Object-Oriented Description of Hybrid Dynamic Systems of Variable Structure

Alfonso Urquia
Sebastian Dormido
Department of Computer Science and Automatic Control
ETS de Ingeniería Informática, UNED
Juan del Rosal, 16
28040 Madrid, Spain
aurquia@dia.uned.es
sdormido@dia.uned.es

Existing object-oriented modeling environments only support the simulation of a limited type of hybrid dynamic models of variable structure: those with exactly the same state variables and the same algebraic variables in all modes. The most general hybrid dynamic model of variable structure is one in which the number of state variables and algebraic variables is not necessarily equal in all modes. A new algorithm, which transforms such a general variable-structure model into a model suitable for simulation by means of existing object-oriented modeling languages of hybrid systems, is proposed.

Keywords: Object-oriented modeling languages, hybrid models, variable-structure models

1. Introduction

The object-oriented design methodology has demonstrated to be a success when it is adapted to the modeling of dynamic systems, increasing the programmer's productivity and the software quality [1-3]. The practice of the object-oriented modeling of dynamic systems is a reality nowadays, thanks to the development in the 1990s of object-oriented modeling languages, supported by efficient software tools (modeling environments). These environments work in conjunction with potent differential algebraic equations (DAE) solver algorithms (e.g., DASSL [4]).

Some examples of general-purpose, object-oriented modeling languages are ABACUSS II [5], ASCEND [6], Dymola [7], EcosimPro [8], gPROMS [1], Modelica [9], and Omola [3]. Dymola (Dynamic Modeling Laboratory) [7, 10] was the first modeling language on the market designed to allow the object-oriented description of hybrid models of large and complex systems [11]. The common characteristics of these modeling languages are the object-oriented, noncausal modeling methodology and the need for automatic symbolic formula manipulation. The modeling knowledge is represented as differential, algebraic, and discrete equations that may change by being triggered by events (i.e., hybrid models). Modeling languages support a declarative description of the model, which permits a better reuse of the model because it is based on equations instead of assignment statements. The software tools supporting these modeling languages implement algorithms to automatically decide which equation to use for calculating each unknown variable.

In addition, the automatic manipulations that these modeling environments carry out on the model include the following: (1) translation of the object-oriented description of the model into the so-called flat model (complete set of model equations and functions, with all object-oriented structure removed) and (2) manipulations intended to transform the model into an efficiently solvable form. This second type of manipulations includes (1) the efficient formulation of the complete-model equations, eliminating the redundant variables and the trivial equations resulting from the submodel connections; (2) the sorting of the equations; (3) the symbolic manipulation of those equations in which the unknown variable appears linearly; and (4) the tearing of the nonlinear algebraic loops. In addition, some modeling environments (e.g., Dymola [7]) support the automatic detection of the models with an index greater than 1, as well as the automatic reduction of the index to 0 or 1 by means of symbolic formula manipulation [12, 13].

General-purpose, object-oriented modeling languages support the description and simulation of certain types of variable-structure models (e.g., by means of clauses of the following type [7-9]: <expression> = if <condition> then <expression1> else <expression2>). A model has a variable structure when its mathematical description changes during the simulation run. This variable-structure model

SIMULATION, Vol. 79, Issue 9, September 2003 485-493

DOI: 10.1177/003759703038882

consists of the equations describing the behavior of the system in each mode and the change-of-mode event conditions. When one of these discrete event conditions is triggered, the corresponding change of mode takes place: the system behavior is no longer described by the actual mode equations but by the new mode equations. This type of model arises not only when intrinsically variable structure systems are modeled but also as a result of modeling strategies (e.g., when different models of a system are available, and a change is required among the different models during the simulation run). This situation is common when different models of a system are developed under a different hypothesis (e.g., to get models with a different degree of detail in the description of the system).

1.1 Restrictions of the Simulation of Variable-Structure Models

An important limitation of the existing modeling languages is that they only support the modeling and simulation of a limited type of variable-structure models. The restriction is the following: the model must have the same algebraic and state variables in all the modes. In other words, the number of state variables and algebraic variables must be equal in all modes, and the computational type (i.e., state variable or algebraic variable) of each variable must be the same in all modes. The equation used to evaluate each variable can vary from one mode to another [14, 15]. To illustrate this point, we next show a very simple model of variable structure, which needs to be reformulated to be simulated using the modeling environments of hybrid systems.

Example 1. The model of a closed recipient, containing a known mass (m_0, constant) of a pure component, is shown in Figure 1. The recipient exchanges heat with the environment. The heat flow, Q, is a known function of time. The boiling and condensation temperature of the component, T_{cf}, and its heat of vaporization (positive) per unit of mass, λ, are also known. The heat capacity per mass unit of the liquid (vapor), C_{liq} (C_{vap}), are considered constant. F_{cf} represents the mass flow, due to the phase change, between the liquid and the vapor. The variable T represents the component temperature.

