Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

Object Modelling of Interconnected Systems in a Behavioral Framework

T. Bastogne
Centre de Recherche en Automatique de Nancy (CRAN),
CNRS UMR 7039, Université Henri Poincaré, Nancy 1,
BP 239, F-54506 Vandœuvre-ls-Nancy Cedex, France,
Phone: (33) 3 83 68 44 73 - Fax: (33) 3 83 68 44 62
thierry.bastogne@cran.uhp-nancy.fr

Abstract— **The problem addressed in this communication deals with the object-modelling of interconnected systems. This communication attempts to provide a bridge between the object-oriented modelling methods based on languages and compilers for simulation purpose, and the behavioral modelling approach based on a reliable mathematical formalism. An object-oriented model structure (multiport diagram) and a modelling procedure are developed. An application to a simple example is presented to illustrate the implementation of the multiport diagram into the modelling language Modelica©.**

I. INTRODUCTION

Modelling and simulation of complex systems are important topics which are common to all fields of engineering and science. The problem addressed in this communication deals with interconnected systems modelling, i.e. physical systems composed of interconnected (energy-conserving) sub-systems. Such systems are widespread in industry, e.g. batch processes in chemical industries [1], drive-train processes in iron and steel industries or power plants in energy industries [2].

The applications of object-modelling techniques in control and automation mainly concern the development of 'High-fidelity' dynamic models, i.e. models for simulation, training, safety certification or reconfigurable control purposes [3]. Since the end of Seventies, efforts have converged to develop object-oriented languages for physical systems modelling : *Dymola* [4], χ [5], or *Modelica* [6]. Some formalisms like the Bond graphs can be allied with the class of object-oriented models [7]. However, it was necessary to await the end of the Eighties to get efficient platforms of simulation able to deal with differential algebraic equations systems with high index [8]. All these advances have provided languages and compilers for a consistent simulation of complex systems. However, one can wonder about the interest in having so many languages and compilers. Moreover, their frequent evolutions can become a serious drawback for the capitalization of the modelling efforts. Hence it would be interesting for the modelling procedure to be initially independent on any programming language. The behavioral formalism of systems theory proposed by Willems in 1986 [9] provides a mathematical paradigm for interconnected systems modelling and could be an answer to this problem.

Reference	Description
$\mathbb{D}$	set, space
$\mathbf{D}$	object-class
$\mathcal{D}$	instance

TABLE I
BASIC NOTATIONS

The first goal of this communication is then to examine the suitability of the behavioral formalism for the description of the object-oriented paradigm in order to provide a bridge between them. A second objective is to develop an object-oriented model structure (multiport diagram) and a modelling procedure that would initially be independent from any existing modelling language. The last objective is to show that the implementation effort of the multiport diagram into a modelling language like Modelica is small.

II. MAIN CONCEPTS OF OBJECT-MODELLING TECHNIQUES

In this section, the behavioral formalism [10] is borrowed to describe the main concepts of object-orientation. Table I summarizes some basic notations used herein.

A. Concept of object

One main specification of the object-modelling paradigm is to gather data and data processings in the same autonomous structure called *object*. In the behavioral framework, this process of encapsulation of data and behavior can be defined by Eq. (1).

$$\mathbf{O} = (\mathbb{U}_{\mathbf{O}}, B_{\mathbf{O}}, P_{\mathbf{O}}) \quad (1)$$

where $\mathbf{O}$ denotes the object, $\mathbb{U}_{\mathbf{O}}$: the data *universum*,$B_{\mathbf{O}}$: its behavior and $P_{\mathbf{O}}$ contains its communication ports or interfaces by which it communicates with its environment. A major difference between the objects used at the origin in programming languages and those used for physical systems modelling is that contrary to conventional objects, physical models are associated to a temporal semantics. Consequently, the data universum can be defined by:

$$\mathbb{U}_{\mathbf{O}} = \mathbb{T} \times \Theta \times \mathbb{W} \quad (2)$$

where $\mathbb{T}$ denotes the time axis, Θ: the parameter space, $\mathbb{W}$: the variable space and $\times$ the cartesian product. The behavior of an object can then be expressed in the form of behavioral equations:

$$B_{\mathbf{O}} = (t \in \mathbb{T}, \theta \in \Theta, w(t) \in \mathbb{W} | (4)) \quad (3)$$
$$f_1(t,p,w) = f_2(t,p,w) \quad (4)$$

