ELSEVIER

Computers in Industry 38 (1999) 79–92

COMPUTERS IN INDUSTRY

Combined simulation for process control: extension of a general purpose simulation tool

G. Mušič *, D. Matko

University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia

Abstract

Combined discrete event and continuous views of production processes are important in designing computer control systems for both process industries and manufacturing. The paper presents an extension of the popular Matlab–Simulink simulation tool to facilitate the simulation of the discrete sequential control logic applied to continuous processes. The control system is modelled as a combined system where the discrete and the continuous parts of the system are separated and an interface is introduced between them. The sequential control logic is represented by a Sequential Function Chart (SFC). A SFC blockset is defined to enable graphical composition of the SFC and its integration into the Simulink environment. A simulation mechanism is implemented which is called periodically from the standard Simulink simulation engine and carries out the correct state transition sequence of the discrete model and executes corresponding SFC actions. Two simulation case studies are given to illustrate the possible application of the developed simulation environment: the simulation of a batch process cell, as an example from the area of process control and an example of a manufacturing system, i.e. the control of a laboratory scale modular production system.

Keywords: Simulation; Hybrid systems; Petri nets

1. Introduction

Simulation plays an increasingly important role in the process of designing various production facilities. It can be applied in many different areas ranging from strategic market prediction and business process simulation at the highest management level of the control hierarchy, to the production cell and process control loop simulation at the lowest process control level. In any case, simulation helps in predicting future behaviour of the observed system and can be used as a testing environment for systems that are being designed. This results in reduced risk and reduced investment on the one hand and shorter throughput time and reduced operating costs on the other.

Technical processes that have to be controlled in process industries such as food, chemical and petrochemical industries exhibit a predominantly continuous nature and continuous control is of major concern in these areas. Continuous control, however, cannot always satisfy the desired overall system performance criteria. Several flexible automation concepts have therefore been adopted from the area of manufacturing systems and are used in modern process plants. These include the highly distributed nature of the underlying control system and the

* Corresponding author. E-mail: gasper.music@fe.uni-lj.si

0166-3615/99/$ - see front matter
PII: S0166-3615(98)00110-9

interaction of several sub-processes as well as co-ordination and scheduling issues. Corresponding control systems are characterised by a combination of continuous and discrete control functions.

Discrete event sequences appear in various stages of process control system operation. A typical example is a start-up sequence, where process units have to be started in a prescribed order and furthermore, some units must not be started before corresponding necessary conditions (e.g., temperature, pressure) inside the process are met. A similar situation emerges with process shutdown. These sequences are normally performed by skilled operators but their complexity of operation and required safe handling in emergency conditions motivate the development of automatic control for such procedures. Emergency scenarios are provided for the case of a failure of a piece of equipment. Sequential control can contribute to the effectiveness of the corresponding actions, which can include plant reconfiguration or safe shutdown.

The area of process control where a discrete event view of the system is particularly important is the area of batch systems. Sequential control functions found here basically correspond to one of two types: they (a) carry out basic system operations, that is a sequence of actions (recipe) required to make a product, or (b) comprise the higher level supervisory functions responsible for allocation of process units, co-ordination and scheduling. These functions introduce many classical discrete manufacturing system design issues into the design and analysis of the batch production plants.

When designing such a system it is desirable to have a simulation model where critical decisions about the system structure and control algorithms can be tested according to a given objective. Many modelling and simulation tools have been developed to support this task. They generally belong to one of the two large families: continuous system simulation tools and discrete event simulation tools. Each of these only covers certain aspects of the system and enforces the use of partial simulation models describing system function from a particular viewpoint. This may however, lead to solutions where several interactions are neglected or even completely overlooked. The overall system simulation would be beneficial in this sense.

The control system, which is a hybrid continuous/discrete event system, can generally be simulated in one of two ways. Either the continuous part of the system is abstracted and described by a corresponding discrete event model and the whole system is then simulated by a discrete event simulation tool, or the system is simulated as a hybrid system, where certain events that appear in the discrete event part are generated as a result of continuous state evolution (state events) and some of the control actions in the continuous part depend on the discrete state. The former approach has the advantage of relatively simple simulation but much effort has to be put into development of an appropriate discrete event model of the continuous part. The later, hybrid simulation approach enables a relatively simple modelling but the simulation is more complicated.

The choice of a particular simulation technique strongly depends on the modelling framework. Several approaches to modelling of hybrid systems can be found in the literature. Some authors propose a continuous model studied in the presence of discontinuities [1]. Another popular approach is the extension of the particular discrete event modelling framework to cover the continuous system aspects and at the same time preserve the main analysis techniques. Most of this work has been devoted to various extensions of Petri nets [2,3]. In another approach, a hybrid system is modelled as a combined system where the discrete and the continuous parts of the system are separated and an interface is introduced between them. The continuous part of the system is modelled by classical quantitative models such as differential equations, while the discrete event part is modelled either by state diagrams [4] or by Petri nets [5]. The interface translates the symbolic output of the discrete event subsystem into the corresponding set of control actions and generates state events.

When this later approach is used for formal analysis of the system, the continuous part can be abstracted and a discrete event model of the continuous system can be developed in order to enable the application of the methods from the area of discrete event systems to the design and formal verification of the hybrid system [6]. On the other hand, this approach is particularly well suited for simulation of overall systems. The abstraction of the continuous part can be omitted and state events can be generated

based on the continuous state during simulation, as in the case during the operation of real system. The corresponding simulation algorithm has to cope with discontinuities, state event handling, state re-initialisation, etc. [7].

In this paper, we present a simulation approach based on a continuous simulation tool that was extended to facilitate the integration of discrete event based control logic into the simulation model. The paper is organised as follows. In Section 2 the combined simulation approach is presented. The Sequential Function Chart extension to Matlab–Simulink is introduced and its integration into the existing Simulink simulation environment is described. Section 3 illustrates the potential application of the developed tool, using two case studies: a simulation study of a batch process cell as an example from the area of process control and a simulation study of a laboratory scale modular production system as an example from the manufacturing area. Finally, some conclusions are drawn in Section 4.

2. Combined simulation approach

For testing purposes it is advantageous or even necessary to be able to simulate both continuous and discrete parts of a hybrid system in the same simulation environment as discussed above. It might look rather obvious to choose the discrete-event simulation domain as the foundation for this combined simulation tool. The continuous simulation however, is numerically more demanding and therefore the continuous simulation tools where more sophisticated continuous simulation algorithms are implemented, yield better results. With the improvement of these algorithms in the sense of correct location of state events and handling discontinuities, the use of such tools for a foundation of combined simulation has also become possible.