The system can be in one of the three following modes: (1) *liq_phase*, while the component is in the liquid phase; (2) *vap_phase*, while the component is in the vapor phase; and (3) *equilibrium*, while the component is in liquid-vapor equilibrium. Two Boolean variables, *liquid* and *equilib*, are required to specify the system mode. The relationship between the value of these Boolean variables and the mode is as follows: *liq_phase* mode = {*liquid* = True, *equilib* = False}, *equilibrium* mode = {*liquid* = False, *equilib* = True), and *vap_phase* mode = {*liquid* = False, *equilib* = False}.

The mathematical descriptions of the *liq_phase* and *vap_phase* modes have the same algebraic variables and the same state variables. Therefore, they can be appropriately described and simulated using the existing modeling languages (e.g., using if-then-else clauses [7-9]). However, it is impossible to obtain a combined formulation of the model valid for its three modes, as explained in detail below.

The model has one state variable, T, and three algebraic variables, Q, m_{liq}, and m_{vap}, while it is in the *liq_phase* or *vap_phase* mode. While it is in the *equilibrium* mode, it has one state variable, m_{vap}, and four algebraic variables, F_{cf}, Q, m_{liq}, and T. The variables T and m_{vap} act as state variables or as algebraic variables, depending on the mode. The variable F_{cf} is defined only in the *equilibrium* mode. As a consequence, the complete model cannot be simulated using the existing modeling environments: it needs to be adequately reformulated.

1.2 Contribution of This Study

The automated processing of hybrid dynamic systems of variable structure, in which the number of state variables and algebraic variables is not necessarily equal in all modes, is considered. A novel algorithm [16], which transforms such a model into a model suitable for description and simulation by means of existing object-oriented modeling languages of hybrid systems, is proposed in this study.

2. Description of the Algorithm

Premise. It is assumed that the model index [4] is not greater than 1 in any mode. This requirement does not impose any practical limitation. Prior to the algorithm application, the index of each mode formulation should be calculated and, when required, reduced to 0 or 1. Some modeling environments (e.g., Dymola [7]) support automatic index calculation and index reduction by means of symbolic formula manipulation [12, 13].

Model Formulation Prior to the Algorithm Application. Consider the following representation of a hybrid dynamic model of variable structure with two modes, mode_0 and mode_1. As stated previously, it is assumed that the index model is not greater than 1 in any of the two modes.

When the model is in mode_0, it is described by the following $\dim(\mathbf{f_0})$ equations:

$$\mathbf{f_0}\left(\mathbf{y_0}, \mathbf{y_{01}}, \mathbf{x_0}, \dot{\mathbf{x}}_0, \mathbf{x_0^*}, \dot{\mathbf{x}}_0^*, \mathbf{x_{01}}, \dot{\mathbf{x}}_{01}, \mathbf{x_1^*}\right) = \mathbf{0}, \qquad (1a)$$

where $\dim(\mathbf{f_0}) = \dim(\mathbf{y_0}) + \dim(\mathbf{y_{01}}) + \dim(\mathbf{x_0}) + \dim(\mathbf{x_0^*}) + \dim(\mathbf{x_{01}}) + \dim(\mathbf{x_1^*})$.

When the model is in mode_1, it is described by the following $\dim(\mathbf{f_1})$ equations:

$$\mathbf{f_1}\left(\mathbf{y_1}, \mathbf{y_{01}}, \mathbf{x_1}, \dot{\mathbf{x}}_1, \mathbf{x_1^*}, \dot{\mathbf{x}}_1^*, \mathbf{x_{01}}, \dot{\mathbf{x}}_{01}, \mathbf{x_0^*}\right) = \mathbf{0}, \qquad (1b)$$

where $\dim(\mathbf{f_1}) = \dim(\mathbf{y_1}) + \dim(\mathbf{y_{01}}) + \dim(\mathbf{x_1}) + \dim(\mathbf{x_1^*}) + \dim(\mathbf{x_{01}}) + \dim(\mathbf{x_0^*})$.

As the discrete part of the model describing both modes may be formulated very easily from the separate description of the discrete part of the two modes, the algorithm

