ECOSIMPRO CODE GENERATION FROM THE PHYSICAL ANALYSIS OF OBJECT-ORIENTED MODELS

Juan José Ramos González
JuanJose.Ramos@uab.es

Miquel Àngel Piera i Eroles
MiquelAngel.Piera@uab.es

Departamento de Telecomunicación e Ingeniería de Sistemas. Universidad Autónoma de Barcelona

Abstract

To cope with the growing demands for simulation models of ever increasing complex industrial systems, the research community effort has been mainly focused in creating different software tools which simplify the modeling process. This work describes how the Object-Oriented Modeling language PML automates the modeling process by using physical knowledge in order to set the mathematical model of the system. The modeling language introduces a new modularization of the physical knowledge required to model a system, making a clear separation between the physical behavior representation (declarative knowledge) and the computational aspects of model simulation (procedural knowledge).The mathematical model can be formulated by means of an equation based language like EL or Modelica.

Keywords: Physical knowledge representation, Object Oriented Modeling, Automated modeling .

1 INTRODUCTION

The modeling process can be defined in a wide sense as the activity of developing models. A model is a collection of information that describes the characteristics of a system in a specific situation which is of interest for a certain purpose. This definition establishes a ternary relation among the modeler, the system and the model. At the core of this relation is the modeler's experimentation interest, which somehow will address the model construction.

The experimentation interest will guide the way in that the modeler observes the system. The modeler focus his modeling interest in those particular issues of the system he wants experiment with. The experimentation interest is frequently stated in terms of the experimentation objectives [14], which can be defined as the set of system issues to which the modeler wants to find answers. The experimentation objectives serve to focus the model construction since they define the role of the model in the application domain (system design, control, management, etc). The formulation of the experimentation objectives and the phenomena of interest establishes the experimental framework.

The modeling activity is difficult and usually requires of expertise in the model application domain with a deep knowledge of the particular modeling and simulation tool used to develop and validate models. Many different experimental frameworks can be defined for a system, so different models can be built for the same system. Therefore, the modeling cost becomes of more relevance since the same system will require different models in order to answer the questions made by each modeler according to his experimentation objectives.

Different modeling and simulation environments have been developed during the last decades in order to reduce the modeling burden and its related cost. Two main aspects can be distinguished in every modeling environment: the *representation formalism* and the *modeling methodology*. The representation formalism will define the language used to describe the system behavior. The modeling methodology will define a set of rules specifying the way in that the physical behavior must be organized (or structured) in order to build the system model.

A third important aspect to be considered is the relationship between the system model (as it is specified according to the language and the methodology defined by the modeling environment) and the model formulation suitable for the simulation environment (simulator). We will name this relationship the *simulator relation*. This relationship states an agreement between the model and its solution engine which must be fulfilled in order to guarantee a correct generation of the experimentation results. Hence, the simulator becomes the fourth entity involved in a modeling and simulation project, together with the system, the experimental framework and the model.

To sum up these factors, we may say that the model construction process is conditioned by the experimental framework, the representation language, the modeling methodology and the simulator relation. This work describes how the task to build a model according these factors can be significantly simplified

by using physical knowledge which can be predefined in a modeling library. The benefits of using physical knowledge are discussed in [9] and [11]. The representation of this knowledge is described in [10] and [12].

The section 2 presents a possible systematization of the modeling process. The section 3 introduces how the modeling process is automated in PML. The section 4 illustrates through an example the generation of the mathematical model, formulated by means of the EcosimPro language. Some conclusion remarks are summarized in section 5.

2 THE MODELING PROCESS

The modeling process was roughly defined at the section 1 as the activity of building models. There are many works devoted to the systematization of the modeling process according to a particular methodology. See, for instance, object oriented modeling [13], object oriented modeling in chemical process [6] or bond-graph modeling [3] amongst others. Nevertheless a unified procedure to perform the model building process has not been described in the literature yet [5].

Even is not our aim to define such unifying procedure, we will give now a set of main objective factors which somehow stipulate the way a model is built. These factors are the experimental framework, the means provided by the modeling tool in order to represent the physical behavior and the adequacy of the model.