Our recent work concentrates on enhancements of the widely used continuous simulation tool Matlab–Simulink toward combined simulation capabilities. Matlab–Simulink is one of the simulation tools commonly used in control systems design. It enables graphical model building through block schemes and the simulation of continuous and discrete time systems. Until recently, not much support has been given to event driven systems.

The emphasis of our work has been on the development of a blockset that facilitates the design of the discrete part of the simulation model and its integration into the Simulink environment. A simulation mechanism for the discrete event model, which runs in parallel with the standard ODE solver, has also been implemented.

In the presented simulation approach the combined system structure discussed above is followed. The continuous process part is simulated by the existing Simulink continuous simulation block library. All continuous sub-processes are simulated there and the continuous controllers are included in the simulation scheme as well. The discrete control logic could be simulated by existing logic blocks in Simulink but this would lead to large and complex simulation schemes which are hard to follow.

In our solution, the discrete part of the model is described by the Sequential Function Chart (SFC), also known as Grafcet, a notation taken from the IEC standard [8] which defines programming languages for programmable logic controllers (PLC). SFC inherited many of its features from the theory of Petri nets. More precisely, safe interpreted Petri net can be defined such that the input–output behaviour is the same as the input–output behaviour of the SFC [2,9]. Its strong relation to Petri net theory enables a SFC to be directly redrawn from a Petri net model and the classical properties of Petri nets, such as marking invariants, can be applied also to SFC. We found SFC notation particularly useful because the simulation model can be easily transformed into a logic controller program.

The SFC part of the simulation scheme is built up of steps and transitions. Actions are assigned to steps and conditions are assigned to transitions. In the standard, an action corresponds either to single variable or a subroutine whose body defines the actual procedure that is performed when the corresponding action is operative. In our current implementation, however, actions are limited to a change of the value of a single Boolean variable only, while we retained all action types defined by the standard. Actions are defined inside SFC steps as global variables followed by the action type qualifier and an optional

parameter. The values of the global variables are then assigned to the discrete subsystem outputs.

A typical action declaration is, for example

$$valve\_a(L, t = \#100\,\mathrm{s})$$

which causes the global variable *valve_a* to be set to 1 (TRUE) when the corresponding step becomes active and the variable is reset to 0 (FALSE) after 100 s or at the deactivation of the step, if this occurs earlier ('Limited' action).

Similarly, the communication between discrete subsystem inputs and the SFC transition blocks is also performed by global variables. Each input is assigned to a global variable and expressions built up of these variables and standard Matlab operators are assigned to transitions. The result of such an expression is interpreted as a Boolean value that enables or disables the corresponding transition.

A typical transition condition might look like

$$level\_b \,\&\, \sim lock\_1$$

which means that the corresponding transition is fired when the preceding step is active and global variable *level_b* has value 1 (TRUE) and global variable *lock_1* has value 0 (FALSE).

The values of global variables are read and written to in the I/O section of the SFC where variables are assigned to inputs and output ports of the SFC part of the overall simulation scheme. The SFC part forms a separate subsystem block which is then linked to the rest of the scheme by its input and output signals as any other Simulink block.

Macro-steps, which contain a portion of the SFC defined elsewhere, can be used to make the simulation schemes of the complex sequential control strategies more transparent. The SFC that corresponds to a macro-step is defined in a separate window and is composed as an ordinary SFC with the following limitations: it must contain exactly one input and one output step and must not contain any additional I/O ports. This allows a macro-step to be expanded and its contents included in place of the macro-step at the preceding level SFC. An arbitrary number of nesting levels are supported with the limitation that all variables used in the steps and

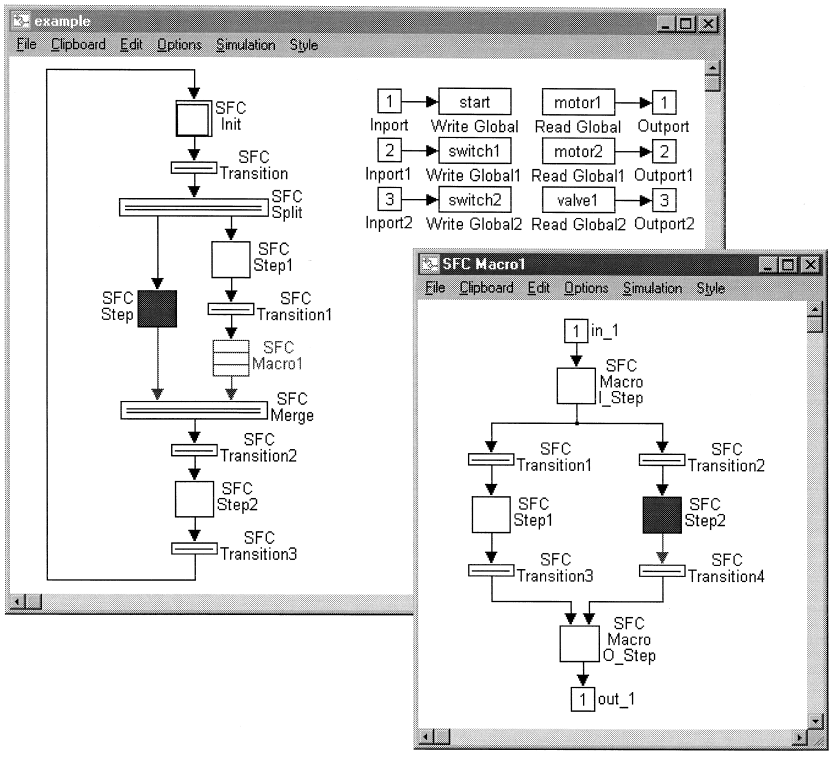


Fig. 1. Macro-step and its expansion.

actions are global and care must be taken not to duplicate variable names. An example of the macro-step and its expansion is shown in Fig. 1.

A separate simulation mechanism for the SFC part of the simulation scheme which runs in parallel with the standard ODE solver has been defined in order to achieve the correct state transition sequence of the SFC and execution of the actions associated with SFC steps. This SFC simulation routine is registered by the Simulink integration algorithm as just another discrete time block but with variable sampling time. The functions of the SFC simulation algorithm performed in the different phases of the Simulink integration step are shown in Fig. 2.

