Mathematical and Computer Modelling of Dynamical Systems
2000, Vol. 6 No. 2, pp. 169–189

1387-3954/00/0602-0169$15.00
©Swets & Zeitlinger

Modelling, Synthesis, and Simulation of Supervisory Process Control Systems

G. MUŠIČ*, D. MATKO* AND B. ZUPANČIČ*

ABSTRACT

Modelling, synthesis, and simulation issues of the supervisory systems in process control are investigated in the paper. Petri nets are used as a basic modelling framework for the supervisory part of the system. It is shown how the final verification effort can be minimised by applying formal synthesis methods. A straightforward approach to the industrial implementation of the developed solutions is suggested by means of sequential function chart representation. A batch process cell case study is used to illustrate the described concepts. Corresponding continuous and discrete event models of the process cell units are developed and a co-ordinating supervisor is designed by the method of place invariants. The system is simulated by the continuous simulation tool Matlab-Simulink, which is enhanced for simulation of the sequential control logic represented by sequential function chart.

Keywords: batch processing, discrete event systems, modelling, petri nets, process control, simulation, supervisor synthesis.

1 INTRODUCTION

Supervisory control mechanisms can be found in a variety of production control systems. Also in the process industries, where the basic control system level usually comprises several mainly continuous control loops that are governed by digitally implemented industrial loop controllers, the complexity of present day production processes often requires the co-ordination of underlying subprocesses to achieve the desired production goals. The basic control level is therefore superimposed by higher levels of the control hierarchy, ranging from supervisory control, production planning to business process management.

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

Despite the fact that the local control deals with processes that are usually continuous in nature, the supervisory control system acts discretely. The supervisor deals with the state-transitions of underlying processes and is a discrete event system, which changes its discrete state as a reaction to external discrete events and performs external actions according to its state. In many cases, the supervisor can be treated as a discrete controller applied to continuous system.

Most of the current research in the area of supervisory systems is focused on the idea of synthesis of the supervisor for a given discrete event model of the plant. The plant is uncontrolled from the supervisory point of view; on the other hand, it usually contains several control units (programmable logic controllers or closed loop controllers) which are already incorporated in the plant model. The aim of such a supervisor is to force the plant model to have desired properties, which correspond to additional system specifications. Modelling of the plant is a crucial phase in the supervisory system synthesis procedure.

The paper concentrates on modelling for the purpose of supervisor synthesis and possible implementation by sequential function charts. The modelling of the process plants and the corresponding supervisors is discussed in Section 2. The supervisory synthesis method based on place invariants is briefly reviewed and illustrated by a simple example in Section 3. The approach is demonstrated by a case study from the area of batch systems in Section 4.

2 MODELLING

Various discrete event modelling techniques can be used to obtain the model of a plant and a supervisor. No general agreement has been achieved yet upon a modelling framework that would best suit the needs of analysis and design of supervisory systems, nevertheless, discrete transition system descriptions such as finite automata [4] or Petri nets are used most commonly.

The finite automata representation of the supervisors and plants is used in most of the early works on supervisory control. Applied synthesis methods are based on searching over the state space of the automata. One of the main problems of the application of the developed supervisory framework to real industrial processes is the state explosion problem. Petri nets, on the other hand, enable the modeller to include additional structural information into the model. The state of the system is distributed over the places of the net. The Petri net based synthesis methods tend to exploit the net structure thus reducing the need to search over the whole state space. In this way the state explosion problem can be avoided. Some recent contributions on the Petri net based supervisory control can be found in [3,7,8,14,17]. The main problem of the application of the developed methods is that they are generally limited to a particular type of Petri net models and allow only certain types of system specifications. Petri net models, however, are relatively easily understood by non-experts and yield the possibility of a more intuitive design based on the knowledge about the process and not purely on the rigid theoretical background. This can be helpful in cases that extend beyond the limitations of a particular synthesis technique.

Perhaps the most significant advantage of the Petri net representation is the straightforward path from the developed models to the industrial implementation. The discrete control logic is most often implemented by programmable logic controllers. The recent IEC standard on programming languages of industrial logic controllers [5] promotes the use of Sequential Function Chart (SFC) representation of the control logic. SFC (also referred as Grafcet) inherited many of its features from the theory of Petri nets. More precisely, a safe interpreted Petri net can be defined such that its input-output behaviour is the same as the input-output behaviour of a SFC [1,2].

A place in such a Petri net corresponds to a step in a SFC. Transitions and directed links have the same meaning in a SFC as they have in a Petri net. If an input/output interpretation is added to the transitions and places of a Petri net, we can obtain an equivalent SFC model. There are however two basic differences between an interpreted Petri net and a SFC. First there is a difference in the marking of a SFC, which is Boolean (step is active or inactive) while the marking of a place in a Petri net can be any positive integer. For that reason the conversion of a Petri net to a SFC is only possible when the net is safe (i.e., for any reachable marking, the marking of every place is less than or equal to one).

There is also a difference in firing rule of a Petri net and a SFC when some transitions are enabled by the marking of the same place (step). This leads to a non-deterministic behaviour of a Petri net since there is no rule to choose which of the enabled transitions will be fired. When such a situation emerges in a SFC the transitions are fired according to their priority to ensure the deterministic behaviour. On the other hand, if a Petri net is such that any pair of transitions in conflict have receptivities which can not be true at the same time, the behaviour of the net is deterministic. If such a Petri net is also safe, it is equivalent to a SFC [2].

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 [1], can be applied also to SFCs. Thus the supervisor can be derived within the Petri net supervisory control framework and the derived model can then be easily transformed into a logic controller program.

