Object-oriented Modeling in the Service of Medicine

François E. Cellier[1], Àngela Nebot[2]

（[1*]Institute of Computational Science, ETH Zurich, CH-8092 Zurich, Switzerland;
[2**]Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona 08034, Spain）

Abstract：The models employed in all aspects of science and engineering have grown increasingly complex, in order to allow them to capture more and more details of the systems that they are representing. Unfortunately, the increased complexity makes these models more difficult to maintain and harder to understand. Especially in a domain, such as medicine, with domain experts, who possess rather limited mathematical skills in general, the complexity of advanced models has become problematic, as these experts no longer are capable of fully understanding and criticizing these models. The object-oriented modeling paradigm offers a means to increase the overall model complexity, thereby enhancing the realism of its simulations, without making these models more difficult to understand or maintain than the simple toy models of the past. In this paper, the approach is demonstrated by means of a fairly elaborate model of human hemodynamics.

Keywords：bond graph modeling; object-oriented modeling; graphical modeling; cardiovascular system; hemodynamics

1. Introduction

Models of human hemodynamics have been around for several decades already. A very early model able of reproducing the human heart beat was the PHYSBE benchmark problem[1]. The model was very simple, and had primarily been designed as a tool to demonstrate the then fairly new area of computer simulation.

Over the years, more realistic and useful models of human hemodynamics were designed. All of these models are essentially mechanical models based on the principles of hydrodynamics. Blood is a liquid, similar to water, and blood vessels resemble pipes, through which the liquid is flowing. The four heart chambers act like hydro-mechanical pumps that circulate the blood through the blood vessels. There are biological valves built into the cardiovascular system that prevent blood from flowing backward. They act just like the mechanical valves that prevent the water inside your house from flowing back into the city line. All of these models are based on the pioneering work by Beneken, Rideout, and others[2,3]. Some of these models make use of lumped parameter approximations[4], whereas others operate on compartmentalized distributed parameter representations[5].

The model that this presentation is based on was developed by Vallverdú in her Ph.D. dissertation[6]. This model was chosen simply, because it is a reasonably complex model, to which we were given full access. The model is summarized in Appendix A of an earlier paper written by the same authors[7].

The model had originally been coded in ACSL[8], a simulation language that allows writing down differential equations in a “natural” form, but is not fully object-oriented. Consequently, the model was hard to read and maintain, even for someone well versed in modeling. The program is fairly monolithic and rightfully qualifies as 2382 lines of “spaghetti code.”

The model was meanwhile recoded in Dymola[9], an object-oriented graphical modeling environment making use of bond graphs[10,11]. The new code is described in this paper.

2. The Dymola/Modelica Object-oriented Physical Systems Modeling Environment

Dymola is an object-oriented modeling methodology developed by Elmqvist in his Ph.D. dissertation[12]. Contrary to the original Dymola definition, the modern Dymola software offers a graphical user interface[9]. In fact, Dymola is today identified with the user interface, rather than with the underlying alphanumerical description language, which is now called Modelica. Modelica in contrast represents a further development of the original Dymola definition that had been proposed by

Elmqvist in his dissertation under the name Dymola.

In modern Dymola, each model (object) is represented by four different layers, an *icon layer* that depicts the object graphically, a *diagram layer* that visualizes graphically how models are being composed from sub-models, an *equation layer* that enables the user to formulate relationships between variables alpha-numerically in the form of equations or tables, and a *documentation layer* that can be used to describe the purpose and limitations of the model verbally and provide pointers to the open literature that may offer more information about the model.

Some models may be described by equations only, in which case the diagram layer is empty. Others may be graphically composed from sub-models entirely, in which case the equation layer is not being used.

Graphical models are more readable than equation models for several reasons. First, topological interconnections are naturally two-dimensional, whereas textual interconnections are necessarily one-dimensional. The human eye is better geared to understanding two-dimensional relationships. Second, graphical models ought to fit on a screen. Therefore, modelers will struggle to decompose models further, until each model fits conveniently on a single screen. Consequently, the “spaghetti code” danger is somewhat mitigated by the physical limitations of the screen. This physical constraint thus offers a natural incentive to keeping the local complexity of a model limited.

