Timed Automata Verification Applied to Performance Analysis of Control Logics Synthesized by Supervisory Control Theory

João Aurélio V. Rodriguez and Antonio E.C. da Cunha
Postgraduate Program in Electrical Engineering, Military Institute of Engineering, Rio de Janeiro 22290-270, Brazil
Corresponding author: João Aurélio V. Rodriguez (joao.rodriguez@outlook.com)

Abstract: In this work, we apply two techniques related to the DES (discrete event systems), the SCT (supervisory control theory) and the formal verification of timed DES, to treat a problem of coordination among different processes of a real industrial system, a window bonding system in a production line of truck cabins. We have first represented the window bonding system as a DES, defining the components and the events for coordination. We have then employed deterministic finite state automata to build models for the system components and coordination specifications. In this model, the timing information was abstracted into artificial timer events. We have then applied the SCT to synthesize a control logic in the form of local modular supervisors to ensure the coordination specifications. By applying the SCT, we have a formal guarantee that the designed control logic attains to the desired coordination specifications while being non-blocking and minimally restrictive. In the designed control logic, some of the coordination specifications that involved time constraints could not be satisfied, issue caused by the abstraction of the time. Moreover, we were willing to have a form to quantify the benefits of the claimed maximally permissivity of the designed control logic. We have then applied the formal verification of timed DES to make the performance evaluation of the closed loop system. In this approach, we have employed Timed Automata to represent the supervisors and the system components, incorporating dense time information. Monitor timed automata and temporal logic specifications in CTL were developed to build verification scenarios related to the performance evaluation. The results of the formal verification were applied to measure the productivity of the various processes of the closed loop system within a time frame.

Key words: Discrete event systems, supervisory control, verification, validation, performance analysis.

1. Introduction

The SCT (supervisory control theory) initiated by Ramadge and Wonham in the mid-80s deals with the synthesis of control logic for discrete-event systems (DES) [1]. The SCT provides an automatic method to generate control logic, called supervisor, given the DES model to be controlled, called plant, and the specifications for the closed loop behavior. The models and specifications are expressed by languages and automata [1].

The control logic synthesized by SCT is formally assured to be “safe”, in the sense of meeting the system closed loop specifications, optimal, in the sense of being minimally restrictive for the system behavior, and “non-blocking”, in the sense of guaranteeing the absence of “deadlocks” and “livelocks” [1].

However, we perform many abstractions in modelling task for control synthesis, as in this work, the abstraction of the passage of time. On the other hand, it is not explicitly provided by SCT ways assess the performance of the system under feedback control loop before its implementation. This would allow quantifying, for instance, the control logic gains due to its minimally restrictive characteristic.

In this paper, we propose a way of using timed automata verification [2] to provide a form of
performance analysis for the control logic generated by SCT. This task could be performance, for instance, using discrete-event simulation. However, by running simulations for different scenarios, we can only infer the good behavior of the system by analyzing specific discrete-event traces. And this task is not exhaustive. Differently from discrete-event simulations, the formal verification allows us to analyze the whole system behavior against a set of possible initial conditions and a set of possible uncontrollable inputs for the system.

We work with the case study of an industrial automation system of truck cabins glazing at MAN Truck and Bus plant in Munich, Germany. The motivation of the choice of this system was the possibility to optimize the bonding process, because the system was originally developed only focusing on its functionality on the production line. Moreover, the system presents interesting aspects of timing to be investigated.

For the synthesis of the control logic, we work with the LMC (local modular control) approach, as proposed by Ref. [3]. We have adopted this approach without considering an explicit timing of events. In order to do so, we use fictitious events, corresponding to the end of counting of certain relevant times. It is an ongoing work the application of the supervisory control of timed DES along with both approaches of Refs. [4, 5].

For the verification of timed automata we have used the tools available in the UPPAAL toolkit [6].

The presentation of this paper is as follows: Section 2 presents the windshield gluing system and the control problem; Section 3 shows the implementation of LMC to the problem of control the windshield gluing system; Section 4 registers the application of verification of timed automata for validation and performance analysis of the control logic; Finally, Section 5 presents some concluding remarks and perspectives for future work.

2. System Description

The system for bonding glass in the truck cabins of the MAN Truck and Bus factory in Munich (Germany) was designed for the application of glue on the rear side windows (side glass) and in the windshield as well as its mounting on the cabin. The system was motivated by the precision of the robots in the application of glue, not only in relation to the coordinates, but also in relation to the volume of glue applied. Another determining factor was the issues of size (2,242.6 mm × 879.5 mm) and weight (29.3 kg) of the windshield glass, which restrain their placement in the cabin by a single employee.

