ABCM Symposium Series in Mechatronics - Vol. 3 - pp.461-470
Copyright © 2008 by ABCM

SIMULATION AND FORMAL VERIFICATION OF INDUSTRIAL SYSTEMS CONTROLLERS

José Mendes Machado, jmachado@dem.uminho.pt
Eurico Augusto Rodrigues Seabra, eseabra@dem.uminho.pt
University of Minho - Mechanical Engineering Department; Campus of Azurém, 4800-058 Guimarães, Portugal

José Creissac Campos, jose.campos@di.uminho.pt
University of Minho - Informatics Department; Campus of Gualtar, 4700-175 Braga, Portugal

Filomena Soares, fsoares@dei.uminho.pt
University of Minho - Electronics Engineering Department; Campus of Azurém, 4800-058 Guimarães, Portugal

Celina Pinto Leão, cpl@dps.uminho.pt
University of Minho - Industrial Engineering Department; Campus of Gualtar, 4700-175 Braga, Portugal

Jaime Carlos L. Ferreira da Silva, jaimefs@dem.uminho.pt
University of Minho - Mechanical Engineering Department; Campus of Azurém, 4800-058 Guimarães, Portugal

Abstract. *Actually, the safety control is one of the most important aspects studied by the international researchers, in the field of design and development of automated production systems due to social (avoid work accidents, ...), economics (machine stop time reduction, increase of productivity,...) and technological aspects (less risks of damage of the components,...). Some researchers of the Engineering School of University of Minho are also studying these aspects of safety control, using simulation and model-checking techniques in the development of Programmable Logic Controllers (PLC) programs.*
The techniques currently used for the guarantee of automated production systems control safety are the Simulation and the Formal Verification. If the Simulation is faster to execute, has the limitation of considering only some system behavior evolution scenarios. Using Formal Verification it exists the advantage of testing all the possible system behavior evolution scenarios but, sometimes, it exists the limitation of the time necessary for the attainment of formal verification results. In this paper it is shown, as it is possible, and desirable, to conciliate these two techniques in the analysis of PLC programs. With the simultaneous use of these two techniques, the developed PLC programs are more robust and not subject to errors. It is desirable the use of simulation before using formal verification in the analysis of a system control program because with the simulation of some possible system behaviors it is possible to eliminate a set of program errors in reduced intervals of time and that would not happen if these errors were detected only through the use of formal verification techniques. Conciliating these two techniques it can be substantially reduced the time necessary for the attainment of results through the use of the formal verification technique.
For the analysis of a system control program for simulation and formal verification it is used the Dymola for the Simulation (through the creation of system models with Modelica language) and UPPAAL (through the creation of system models with timed automata)[*].

***Keywords**: Dependable systems, Safety Control, Formal Verification, Simulation, Timed Systems.*

1. INTRODUCTION

When designing and implementing the control of complex manufacturing systems, automation engineers are required to check that the behavioral models and controller programs they develop indeed fulfill all application requirements, especially those related to dependability. Simulation and Formal verification methods are used, many times, separately to achieve these goals.

Among the several available techniques for the industrial controllers analysis, Simulation (Baresi *et al.* 2000, Baresi *et al.* 2002) and Formal Verification "Moon (1994)", "Roussel and Denis (2002)", can be distinguished due to their utility. In the research works on industrial controllers' analysis, these two techniques are rarely used simultaneously. If the Simulation is faster to execute, it presents the limitation of considering only some system behavior evolution scenarios. Formal Verification presents the advantage of testing all the possible system behavior evolution scenarios but, sometimes, it takes a large amount of time for the attainment of formal verification results. In this paper it is shown, as 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.

Using this approach, the command of those systems can be simulated and tested when the physical part of the machine still does not exist. This way of simulation allows to reduce the production times of the automation systems because the manufacture do not need the physical part of the machine for later perform tests and simulation of the

[*] 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.

command of the system. This paper is focused in giving an overview how it is possible to use these two techniques in simultaneous.