Although graphical modeling is thus preferable, all models must, at the bottom layer, be expressed by equations. Decomposing complex graphical models into simpler ones only delays the need for writing equations; it does not save us from having to do so in the end.

If we model in an application area that is well standardized, such as electronic circuits, we may be able to create models without ever having to write down a single equation. However, this is only possible, because someone else already composed the basic models for us and deposited them in a library of circuit components.

Unfortunately, medicine is not well standardized at all. There has not been defined a generally accepted standard for creating models in medicine, and thus, we may not get around having to write basic models of our own.

In this paper, we shall make use of a generic graphical modeling technique, called bond graphs, that is application area independent, and therefore allows us to express even the most basic components of cardiovascular models graphically. Unfortunately, this modeling methodology will be unknown to most medical domain experts, and therefore, it needs to be introduced next.

3. Bond Graph Physical Systems Modeling

A bond graph is a graphical representation of power flowing through a physical system. A bond is depicted as a harpoon (half arrow):

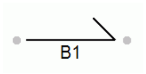


***Fig.** 1 The Bond*

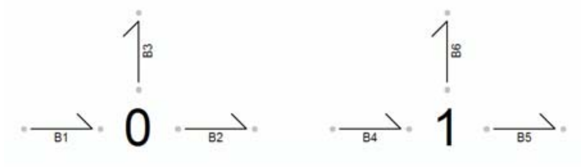
representing power flowing from one point to another. In every physical system, power, P, can be represented as the product of two adjugate variables: an effort, e, and a flow, f:

$$P = e \cdot f \tag{1}$$

The hook always points to the left in the direction of the harpoon, and if the two variables are written next to the bond instead of its name, then the effort variable is written on the (left-hand) side with the hook, whereas the flow variable is written on the opposite (right-hand) side of the harpoon.

In hydrology, it has become customary to define the pressure, p, as the effort variable, and the volumetric flow rate, q, as the flow variable, as pressure times volumetric flow rate equals power.

When a power flow splits, or when two power flows join, this can happen in two natural ways: either the effort is the same all around the junction and the flows into the junction add up to zero, or vice-versa, the flows are identical all around the junction, and the efforts add up to zero. Both types of junctions are common in physics, and consequently, both of them have found graphical representations in the bond graph methodology:



$$e_1 = e_2 = e_3 \qquad f_4 = f_5 = f_6$$
$$f_1 - f_2 - f_3 = 0 \qquad e_4 - e_5 - e_6 = 0$$

Fig. 2 Bondgraphic junctions

The resistance to flow due to friction can be written as:

$$q = G \cdot \Delta p \tag{2}$$

in the case of a laminar flow, and as:

$$q = k \cdot \mathrm{sign}(\Delta p) \cdot |\Delta p|^{1/2} \tag{3}$$

in the case of a turbulent flow, i.e., a flow restricted by a narrow orifice, such as a valve. In bond graph notation:

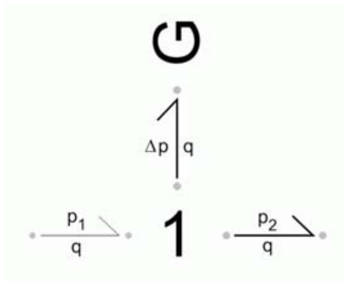


Fig. 3 *Hydraulic conductance*

The hydraulic conductance element, G, defines a static relationship between effort and flow:

$$f = G(e) \tag{4}$$

Since the effort (pressure) is always a pressure gradient (pressure difference), the hydraulic conductor is always attached to a 1-junction.

It is always possible to replace the conductance by a resistance, **R**:

$$e = R(f) \tag{5}$$

defining the inverse relationship. In the bond graph methodology, it doesn't really matter, which of the two elements is being used, since bond graphs are *per se* a-causal.