where t denote the time variable, θ: the vector of parameters and $w(t)$: that of variables. But the principal utility of the encapsulation remains the privatization of the access to the data. Indeed, the concept of object also makes it possible to legalize and hold the access of a limited number of variables (known as external). These variables enable it to communicate with outside via interfaces or ports of communication. We will model the external variables of an object by *manifest* variables and the internal or local variables by *latent* variables, by respecting the terminology suggested by Willems. The complete behavior of the object is then defined by:

$$B_{\mathbf{O}} = (t \in \mathbb{T}, \theta \in \Theta, w(t) \in \mathbb{W} | \exists l \in \mathbb{L}, (6)) \quad (5)$$
$$f_1(t,\theta,w,l) = f_2(t,\theta,w,l)), \quad (6)$$

where: $w(t)$ and $l(t)$ correspond to the vectors of manifest and latent variables respectively. $\mathbb{W}$ is the manifest signal space and $\mathbb{L}$ the latent variable space.

B. Concepts of class and instance

Objects are organized in *classes*. A class is a paradigm defining the behavior and the variables for a particular type of object. Any object designed from this paradigm is an *instance* of this class. Instances are the physical representations of objects in the model. The class-instance relationship is symbolized by $\Rightarrow$. For example, $\mathbf{A} \Rightarrow \mathcal{A}_1, \mathcal{A}_2$ means that $\mathcal{A}_1$ and $\mathcal{A}_2$ are two instances of $\mathbf{A}$. As shown in Eq. (7), the instances and class are identical by their form and their behavior, but their variables generally contain different values.

$$B_{\mathcal{A}1} = B_{\mathcal{A}2} = B_{\mathbf{A}}. \quad (7)$$

C. Concept of inheritance

The various types of classes result from a stage of system structuring. They are organized in a specialization-generalization hierarchy. The latter naturally induces super-class/subclass associations among classes. The behavior of superclasses are inherited by their subclasses. Given two object-classes: $\mathbf{A}$ and $\mathbf{B}$, if $\mathbf{A}$ is a super-class of $\mathbf{B}$, noted $\mathbf{B}/\mathbf{A}$, then the complete behavior of $\mathbf{B}/\mathbf{A}$ is defined by :

$$B_{\mathbf{B}/\mathbf{A}} = B_{\mathbf{B}} \cap B_{\mathbf{A}}. \quad (8)$$

where $B_{\mathbf{B}}$ represents the specific behavior of the free-class $\mathbf{B}$. The ports of $\mathbf{B}/\mathbf{A}$ are given by :

$$P_{\mathbf{B}/\mathbf{A}} = \{P_{\mathbf{B}}, P_{\mathbf{A}}\}. \quad (9)$$

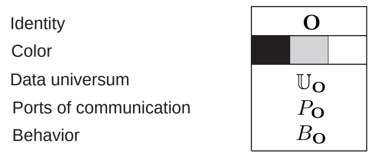


Fig. 1. General description of a module

But the inheritance process also implies that each class of a hierarchy inherits the data of its superclasses, which implies that :

$$\mathbb{U}_{\mathbf{A}} \subset \mathbb{U}_{\mathbf{B}/\mathbf{A}}. \quad (10)$$

III. A Multiport Diagram for interconnected systems

Object-oriented models of interconnected systems are generally based on a graphic structure. The one proposed herein is entitled *multiport diagram*. This section aims at describing the composition of a multiport diagram, noted Δ, in the behavioral framework. As shown in Eq. (11), such a diagram is composed of two object-classes: modules and links.

$$\Delta = (O, L) \quad (11)$$

$O = \{\mathcal{O}_1, \cdots, \mathcal{O}_m\}$ is a set of module instances which compose the diagram and $L = \{\mathcal{L}_1, \cdots, \mathcal{L}_n\}$ is a set of link instances which allow the modules to exchange energy and information with other modules. Each module instance of O is associated with a component of the system to model, and L describes the interconnection architecture of the diagram. m and n are the number of modules and links in Δ.