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 system with two tanks, a heating device, a mixer device, level control sensors and valves to control the liquids flow. Further, it is presented the methodology to obtain the controller program deduced from an IEC 60848 SFC specification of the system desired behavior. Section 3 is exclusively devoted to the plant modeling, being presented the adopted approaches. In sections 4 and 5 are presented, respectively, the simulation and formal verification results and, finally, in section 6 are presented some conclusions and there are given some indications about possible future work.

2. CASE STUDY BEHAVIOR DESCRIPTION

In this work a modified version of the benchmark example for an evaporator system presented by "Kowalewski *et al.* (2001)" and "Huuck *et al.* (2001)" is used. The system (Figure 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.

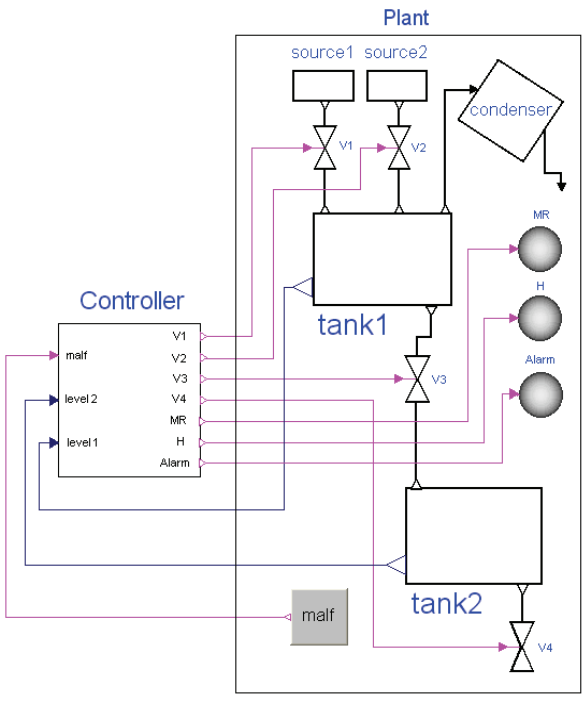


Figure 1. Evaporator system. Closed-loop system composed by controller and plant.

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 of the tanks and the Malfunction sensor are controller program inputs and the on-off valves (V1, V2, V3 and V4), the Heater, the Mixer and the Alarm are controller program outputs.

Table 1. Input/Output variables of the controller

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

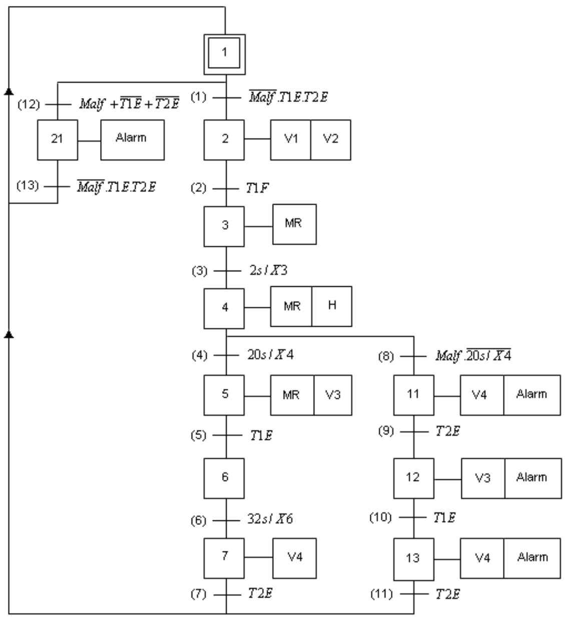


Figure 2. SFC specification of the controller

In order to guarantee the desired behavior for the described system, a IEC 60848 SFC specification is presented in Figure 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). This translation is done using a methodology, having as base the specification algebraic representation and considering also the controller program behavior presented at the next sub-chapter.

2.2. Controller program and respective behaviour

In that case, the program will encompass three modules, which will be sequentially executed (Figure 3): computation of clearing conditions of the transitions, computation of the step variables, and computation of actions.