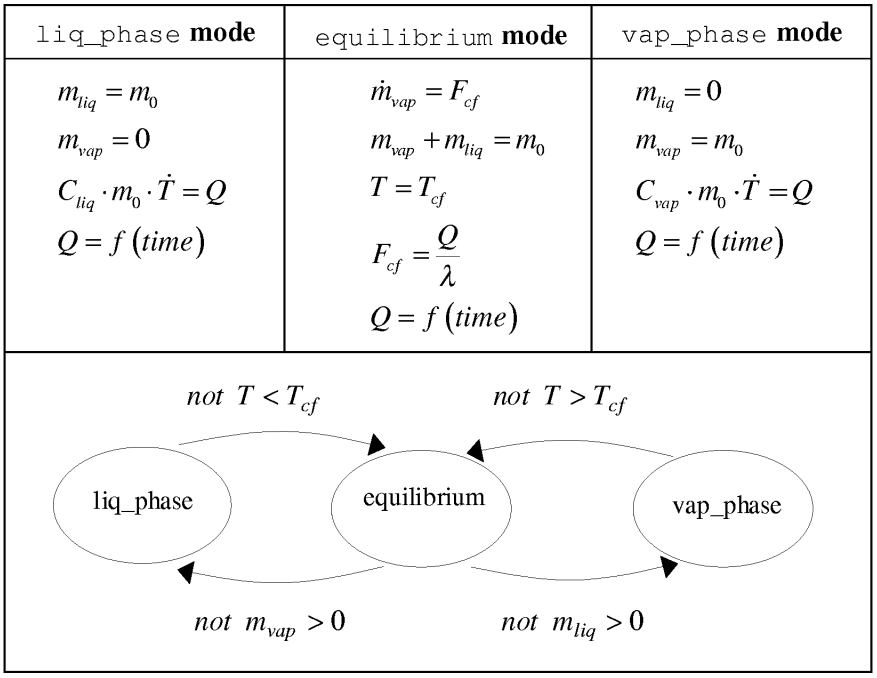

Figure 1. Model of variable structure with three modes

does not consider the discrete part of the model. Consequently, for the sake of clarity, the discrete part of the model has been omitted from the notation.

The following notation is introduced for the model variables (see Fig. 2):

- $\mathbf{x_0}$: state variables in mode_0 and not defined in mode_1
- $\mathbf{x_0^*}$: state variables in mode_0 and algebraic variables in mode_1
- $\mathbf{x_1}$: state variables in mode_1 and not defined in mode_0
- $\mathbf{x_1^*}$: state variables in mode_1 and algebraic variables in mode_0
- $\mathbf{x_{01}}$: state variables in mode_0 and in mode_1
- $\mathbf{y_0}$: algebraic variables in mode_0 and not defined in mode_1
- $\mathbf{y_1}$: algebraic variables in mode_1 and not defined in mode_0
- $\mathbf{y_{01}}$: algebraic variables in mode_0 and in mode_1

First Step of the Algorithm. Variables $\mathbf{x_1}$ and $\mathbf{y_1}$ are defined in mode_1 and undefined in mode_0. Variables $\mathbf{x_0}$ and $\mathbf{y_0}$ are defined in mode_0 and undefined in mode_1. The purpose of this first step of the algorithm is to extend the vector of variables in every mode, in such a way that the extended vector of variables is the same in both modes (i.e., $\{\mathbf{y_0}, \mathbf{y_1}, \mathbf{y_{01}}, \mathbf{x_0}, \mathbf{x_0^*}, \mathbf{x_{01}}, \mathbf{x_1}, \mathbf{x_1^*}\}$).

To obtain the same variables vector in both modes, the undefined variables of each mode are (arbitrarily) defined, setting them equal to zero. The resulting model has

$$\dim(\mathbf{y_0}) + \dim(\mathbf{y_1}) + \dim(\mathbf{y_{01}}) + \dim(\mathbf{x_0}) + \dim(\mathbf{x_0^*}) + \dim(\mathbf{x_{01}}) + \dim(\mathbf{x_1}) + \dim(\mathbf{x_1^*})$$

equations per mode. The model equations after the first step of the algorithm are the following:

Equations of mode_0:

$$\begin{cases} \mathbf{f_0}\left(\mathbf{y_0}, \mathbf{y_{01}}, \mathbf{x_0}, \dot{\mathbf{x}}_\mathbf{0}, \mathbf{x_0^*}, \dot{\mathbf{x}}_\mathbf{0}^*, \mathbf{x_{01}}, \dot{\mathbf{x}}_\mathbf{01}, \mathbf{x_1^*}\right) = \mathbf{0} \\ \mathbf{x_1} = \mathbf{0} \\ \mathbf{y_1} = \mathbf{0} \end{cases} \quad (2a)$$

Equations of mode_1:

$$\begin{cases} \mathbf{f_1}\left(\mathbf{y_1}, \mathbf{y_{01}}, \mathbf{x_1}, \dot{\mathbf{x}}_\mathbf{1}, \mathbf{x_1^*}, \dot{\mathbf{x}}_\mathbf{1}^*, \mathbf{x_{01}}, \dot{\mathbf{x}}_\mathbf{01}, \mathbf{x_0^*}\right) = \mathbf{0} \\ \mathbf{x_0} = \mathbf{0} \\ \mathbf{y_0} = \mathbf{0} \end{cases} \quad (2b)$$

Second Step of the Algorithm. The variables $\mathbf{x_0}$ and $\mathbf{x_0^*}$ are state variables of mode_0 and algebraic variables of mode_1. The variables $\mathbf{x_1}$ and $\mathbf{x_1^*}$ are state variables of mode_1 and algebraic variables of mode_0. The purpose