The compressibility of a liquid in a container is computed as:

$$\text{der}(p) = c \cdot \Sigma(\, q_{in} - q_{out}\,) \tag{6}$$

The change in pressure is proportional to the difference between inflows and outflows. The der(.) operator denotes a time derivative in Modelica. In bond graph terminology, this phenomenon is captured by a capacitive storage element, **C**:

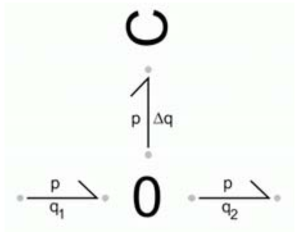


Fig. 4 *Hydraulic capacitance*

Since the change in pressure depends on a sum of flows, the hydraulic capacitor is usually attached to a 0-junction.

The inertance of a liquid to a change of its velocity is computed as:

$$\text{der}(q) = c \cdot \Delta p \tag{7}$$

The change in volumetric flow rate is proportional to the difference between the pressure at the input and that at the output. In bond graph terminology, this phenomenon is captured by an inductive storage element, **I**:

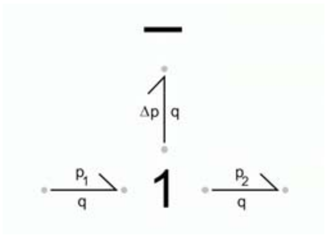


Fig. 5 *Hydraulic inductance*

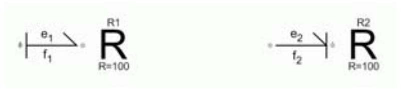
Since the change in flow rate depends on a difference between pressure values, the hydraulic inductor is always attached to a 1-junction.

4. Computational Causality in Bond Graphs

As had been mentioned above, bond graphs are *per se* a-causal. Yet, when generating a code that can be simulated, we need to decide for each equation, what variable to solve for.

Since a bond carries two variables *e* and *f*, two equations will be needed for computing their values. It turns out that one of the two variables is computed at each end of the bond. It has become customary to mark the side where the flow is being evaluated by a vertical bar, the so-called causality stroke in bond graph terminology. In doing so, the formerly a-causal bond graph has now become causal.

Figure 6 illustrates the concept.



$e_1 = R_1 \cdot f_1$ $\qquad$ $f_2 = e_2 / R_2$

Fig. 6 Causality strokes in bond graphs

In Dymola, it would not be necessary to use causal bonds ever, since the software is perfectly capable of determining the computational causality of all equations on its own. Yet, we recommend using causal bonds whenever possible, as this increases the readability (understandability) of the resulting bond graphs.

We are now ready to start modeling the hemodynamics of the human body.

5. Human Hemodynamics

5.1 Transport Models

Let us begin with the transport models that describe how the blood is being moved through the vessels. All transport models contain at least laminar friction. The more important vessels contain an inertance element in addition. For example, the blood flow from the ascending aorta to the aortic arch has been modeled as follows:

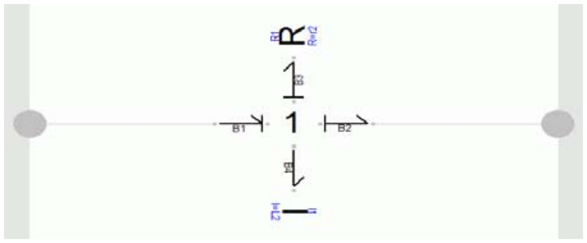


Fig. 7 *Bond graph of ascending aorta flow*

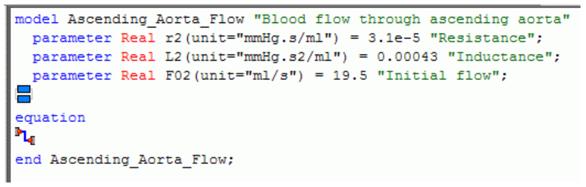
The two large grey dots to the left and right of the bond graph are the external connectors through which the model is connected to its environment.

The ascending aorta flow is represented by the following icon:

Fig. 8 *Icon of ascending aorta flow*

All transporter models were represented by means of arrows. Veins are depicted as blue arrows, whereas arteries are shown as red arrows, except for the pulmonary arteries and veins that use the inverse convention.