The application of glue to the glass goes through three steps that are: the application of the activator, application of the primer and finally the glue itself. It must be respected the minimum drying times between the activator/primer and primer/glue applications as well as after application of the glue there is a maximum time for the placement of the windows in the cabin.

As the side glass are applied manually in the cabin, it is produced more side glasses than the windshields and they can be applied to the cabin two workstations prior to the workstation that mounts the windshield glass and one workstation after. Due to this flexibility, the drying time, or curing, of the glue applied to the side glass is larger than the glue applied on the windshield.

The glass bonding system, illustrated in Fig. 1, is composed by

- Two robots, the SGR (side glass robot) and the WSR (windshield robot);

![Fig. 1 Truck cabin’s bonding system.](image)
Two intermediate buffers for depositing glass, B1 and B2;

Three matts, one for SGI (side glass input), one for WSI (windshield input) and one for SGO (side glass output);

A cabin optical coordinates measurement system, called PER (Perceptron);

The entry of cabins (EC) of the main production line.

In Fig. 1, the main components of the system will be described.

The SGR is used to apply the primer and the activator in the side glass. Assuming that SGI is always reloaded by the operator of production, the SGR cups a pair of side glass, puts them in position for application of the activator and starts its application. After that, the robot waits 20 s, while, at the same time, makes the process of self-cleaning and changing nozzle for application of the next product (the primer). The robot then applies the primer and puts the side glass in buffer B1 to wait for the drying time of 45 s.

The WSR is responsible for three different processes described as follows:

The first process consists of the withdrawal the side glasses from B1 through a cupping glass, and to position them for the application of glue. The robot then applies the glue and releases them in the SGO mat, so that the operator can pick them up to apply to the cabin;

The second process is the withdrawal of the windshield from the WSI mat to apply the primer on it. The windshield is already available on WSI, uploaded beforehand by a production operator, with the activator manually applied. The robot then applies the primer and deposits the windshield into buffer B2 to wait the minimum time of 45 s for drying;

The third process consists of the withdrawal the windshield from buffer B2 and positioning it for the application of glue. This operation can only occur after WSR receives the cab coordinates that comes from the resulting of the measurement made by PER. After the windshield being positioned to apply the glue and WSR received the cab coordinates, the application of glue starts. After applying the glue, the robot positions the windshield and makes the junction of it to the cabin.

The Perceptron is used to perform the measurement of the principal coordinates of the cab and send them to the robot WSR. The measurement process occurs after the cab arrives at the work station and an alignment platform coupled to the cabin (position A).

Buffer B1 is responsible for temporary storage of side glasses as well as the positioning of them so that WSR can pick them up. This position consists on rotation of a cradle of + or -270°.

At position A there exists a reference platform that it is used for measuring. This guarantees no variations in the position of the cabin at the time of measurement, and also sustains the application of the windshield to the cabin, referencing it in X, Y and Z axes.

The production line has a maximum time for treatment of a cabin by the bonding system of 150 s, which when violated, force the triggering of a production line emergency stop.

The control objective is the concurrent coordination of the entry cabins and the operation of the robots for application of the activator, primer and glue to the side glass, application of primer and glue on the windshield and its placement in the cabin, all respecting the operational timing restrictions of system.

Some important points should be taken into account for this process:

The windshield has priority over the side glass as soon as the cab is in position A, due to the importance of the TAKT time (production line time). When this priority is not respected, it can cause the stop of the production line.

The application of glue on the windshield and subsequent application to the cabin begins only after it is positioned (position A) and measured by Perceptron.
The vulcanization time (drying) of the glue applied to the windshield is short, so the anchoring of the windshield can occur in the shortest possible time (5-10 s).

If the measurement made by the Perceptron is not within the acceptable process parameters for placing the windshield by the robot, as cabin skew or dimensional outside the tolerances, the glue is not applied to the windshield and this cabin is removed from the flow to a later failure analysis and application of the windshield manually.

3. TCS Application

In this paper, for the synthesis of the control logic in the context of TCS, we have employed methods associated with CML (local modular control) as proposed by Ref. [3].