As it has been discussed at Section 1, the model development is performed within certain experimental framework and the built model should satisfy the experimentation purpose. Since different experimentation objectives will demand different system outcomes to be observed through the model, it can be expected that different models will be required, provided that an universal model is not a realistic option. Therefore, the first essential factor driving the modeling process is the experimental framework.

Intuitively, the model construction process, performed to contemplate the experimentation objectives, has an inherent notion of different representations of the system: we begin from a system representation which is successively manipulated and adapted according to the experimentation objectives. These representations of the system can be seen as different specifications of the modeled system describing the system knowledge at different levels (e.g. see [4],[6] and [14]). In order to characterize the modeling process, we will distinguish four layers to define the system specification:

Table 1: Levels of system specification.

Level	Name
0	Causal explanation
1	Non causal explanation
2	Phenomenological structure
3	Topological structure

Topological structure: This specification of a system describes its physical structure, i.e., which are its subsystems and how are they interconnected. This is a usual way to think about a system.

Phenomenological structure: This specification is more concerned with the phenomena occurring in the system and the physical interactions derived from the matter and energy transfer among its subsystems. The analysis of this structure is essential in the modeling process since at this layer the modeler should contemplate the experimental framework in order to define the scope of a system specification able to represent the expected behavior.

Non causal explanation: at this layer the phenomenological structure is formulated by means of mathematical equations. Is named non causal since the specification does not express any assumption on the physical causality, i.e., the formulation does not bother about which are the causes and which are the effects. For example, the formulation of a electrical resistor behavior with the equation $V=IR$ does no assume the current is the cause and the voltage drop is the effect, it is simply a equality relation among variables and parameters.

Causal explanation: this system specification must resolve the physical causality, which was no relevant at the previous layer. While the non causal explanation is the formulation of the expected behavior (what the system does in relation with the experimental frame), the causal explanation is the formulation able to explain how the system does what it is expected to do. Therefore, the causal mathematical formulation should have the computational structure suitable to the numerical solver. This means that the system specification at this layer should fulfill the *simulator relation*.

Each level provides relevant information about the system. In the traditional context the simulation community, the system specification is done at the causal explanation level by defining a program able to generate the required experimentation results. So in certain way, we may think in the modeling and simulation activity as a procedure where the modeler moves from the higher levels of system knowledge to the lower levels. The modeling tool should give to

the modeler the means to start the model construction at one of these layers. Hence, the second important factor to be considered in the modeling process is the development methodology defined by the modeling tool and the mechanisms provided to represent the system behavior since they will establish the layer at which the system specification has to be built. Depending on the modeling methodology, the system can be specified at a level which does not fulfill the *simulator relation*, i.e., the model can not be used straightforward to generate the experimentation results. In those cases, the modeling tool should provide with the means to move from the level were the model has been built into the level where the system data can be recreated (e.g. the object oriented modeling tools based on the equation formalism such EL [1] and Modelica [2] are able to move from the non causal level into the causal level).

The third factor to be considered in the modeling process is the adequacy of the model. The adequacy of a model can be considered as a measure established with respect the ability of a model to give a causal explanation for the phenomena of interest [7], but also it should be considered as a combination of validity and simplicity. We will understand by validity, or correctness, the capability of the model to generate, at least, the data collected from the system within the experimental framework, even though there are stronger forms of validity (see for instance [14]). Finally, simplicity is a desirable property for a model to be adequate. Since we have discarded the viability of universal models, we should consider simplicity to be always linked to the experimentation objectives and to the model application purpose. The simplicity property involves two aspects: the set of represented phenomena, which should be the minimum able to give a causal explanation for the phenomena of interest, and the accuracy or degree of detail used to formulate the phenomena. In other words, a simple model should not include irrelevant phenomena and should not make needlessly complex formulations of the relevant phenomena.

Now we will try to explain the modeling process according to these three factors: the experimental framework, the means provided by the modeling tool in order to define the system specification and the adequacy of the built model. A similar visualization of the modeling process is given in [5]. As Figure 1 shows, the modeling process can be seen as a trajectory in a three dimensional space spanned by the coordinates of specification, objectives and experimental framework and adequacy.