A. Module class

As shown in Fig. 1, the general description of a module object class: $\mathbf{O}$ is based on five attributes. Its **identity** is composed of its name relative to the function of the component and can be completed by an icon which graphically represents the object. Its **behavioral model** : $B_{\mathbf{O}}$ is defined by the Eq. (6) where $f_1(\cdot)$, $f_2(\cdot)$ express the behavioral equations of the object. Other formalisms such as transfer functions, block diagrams, bond graphs, Petri nets, etc. can used to describe the behavior of the object. θ is a vector of parameters and $(w(t), l(t))$ are the manifest and latent variables of the model. Its data **universum** is defined by $\mathbb{U}_{\mathbf{O}} = (\mathbb{T} \times \Theta \times \mathbb{W} \times \mathbb{L})$. Its **communication ports** or interfaces by which it communicates with its environment, $P_{\mathbf{O}}$ is a set of port instances, defined by :

$$P_{\mathbf{O}} = \{\mathcal{P}_{\mathbf{O}.k}\} \quad (12)$$

where $\mathcal{P}_{\mathbf{O}.k}$ is the k^{th} port instance of $\mathbf{O}$. Its **colour** defined, by analogy with the system identification terminology, according to the *a priori* knowledge about the object, i.e. : *white box* if the theoretical laws or physical equations and the values of parameters are known, *grey box* if there only exists a partial knowledge about the object, i.e. values of

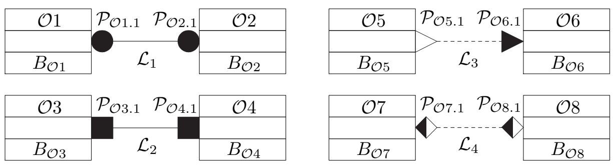


Fig. 2. Types of links and ports

some parameters or the mathematical structure of physical equations or *black box* if no *a priori* knowledge about the object is available.

B. Port class

A port is a terminal of communication attached to an object. Two main classes of ports are considered: the **power** ports and the **information** ports which allow objects to exchange energy and information flows respectively. The class of power ports is seperated in two subclasses : the physical ($\mathbf{P}_P$) and the thermodynamical ($\mathbf{P}_T$) ports while the class of information ports is decomposed in signal ($\mathbf{P}_S$) and data ($\mathbf{P}_D$) ports.

As shown in Fig. 2, **physical** ports : $\mathcal{P}_{\mathcal{O}1.1}, \mathcal{P}_{\mathcal{O}2.1} \Leftarrow \mathbf{P}_P$ are symbolized by a black circle. The state of a physical port is defined by a couple of *across*/*through* variables : $\mathbf{P}_P = (\alpha(t), \varphi(t))$. The power $P(t)$ associated with a physical port is given by : $\mathrm{P}(t) = \alpha(t) \cdot \varphi(t)$. By convention, the positive flow of *through* variables is oriented into the module. This convention is used to establish the power balance equation in each module.

Thermodynamical ports : $\mathcal{P}_{\mathcal{O}3.1}, \mathcal{P}_{\mathcal{O}4.1} \Leftarrow \mathbf{P}_T$ are symbolized by a black square. The state of a thermodynamical port is defined by a triplet of physical variables : two *across* variables : one pressure and one temperature, and one *through* variable : the volume flowrate.

Signal ports : $\mathcal{P}_{\mathcal{O}6.1} \Leftarrow \mathbf{P}_{S+}, \mathcal{P}_{\mathcal{O}5.1} \Leftarrow \mathbf{P}_{S-}$ are causal interfaces by which objects exchange input and output signals : $\mathbf{P}_{S+} = (u(t))$ or $\mathbf{P}_{S-} = (y(t))$ symbolized by a black and a white arrow respectively.

Data ports : $\mathcal{P}_{\mathcal{O}7.1}, \mathcal{P}_{\mathcal{O}8.1} \Leftarrow \mathbf{P}_D$ are non-causal interfaces by which objects exchange data. The flow causality is not pre-established. Those ports are symbolized by a black and white diamond.

Note that this proposition of interfaces (information and power ports) is not a limited description and can be extended by personal port developed by the user. However, as suggested in [11], modelers are advised not to define the ports of their models arbitrarily, but to restrain themselves, and only use proven connection mechanisms.

C. Link class

A link class $\mathbf{L}$ is a specific object which describes the mode of connexion between modules. A link class is defined by two attributes :

$$\mathbf{L} = (P_{\mathbf{L}}, B_{\mathbf{L}}) \tag{13}$$

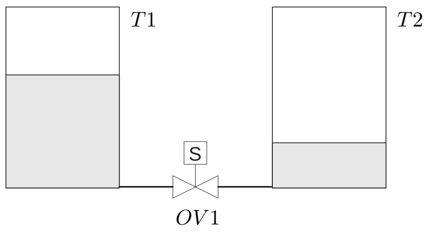


Fig. 3. Tank system

where: $P_{\mathbf{L}} = \{\mathbf{P_{Oi}}.k, \cdots, \mathbf{P_{Oj}}.l\}$ is the set of ports connected by $\mathbf{L}$, $\mathbf{P_{Oi}}.k$ denoting the k^{th} port of the module class $\mathbf{Oi}$, and $B_{\mathbf{L}}$ is the behavioral model of the link which defines the interconnexion law. This interconnection law depends on the class of the connected ports. Given the four classes of ports described in the previous paragraph, four interconnection laws, noted $B_{\mathbf{L}_P}$, $B_{\mathbf{L}_T}$, $B_{\mathbf{L}_S}$ and $B_{\mathbf{L}_D}$, are defined:

