SIMULATION OF REAL TIME SYSTEMS BEHAVIOR CONSIDERING HUMAN-MACHINE INTERFACE

Eurico Seabra, José Machado

University of Minho; School of Engineering; Mechanical Engineering Department; Campus of Azurém; 4800-058 Guimarães; Portugal

eseabra@dem.uminho.pt (Eurico Seabra)

Abstract

In this paper it is shown, as it is possible, and desirable, the use of a simulation technique in the development and analysis of industrial controllers, mainly to assure, to them, more robustness and less errors. In particular, the use of digital controllers has been proven that small errors in their design may lead to catastrophic failures. Simulation is used in the design process of dynamic systems. The results of simulation are employed for validating a model, and they are helpful for the improvement of the design of a system with respect to both, qualitative and quantitative properties. The paper concentrates on these aspects and applications of simulation, presents building modules blocks for modelling and simulation with respect to both, contents and methods, and discusses the requirements for simulation and validation. Modelling and simulation is almost necessarily based on modelling languages with precise semantics. In this paper the modelling was performed by using the object-oriented language Modelica and the library for hierarchical state machines StateGraph. The simulation used to evaluate the controller and plant behaviour has been developed and proposed in this paper. The present research proved to be successful using the Modelica programming Language to obtain plant models. Also it will be proved that is adequate the use of the Dymola software with the library for hierarchical state machines StateGraph for the analysis of industrial controllers.

Keywords: Real-time systems, Simulation, Systems models, Object-Oriented Language

Presenting Author's biography

Eurico Seabra. Is an assistant professor in the Department of Mechanical Engineering at the University of Minho in Portugal. His research and teaching interests include simulation modelling and analysis, data acquisition systems, industrial automation, mechanical design and biomechanics. In the year of 2005 he has obtained his Phd in Mechanical Engineering in the University of Minho (Portugal).

1 Introduction

In recent years, many software engineering researchers have identified the software process as the key issue to obtain higher quality products, improved productivity, and more controllable projects [1,2,3,4]. By software process we mean the set of activities, rules, methodologies, tools, and roles that participate in the development of software within a given organization. For this purpose the software engineering community is producing an increasing effort in designing and developing languages and the related support technology to formally describe, assess, and - wherever possible - automate software processes.

Such languages have been designed to allow automatic generation of efficient simulation code from declarative specifications. The Modelica language [5,6,7] and its associated support technologies have achieved considerable success through the development of specific libraries.

Modelica supports both high level modelling by composition and detailed library component modelling by equations. Models of standard components are typically available in model libraries. Using a graphical model editor, a model can be defined by drawing a composition diagram (also called schematics) by positioning icons that represent the models of the components, drawing connections and giving parameter values in dialogue boxes. Constructs for including graphical annotations in Modelica make icons and composition diagrams portable between different tools. In this paper it is shown, as it is possible, and desirable, the use of this simulation technique in the analysis of industrial controllers, because the industrial controllers developed will be more robust and less subject to errors.

To accomplish our goals, in this work, the paper is organized as follows. In Section 1, it is presented the challenge proposed to achieve in this work. Section 2 presents a general presentation of the case study involving a tank filling/emptying system. Further, it is presented the methodology to obtain the controller program deduced from an IEC 60848 SFC specification. Section 3 is exclusively dedicated to the plant modelling, being presented the adopted approach. Section 4 presents and discusses the obtained results on simulation performed with Modelica Language. Finally, in Section 5, the main conclusions and future work are presented.

2 System description

Figure 1 illustrates an example of a tank filling/emptying system, which consists of two tanks, two level analogue sensors (one for each tank) and three on-off valves. In the normal operation mode, the system works as follows.

Tank1 is filled by opening valve V1. When the level of the tank1 becomes high, the valve V1 is closed. After a waiting time of ten time units, valve V2 is opened and the fluid flows from tank1 into tank2. When tank1 is empty, valve2 is closed and, after a waiting time of fifteen time units, valve3 is opened and the fluid flows out of tank2. Finally when tank3 is empty, valve V3 is closed. In this work we consider that one time unit is equal to one second.