Each SFC block is registered to a reserved set of global variables during the initialisation phase of the simulation. A special data structure is built, which contains all the topological information about SFC, including the incidence matrix, initial SFC marking vector and actions and conditions associated with SFC steps and transitions.

A routine that calculates the new state of the chart is called once during each major integration step. It evaluates the transition conditions and based on the result, the set of enabled transitions is determined. The new marking is calculated and step action execution flags are set accordingly. Step actions are executed according to these flags and action type qualifiers which determine the start and the duration of the actions relatively to the activation of the corresponding step. The time event schedule is updated if the new activated action contains a time delay qualifier. The nearest time event is scheduled during the integration algorithm 'next sample time' call where the corresponding time is returned by the SFC simulation routine as the next sampling time of the block. In the case of unstable markings, the next sample time in returned as the current time increased by the minimum integration step. This assures that no state transition, which may schedule a new time event, is lost.

The SFC simulation routine only takes care of time events generated by step actions. State events that originate from the evolution of the continuous part of the system are handled by the interface which forces the integration routine to decrease the integra-

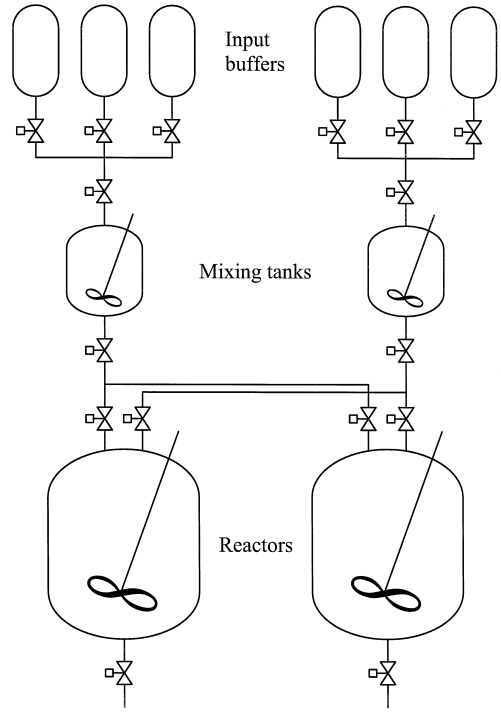


Fig. 3. Batch process cell.

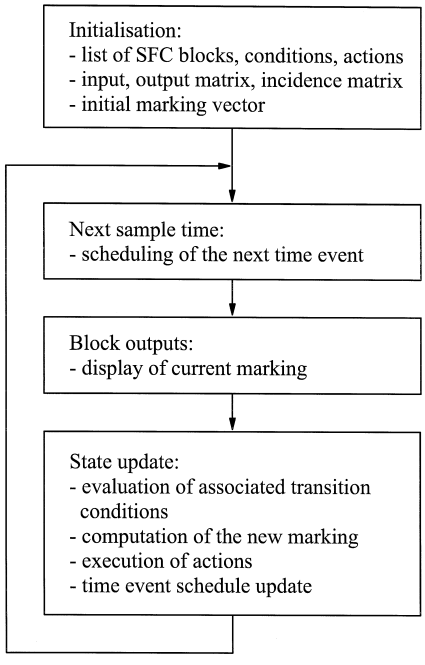


Fig. 2. SFC simulation algorithm.

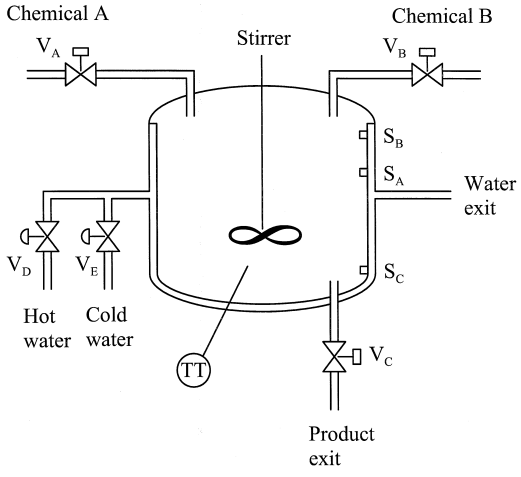


Fig. 4. Batch reactor.

tion step in the vicinity of the crossing points. Note that 'state update' calls are performed by the Simulink integration routine in each major integration step regardless of the sampling time, which guaranties that all state events detected by the main integration routine are detected by the SFC simulation routine as well.

The interface between the continuous and the discrete section consists of two parts. The first part performs the generation of events, which trigger the transition enabling conditions. State events are generated by comparison of the signals of interest to the specified threshold values. The result of each comparison is fed as input into a discrete subsystem. The second part of the interface maintains the processing of the results of the discrete actions. The discrete subsystem outputs are appropriately transformed and can be used in the continuous part of the simulation scheme as switching signals or step shaped inputs.

Note that Simulink version 1.3, which was used during the development of the SFC blockset, does not provide any accurate simulation method that would enable a precise simulation in the presence of discontinuities such as switching a binary signal upon detection of a particular threshold. However, by an appropriate decrease of the integration step within the interval of interest, satisfactory results can be obtained. To achieve this, the result of the comparison is multiplied inside the interface by a large constant and fed to the auxiliary integrator. This forces the integration algorithm to decrease the integration step in the vicinity of the switching point and so increase the precision of the switching point location. Later versions of Simulink, such as 2.0 and 2.1 include the precise state event location mechanism, which is activated automatically whenever a comparison or other similar block is used. In this case the integration step decrease is no longer necessary and the interface can be simplified.

3. Case studies

Two simulation case studies are given in the remainder of the paper to illustrate the possible application of the developed simulation environment: simulation of a batch process cell as an example from the area of process control and an example of a manufacturing system, i.e., control of a laboratory scale modular production system. The relation of Petri nets to a SFC industrial standard makes it natural to use Petri nets as a sequential modelling tool in order to remain close enough to the industrial implementation. Compared to other state transition modelling tools, a Petri net based model contains structural information that is absent from a simple state transition graph, for example. In both case

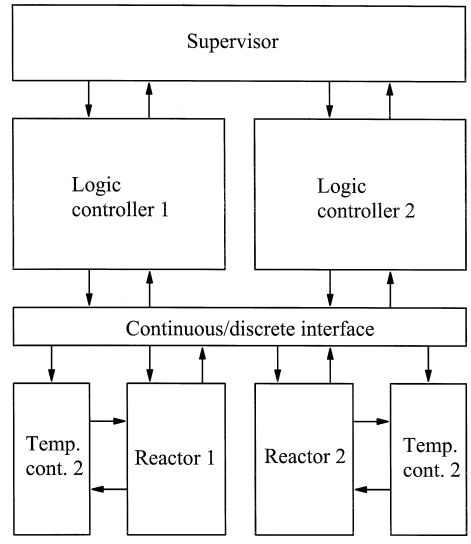


Fig. 5. Model structure.