This model is represented through graphics exclusively. The equation window only contains definitions of the parameters:



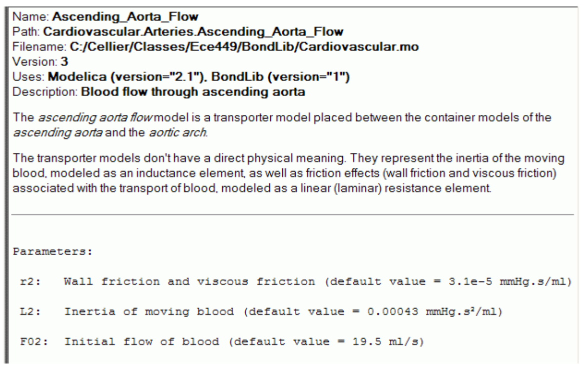
```
model Ascending_Aorta_Flow "Blood flow through ascending aorta"
  parameter Real r2(unit="mmHg.s/ml") = 3.1e-5 "Resistance";
  parameter Real L2(unit="mmHg.s2/ml") = 0.00043 "Inductance";
  parameter Real F02(unit="ml/s") = 19.5 "Initial flow";

equation

end Ascending_Aorta_Flow;
```

Fig. 9 *Parameters of ascending aorta flow*

Finally, the documentation window contains the following information:



Name: **Ascending_Aorta_Flow**
Path: **Cardiovascular.Arteries.Ascending_Aorta_Flow**
Filename: **C:/Cellier/Classes/Ece449/BondLib/Cardiovascular.mo**
Version: **3**
Uses: **Modelica (version="2.1"). BondLib (version="1")**
Description: **Blood flow through ascending aorta**

The *ascending aorta flow* model is a transporter model placed between the container models of the *ascending aorta* and the *aortic arch*.

The transporter models don't have a direct physical meaning. They represent the inertia of the moving blood, modeled as an inductance element, as well as friction effects (wall friction and viscous friction) associated with the transport of blood, modeled as a linear (laminar) resistance element.

```
Parameters:

 r2:   Wall friction and viscous friction (default value = 3.1e-5 mmHg.s/ml)

 L2:   Inertia of moving blood (default value = 0.00043 mmHg.s²/ml)

 F02:  Initial flow of blood (default value = 19.5 ml/s)
```

Fig. 10 *Documentation of ascending aorta flow*

The documentation is coded in HTML.

5.2 Container Models

Let us now consider the models representing the storage of blood in a container. As an example, let us look at the model of the aortic arch.

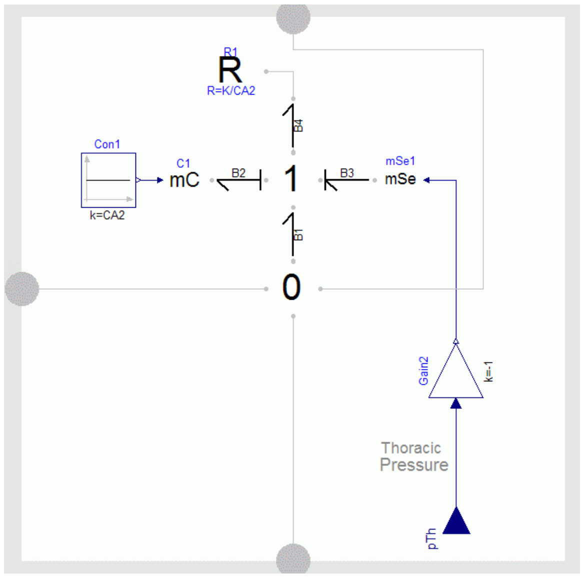


Fig. 11 *Bond graph of the aortic arch*

This model requires a bit more of an explanation. We would have expected to find a 0-junction with bonds to all external connector as well as a capacitor attached to it through a bond.

First, the bonds between the 0-junction and the external connectors are missing. The reason for this choice is simple. In a bond graph, bonds and junctions always toggle. We chose to make all transporter models end in bonds and all container models end in junctions, so that the transporter models can be connected to the container models directly at the next higher level of the modeling hierarchy without requiring either bonds or junctions in between them.

