A NEW PLANT MODELLING APPROACH FOR FORMAL VERIFICATION PURPOSES

José Machado[*], Eurico Seabra[*], Filomena Soares[**], José Campos[***]

[*]Mechanical Engineering Department, [**]Electronics Engineering Department, [***]Informatics Department; School of Engineering; University of Minho, Campus of Azurém, 4800-058 Guimarães, PORTUGAL

Abstract: This paper presents a new approach in plant modeling for the formal verification of real time systems. A system composed by two tanks is used, where all its components are modeled by simple modules and all the interdependences of the system's modular models are presented. As innovating parameters in the plant modeling, having as purpose its use on formal verification tasks, the plant is modeled using Dymola software and Modelica programming language. The results obtained in simulation are used to define the plant models that are used for the formal verification tasks, using the model-checker UPPAAL. The paper presents, in a more detailed way, the part of this work that is related to formal verification, being pointing out the used plant modeling approach. *Copyright © 2007 IFAC*

Keywords: Safe Systems, Real Time Systems, Plant Models, Formal Verification

1. INTRODUCTION

Safety control has, as its main goal, the assurance of the requirements of reliability, availability, and maintainability of automated systems[1]. Because of its direct impact on people and goods safety, the reliability of critical systems (transports, space, nuclear,...) has, since some time, mobilized the scientific community efforts. To assure the safety of a system, it is necessary to use a global approach (to guarantee that weaknesses do not exist) that takes into account a set of engineering activities and, later, after entering into functioning, the set of activities of exploitation and maintainability under conditions of operational exploitation of the system. Thus, the scientific community makes an effort to find, for the engineers, models, methods and tools, having as complementary objectives to anticipate possible predictions of malfunctioning.

Among the several techniques of industrial controllers analysis there are distinguished, for its utility, Simulation (Baresi *et al.* 2000) and Formal Verification (Moon, 1994). In the research works on industrial controllers analysis, these two techniques are rarely used simultaneously. If Simulation is faster to execute, has the limitation of considering only some system behavior evolution scenarios. Using Formal Verification has the advantage of testing all the possible system behavior evolution scenarios but, sometimes, it has the limitation of the time necessary for the attainment of formal verification results. This paper shows, how it is possible, and desirable, to conciliate these two techniques in the analysis of industrial controllers. With the simultaneous use of these two techniques, the developed industrial controllers are more robust and not subject to errors.

[1] 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.

This paper is focused on the formal verification of timed systems.

There are several approaches to applying formal verification techniques (not considering timed aspects) on automation systems dependability: from the formal verification by theorem proving (Roussel and Denis, 2002) to formal verification by model-checking (Rossi, 2004) and considering (Machado *et al.* 2006), or not (Rossi, 2004), a plant model. In the formal verification of timed systems there are several approaches to increase the obtained results in proving properties. Between these works there are distinctions between the work by (Remelhe *et al.* 2004), where the translation of IEC 61131-3 SFC into timed automata is studied, and the work by (Gaid *et al.* 2005) where a way for constructing the controller model is proposed, using the initial approach proposed by (Mader and Wupper, 1999), and some work hypotheses related with the controller model evolution and with time aspects, that increase the obtained results on formal verification tasks.

In this work we are adopting the base system and the work hypothesis considered by (Gaid *et al.* 2005) for the controller behavior and we propose a way to build plant timed models. To achieve our goals, in this work, the paper is organized as follows. Section 1, presents the challenge proposed in this work. Section 2 is devoted to the general presentation of the case study involving a system with two tanks, a heating device, level control sensors and valves to control the liquids flow and it presents also the methodology to obtain the controller program deduced from a specification of the system desired behavior. Section 3 is entirely dedicated to plant modeling, being presented the adopted approach. Next (section 4) a set of system behavior properties to prove is provided and also its formalization using TCTL. Section 5 presents and discusses the obtained results on formal verification using the model-checker UPPAAL and, finally, section 6, presents some conclusions and future work.