The above normal operation can be influenced by three buttons: start, stop and shut. In order to guarantee the desired functioning of the system it is necessary to simulate the following desired behaviours, traduced by three system behaviour properties:

Property 1: Button “start” starts the above process. When this button is pressed, after a “stop” or “shut” operation, the process operation continues;

Property 2: Button “stop” stops the above process by closing all valves. Then, the controller waits for further input (either “start” or “shut” operation);

Property 3: Button “shut” is used to shutdown the process, by emptying, at once, both tanks. When this is achieved, the process goes back to its starting configuration. Clicking on the button “start” restarts the process.

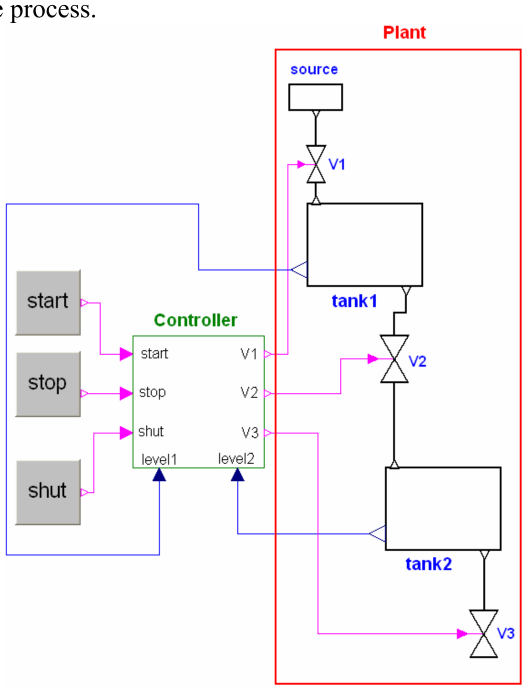


Fig. 1 Scheme of the entire filling/emptying system

2.1 Controller

In order to guarantee the desired behaviour of the system described above, the controller was developed according to IEC 60848 SFC specification, which is presented in Fig. 2.

The PLC program which controls the process in closed-loop has input and output variables as described in Tab. 1.

Tab. 1 Input/Output variables of the controller

Input	Output
start – system start	V1 – open valve1
stop – system stop	V2 – open valve2
shut – system shutdown	V3 – open valve3
level1 – % fill tank1	
level2 – % fill tank2	

The tank level is given in % of the fill tank. The Boolean variables T1F (tank1 full) and T2F (tank2 full) were considered true when the level1 and level2 was greater than 0.98, respectively. On the other hand, the Boolean expression T1E (tank1 empty) and T2E (tank2 empty) were assumed true when the level1 and level2 was less than 0.01, respectively.