2.1 Petri net modelling

A Petri net is a directed bipartite graph which has two types of nodes called places and transitions [1,9,12]. Formally, a Petri net is a five-tuple $\mathrm{PN} = (P, T, A, W, M_0)$, where $P = \{p_1, p_2, \ldots, p_m\}, m > 0$ is a finite set of places, $T = \{t_1, t_2, \ldots, t_n\}, n > 0$ with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$ is a finite set of transitions, $A \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs, $W : A \rightarrow \{1, 2, \ldots\}$ is a weight function, and $M_0 : P \rightarrow \{0, 1, 2, \ldots\}$ is the initial marking.

For the purpose of logical modelling required in supervisor synthesis we use the class of ordinary Petri nets. This means all the arc weights are equal to one and no time is involved in the firing of transitions. The switching rule is given as follows: a) a transition is enabled if each of the input places of this transition contains at least one token, b) an enabled transition may or may not fire, which depends on an additional interpretation, c) a firing of a transition is immediate (includes no delay) and removes a token from each of the input places of the transition and adds a token to each of the output places of the transition. For the purpose of simulation and possible implementation by industrial controllers, the input/output interpretation can be added to resulting models.

A review of Petri net synthesis techniques for modelling of automated manufacturing systems can be found in [6]. The research on synthesis of Petri net models can be summarised into two basic approaches: bottom-up and top-down. The bottom-up approach begins with the synthesis of subnets that correspond to separate sub-processes. These subsystem models are usually simple to verify for the desired properties. The subnets are then combined into final net by merging places and/or transitions of the subnets. If some rules are followed during merging, place and/or transition invariants [9] of the resulting net can be obtained from the invariants of the subnets. Some properties of the resulting net can be analysed using these invariants. The disadvantage of such an approach is that invariants do not convey the complete information of the system and some properties are difficult to analyse [9].

Top-down approach starts with the top-level model of a system in a form of a Petri net with desired properties. Then the starting net is refined in a step-wise manner. Refinements can be done by expanding places or expanding transitions and continue until the implementation level is reached. Methods were derived which ensure that the important properties are kept during refinements so that a final analysis of the system is not necessary. However, the top-down method is difficult to apply when several interactions exist in the system, such as when the system contains a number of shared resources. This motivates the development of hybrid methods that combine the top-bottom system refinements with shared resources that are added in a bottom-up manner [16].

Plants in process industries generally exhibit less flexible structure than manufacturing processes. The individual process units are well defined and standardised, which indicates that bottom-up model synthesis method might be appropriate. The area of batch systems, involving many higher-level supervisory functions, for example, comprises many generic concepts that are found in practically every plant. The Instrument Society of America (ISA) has set a standard (SP88) which defines the consistent set of terminology and models used to describe the control requirements for batch manufacturing plants. Generic Petri net submodels can be defined in accordance with the standard and be used as a base for modelling of such plants.

A batch process plant usually consists of many pipes and vessels that are involved in transforming a batch from raw materials to final products. The transport of material over the system is under control of a set of discretely operating devices such as pumps and valves that can only be in one of two states (on/off, open/closed). Inside reactor vessels several continuous processes that are controlled by continuous controllers take place. The controllers are only operational when the material is present in the reactor and their status can also be described by two states (in operation/idle). Similarly, many continuous variables are only checked for a few limit values. Corresponding sensors give either a discrete output value (e.g., liquid level is above/below the limit) or a continuous value that is compared to limits within the local controller program. In both cases, such a sensor can be modelled by a two-state (or finite-state) automaton. In this view the batch process can be described by a set of finite-state sensors and actuators. The sequence of state changes of these devices is defined by a recipe. Such a sequence can be modelled by a Petri net where places in the net correspond to operations or availability of various resources. Transitions correspond to state changes of the system and are linked to events in the system (transitions of sensor states) that should trigger the start or the end of a particular operation. The Petri net model of the recipe is related to models of physical devices as depicted in Figure 1.

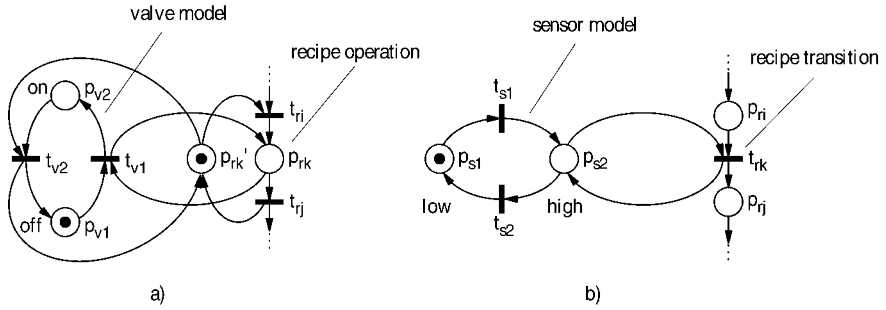


Fig. 1. Relation of recipe operations to physical devices.

Figure 1a shows the relation of a recipe operation to an actuator device such as an on/off valve. The operation is represented as a place in the corresponding Petri net model. The state of the valve is changed from *off* to *on* after firing of transition t_{ri} (start of the operation) when the operation place p_{rk} becomes marked. The state is changed back to *off* after firing of transition t_{rj} which corresponds to the end of the operation. It is assumed that transitions t_{v1} and t_{v2} fire as soon as they are enabled. Place p'_{rk} acts as a complementary place to p_{rk}. Figure 1b shows the link between a sensor device and a recipe transition. The transition t_{rk} between two successive operations (represented by places p_{ri} and p_{rj}) can only fire when the corresponding sensor is in the state *high*, which is represented by marking of the place p_{s2}. Transitions t_{s1} and t_{s2} in the sensor model are associated with events in the process, e.g., a measured variable crossing a threshold value in the specified direction.