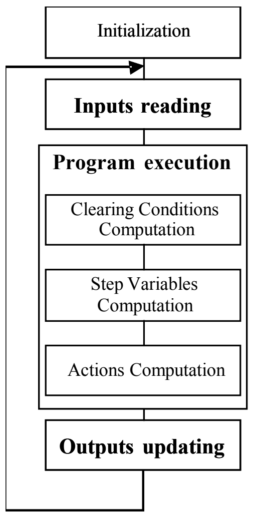


Figure 3. Cyclic scan monitor of the PLC.

Let CC(q) (Clearing Condition) a Boolean variable associated to each transition of a SFC. A transition q (Figure 4) can be cleared if it is enabled (all the steps that precede immediately this transition are active) and if its associated transition condition TC(q) is true. So, in a general case CC(q) can be formulated as follows:

$$CC(q) = (\prod_{j=1}^{m} X_j).TC(q)$$

with:

- X_j: step Boolean variable associated to step j,
- TC(q): Transition Condition associated to the transition q,
- m : number of steps that precede immediately step j.

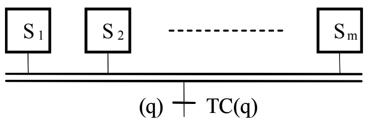


Figure 4. Transition condition after simultaneous sequences

According to the IEC 60848 evolution rules, the Boolean step variable X_i associated to each SFC step i can be computed in the following manner:
with:

- $X_i(t)$: value of the step variable of step i for the t^{th} scan cycle,
- $X_i(t+1)$: value of the step variable of step i for the $(t+1)^{th}$ scan cycle,
- p: number of transitions that precede step i,
- n: number of transitions that follow step i,
- $CC(p_j)$: Clearing Condition of the transition (p_j),
- $CC(n_k)$: Clearing Condition of the transition (n_k).

The step activation/de-activation is done by:

$$X_i(t+1) = \sum_{j=1}^{p} CC(p_j) + X_i(t).\prod_{k=1}^{n} \overline{CC(n_k)}$$

In the case of step 2 of the above SFC, for instance, it comes then:

- $CC(9) = X_{11}.T2E$ $\quad\quad$ $CC(10) = X_{12}.T1E$
- $X_{12}(t+1) = CC(9) + X_{12}(t) \ . \ /CC(10)$

Computation of actions

Each action is set when the logical OR of the step variables of the steps to which this action is associated is true. For instance:

- V1 (t) = $X_1(t)$;
- Alarm(t) = $X_{11}(t) + X_{12}(t) + X_{13}(t) + X_{21}(t)$

With all the rules presented above we construct the Ladder program base, on the deduced equations, elaborated following the steps presented on Figure 3.

3. PLANT MODELING

For the development of the formal verification tasks it is considered the controller model (developed according the rules described before) and the plant model. The two models are connected according the Figure 5 in a closed-loop configuration. The plant model elaboration is treated in this chapter.

