des relations de compositions, des agrégations, des dépendances, des généralisations de blocs.

Le diagramme de blocs internes

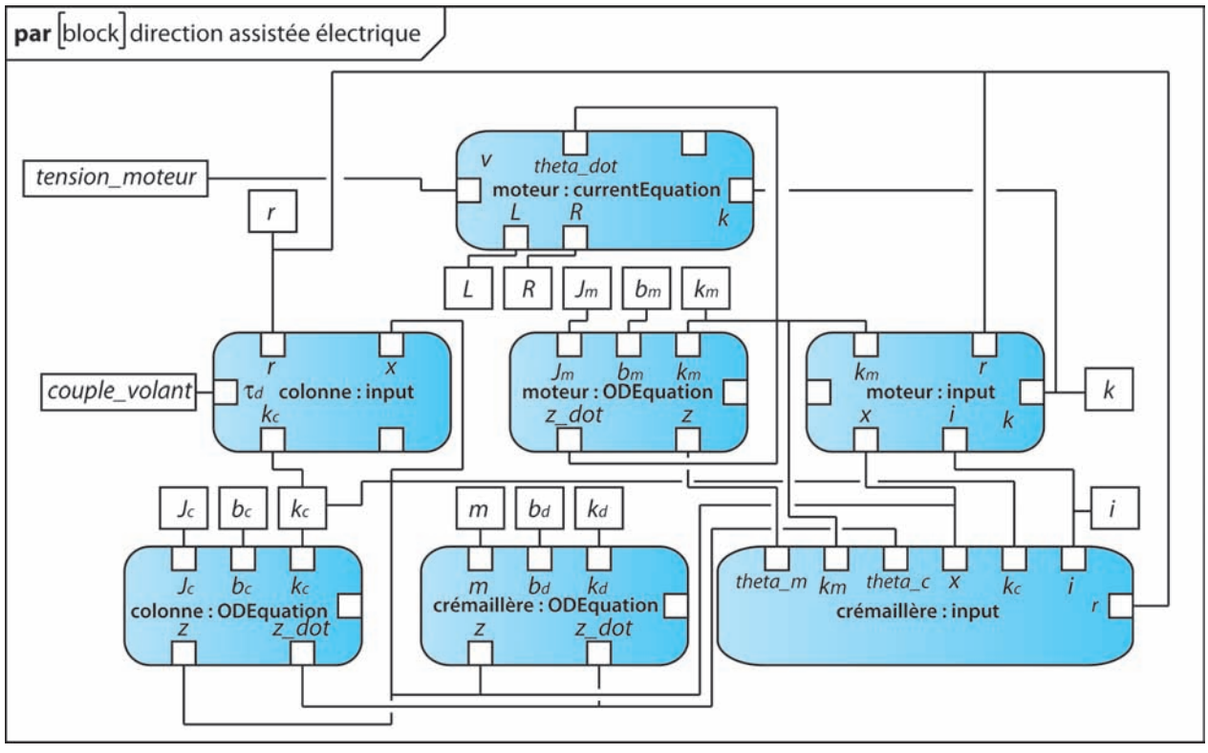
En lien direct avec le diagramme de définition de blocs, le diagramme de blocs internes représente la structure interne des blocs et les connexions entre les éléments du système étudié à travers des « ports » 19. Les blocs de SysML sont identiques aux classes du langage UML, que l'on retrouve également dans Modelica. On note deux types de ports dans un diagramme de blocs internes, à savoir les *flow ports* et les *standard ports*. Ces derniers représentent un point de communication lié à un service entre les blocs (état d'un capteur, interruption par logiciel, par exemple) et n'ont pas de correspondance