The first step is the modeling of the plant. Some simplifying assumptions were considered, as described below:

- First, the supply of glasses is much faster than the processing time of the system, and there is always available a side glass and a windshield;
- Second, the side window deposited in SGO is immediately forwarded;
- Third, PER is triggered on the arrival of the cabin in position A, and the event that marks that the cabin is ready to receive the windshield corresponds to sending the coordinates to the robot WSR.

The consequence of these simplifying assumptions is that the components of interest to be modeled in the system are SGR, WSR and EC, which are modeled respectively by automata G1, G2 and G3 in Fig. 2. It is assumed that the reader is familiar with the standard graphical notation of automata [1].

For modeling without consideration of the timing of events, two operational restrictions are necessary. The ROp1 and ROP2 restrictions, illustrated in Fig. 2, are abstractions of timers for counting the minimum drying time of the primer that should occur, respectively, to the side window after filing in the buffer B1 and the windshield after filing in the buffer B2.

In SCT, the events are partitioned into controllable and uncontrollable. Controllable events can have their occurrence inhibited or disabled, while uncontrollable events are always enabled [1]. Table 1 shows the description of the events for the system, their controllability attribute (C for controllable and NC for non-controllable) and an indication of the delay time for the event, which will be used further in Section 4.

The next step in the method is the modeling of each specification isolated, just considering relevant events [3]. The specifications for the coordination and proper operation of the system are as follows:

- E1: Avoid overflow (put more than one glass) and underflow (try to pick up the glass without having any) in buffer B1;
- E2: Avoid overflow and underflow in the buffer B2;
- E3: Attend the minimum drying time of the primer in B1;
- E4: Attend the minimum drying time of the primer in B2;
- E5: Do not release the cabin without having the windshield glued on it;
- E6: Avoid placing the windshield in a vacuum;
- E7: Avoid timeout on the production line.

Automata SP1, SP2 and SP3 in Fig. 3 were designed in order to meet the previous specifications. In Fig. 3, automaton SP1 expresses that, after the deposit of the side windows by SGR at B1 (event b1),
Table 1  Summary table of the event’s description.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Control</th>
<th>Delay (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>SGR command to get two side windows at SGI.</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>b1</td>
<td>SGR takes two side windows from SGI; apply activator and waits 20s; switches the felt of the activator application; positioning the nozzle of the primer; applies the primer; puts the two side windows at B1; B1 rotates 270° and turn to WSR.</td>
<td>NC</td>
<td>50</td>
</tr>
<tr>
<td>a21</td>
<td>WSR command to get two side windows in B1.</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>b21</td>
<td>WSR takes two side windows from B1; applies the glue; deposit the side windows on the exit mat; B1 rotates 270° and turn to SGR.</td>
<td>NC</td>
<td>20</td>
</tr>
<tr>
<td>a22</td>
<td>Command WSR to get the windshield glasses in WSI.</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>b22</td>
<td>WSR pick up the windshield in WSI; applies the primer; and deposits the windshield in B2.</td>
<td>NC</td>
<td>30</td>
</tr>
<tr>
<td>a23</td>
<td>Command WSR to get the windshield glasses in B2.</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>b23</td>
<td>WSR pick up the windshield in B2; positions the glue applicator nozzle; apply the glue; and positioning the windshield in the cabin.</td>
<td>NC</td>
<td>40</td>
</tr>
<tr>
<td>a3</td>
<td>Command to release the cabin from position A and get a new cabin.</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>b3</td>
<td>Go down the elevator and releases the cabin from position A; presents a new cab and it is elevated into position A; Perceptron measures the coordinates of the new cab and send it to the WSR controller.</td>
<td>NC</td>
<td>50 a 100</td>
</tr>
<tr>
<td>c3</td>
<td>Timeout to meet the restrictions of the assembly line (TACKT time ~ 2.5 min.).</td>
<td>NC</td>
<td>150</td>
</tr>
<tr>
<td>t1</td>
<td>Timeout of 45s timer after the side glass be placed at B1.</td>
<td>NC</td>
<td>45</td>
</tr>
<tr>
<td>t2</td>
<td>Timeout of 45s timer after the windshield be placed at B2.</td>
<td>NC</td>
<td>45</td>
</tr>
</tbody>
</table>

Fig. 3  Specifications.