$$B_{\mathbf{L}_P} = \left\{ \begin{pmatrix} \mathbf{P}_1.\alpha \\ \vdots \\ \mathbf{P}_r.\alpha \end{pmatrix} \in \mathbb{A}^r, \begin{pmatrix} \mathbf{P}_1.\varphi \\ \vdots \\ \mathbf{P}_r.\varphi \end{pmatrix} \in \mathbb{F}^r \middle| (14) \right\}$$

$$\begin{pmatrix} \mathbf{P}_1.\alpha(t) = \cdots = \mathbf{P}_r.\alpha(t) \\ \mathbf{P}_1.\varphi(t) + \cdots + \mathbf{P}_r.\varphi(t) = 0 \end{pmatrix} \tag{14}$$

$$B_{\mathbf{L}_T} = \left\{ \begin{pmatrix} \mathbf{P}_1.p \\ \vdots \\ \mathbf{P}_r.p \\ \mathbf{P}_1.\tau \\ \vdots \\ \mathbf{P}_r.\tau \end{pmatrix} \in \mathbb{A}^{2r}, \begin{pmatrix} \mathbf{P}_1.q \\ \vdots \\ \mathbf{P}_r.q \end{pmatrix} \in \mathbb{F}^r \middle| (15) \right\}$$

$$\begin{pmatrix} \mathbf{P}_1.p(t) = \cdots = \mathbf{P}_r.p(t) \\ \mathbf{P}_1.\tau(t) = \cdots = \mathbf{P}_r.\tau(t) \\ \mathbf{P}_1.q(t) + \cdots + \mathbf{P}_r.q(t) = 0 \end{pmatrix} \tag{15}$$

$$B_{\mathbf{L}_S} = \{\mathbf{P}_1.u \in \mathbb{I}, \mathbf{P}_2.y \in \mathbb{O} | \quad \mathbf{P}_1.u(t) := \mathbf{P}_2.y(t)\}$$

$$B_{\mathbf{L}_D} = \left\{ (\mathbf{P}_1.w_1, \mathbf{P}_2.w_2) \in \mathbb{R}^2 \, | (16) \right\}$$

$$\mathbf{P}_1.w_1(t) = \mathbf{P}_2.w_2(t) \tag{16}$$

$B_{\mathbf{L}_P}$, $B_{\mathbf{L}_T}$, $B_{\mathbf{L}_S}$ and $B_{\mathbf{L}_D}$ represent the behavioral models of physical, thermodynamical, signal and data links respectively. $\mathbf{P}.x(t)$ corresponds to the variable $x(t)$ attached to the port $\mathbf{P}$. $\mathbb{A}$, $\mathbb{F}$, $\mathbb{I}$ and $\mathbb{O}$ denote the domains of *across*, *through*, *input* and *output* variables respectively. $r = \mathrm{card}(\mathbf{L})$ is the number of ports interconnected by the power link $\mathbf{L}$. In order to make the distinction between the information and energy flows in the multiport diagram, power ($\mathbf{L}_P, \mathbf{L}_T$) and information ($\mathbf{L}_S, \mathbf{L}_D$) links are represented by dotted and solid lines respectively.

IV. An example

A. Tank system

As shown in Fig. 3, the system is composed of two identical tanks T_1 and T_2 interconnected by an On/Off

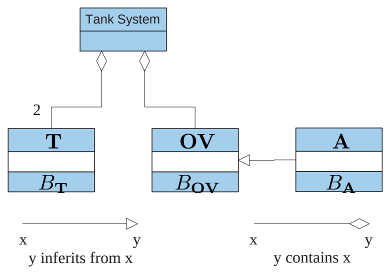
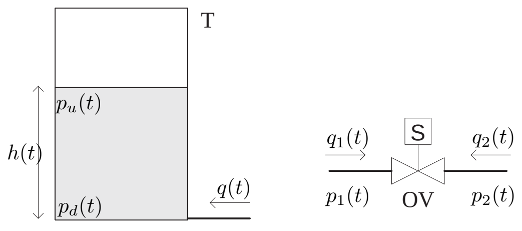