19 **Le diagramme de blocs internes de la carte d'acquisition de la DAE**

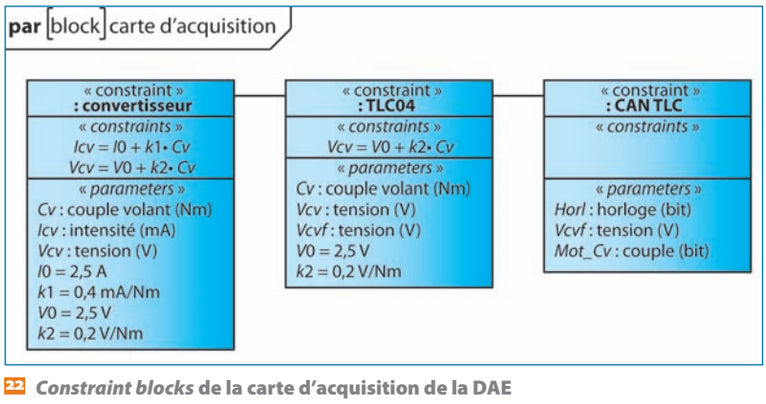


20 **Stéréotype de la classe *block***



21 **Le diagramme paramétrique de la carte d'acquisition de la DAE**

dans Modelica. Les *flow ports* décrivent un transport d'énergie, de matière ou d'information entre les blocs et sont plus proches de la sémantique de Modelica, malgré quelques nuances. On peut dire qu'un *flow port* atomique (au sens de SysML) correspond à un connecteur transportant un signal (au sens de Modelica) et qu'un *flow port* non atomique correspond à un connecteur transportant des quantités de types matière et énergie représentées par deux variables (flux et non-flux) à l'instar d'un lien d'énergie au sens des bond graphs. Notons également que les connexions acausales entre des *flow ports* (SysML) ne suivent pas explicitement les lois de Kirchhoff pour l'énergie et la matière mentionnées pour Modelica. Même si l'approche est similaire, il faut rester vigilant quant à une analogie complète entre le langage Modelica et le diagramme de blocs internes. Il sera donc nécessaire de créer des stéréotypes dans SysML pour préciser la sémantique de Modelica 20.

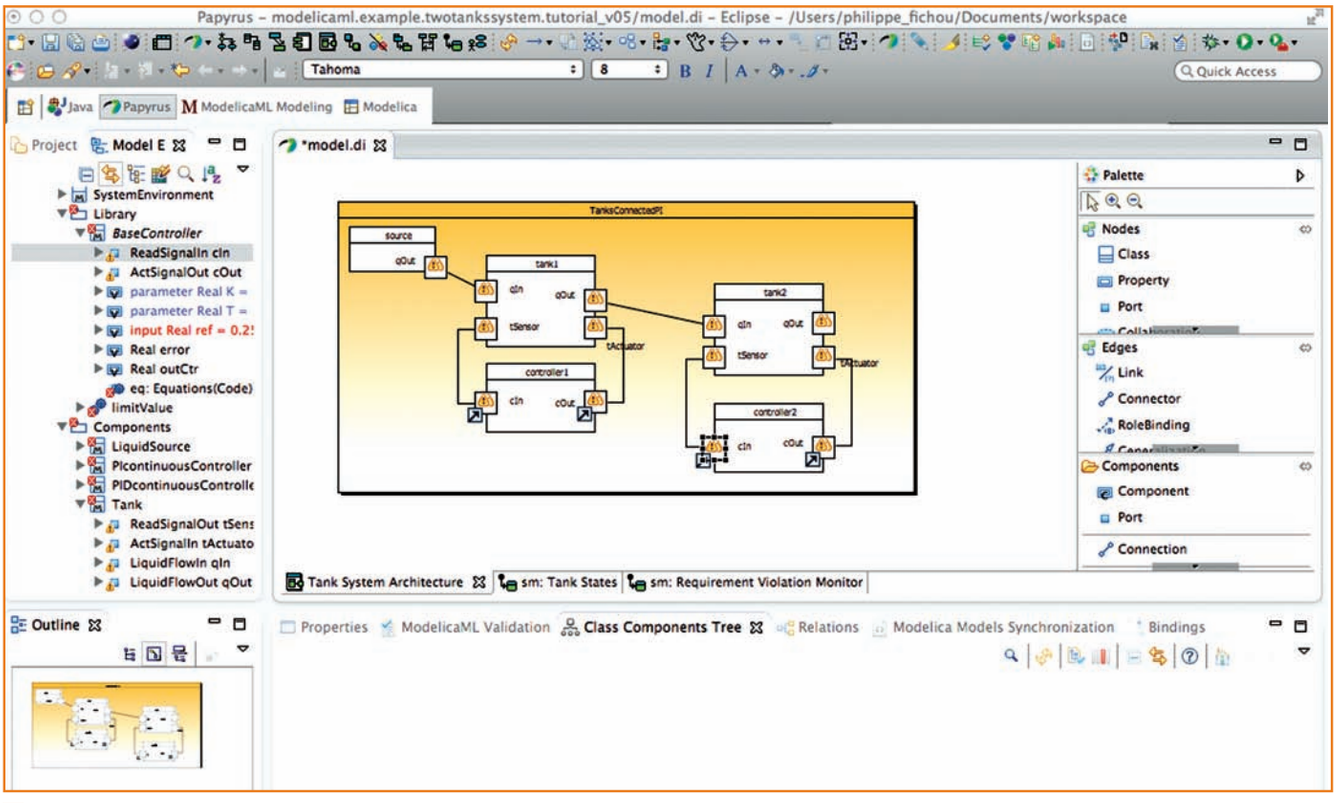


22 ***Constraint blocks* de la carte d'acquisition de la DAE**

Le diagramme paramétrique

Le diagramme paramétrique modélise les relations mathématiques entre des paramètres du modèle 21. Dans ce diagramme, un bloc sera stéréotypé *constraint block* 22, les ports représentent des *constraint parameters* et les connections des *binding connectors* dont la sémantique indique une égalité mathématique (acausale) entre les paramètres contraints connectés. Les *constraint blocks* définis dans le diagramme de définition de blocs et utilisés dans le diagramme paramétrique indiquent les relations mathématiques entre les paramètres contraints.

