MODELING OF HYDRAULIC SYSTEMS FOR HARDWARE-IN-THE-LOOP SIMULATION: A METHODOLOGY PROPOSAL

Jorge A. Ferreira
Department of Mechanical Engineering
University of Aveiro
3810 Aveiro, Portugal

João E. de Oliveira
Department of Electronic Engineering
University of Aveiro
3810 Aveiro, Portugal

Vítor A. Costa
Department of Mechanical Engineering
University of Aveiro
3810 Aveiro, Portugal

ABSTRACT

The present paper proposes a methodology to organize model libraries of electro-hydraulic components. This methodology holds in the association of an object oriented modeling, equation based, language for model structure description, with a graphical formalism suitable for the dynamic behavior description of reactive hybrid systems. That is, a recent general purpose language for physical modeling, called Modelica, is used to develop object-oriented libraries of models for different physical domains; the hierarchical description of the dynamic behavior of each model is obtained by means of the Statecharts formalism.

With the method proposed, complex models are built by model interconnection schemes; it is possible to organize the models for different complexity levels adapting them for distinct simulations (real-time or off-line), by refining their behavior using the Statecharts graphical formalism.

The developed hydraulic models, once compiled into C-code, are assembled as a whole application, and executed by a digital signal processing card (DSP). The simulation task is based on hardware-in-the-loop techniques in such a way that the "virtual" hydraulic application is controlled by real hardware.

1. INTRODUCTION

Hydraulic systems have been, for a long time, often used in industrial manufacturing and in heavy machinery. Hydraulic hardware suffered a great evolution during the last years, from hydro-mechanical devices to microprocessor controlled electro-hydraulic systems. The use of electronics and microprocessors contributes to improve the dynamic performance and to enhance the traditional systems with new features, as well as with new control possibilities. Due to the complex dynamics and non-linearity of these systems, the control algorithms usually applied in linear systems, such as the traditional PID algorithms, can have a poor performance for sophisticated hydraulic applications. Nowadays, there is an increased interest on strategies for using electro-hydraulics in advanced manufacturing systems, where dynamic performance and precision are very important parameters as, for example, in high-speed machining, injection molding systems or high-speed assembly hydraulic robots. It is important to investigate how advanced control schemes can improve the hydraulic actuation of this type of machinery. According to [Edge, 1997], it is important that, for a given application, the relative merits of different control schemes can be evaluated, being the computer simulation one of the best evaluation tool. Supporting this idea is [Ellman et al., 95] referring that a simulated environment is the cheapest and fastest way to test control algorithms. Modeling and real time simulation of complex systems still is an area to explore [Burrows, 1998] and, with the growing of computating power, more complex systems can be simulated in real time with decreasing costs [Lennevi et al., 1995].

The application of new control schemes on real hydraulic systems is a difficult task due to the cost and/or size of the hardware and its working conditions. In many applications, it is impossible to reproduce, in the laboratory, the hydraulic systems and their operating conditions. Some specific studies have been performed, where the operating environment and the hydraulic machinery hardware were reproduced by simulation and 3D visualization of the simulated model's dynamic behavior. Examples of such studies can be found in [Gonthier and Papadopoulos, 1998], [DiMaio at al., 1998] or [Schothorst, 1997]. However, these works use private modeling methodologies, thus precluding the interchange and the refining of the models.

Why a new methodology proposal?

Although the strong recent evolution in electro-hydraulic hardware, the project of hydraulic systems still is essentially based on tradition and experience. This fact, added to the growing complexity of modern hydraulic systems, can lead to unexpected behavior and errors. One way to predict these situations is the use of computer simulation, in many cases with ad-hoc simulation programs, usually written in FORTRAN, or with physical prototypes [Ellman et al., 1995].

The modeling language to be used must deal with various physical domains involved (hydraulics, mechanics, electrics, etc.) and with hybrid systems, where the involved components have a continuos and/or discrete dynamic behavior.

