# An Application of the DEDS Control Synthesis Method*

František Čapkovič

Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovak Republic
{Frantisek.Capkovic,utrrcapk}@savba.sk
http://www.ui.sav.sk/home/capkovic/capkhome.htm

**Abstract.** An application of the method suitable for modelling and control of general discrete event dynamic systems (DEDS) to special kinds of communication systems is presented in this paper. The approach is based on Petri nets (PN) defined in [12] and directed graphs (DG) described in [11]. It is supported by the previous author's works [1]-[10], [13].

## 1 Introduction

DEDS are the systems driven by discrete events. A sequence of discrete events can modify the DEDS behaviour. There are two kinds of discrete events - spontaneous events (peculiar to the system) and controllable ones (forced from without). Typical DEDS are flexible manufacturing systems, communication systems, transport systems. Processes in Web and/or multiagent systems are special kinds of communication systems. Thus, the modelling and control methods suitable for DEDS in general can be applied to modelling and control of them. We will use the analytical PN-based model of the DEDS dynamics development as follows

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}.\mathbf{u}_k \quad , \quad k = 0, N \tag{1}$$

$$\mathbf{B} = \mathbf{G}^T - \mathbf{F} \tag{2}$$

$$\mathbf{F}.\mathbf{u}_k \leq \mathbf{x}_k \tag{3}$$

where $k$ is the discrete step; $\mathbf{x}_k = (\sigma^k_{p_1}, ..., \sigma^k_{p_n})^T$ is the $n$-dimensional state vector of DEDS in the step $k$; $\sigma^k_{p_i} \in \{0, c_{p_i}\}$, $i = 1, ..., n$ express the states of the DEDS elementary subprocesses or operations - 0 (passivity) or $0 < \sigma_{p_i} \leq c_{p_i}$ (activity); $c_{p_i}$ is the capacity of the DEDS subprocess $p_i$ as to its activities; $\mathbf{u}_k = (\gamma^k_{t_1}, ..., \gamma^k_{t_m})^T$ is the $m$-dimensional control vector of the system in the step $k$; its components $\gamma^k_{t_j} \in \{0, 1\}$, $j = 1, ..., m$ represent occurring of the DEDS elementary discrete events (e.g. starting or ending the elementary subprocesses or their activities, failures, etc.) - i.e. the presence (1) or the absence (0) of the discrete event; $\mathbf{B}$, $\mathbf{F}$, $\mathbf{G}$ are constant matrices; $\mathbf{F} = \{f_{ij}\}_{n \times m}$, $f_{ij} \in \{0, M_{f_{ij}}\}$, $i = 1, ..., n$, $j = 1, ..., m$ expresses

* Partially supported by the Slovak Grant Agency for Science (VEGA) under grant # 2/3130/23.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 529–536, 2004.

the causal relations among the states of the DEDS (as causes) and the discrete events occuring during the DEDS operation (as consequences) - i.e. the nonexistence (0) or the existence and multiplicity ($M_{f_{ij}} > 0$) of the causal relations; $\mathbf{G} = \{g_{ij}\}_{m\times n}$, $g_{ij} \in \{0, M_{g_{ij}}\}$, $i = 1, ..., m$, $j = 1, ..., n$ expresses very analogically the causal relations among the discrete events (causes) and the DEDS states (consequences); $\mathbf{F}$ and $\mathbf{G}$ are the arcs incidence matrices and $\mathbf{B}$ is given by means of them according to (2); $(.)^T$ symbolizes the matrix or vector transposition.

Simultaneously, we will utilize the DG-based model in the form

$$\mathbf{X}(k+1) = \mathbf{\Delta}_k.\mathbf{X}(k) \quad , \quad k = 0, N \tag{4}$$

where $k$ is the discrete step; $\mathbf{X}(k) = (\sigma^{(k)}_{\pi_1}(\gamma), ..., \sigma^{(k)}_{\pi_{n_{RT}}}(\gamma))^T$, $k = 0, N$ is the $n_{RT}$-dimensional state vector of the DG in the step $k$; $\sigma^{(k)}_{\pi_i}(\gamma), \in \{0, 1\}$, $i = 1, n_{RT}$ is the state of the elementary DG node $\pi_i$ in the step $k$. Its value depends on actual enabling its input transitions. $\gamma$ symbolizes this dependency; $\mathbf{\Delta}_k = \mathbf{A}^T_{DG_f} = \{\delta^{(k)}_{ij}\}_{n_{RT}\times n_{RT}}$, $\delta^{(k)}_{ij} = \gamma^{(k)}_{t_{\pi_i|\pi_j}}$, $i = 1, n_{RT}$, $j = 1, n_{RT}$ is the functional matrix; $\gamma^{(k)}_{t_{\pi_i|\pi_j}} \in \{0, 1\}$ is the transition function of the PN transition fixed on the edge oriented from the DG node $\pi_j$ to the DG node $\pi_i$. It is necessary to say that the PN places $p_i$ are completely different form the DG nodes $\pi_i$. While $p_i$ represent the states of elementary activities inside PN, $\pi_i$ represent the complete state vectors of the PN. It corresponds with the fact that the DG (with the nodes $\pi_i$, $i = 1, ... n_{RT}$) is the RG of the PN (with the places $p_i$, $i = 1, ... n$) and represents the alternative artificial fictive state machine (SM) with the nodes $\pi_i$ , $i = 1, ... n_{RT}$ representing the reachable state vectors of the PN.