From this point of view, modeling can be seen as an incremental process where successive steps are taken towards the desired system specification.

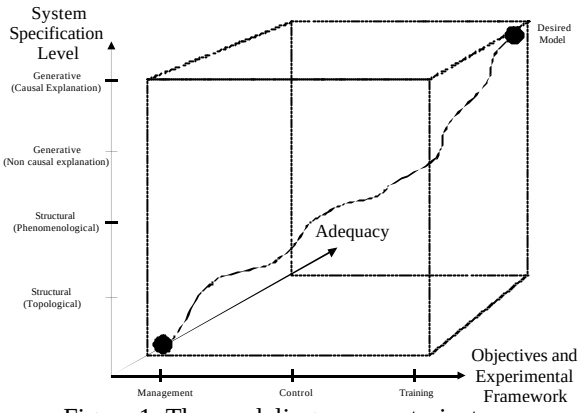

Figure 1: The modeling process trajectory.

By starting form scratch, a system specification is usually defined at the topological structure level looking at system as a set of components (subsystems) connected to perform certain task. According to the experimental framework, the phenomena of interest should be selected and the physical interactions have to be identified from the system topology. This specification has been defined as the phenomenological structure. In the next step, the phenomena of interest and the physical interactions can be formulated by means of the mathematical formalism. Finally, the mathematical specification should be manipulated in order to set the proper computational causality according to the simulator relation. Model adequacy should be present at every step, trying to find an agreement between the model validity and simplicity.

The starting point of the modeling trajectory will be heavily conditioned by the modeling tool. Both the methodology and the representation formalism supported by the modeling tool will introduce certain rules to define the system specification at one of the described levels.

2.1 MODELING AUTOMATION BY MEANS OF REUSABILITY

Automating the modeling process has been a common objective in many of the present modeling tools. A usual trend in many of them is to facilitate the modeling process supporting the reuse of predefined modeling components.

A *modeling component* can be defined as a basic model building block used to represent some physical behavior. A modeling component should also define, in an explicit or implicit manner, the interaction with other modeling components in order to be able to link them with some network structure to declare additional physical behavior.

A modeling component may be almost everything from a simple variable representing a physical property to a complex software structure representing

a process unit such as a chemical reactor. The set of definable modeling components configures the model development methodology since the way in that models are constructed is dependent on the nature of the modeling component.

We should consider that the reuse of a modeling component has two different facets: first, a *technical facet* which should guarantee to a great extent that, when two valid modeling components are properly reused to define a new modeling component, the resulting modeling component is also valid and is able to explain the aggregated physical knowledge; second, a *pragmatics facet* which should make easy to the modeler the selection of the appropriate modeling component. This second facet will depend on the capability of the modeling component to represent physical elements and concepts familiar to the modeler. From the model user's point of view, a modeling component representing a complex process unit provides more readable information about the represented physical behavior than, for instance, a set of equations and variables do.

Developing a modeling environment able to support the system specification at the topological structure level and the automation of the process needed to follow the modeling trajectory towards the causal explanation level, requires first to establish which is the scope of the reusability of a modeling component. We will consider two main aspects with respect to the reusability:

- **Aggregation of modeling components**: this is the basis for a structured modeling method. It should be possible to aggregate, or couple, modeling components in order to declare new physical behavior. A clear example is the aggregation of subsystem models to build a new model
- **Free reusing physical context**: the reuse of a modeling component should not be constrained by the physical context where it was defined. The physical context is usually determined by the experimental framework since it drives the way in that the modeler looks at the system when he defines the modeling component.

Let us assume that the specification can be performed at the topological structure level, so the model of a system can be set up by aggregating predefined modeling components representing to its subsystems.

We may consider that the physical reusing context of a modeling component can vary in two forms: one related with the physical causality of the modeling component as a part of a larger model; the other related with the experimentation objectives and the desired adequacy.