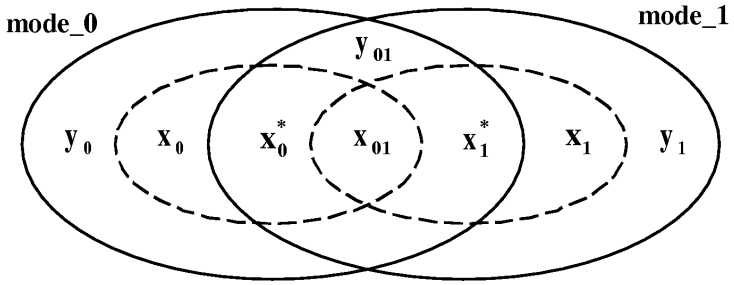

Figure 2. Variables classification of a model with two modes. Two concentric ellipses per mode. Inner ellipse: state variables. Outer ellipse: algebraic variables.

of this and the following steps of the algorithm is to extend the vector of state variables and the vector of algebraic variables in each mode, in such a way that the vectors of extended state variables and extended algebraic variables are the same for both modes.

To achieve this goal, we define $2 \cdot N$ new variables in each mode: $\{\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_1^*, \boldsymbol{\beta}_0, \boldsymbol{\beta}_0^*, \boldsymbol{\beta}_1, \boldsymbol{\beta}_1^*\}$, where $N = \dim(\mathbf{x_0}) + \dim(\mathbf{x}_0^*) + \dim(\mathbf{x_1}) + \dim(\mathbf{x}_1^*)$. These vectors of variables have the following number of elements:

$$\dim(\boldsymbol{\alpha}_0) = \dim(\boldsymbol{\beta}_0) = \dim(\mathbf{x_0})$$
$$\dim(\boldsymbol{\alpha}_0^*) = \dim(\boldsymbol{\beta}_0^*) = \dim(\mathbf{x}_0^*)$$
$$\dim(\boldsymbol{\alpha}_1) = \dim(\boldsymbol{\beta}_1) = \dim(\mathbf{x_1})$$
$$\dim(\boldsymbol{\alpha}_1^*) = \dim(\boldsymbol{\beta}_1^*) = \dim(\mathbf{x}_1^*)$$

These new $2 \cdot N$ variables are defined by means of $2 \cdot N$ equations per mode as follows:

Equations of mode_0:

$$\begin{cases} \mathbf{f_0}\left(\mathbf{y_0}, \mathbf{y_{01}}, \mathbf{x_0}, \dot{\mathbf{x}}_0, \mathbf{x}_0^*, \dot{\mathbf{x}}_0^*, \mathbf{x_{01}}, \dot{\mathbf{x}}_{01}, \mathbf{x}_1^*\right) = \mathbf{0} \\ \mathbf{x_1} = \mathbf{0} \quad , \quad \mathbf{y_1} = \mathbf{0} \\ \dot{\boldsymbol{\alpha}}_0 = \boldsymbol{\beta}_0 \quad , \quad \boldsymbol{\alpha}_0 = \mathbf{x_0} \\ \dot{\boldsymbol{\alpha}}_0^* = \boldsymbol{\beta}_0^* \quad , \quad \boldsymbol{\alpha}_0^* = \mathbf{x}_0^* \\ \dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\beta}_1 \quad , \quad \dot{\mathbf{x}}_1 = \boldsymbol{\beta}_1 \\ \dot{\boldsymbol{\alpha}}_1^* = \boldsymbol{\beta}_1^* \quad , \quad \dot{\mathbf{x}}_1^* = \boldsymbol{\beta}_1^* \end{cases} \tag{3a}$$

Equations of mode_1:

$$\begin{cases} \mathbf{f_1}\left(\mathbf{y_1}, \mathbf{y_{01}}, \mathbf{x_1}, \dot{\mathbf{x}}_1, \mathbf{x}_1^*, \dot{\mathbf{x}}_1^*, \mathbf{x_{01}}, \dot{\mathbf{x}}_{01}, \mathbf{x}_0^*\right) = \mathbf{0} \\ \mathbf{x_0} = \mathbf{0} \quad , \quad \mathbf{y_0} = \mathbf{0} \\ \dot{\boldsymbol{\alpha}}_0 = \boldsymbol{\beta}_0 \quad , \quad \dot{\mathbf{x}}_0 = \boldsymbol{\beta}_0 \\ \dot{\boldsymbol{\alpha}}_0^* = \boldsymbol{\beta}_0^* \quad , \quad \dot{\mathbf{x}}_0^* = \boldsymbol{\beta}_0^* \\ \dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\beta}_1 \quad , \quad \boldsymbol{\alpha}_1 = \mathbf{x_1} \\ \dot{\boldsymbol{\alpha}}_1^* = \boldsymbol{\beta}_1^* \quad , \quad \boldsymbol{\alpha}_1^* = \mathbf{x}_1^* \end{cases} \tag{3b}$$

Third Step of the Algorithm. As a consequence of the model extension made in the second step of the algorithm, the index [4] of each mode formulation is greater than 1. This third step of the algorithm consists of reducing the index to 0 or 1 [12, 13] of the two modes separately. In both modes, the variables $\{\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_0^*, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_1^*\}$ should be selected as state variables, and the variables $\{\mathbf{x_0}, \mathbf{x}_0^*, \mathbf{x_1}, \mathbf{x}_1^*\}$ should be selected as algebraic variables (i.e., their derivatives are substituted by auxiliary variables [13]). Modeling environments supporting index reduction by means of symbolic formula manipulation (e.g., Dymola [7]) can be used to carry out these index reductions.