Both sensor and actuator models are connected to the recipe model by means of self-loops only. This implies that for the purpose of supervisory controller synthesis it is adequate to model the recipes only, assuming that the recipes themselves maintain the proper operation of the corresponding process units. Note that the addition of the place p'_{rk} does not change the behaviour of the recipe model only when the place p_{rk} is safe. Safety of places in a recipe model is an important property, which enables an interpretation of the recipe model as a logic controller specification.

Consider a simple example from the area of batch systems. Part of a batch process cell is shown in Figure 2. Two mixing tanks share the same supply tank. Mixing tanks are repeatedly filled and discharged with the restriction that only one tank can be filled at a time.

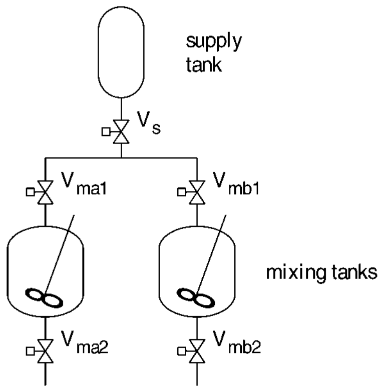


Fig. 2. Part of a batch process cell.

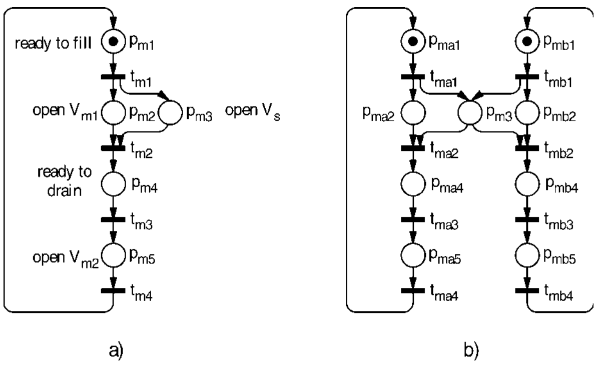


Fig. 3. Petri net model.

First the Petri net models of the two individual process lines are derived. A process line is defined by the ISA SP88 standard as the set of equipment used to produce one batch. Lines can be configured to combine the equipment differently for different products or batches. In the given case, the two lines consist of a supply tank and a mixing tank each. The two models are identical and a corresponding Petri net is depicted in Figure 3a. The models are combined by merging the places that correspond to the same status or operation. In the example these are the places p_{ma3} and p_{mb3} that both correspond to the outlet valve of the supply tank. Letter 'a' or 'b' that is introduced into place/transition subscript denotes to which tank the corresponding place/transition belongs. Figure 3b shows the Petri net, obtained by merging the two places.

The classical bottom-up approach to Petri net modelling provides special rules, which define places that are allowed to be merged. This ensures that important properties of the subnets are preserved in the final net. These rules were not taken into account in the previous example and such merging does not guarantee that any of the important properties is preserved. However, this concept enables much greater flexibility in modelling individual units and better suits the distributed system architecture. The problems that arise can be overcome by the synthesis of a supervisor that prevents any undesired behaviour of the plant. Such a supervisor corresponds to a co-ordination level required when merging several locally controlled subprocesses.

3 SUPERVISOR SYNTHESIS

One way of including supervisory mechanisms is the mutual exclusion concept introduced in [15,17]. Two concepts, parallel and sequential mutual exclusions are defined and used to synthesise bounded, live and reversible Petri net. In this classical mutual exclusion concept, all transitions are assumed to be controllable, i.e., may be prevented from firing by a supervisor. This assumption however, is rather unrealistic. Therefore, in the traditional supervisory control framework [13] the complexity of enforcing desired properties is enhanced by the presence of uncontrollable transitions. Different approaches based on Petri net models and which also consider the problem of uncontrollable transitions are given in [3,7,8,14]. The last approach [8,14] is based on place invariants and is particularly interesting, because the resulting supervisory mechanism is computed very easily. It is briefly summarised here and illustrated by the above example.

By the method of place invariants it is possible to enforce a set of n_c constraints on the plant state m_p. The plant state is represented by a $m \times 1$ marking vector of non-negative integers, where each vector component is equal to the marking of the corresponding place in the Petri net model of the plant. Constraints are written in the form

$$Lm_p \leq b \tag{1}$$

where L is a $n_c \times m$ integer matrix and b a $n_c \times 1$ integer vector [14]. The inequality (1) is read with respect to each element on the corresponding left and right hand sides of the inequality. It is shown in [14] that if the initial marking does not violate the given set of constraints, (1) can be enforced by a supervisor with the incidence matrix

$$D_c = -LD_p \tag{2}$$

where D_p is the $m \times n$ incidence matrix of the plant. The initial marking of the controller is computed by

$$m_{c_0} = b - Lm_{p_0} \tag{3}$$

where m_{p_0} is the $m \times 1$ initial plant marking vector of non-negative integers. The supervisor consists of n_c places that are linked to the existing transitions of the plant. With the addition of supervisor places the overall system is given by

$$D_s = \begin{bmatrix} D_p \\ D_c \end{bmatrix} \qquad m_s = \begin{bmatrix} m_p \\ m_c \end{bmatrix} \tag{4}$$

and every single constraint is transformed to a marking invariant that corresponds to a place invariant [1] of the supervised system.