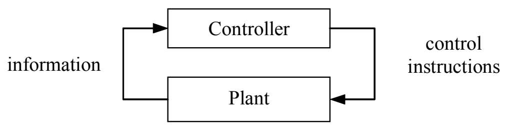


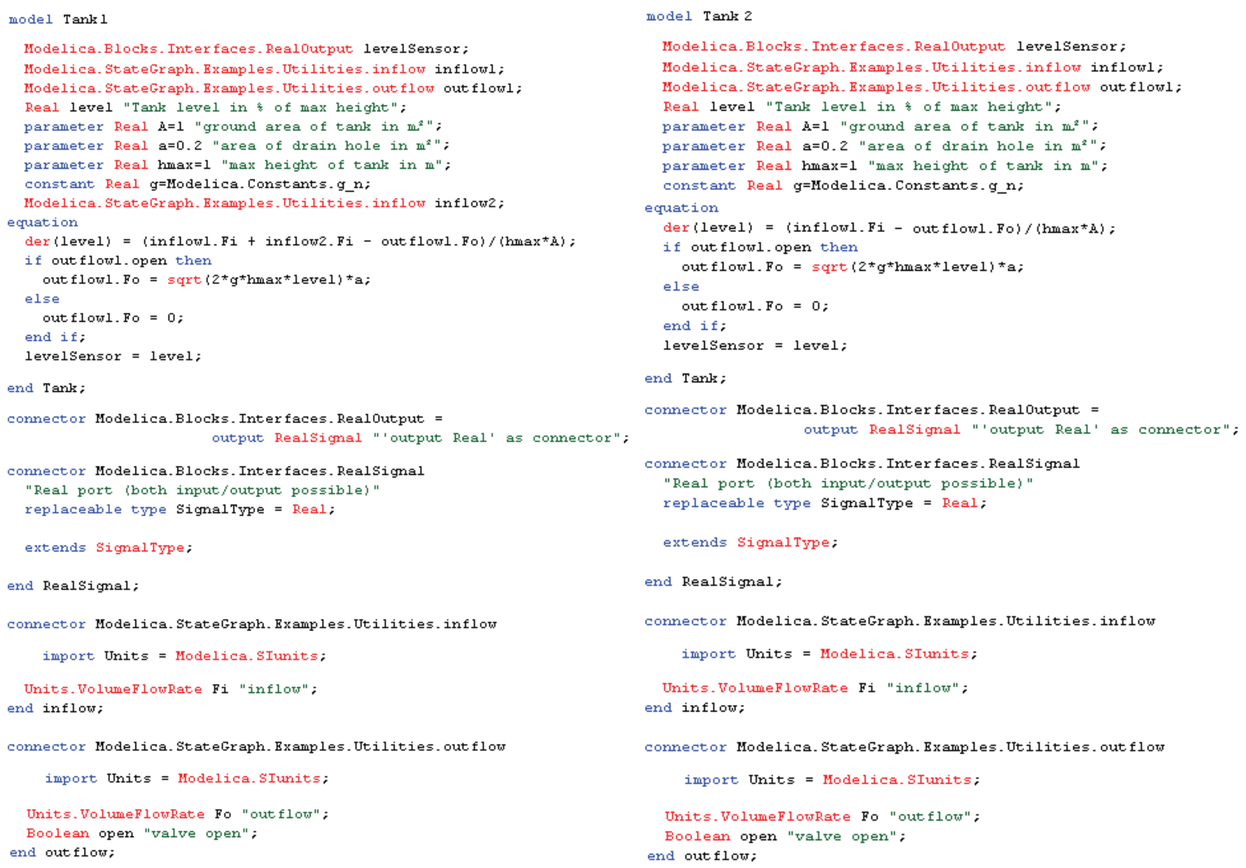
Figure 5. Closed-loop system composed by the controller and the plant.

In the plant modeling, first, the plant is modeled with the Modelica programming language “Elmqvist and Mattson, (1997)” and simulated with the Dymola software and, second, it is modeled by timed automata to be used as input of the UPPAAL software “David *et al.* (2003)”. The delays obtained, in the simulation with the Dymola software, are used to create the timed automata that are used on Formal Verification with the UPPAAL tool.

3.1. Plant modelling for simulation purpuses

All the system was modeled. The tank1 and tank2 models are presented on this sub-chapter. Lets consider the case of the tank 1 we have, for the Modelica programming language the model presented on Figure 6.

The Modelica program code for modeling the tank2 is similar to the code obtained for the tank1 model. The main difference between these two codes is due to the tanks have different numbers of fill sources. The tank 1 has two fill sources while the tank 2 has one only.



```
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;
```

```
model Tank 2

  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;
```

Figure 6. Modelica code for tank1 and tank2 models

3.2. Plant modeling for formal verification purpuses

For modeling the plant with formal verification purposes there are considered the following eight modules for the plant modeling: Tank1, Tank2, Heater, Mixer, Alarm, Steam, Condenser and Liquid. Only the tank1, tank2 and condenser models are presented here.

<u>Model of tank1</u>

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 Figure 7.

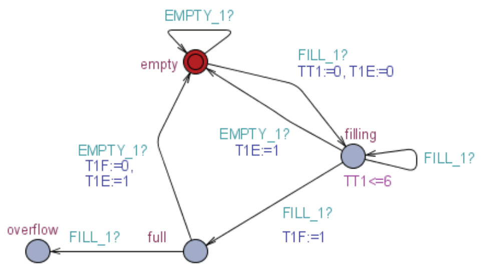


Figure 7. UPPAAL model of tank1.