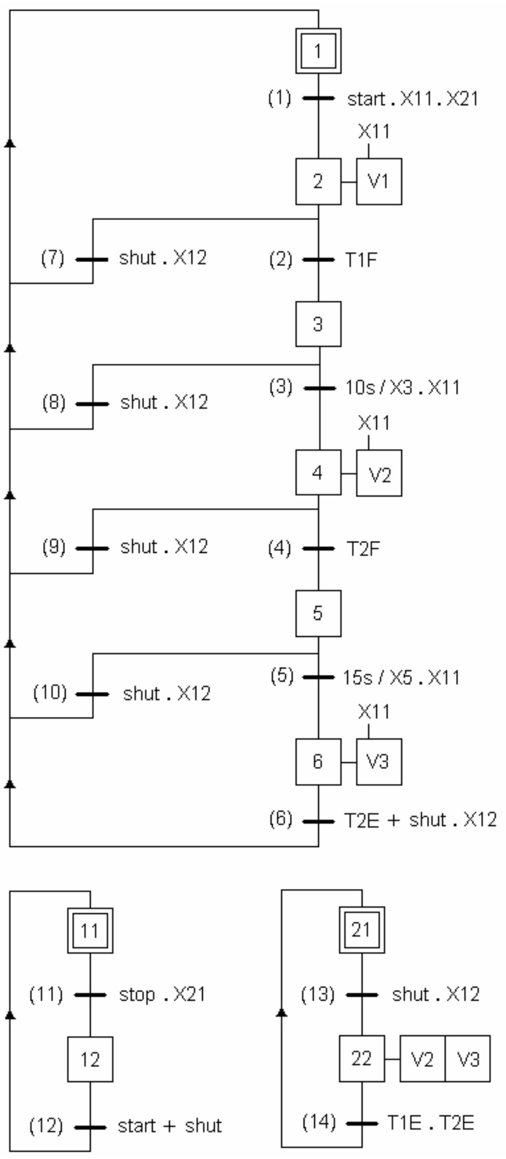


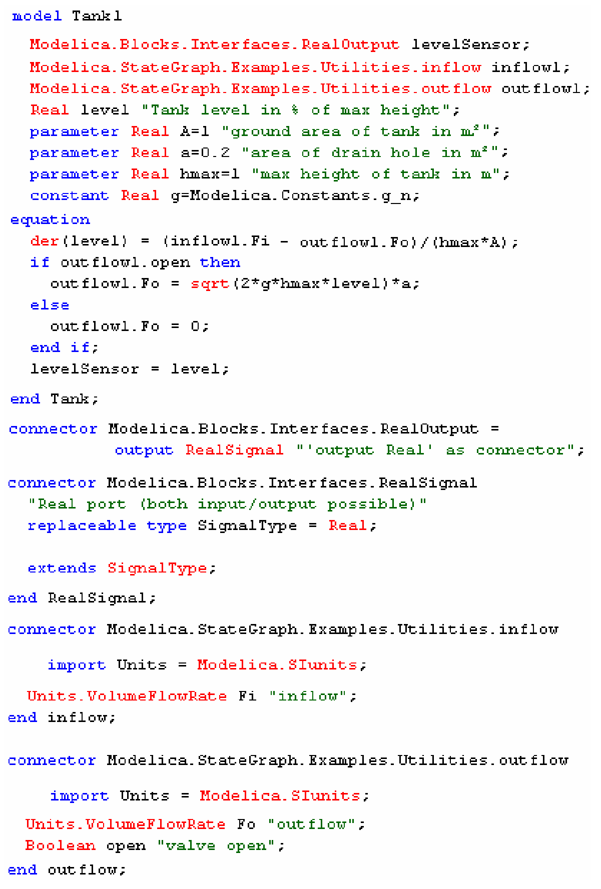
Fig. 2 SFC specification of the controller

The controller was modelled using the Dymola software and the object-oriented programming language Modelica [8,9] with the library for hierarchical state machines StateGraph [10].

3 Plant model

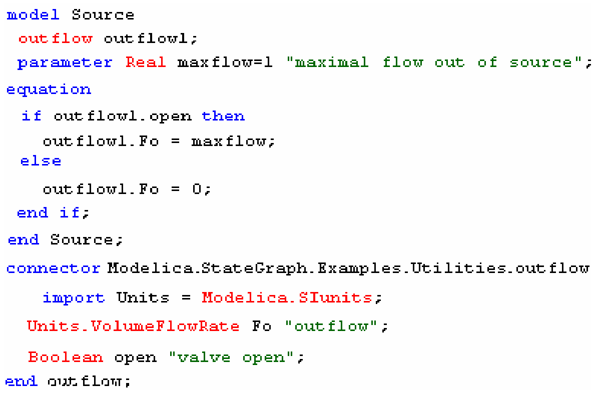
The plant was modelled as the controller using the Dymola software and the object-oriented programming language Modelica.

Figure 3 presents the program code for the tank1 model. The Modelica code for modelling the tank2, is similar to the code obtained for the tank1 model because is equal in the number of fill sources. Additionally, Fig. 4 and Fig. 5 present the program code used for modelling, respectively, the source and the valves.