Fig. 4. UML class diagram



(a) Tank *(b)* On/Off Valve

Fig. 5. Icons of modules

valve: OV_1. The modelling procedure is broken down in four main steps :

1) Hierarchical object-decomposition
2) Definition of the module classes
3) Multiport diagram
4) Implementation into Modelica

B. Hierarchical decomposition

As shown in the *UML class diagram* in Fig. 4, the model of the tank system is based on three module-classes: **T** (tank), **OV** (On/Off valve) and **A** (actuator). This diagram points out the 'static' structure of the model by specifying the dependance links between the constitutive module classes in terms of composition and inheritance relationships. The model of the hydraulic system is composed of two instances : $\mathcal{T}1, \mathcal{T}2 \Leftarrow \mathbf{T}$ of the *Tank* module and one instance of the *On/Off valve* module : $\mathcal{OV}1 \Leftarrow$ **OV**. Moreover, the valve inherits generic properties of the actuator class **A**.

C. Definition of the module classes

1) Tank module class: The icon of the tank module class : **T** is presented in Fig. 5(a) in which the *through* variable: $q(t)$ is, by convention, oriented into the module. **T** is defined by :

$$\mathbf{T} = (\mathbb{U}_T, B_T, P_T) \tag{17}$$

Its behavior is defined by :

$$B_T(\theta, w, l) = \{t \in \mathbb{T}, \theta \in \Theta_T, w \in \mathbb{W}_T, l \in \mathbb{L}_T | (18)\}$$

$$\begin{cases} p_d(t) - p_u(t) = \rho g h(t) \\ q(t) = A\dot{h}(t) \\ p_u(t) = 10^5 Pa \end{cases} \tag{18}$$

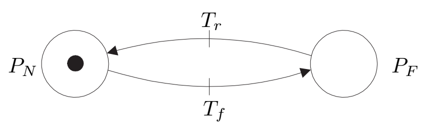


Fig. 6. Petri net of the actuator class

Its data *universum* is given by :

$$\mathbb{U}_T = (\mathbb{T} \times \Theta_T \times \mathbb{W}_T \times \mathbb{L}_T), \tag{19}$$

with :

$$\begin{aligned} t &\in \mathbb{T} = \mathbb{R}^+ \\ \theta &= \begin{pmatrix} \rho & g & A \end{pmatrix} \in \Theta_T = \mathbb{R}^{+3} \\ w(t) &= \begin{pmatrix} p_d(t) & q(t) \end{pmatrix} \in \mathbb{W}_T = \mathbb{R}^+ \times \mathbb{R} \\ l(t) &= \begin{pmatrix} p_u(t) & h(t) \end{pmatrix} \in \mathbb{L}_T = \mathbb{R}^{+2} \end{aligned}$$

where $p_u(t)$ and $p_d(t)$ are the pressures of the fluid at the top and at the bottom of the tank respectively. $q(t)$ is the flow rate and $h(t)$ is the level of water in the tank. ρ is the water density, g is the gravitation constant and A is the section area of the tank. Its interface is defined by :

$$P_T = \{\mathcal{P}_{T.1}\}, \tag{20}$$

where $\mathcal{P}_{T.1} \Leftarrow \mathbf{P}_P$ is a physical port given by :

$$\mathcal{P}_{T.1} = (p_d(t), q(t)) \tag{21}$$

2) Actuator module class: **A** is defined by :

$$\mathbf{A} = (\mathbb{U}_A, B_A) \tag{22}$$

Its behavior and its data *universum* are defined by :

$$\begin{aligned} & B_A(l) = \{t \in \mathbb{T}, l \in \mathbb{L}_A | \, Fig.\ 6\} \\ & t \in \mathbb{T} = \mathbb{R}^+ \\ & l(t) = (P_N, P_F, T_f, T_r) \in \mathbb{L}_A = \mathbb{B}^4 \\ & \mathbb{U}_A = (\mathbb{T} \times \mathbb{L}_A) \end{aligned} \tag{23}$$

P_N, P_F denote the places associated to the normal and fault states of the actuator, and T_f, T_r the logical conditions associated with the transitions between P_N and P_F.