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 behaviour. 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.

Model of tank2

The model of tank2 is presented on Figure 8 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.

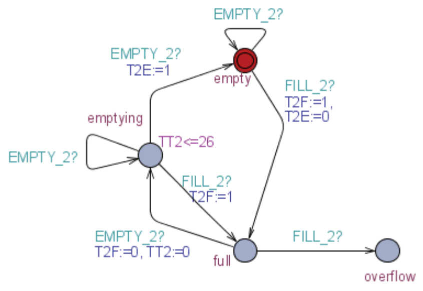


Figure 8. UPPAAL model of tank2

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.

Condenser model

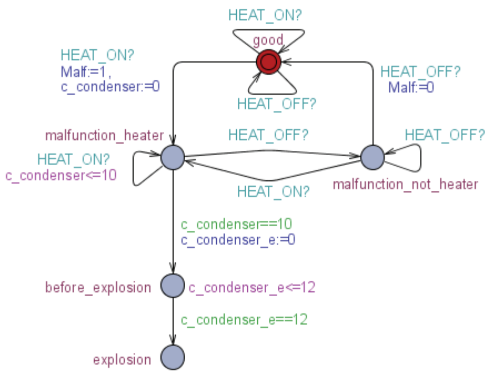


Figure 9. Model of the condenser

The model of the condenser, presented in Figure 9, 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. 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.

4. SIMULATION RESULTS

In order to perform the simulation, it is necessary to define the parameters, start and stop time of the simulation, the interval output length or number of output intervals and the integration algorithm. In the present work, in all simulations performed, the Dass algorithm "Basu *et al.* (2006)" with 500 output intervals was used.

In order to study the system behavior different values for physical variables of the plant were used. Table 2 shows the variables considered in the simulation of the system behavior.

Table 2: Variables of the plant.

Plant	**Variable**
source1, source 2	Q1, Q2 - 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 first two simulation performed were devoted to verify if the SFC of the controller system (Figure 2) modeled with Modelica language with the library for hierarchical state machines StateGraph simulated correctly the evaporator system, respectively, in their normal and malfunction operation. The values for the plant variables considered in these simulations were Q1=1, Q2=0.5, G1=G2=1, Ht1=Ht2=1, A1=0.2 and A2=0.05.

Figure 10 shows results of the simulation without the occurrence of the condenser malfunction during the production cycle, which corresponds to the normal operation, respectively, for the level tanks and for the controller outputs. It can be also observed, in the same figure, that the system is properly simulated by the developed program, since during the time specified by the SFC the tanks remain filled and empty, as well as, the switch logical state of the controller outputs.

On the other hand, Figure 11 shows results of the simulation with the occurrence of the condenser malfunction during the production cycle, which corresponds to the malfunction operation. The malfunction occurred in a random way 15s after the start of the plant functioning. Analyzing Figure 11 it can be also concluded that the proposed program properly simulates the malfunction operation. Because it can be verified, taking into account the figure 11, that at the malfunction occurrence (time 15s) the solution present in the tank1 is immediately drained for the tank2 and later emptied. In the same way, analyzing the Figure 11, it can be verified that at the time 15s occur simultaneous the switch off the mixer and the heater and the alarm switch on, which corresponds to the SFC specification of the controller.