In [1], [2] the procedure enumerating the *quasi-functional* adjacency matrix $\mathbf{A}$ of the RG and the space of the PN reachable states in the form of the matrix $\mathbf{X}_{reach}$ was presented in a different depth. The columns of the matrix $\mathbf{X}_{reach}$ are the PN state vectors $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, ... reachable from the initial state $\mathbf{x}_0$. The inputs of the procedure are the PN structural matrices $\mathbf{F}$, $\mathbf{G}^T$ and the PN initial state vector $\mathbf{x}_0$. While the PN-based model in general (where any transition can have more than one input places as well as more than one output places) cannot be understood to be the classical SM (because of synchronization problems), the DG-based model (where DG is RG of the PN in question) is the classical SM.

To illustrate the operation of the above mentioned procedure let us model the client-server cooperation as simply as possible. There can be distinguished the following principal partial activities expressed by PN places: $p_1$ = the client requests for the connection, $p_2$ = the server is listening , $p_3$ = the connection of the client with the server, $p_4$ = data sent by the client to the server, $p_5$ = the disconnection of the client by the client himself. The PN representing the problem is given on the left side in Fig. 1. The PN transitions $t_1$ - $t_3$ represent discrete events that realize the system dynamics. The order of their occurrence influences the actual development of the system dynamics. The inputs $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{x}_0$ are the following as well as the outputs - i.e. the *quasi-functional* adjacency matrix $\mathbf{A}$ of the RG (given on the right in Fig. 1), the corresponding transpose of the functional matrix $\mathbf{\Delta}_k$, and the state space of its reachable states $\mathbf{X}_{reach}$.

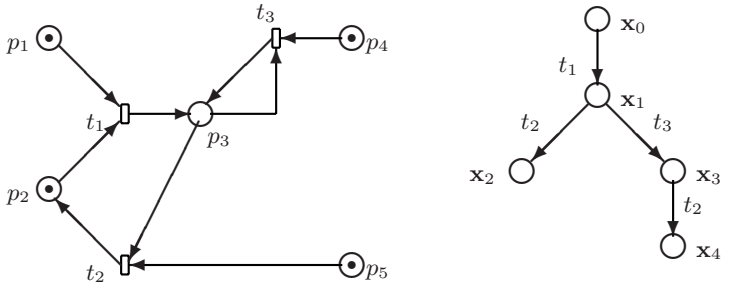


**Fig. 1.** The PN-based model of the client-server cooperation (on the left) and the corresponding reachability graph (on the right)

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \mathbf{x}_0 = (1,\ 1,\ 0,\ 1,\ 1)^T$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{\Delta}_k^T = \begin{pmatrix} 0 & t_1 & 0 & 0 & 0 \\ 0 & 0 & t_2 & t_3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{X}_{reach} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

As we can see, the nonzero elements of $\mathbf{A}$ represent the indices of the PN transitions. In this very simple example the control synthesis is very simple. After occurrence of the discrete event represented by the transition $t_1$ the client is connected with the server (the state $\mathbf{x}_1$). Now the client has two possibilities - to disconnect once more (by the event $t_2$ to the state $\mathbf{x}_2$) or to sent data to the server (by $t_3$ to $\mathbf{x}_3$). After sending data the client can work on the server and after finishing the work he can disconnect (by $t_2$ to $\mathbf{x}_4$). In more complicated cases (with more places and transitions and/or with more complicated structure of PN) it is necessary to perform the automatic control synthesis.