3) On-Off valve module class: The scheme of the valve module class: **OV** is presented in Fig. 5(b) in which the *through* variables: $q_1(t)$ and $q_2(t)$ are, by convention, oriented into the module. **OV** is defined by :

$$\mathbf{OV} = (\mathbb{U}_{OV}, B_{OV}, P_{OV}) \tag{24}$$

Its behavior and its data *universum* are defined by :

$$B_{OV}(\theta, w, l) = \{t \in \mathbb{T}, \theta \in \Theta_{OV}, w \in \mathbb{W}_{OV}, l \in \mathbb{L}_{OV} | (25)\}$$

$$\begin{cases} z(t) = Open & if\ \{u(t) > 0\} \\ z(t) = Close & if\ \{u(t) \leq 0\} \\ q(t) - K_o * \Delta P(t) = 0 & if\ C_1 = true \\ q(t) = 0 & if\ C_2 = true \\ C_1 = (z(t) = Open)\ \textbf{and}\ \mathbf{A}.P_N = true & \\ C_2 = (z(t) = Closed)\ \textbf{or}\ \mathbf{A}.P_F = true & \\ q(t) = q_1(t) & \\ q(t) = -q_2(t) & \\ \Delta p(t) = p_1(t) - p_2(t). & \end{cases} \tag{25}$$

$$\mathbb{U}_{OV} = (\mathbb{T} \times \Theta_{OV} \times \mathbb{W}_{OV} \times \mathbb{L}_{OV}) \tag{26}$$

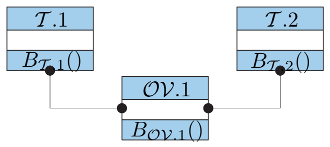


Fig. 7. Multiport diagram of the tank system

where:

$$t \in \mathbb{T} = \mathbb{R}^+$$
$$K_o \in \Theta_{OV} = \mathbb{R}$$
$$w(t) = \begin{pmatrix} p_1(t) & p_2(t) & q_1(t) & q_2(t) & u(t) \end{pmatrix} \in \mathbb{W}_{OV} = \mathbb{R}^5$$
$$l(t) = \begin{pmatrix} z(t) & \Delta p(t) & Q(t) \end{pmatrix} \in \mathbb{L}_{OV}$$
$$\mathbb{L}_{OV} = \{Open; Close\} \times \mathbb{R}^+ \times \mathbb{R}$$

$(p_1(t), p_2(t)) \in \mathbb{A}$, $(q_1(t), q_2(t)) \in \mathbb{F}$, $u(t) \in \mathbb{I}$ denote the upstrem/downstream pressures, the flow rates at the upstrem/downstream ports and the binary control signal respectively. The latent variable $l(t)$ of the valve is composed of its opening/closure state, the differential pressure and the internal flow rate. It is assumed that the valve is always kept in a normal state, i.e : $P_1 = true$ at $t = 0$ with $T_1 = T_2 = false$. Its ports are defined by :

$$P_{OV} = \{\mathcal{P}_{OV.1}, \mathcal{P}_{OV.2}, \mathcal{P}_{OV.3}\}, \tag{27}$$

where $\mathcal{P}_{OV.1}, \mathcal{P}_{OV.2} \Leftarrow \mathbf{P}_P$ are two physical ports and $\mathcal{P}_{OV.3} \Leftarrow \mathbf{P}_{S+}$ an input signal port given by :

$$\mathcal{P}_{OV.1} = (p_1(t), q_1(t)) \tag{28}$$
$$\mathcal{P}_{OV.2} = (p_2(t), q_2(t)) \tag{29}$$
$$\mathcal{P}_{OV.3} = (u(t)). \tag{30}$$

Parameters and constitutive laws of the components are supposed to be completely known. Consequently, their modules are described by white objects in the UML class diagram, Fig. 4.

D. Multiport diagram

The three modules instances: $\mathcal{T}1$, $\mathcal{T}2$ and $\mathcal{OV}1$, used in the model, are defined by :

$$B_{\mathcal{T}1} = B_{\mathbf{T}}(\rho, g, A_1, p_{u1}(t), p_{d1}(t), q_1(t), h_1(t))$$
$$B_{\mathcal{T}2} = B_{\mathbf{T}}(\rho, g, A_2, p_{u2}(t), p_{d2}(t), q_2(t), h_2(t))$$
$$B_{\mathcal{OV}1} = B_{\mathbf{OV/A}}(K_{o1}, p_3(t), p_4(t), q_3(t), q_4(t), u_1(t))$$

The latent variables have been removed from those definitions in order to lighten the equations. The multiport diagram of the tank system, presented in Fig. 7, is finally made up by connecting together the three modules instances in the same way as an experimenter would plug together the real components of the system. The three modules are connected by two power links, $\mathcal{L}_1$ and $\mathcal{L}_2$, corresponding to the exchanges of hydraulic energy between the components.