Les *binding connectors* n'ont pas leurs équivalents directs en langage Modelica, mais il est possible d'écrire une égalité en langage textuel entre deux variables reliées. À l'inverse, les connecteurs du diagramme paramétrique ne traduisent pas ceux de Modelica (variables acausale de flux et potentielle d'énergie et de matière, et variables causales d'information). Plusieurs possibilités dans SysML pourraient voir le jour pour traduire la sémantique de Modelica (introduction d'un nouveau connecteur



23 **L'environnement Papyrus dans Eclipse intégrant Modelica et ModelicaML**

dans SysML basé sur les lois de Kirchhoff, ou explicitation de celles-ci dans des *constraint properties* qui rendraient sans doute le diagramme plus difficile à lire et donc à exploiter). Là encore, on peut noter certes une proximité entre les langages, mais également des différences qui ne permettent pas d'analogie complète.

Le diagramme d'activité

Un diagramme d'activité précise la séquence d'actions à réaliser ainsi que ce qui est produit, consommé ou transformé lors de l'exécution de l'activité. Le comportement est basé sur le transfert d'un « jeton » entre actions pour les démarrer ou les terminer. Dans un diagramme d'activité, il existe deux types de flux, le flux de contrôle entre actions et le flux d'objets. Ce dernier correspond à des transferts d'objets à des instants précis, bien qu'il soit possible d'envisager des flux continus qui soient décrits par des équations différentielles, qui rapprocheraient la sémantique des diagrammes d'activité de celle de Modelica. Mais, hormis cette dernière situation, un rapprochement direct de ce diagramme avec Modelica n'est pas envisageable.

Ainsi, il apparaît quelques similitudes entre les deux langages SysML et Modelica, mais il n'est pas anormal que les modélisations qu'ils permettent, n'ayant ni le même objectif ni les mêmes origines, ne se recoupent que partiellement. Un passage possible de l'un à l'autre est envisageable, notamment grâce à leur souplesse d'expression (celle de SysML plus particulièrement, car il permet des profils et des stéréotypes). L'ingénierie système, basée sur les modèles, est la démarche vers laquelle la création des produits industriels s'oriente aujourd'hui. Elle ne peut être ignorée dans l'enseignement scolaire et universitaire. C'est la raison pour laquelle, en STI2D et en sciences industrielles de l'ingénieur en CPGE, les programmes introduisent progressivement les outils présentés ici, qui seront sans aucun doute, quand nos élèves seront formés, au cœur des métiers de l'ingénierie – et le sont d'ailleurs déjà dans beaucoup d'entreprises.

ModelicaML : un profil d'UML, la solution intégrée ?

De nombreux travaux de recherche sont en cours pour une intégration des langages de description des systèmes (SysML, par exemple) et ceux qui permettent la simulation de leur comportement (Modelica, par exemple). ModelicaML, créé par Peter Fritzson [7] et Adrian Pop, en cours de développement, est un langage de modélisation graphique, profil d'UML, pour décrire l'architecture et le comportement dynamique des systèmes (simulation des comportements continus). Il permet notamment de générer du code Modelica à partir d'un langage graphique décrivant, par exemple, le digramme d'état ou le

diagramme d'activité de SysML. ModelicaML permet ainsi d'étendre le langage Modelica à certains diagrammes de SysML. ModelicaML est un langage *open source* écrit en Java, ce qui autorise une portabilité sur tous les systèmes d'exploitation. Chacun peut donc le télécharger et l'utiliser à partir de la plate-forme Java Eclipse proposant un environnement de développement intégré libre, extensible, universel et polyvalent. Il s'agit en fait d'un « atelier » contenant une panoplie d'objets qui permettent de résoudre un ensemble de problèmes reliés. Le *framework* Eclipse inclut déjà les logiciels Papyrus [23] et TopCased pour une description des systèmes en langage SysML (d'autres *frameworks* comme .NET ou NetBeans proposent ce même type de regroupement). Nous avons ainsi à notre disposition un atelier de conception virtuelle et de simulation intégrées. ModelicaML y trouve naturellement sa place, puisqu'il offre un outil de simulation efficace tout en conservant la continuité de la chaîne numérique générée lors d'une conception écrite en langage SysML (qui est, répétons-le, un langage et non une méthode).