It must wait for at least 45s (t1) so that WSR can withdraw it (a21). Thus SP1 meets specifications E1 and E3. Automaton SP2 was constructed analogously to SP1 to meet E2 and E4. Automaton SP3 expresses that WSR only does the placement of the windshield (b23) after receiving of the cabin’s coordinates (b3), that another placement will only occur when there is another cabin available, and that the withdrawal of the side windows of B1 by WSR is impeded after the arrival of the cabin (selfloop of a21 in state 0). Thus SP3 aims to meet specifications E5 and E6 directly, and to meet E7 indirectly by prioritizing the placement of the windshield over the application of the glue in the side windows. This was necessary because the modeling of specification E7, in a straightforward way, would lead to an uncontrollable behavior: without considering the timing of events, it is not possible to prevent the occurrence event c3 in the initial state of G3.

The next step is to obtain the local plants for each specification, through the junction of the plant components that have events in common with the respective specification. The composed behavior of automata is obtained by the synchronous composition operation, sync(), where the components evolve in parallel having the transitions with common events synchronized [1]. Components that are part of local plants Gloci to specifications relating thereto SPi and the number of states of local plants Gloci are shown in Table 2.

The next step is the calculation of the automaton representing the behavior of each local plant that meets its specification. For SPi specification and the
local plant Gloci such automaton is obtained by $R_i = \text{trim} \left( \text{sync}(S_{Pi}, Gloci) \right)$, where $\text{trim}()$ denotes the trim operation [1]. The “$R_i$” column of Table 2 shows the number of states of the respective automata $R_i$.

Then, we compute the automata $Z_i$, that represents the behaviors contained at $R_i$ that are controllable, in the sense that the control action respects the restriction of the existence of controllable and uncontrollable events at Gloci, and minimally restrictive to the behavior of the local plants [1]. The “$Z_i$” column of Table 2 shows the number of states of automata $Z_i$.

In order to apply the modular supervision, the supervisors $Z_i$, $i = 1 \ldots 3$, have to be non-conflicting [3]. For this, we perform the local modularity test, which is accomplished by computing the synchronous composition $Z_c = \text{sync}(Z_1, Z_2, Z_3)$, and checking whether the resulting of automaton $Z_c$ is trim [3]. For the set of calculated supervisors, we have obtained as result a trim automaton $Z_c$, with 88 states and 280 transitions.

Finally, we can compute the reduced local modular supervisors, using the methods as in Ref. [7]. The “$Z_{ri}$” column indicates the number of states of each reduced supervisor calculated. The reduced supervisors are shown in Fig. 4, with their disabling events maps, which are events to be disabled at the plant for each state of the supervisor, are indicated by red dotted lines [1].

4. Performance Analysis

In this section, we present a proposal for performance analysis of the control logic synthesized in Section 3 employing the verification of timed automata.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Local modular synthesis summary.</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>SPi</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(G1 G2 ROp1 ROp2)</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(G2 ROp2)</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(G2 G3 ROp2)</td>
</tr>
</tbody>
</table>

4.1 General Proposal

Fig. 5 illustrates the general structure and the elements used in the proposal.

In Fig. 5, the plant components and the supervisors compose the closed loop system. The plant components generate events and the supervisors determine the disablements of events in the plant. TA (Timed Automata) represents the components of the plant and the supervisors. Events generated by the closed-loop system are observed by monitors, which are also TA that evolve according to the occurrence of events. The purpose of the monitors is to identify specific conditions to be observed in the closed-loop system.

Monitors are divided into validation and performance evaluation monitors. Validation monitors aim to test conditions related to compliance with the control specifications. Performance evaluation monitors are intended to associate conditions to certain
performance indexes, such as minimum or maximum number of parts produced in a given time, or minimum or maximum time to produce certain part.

Finally, CTL specifications are associated with the conditions indicated by the monitors. The CTL specifications are used to evaluate, under certain temporal constraints, the occurrence of conditions.

The general steps for the proposed method are: (1) modelling of plant components; (2) modelling of supervisors; (3) modelling the validation monitors and associated CTL specifications; (4) modelling the performance evaluation monitors and associated CTL specifications; and (5) verification.

In the following, the modeling steps of the plant components, supervisors and validation monitors are extensions of the models already developed for the synthesis of supervisors. For performance evaluation monitors, will be presented some typical structures that can be built, leaving it to the designer to create new possibilities.

4.2 Plant Components

Fig. 6 illustrates the plant components implemented as a network of Timed Automata. The components are constructed according to the ordinary automata in Fig. 2, with the inclusion of the timing of events presented in Table 1.