We could have used a linear capacitor model, C, as the capacitance in the aortic arch model is indeed assumed linear. However, we chose to use a non-linear (modulated) capacitance model, mC, instead. The reason is once again quite simple. The linear capacitance model implements the equation:

$$C \cdot \mathrm{der}(p) = \Sigma\,(q_{in} - q_{out}) = q \qquad (8)$$

By integration, we find that:

$$C \cdot p = \int q\, dt = V \qquad (9)$$

Thus, we can replace the original capacitance model by:

$$\mathrm{der}(V) = \Sigma\,(q_{in} - q_{out}) \qquad (10a)$$

$$p = V / C \qquad (10b)$$

which allows to specify an initial condition for the volume, *V*, rather than for the pressure, *p*. In the case of a non-linear capacitor, this is anyway the more correct model. The block diagram element to the left of the capacitor generates the modulation signal that computes the capacitance value. In the aortic arch model, the capacitance value is constant.

Although the blood flow is no longer turbulent at the aortic arch, it was decided to add a laminar resistance element as well.

The **mSe** element represents a modulated effort source, i.e., an externally computed pressure source. This element models the effect of breathing. As the lungs expand, they occupy more space in the thorax, thereby exerting additional pressure on the heart and the blood vessels in that region of the body. The thoracic pressure is calculated centrally and is distributed to all models that require it.

Figure 12 shows the icon that represents the aortic arch:

Aortic
Arch

Fig. 12 *Icon of the aortic arch*

All container models are drawn as boxes. The aortic arch receives arterial blood from the left through the ascending aorta. In the aortic arch, the blood stream splits. One part moves up to the head and the brain, whereas the other part moves down through the thoracic aorta.

5.3 Valve Models

Some of the vessels are more complicated, because they contain valves. To illustrate this concept, let us discuss the model describing the aortic valve.

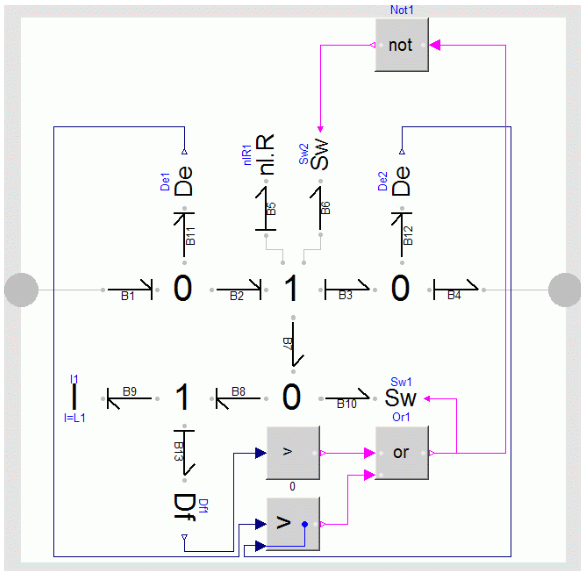


Fig. 13 *Bond graph of the aortic valve*

The effort detector model, **De**, measures the pressure (effort) at the 0-junction, to which it is attached. The effort detector never consumes any flow, i.e., it measures the pressure without consuming power.

The flow detector model, **Df**, measures the volumetric flow at the 1-junction, to which it is attached. It does so, without consuming any power, i.e. the effort into this model is always zero.

The valve is open, when either the pressure at the input is larger than the pressure at the output, or when there is currently a positive flow through the valve.

The bond-graphic detector models generate real-valued (blue) signals that are used by comparator blocks, which in turn generate logical (pink) signals. The logical signal coming out of the OR block is true, when the valve is open.

The bond-graphic switch element, **Sw**, behaves like a source of zero flow, when the modulating logical input signal is true, and it behaves like a source of zero effort otherwise.

When the valve is open, the flow through bond B_{10} is zero. Since that bond is attached to a 0-junction, the switch is invisible, i.e., it has no effect on the flow through that junction. At the same time, the effort in bond B_6 is also zero. Since that bond is attached to a 1-junction, also this switch is invisible.

In contrast, when the valve is closed, the effort at bond B_{10} is zero. Hence there is zero effort everywhere around the 0-junction. Consequently, the inertance model sees zero effort, and therefore holds its flow at a constant value. At the same time, the flow through bond B_6 is also zero. Since that bond is attached to a 1-junction, there is no flow through that junction at all.

The valves are transporter models with inertance and resistance elements. Since the flow through the aortic valve is turbulent, a non-linear resistor model, **nl.R**, is being used.