The representation of models in a systematic and flexible way has been studied in recent years, and there are specific domain and general purpose modeling languages [Otter and Cellier, 1995]; in either case, two main philosophies can be identified: traditional programming techniques and object-oriented methodologies. Nowadays, the tendency is focused toward object-oriented languages mainly due to the simplicity when reusing, expanding or adapting models. Confirming this idea is the work done by [Beater, 1998], referring that the largest time consuming step in system modeling can be speed up by using modern object-oriented simulation languages and component libraries.

The integration of models from different domains is a complex and time consuming task because, although powerful libraries exist, they are generally based on different modeling languages, almost invariably, not compatible. To minimize this situation there is a strong effort of a working group (including simulation tool builders, computer scientists and users from different domains) in order to build a unified object-oriented language for physical systems modeling. This language, named Modelica [Mattsson and Elmqvist, 1997], is intended for modeling virtually any application domain (electrical circuits, multi-body systems, hydraulics, thermodynamic systems, chemical systems, etc.), and it allows the inclusion of several formalisms for behavior description (ODE, DAE, bond graphs, finite state automata, petri nets or statecharts).

It is believed that the association of an object oriented modeling language, for model structure description, with a graphical formalism suitable for the dynamic behavior description of hybrid systems can be an interesting and useful approach for electro-hydraulic systems modeling.

The objective of this paper is to present a methodology where such an association, using Modelica and Statecharts, is embedded in a hardware-in-the-loop simulation environment.

2. HYBRID STATECHARTS AND MODELICA

A brief review on the formalism of Statecharts

The formalism of Statecharts [Harel, 1987] is intended to describe the dynamic behavior of complex reactive systems. It is viewed as an extension of the finite-state-machine (FSM) formalism with the add-ons of *hierarchy*, *parallelism* and *broadcast communication*.

Hierarchy is a well accepted approach for dealing with complexity, helping the human abstraction process. In Statecharts, hierarchy is used to group sets of states together, allowing high level description and step-wise development. The designer can start with an high level description of the model and proceed with the refinement of states by means of *AND/OR* decomposition operations. This hierarchical organization encourage "zoom" capabilities for moving easily back and forth between different levels of abstraction.

Concurrence inside a statechart can be described with AND states, allowing modeling of concurrent activities (*parallelism*) in the same model via *orthogonal* states. All these orthogonal states are activated when an *AND* state is entered and deactivated when it is exited.

Figure 1 shows the high level description of a system with states A and B. These states can be refined through state decomposition. State A will be a compound *OR* state with substates A1, A2 and A3. State B will be a compound *AND* state with substates B1 and B2. This process can proceed until low level description is achieved.