There are some differences between the UPPAAL TA and the ordinary automata, the reader could refer to Ref. [6] for more details. Besides the states and the transitions between states, the TA has a clock variable to count the passage of time, for example, the variable “q” at G1 (Fig. 6). We associate invariants to the states of the TA, which are the logical conditions that are active for the state, for example, “q <= 50” at s1 in G1, Fig. 6. To the TA transitions we associate labels, guards and resets. The labels indicate the occurrence of events, such as “a1!” to transition from s0 to s1 in G1. The guards are conditions that enable the occurrence of the transitions, such as “q == 50” to transition from s0 to s1 in G1. The resets are functions that are executed with TA transitions and work to update the value of variables with the occurrence of transitions, such as “q; = 0” in the transition from s0 to s1 at G1. In the TA the initial state is indicated by a double circle, and the marked states are not defined.

The synchronization of events is implemented in UPPAAL as a broadcast channel. Transitions with labels e? are synchronized with the occurrence of a transition with label e! [6]. To implement a synchronization scheme as established in the synchronous composition in section 3, the labels type e! are defined only for the components of the plant, considered as generators of events, while the events of the supervisors and monitors have the label e?, in order to follow the events generated by the components of the plant. If there are plant components or operational constraints with synchronized events, such as G1 and ROp1, we choose the component of the plant as the event generator.

For each controllable event “e” a Boolean vector d_e is defined. Each position d_e indicates if a supervisor is enabling or disabling event “e”. The

Fig. 6 Plant components as a network of Timed Automata.
vector \( d_e \) is indexed from 0 to \( NS-1 \), wherein \( NS \) is the number of supervisors. If the value of \( d_e[i-1] \) is zero, with \( i = 1, \ldots, NS \), this indicates that the supervisor \( Zr_i \) is enabling event “e”, and if \( d_e[i-1] \) is one, \( Zr_i \) is disabling event “e”.

An event “e” is then enabled when the guard of the transition “e!” on the plant, defined by the condition pool \( (d_e) == 0 \) is true. The function pool(d_e) is the “OR” logic of all elements of the vector \( d_e \). Thus, an event “e” is enabled in the plant only if all supervisors “agree” that it should be enabled at any given time.

In addition to the timing of events, the plant components could include other devices or interactions that were not taken into account in the modeling for the supervisor synthesis. In the case of the window bonding system, this could include the models for the entry and exit mats or aspects related to the implementation, as the operational sequences presented by Ref. [8].

4.3 Local Modular Supervisors

Fig. 7 shows the local modular supervisors of Fig. 4 implemented as a Network of Timed Automata.

Instead of the disabling event maps shown in Fig. 4, the enabling and disabling of events is performed by functions enable() and disable(), respectively, defined for the reset conditions of the supervisor transitions, as shown in Fig. 7. For supervisor \( Zr_i \) and the event “e”, the element \( d_e[i-1] \) becomes 0 in a transition with enable\( (d_e[i-1]) \), and the element \( d_e[i-1] \) becomes 1 in a transition with disable\( (d_e[i-1]) \).

Instead of implementing the local modular supervisors, it would be possible to implement the monolithic supervisor defined by the automaton \( Zc \) mentioned in Section 3, that represents the expected behavior of the closed-loop system. However local modular supervisors allow easy replacement and insertion of new components in the architecture, as well as being easier to implement, because they have potentially fewer states.

4.4 Monitors for Validation of Specifications

We defined a set of monitors associated with specifications using CTL (computation tree logic) written in the specification language of UPPAAL [6]. Monitors for validation of the specification can be built based on the desired specifications for the system, the automata \( SP_i \) defined in Fig. 3. Fig. 8 illustrates the performance monitors \( M_1 \) to \( M_3 \), constructed based on the specifications \( SP_1 \) to \( SP_3 \) respectively. Associated monitors \( M_1 \) to \( M_3 \) we’ve written the CTL specifications \( C_1 \) to \( C_3 \) shown in Fig. 9, respectively. Refer to Ref. [6] for CTL language of UPPAAL toolkit.