2. CASE STUDY

In this work a modified version of the benchmark example for an behavior evaporator system presented by (Kowalewski *et al.* 2001) and (Huuck *et al.* 2001) is used. The system (Fig. 1) consists of two tanks (one of them is heated and mixed), a condenser, level sensors and on-off valves (Vi). In the normal operation mode the system works as follows. Tank1 is filled with two solutions by opening valves V1 and V2. Then, the mixer starts working in order to promote the dilution. After two time units, the heated device is switched on for 20 time units to increase temperature solution. During this period part of the liquid is evaporated and cooled by the condenser. At that point the required liquid concentration has been reached and the heater is switched off. The remaining liquid is drained to tank2 by opening valve V3. The mixing device is switched off when tank1 is empty. The solution stays in tank2 for post-processing, to stay liquid, for 32 time units and then valve V4 is open to empty tank2.

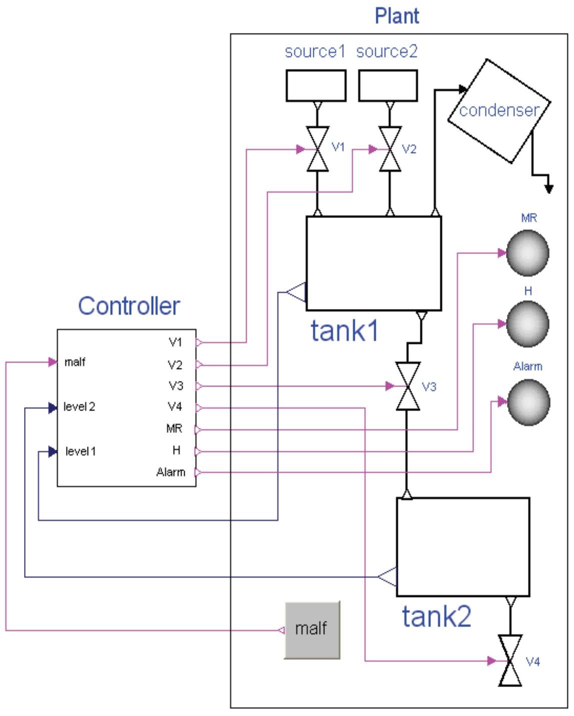


Fig. 1. Evaporator system. Closed-loop system composed by controller and plant.

Throughout normal operation mode, the system may malfunction. During evaporation, the condenser may fail: the steam can not be cooled and the pressure inside the condenser rises. Therefore, the heater must be switched off to avoid the condenser explosion. By doing so, the temperature of tank1 decreases and the solution may become solid and can not be drained in tank2. Hence, valve V3 must be opened early enough, but after opening first valve V4, for preventing tank2 overflow.

In the case of a condenser malfunction, we also need to guarantee some response times of the control program, taking into account the timing characteristics of the physical devices:

- whenever a condenser malfunction starts, the condenser can explode if steam is produced during 22 time units;
- if the heating device is switched off, the steam production stops after 12 time units;
- if no steam is produced in tank 1, the solution may solidify after 19 time units;
- emptying tank 2 takes between 0 and 26 time units;
- emptying tank 1 can be very fast with respect to the other durations, so that it is considered instantaneous;
- filling tank 1 takes at most 6 time units.

2.1 Controller specification

As we use the Simulation and Formal Verification, using different tools and intending to conciliate the obtained results, we adopted a controller specification that is the same for the basis of the controller program in the two analysis techniques. Thus, the controller specification was developed in IEC 60848 SFC because it can be used as the basis for the development of the Programmable Logic Controller program (PLC), to be verified with UPPAAL based on timed automata (Alur and Dill, 1990), and also it is the basis for the controller program to be used by StateGraphs Modelica library (Otter *et al.* 2005).

The input and output variables of the controller model are summarized on Table 1; minimum and maximum level sensors and malfunction sensor are considered as inputs and on-off valves and Heater, Mixer and Alarm are controller program outputs.