Consider the above example with the current marking of Petri net as shown in Figure 3b. Obviously, if transitions t_{ma1} and t_{mb1} fire, the place p_{m3} contains two tokens and is therefore not safe. The safeness of the place p_{m3} is required, because it represents the operation of opening the outgoing valve of the supply tank and this can not be opened twice at the same time. The described situation is a malfunction of the system. To prevent this the supervisor has to be designed that will co-ordinate the two mixers in such a way that only one will be filled at a time. This requirement is written as

$$\mu_{m3} \leq 1$$

where μ_{m3} is the marking vector component that corresponds to place p_{m3}. The requirement can be easily transformed to the form (1). With the marking vector being $m_p = [\mu_{ma1}, \mu_{ma2}, \mu_{m3}, \mu_{ma4}, \mu_{ma5}, \mu_{mb1}, \mu_{mb2}, \mu_{mb4}, \mu_{mb5}]^T$, we have $m_{p_0} = [1,0,0,0,0,1,0,0,0]^T$, $L = [0,0,1,0,0,0,0,0,0]$, and $b = 1$, the supervisor can be computed by (2) and (3). Given D_p as

$$D_p = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

we obtain the supervisor

$$D_c = [-1 \quad 1 \quad 0 \quad 0 \quad -1 \quad 1 \quad 0 \quad 0]$$

$$m_{c_0} = 1$$

The supervisor consists of a single place that is connected to the plant Petri net as shown in Figure 4.

The marking invariant that is enforced by the supervisor is:

$$\mu_{m3} + \mu_{c1} = 1$$

It can be shown that markings of all other places of the resulting net are all included by at least one marking invariant that has a sum of tokens equal to one. Such a set of marking invariants is, for example

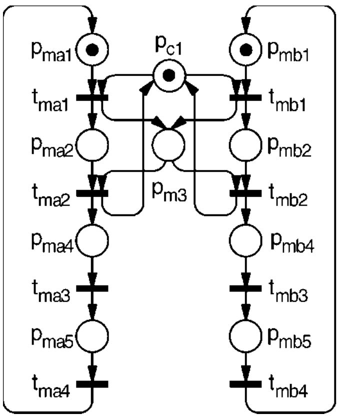


Fig. 4. Petri net model of the supervised system.

$$\mu_{ma1} + \mu_{ma2} + \mu_{ma4} + \mu_{ma5} = 1$$
$$\mu_{mb1} + \mu_{mb2} + \mu_{mb4} + \mu_{mb5} = 1$$

The net is therefore proven to be safe.

The problem of uncontrollable transitions does not appear in the given example. But generally, some transitions are always found uncontrollable. These are, e.g., all transitions that represent sensor readings as well as come control actions that must not be prevented due to required process operation or safety reasons.

Let D_{uc} represent the columns in the process incidence matrix that correspond to uncontrollable transitions. Clearly, the firing of the uncontrollable transitions must not depend on the marking of any place that belongs to the supervisor. The supervisor matrix D_c must therefore contain no negative elements in the columns that correspond to uncontrollable transitions (a supervisor designed by the described method contains no self loops, so this condition is sufficient). This is true when the matrix LD_{uc} contains no positive elements as these will appear as negative elements in D_c calculated by (2). The set of constraints must therefore satisfy

$$LD_{uc} \leq 0 \tag{5}$$

If this is not the case, matrix L (and eventually vector b) must be transformed so that (5) will be satisfied while the supervisor designed to fulfil the new set of constraints will also maintain the original set of constraints. This can be achieved by performing row operations on D_{uc} and LD_{uc}. A corresponding algorithm is given in [8]. The given constraint transformation is linear and therefore does not always yield an optimal solution in the sense that a supervisor should not prevent any transition from occurring unless it is really necessary. However, the supervisor can always be calculated by (2) and (3), which is computationally very efficient and presents an advantage compared to some other approaches (e.g., [7]).

The described concept can be effectively used to introduce resource allocation strategies and co-ordination mechanisms in the areas such as batch system control which will be further illustrated later in the case study.

4 CASE STUDY

A case study of a batch process cell is given in this section to illustrate the discussed concepts. The process cell is shown schematically in Figure 5. It consists of several input buffers, two mixing tanks and two reactor vessels. A single batch reactor is shown in detail in Figure 6a. Each reactor has two inlets for two incoming substances and the output 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_{ra} and V_{rb}) and the discharging is controlled by the on/off valve V_{rc}. The maximum and minimum level is sensored by two level switches (S_{ra} and S_{rb}). The temperature of the reaction is controlled by feeding hot or cold water through the water jacket, which surrounds the reactor vessel.

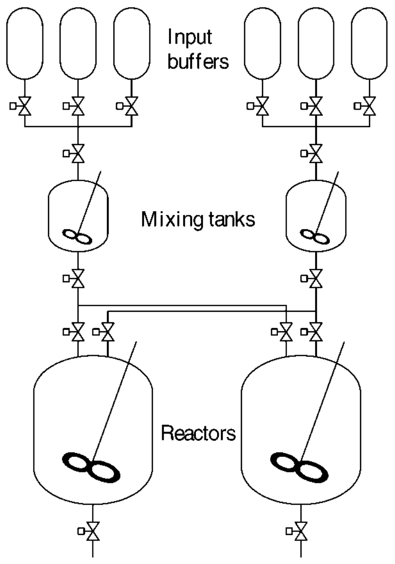


Fig. 5. Batch process cell.