$$\mathcal{L}_1 = \mathbf{L}_P(\mathcal{P}_{\mathcal{T}1.1}, \mathcal{P}_{\mathcal{OV}1.1}) \tag{31}$$
$$\mathcal{L}_2 = \mathbf{L}_P(\mathcal{P}_{\mathcal{T}2.1}, \mathcal{P}_{\mathcal{OV}1.2}) \tag{32}$$

Multiport diagram	Modelica syntax
module class: $\mathbf{O}$	`model` class
link class: $\mathbf{L}$	`connect` instruction
port class : $\mathbf{P}$	`connector` class
time variable : $t \in \mathbb{T}$	`time` variable
types of parameters: Θ	`parameter` type
types of data: $\mathbb{N}$, $\mathbb{R}$, B	`integer, real, boolean`
through variables: $\in \mathbb{F}$	`flow` type
input variables: $\in \mathbb{I}$	`input` type
output variables: $\in \mathbb{O}$	`output` type
local variable: $\in \mathbb{L}$	`local` type
behavioral equations	`equation` section
input-output equations	`algorithm` section
$\mathbb{U}_{\mathbf{O}}$	data statement section
$P_{\mathbf{O}}$	port statement section
$B_{\mathbf{O}}$	model definition section

TABLE II

MULTIPORT DIAGRAM / MODELICA LANGUAGE

The multiport diagram of the system is defined by :

$$\Delta = (O, L) \tag{33}$$

with : $O = \{\mathcal{T}1, \mathcal{T}2, \mathcal{OV}1\}$ and $L = \{\mathcal{L}1, \mathcal{L}2\}$. Its full behavior is given by :

$$B_\Delta = B_{\mathcal{T}1} \cap B_{\mathcal{T}2} \cap B_{\mathcal{OV}1} \cap B_{\mathcal{L}1} \cap B_{\mathcal{L}2}. \tag{34}$$

The latter equation means that the mathematical expression of B_Δ is obtained by gathering all the behavioral equations of modules and links, finally leading to a differential algebraic equation system.

E. Implementation into Modelica

Table II points out some correspondances between the elements of the multiport diagram and the instructions of the Modelica language. This table makes the translations of $\mathbf{T}, \mathbf{OV}$ and $\mathbf{L}_P$ into Modelica easier. Algorithms III and IV present the implementation of $\mathbf{OV}$ and $\mathbf{T}$ into Modelica . In both cases, $\mathbb{U}_{\mathbf{O}}$, $P_{\mathbf{O}}$ and $B_{\mathbf{O}}$ correspond to the sections: *data statement*, *port statement* and *behavioral model* of their Modelica algorithm. The **extends** clause specifies that the partial model of the *actuator* class is extended to build the complete (full) model of the valve and the instruction **der**(w) means the time derivative of w. The algorithm V shows the implementation of the physical ports into Modelica. For each port, a **connector** class is defined. Connecting power ports means that *across* variables are equal while *through* variables (marked by the prefix **flow**) are sum to zero. The algorithm VI presents the implementation of the multiport diagram. As indicated in table II, the power links $\mathcal{L}_1$ and $\mathcal{L}_2$, defined in Eq. (31) and Eq. (32) are implemented with the **connect** instruction.

The graphical user interface of Dymola© allows to add icons in the definition of the modules. The implementation of the multiport diagram into the graphical environment of Dymola is presented in Fig. 8. The latter emphasizes the similarity between the implemented model and the process and instrumentation diagram of the system in Fig. 3. Such a model can now be used for a simulation purpose.

```
model ONOFFValve
extends MOODModels.ASTRIDLibrary.ElementActuator;
  // DATA STATEMENT
  parameter Real Ko=6.26*1e-8;     //Valve constant
  Real P2;     //(Pa) Output pressure
  Real P1;     //(Pa) Input pressure
  Real DP;     //(Pa)
  Real Q;     //(m3/s) Throughput via the valve
  Real Q1;     //(m3/s) Throughput via the valve
  Real Q2;     //(m3/s) Throughput via the valve
  Real u;     // Control signal of the valve
  Boolean z;     // State of the valve
  // PORT STATEMENT
  Power Power1;
  Power Power2;
  SignalIn SignalIn1;
  // BEHAVIORAL MODEL
equation
  // Affection of manifest variables to ports
  Power2.e = P2;
  Power1.e = P1;
  Power2.f = Q2;
  Power1.f = Q1;
  SignalIn.signal[1]= u;
  // Behavioral equations
  z = u>0;
  T2.condition := false;     // Error condition of the Actuator
  T1.condition := false;     // Release condition of the Actuator
  if P2.state then     // Fault state
    Q = 0;
  elseif 1 then
    Q = Ko*DP;
  else
    Q = 0;
  end if;
  DP=P1-P2;
  Q=Q1;
  Q=-Q2;
end ONOFFValve
```