Consider then monitor \( M_1 \) in Fig. 8. The cycle formed by the states \( s0, s1 \) and \( s2 \) of \( M_1 \) corresponds to the behavior expressed by \( SP_1 \) in Fig. 3. The states \( BAD1, BAD2 \) and \( BAD3 \) reflect the undesirable conditions that are: SGR does not remove a side glass without it being there (BAD2), WSR does not remove the side glass at B1 without having spent the time to “dry” the primer (BAD1), and SGR does not put two side glass one on top of another (BAD3). The CTL specification \( C_1 \) in Fig. 9, expresses that the undesirable condition defined by the states \( BAD1, BAD2 \) and \( BAD3 \) does not occur for all traces of events that the system can generate. Analogously, the
Timed Automata Verification Applied to Performance Analysis of Control Logics Synthesized by Supervisory Control Theory

4.5 Monitors for Performance Evaluation

The following are some general guidelines for developing performance monitors for the system. Consider, for the simple system, the monitors D1 to D4 and specifications C7 to C14 shown in Figs. 10 and 11, respectively.

Monitor D1, associated to specifications C7 and C8, takes care of the minimum and maximum time for the system to process one cabin, while monitor D3, associated with specifications C9 and C10, concerns to the number of cabins processed by the system in $Q_{max} = 1,000$ units of time. Monitors D2 and D4 are analogous to monitor D1 and D2, now considering the throughput of side glasses.

Monitor D1 checks the time between the start of the operation of a cabin, event $a_3$, until its departure from the system, event $b_{23}$. The time clock is stored in the variable $q$ and it is checked in the urgent state PROBE. It is employed an urgent state to have exactly the final time when $b_{23}$ occurred [6].

The procedures for validating the behavior of modular supervisors together with the components of the plant using formal verification based on monitors and CTL specifications are somehow general, since it depends on the models developed during the synthesis of supervisors.

Monitor M2(M3) and the CTL specification C2(C3) refer to specification SP2 (SP3).

Monitor M4 and the CTL specification C4, Figs. 8 and 9, respectively, relate to the E7 specification of Section 3, i.e., no timeout of the production line.

Other specifications can be written and the presented below do not come to exhaust the possibilities.

The CTL specification C5 in Fig. 9 is used to check whether there is a state of deadlock in the system, i.e., the state in which there is no possibility of evolution of the model.

On the other hand, the CTL specification C6 in Fig. 9 can be interpreted as follows: “is it possible that from any state can reach the initial configuration of the plant models and supervisors?” As these states are marked in the models of SCT, then from any state one reaches the marked state in closed loop. Hence it can be concluded that the closed loop system is nonblocking.

The procedures for validating the behavior of modular supervisors together with the components of the plant using formal verification based on monitors and CTL specifications are somehow general, since it depends on the models developed during the synthesis of supervisors.
Specification C7 checks if there is a trace in the system that D2.PROBE is true and D2.q is less or equal than a Qmin value, while C8 tests whether the same conditions, D2.q is greater than Qmax. The proposed method for the verification of C7 and C8 properties is as follows. It begins with checking C7, starting with a low Qmin, for example, zero, and simultaneously performing verification and incrementing the value of Qmin until the specification becomes true. This Qmin value is the minimum time to produce a part by the system. The checking of C8 starts with the value of Qmax equal to the value obtained in the foregoing specification and proceeds running the specification at the same time that Qmax is increased until the specification becomes false. The next lower value of Qmax to the first false specification is the maximum time of production of a part by the system.

Monitor D2 corresponds to the number of cabins processed by the system within a time interval [0, Tmax], in the case of Fig. 10, with Tmax = 1000s. The number of processed cabins is stored in the variable n. One may choose a value of Tmax big enough to have reasonable time range for processing a certain number of cabins.

The method to analyze C9 and C10 starts by checking C9 varying Nmin until it becomes true. This value of Nmin is the minimum number of cabins that can be processed by the system in [0, Tmax]. Then, we follow with the verification of C10 varying the value of Nmax from the value of Nmin found in the first specification verification until C10 becomes false. The immediate next lower value of Nmax is the maximum number of cabins that can be processed in the interval [0, Tmax].

Specifications C11 to C14 are analyzed in a similar way to monitors D2 and D4.

Other performance specifications can be written, depending on the skill of the designer. Regardless, the two types of monitors and measures presented in this Section, D1 to D4, are somewhat general and apply to a wide range of systems. Consider then monitor D5 in Fig. 12 and the CTL specifications in Fig. 13.

As an illustration, consider the monitor D5 in Fig. 12. Monitor D5 deals, essentially the processing of a cabin, since the authorization for the arrival of that (a3), the arrival in position A (b3), up to the placement of the windshield (b23), when it reaches the state ONE.