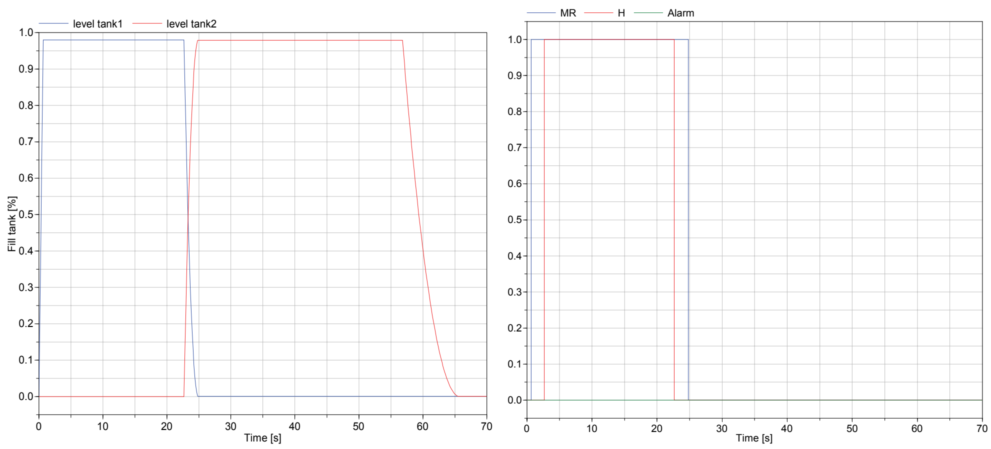


Figure 10: Level tanks in function of time in normal operation of the evaporator system and Switch state of the mixer, heater and alarm in normal operation of the evaporator system.

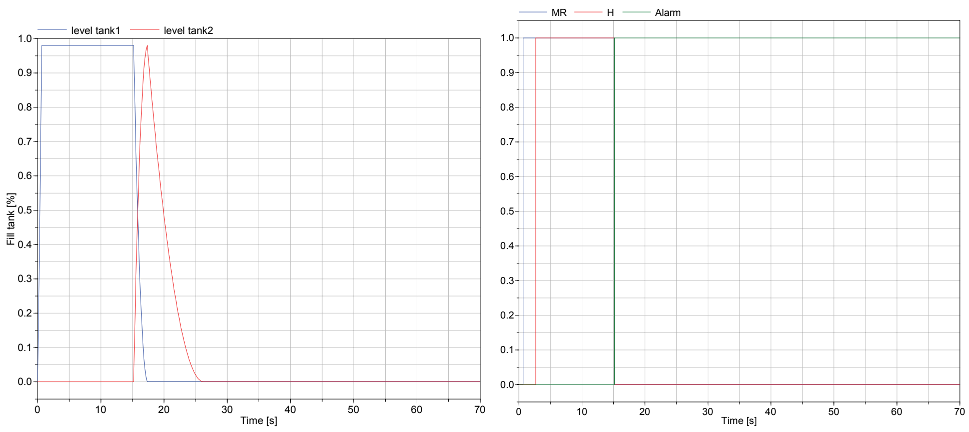


Figure 11: Level tanks in function of time with occurrence of condenser malfunction (time = 15s) and Switch state of the mixer, heater and alarm with occurrence of condenser malfunction (time =15s).

It becomes still necessary, in addition to the verification that the modeling of the system, obey to the SFC of the system controller, to guarantee that in the case of condenser malfunction does not occur explosion or solidification of the solution in the tanks. Thus it is necessary taking into account the timing characteristics of the physical devices.

5. FORMAL VERIFICATION RESULTS

This paper is oriented for the formal verification of the controller properties. In "Machado *et al.* (2007)" there are described some important results of the proof of the system behavior properties, that are not studied, in detail, in this paper. It exists a big set of the system behavior properties that were studied on the context of a much more large work, but here, in this paper, there are presented only three of these properties that are related with the timed automata plant models presented in sub-section 3.2 of this paper.

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 [] p, means *for all paths p always hold* and E <> p, means *there exists a path where p eventually hold*.

The proof of the liveness properties presented for the controller behavior is stronger if it is used a complete plant model "Machado (2006)".

The controller properties that we intend to prove are:

- **PC1**: The controller does not deadlock;
- **PC2**: The step X1 of the controller is reachable;
- **…**
- **PC8**: The step X7 of the controller is reachable;
- **PC9**: The step X11 of the controller is reachable;
- **…**
- **PC12**: The step X21 of the controller is reachable;

The system behavior properties to prove are:

- **PS1**: Condenser must not explode.
- **PS2**: Tank1 must not overflow.
- **PS3**: Tank2 must not overflow.