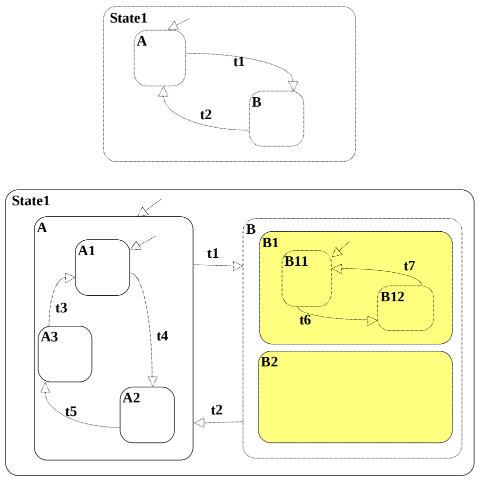

Fig. 1 - Refining states to achieve low level system's description

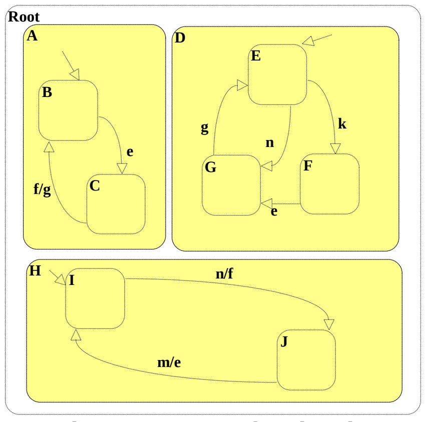

Fig. 2 - Broadcast communication in Statecharts

When orthogonal components are not truly independent, communication between them must be specified by associating an action with a transition. This action is assumed to be broadcasted and so every system component in the statechart will recognize the message. This *broadcast communication* mechanism allows that, when one part generates an event (attaching an action to a transition), all the other parts sense it, acting in response if it is so specified. In the example of figure 2, state *A* generates an event when transition *f* is fired; this event is sensed if state *G* is active, although state *G* belongs to another orthogonal state.

Another enhancement over the FMS formalism is the association of an event *action* with a transition, when it is taken, or with a state, when the state is entered or exited. Continuous *activity* can also be associated with a state for modeling continuous behavior when the state is active. These *action* and *activity* concepts allow the modeling of hybrid systems: *actions* capture the discrete features of the system while *activities* describe the continuous part.

Modelica

Modelica is a new language for physical modeling that is being developed as an international effort with the main objective of making easy the exchange of models and model libraries. The language is built on non-causal modeling with algebraic and differential equations, and uses object-oriented constructs to facilitate model reuse, through hierarchical modeling, encapsulation, and inheritance.

Models and submodels are declared as classes with connection interfaces called *connectors*. This connection capability allows the use of model libraries to compose complex models with the drag and drop, and connection drawing facilities of modern graphical editors.

With Modelica, the modeling of hybrid systems is supported via mixed continuous/discrete systems of equations. Discontinuous models can be handled with *if-then-else* expressions, allowing the modeling of phenomena with different expressions in different operating regions. Models with different complexity levels can be supported by the use of conditional equations, in such a way that changes on behavior are obtained by just setting a parameter. Discrete event and discrete time models are supported by *when* statements. The equations in a *when* clause are conditionally activated at event instants where the *when* condition becomes true.

The Statecharts library in Modelica

A small library to implement the hybrid Statecharts formalism in Modelica language was already developed [Ferreira and Oliveira, 1999]. Two statecharts implementation levels were considered: library models and component models. Statecharts library models are responsible for capturing events related to the firing of transitions and to the activation and deactivation of states that must be performed when transitions are taken. Library models have also to code the activation or deactivation of the substates, in order to implement what is expected with OR states and AND states. Component models create the statechart, with the basic models provided by the Statecharts library, make the state-transition-state connections and define the transition events or describe the continuous activities within states.

The approach followed to generate the code for statechart implementation is to consider a statechart as a model in Modelica. This model will be composed by states and transitions. Modelica models are developed for the basic elements of the statechart. The final model of the statechart is a set of Modelica models connected by state-transition-state connectors.

The equation based modeling of the Modelica language, along with the connector constructions, proves its efficiency by passing activation/deactivation messages instantaneously through nested states. Also, the broadcast communication mechanisms of Statecharts can easily be fulfilled, just by setting the value of a variable; this is automatically transmitted to all the statechart components because, in fact, the statechart is implemented with differential algebraic equations (DAE) that are evaluated concurrently.

Behavior inheritance of the Statecharts library

The main guideline followed by Harel [Harel and Gery, 1997], concerning the Statecharts behavior inheritance, is to base the two statecharts on the same underlying state/transition topology. Thus, B inherits all A states and transitions. Although these cannot be removed, certain changes are allowed. States can be modified in three ways: breaking down a basic state by OR (into substates) or by AND (into orthogonal components) decompositions; adding substates to an OR state; adding orthogonal components to any state. This last way is the most important because it enriches A behavior capabilities.

Transitions can also be added to the statechart, and some modifications are allowed in the inherited ones. For example, if the transition is labelled by *event*[*guard*]*action*, changes can be made in the trigger *event*, the *guard* or even in the *action* list. Although, explicitly, it is not possible to remove a transition, it can be done implicitly by making its *guard* false.