Monitor D5 is provided with the following auxiliary variables. The variable q is a clock that counts the time between the arrival of the first cabin at position A and the placement of the windshield by WSR in the same cabin. The variables n1, n2, n3 and n4 are counters for the number of side glass treated in the following situations. These variables are limited to the NL value to guarantee the convergence in the verification. The n1/n3 variables count the number of side glass that are processed by SGR (primer application and deposit at B1) before/after the arrival of the first cabin. The n2/n4 variables count the number of side glass that are processed by WSR (glue application and release) before/after the arrival of the first cabin.

The CTL specifications C15 to C19 in Fig. 13 are relate to the evaluation of properties expressed in D5.

CTL specification C15 aims to evaluate the minimum time of placing a side window in the cabin. It literally expresses the matter: “There is a way in which the state
ONE is true and q is lower than the value specified at Qmax?"

CTL specifications C16, C17, C18 and C19 specifications deal with the number of side glass processed by the system in situations characterized by the counters n1, n2, n3 and n4, respectively. In general, the system tests whether there is a state in which ONE is valid and ni is equal to a given value Nimax with i = 1 ... 4. The systematic to verification of these specifications was start Nimax (i = 1 ... 4) with a low value and raise it until its specification becomes false.

4.6 Verification

The verification step is performed using the verification tool and correspond to queries on the veracity of each specification. As shown in previous sections, there are results that arise from the truth or not of a given specification, as in the case of specifications associated to validation monitors. On the other hand, there are results arising from the manipulation of variables that make specifications true or false, as in the case of specifications associated with the performance monitors.

It is important to highlight the problem of computational complexity of the verification. There are well-known computational limits in terms of resource requirements of memory and time in the verification of timed automata networks [6]. Although the analysis employing techniques of relief computational complexity of the verification, as mentioned in Ref. [6], is not in the focus of this paper. But this is a problem that should be taken into account when applying the proposed method for large-scale systems.

Another important point is the analysis of the error trace when a CTL specification is false. Although the tool provides resources for simulating a trace of error, which illustrates a counter-example of a false specification, the understanding of the reason for that the specification is false is a nontrivial task and requires expert knowledge.

In the following, we present the results for the verification of the properties for the logic controller developed to the window bonding system.

Verification of C1, C2 and C3 returned true, indicating the correct operation of local modular supervisors Zr1, Zr2 and Zr3, respectively, despite the timing issues inserted in the plant.

As stated before the proposed control system cannot handle the timeout of the production line, because although supervisors express all necessary interlocks, one solution is to do nothing with the plant in the initial state. In this way, monitor M4 reaches the undesirable state BAD10 and the verification of C4 returns false.

The result of the verification is that specifications C5 and C6 are true, which leads to the conclusion that the system composed by the plant components associated with the supervisors has no deadlock or blocking states.

The results of specifications C7 to C14 are listed below:

- Specification C7 (C11) is true for Qmin equal to 85 s (115 s), signifying the minimal time for installation of a windshield in a cabin (for production of one pair of side glasses);
- Specification C8 (C12) is true for any value of Qmax, meaning that without the forcing of events, the supervisors can remain indefinitely waiting the generation of events by the plant;
- Specification C9 (C13) is true for any value of Nmin, also due to the non-enforcing nature of the events; and
- Specification C10 (C14) are true for Nmax equal to 8 (10), which means that, in 1,000 s, 8 cabins (10 side glasses) can be finished by the system.

The verification of C15 starts with a high value for Qmax, for example, 50, and decreases until the specification becomes false, which occurred in the passage from Qmax from 40 to 39. Therefore 40s would be the minimum time for the placement of the windshield in the cabin.
As a result of the verification of specifications C16 to C19, the maximum values for which the specifications are true were $N_{1{\text{max}}} = 2$, $N_{2{\text{max}}} = 2$, $N_{3{\text{max}}} = 1$ and $N_{4{\text{max}}} = 1$. Therefore, we conclude that SGR can process up to 2 pairs of side glasses before the command to enter a cabin; WSR can process up to 2 pairs of side glasses before the command to enter a cabin; SGR can process up to 1 pair of side glass after the arrival of a first cabin; and WSR can process up to 1 pair of side glasses after the arrival of the first cabin.

Table 3 summarizes the results of the verification procedures. The procedures were performed in an Intel i7 machine with 8 Gb of RAM using UPAAL TIGA 4.1.4 (ref. 4740). The maximum time for verifying any property was around one minute.

### 5. Conclusions

In this paper, we have presented an approach for coordination of the components a window bonding system based on two techniques associated to DES: the local modular approach for supervisor synthesis in SCT and the Timed Automata verification.