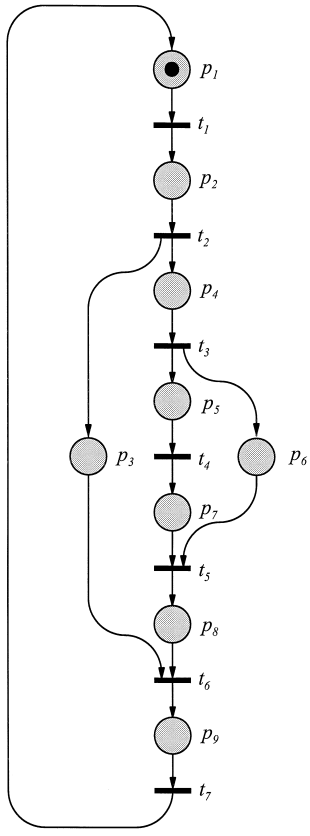


Fig. 6. Petri net model of the logic controller.

studies, a Petri net model of the desired system behaviour is developed and a Sequential Function Chart of the control program is derived from the given model. Ordinary Petri nets are used for the development of a base model and control interpretation of places and transitions is added to enable the transformation to a SFC. In parallel to this a continuous process model is developed and the behaviour of the controlled system is then simulated in the proposed simulation environment.

3.1. Batch process cell

A typical batch process cell shown schematically in Fig. 3 is considered in the first case study. The cell consists of several input buffers, two mixing tanks and two reaction vessels. We will focus on the operation of a single batch reactor shown in detail in Fig. 4.

Each reactor has two inlets for two incoming chemicals and a third chemical is produced by the reaction of the two chemicals at a specified temperature. The filling of the reactor is controlled by the two on/off valves V_A and V_B and the discharging is controlled by the on/off valve V_C. The level is measured by three level switches (S_A, S_B and S_C). The temperature of the reaction is controlled by feeding hot or cold water through the water jacket, which surrounds the reaction vessel. The water flow is controlled by adjusting proportional valves V_D and V_E and the temperature of the reactor contents is measured by temperature sensor TT.

The operation of the single batch reactor is specified as follows:

1. Open valve V_A to start charging the reactor with chemical A.
2. When the level switch S_A closes, close V_A.
3. Start the stirrer.
4. Open valve V_B to start charging the reactor with chemical B and start the timer to time the duration of the filling.
5. When the timer indicates that the predefined amount of the chemical has been charged, or the upper level switch (S_B) closes, close V_B.
6. Switch on the temperature controller to heat up the mixture to the required reaction temperature.

Table 1
Control interpretation of places and transitions in Fig. 6

Places	
p_1	Initialisation
p_2	V_A open
p_3	Stirrer operating
p_4	V_B open, Filling timer running
p_5	Heating up to the setpoint
p_6	Temperature controller enabled
p_7	Reaction timer running
p_8	V_E fully open
p_9	V_C open
Transitions	
t_1	Start of cycle
t_2	S_A closed
t_3	Filling timer run out or S_B closed
t_4	Setpoint temperature reached
t_5	Reaction timer run out
t_6	Output temperature reached
t_7	S_C open

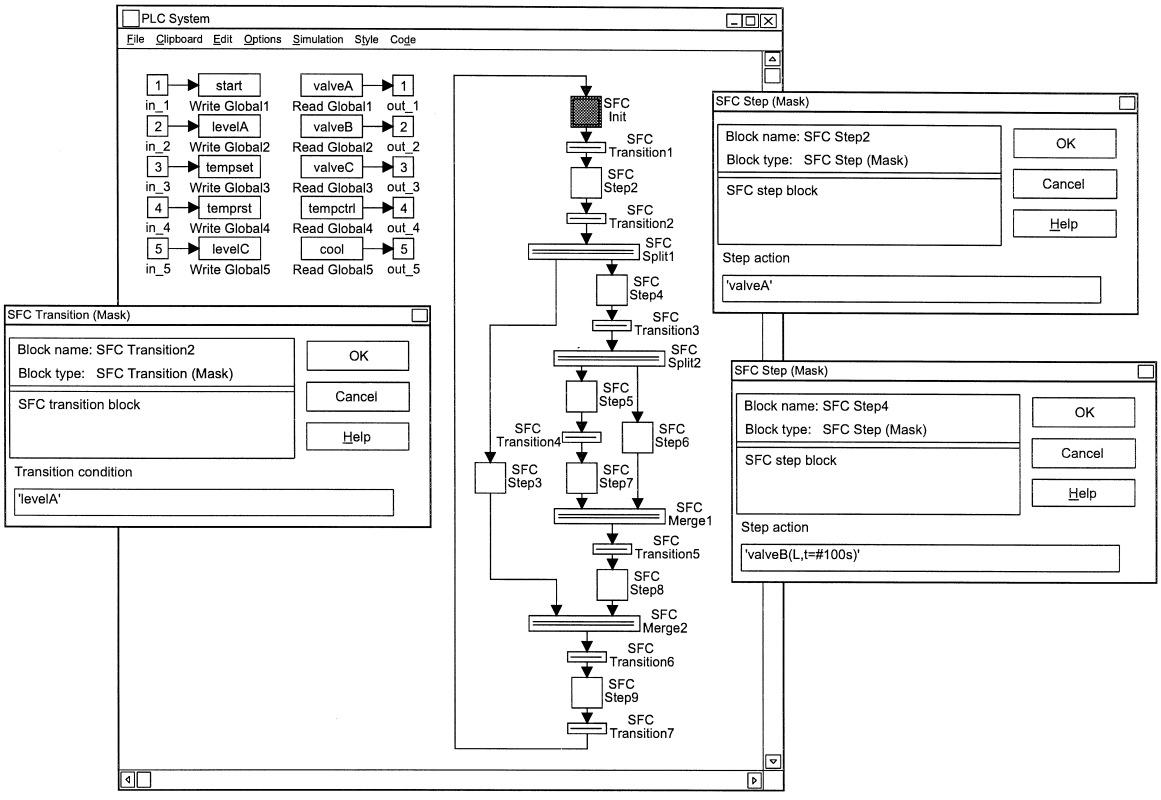


Fig. 7. Simulation scheme of the logic controller.