All these features were implemented [Ferreira and Oliveira, 1999] in the statecharts Modelica library by means of boolean switch parameters associated with the status of each state or transition. That is, each state or transition can be inhibited, at compile time, by setting its enabling parameter to false.

3. MODELING METHODOLOGY

The main directions to model the dynamic behavior of modern hydraulic systems are: object oriented libraries of models; hierarchical description of the dynamic behavior; model interconnection is used to build more complex models; various levels of model complexity achieve different simulation experiments; ability to refine or redefine behavior; graphical description of dynamic behavior to enhance model understanding.

The concept behind the methodology is to consider a model, of a physical component, as a composition of two complementary perspectives: its structure and its behavior.

The structure characterize the static part of the model, its parameters, its connection terminals, etc. Object oriented techniques and the design of hierarchies of models, connected by inheritance mechanisms, are important issues to develop reusable model's libraries. Mechanisms for model interconnection can also make easier complex system's modeling.

The dynamic part of the model, that is, its time evolution, is described by its behavior. This behavior depends on time, captured events or changing attributes that derive from the model's connection terminals. The behavior of an hydraulic model can be reactive when stimulated through its connection terminals. Thus, it is believed that a graphical formalism (such as Statecharts) is suitable to describe reactive hybrid systems and very useful for the hierarchical description of the model's dynamics.

The proposed methodology holds in the association of an object oriented modeling, equation based, language (Modelica) for model structure description, with a graphical formalism (Statecharts) suitable for the dynamic behavior description of reactive hybrid systems. The implementation of this graphical formalism inside the modeling language allows the methodology to be supported by a single global language.

The association of the object oriented modeling language with the graphical formalism leads to two types of inheritance for the component models. The model structure can be inherited through the usual mechanisms of object oriented languages. For model behavior description, this work purposes the adoption of the hybrid Statecharts formalism to obtain the required behavior inheritance.

The organization of model's knowledge is simple because the model's behavior can be hierarchically detailed by nesting statecharts. In fact, the statechart states can be decomposed until low level behavior description is achieved. An interesting feature of this approach is the possibility of inheriting and refining model's behavior in a very understandable manner, and with well-defined rules; this allows the organization, for the same component, of models whose behaviors can have various complexity stages suitable for different simulation experiments. For example, component models can be organized with a complexity order number; then, by just defining a certain degree of complexity, a complex model can be composed by several connected component models, either for real time simulation or for off line simulation experiments. The user can always inhibit the behavior of models and define a new behavior, for a part or for the whole model, in a very elegant way.

A simple example is presented shortly to illustrate the method.

The relief valve example

Consider the example of a component, relief valve (fig. 3), that limits the pressure in a hydraulic system. It is a closed loop system but it is usually modelled by its (static) input/output characteristic [Beater, 1998]. Consider, as a first approach, that the valve has the behavior depicted on the left side of figure 4.

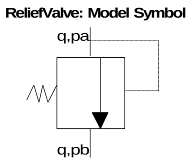

Fig. 3 – Symbol for a hydraulic relief valve

The behavior of the simple relief valve model can be described by considering that the valve has two possible stable states, and also hysteresis when it changes from one state to another. When the pressure difference (*dp*) is bigger than the pressure required to open the valve (*pressureOpen*), the valve will be totally open (*q=(dp–pressureClose)*gOpen*); when the pressure difference is smaller than the close pressure (*pressureClose*) the valve will be totally closed (*q = dp*gLeak*).