The properties are formalized as follows, using TCTL:

- **PC1**: A[] !deadlock
- **PC2**: E<> X1;
- **...**
- **PC8**: E<> X7;
- **PC9**: E<> X11;
- **...**
- **PC12**: E<> X21;

- **PS1**: A[ ] not condenser.explosion;
- **PS2**: A[ ] not tank1.overflow;
- **PS3**: A[ ] not tank2.overflow;

For the property PC1 the time necessary ot verify the property is about 10 seconds and all other 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 formulae related with the system behavior, as is the case of properties PS1, PS2 and PS3.

6. CONCLUSIONS

It is possible, and desirable, to conciliate the use of the Simulation and Formal Verification techniques in the analysis of industrial controllers. With the simultaneous use of these two techniques, the developed controllers programs are more robust and not subject to errors. It is desirable the use of simulation before using formal verification in the analysis of a system control program because with the simulation of some possible system behaviours it is possible to eliminate a set of program errors in reduced intervals of time and that would not happen if these errors were detected only through the use of formal verification techniques. Conciliating these two techniques it can be substantially reduced the time necessary for the attainment of results through the use of the formal verification technique.

Also for obtaining the plant models, the simulation delays, obtained, are used to build the timed automata models of the plant that are the input of the UPPAAL software.

As future work it will be analyzed the possibility of studying the robustness of industrial controllers taking into account the Human-Machine interface.

7. REFERENCES

Alur R., Dill D. L 1990, "Automata for modeling real-time systems" Proc. 17th Int. Coll. Automata, Languages, and Programming (ICALP'90), Warwick University, England, July 1990, vol. 443 of Lecture Notes in computer Science, Springer, p. 322-335.

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

Baresi L., Mauri M., Monti A., Pezzè M., 2000, "PLCTOOLS: Design, Formal Validation, and Code Generation for Programmable Controllers", Special Session at IEEE Conference on Systems, Man, and Cybernetics. Nashville USA.

Baresi L., Mauri M., Pezzè M., 2002, "PLCTools: Graph Transformation Meets PLC Design", Electronic Notes in Theoretical Computer Science 72 No. 2.

Basu S., Pollack R., Roy M., 2006, "Algorithms in Real Algebraic Geometry - Algorithms and Computation in Mathematics", Springer Editions, vol. 10, 2nd edition.

David A., Behrmann G., Larsen K. G., Yi W., 2003, "A Tool Architecture for the Next Generation of UPPAAL" Technical Report n. 2003-011, Department of Information Technology, Uppsala University, February, 20 pages.

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

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

Kowalewski S., Stursberg O., Bauer. N. 2001, "An Experimental Batch Plant as a Test Case for the Verication of Hybrid Systems", European Journal of Control.

Machado J., Seabra E., Soares F., Campos J., 2007, "A new Plant Modelling Approach for Formal Verification Purposes" Accepted for oral presentation at 11th IFAC/IFORS/IMACS/ IFIP Symposium on Large Scale Systems: Theory and Applications. Gdansk, Poland.

Machado J., 2006, "Influence de la prise en compte d'un modèle de processus en vérification formelle des Systèmes à Evénements Discrets", PhD Thesis in cooperation between the University of Minho and École Normale Supérieure de Cachan; School of Engineering, University of Minho, June 2006.

Moon I. 1994 "Modeling programmable logic controllers for logic verification", IEEE Control Systems, 14, 2, pp. 53-59.

Otter M., Årzén K., Dressler I., 2005, "StateGraph - A Modelica Library for Hierarchical State Machines", Modelica 2005 Proceedings.

Roussel J.-M., Denis B., 2002 "Safety properties verification of ladder diagram programs" Journal Européen des Systèmes Automatisés, vol. 36, pp. 905-917.

Seabra E., Machado J., Soares F., Leão C., Silva J. 2007, "Simulation and formal verification of real-time systems: A case study" Accepted for oral presentation at ICINCO'2007 – 4th International Conference on Informatics in Control, Automation and Robotics. 9-12th may, Angers, France.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.