7. When the temperature enters a predefined tolerance band around the setpoint, start the timer to time the duration of the reaction.
8. When the timer runs out, switch off the temperature controller and fully open valve V_E to cool down the reactor contents.
9. When temperature falls below a predefined limit, switch off the stirrer and open valve V_C to remove the product from the reactor.
10. When the lower level switch (S_C) opens, close V_C, reactor is ready for the next batch.

The specified operation of the reactor is maintained by a programmable logic controller, which carries out the described sequence. Each reactor is controlled by its own controller to enhance the reliability of the whole system. However, certain operations of the reactors need to be co-ordinated. For example, filling of the reactor with chemical B is possible for only one reactor a time. Similarly, in the heating phase, a large amount of energy is required to heat the mixture, while only a small flow of hot water is required at the reaction temperature to keep the mixture at the constant temperature. Therefore it is desirable to co-ordinate the heating up phase between the two reactors as well. The co-ordination is a task for a supervisory system.

The described batch system is modelled as a combined continuous/discrete event system with the separated continuous and discrete subsystems and an interface in between as shown in Fig. 5.

The continuous part of the single reactor model consists of a temperature controller and two subprocesses, describing the liquid level in the reactor (Eqs. (1) and (2)) and the temperature of the mixture in the reactor (Eq. (3)). The temperature controller is a continuous PI controller.

$$A\frac{\mathrm{d}h(t)}{\mathrm{d}t} = \Phi_A(t) + \Phi_B(t) \tag{1}$$

$$A\frac{\mathrm{d}h(t)}{\mathrm{d}t} = -K_C\sqrt{2gh(t)} \tag{2}$$

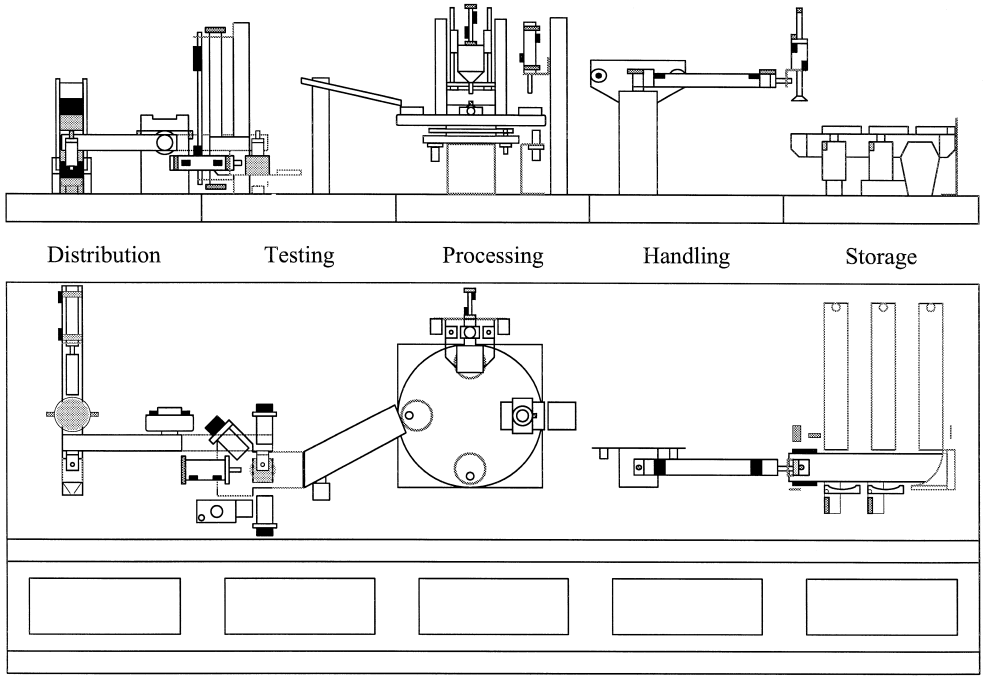


Fig. 8. Layout of the laboratory scale modular production system.

In Eqs. (1) and (2), $h(t)$ denotes liquid level, A is the transverse section of the reactor vessel, K_C is a valve constant and Φ_A and Φ_B are the volume flows at the corresponding inlets.

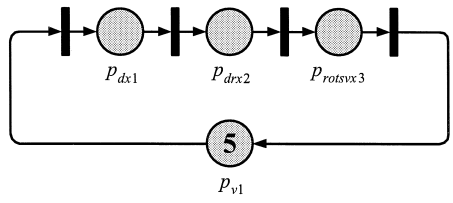
With the assumption of ideal heat exchange between heating/cooling water and the liquid in the reactor, the temperature in the reactor follows Eq. (3):

$$m_{\mathrm{mix}} c_{\mathrm{pmix}} \frac{\mathrm{d}\vartheta(t)}{\mathrm{d}t} = \rho_{\mathrm{w}} \Phi_{\mathrm{D}}(t) c_{\mathrm{pw}} (\vartheta_{\mathrm{D}} - \vartheta(t)) - \rho_{\mathrm{w}} \Phi_{\mathrm{E}}(t) c_{\mathrm{pw}} (\vartheta(t) - \vartheta_{\mathrm{E}}) \quad (3)$$

where $\upsilon(t)$ is the temperature of the mixture, m_{mix} and c_{pmix} are the mass and the specific heat of the mixture, ρ_{w} and c_{pw} are the density and specific heat of the water, $\Phi_{\mathrm{D}}(t)$, υ_{D}, $\Phi_{\mathrm{E}}(t)$, υ_{E} are the water flows and temperatures of the incoming water at the hot water inlet and the cold water inlet.

The temperature controller is a simple PI controller, which is tuned to give a response with no overshoot. The input to the controller is the temperature $\upsilon(t)$ while its output corresponds to Φ_{D} when the output is positive and the output corresponds to Φ_{E} when it is negative.

Upper, discrete event part of the model in Fig. 5 represents the local and supervisory sequential logic controllers and is modelled by Petri net. The Petri net model of the local controller is drawn on the basis of the functional specification as shown in Fig. 6. The places and transitions are related to process sensors and actuators as seen in Table 1. The model represents a control recipe applied to a particular reactor unit and can also be considered as a model of



p_{dx1}	distribution of workpieces
p_{drx2}	transport to testing station and testing
$p_{rotsvx3}$	subsequent operations (depend on the result of the testing)
p_{v1}	external place (represent the maximum number of workpieces that can be processed simultaneously)

Fig. 9. Starting Petri net model.

a batch reactor seen from the supervisory system. The supervisory part of the presented system is not covered here in detail. More emphasis on the supervisory part has been given in Ref. [10].