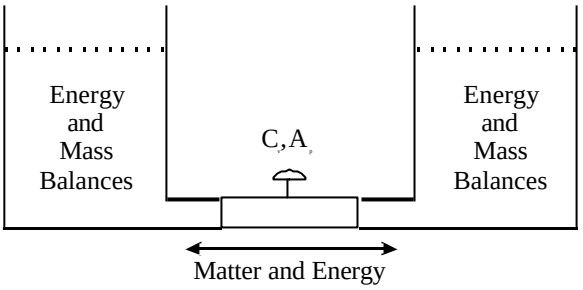

Figure 2: Two tank system exchanging matter through a valve.

The physical causality of a component in a system may vary without any variation on its behavior. A typical example is an electrical resistor connected to a voltage source or to a current source. Its behavior does not change in those situations, so it can be formulated by a non causal explanation such as $V=IR$ (Ohm's law) . However, when we need to make the causal explanation of the resistor, two different causal equations are required: $I:=V/R$ and $V:=IR$ respectively. Even we have made a change at the topological level (substitution of a voltage source by a current source), the phenomenological structure has not been changed. This problem is treated by some equation based modeling approaches, such as the equation-based object oriented modeling languages (e.g. [1] and [2]), which are able to manipulate the equations to assign the proper computational causality in order to set the causal explanation.

However, physical causality may affect to the set of phenomena which should be described by a component model depending on its reusing context. A very simple example can be illustrated with the valve model in the system at Figure 2. If the tank model just represents the matter mass balance, the valve model should only describe the matter transport. But, if the tank model also describes the matter internal energy, the valve model will have to describe the thermal interaction between both tanks. So, even the topological specification is the same, the phenomenological structure is different whether the tank model describes the energy balance or not. Looking at the valve model, its definition is driven by its reusing context. Therefore, if it is specified at the non causal explanation level (by means of an equation formalism), its reusing context becomes determined since at this level the phenomenological structure has been previously stated.

Finally, we should also consider that the variation of the experimental framework affects to the phenomena of interest. To consider this possibility is much more complicated since different phenomenological structures can be set according to the experimentation objectives, each of them representing the proper set of phenomena. Supporting this facility implies that the modeling tool has to be able to move automatically from the topological level into the

phenomenological level, giving rise to different structures according to the experimental framework, and afterwards into the non causal explanation level. Furthermore, the adequacy axis should be taken into account through this movement. This implies that the modeling tool should offer the possibility to select the phenomena of interest and their formulation with the desired degree of accuracy.

Object-Oriented modeling languages such as EL or Modelica are based on a non causal equation formalism. This means that a modeling component is defined at the non causal explanation level and, therefore, its reusing context is constrained to the phenomenological structure where it was conceived. As it has been illustrated by the previous example, a valve model conceived to connect two tanks representing a matter mass balance can not be reused to connect two tanks models which additionally describe an energy balance.

3 THE PML APPROACH

The PML (stands for Physical Modeling Language) language has been designed to support reusability by means of aggregation of the definable modeling components [10][11][12]. Their reuse context is free in the sense discussed above (physical causality, experimental framework and adequacy). The modeling process in PML is illustrated at Figure 3. The model of a system is specified at the topological level: models of system components are connected as they are connected in the system. The interaction between coupled models preserves also the analogy with the subsystem connections since it is defined in terms of energy or matter transfer. The specification at this level in PML is named as *Topological Model*.

This topology is analyzed from a physical point of view in order to derive the phenomenological structure. This structure represents the set of phenomena described at the topological model together with the physical interactions derived from the matter and energy transfers. The specification at this level in PML is named as *Functional Model*. This modular structure is not predefined in the PML modeling classes. It is the result of analyzing the physical causality of a model once its reusing context is determined at the topological model. For example, a valve PML model would describe a matter transport phenomena. If the valve model is connected to two tanks PML models describing matter and energy accumulation phenomena, the energy interaction is automatically derived from the matter transfer described at topological model. Note that the valve model is not predefining which physical interactions will be derived from its connection. It is important to remark that PML permits a this level to vary the experimental framework by neglecting phenomena which are not of interest for the experimentation objectives.