Fourth Step of the Algorithm. The index reductions of the third step can introduce new algebraic variables. Each of these new variables may be defined either in both modes or only in one of them. This fourth step of the algorithm consists of extending the definition of those new variables defined only in one of the modes to the mode where they are undefined. As it was done in the first step of the algorithm, the variables are defined by setting them (arbitrarily) equal to zero.

Fifth Step of the Algorithm. The following are some comments about the change of the state variable values (i.e., $\{\boldsymbol{\alpha}_0, \alpha_0^*, \boldsymbol{\alpha}_1, \alpha_1^*\}$) when the mode transitions take place:

- The definition of those variables that acted as state variables in one of the modes and were not defined in the other mode (i.e., $\mathbf{x_0}$ and $\mathbf{x_1}$) has been extended in the third step of the algorithm. The variables $\boldsymbol{\alpha_0}$ and $\boldsymbol{\alpha_1}$ constitute these extended definitions (see equations (3a) and (3b)): (1) the state variable $\boldsymbol{\alpha_1}$ equals $\mathbf{x_1}$ in mode_1, and $\dot{\boldsymbol{\alpha}}_\mathbf{1}$ equals zero in mode_0; in addition, (2) the state variable $\boldsymbol{\alpha_0}$ equals $\mathbf{x_0}$ in mode_0, and $\dot{\boldsymbol{\alpha}}_\mathbf{0}$ equals zero in mode_1. In other words, while the system is in the mode where $\mathbf{x_i}$ was undefined, $\boldsymbol{\alpha_i}$ is equal to the value that $\mathbf{x_i}$ had just before the transition to this mode. $\boldsymbol{\alpha_i}$ maintains this constant value until the next change of mode.
- In the original formulation of the model, the variables $\{\mathbf{x}_0^*, \mathbf{x}_1^*\}$ act as state variables in one of the modes ($\mathbf{x}_0^*$ in mode_0 and $\mathbf{x}_1^*$ in mode_1) and as algebraic variables in the other mode ($\mathbf{x}_0^*$ in mode_1 and $\mathbf{x}_1^*$ in mode_0). When the mode transition takes place and one variable changes from acting as an algebraic variable to acting as a state

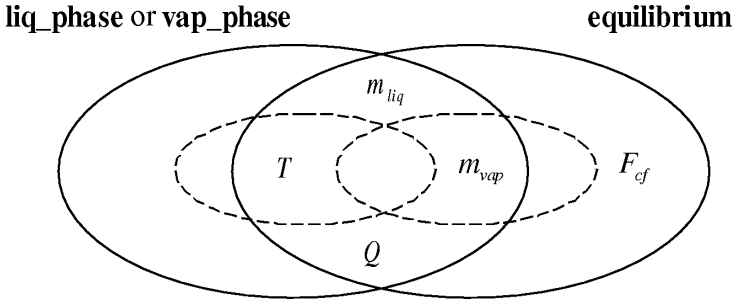

Figure 3. Variables classification

variable, the transition of the variable value is continuous. On the contrary, when the mode transition takes place and one variable changes from acting as a state variable to acting as an algebraic variable, the transition of the variable value is, in general, discontinuous. The variables $\boldsymbol{\alpha}_0^*$ and $\boldsymbol{\alpha}_1^*$ do not reflect (as they should) this discontinuous transition: they are equal to $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ while these act as state variables, and they change in time like $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ while these act as algebraic variables. The fifth step of the algorithm consists of adding to the model the instantaneous equations needed to modify the value of $\boldsymbol{\alpha}_0^*$ and $\boldsymbol{\alpha}_1^*$ when $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ change from state variables to algebraic variables.

$$\begin{aligned}&\text{when (transition from mode\_0 to mode\_1) then}\\&\boldsymbol{\alpha}_0^* \leftarrow \mathbf{x}_0^*\\&\text{end when}\\&\text{when (transition from mode\_1 to mode\_0) then}\\&\boldsymbol{\alpha}_1^* \leftarrow \mathbf{x}_1^*\\&\text{end when}\end{aligned} \tag{4}$$

A Final Comment. To transform a variable-structure model with M modes, the proposed algorithm has to be applied $M - 1$ times. First, it is applied to two modes. Next, it is applied to the obtained two-modes model, to a third mode, and so forth.

3. An Application Example

The algorithm is applied to the variable-structure model discussed in example 1. The model formulation prior to the algorithm application is shown in Table 1, and a classification of the model variables, following the same criterion as in Figure 2, is shown in Figure 3. The definition of the variable F_{cf} is extended in the first step of the algorithm (see Table 1). The second step of the algorithm consists of defining as many couples of auxiliary variables $\{\alpha_i, \beta_i\}$ as there are noncommon state variables contained in the model. In this example, there are two noncommon state variables: T and m_{vap} (see Table 1).