To simulate the behaviour of the controller the corresponding SFC model has to be derived. The Petri net model can only be transformed to SFC, if the corresponding net is safe [2,9]. Safety of the Petri net in Fig. 6 can easily be verified by the use of place invariants. Place invariants are sets of places whose weighted token count remains constant for all possible markings. A single place invariant is represented by a column vector $\boldsymbol{x}$ of length m, where m is the number of all places composing the Petri net. Nonzero entries of the vector $\boldsymbol{x}$ correspond to the places that form the place invariant. A sample set of

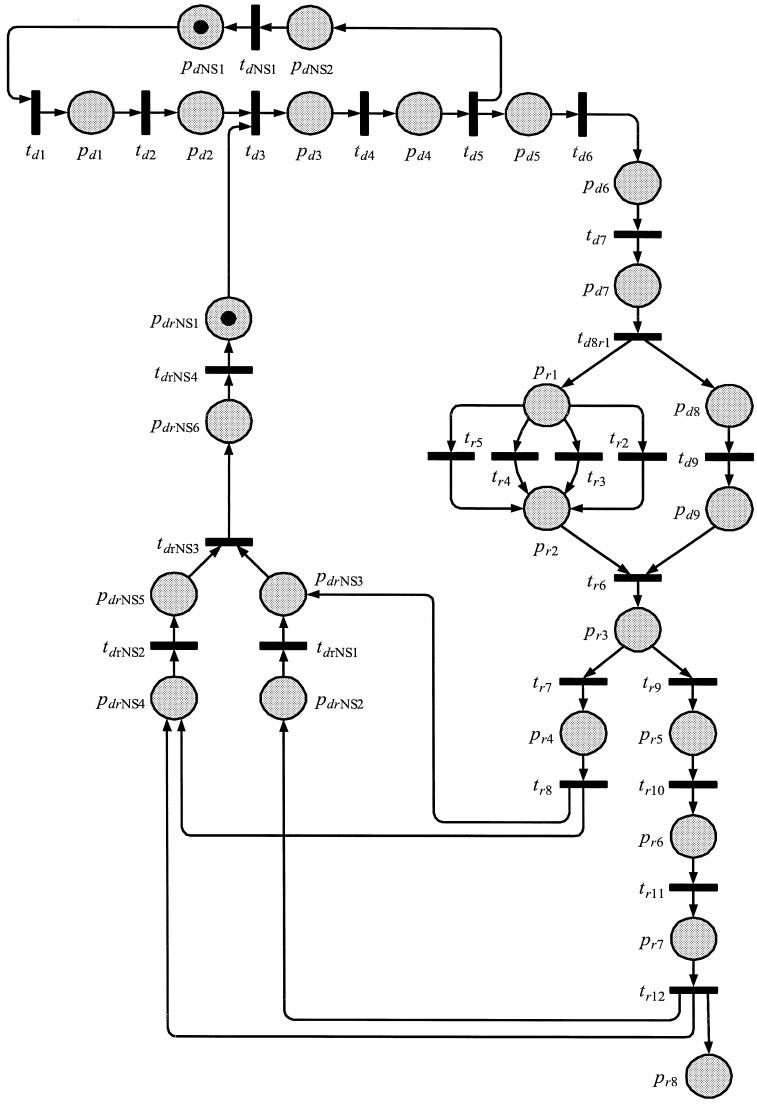


Fig. 10. Petri net model of the first two stations.

place invariants for the Petri net in Fig. 6 is listed in Eq. (4).

$$\begin{aligned} x_1^T &= (111000001) \\ x_2^T &= (110101011) \\ x_3^T &= (110110111) \end{aligned} \tag{4}$$

Considering the initial marking of the Petri net, a marking invariant can be assigned to every place invariant. Marking invariants that correspond to place invariants in Eq. (4) and initial marking shown in Fig. 6 are shown in Eq. (5).

$$\begin{aligned} &\mu_1 + \mu_2 + \mu_3 + \mu_9 = 1 \\ &\mu_1 + \mu_2 + \mu_4 + \mu_6 + \mu_8 + \mu_9 = 1 \\ &\mu_1 + \mu_2 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 = 1 \end{aligned} \tag{5}$$

It can be seen from Eq. (5) that every place of the given Petri net is included in at last one of the marking invariants with the sum of tokens on the right hand side equal to 1. This implies that none of the places can ever be marked by more than one token, therefore the Petri net is safe.

The derived model has been simulated within the extended Matlab–Simulink simulation environment. The corresponding simulation scheme is hierarchically decomposed into several subsystems following the model structure shown in Fig. 5. The use of the SFC library is demonstrated in Fig. 7 where the simulation scheme of the logic controller subsystem is presented and the linking of the discrete part to the rest of the simulation scheme is indicated. Different background colours of certain steps indicate that these steps are active at that moment. Colours are changed during the simulation run according to the current marking and so indicate the evolution of the chart.

At the start of simulation the *SFCInit* step is active by default. The evolution of the SFC chart starts when the input 1, which is linked to the global variable *start*, receives a value greater than 0. Variable *start* by itself forms a transition condition of the *SFCTransition1*, which is fired immediately, and the *SFCStep2* is activated. Thecorresponding action is *valveA*, which means that the global variable *valveA* will be set to 1 as long as the step is active. Variable *valveA* is linked to the logic controller output 1 that is then fed to the discrete/continuous part of the interface and transformed to the volume flow, which presents the continuous process input signal. Inside the reactor model the input flow signal is integrated to simulate the filling of the tank. The integrator output (level) is monitored by the continuous/discrete part of the interface and when the predefined level is reached, the logic controller input 2 is set to 1. This is linked to the global variable *levelA*, which is assigned to transition condition of *SFCTransition2*. Transition fires, which causes *SFCStep2* to deactivate and at the same time *SFCStep3* and *SFCStep4*