5.4 Heart Chamber Models

The heart chambers are container models.

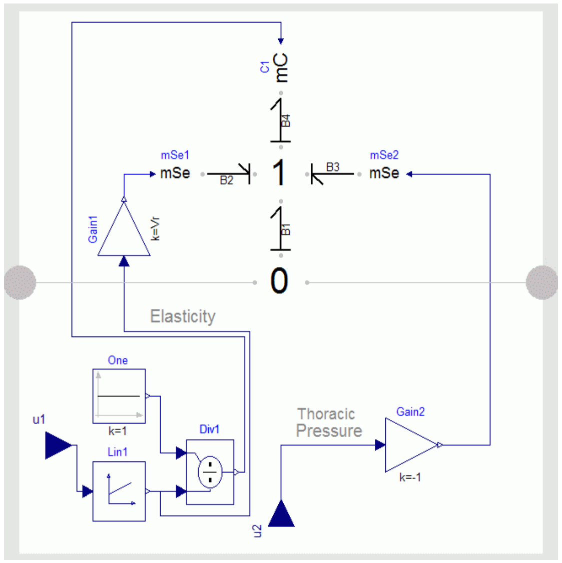


Fig. 14 *Bond graph of a heart chamber*

However, these container models are modulated by two separate pressure sources, one representing the thoracic pressure (breathing), the other representing the contraction of the chamber (heart beat). Also, the capacitance of the heart chamber varies as a function of the degree of contraction.

The heart chamber is represented by the icon of Fig. 15.

Heart Chamber

Fig. 15 *Icon representing a heart chamber*

The left ventriculum is one of four heart chambers. Its model is shown in Fig.16.

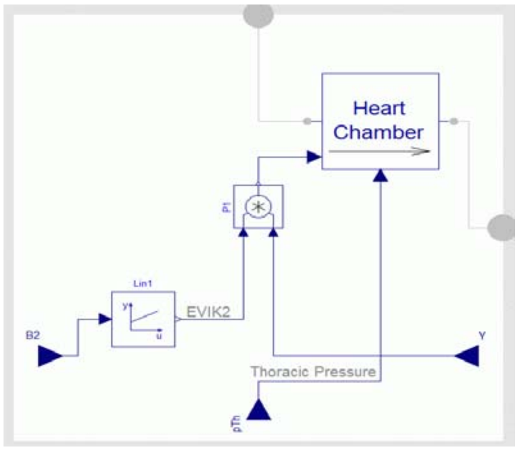


Fig. 16 *Model of left ventriculum*

The left ventriculum *is a* heart chamber. It invokes the generic heart chamber model, which, at the higher hierarchical level, is represented through its icon. The modulating signal determining the heart beat is a function of two variables: variable Y determines, when the chamber contracts, whereas B_2, representing the myocardiac contractility controller signal, determines, how strongly it contracts.

5.5 The Heart

We are now ready to assemble the heart.

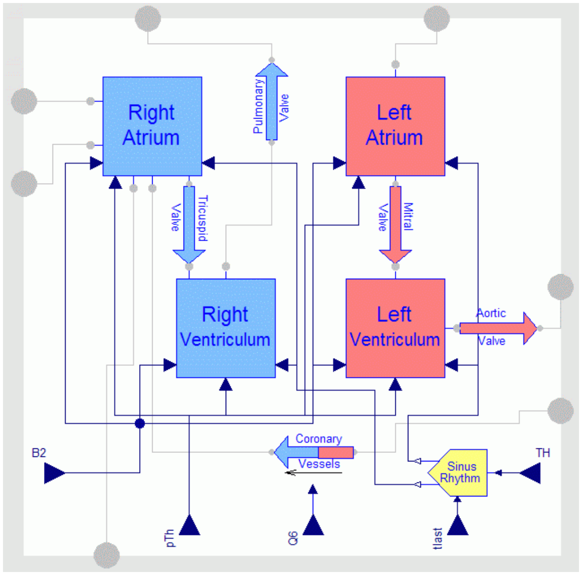


Fig. 17 *Model of the heart*

The heart contains four chambers and four valves. The sinus rhythm block computes the signals that trigger a contraction as a function of variable *TH*, which contains the current heart rate. Two separate signals are computed, since the left side and the right side of the heart contract at different times.