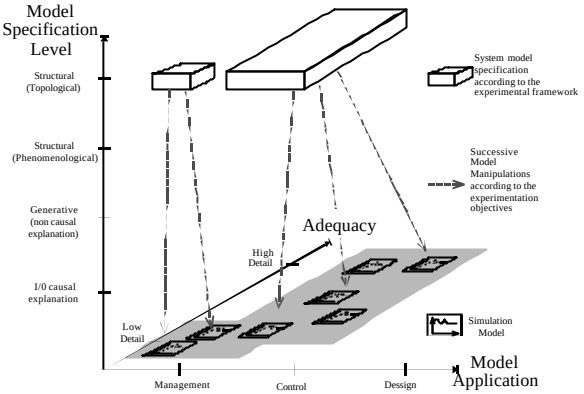

Figure 3: Modeling trajectories with PML

For example, we may neglect the energy accumulation phenomena described in the tank model if it is not of our interest. The obtained functional model will be the same as if the tank model would not had included such behavior description. It is important to note that there are not side effect on the valve model.

The next step of the modeling process consist in formulate the phenomenological structure by means of non causal equations. This step consist in making the proper instance of the laws which describe how the phenomena occur. It is important to remark that the adequacy of the model can be adapted to the desired level of accuracy by selecting the proper law formulation.

This specification is named as *non causal explanation* since , even the physical causality has been derived, it can not be decided which are the *causes* and which the *effects* at each physical interaction. For example, it is not predefined if a pressure drop causes the mass flow through the valve or if a matter flow (e.g. forced by a pump) provokes a pressure drop in the valve. This matter is left to languages such as EL or Modelica since they can resolve the computational structure suitable for the simulation engine. This final level is named as *I/O causal explanation* and match the simulator relation.

To support this modeling process, a new representation formalism has been designed for PML. The formalism defined by PML offers a classification method supporting the *physical knowledge organization* according to physical concepts totally decoupled from the mathematical foundation required for the physical behavior computation. The physical knowledge is organized around five main modeling components (classes): *entities, phenomena, laws, models* and *ports*. Next section will illustrate the use of these modeling components by means of an example. With respect to the reusability pragmatics,

the language semantics is very closed to the user's physical understanding of the system, which could lead to the improvement of the assistance capabilities provided to both the model developer and the model user. The model reusability can be extended by supporting a computer-aided modeling process to cope with different simulation or experimentation purposes.

4 GENERATION OF ECOSIMPRO CODE

The system shown at Figure 2 will be used to illustrate the modeling process in PML. Every modeling components has to be defined in terms of one of the PML classes: entity, phenomenon, law, model and port (a more detailed language description can be found in [12]).

The *Entity* class represents some type of matter or energy defining its properties. We can define the following class or the two tank system:

```
(entity [matter]
 [properties({mass,
    massFlow,pressureDrop,temperature,
    Enthalpy,enthalpy,Cp,volume,density})]);
```

We can now declare the phenomena occurring in the system. It will suffice the energy accumulation and the matter accumulation and transport phenomena for the example purposes. The phenomena classes express the physical behavior occurring in a system:

```
(phenomenon [store(matter)]
 [massBalance(matter)->mass(matter)]
);
(phenomenon [storeInternalEnergy(matter)]
 [enthalpyBalance(matter)->
  {Enthalpy(matter),enthalpy(matter)}]);
(phenomenon [transport(matter)]
   [{matterTranport(matter)->
      {massFlow(matter),pressureDrop(matter)}
     linearMatterTransport(matter)->
      {massFlow(matter),pressureDrop(matter)}
   }]
);
```