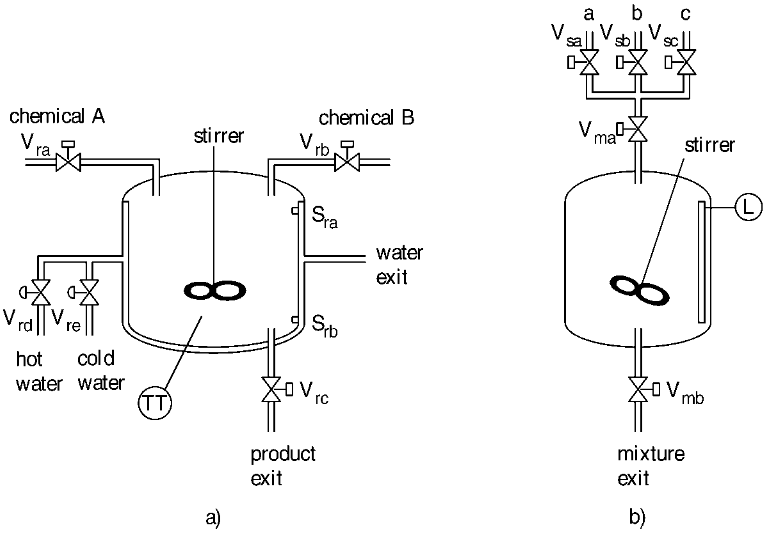


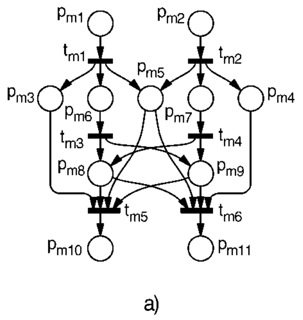
Fig. 6. Detailed view of the batch reactor (a) and the mixing tank (b).

The water flow is controlled by adjusting the proportional valves V_{rd} and V_{re} and the temperature of the reactor contents is measured by the temperature sensor *TT*.

The preparation of the input substances takes place in two mixing tanks to which the raw material is supplied from three supply tanks. A mixing tank is shown in detail in Figure 6b. The substance is composed from one of the two basic components (component 'a' or component 'b') that is diluted to the required concentration by component 'c'. The filling of the mixing tank is controlled by the on/off valve V_{ma} in combination by one of the supply tank valves V_{sa}, V_{sb} or V_{sc}. The discharging of the mixer is controlled by the on/off valve V_{mb}. The level in the mixing tank is measured by the level sensor *L*. The required quantities of each input depend on the recipe.

4.1 Modelling

Petri net models of the system's components (mixer and reactor) are shown in Figure 7, the interpretation of places and transitions is summarised in Table 1. The basic recipe for the mixer is as follows: a) fill component a (places p_{m3}, p_{m5}, p_{m6}) or b (places p_{m4}, p_{m5}, p_{m7}) b) start the stirrer and add component c (places p_{m8}, p_{m9}) c) stop the stirrer and wait for discharging (places p_{m10} and p_{m11}). The basic recipe for the reactor is as follows: a) load chemical A (places p_{r3}, p_{r4}), b) start the stirrer and load chemical B (places p_{r5}, p_{r6}, p_{r7}), c) switch on the temperature controller to heat up the mixture to the required reaction



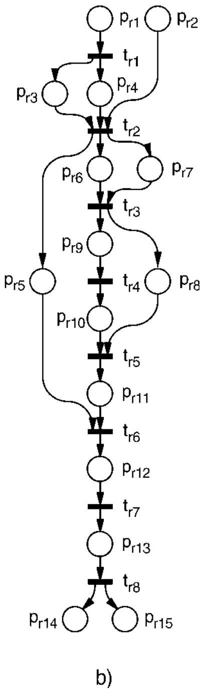


Fig. 7. Mixing tank model (a) and reactor model (b).

temperature (places p_{r8}, p_{r9}), d) when the set-point temperature is reached start the timer to time the duration of the reaction (place p_{r10}), e) cool down the reactor contents (place p_{r11}) f) remove the product from the reactor (place p_{r12}) g) wait for the start of new batch (place p_{r13}).

The final batch recipe model is obtained by merging the models of the two mixers and the two reactors. To distinguish between the places and transitions that correspond to one of the two mixing tanks and the two reactors, the letter 'a' or 'b' is introduced into the place and transition subscripts. Place p_{ma1} (mixing request A) is merged with place p_{ra14} (request A), p_{mb2} (mixing request B) is merged with place p_{ra15} (request B). Similarly, p_{mb1} is merged with p_{rb14} and p_{ma2} with p_{rb15}. Place p_{ma10} (substance A ready) is merged with place p_{ra1} (chemical A ready), place p_{ra2} (chemical B ready) is merged with place p_{mb11} (substance B ready) of the second mixer. A similar procedure is performed for the second reactor. Finally, the places that correspond to operation of the same physical device are merged. In the given case these are places p_{ra4} and p_{rb7} that correspond to opening of the mixing tank discharge valve. Similarly we proceed with the places p_{ra7} and p_{rb4} that correspond to the discharge valve of the second mix-

Table 1. Control interpretation of places and transitions.

Mixer places

p_{m1}	Mixing request A
p_{m2}	Mixing request B
p_{m3}	Preparing A
p_{m4}	Preparing B
p_{m5}	Open mixer filling valve
p_{m6}	Fill component a
p_{m7}	Fill component b
p_{m8}	Fill component c
p_{m9}	Stir mixing tank contents
p_{m10}	Substance A ready
p_{m11}	Substance B ready

Mixer transitions

t_{m1}	Start preparing substance A
t_{m2}	Start preparing substance B
t_{m3}	Component a filling level
t_{m4}	reached
$t_{m5,m6}$	Component b filling level

Reactor places

p_{r1}	Chemical A ready
p_{r2}	Chemical B ready
p_{r3}	Fill reactor with chemical A
p_{r4}	Discharge the mixing tank a (b)
p_{r5}	Stir reactor tank contents
p_{r6}	Fill reactor with chemical B
p_{r7}	Discharge the mixing tank b (a)
p_{r8}	Temperature controller enabled
p_{r9}	Heat up the mixture
p_{r10}	Reaction timer running
p_{r11}	Cool down reactor contents
p_{r12}	Discharge the reactor
p_{r13}	Reactor ready
p_{r14}	Request A
p_{r15}	Request B

Reactor transitions

t_{r1}	Start filling chemical A
t_{r1}	Mixing tank A empty
t_{r1}	Mixing tank B empty
t_{r1}	Setpoint temperature reached
t_{r1}	Reaction timer run out
t_{r1}	Output temperature reached
t_{r1}	Reactor empty
t_{r1}	Start a new batch

er. After merging these places, the final Petri net of the process cell shown in Figure 8 was obtained. The control interpretation of the places and transitions of the derived Petri net model is the same as for the individual unit models (given in Table 1).