The coronary vessels model the arteries that provide oxygen to the heart muscle itself. This model is modulated by Q_6, the coronary resistance controller signal.

The left side of the heart is drawn on the right side of the diagram and vice-versa, which is customary in medicine, since a heart surgeon, operating on a patient, sees the left side of the patient to his right.

The heart is represented by the icon of Fig.18.

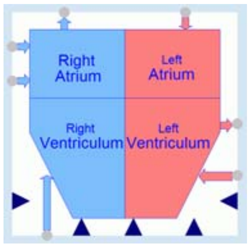


Fig. 18 *Icon representing the heart*

In our model, the heart has 7 external connections for blood flow. The heart puts out arterial blood from the left ventriculum to the body through the aortic valve. It collects the deoxygenized venous blood back into the right atrium through two major veins, the superior vena cava that collects the blood from the head and the brain, and the inferior vena cava that collects the blood from the lower body. It also collects venous blood through a separate connection coming from the bronchi. The heart puts out venous blood from the right ventriculum through the pulmonary valve to the lungs, and collects the re-oxygenized arterial blood back into the left atrium.

5.6 The Thorax

We are now ready to assemble the cardiovascular system of the thorax. It contains the heart as well as the major blood vessels of the thoracic region.

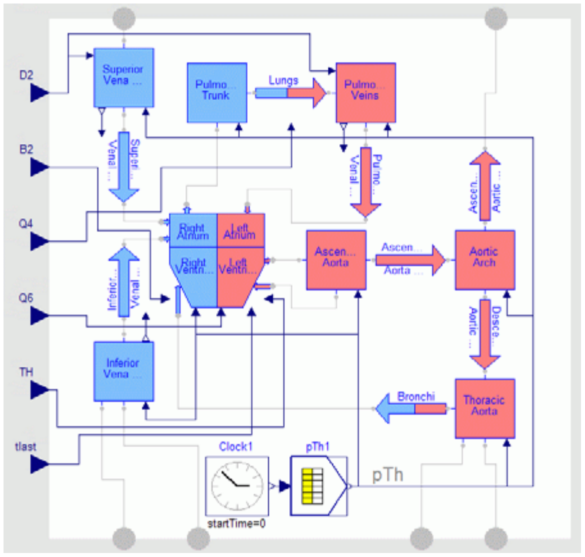


Fig. 19 *Model of the thorax*

The thorax model computes the thoracic pressure by means of a table lookup function. The thoracic pressure is then distributed as a modulating signal to all of the container models within the thoracic region. The signals shown on the left side of the model are the external modulating signals stemming from the central nervous control of the cardiovascular system. Beside from the signals B_2, Q_6, and *TH* that we already encountered before, they also include the signals: D_2, the venous tone controller signal, and Q_4, the peripheric resistance controller signal. These signals are "programmed" by the brain as a function of the signal *PAC*, the carotid sinus pressure, measuring the pressure of the arteries in the brain.

6. Conclusions

We have shown in this paper that object-oriented modeling offers a useful metaphor for mathematically describing medical systems in a hierarchical fashion. Each layer of the hierarchy is limited in its complexity, by being expressed on a single screen. Thereby, each model remains easily understandable and manageable.

With increasing abstraction, the higher levels of the modeling hierarchy employ more and more notations and terminologies that the domain expert is familiar with, hiding the more mathematical descriptions of the lower levels of the hierarchy from the high-level user of these models. In this way, the models can be understood and criticized by medical experts. The lowest levels of the modeling hierarchy employ bond graphs, a generic graphical physical systems modeling methodology, enabling us to model even the lowest levels of the model hierarchy in graphical terms, rather than having to rely on textual models, expressed in terms of mathematical equations.

References

[1] J. McLeod. PHYSBE: A physiological simulation benchmark experiment, *Simulation*, vol.7(6), pp. 324-329, 1966.

[2] J.E. Beneken, V.C. Rideout. The use of multiple models in cardiovascular system studies: transport and perturbation methods, *IEEE Trans Biomed Eng*, vol. 15(4), pp. 281-289, 1968.