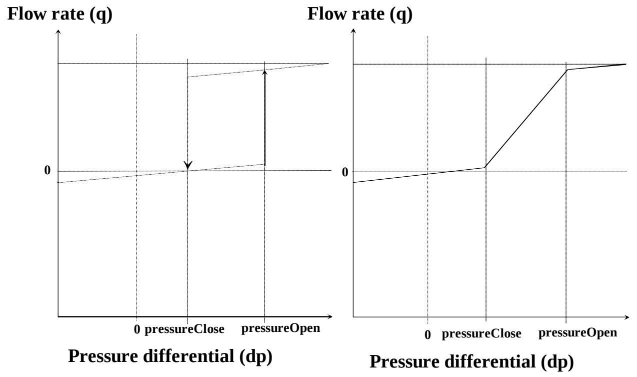

Fig. 4 – Two different behaviors for the relief valve

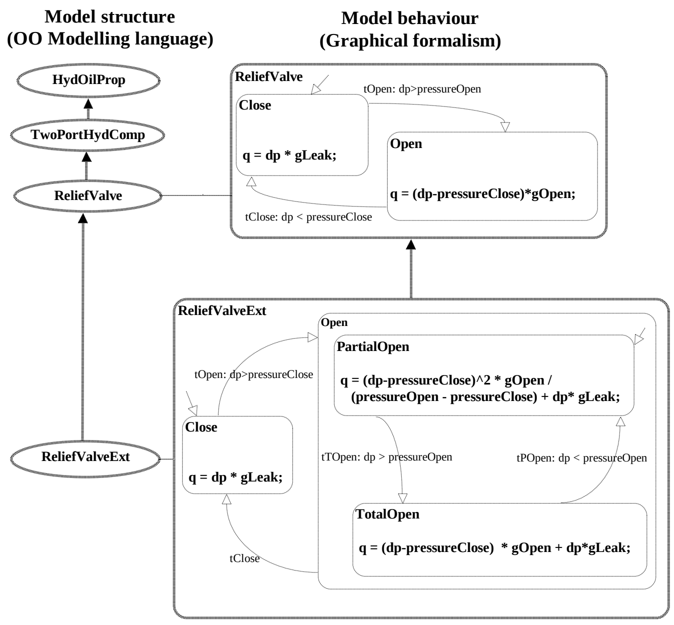

Fig. 5 – Inheritance hierarchy for the relief valve model

Using the formalism of Statecharts, the dynamic behavior of the valve (fig. 5) can be represented by an OR statechart (*ReliefValve*) that contains two substates (*Close* and *Open*) and two state transitions (*tOpen* and *tClose*). Initially the valve is closed (*Close* is the default substate). The continuous behavior of the relief valve is described by its during activity (*q = dp * gLeak*), that is enabled while the valve is in its *Close* state and is not exiting. If the valve is totally closed and its pressure difference (*dp*) exceeds *pressureOpen*, then a transition (*tOpen*) takes place and the state *Open* becomes active. The during activity of state *Open* will be evaluated for all the simulation steps while the state *Open* is active, that is, while transition *tClose* does not occur. This transition is fired when pressure *dp* reaches a value below the pressure difference defined by the parameter *pressureClose*.

Refining the relief valve model

To improve the relief valve model, the characteristics displayed at the right of figure 4 can be used. One possible solution is carried out by the statechart associated with the *ReliefValveExt* model of figure 5. The statechart shows that the states *Close* and *Open* and transition *tOpen* were redefined. The transition *tClose* is the only one that is inherited without changes from the ancestor model (*ReliefValve*).

It should be noted that the state *Open* was refined, being now an OR state with *PartialOpen* and *TotalOpen* substates. The model will inherit all the structure and, concerning the behavior, all the equations that were not redefined. For instance, the equation that defines the trigger event for the transition *tClose* (*tClose.event = dp<pressureClose*) will be inherited from the model *ReliefValve*, while the trigger event equation for transition *tOpen* must be redefined.