Table 2
Control interpretation of places and transitions in Fig. 10

Places	
p_{dNS2}	Move distribution piston backward
p_{d3}	Move distribution piston forward
p_{d4}	Move manipulator arm toward input buffer
p_{d5}	Hold the workpiece
p_{d6}	Move manipulator arm toward lift (carrying workpiece)
p_{d7}	Release the workpiece
p_{d8}	Move manipulator arm toward input buffer
p_{r4}	Eject the workpiece (to waste), delay
p_{r5}	Move the lift up
p_{r6}	Delay
p_{r7}	Eject the workpiece (to the next station), delay
p_{drNS2}	Move the lift down
p_{drNS4}	Move eject piston backward
p_{drNS6}	Move manipulator arm toward lift
Transitions	
t_{dNSI}	Distribution piston back (switch closed)
t_{d2}	Workpiece present (optical sensor)
t_{d4}	Distribution piston forward (switch closed)
t_{d5}	Manipulator arm at input buffer (position switch closed)
t_{d6}	Vacuum on (vacuum switch closed)
t_{d7}	Manipulator arm at lift (position switch closed)
t_{d8r1}	Vacuum off (vacuum switch open)
t_{d9}	Manipulator arm at input buffer (position switch closed)
t_{r2}	Black workpiece detected
t_{r3}	Red workpiece detected
t_{r4}	Metal workpiece detected
t_{r5}	Bad workpiece detected
t_{r7}	Bad workpiece
t_{r8}	Delay timer on
t_{r9}	Good workpiece
t_{r10}	Lift up (position switch closed)
t_{r11}	Delay timer on
t_{r12}	Delay timer on
t_{drNS1}	Lift down (position switch closed)
t_{drNS2}	Eject piston back (switch closed)
t_{drNS4}	Manipulator arm at lift (position switch closed)

are activated. Variable *valveA* is set to 0 and global variable *valveB* that corresponds to *SFCStep4* action is set to 1. The *SFCStep4* action contains an *L* qualifier and a parameter 100. A time event is therefore scheduled for time of the current simulation time plus 100 s. Up to this time the integration continues with a changed input signal that now corresponds to the input flow B.

The rest of the SFC steps and actions are executed in a similar manner. In this way the bi-directional communication is established between the discrete and the continuous parts of the model. By simulating, the correct operational sequencing can be tested and the duration of a batch cycle can be evaluated according to different temperature control loop tuning parameters and disturbances on the reactor input flows.

3.2. Laboratory scale manufacturing system

The second case study deals with a laboratory scale manufacturing system from FESTO DIDACTIC. The system represents a manufacturing production line and includes typical functions like distribution of workpieces, testing, processing, handling and storage of final products. The system is decomposed into five stations, covering one of the above functions each (Fig. 8).

Because of the complexity of the complete model only the first two stations (distribution and testing) are considered here in this paper. The first station performs the distribution of the workpieces, which are stored in the input buffer. A workpiece is pushed from the buffer by a pneumatic piston and then carried to the next station by a manipulator, which consists of a rotating arm and a vacuum holder. In the second station the workpiece is tested for colour and dimension. If the results meet the requirements, the workpiece is lifted up and pushed toward the processing station. If a workpiece is classified as bad, it is ejected to the waste buffer. In order to design a logic control for the two stations the desired behaviour is modelled by a Petri net. A systematic hybrid modelling approach is used which combines top–down model refinement with shared resources that are added in a bottom–up manner [11].

A simple Petri net, which has the desired properties of boundedness, liveness and reversibility, is chosen as a coarse starting model of the modelled system (Fig. 9). The places of the starting Petri net are then systematically refined by the use of well-defined modules, which maintain the desired properties of the model. When the model becomes appropriately detailed, resources are added, in our case these are the distribution piston and the joint of manipulator arm and lift.

The resulting model (for the first two stations only) is shown in Fig. 10 with the control interpretation of places and transitions given in Table 2. Places not listed in the Table 2 correspond to states of the process that do not require any specific control actions, they merely describe the status of a corresponding device (e.g., P_{drNS1}—manipulator arm and lift ready) or result from the system decomposition and are used as linking elements (e.g., p_{r3}, p_{r8}). Transitions not listed in the Table 2 are either uncontrollable (e.g., t_{d1}) or fired as soon as they are enabled (t_{d3}, t_{r6}, t_{drNS3}).

The SFC model of the corresponding logic controller is derived from the Petri net model, through a similar procedure to the previous example. A model of the process itself is also needed in order to perform the simulation of the controlled system. The process consists of several partially independent

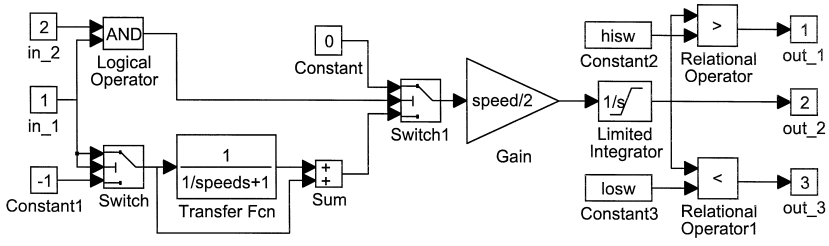


Fig. 11. Continuous model of a discretely controlled pneumatic piston.

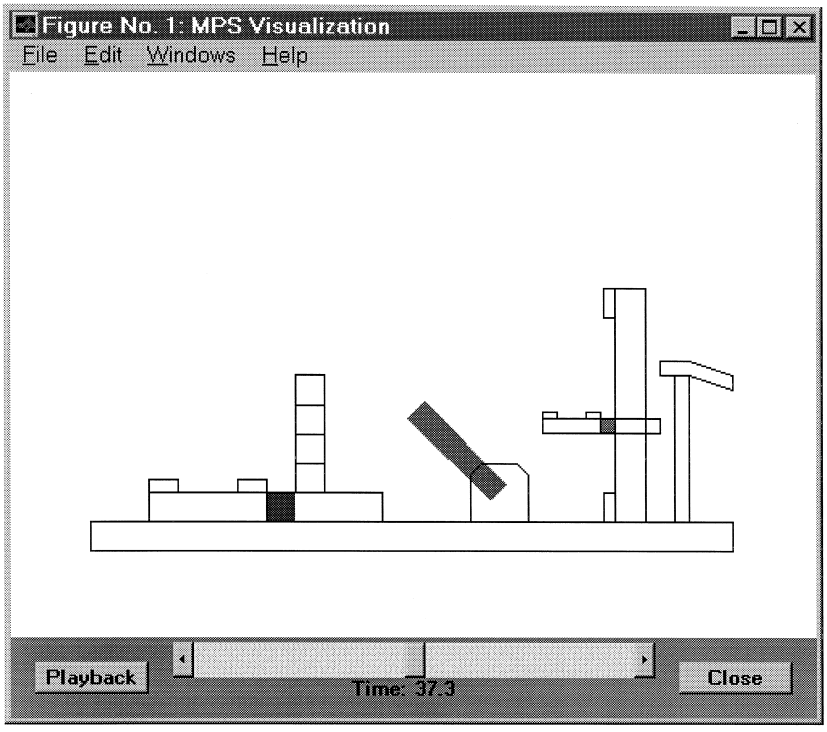


Fig. 12. Matlab animation of the first two stations.