Table 1 Input and output variables of the controller program

Inputs	**Outputs**
T1F- tank1 full	V1- valve 1
T1E- tank1 empty	V2 - valve 2
T2F - tank2 full	V3 – valve 3
T2E - tank2 empty	V4 – valve 4
Malf - condenser malfunction	H- heater
	MR – mixer
	Alarm

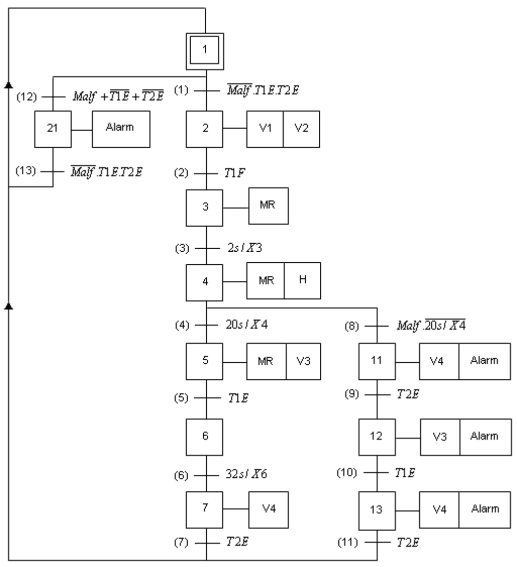


Fig. 2. IEC 60848 SFC specification of the controller program.

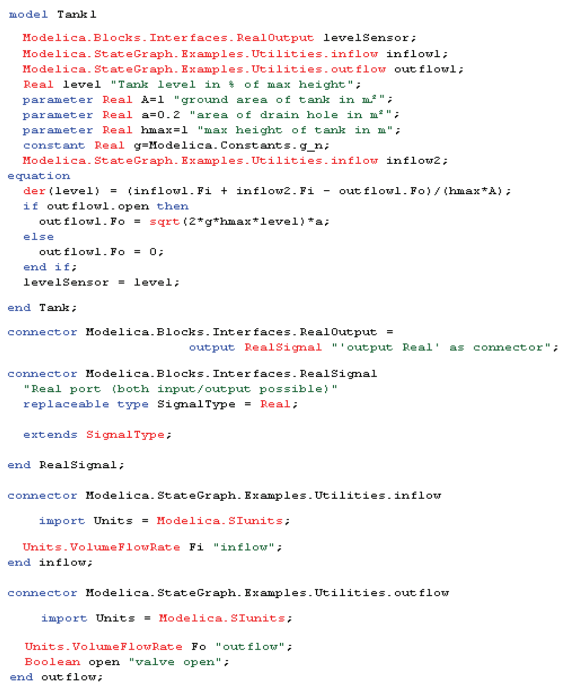
In order to guarantee the desired behavior for the described system, a IEC 60848 SFC specification is presented on Fig. 2. As IEC 60848 SFC is a specification language (and not a programming one), it is necessary to translate the SFC specification, first to a StateGraph program, presented in (Seabra et al. 2007) and, second, to translate it into a program written in a PLC programming language (in this case it will be used the ladder language).

3. PLANT MODELLING

The interesting point about the plant modeling was that it consisted of two steps: first, to model the plant using Dymola software and Modelica programming language (Elmqvist and Mattson, 1997), and, then, to use the obtained models as a basis for the development of the UPPAAL (David *et al.* 2003) models which are used on the formal verification tasks. We consider the following eight modules for the plant modeling: Tank1, Tank2, Heater, Mixer, Alarm, Steam, Condenser and Liquid. As this work is devoted to formal verification tasks, only the case of tank1, modeled using Modelica programming Language, and the corresponding UPPAAL model are presented. All the plant is modeled using Modelica programming language in (Seabra *et al.* 2007). It is remembered that the most important information that is taken into account is the set of simulation functioning delays that are obtained by simulation. These delays are, afterwards, used for the formal verification tasks, in order to define the time units used in the modules of the plant model.