A subset of the Modelica code for the relief valve model's hierarchy is presented bellow. The models use the Statecharts Modelica library described in [Ferreira and Oliveira, 1999].

```
model HydOilProp "Oil properties"
  parameter Real KVisc=46e-6;        // m^2/s; kinematic viscosity
  parameter Real rho=865;            // kg/m^3; mass density
  parameter Real bulkModulus=1e9;    // Pa; bulk modulus
end HydOilProp;
```

```
connector HydConnector "Hydraulic connector"
  Real p;        // Pa; fluid pressure
  flow Real q;   // m^3/s; fluid flow
end HydConnector;
```

```
model TwoPortHydComp
  extends BasicOil;
  HydConnector HydA; // hydraulic connector A
  HydConnector HydB; // hydraulic connector B
  Real dp;        // Pa, pressure difference
  Real q;         //m^3/s; flow through component
equation
  dp = HydA.p - HydB.p;
  …
end TwoPortHydComp;
```

```
model ReliefValve
  extends TwoPortHydComp;
  parameter Real pressureOpen=55e5; //Pa
  …
  RootStateS Root;
  StateS Close(defaultState = true);
  StateS Open;
  TransitionS tOpen;
  TransitionS tClose;
equation
  tOpen.event = event(dp > pressureOpen);
  tClose.event = event(dp < pressureClose);
  q = if Open.active then (dp-pressureClose)*gOpen
      else dp * gLeak;
end ReliefValve;
```

```
model ReliefValveExt
  extends ReliefValve;
  …
  StateS PartialOpen(defaultState = true);
  StateS TotalOpen;
  TransitionS tPOpen;
  TransitionS tTOpen;
equation
…
  tOpen.event = event(dp > pressureClose);
  tPOpen.event = event(dp > pressureOpen);
  tTOpen.event = event(dp < pressureOpen);
  q=if Closed.active then dp * gLeak
    else if PartialOpen.active then (dp-pressureClosed)^2 *
        gOpen / (pressureOpen-pressureClose) + dp*gLeak
    else (dp-pressureClosed) * gOpen + dp*gLeak;
end ReliefValveExt;
```

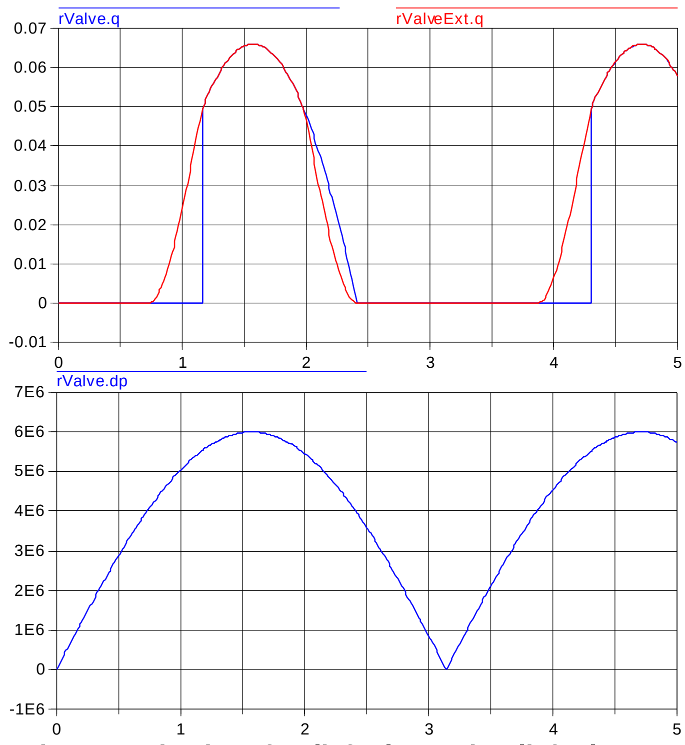

Fig. 6 – Behavior of ReliefValve and ReliefValveExt models

The dynamic behavior of instances of *ReliefValve* and *ReliefValveExt* models (*rValve* and *rValveExt*, respectivelly) is shown in the following graphics; the absolute value of a sinusoidal signal was used as the pressure difference *dp*. This model was tested with the Dymola package [Dymola] that supports the Modelica language.

This example shows how simple can be the refining of behavior of, for example, model libraries of hydraulic components, if the present methodology is used.