The *Phenomenon* class basically enunciates the behavior affecting an entity in the system, and links its description in to a law or set of laws. The last can be used to declare several formulations with, for instance, different degrees of accuracy.
The *Law* class describes how a phenomenon occurs representing the law which rules it. The law class is close to the computational aspects of the represented behavior since it is used to obtain the behavior mathematical formulation. The laws referenced by the above phenomena can be declared as follows:

```
(law [massBalance(matter)]
  mass(matter) =
  intgr(sum(portInstances(massFlow(matter))))
]);
(law [matterTransport(matter)]
 [{DATA(Rfluid = 1.0),
   massFlow(matter) =
   prod({Rfluid,sqrt(pressureDrop(matter))}),
   sum(portInstances(massFlow(matter)))=0.0
  }]);
(law [enthalpyBalance(matter)]
  {der(Enthalpy(matter)) =
     sum(portInstances({matter,
         prod({massFlow(matter),
              enthalpy(matter)})})),
   Enthalpy(matter)=prod({mass(matter),
                          enthalpy(matter)}),
   enthalpy(matter) = prod({Cp(matter),
                       temperature(matter)}),
   PHENOMENON(store(matter))
}]);
```

The *Port* class describes the physical interactions between physical devices in terms of an entity exchange. Its declaration must include, at least, a pair of properties representing the causality in the exchange phenomenon. We declare the following port to describe the matter exchange in the two tanks system:

```
(port [matterPort(matter)]
      [{massFlow,pressure}]);
```

The *Model* class is used to represent some physical device or a part of it. The model class aggregates entity classes to specify the involved entities, phenomenon classes to declare the device behavior, port classes to specify the device interactions and, likely, model classes themselves to support the structured modeling process. We can declare the following model class for our example:

```
(model [duct][{
  phenomena(transport(matter)),
  entities(matter),
  ports({matterPort(P1(matter)),
         matterPort(P2(matter))}),
  equations({
      DATA({ductVolume=5.0}),
      pressureDrop(matter) =
         sum({P1(pressure(matter)),-
              P2(pressure(matter))}),
      massFlow(matter)=P1(massFlow(matter)),
      volume(matter) = ductVolume
     })
 }]);
(model [tank][{
  entities({matter}),
  ports({matterPort(P1(matter)),
         matterPort(P2(matter))}),
  phenomena(store(matter)),
  equations({
   DATA({section=1.0,pTop=1.0,g=9.8}),
   mass(matter)=prod({volume(matter),
                      density(matter)}),
   volume(matter)=prod({section,level}),
   P2(pressure(matter))=sum({
      pTop,prod({density(matter),g,level})}),
   P1(pressure(matter))=pTop})
  }]
);

(model [twoTanks]
  [{submodels({duct(D),tank({T1,T2})}),
    connections({T1.P2-D.P1,D.P2-T2.P2})
  }]
);
```

As it can be observed, the system at Figure 2 is specified at the topological level by means of the `twoTanks` PML model class. The phenomenological structure is derived by analyzing the physical interactions of the coupled submodels. The tank model requires the knowledge of the matter specific enthalpy at its ports. (see the sentence *portInstance* at the `enthalpyBalance` class code). The matter path is followed through the model topology to determine who is affecting that property of the flowing matter. Note that in the valve there is not declared a phenomenon affecting to the matter specific enthalpy whereas the storeInternalEnergy phenomenon declares to affect this property in the tank model. This the reason why it is deduced a thermal interaction between both tanks. With respect the matter transfer, there is an interaction among the tank T1, the duct D and the tank T2, since the duct declares a phenomenon determining the physical relationship between the mass flow and the pressure drop at the duct ends. The phenomenological structure is represented at the functional model (see Figure 4). The next step is to formulate the non causal model. This is performed by translating the PML law classes into EcosimPro code and by defining the proper terminal variables (EcosimPro ports) according to the interactions represented in the functional model. The result is shown in Appendix A.