TABLE III

MODELICA MODEL OF THE ON/OFF VALVE

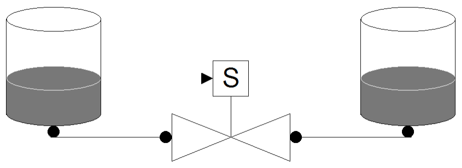


Fig. 8. Multiport diagram implemented into Dymola/Modelica

V. CONCLUSIONS

The formalism of the behavioural modelling approach proposed by Willems in the eighties is shown to be a reliable and suitable mathematical framework for the description of the main concepts of the object-oriented paradigm. An object-oriented model structure, entitled multiport diagram, lied within this behavioural framework is developed for interconnected systems modelling. A procedure for working out this diagram has been pointed out. Finally, an implementation solution of the multiport diagram into the modelling language Modelica© is brought out.

REFERENCES

[1] T. Bastogne, "A multiport object-oriented diagram for batch process modelling," in *IFAC Conference on Analysis and Design of Hybrid Systems*, Saint-Malo, France, 2003.

[2] T. Bastogne and A. Libaux, "An experimental object-oriented modelling of an hydraulic valley," in *13th IFAC Symposium on System Identification*, Rotterdam, Netherland, August 2003.

```
model Tank
  // DATA STATEMENT
  constant Real g=9.81;
  constant Real rho=1000;
  parameter Real A=0.04;
  Real Pu=1e5;
  Real Pd;
  Real Q;
  Real levelh;
  // PORT STATEMENT
  Power Power1;                    // output power
  // BEHAVIORAL MODEL
equation
  Power1.e = Pd;                   // Output
  Power1.f = -Q;
  Pd-Pu = rho*g*level;
  der(level) = -Q/A;
end Tank
```

TABLE IV

MODELICA MODEL OF THE TANK

```
connector Power
    Real a;
    flow Real f;
end Power
```

TABLE V

MODELICA MODEL OF $\mathbf{P}_P$

[3] J. M. Maciejowski, "Reconfigurable control using constrained optimization," in *Proc. of the 4th European Control Conference*, G. Bastin and M. Gevers, Eds., vol. Plenary Lectures and Mini-Courses, July 1997, pp. 107–130.

[4] H. Elmqvist, "A structured model language for large continuous systems," Ph.D. dissertation, Dept. of Automatic Control, Lund Institute of Technology, Sweden, 1978, report CODEN: LUTFD2(/TFRT-1015).

[5] G. Fabian, D. A. van Beek, and J. E. Rooda, "Integration of the discrete and the continuous behavior in the hybrid Chi simulator," in *European Simulation Multiconference*, Manchester, 1998, pp. 252–257.

[6] H. Elmqvist, D. Brck, and M. Otter, *Dymola - Dynamic Modeling Laboratory. User's Manual*. Dynasim AB, 1999.

[7] W. Borutzki, "Relations between bond graph based and object-oriented physical systems modeling," in *International Conference on Bond Graph Modeling and Simulation, ICBGM'99*, San Francisco, CA, January 17-20 1999, pp. 11–17.

[8] C. Pantelides, "The consistent initialization of differential-algebraic systems," *SIAM Journal of Scientific and Statistical Computing*, no. 9, pp. 213–231, 1988.

[9] J. C. Willems, "From time series to linear systems," *Automatica*, 1986, part I: Vol. 22, No. 5, pp. 561-580, 1986, Part II: Vol. 22, No. 6, pp. 675-694, 1986, Part III: Vol. 23, No. 1, pp. 87-115, 1987.

[10] J. W. Polderman and J. C. Willems, *Introduction to Mathematical Systems Theory - A Behavioral Approach*, ser. Texts in Applied Mathematics, 26. Springer, 1997.

[11] F. E. Cellier, *Continuous System Modeling*. Springer-Verlag, 1991.

```
model TankSystem
    ONOFFValve Valve1;
    Tank Tank1;
    Tank Tank2;
equation
    connect(Tank1.Power1, Valve1.Power1);
    connect(Valve1.Power2, Tank2.Power1);
end TankSystem
```

TABLE VI

MODELICA MODEL OF THE TANK SYSTEM