4.2 Synthesis of the supervisor

The derived Petri net represents the sequence of operations required to make a product and is a part of a master recipe for the batch process cell. It is required that all the control places of the net (places that represent the state of a physical device) are safe. The analysis shows, however, that the obtained net is not safe. Consider the situation where places p_{ma1} and p_{ma2} are marked. If the transitions t_{ma1} and t_{ma2} fire, the place p_{ma5}, which represents the mixer filling valve, con-

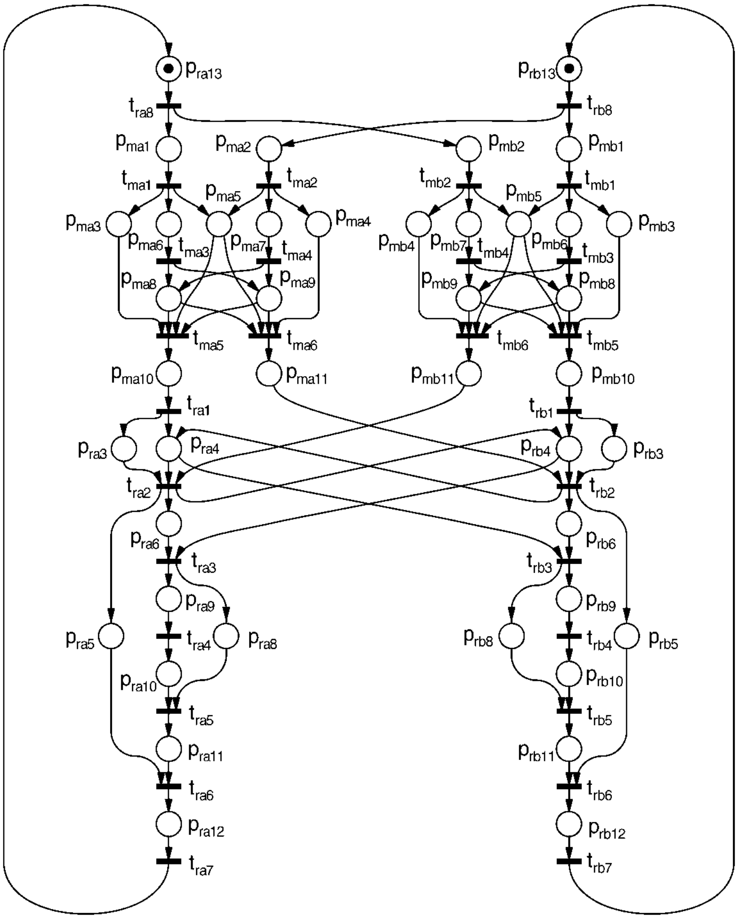


Fig. 8. Batch recipe model.

tains two tokens. This signals an attempt to use a device more than once at the same time. Obviously, some co-ordination between the two reactors should be performed to prevent such attempts. Since both mixers are used to fill both reactors, this is solved by a resource allocation mechanism that prevents a double booking of a mixing tank.

An additional co-ordination requirement is given by the system specification. In the phase of heating up the mixture to the reaction temperature a large amount of energy is required, while only a small flow of hot water is required afterwards to keep the mixture at the constant temperature. Therefore it is desired to co-ordinate the heating up phase between the two reactors as well.

Here we have an additional restriction that the transitions t_{ra2}, t_{ra3}, t_{rb2} and t_{rb3} are uncontrollable, because the filling of the reactor with chemical A must be immediately followed by filling of chemical B and the heating-up phase must start immediately after the reactor is filled with both raw materials. All these requirements are fulfilled by a supervisory mechanism that is designed formally by the method of place invariants.

The co-ordination requirement is written as marking constraint in the form:

$$\mu_{ra9} + \mu_{rb9} \leq 1 \tag{6}$$

Because the transitions t_{ra3} and t_{rb3} are uncontrollable, the marking of places p_{ra9} and p_{rb9} can not be controlled directly (if we compose the constraint matrix L and the uncontrollable part of the incidence matrix D_{uc}, the condition (5) is not satisfied) and the constraint (6) must be modified. The problem is solved as indicated in Section 3. In the modified constraint the marking of places p_{ra3}, p_{ra6}, p_{rb3} and p_{rb6} is included and the condition becomes

$$\mu_{ra3} + \mu_{ra6} + \mu_{ra9} + \mu_{rb3} + \mu_{rb6} + \mu_{rb9} \leq 1$$

Similarly, the resource allocation requirements are written as

$$\mu_{ma5} + \mu_{ma10} + \mu_{ma11} + \mu_{ra4} \leq 1 \tag{7}$$

$$\mu_{mb5} + \mu_{mb10} + \mu_{mb11} + \mu_{rb4} \leq 1 \tag{8}$$

Note, that markings of places p_{ma3}, p_{ma4}, p_{ma6}, p_{ma7}, p_{ma8} and p_{ma9} are not included in the constraint (7), because these places can only be marked in parallel with p_{ma5}. A similar observation holds for constraint (8).

A simple resource allocation strategy as given by constraints (7) and (8) may often lead to a deadlock in the process. This is also the case in the above example. In the case when two batches are started simultaneously, both mixers may start preparing the same substance, e.g., substance B. When this is ready, reactors keep waiting to substance A but this is never ready because both mixers are booked. The system falls into a deadlock. Obviously, additional co-ordination between the two reactors is required to prevent such a deadlock. This is achieved by the following constraints

$$\mu_{ma3} + \mu_{ma10} + \mu_{ra3} + \mu_{mb3} + \mu_{mb10} + \mu_{rb3} \leq 1$$

$$\mu_{ma4} + \mu_{ma11} + \mu_{ra6} + \mu_{mb4} + \mu_{mb11} + \mu_{rb6} \leq 1$$

which prevent simultaneous preparation of the same substance in both mixers.