Nous n'avons pas mentionné de logiciels payants qui intègrent plusieurs outils dans l'esprit de ceux présentés ; le lecteur saura faire le lien entre ce qu'il utilise (ou utilisera) et les produits libres rapidement évoqués ici.

Une conception de plus en plus abstraite

La compétitivité des produits est au cœur de la réussite des entreprises et des pays dans nos sociétés ouvertes. Les outils de développement évoluent très vite pour permettre une réactivité optimale. Les sciences de l'ingénierie ont un rôle majeur dans cet environnement : nous sommes passés progressivement d'une conception artisanale à une conception empirique pour arriver aujourd'hui à une conception abstraite. Une telle abstraction semble être un prérequis pour maîtriser le développement de solutions techniques complexes, c'est-à-dire répondant à des exigences nombreuses et contradictoires, et ce, en synthétisant de façon non triviale des composants nombreux, variés, hétérogènes, voire incompatibles. L'ingénierie système fondée sur les modèles pousse le plus loin possible l'hypothèse centrale de la conception abstraite selon laquelle il convient de développer la solution concrète au plus tard et d'utiliser des modèles à toutes les étapes de la conception. Les langages évoqués ici sont au cœur de ce mode de pensée et d'action. Il est donc essentiel de les partager au plus tôt avec les futurs ingénieurs et techniciens, afin qu'ils s'y familiarisent au plus vite, d'autant qu'il s'agit d'outils qu'en tant que *digital natives* ils n'auront pas forcément de difficultés à maîtriser. La modélisation multiphysique, rapidement évoquée ici, paraît donc être pertinente pour une appréhension des systèmes réels ou didactisés et des modèles associés et pour permettre d'utiliser des outils de conception et d'analyse actualisés et en prise direct avec le monde industriel. L'analyse des écarts revendiquée dans les programmes de 2013 de CPGE est parfaitement en phase avec les évolutions de l'ingénierie système. Les élèves des classes préparatoires devraient sans aucun doute pouvoir tirer tout le bénéfice de cette approche dans leur formation d'ingénieur, qui débute dès la sortie du lycée. Un nouveau défi pour les enseignants de sciences industrielles de l'ingénieur en CPGE. ■

Références et bibliographie

[1] FICHOU (Ph.), « Bond graphs : une méthode pluridisciplinaire », in *Technologie,* n° 133, sept.-oct. 2004

[2] FICHOU (Ph.), « Modéliser les systèmes par langage graphique : Éléments de bond graph », in *Technologies et formations,* n° 128, nov.-déc. 2006

[3] DAUPHIN-TANGUY (G.) (sous la dir. de), *Les Bond Graphs,* Hermes science / Lavoisier, 2000

[4] TURKI (S.), *Ingénierie système guidée par les modèles : Application du standard IEEE 15288, de l'architecture MDA et du langage SysML à la conception des systèmes mécatroniques,* thèse soutenue le 2 octobre 2008, téléchargeable sur

http://tel.archives-ouvertes.fr

[5] HOLT (J.), PERRY (S.), *SysML for System Engineering,* The Institution of Engineering and Technology, 2008, téléchargeable (taper le titre dans le champ de recherche de son navigateur).

[6] ROQUES (P.), *SysML par l'exemple : Un langage de modélisation pour systèmes complexes,* Eyrolles, 2009, disponible uniquement en version PDF (payante) à télécharger sur

www.eyrolles.com

[7] FRITZSON (P.), *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1,* John Wiley & Sons, 2004

AÏT-EL-HADJ (S.), BOLY (V.) (sous la dir. de), *Les systèmes techniques : Lois d'évolution et méthodologies de conception,* Hermes science / Lavoisier, 2009

BERGMANN (C.), « L'automatique, une approche moderne », in *Technologies et formations,* n^{os} 127 et 128, sept.-oct. et nov.-déc. 2006

CHAMBADAL (L.) OVAERT (J.-L.), *Algèbre linéaire et algèbre tensorielle,* Dunod, 1968

DAGUE (Ph.), TRAVÉ-MASSUYÈS (L.), « Raisonnement causal en physique qualitative »,in *Intellectica,* n° 38, 2004

RENIER (R.), CHENOUARD (R.), « De SysML à Modelica : Aide à la formalisation de modèles de simulation en conception préliminaire », actes du 12^{e} Colloque national AIP Primeca, 2011, téléchargeable sur

http://hal.archives-ouvertes.fr