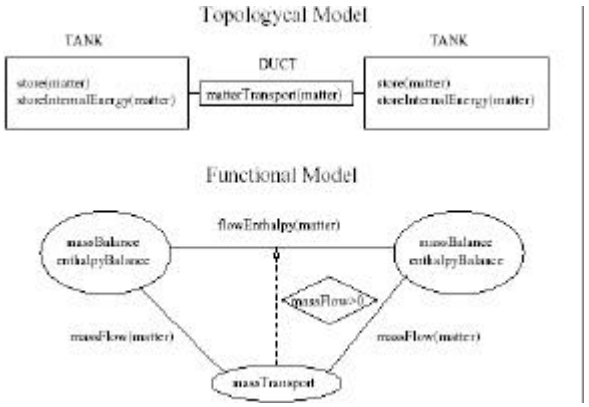

Figure 4: Topological and functional model of the two tank system

5 CONCLUSIONS

This work has illustrated how the PML environment can help to the modeler to build models with appropriate modeling rules, which provide information on the selection of the modeling components (component and interaction description mechanisms). The defined modeling process leads to a significant reduction of the model construction cost, while enhancing the quality of the models in terms of adequacy, consistency and robustness. PML extends the classical *model* reusability supported by the equation based approaches to the new modeling components such as *entities*, *phenomena* or *laws*.

Acknowledgments

Financial support of this work by the CICYT (TAP98-0364) is gratefully acknowledged.

References

[1] Cobas, P. and Al. (1999). EcosimPro Simulation Language. Empresarios Agrupados. http://www.ecosimpro.com.

[2] Elmquist, H. and Al. (1999). Modelica - A Unified Object-Oriented Language for Physical System Modeling. Tutorial and Rationale. Modelica Design Group.

[3] Karnopp, D. and R.C. Rosenberg (1968). Analysis and Simulation of Multiport Systems. The Bond Graph approach to Physical System Dynamics. The M.I.T. Press.

[4] Klir, G.J. (1985). Architecture of System Problem Solving. Plenum Press. New York.

[5] Lohmann, B. and W. Marquardt (1996) "On the systematization of the process of model development". *Computers Chemical Engineering* V. 20S, pp. S213-S218.

[6] Marquardt, W. (1991). "Dynamic process simulation - recent progress and future challenges". *Chemical Process Control (CPC-IV)*. pp.131-180.

[7] Nayak,P. (1995). Automated Modeling of Physical Systems. Ph.d-thesis. NASA Ames Research Center. Moffet Field, CA (USA).

[8] Nilsson, B. (1993). Object-Oriented Modeling of Chemical Processes. Ph.d-thesis. Department of Automatic Control. Lund Institute of Technology. Lund, Sweden.

[9] Ramos, J.J., M.A. Piera and I.Serra (1998*a*). "The use of physical knowledge to guide formula manipulation in system modelling". *Simulation Practice and Theory*. V6(3), pp.243-254.

[10] Ramos, J.J., Piera M.A and I.Serra (1998*b*). "A free context physics knowledge representation suitable for object-oriented modeling tools". *Eurosim'98*. Helsinki, Finland.

[11] Ramos, J.J. and M.A. Piera (1999). "Need of O-O languages for Physics Knowledge representation in the Simulation field".*Technology of Object-Oriented Languages and Systems (TOOLS 29)*. pp.162-171.

[12] Ramos, J.J., Piera M.A (2001). "PML: an Object-Oriented Modelling Language for Physics Knowledge Representation". *Eurosim 2001*. Delft, The Netherlands.

[13] Rumbaugh, J., M.Blaha, W.Premernali, F.Eddy and W.Lorensen (1991). Object Oriented Modeling and Design. Prentice Hall.

[14] Zeigler, B.P., H.Praehofer and T.G.Kim (2000). Theory of Modeling and Simulation. Academic Press.