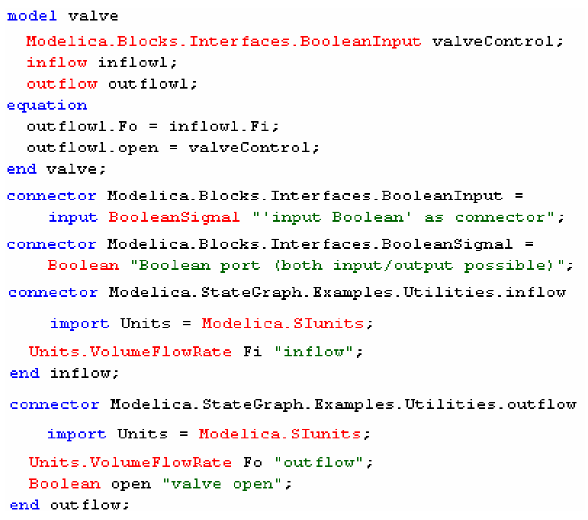
```
model Tank1
  Modelica.Blocks.Interfaces.RealOutput levelSensor;
  Modelica.StateGraph.Examples.Utilities.inflow inflow1;
  Modelica.StateGraph.Examples.Utilities.outflow outflow1;
  Real level "Tank level in % of max height";
  parameter Real A=1 "ground area of tank in m²";
  parameter Real a=0.2 "area of drain hole in m²";
  parameter Real hmax=1 "max height of tank in m";
  constant Real g=Modelica.Constants.g_n;
equation
  der(level) = (inflow1.Fi - outflow1.Fo)/(hmax*A);
  if outflow1.open then
    outflow1.Fo = sqrt(2*g*hmax*level)*a;
  else
    outflow1.Fo = 0;
  end if;
  levelSensor = level;
end Tank;

connector Modelica.Blocks.Interfaces.RealOutput =
           output RealSignal "'output Real' as connector";

connector Modelica.Blocks.Interfaces.RealSignal
  "Real port (both input/output possible)"
  replaceable type SignalType = Real;

  extends SignalType;
end RealSignal;

connector Modelica.StateGraph.Examples.Utilities.inflow

    import Units = Modelica.SIunits;

  Units.VolumeFlowRate Fi "inflow";
end inflow;

connector Modelica.StateGraph.Examples.Utilities.outflow

    import Units = Modelica.SIunits;

  Units.VolumeFlowRate Fo "outflow";
  Boolean open "valve open";
end outflow;
```

Fig. 3 Modelica program code for the tank1model



```
model Source
 outflow outflow1;
 parameter Real maxflow=1 "maximal flow out of source";
equation
  if outflow1.open then
    outflow1.Fo = maxflow;
  else
    outflow1.Fo = 0;
 end if;
end Source;
connector Modelica.StateGraph.Examples.Utilities.outflow
    import Units = Modelica.SIunits;
  Units.VolumeFlowRate Fo "outflow";
  Boolean open "valve open";
end outflow;
```

Fig. 4 Modelica program code for the model source



```
model valve
  Modelica.Blocks.Interfaces.BooleanInput valveControl;
  inflow inflowl;
  outflow outflowl;
equation
  outflowl.Fo = inflowl.Fi;
  outflowl.open = valveControl;
end valve;
connector Modelica.Blocks.Interfaces.BooleanInput =
    input BooleanSignal "'input Boolean' as connector";
connector Modelica.Blocks.Interfaces.BooleanSignal =
    Boolean "Boolean port (both input/output possible)";
connector Modelica.StateGraph.Examples.Utilities.inflow
    import Units = Modelica.SIunits;
  Units.VolumeFlowRate Fi "inflow";
end inflow;
connector Modelica.StateGraph.Examples.Utilities.outflow
    import Units = Modelica.SIunits;
  Units.VolumeFlowRate Fo "outflow";
  Boolean open "valve open";
end outflow;
```

Fig. 5 Modelica program code for the valves model

4 Simulation results

In order to perform the simulation, it is necessary to define the plant variables (Tab. 2), start and stop time of the simulation, the number of output intervals and the integration algorithm. In the present work, in all simulations performed, the Dass algorithm [11] with 500 output intervals was used.

Tab. 2 Variables of the plant

Plant	Variable
source	Q - flow rate [m^3/s]
tank1, tank2	G1, G2 – ground area [m^2] Ht1, Ht2 – height [m] A1, A2 – drain hole area [m^2]

The simulations were performed with the purpose of verifying if the SFC corresponding to the controller system specification (Fig.2) - modelled with Modelica language, with the library for hierarchical state machines StateGraph - is adequate for correctly simulating the desired system behaviour. This desired behaviour considers the system behaviour properties, indicated before, related to the human-machine interface that is traduced by the start, stop and shut buttons actuation. The values for the plant variables considered in these simulations were Q=1, G1=G2=1, Ht1=Ht2=1, A1=0.2 and A2=0.05. The Tab. 3 depicts the buttons actuation period times used in the three simulations performed.