3.1 Model of tank1

The model of tank1 is, first, simulated with Dymola, presented as Modelica code in Fig. 3.



```
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;
  Modelica.StateGraph.Examples.Utilities.inflow inflow2;
equation
  der(level) = (inflow1.Fi + inflow2.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 code for the tank1 model.

The obtained delays on simulation were used on formal verification with UPPAAL. The corresponding model of the tank developed in UPPAAL for formal verification purposes is presented in Fig. 4.

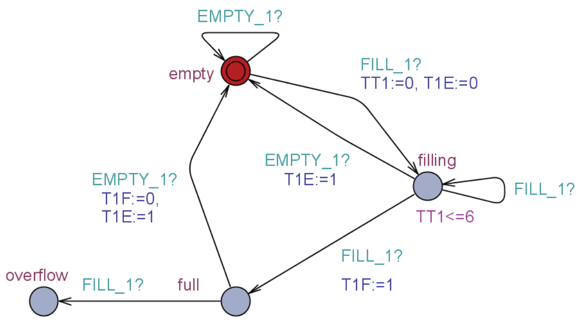


Fig. 4. UPPAAL model of the tank1.

We consider four states: *empty* models that tank1 is empty; *filling* models that the liquid is entering in tank1; *full* models that tank1 is full; state *overflow* is also considered, this is a possible state for the tank, but describes an undesired behavior. In this model, it is also considered that the tank1 is emptied in a very short time, when compared with the filling time. We have considered this time null. It is for that reason that the model goes from the *full* state directly to the *empty* state, without an intermediate state. The Boolean variables T1E and T1F are associated with *tank1.empty* and *tank1.full,* respectively. These variables represent the level sensors' signals sent by the sensors from the plant to the controller. The maximum time for filling tank1 is six time units.

3.2 Model of tank2

The model of tank2 is similar at the tank1 model and the reasoning followed to obtain this model was the same as presented before for obtaining the tank1 model. As empting tank1 is considered to take a short (null) time, the filling of the tank2 is done in the same conditions, since the liquid is transferred from tank1 to tank2. Four states are considered: *empty*, *full*, *emptying* and *overflow,* which is a possible state for the tank, but describes an undesired behavior. The variables T2E and T2F have the same behavior on the tank2 model as the T1E and T1F described above on the tank1 model. Empty tank2 takes, at maximum, twenty-six time units.

3.3 Models of Heater Mixer and Alarm

The reasoning adopted for building these three models is the same for all of them and consists in considering two states for each one: *off* (as initial state for all of them) and *on* that models that respective orders sent from the controller to the plant are actives. In Fig. 5 is presented the model of the mixer, for illustration of all these three models.

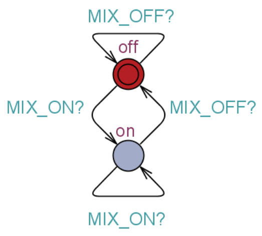


Fig. 5. Model of the mixer

3.4 Model of the condenser

The model of the condenser, presented in Fig. 6, is composed by five states: the state *good* models the good functioning of the condenser; the state *malfunction_heater* models that the condenser is malfunctioning and the heater is in state *on*; the state *malfunction_not_heater* models that the condenser is malfunctioning but the heater has been switched off; the state *before_explosion* models the behavior where it is not possible to avoid the condenser explosion and the state *explosion* models the behavior of the condenser explosion.

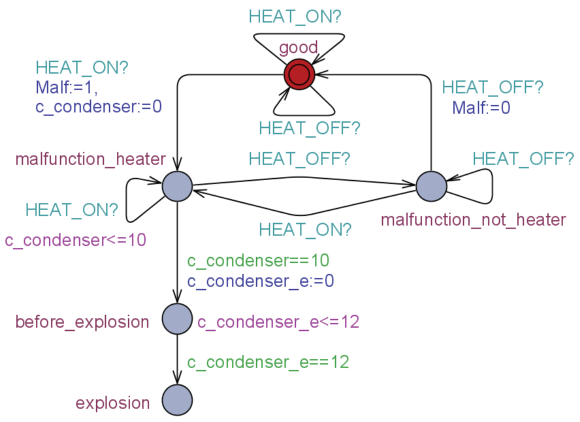


Fig. 6. Model of the condenser

In the behavior described in the case study it is presented that the condenser may explode if it is in malfunctioning behavior and if steam exists for during twenty-two time units after that. In this case, if the system behavior is the same as described in the *malfunction_heater* state (the heater is in the *on* state) during ten time units, then the condenser will explode because we will have steam at least for more twelve time units which implies that the condenser will explode (twenty-two time units after). The malfunction behavior happens in a random way from the state *good* of the condenser model: we have added to the condenser model a Boolean variable *Malf* that indicates that behavior.

3.5 Model of the steam