4. HARDWARE-IN-THE-LOOP SIMULATION OF HYDRAULIC SYSTEMS

Hardware-in-the-loop simulation refers to a technology where some of the components of a pure simulation are replaced with actual hardware. This type of procedure is useful, for example, to test a controller which, instead of being connected to the real equipment under control, is connected to a real time simulator. The controller must “think” that it is working with the real system and so the accuracy of the simulation and its electrical interfacing to the controller must be adequate [Maclay, 1997]. This technology provides a way for testing control systems over the full range of operating conditions, including failure modes. Testing a control system prior to its use in a real plant can reduce the cost and the development cycle of the overall system. Hardware-in-the-loop simulation has been used, with success, in the aerospace industry and is now emerging as a technique for testing electronic control units. It has been applied to solve some specific problems but is seldom used as a platform to test the real time behavior of hardware components.

The main purpose of the present methodology is to give a well defined support for the model libraries of electro-hydraulic components, to be used in hardware-in-the-loop simulation experiments. The global performance is related to the model’s complexity, thus, for diverse type of simulations, different model’s behavior shall be used. When using real time simulation, the model’s complexity shall take into account the dedicated real time hardware that will “run” the codified model. With the proposed methodology, the real-time simulation experiments can be done with the same model topology as the off-line simulations, because the time required for simulation can be defined by selecting the appropriate degree of complexity.

In order to use this methodology for hardware-in-the-loop simulation of electro-hydraulic systems, libraries of models with different complexities shall be developed for the different physical domains involved in such systems. This means that new basic models must be designed and also that some others must be redefined and/or refined.

The main use of hardware-in-the-loop simulation applied to hydraulic systems is the testing of control algorithms and the development of new control schemes. Usually this implies a real controller, operating over a real time simulated hydraulic plant.

The objective of future work is to be able to modify the complexity of a composed hydraulic model, just by setting parameter values; that way a model can be compiled with the complexity adjusted to the hardware platform.

Figure 7 presents an outline of the general block diagram for the modeling methodology testbench. This platform will be used in a near future to emulate electro-hydraulic systems in order to test real controllers and algorithms.

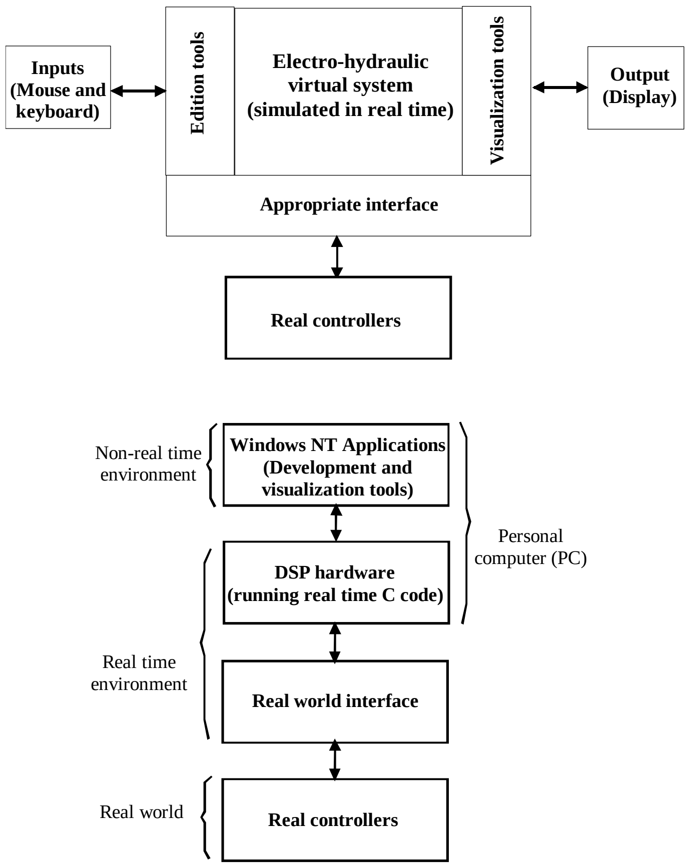
Inputs (Mouse and keyboard)
Edition tools
Electro-hydraulic virtual system (simulated in real time)
Visualization tools
Output (Display)
Appropriate interface
Real controllers
Non-real time environment
Windows NT Applications (Development and visualization tools)
Personal computer (PC)
DSP hardware (running real time C code)
Real time environment
Real world interface
Real world
Real controllers

Fig. 7 – Outline of the block diagrams of hardware and software components

5. CONCLUSIONS

The present paper proposes a methodology to organize model libraries of electro-hydraulic components, in order to easily manage complex models for hardware-in-the-loop simulation experiments.