Tab. 3 Time actuation of the buttons

Simulation	Period time actuation [s]		
	Start	Stop	Shut
Start system	[0-1]		
Stop system	[0-1, 24-25]	[11-12]	
Shut system	[0-1]	[11-12]	[15-16]

Figure 6 shows the level tanks results of the first simulation that they were not pressed the buttons "stop" and "shut" during the production cycle, which corresponds to the system normal operation mode.

Observing Figure 6 it can be concluded that the system is properly simulated by the developed Modelica program code, since during the time specified by the SFC specification the tanks remain filled and empty.

On the other hand, Figure 7 shows results of the second simulation performed with the actuation of the "stop" button during the production cycle.

Analyzing Figure 7 it can be also concluded that the stop operation (closing all valves) is properly simulated by the proposed program. Because it can be verified, that after the actuation of the "stop" button (time 11s) the tanks levels stay stationary until being pressed the "start" button (time 24s). After that time, by opening the appropriate valves, the filling/emptying process continues.

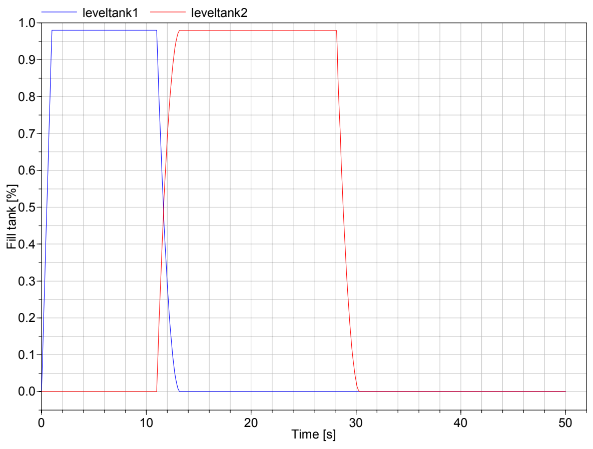


Fig. 6 Level tanks in function of time in normal operation of the system

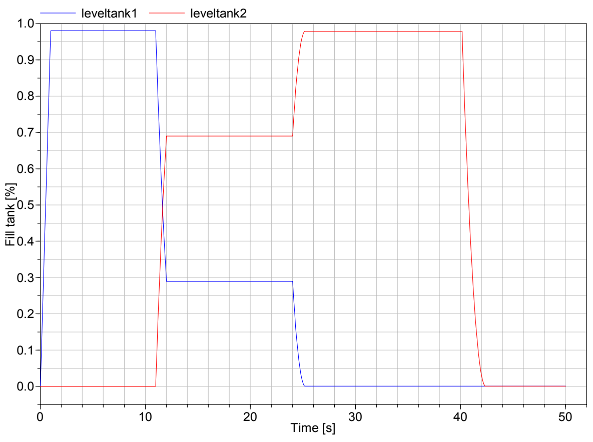


Fig. 7 Level tanks in function of time in the stop operation of the system

Figure 8 shows results of the third simulation performed with the actuation of the "shut" button after the actuation of "stop" button during the production cycle.

Observing Fig. 8 it can be also concluded that the shutdown operation is properly simulated by the proposed program. It can be verified, that after the actuation of the "stop" button (time 11s) the tanks levels stay stationary until being pressed the "shut" button (time 15s). After that time, the solution present in the tank1 is immediately drained for the tank2 and later emptied by the opening of valves 2 and 3.