modules like pistons, rotational pneumatic gears, etc. Depending on the amount of detail required the individual units can be modelled by continuous models or by more abstract discrete models (e.g., Petri nets or SFC). An example of an unidirectional pneumatic piston simulation model is shown in Fig. 11. The piston moves forward when the input 1 is at logical 1 and moves backward when the input 1 is at logical 0. Movement of the piston is limited to the interval between zero and maximum distance. Input 2 represents a mechanical blockage in front of the piston, therefore the piston holds if moving forward and input 2 receives logical 1, while movement backward is not affected by input 2. Outputs of the model represent the position switch at the maximum outer position of the piston, position of the piston and the position switch at the starting position of the piston, respectively. Note that outputs 1 and 3 are actually part of the continuous/discrete interface.

In any case, whether the individual units are modelled by continuous models or by more abstract discrete models, the resulting combination of SFC and continuous models can be simulated in the Simulink environment to verify the desired system properties. The graphic animation capabilities of Matlab are used to present the simulated model behaviour (Fig. 12).

4. Conclusions

The Sequential Function Chart extension to the Matlab–Simulink environment has been developed and it has been shown how it can support control design in process control and manufacturing. Such an extension of a continuous simulation tool enables a combination of continuous and discrete event parts of the model, verification of the desired behaviour and evaluation of various changes in the system. The simulation environment proved itself valuable in testing various sequential control strategies and can be used as a general-purpose sequential control logic program testing environment. One of the issues for further work, is the presentation of the simulation

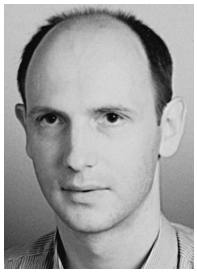
results. In the predominantly continuous systems, time responses are appropriate, while another representation is required in the manufacturing area. One of the possible solutions is an animated graphical display. A link between Simulink and an industrial standard SCADA package is foreseen which will enable the presentation of the simulation results in various graphic display formats.

Acknowledgements

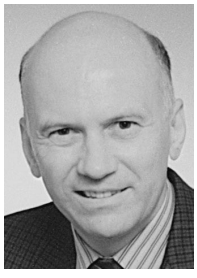
The authors are very grateful to G.C. Premier for his comments and suggestions that helped us to improve the presentation of the paper.

References

[1] J. Buisson, Analysis and Characterisation of Hybrid System with Bond-Graphs, IEEE International Conference on SMC, Le Touquet, France, 1993, pp. 264–269.
[2] R. David, H. Alla, Petri nets for modeling of dynamic systems—a survey, Automatica 30 (2) (1994) 175–202.
[3] I. Demongodin, N.T. Koussoulas, Modeling Dynamic Systems Through Petri Nets, CESA'96 IMACS Multiconference, Symposium on Discrete Events and Manufacturing Systems, Lille, 1996, pp. 279–284.
[4] P.J. Anstsaklis, J.A. Stiver, M. Lemmon, Hybrid System Modeling and Autonomous Control Systems, in: R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.), Hybrid Systems, Lecture Notes on Computer Science, Vol. 736, Springer Verlag, 1993, pp. 366–392.
[5] D. Andreu, J.C. Pascal, R. Valette, Events as a Key of a Batch Process Control System, CESA'96 IMACS Multiconference, Symposium on Discrete Events and Manufacturing Systems, Lille, 1996, pp. 297–302.
[6] O. Stursberg, S. Kowalewski, S. Engell, Generating Timed Discrete Models of Continuous Systems, Proceedings of the 2nd MATHMOD, IMACS Symposium on Mathematical Modelling, Vienna, 1997, pp. 203–209.
[7] F.E. Cellier, H. Elmquist, M. Otter, J.H. Taylor, Guidelines for Modelling and Simulation of Hybrid Systems, Proc. IFAC 12th Triennial World Congress, Vol. 8, Sydney, 1993, pp. 391–397.
[8] IEC, International Electrotechnical Commission, Programmable Controllers: Part 3. Programming Languages, publication 1131.3, 1992.
[9] R. David, Grafcet: a powerful tool for specification of logic controllers, IEEE Trans. on Control Systems Technology 3 (3) (1995) 253–268.
[10] G. Mušič, D. Matko, Modelling and Simulation of Supervisory Process Control Systems, Proceedings of the 2nd MATHMOD, IMACS Symposium on Mathematical Modelling, Vienna, 1997, pp. 247–252.
[11] M. Zhou, F. DiCesare, A.A. Desrochers, A hybrid methodology for synthesis of Petri net models for manufacturing systems, IEEE Trans. on Robotics and Automation 8 (3) (1992) 350–361.



Gašper Mušič received his BSc and MSc degrees in electrical engineering from the University of Ljubljana, Slovenia in 1992 and 1995, respectively. He is currently a probationary assistant at the Faculty of Electrical Engineering, University of Ljubljana, Slovenia and is working toward PhD degree in automatic control. His research interests are in discrete event and hybrid dynamical systems, supervisory control, and applications in industrial process control.



Drago Matko received his BSc, MSc and PhD in electrical engineering in 1971, 1973 and 1977, respectively, from the University of Ljubljana, Slovenia for work in the field of Adaptive control systems. He is a professor of Discrete control systems, Computer controlled systems, Identification and Computer aided design of control systems on the Faculty of Electrical Engineering, University of Ljubljana. He visited the Institute for Control Engineering Darmstadt, Germany several times, from 1980–1982, 1984, 1985 and 1986 as a Humboldt fellow, from 1987–1991 in the frame of the project Adaptive control systems sponsored by International Bureau KFA Jülich. In 1995–1996 he was visiting as Foreign Research Fellow the Institute of Space and Astronautical Science in Sagamihara Kanagawa, Japan for nine months. He has published seven journal articles, more than 150 conference papers and four student edition books (in Slovene). He is co-author of two books published by Prentice Hall. In 1989, he received the award of Slovenian ministry for research and technology for his work in the field of computer aided design of control systems.