APPENDIX A. EcosimPro code

```
PORT C4MatterPort
  SUM REAL C4Matter_massFlow
  EQUAL REAL C4Matter_pressure
END PORT

PORT matterFlowingProperties
       EQUAL REAL matter_enthalpyAUX
       EQUAL REAL matter_enthalpy
END PORT


PORT directionInformation SINGLE IN
  EQUAL REAL C4Matter_massFlow
END PORT

COMPONENT matterFlowEdge
   PORTS
       IN matterFlowingProperties PA
       OUT matterFlowingProperties PB
       IN directionInformation PI
   DECLS
       REAL Hq
       REAL alpha
   CONTINUOUS
       alpha = ZONE(PI.C4Matter_massFlow>0
TOL 1e-2) 1
                OTHERS 0
       Hq = alpha*PA.matter_enthalpyAUX+(1-
alpha)*PB.matter_enthalpyAUX
       PB.matter_enthalpy = Hq
       PA.matter_enthalpy = Hq
END COMPONENT

COMPONENT   C4Duct --IS_A C4::duct
  PORTS
     IN  C4MatterPort P1 -- On Entity:
C4::matter
     IN  C4MatterPort P2 -- On Entity:
C4::matter
     OUT directionInformation dI -- From
functional model

  DATA
  -- Data declared by laws
     REAL Rfluid=1.0
  -- Data declared by local equations
     REAL ductVolume=5.0
  DECLS
  -- Entity properties referenced by laws
     REAL C4Matter_massFlow
  -- Entity properties referenced by local
equations
     REAL C4Matter_pressureDrop
     REAL C4Matter_volume
  CONTINUOUS
  -- PHENOMENON: C4::transport(matter)
  --        LAW: C4::matterTransport(matter)

C4Matter_massFlow=Rfluid*ssqrt(C4Matter_press
ureDrop)
     C4Matter_massFlow=P1.C4Matter_massFlow

P1.C4Matter_massFlow+P2.C4Matter_massFlow=0.0
  -- Local model equations
     C4Matter_pressureDrop =
P1.C4Matter_pressure-P2.C4Matter_pressure
     C4Matter_volume=ductVolume
  -- From functional model
     dI.C4Matter_massFlow = C4Matter_massFlow
END COMPONENT
COMPONENT   C4Tank --IS_A C4::tank
  PORTS
     --On Entity: C4::matter
     IN C4MatterPort P1
     IN C4MatterPort P2
     -- From functional model
     IN matterFlowingProperties FP1
     IN matterFlowingProperties
  DATA
  -- Data in law formulations
     REAL C4Matter_density = 1.0
     REAL C4Matter_Cp = 1.0
     REAL C4Matter_Tref = 0.0
  -- Data declared by local equations
     REAL section=0.1
     REAL pTop=1.0
     REAL g=9.8
  DECLS
  -- Variables declared by laws
     REAL C4Matter_Enthalpy
     REAL C4Matter_enthalpy
     REAL C4Matter_temperature
  -- Variables declared by local equations
     REAL level
  -- Entity properties referenced by local
equations
     REAL C4Matter_mass
     REAL C4Matter_volume
  CONTINUOUS
  -- PHENOMENON: C4::store(matter)
  --        LAW: C4::massBalance(matter)

  C4Matter_mass'=
  P1.C4Matter_massFlow+P2.C4Matter_massFlow
  -- PHENOMENON:
  --   C4::storeInternalEnergy(matter)
  -- LAW: C4::enthalpyBalance(matter)
  C4Matter_Enthalpy'=
   P1.C4Matter_massFlow*FP1.matter_enthalpy+
   P2.C4Matter_massFlow*FP2.matter_enthalpy
  C4Matter_Enthalpy=
   C4Matter_density*C4Matter_volume*
   C4Matter_enthalpy
  C4Matter_enthalpy=
   C4Matter_Cp*(C4Matter_temperature-
   C4Matter_Tref)
  -- Local model equations
  P1.C4Matter_pressure=pTop
  P2.C4Matter_pressure=pTop+
   g*level*C4Matter_density
 C4Matter_mass=
   C4Matter_volume*C4Matter_density
 C4Matter_volume=section*level
  -- Local model equations adapted from
functional model
 FP1.matter_enthalpyAUX = C4Matter_enthalpy
 FP1.matter_enthalpyAUX = C4Matter_enthalpy
 FP2.matter_enthalpyAUX = C4Matter_enthalpy
 FP2.matter_enthalpyAUX = C4Matter_enthalpy
END COMPONENT

COMPONENT C4TwoTanks
  TOPOLOGY
     C4Tank T1,T2
     C4Duct D
     matterFlowEdge edge

     -- matter exchange connections
     CONNECT T1.P2 TO D.P1
     CONNECT D.P2 TO T2.P2
     -- matter properties connections from
functional model
     CONNECT T1.FP2 TO edge.PA
     CONNECT T2.FP2 TO edge.PB
     CONNECT D.dI TO edge.PI
END COMPONENT
```