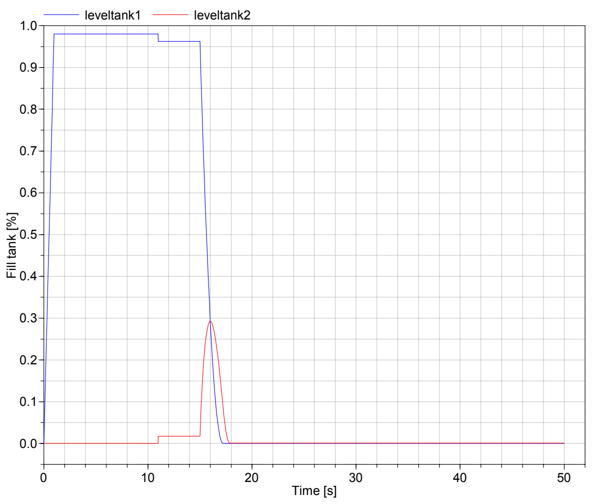


Fig. 8 Level tanks in function of time in shutdown operation of the system

As it was shown, in this chapter, all the desired behaviour properties are guaranteed, by simulation, of our controller.

5 Conclusions

The simulation used to evaluate the controller and plant behaviour has been developed and proposed in this paper.

The present research proved to be successful using the Modelica programming Language to obtain plant models. Also it will be proved that is adequate the use of the Dymola software with the library for hierarchical state machines StateGraph for the analysis of industrial controllers. It is possible to eliminate a set of program errors that correspond to possible dangerous system behaviours in reduced intervals of time. This approach implies that the industrial controllers developed will be more robust and less subject to errors.

Moreover, the simulation techniques allow us to test different delays of the plant functioning and to see if a property, for different considered delays, is still true or if different delays imply that a property is true and after is false.

In conclusion, using this approach a manufacturer of industrial automated systems does not need the plant (physical part of the machine) for later perform tests and simulation of the system controller. In consequence, this approach allows to reduce the times of production of automated systems.

6 Acknowledgements

This research project is carried out in the context of the SCAPS Project supported by FCT, the Portuguese Foundation for Science and Technology, and FEDER, the European regional development fund, under contract POCI/EME/61425/2004 that deals with safety control of automated production systems.

7 References

[1] Humphrey W.S. Managing the Software Process. In Addison-Wesley (Eds), *SEI Series in Software Engineering*, 1989.

[2] Junkermann G., B. Peuschel, W. Schäfer, and S. Wolf. In Research Studies Press Ltd, Merlin: Supporting Cooperation in SoftwareDevelopment Through a Knowledge-Based Environment, *Software Process Modelling and Technology*, 103–130, 1994.

[3] Montangero C. and V. Ambriola.. OIKOS: Constructing Process-Centered SDEs. In A. Finkelstein, J. Kramer, and B. Nuseibeh, editors, *Software Process Modelling and Technology*, 187–222, 1994.

[4] Lamb C.W., G. Landis, J.A. Orestein, and D.L. Weinreb.. *The Object Store Database System*. Communications of the ACM, 34(10), October, 1991.

[5] Fritzson, Peter, Vadim E. *Modelica, a general object-oriented language for continuous and discrete event system modeling and simulation*, 12th European Conference on Object-Oriented Programming (ECOOP'98). Brussels, Belgium, 1998.

[6] Elmqvist, Hilding, Mattsson S., Otter M. *Modelica - a language for physical system modeling*, Proceedings of the IEEE Symposium on Computer-Aided Control System Design. August, Hawaii, 1999.

[7] Fritzson, Peter, Bunus P. *Modelica, a general object-oriented language for continuous and discrete event system modelling and simulation*, Proceedings of the 35th Annual Simulation Symposium. April, San Diego, CA, 2002.

[8] Fritzson, Peter, Vadim E.,. *Modelica, a general object-oriented language for continuous and discrete-event system modeling and simulation*, 12th European Conference on Object-Oriented Programming (ECOOP'98). Brussels, Belgium, 1998.

[9] Elmqvist E., Mattson S. *An Introduction to the Physical Modelling Language Modelica*. Proceedings of the 9th European Simulation Symposium, ESS'97. Passau, Germany, 1997.

[10] Otter M., Årzén K., Dressler I. *StateGraph - A Modelica Library for Hierarchical State Machines*. Modelica 2005 Proceedings, 2005.

[11] Basu S., Pollack R., Roy M. Algorithms in Real Algebraic Geometry, In Springer (Eds), *Algorithms and Computation in Mathematics*, (10), 2ªedition, 2006.