After modeling the system components and events, we have developed automata models for the components and the coordination specifications. By applying the local modular control of Ref. [3], we have obtained a set of local supervisors for coordination of the system. Some of the temporal coordination specifications could not be attained due to the abstraction of real timing into artificial timer events.

In order to execute a performance evaluation of the system under the designed control law, we have applied the Timed Automata Verification. In order to do that, besides timed automata models for the system components and supervisors, we have developed timed automata monitors and CTL specifications that characterize the diverse scenarios for performance evaluation. As results, we have got information on the minimal and maximal production throughput and production times for the controlled system.

The application of formal verification for performance evaluation of control systems is not a novelty, as can be seen in Ref. [9], where formal verification is applied to ensure the robustness of a controller synthesized for a nonlinear system. We believe that this verification approach can complement

<table>
<thead>
<tr>
<th>Specification</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>True</td>
<td>$Z_r_1$ satisfies specifications.</td>
</tr>
<tr>
<td>C2</td>
<td>True</td>
<td>$Z_r_2$ satisfies specifications.</td>
</tr>
<tr>
<td>C3</td>
<td>True</td>
<td>$Z_r_3$ satisfies specifications.</td>
</tr>
<tr>
<td>C4</td>
<td>False</td>
<td>Timeout of production line not avoided.</td>
</tr>
<tr>
<td>C5</td>
<td>True</td>
<td>Absence of deadlock.</td>
</tr>
<tr>
<td>C6</td>
<td>True</td>
<td>System is non-blocking.</td>
</tr>
<tr>
<td>C7</td>
<td>True</td>
<td>Minimal time to produce WS is 85 s. $(Q_{\text{min}} \geq 85)$</td>
</tr>
<tr>
<td>C8</td>
<td>True</td>
<td>Unlimited maximal time to produce WS. $(Q_{\text{max}} \geq 0)$</td>
</tr>
<tr>
<td>C9</td>
<td>True</td>
<td>$p/qq N_{\text{min}}$ is the minimum produced in 1,000 s.</td>
</tr>
<tr>
<td>C10</td>
<td>True</td>
<td>8 WS is the maximum produced in 1,000 s. $(N_{\text{max}} \leq 8)$</td>
</tr>
<tr>
<td>C11</td>
<td>True</td>
<td>Minimal time to produce SG is 115 s. $(Q_{\text{min}} \geq 115)$</td>
</tr>
<tr>
<td>C12</td>
<td>True</td>
<td>Unlimited maximal time to produce SG. $(Q_{\text{max}} \geq 0)$</td>
</tr>
<tr>
<td>C13</td>
<td>True</td>
<td>0 WS is the minimum produced in 1,000 s. $(p/qq N_{\text{min}})$</td>
</tr>
<tr>
<td>C14</td>
<td>True</td>
<td>10 SG is the maximum produced in 1,000 s. $(N_{\text{max}} \leq 10)$</td>
</tr>
<tr>
<td>C15</td>
<td>True</td>
<td>Minimal time to install WS in a cabin in 40 s. $(40 \leq Q_{\text{max}} &gt; 39)$</td>
</tr>
<tr>
<td>C16</td>
<td>True</td>
<td>SGR can process up to 2 SG before allowing the entrance of the first cabin. $(N_{1{\text{max}}} = 2)$</td>
</tr>
<tr>
<td>C17</td>
<td>True</td>
<td>WSR can process up to 2 SG before allowing the entrance of the first cabin. $(N_{2{\text{max}}} = 2)$</td>
</tr>
<tr>
<td>C18</td>
<td>True</td>
<td>SGR can process up to 1 SG after the entrance of the first cabin. $(N_{3{\text{max}}} = 1)$</td>
</tr>
<tr>
<td>C19</td>
<td>True</td>
<td>WSR can process up to 1 SG after the entrance of the first cabin. $(N_{4{\text{max}}} = 1)$</td>
</tr>
</tbody>
</table>
exhaustive simulations for performance evaluation. Moreover, besides including timing information we can also include in the verification more of the components and aspects that were abstracted out during the control design. This is the case, for instance, of the operational sequences that define the components of the plant used in Ref. [8].

We are currently developing controllers for the window bonding system using timed petri nets [1] and the timed DES control approaches of Refs. [4, 5]. Moreover, we intend to compare the performance of the controllers obtained by the different approaches using formal verification.

References