The third step of the algorithm consists of reducing the index of each mode separately. In this case, the index reductions are done using the Dymola modeling environment [7]. The model and the commands to reduce the index are shown in Table 2. The equations added to the model by Dymola, to reduce the index, are shown in Table 3. The variables α_1 and α_2 are chosen as state variables.

The derivatives of those variables that do not behave as state variables are substituted by auxiliary variables [13]: $\dot{T} \rightarrow derT$ and $\dot{m}_{vap} \rightarrow derm_{vap}$. The model, valid in its three modes and written according to the modeling language rules, is shown in Table 4. The last step of the algorithm consists of adding to the model the instantaneous equations to initialize the value of the variables α_1 and α_2 after the change-of-mode transitions. The two instantaneous equations are shown in Table 5.

The code of the complete model written in Dymola [7] is shown in Table 6. Arbitrary values have been assigned to the parameters. In this example, the state variable values (temperature and vapor mass) change in a continuous way when the mode transitions take place. However, to illustrate the state variables initialization at the change-of-mode events, discontinuous changes in the state variable values are forced. Initially, the system is in the *liq_phase* mode. At $Time = 10$, the system changes to the *equilibrium* mode. At $Time = 20$, it changes to the *vap_phase* mode and returns to the equilibrium mode at $Time = 30$ (see Fig. 4). The evolution in time of the liquid mass, vapor mass, and temperature is shown in Figure 4.

4. Conclusions

Specific limitations of the existing modeling languages for describing and simulating hybrid dynamic systems of variable structure have been identified. In this article, a novel algorithm has been proposed to transform any variable-structure model into a formulation suitable for description and simulation by means of existing object- oriented modeling languages of hybrid systems. Finally, the application of the proposed algorithm has been illustrated by means of an example.

Table 1. Initial model formulation. Algorithm application: first and second steps

	liq_phase or vap_phase mode	*equilibrium* mode
Initial	$m_{liq} = \text{if } liquid \text{ then } m_0 \text{ else } 0$ $m_{vap} = \text{if } liquid \text{ then } 0 \text{ else } m_0$ $\dot{T} = \text{if } liquid \text{ then } \frac{Q}{m_0 \cdot C_{liq}} \text{ else } \frac{Q}{m_0 \cdot C_{vap}}$ $Q = f(time)$	$\dot{m}_{vap} = F_{cf}$ $m_{vap} + m_{liq} = m_0$ $T = T_{cf}$ $F_{cf} = \frac{Q}{\lambda}$ $Q = f(time)$
After the first step	$m_{liq} = \text{if } liquid \text{ then } m_0 \text{ else } 0$ $m_{vap} = \text{if } liquid \text{ then } 0 \text{ else } m_0$ $\dot{T} = \text{if } liquid \text{ then } \frac{Q}{m_0 \cdot C_{liq}} \text{ else } \frac{Q}{m_0 \cdot C_{vap}}$ $Q = f(time)$ $F_{cf} = 0$	$\dot{m}_{vap} = F_{cf}$ $m_{vap} + m_{liq} = m_0$ $T = T_{cf}$ $Q = f(time)$ $F_{cf} = \frac{Q}{\lambda}$
After the second step	$m_{liq} = \text{if } liquid \text{ then } m_0 \text{ else } 0$ $m_{vap} = \text{if } liquid \text{ then } 0 \text{ else } m_0$ $\dot{T} = \text{if } liquid \text{ then } \frac{Q}{m_0 \cdot C_{liq}} \text{ else } \frac{Q}{m_0 \cdot C_{vap}}$ $Q = f(time)$ $F_{cf} = 0$ $\dot{\alpha}_1 = \beta_1, \quad \alpha_1 = T$ $\dot{\alpha}_2 = \beta_2, \quad \dot{m}_{vap} = \beta_2$	$\dot{m}_{vap} = F_{cf}$ $m_{vap} + m_{liq} = m_0$ $T = T_{cf}$ $F_{cf} = \frac{Q}{\lambda}$ $Q = f(time)$ $\dot{\alpha}_1 = \beta_1, \quad \dot{T} = \beta_1$ $\dot{\alpha}_2 = \beta_2, \quad \alpha_2 = m_{vap}$