The concept behind the methodology is the following: the model of a physical component is composed by two complementary perspectives, its structure and its behavior. For that, the methodology is based on the association of an object-oriented language, Modelica, to model the structure, with the hybrid Statecharts formalism to describe the dynamic behavior of the model.

By making use of the Statecharts inheritance rules, this work also proposes a solution to refine or redefine the behavior of hydraulic systems. This way, different simulation experiments can be achieved by choosing the appropriate behavior for each model.

REFERENCES

[Beater, 1998] – P. Beater, “Object Oriented Modeling and Simulation of Hydraulic Drives,” Simulation News Europe, March, 1998.

[Burrows, 1998] – C. R. Burrows, “Fluid Power Systems – An Academic Perspective,” JHPS Journal of Fluid Power Systems, vol 29, nº1, January 1998.

[DiMaio et al., 1998] – S. P. DiMaio et al., “A Virtual Excavator for Controller Development and Evaluation,” Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Belgium, May 1998.

[Dymola] – Dymola package: Homepage: «hyperlink http://www.dynasim.se»

[Edge, 1997] - K. A. Edge, “The control of fluid power systems – responding to the challenges,” Proceedings of Institute of Mechanical Engineers, Vol 211 part 1, 1997.

[Ellman et al., 1995] - A. Ellman, S. Sanerma, M. Salminen, R. Piché and T. Virvalo, “Tools for Control and Hydraulic Circuit Design of a Hydraulic-Driven Manipulator Mechanism,” Proceedings of the 9th European Simulation Multiconference, June, Czech Republic, 1995.

[Ferreira and Oliveira, 1999] – J. A. Ferreira, J. Estima de. Oliveira, “Modeling hybrid systems using Statecharts and Modelica,” To be presented in the 7th IEEE International Conference on Emerging Technologies and Factory Automation, October 1999, Spain.

[Gonthier and Papadopoulos, 1998] – Y. Gonthier, E. Papadopoulos, “On the Development of a Real Time Simulator for an Electro-hydraulic Forestry Machine,” Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Belgium, May 1998.

[Harel, 1987] - D. Harel, “Statecharts: A visual formalism for complex systems,” Science of Computer Programming 8, 231-274, 1987.

[Harel and Gery, 1997] - D. Harel, E. Gery, “Executable Object Modeling with Statecharts,” IEEE Computer, July 1997.

[Lennevi et al., 1995] – J. Lennevi, J. Palmberg, A. Jansson, “A Simulation Tool for the Evaluation of Control Concepts for Vehicle Drive Systems,” Proceedings of the 4th Scandinavian International Conference on Fluid Power, Tampere, September, 1995.

[Maclay, 1997] – D. Maclay, “Simulation gets into the loop,” IEE Review, May 1997.

[Mattsson and Elmqvist, 1997] – S. E. Mattsson, H. Elmqvist, “An International Effort to Design the Next Generation of Modeling Language,” Proceedings of the IFAC Symposium on Computer Aided Control Systems Design - CACSD’97, Gent, Belgium, April 1997.

[Otter and Celier, 1995] – M. Otter, and F.E. Cellier, “Software for Modeling and Simulating Control Systems,” The Control Handbook (W.S. Levine, ed.), CRC Press, Boca Raton, FL, 1995.

[Schothorst, 1997] – G. Schothorst, “Modelling of Long-Stroke Hydraulic Servo-Systems for Flight Simulator Motion Control and System Design,” PhD thesis, Mechanical Engineering Systems and Control Group, Delft University of Technology, The Netherlands, 1997.