5

Modelica, un langage pour la modélisation

Modelica est un langage orienté objet pour la modélisation de systèmes physiques complexes et hétérogènes. Le langage Modelica est une description textuelle permettant de définir toutes les parties d'un modèle et de structurer les modèles de composants dans des bibliothèques appelés *packages*. Le langage Modelica se fonde principalement sur la définition d'ensembles d'équations reliant les variables plutôt que par des séquences d'affectations.

Un modèle Modelica est constitué de composants tels qu'une résistance ou un vérin hydraulique. Ces composants sont connectés par l'intermédiaire de connecteurs (souvent aussi appelés *ports*). Ces connecteurs décrivent les possibilités d'interaction, par exemple une broche électrique, un bride mécanique ou un signal d'entrée.

Dans ce chapitre, nous présentons les traits essentiels de ce langage pour vous aider à mieux comprendre le fonctionnement de Xcos avec les composants Modelica et vous permettre de créer des composants Modelica simples. Pour une description complète du langage Modelica, vous pouvez vous reporter au tutoriel édité par l'Association Modelica ou à son document de spécification (en anglais).

Astuce *› Vous pouvez trouver des exemples de modèles de composants électriques ou hydrauliques dans les fichiers* `Electrical.mo` *ou* `Hydraulics.mo` *que l'on peut trouver dans les répertoires désignés par la fonction* `getModelicaPath`*. Ces fichiers peuvent s'ouvrir avec n'importe quel éditeur de texte. Vous pourrez y reconnaître les différents éléments du langage présentés dans ce chapitre.*

5.1. Les classes Modelica

La représentation et la description des systèmes dans un langage tel que Modelica repose sur une double analyse : de *quoi* est fait l'objet ou le système et *comment* fonctionne-t-il ? Quelques exemples vont vous permettre de vous familiariser avec ce mode de raisonnement.

La première question *de quoi est fait le système ?* sert à décrire le système ou l'objet, à le représenter sous forme de table(s), de graphique(s), d'icône(s), à lister ses connecteurs,

ses variables internes ou publiques, et pour chacun de ces éléments leur signification, leur sémantique. Par exemple :

- pour une inductance : c'est l'objet même avec ses dimensions physiques : section cuivre, nombre de tours d'enroulement, la position des organes de connexion, dimensions du noyau ferromagnétique, etc. ;
- pour une masse mécanique d'inertie : caractéristiques mécaniques, position du centre de gravité, du centre d'inertie, sa masse, etc. ;
- pour une voiture : masse, puissance, type de transmission, paramètres techniques, etc.

La seconde question *comment se comporte le système ?* décrit ou renseigne sur ses propriétés comportementales. Pour les mêmes exemples précédents, le *comment* sera :

- une inductance : la loi $v = L * (d\,i / d\,t)$;
- une masse : $f = M * \gamma$;
- une voiture : comportements dynamiques en ligne droite, en courbe, en fonction de l'altitude, etc.

Les deux parties de la description d'un objet forment un tout indissociable car des objets différents peuvent avoir en commun une des deux parties, sans pour autant être semblables ou interchangeables.

Le tout est encapsulé dans une entité conceptuelle la *classe*, notée `class`, qui est l'unité fondamentale de modélisation dans Modelica.

Tous les objets sont instanciés à partir de classes, y compris les types de données de base : réel, entier, chaîne, booléen, etc.

À partir de la définition d'une classe, on obtient par filiation-descendance des *objets* encore appelés *instances*. La classe sert alors de modèle de référence pour les *objets* qui en *descendent*.

Pour Modelica "tout est classe", autrement dit tout système technologique est descriptible, tant du point de vue structural (le *quoi*) que du point de vue comportemental (le *comment*), par des instances de classes préalablement et judicieusement construites.

La définition d'une classe est organisée en plusieurs sections contenues entre la déclaration du type, du nom et des attributs de la classe et le mot clé `end`. Une classe doit être composée au moins d'une des sections suivantes :

- déclarations des variables, des constantes ou tableaux (éléments). Les déclarations sont les constructions syntaxiques nécessaires pour introduire les classes et les objets. La section des déclarations des variables suit la déclaration du nom et des attributs de la classe ;
- définition du comportement par des équations précédée du mot clé `equation` ;
- définition du comportement par des algorithmes précédée du mot clé `algorithm`.

Exemple 5.1 : *Exemple de structure d'une classe*

```
class Voiture
   Déclaration1
   Déclaration2
   ...
equation
   Equation1
   Equation2
   ...
end Voiture ;
```

Une fois une classe définie, il est possible de l'instancier par déclaration de variable au sein de la définition d'une autre classe :

```
Voiture vehiculUn ;   // Première instanciation de la class Voiture
Voiture vehiculDeux ; // Seconde instanciation
Voiture vehiculTrois ; // Troisième instanciation
```

Les `//` marquent le début d'un commentaire (voir section Commentaires).

Les déclarations ci-dessus créent trois instances dérivées de la classe `Voiture`. Ainsi `VehiculTrois` est une variable dont l'architecture est définie par `la classe Voiture`. Lors de l'instanciation, il y a création de l'allocation en mémoire nécessaire pour cet objet. Toutes les instances `vehiculXxx` sont indépendantes les unes des autres.

Les déclarations

La section des déclarations intervient immédiatement après la définition du nom de la classe. La déclaration d'un élément se présente sous la forme :

```
component_clause:
  type_prefix type_specifier [ array_subscripts ] component_list
type_prefix :
  [ flow | stream ]
  [ discrete | parameter | constant ] [ input | output ]
type_specifier :
```

```
  name
component_list :
  component_declaration { "," component_declaration }
    component_declaration :
      declaration [ conditional_attribute ] comment
        declaration :
          IDENT [array_subscripts] [modification]
        conditional_attribute:
          if expression
```

Exemples de déclarations :

```
constant Real Pi=3.1415927; ❶
parameter Real Mass =2.5; ❷
Real y (start = 1.0);  //Déclaration d'une variable réelle. ❸
```

❶ Déclare `Pi` comme une constante réelle et lui assigne une valeur.

Note *› Nommer les constantes présente de nombreux avantages dont le premier est de n'avoir qu'un seul point de définition de leur valeur ce qui facilite grandement les opérations de modification ou de maintenance d'un programme. Accessoirement, le nom, s'il est bien choisi, aide à la compréhension du code.*

❷ Définit le paramètre réel `Mass` et lui assigne une valeur.

Note *› Un paramètre garde une valeur constante au cours d'une simulation.*

❸ La définition de `y` spécifie une variable réelle dont la valeur au début de la simulation (typiquement à l'instant t=0) est indiquée par `(start = 1.0)`.

Nous allons préciser dans les sections suivantes les principaux éléments pouvant intervenir dans les déclarations.

Types des éléments

La spécification du type de l'élément (`type_specifier`) est fournie par le nom du type (`name`). Modelica propose les types de base classiques : `Real`, `Integer`, `String`, `Boolean` ou `Enumeration`.

`Real`	Nombre flottant sur 64 bits au minimum, suivant la norme IEEE 754.
`Integer`	Entier sur 32 bits.
`String`	Chaîne de caractères. Les caractères sont codés sur 8 bits.
`Boolean`	True ou False, Vrai ou faux.