[3] M.F. Snyder, V.C. Rideout. Computer simulation studies of the venous circulation, *IEEE Trans Biomed Eng*, vol. 16(4), pp. 325-334, 1969.

[4] J.F. Martin, A.M. Schneider, J.E. Mandel, R.J.

Prutow, N.T. Smith. A new cardiovascular model for real-time applications, *Transactions of SCS*, vol. 3(1), pp. 31-65, 1986.

[5] V.K. Sud, R.S. Srinivasan, J.B. Charles, M.W. Bungo. Mathematical modeling of flow distribution in the human cardiovascular system, *Med Biol Eng Comput*, vol. 30(3), pp. 311-316, 1992.

[6] M. Vallverdú. *Modelado y Simulación del Sistema de Control Cardiovascular en Pacientes con Lesiones Coronarias*, Ph.D. dissertation, Instituto de Cibernética, Universitat Politècnica de Catalunya, 1993.

[7] A. Nebot, F.E. Cellier, M. Vallverdú. Mixed quantitative-qualitative modeling and simulation of the cardiovascular system, *Comput Methods Prog Biomed*, vol. 55(2), pp. 127-155, 1998.

[8] E.E.L. Mitchell, J.S. Gauthier. *ACSL: Advanced Continuous Simulation Language – User Guide / Reference Manual*, Mitchell and Gauthier, Inc., Boston, Mass., 1986.

[9] D. Brück, H. Elmqvist, H. Olsson, S.E. Mattsson. Dymola for Multi-engineering Modeling and Simulation, *Proc 2nd Intl Modelica Conf*, Oberpfaffenhofen, Germany, pp. 55:1-8, 2002.

[10] F.E. Cellier, R.T. McBride. Object-oriented modeling of complex physical systems using the Dymola bond-graph library, *Proc ICBGM'03 6th SCS Intl Conf Bond Graph Modeling and Simulation*, pp. 157-162, 2003.

[11] F.E. Cellier, A. Nebot. The Modelica bond graph library, *Proc 4th Intl Modelica Conf*, Hamburg, Germany, vol.1, pp. 57-65, 2005.

[12] H. Elmqvist. *A Structured Model Language for Large Continuous Systems*, Ph.D. dissertation, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1978.

Authors' Biographies

François E. Cellier received his B.S. degree in electrical engineering from the Swiss Federal Institute of Technology (ETH) Zurich in 1972, his M.S. degree in automatic control in 1973, and his Ph.D. degree in technical sciences in 1979, all from the same university. Dr. Cellier worked at the University of Arizona as professor of electrical and computer engineering between 1984 and 2005. He recently returned to his alma mater of ETH Zurich.

Dr. Cellier's main scientific interests concern modeling and simulation methodologies, and the design of advanced software systems for simulation, computer aided modeling, and computer-aided design. Dr. Cellier has authored or co-authored more than 200 technical publications, and he has edited several books. He published a text book on *Continuous System Modeling* in 1991 with Springer-Verlag, New York. He just finished his second text book on *Continuous System Simulation*, which shall appear in 2005 with Springer-Verlag, New York.

Dr. Cellier served as general chair or program chair of many international conferences, and serves currently as president of the Society for Modeling and Simulation International.

Àngela Nebot received her licentiate and Ph.D. degrees in computer science from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1988 and 1994, respectively. From 1988 until 1992, she was with the Institut de Cibernètica, holding a pre-doctoral research grant from the government of Catalunya, and completing the doctoral courses in software and artificial intelligence. She joined the department of Llenguatges i Sistemes Informàtics in 1994 as an assistant professor, and since March 1998, she has been associate professor in the same department. She currently belongs to the soft-computing group of the UPC.

Dr. Nebot is an associate editor of the journal Simulation: Transactions of the Society for Modeling and Simulation International, and serves as a reviewer for other international journals such are the International Journal of General Systems, Neurocomputing, Artificial Intelligence in Environmental Engineering, and Artificial Intelligence Communication. Her current research interests include fuzzy systems, neuro-fuzzy systems, genetic algorithms, simulation, and e-learning.