Table 2. Model description (above) and index reduction commands (below) [7]

líq_phase or *vap_phase* mode	*equilibrium* mode

model liq_vap_phase
  local T, Q, mvap, mliq, Fcf, liquido
  local alfa1, alfa2, beta1, beta2
  parameter m0, lam, Cliq, Cvap
  mliq = **if** liquid **then** m0 **else** 0
  mvap = **if** liquid **then** 0 **else** m0
  der(T) = **if** liquid ->
  then Q / (m0 * Cliq) ->
  else Q / (m0 * Cvap)
  Fcf = 0
  der(alfa1) = beta1
  der(alfa2) = beta2
  alfa1 = T
  der(mvap) = beta2
  Q = f(Time)
  new(liquid) = g(Time)
end
	model equilibrium
	  local T, Q, mvap, mliq, Fcf
	  local alfa1, alfa2, beta1, beta2
	  parameter m0, lam, Cliq, Cvap, Tcf
	  der(mvap)=Fcf
	  mvap + mliq = m0
	  T = Tcf
	  Fcf = Q / lam
	  der(alfa1) = beta1
	  der(alfa2) = beta2
	  der(T) = beta1
	  alfa2 = mvap
	  Q = f(Time)
	end

set eliminate on
set LogDeriv on
enter model
  @./estado_liq_vap.dym
differentiate
variable known alfa1
variable known alfa2
variable unknown T
variable unknown mvap
partition
output solved equations
	set eliminate on
	set LogDeriv on
	enter model
	  @./estado_eq.dym
	differentiate
	variable known alfa1
	variable known alfa2
	variable unknown T
	variable unknown mvap
	partition
	output solved equations

Table 3. Equations added to the model by Dymola [7]

líq_phase or *vap_phase* mode	*equilibrium* mode
$\dot{\alpha}_1 = \dot{T}$ $\dot{m}_{vap} = 0$	$\dot{\alpha}_2 = \dot{m}_{vap}$ $\dot{T} = 0$

Table 4. Model equations describing the three modes

líq_phase or *vap_phase* or *equilibrium* mode
$0 = \text{if } equilib \text{ then } derm_{vap} - F_{cf} \text{ else if } liquid \text{ then } m_{liq} - m_0 \text{ else } m_{liq}$
$0 = \text{if } equilib \text{ then } m_{vap} + m_{liq} - m_0 \text{ else if } liquid \text{ then } m_{vap} \text{ else } m_{vap} - m_0$
$0 = \text{if } equilib \text{ then } T - T_{cf} \text{ else if } liquid \text{ then } derT - \frac{Q}{m_0 C_{liq}} \text{ else } derT - \frac{Q}{m_0 C_{vap}}$
$0 = \text{if } equilib \text{ then } F_{cf} - \frac{Q}{\lambda} \text{ else } F_{cf}$
$\frac{d\alpha_1}{dt} = \beta_1$
$0 = \text{if } equilib \text{ then } derT - \beta_1 \text{ else } \alpha_1 - T$
$\frac{d\alpha_2}{dt} = \beta_2$
$0 = \text{if } equilib \text{ then } \alpha_2 - m_{vap} \text{ else } derm_{vap} - \beta_2$
$0 = \text{if } equilib \text{ then } \frac{d\alpha_2}{dt} - derm_{vap} \text{ else } derT - \frac{d\alpha_1}{dt}$
$0 = \text{if } equilib \text{ then } derT \text{ else } derm_{vap}$

Table 5. Instantaneous equations

líq_phase or *vap_phase* or *equilibrium* mode
when *equilib* then $\text{init}(\alpha_1) = T$ endwhen when not *equilib* then $\text{init}(\alpha_2) = m_{vap}$ endwhen

Table 6. Model written in Dymola [7]. Commands to set the simulation initial conditions (below)

```
model liq_vap_eq
   local  T, Q, mvap, mliq, Fcf, liquid, equilib
   local dermvap, derT
   local alfa1,alfa2,beta1,beta2
   parameter  m0=100, lam=5, Cliq=0.2, Cvap=0.08, Tcf=373
   0 = if equilib                      ->
       then dermvap-Fcf                ->
       else if liquid then mliq-m0 else mliq
   0 = if equilib                      ->
       then mvap + mliq - m0           ->
       else  if liquid then mvap else mvap-m0
   0 = if equilib                      ->
       then T - Tcf                    ->
       else if liquid then derT-Q/(m0*Cliq) else derT-Q/(m0*Cvap)
   0 = if equilib                      ->
       then Fcf - Q / lam              ->
       else Fcf
   der(alfa1) = beta1
   der(alfa2) = beta2
   0 = if equilib                      ->
       then  derT - beta1              ->
       else  alfa1 - T
   0 = if equilib                      ->
       then alfa2 - mvap               ->
       else dermvap - beta2
   0 = if equilib                      ->
       then der(alfa2) - dermvap       ->
       else der(alfa1) - derT
   0 = if equilib                      ->
       then derT                       ->
       else dermvap
   Q = if Time < 28 then 20 else -20
   new(liquid) = Time < 10
   new(equilib) = Time >10 and Time < 20 or Time>30
   when equilib then
      init(alfa1) = T
   endwhen
   when not equilib then
      init(alfa2) = mvap
   endwhen
end
```

```
enter model
   @./liq_vap_eq.dym
partition
output solved equations
initial alfa1 = 300
initial alfa2 = 0
initial liquid    = True
initial equilib   = False
experiment stopTime = 40
simulate
```