One of the hardest tasks on plant modeling is to model plant products (as the case of steam) because these models are specific for each plant. In this case, and taking into account the above described behavior for the system, the model of steam is extremely useful to allow us to prove some system behavior properties. We, thus consider three states for the model: state *off* models that steam does not exist; state *on_heat_on* models that steam exists and the heater is also in state *on* (the model of the steam is dependent of the model of the heater) and state *on_heat_off* models that steam exists but the heater is in the *off* state. We remember that after the heater switches off, steam exists during twelve time units. We have introduced the Boolean variable *steam_var* that indicates when steam exists or not.

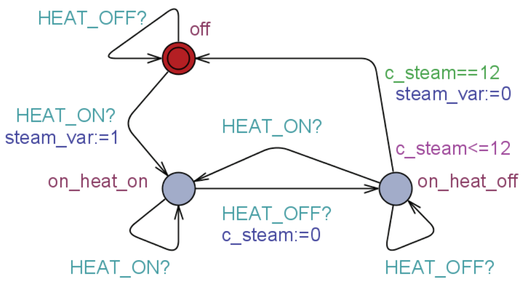


Fig. 7. Model of the steam

3.6 Model of the liquid

As the steam model presented before, the model of the liquid is also a specific model related with a plant product.

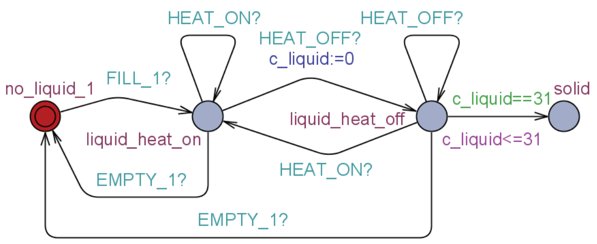


Fig. 8. Model of the liquid

This model has been created from the point of view of the presence of liquid in tank1 to look at its behavior regarding the solidification state of the liquid, possible in tank1. From point of view of the presence of liquid on tank1, Fig. 8, we have three possibilities: there is no liquid in tank1, modeled by state *no_liquid_1*; there is liquid in tank1 and the liquid is heating, modeled by the state *liquid_heat_on*; there is liquid in tank1 and the liquid is not heating, modeled by state *liquid_heat_off*; and there is liquid in tank1 that changed to the solid state after thirty-one time units (these thirty-one time units mean that the liquid solidifies after nineteen time units after the absence of steam; we remember that after the heater is switched off, the steam is present for more twelve time units).

We point out that, for all the models, we have adapted the simulation conditions to obtain the same delays proposed on (Gaid *et al.* 2005), in order to have a basis to compare the formal verification results obtained.

4. SYSTEM BEHAVIOR PROPERTIES AND PROPERTIES FORMALIZATION

In order to express the properties in UPPAAL syntax, we need a small part of TCTL formalism (Alur *et al.*, 1993). In the formulas below which are all (possibly timed) invariants, A is the universal quantifier on paths: for *any path*…, and [ ] means *always*… The combination A [ ] means *for all states in the future*…

The properties that we intend to prove are:

- **P1**: To prevent tank 1 from damage by overheating, it must be checked that the tank is full when the heater is working.
- **P2**: During the evaporation step, steam can leave tank 1 into the condenser only when the heater is on and the valves V1, V2 and V3 are closed.
- **P3**: To prevent uncontrolled liquid flow, the input and output valves of a tank must not be simultaneously open.
- **P4**: Condenser must not explode.
- **P5**: Solution must not solidify in tank 1.
- **P6**: Tank1 must not overflow.
- **P7**: Tank2 must not overflow.

The properties are formalized as follows, using TCTL:

- **P1**: A[ ] (H imply T1F);
- **P2**: A[ ] (H imply (not (V1) and not (V2) and not (V3)));
- **P3**: A[ ] not (((V1 || V2) and V3) or (V3 and V4));
- **P4**: A[ ] not condenser.explosion
- **P5**: A[ ] not liquid.solid
- **P6**: A[ ] not tank1.overflow
- **P7**: A[ ] not tank2.overflow

5. FORMAL VERIFICATION

The adopted strategy for formal verification considers that the program is written in Ladder language (according section 2) and some work hypotheses, as presented on (Gaid et al. 2005):

- the PLC executes the control program with a three phases cyclic behavior;
- there are considered timer blocks IEC 61131-3 TON;
- the duration of PLC cycle is considered to be between ε_1 and ε_2 and we have adopted ε_1

equal to 0 and ε_2 equal to 1;
- the unit of time used for the plant is irrelevant;
- discovering the maximal value of ε_2 for the properties to hold was not our objective.