All constraints are grouped into the matrix and an inequality in the form (1) results. Here the marking vector has a dimension of 42×1, the matrix L is of dimension 5×42 and the incidence matrix D_p is of dimension 42×28. The supervisor is computed by equations (2) and (3). The supervisor consists of five places that are linked to the plant Petri net as shown by dotted arcs in Figure 9 (mixer models are not shown in full detail in this figure, places $p_{m(a,b)3'}$ and $p_{m(a,b)4'}$ are used to denote preparing of substances A and B, respectively). Place p_{c1} co-ordinates the heating-up phase of the two reactors, places p_{c2} and p_{c3} perform the resource allocation where the two shared resources correspond to the two mixing tanks and places p_{sc4} and p_{sc5} co-ordinate the batches to prevent deadlock of the system. The analysis of the resulting net shows that all places are safe and that the net is deadlock-free.

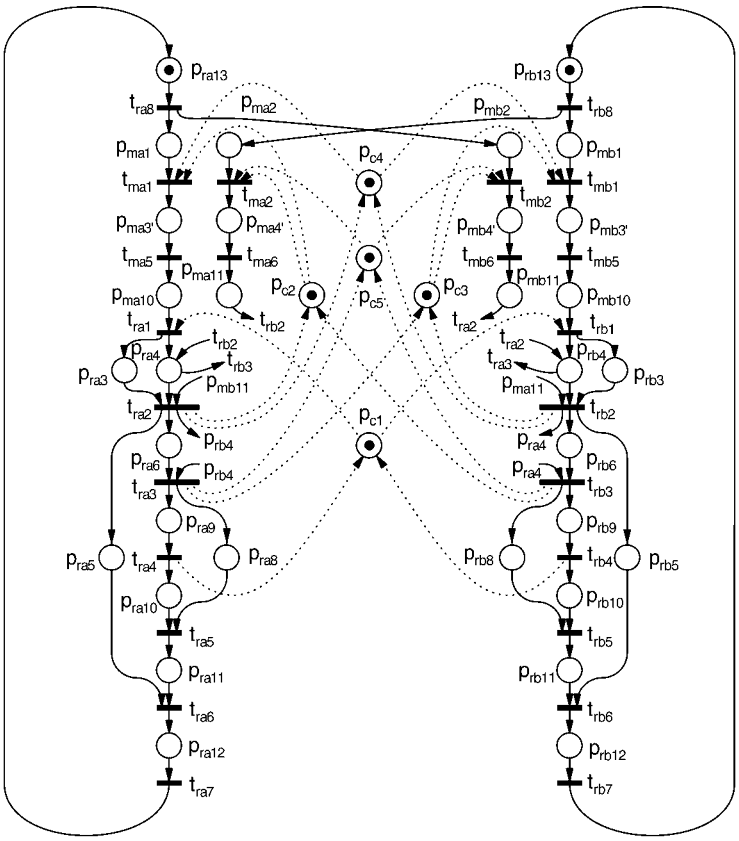


Fig. 9. Addition of the supervisor.

4.3 Simulation model

Because the supervisor has a discrete event view of the underlying process, no need for continuous modelling appeared in the case study so far. If the simulation of the designed system is required (e.g., for performance estimation, validation or training purposes), the time component has to be introduced. This can be achieved directly by estimating the duration of various operations or indirectly by modelling the continuous subprocesses and establishment of interaction of continuous and discrete event models. The combined continuous/ discrete event system structure usually better reflects the control system implementation. Upper, discrete event models have been derived in the previous subsection and the continuous part will be considered in the following.

The continuous model of the mixing tank describes the liquid level in the tank. The following two equations describe the filling and the discharge phase, respectively:

$$A_m \frac{dh_m(t)}{dt} = \phi_a(t) + \phi_b(t) + \phi_c(t) \tag{9}$$

$$A_m \frac{dh_m(t)}{dt} = -K_{mr}\sqrt{2gh_m(t)} \tag{10}$$

The continuous part of the single reactor model consists of a temperature controller and two sub-processes, describing the liquid level in the reactor (eq. (11) – filling phase, eq. (12) – discharging phase) and the temperature of the mixture in the reactor (eq. (13)).

$$A_r \frac{dh_r(t)}{dt} = K_{mr}\sqrt{2gh_m(t)} \tag{11}$$

$$A_r \frac{dh_r(t)}{dt} = -K_r\sqrt{2gh_r(t)} \tag{12}$$

In the given equations, $h(t)$ denotes the liquid level and A is the transverse section of the corresponding vessel. Subscript 'm' denotes the mixing tank and subscript 'r' the reactor vessel. K_{mr} and K_r are constants that include characteristics of the pipes and valves. ϕ_a, ϕ_b and ϕ_c are the volume flows at the corresponding inlets of the mixing tank and are assumed to be constant. With the assumption of ideal heat exchange between heating/cooling water and the liquid in the reactor, the temperature in the reactor follows the equation

$$\begin{aligned} m_{mix}c_{pmix}\frac{d\vartheta(t)}{dt} =& \rho_w\phi_d(t)c_{pw}(\vartheta_d - \vartheta(t)) - \rho_w\phi_e(t)c_{pw}(\vartheta(t) - \vartheta_e) \\ &- A_{rl}K_l(\vartheta(t) - \vartheta_0) \end{aligned} \tag{13}$$

where $\vartheta(t)$ 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 the specific heat of the water, $\phi_d(t)$, ϑ_d, $\phi_d(t)$, ϑ_d are the water flows and temperatures of the incoming water at the hot water inlet and the cold water inlet. The last term models the heat loss from the reactor contents to the environment, A_{rl} is the corresponding area, K_l is the heat transfer coefficient and ϑ_0 is the ambient temperature. The temperature controller is a simple PI controller, which is tuned to perform a response with no overshoot.