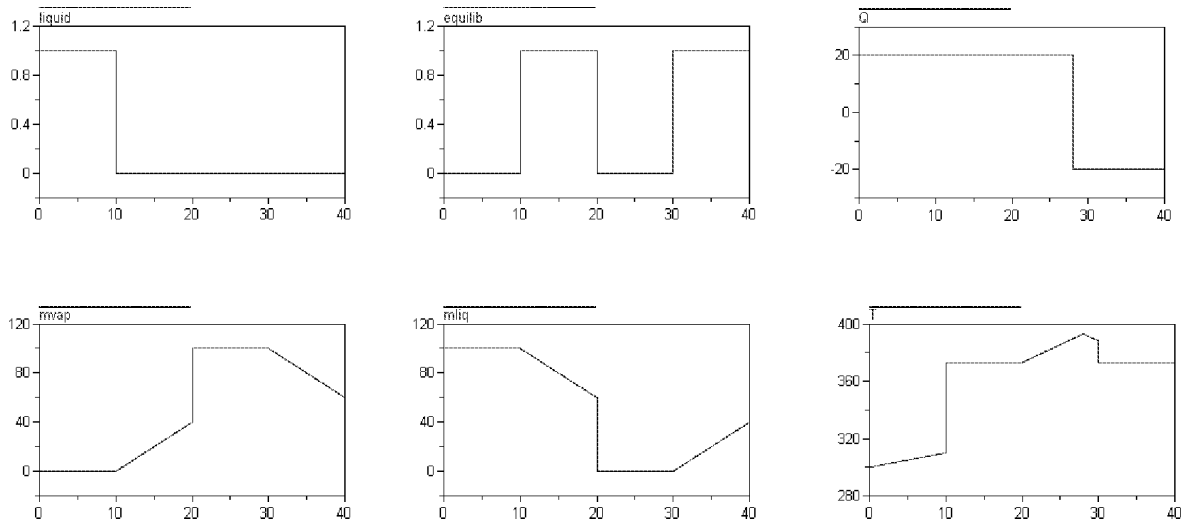

Figure 4. Simulation results

5. References

[1] Barton, P. I. 1992. The modelling and simulation of combined discrete/continuous processes. Ph.D. diss., Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, London.

[2] Nilsson, B. 1993. Object-oriented modeling of chemical processes. Ph.D. diss., Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[3] Andersson, M. 1994. Object-oriented modeling and simulation of hybrid systems. Ph.D. diss., Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[4] Brenan, K. E., S. L. Campbell, and L. R. Petzold. 1996. *Numerical solution of initial-value problems in differential-algebraic equations*. Philadelphia: SIAM.

[5] Barton, P. I. 1999. ABACUSS II. Retrieved from http://yoric.mit.edu/abacuss2/abacuss2.html

[6] Piela, P. C. 1989. ASCEND: An object-oriented environment for modeling and analysis. Ph.D. diss., Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

[7] Elmqvist, H., D. Bruck, and M. Otter. 1999. *Dymola: Dynamic modeling laboratory: User's manual*. Version 4.0. Lund, Sweden: Dynamic AB.

[8] EA International and ESA. 1999. *EcosimPro ver. 3.0: Getting started, users manual, modeling language (EL), modeling and simulation guide, and mathematical algorithms*. Madrid, Spain: EA International.

[9] Modelica Association. 1999. *Modelica: A unified object-oriented language for physical systems modeling: Tutorial, rationale and language specification*. Version 1.2. http://www.modelica.org.

[10] Elmqvist, H. 1978. A structured model language for large continuous systems. Ph.D. diss., Lund Institute of Technology, Lund, Sweden.

[11] Cellier, F. E. 1993. Integrated continuous-system modeling and simulation environments. In *CAD for control systems*, edited by D. Linkens, 1- 29. New York: Marcel Dekker.

[12] Pantelides, C. C. 1988. The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific Statistical Computing* 9 (2): 213-31.

[13] Mattsson, S. E., and G. Söderlind. 1992. A new technique for solving high-index differential equations using dummy derivatives. Paper presented at the IEEE Symposium on Computer-Aided Control System Design, March, Napa, CA.

[14] Elmqvist, H. 1993. Object-oriented modeling and automatic formula manipulation in Dymola. Paper presented at SIMS'93, Scandinavian Simulation Society, June, Kongsberg, Norway.

[15] Cellier, F., M. Otter, and H. Elmqvist. 1995. Bond graph modeling of variable structure systems. In *Proceedings of ICBGM'95, 2nd SCS International Conference on Bond Graph Modeling and Simulation*, Las Vegas, NV, pp. 49-55.

[16] Urquia, A. 2000. Modelado Orientado a Objetos y Simulación de Sistemas Híbridos en el Ámbito del Control de Procesos Químicos. Ph.D. diss., Departamento de Informática y Automática, UNED, Madrid, Spain.

Alfonso Urquia *is associate professor in the department of Computer Science and Automatic Control, ETS Ingeniería Informática, UNED, Madrid, Spain.*

Sebastian Dormido *is full professor in the department of Computer Science and Automatic Control, ETS Ingeniería Informática, UNED, Madrid, Spain.*