The controller and the plant models, coupled as a closed-loop system, compose the global model verified. All the properties were verified in less than 2 seconds each, using. UPPAAL version 4.0.3 and an Intel Core Duo, 1,87GHz, machine with 4GB of RAM. The modular approach to building the plant model is well suited to facilitate the tasks of writing the properties, as is the case of properties P4, P5, P6 and P7.

6. CONCLUSIONS AND FUTURE WORK

The presented way to build plant models, from an automation system, is useful for two important points: first, we can avoid, using simulation, a set of program errors in reduced time intervals which would not happen if these errors were detected only through the use of formal verification techniques and, second, our modular approach allow us an easier way to build plant models which is well suited to facilitate the tasks of writing the properties formulae. The consideration of undesirable states on the created plant models allows us an easier writing of properties.

The use of Modelica programming language, to obtain these modular plant models, is useful to define the delays in which a property can, or cannot, be proved and the delays to elaborate the considered plant models.

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 that is true, for a delay, will become false for another. This study is not presented in this work but it is very important from the point of view of system safety.

REFERENCES

Alur R., C. Courcoubetis and D. L. Dill (1993). Model-Checking in Dense Real-Time, *Information and Computation*, **vol. 104**, n_ 1, pp. 2-34.

Alur R. and D. L. Dill (1990).,. Automata for modeling real-time systems. *Proc. of 17th Int. Coll. Automata, Languages, and Programming* (ICALP'90), England.

Baresi L., M. Mauri, A. Monti and M. Pezzè (2000). *PLCTOOLS: Design, Formal Validation, and Code Generation for Programmable Controllers*. Simulation software.

David A., G. Behrmann, K. G. Larsen and W. Yi (2003). A Tool Architecture for the Next Generation of UPPAAL, *Technical Report n. 2003-011,* Uppsala University, February. 20 pages.

Elmqvist E. and S. Mattson (1997). An Introduction to the Physical Modelling Language Modelica. *Proceedings of the 9th European Simulation Symposium, ESS'97*, Oct 19-23, Passau, Germany.

Gaid M., B. Bérard and O. Smet (2005). Verification of an evaporator system with UPPAAL. *European Journal of Automated Systems*, **vol. 39**, nº9, pp. 1079-1098.

Huuck R., B. Lukoschus and Y. Lakhnech (2001). Verifying Untimed and Timed Aspects of the Experimental Batch Plant. *European Journal of Control*, **vol. 7**, n_ 4, p. 400-415.

Kowalewski S., O. Stursberg and N. Bauer (2001). An Experimental Batch Plant as a Test Case for the Verication of Hybrid Systems. *European Journal of Control*. **vol. 7**, n_4, pp. 400-415.

Machado J., B. Denis, J.-J. Lesage (2006). A generic approach to build plant models for DES verification purposes. *Proc. of Wodes'2006 – 8th Workshop on Discrete Event Sistems*. July, 10-12, Ann Arbor, Michigan, USA.

Mader A. and H. Wupper (1999). Timed Automaton Models for Simple Programmable Logic Controllers. *Proc. Of the 11th Euromicro Conference on Real-Time Systems (ECRTS'99)*, York, UK.

Moon I. (1994). Modeling programmable logic controllers for logic verification. *IEEE Control Systems*, **vol. 14,** nº2, pp. 53-59.

Otter M., K. Årzén and I. Dressler (2005). StateGraph - A Modelica Library for Hierarchical State Machines. In: *Modelica 2005 Proceedings*.

Remelhe M., S. Lohmann, O. Stursberg and S. Engell (2004). Algorithmic Verification of logic Controllers given as Sequential Function Charts. *Proceedings of the IEEE Intemational Symposium on Computer Aided Control Systems Design*, Taipei, Taiwan.

Roussel J-M. and B. Denis. (2002). Safety properties verification of ladder diagram programs. *European Journal of Automated Systems*, **Vol. 36**, pp. 905-917.

Rossi O. (2004). Validation formelle de programmes ladder pour automates programmables industriels. *PhD thesis*, June, France.

Seabra E., J. Machado, F. Soares, C. Leão and J. Silva (2007). Simulation and formal verification of real-time systems: A case study. In: *ICINCO'2007 proceedings*, 9-12th may, Angers, France.