To bring the simulation model closer to industrial implementation the discrete model is transformed to SFCs. The single Petri net from Figure 9 is decomposed into several charts that correspond to separate logic controllers which control the individual units. The net decomposition is illustrated by Figure 10. The interaction among different SFC modules is performed through synchronised transitions. The synchronisation of transitions is easily achieved within the simulation model. Much more difficult is to achieve such a synchronisation among different logic controllers in practical implementation. Because of the communication delays it can not be guaranteed that transitions in different controllers fire simultaneously. This can be solved by defining the firing order of the transitions. Detailed elaboration of the corresponding implementation procedure can be found in [10].

The resulting model was simulated within the extended Matlab-Simulink simulation environment [11]. The corresponding simulation scheme is hierarchically decomposed into several subsystems following the implementation structure. By the simulation, 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.

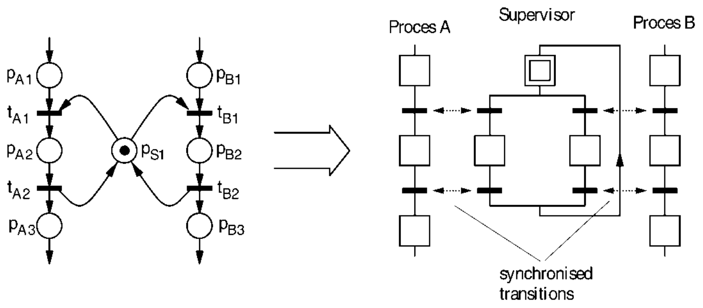


Fig. 10. Conversion of Petri net model to individual SFC modules.

5 CONCLUSIONS

It has been shown how the Petri net modelling framework can be used to model subprocesses in the distributed process control environment and how desired system specification can be met by the introduction of the co-ordinating supervisor. The case study of a batch process cell was presented and discrete event and continuous modelling frameworks were applied to the modelling and simulation of the given system. One of the main advantages of Petri nets is the straightforward path from developed controller models to the industrial implementation. The paper indicates how a Petri net model can be translated to sequential function chart, which can be simulated or directly implemented by a programmable logic controller. The simulation can be used to validate the functional correctness of the control system and to perform analysis of the system throughput in the presence of disturbances on the input material flows.

REFERENCES

1. David, R. and Alla, H.: Petri Nets for Modeling of Dynamic Systems — A Survey. *Automatica*, 30 (1994), pp. 175–202.
2. David, R.: Grafcet: A Powerful Tool for Specification of Logic Controllers. *IEEE Trans. on Control Systems Technology*, 3 (1995), pp. 253–268.
3. Giua, A. and DiCesare, F.: Generalized mutual exclusion constraints on nets with uncontrollable transitions. In: Proc. 1992 IEEE Int. Conf. on Systems, Man, and Cybernetics (Chicago, Illinois), pp. 974–979.
4. Hopcroft, J.E. and Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, 1979.
5. IEC, International Electrotechnical Commission, Programmable Controllers — Part 3: Programming Languages, publication 1131.3 (1992).
6. Jeng, M.D and DiCesare F.: A Review of Synthesis Techniques for Petri Nets with Applications to Automated Manufacturing Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 23 (1993), pp. 301–312.
7. Krogh, B.H. and Holloway, L.E.: Synthesis of feedback logic for discrete manufacturing systems. *Automatica*, 27 (1991), pp. 641–651.
8. Moody, J.O. and Antsaklis, P.J.: Supervisory Control of Petri Nets with Uncontrollable/Unobservable Transitions. Tech. Rep. ISIS-96-004, University of Notre Dame, 1996.
9. Murata, T.: Petri Nets: Properties, Analysis and Applications. *Proc. IEEE*, 77 (1989), pp. 541–580.
10. Mušič, G. and Matko, D.: Petri Net Based Supervisory Control of Flexible Batch Plants. In: Prepr. 8^{th} IFAC Symp. on Large Scale Systems: Theory & Application, Vol. II, Rio Patras, 1998, pp. 989–994.
11. Mušič, G. and Matko, D.: Combined Simulation for Process Control: Extension of a General Purpose Simulation Tool, *Computers in Industry*, 38 (1999), pp. 79–92.
12. Proth, J.M. and Xie, X.: Petri Nets, A Tool for Design and Management of Manufacturing Systems. Wiley (UK), Chichester, 1996.
13. Ramadge, P.J.G. and Wonham, W.M.: The Control of Discrete Event Systems. *Proc. IEEE*, 77 (1989), pp. 81–97.

14. Yamalidou, K., Moody, J., Lemmon, M., and Antsaklis, P.: Feedback Control of Petri Nets Based on Place Invariants. *Automatica*, 32 (1996), pp. 15–28.
15. Zhou, M.C. and DiCesare, F.: Parallel and Sequential Mutual Exclusions for Petri Net Modeling of Manufacturing Systems with Shared Resources. *IEEE Trans. on Robotics and Automation*, 7 (1991), pp. 515–527.
16. Zhou, M.C., DiCesare, F., and Desrochers, A.A.: A Hybrid Methodology for Synthesis of Petri Net Models for Manufacturing Systems. *IEEE Trans. on Robotics and Automation*, 8 (1992), pp. 350–361.
17. Zhou, M.C., DiCesare, F., and Rudolph, D.L.: Design and Implementation of a Petri Net Based Supervisor for a Flexible Manufacturing System. *Automatica*, 28 (1992), pp. 1199–1208.