## 2 The Procedure of the Control Synthesis

The problem of control in general is the following: to transform the system to be controlled from a given initial state $\mathbf{x}_0$ to a prescribed terminal state $\mathbf{x}_t$ at simultaneous fulfilling the prescribed control task specifications (like criteria, constraints, etc.). For DEDS control synthesis the very simple idea can be utilized. Consider a system being in the initial state $\mathbf{x}_0$. Consider the desirable terminal state $\mathbf{x}_t$. Develop the straight-lined reachability tree (SLRT) from the state $\mathbf{x}_0$ towards $\mathbf{x}_t$ directed to $\mathbf{x}_t$. Develop the backtracking reachability tree (BTRT) from the state $\mathbf{x}_t$ towards $\mathbf{x}_0$ however, directed to $\mathbf{x}_t$. Intersect both the SLRT and the BTRT. The trajectory (or several ones) starting from $\mathbf{x}_0$ and

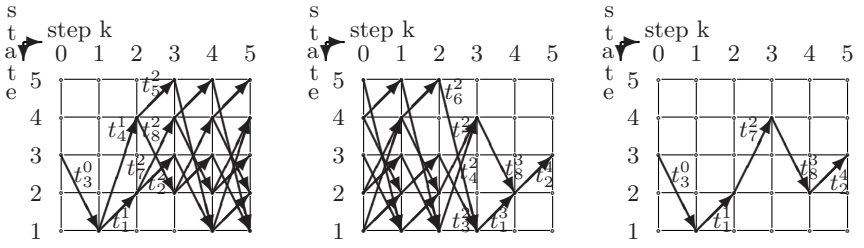


**Fig. 2.** The straight-lined system development from the initial state (on the left), the backtracking system development from the terminal state (in the centre), and the intersection (on the right)

finishing in $\mathbf{x}_t$ is (are) obtained. To illustrate such an approach see Fig. 2 (where the SLRT is given on the left, the BTRT in the centre, and their intersection on the right). Although in general there can be more than one trajectory after intersection, in any case we obtain the feasible possibilities of the system behaviour between the $\mathbf{x}_0$ and $\mathbf{x}_t$. When a criterion for optimality is given we can find even the optimal trajectory.

To avoid problems with symbolic operations at computer handling $\mathbf{\Delta}_k$ in (4) we will understand all of the transitions to be enabled (i.e. their transition functions having the values 1). In such a way we can replace the functional matrix $\mathbf{\Delta}_k$ by the constant matrix $\mathbf{\Delta}$. Thus, the DG-based approach operates with the constant matrix $\mathbf{\Delta}$ being the transpose of the constant adjacency matrix of the RG (representing the SM corresponding to the original PN). The constant RG adjacency matrix can be obtained from the *quasi-functional* adjacency matrix $\mathbf{A}$ by means of the replacement all of the nonzero elements by the integer 1. Hence, the SLBT can be constructed in analytical terms as follows

$$\{\mathbf{X}_1\} = \mathbf{\Delta}.\mathbf{X}_0;\ \{\mathbf{X}_2\} = \mathbf{\Delta}.\{\mathbf{X}_1\} = \mathbf{\Delta}^2.\mathbf{X}_0;\ \ldots;\ \{\mathbf{X}_N\} = \mathbf{\Delta}.\{\mathbf{X}_{N-1}\} = \mathbf{\Delta}^N.\mathbf{X}_0$$

where $\mathbf{X}_N = \mathbf{X}_t$. In general, $\{\mathbf{X}_j\}$ is an aggregate all of the states that are reachable from the previous states. According to graph theory $N \leq (n_{RT} - 1)$. The BTRT is developed from the $\mathbf{X}_t$ towards $\mathbf{X}_0$, however, it contains the paths oriented towards the terminal state. It is the following

$$\{\mathbf{X}_{N-1}\} = \mathbf{\Delta}^T.\mathbf{X}_N;\ \{\mathbf{X}_{N-2}\} = (\mathbf{\Delta}^T)^2.\mathbf{X}_N;\ \ldots;\ \{\mathbf{X}_0\} = (\mathbf{\Delta}^T)^N.\mathbf{X}_N$$

Here, $\{\mathbf{X}_j\}$ is an aggregate all of the states from which the next states are reachable. It is clear that $\mathbf{X}_0 \neq \{\mathbf{X}_0\}$ and $\mathbf{X}_N \neq \{\mathbf{X}_N\}$. It is the consequence of the fact that in general $\mathbf{\Delta}.\mathbf{\Delta}^T \neq \mathbf{I_n}$ as well as $\mathbf{\Delta}^T.\mathbf{\Delta} \neq \mathbf{I}_n$ (where $\mathbf{I}_n$ is $(n \times n)$ identity matrix). The intersection of the trees is made as follows

$$\mathbf{M}_1 = (\mathbf{X}_0, {}^1\{\mathbf{X}_1\}, \ldots, {}^1\{\mathbf{X}_{N-1}\}, {}^1\{\mathbf{X}_N\});\ \mathbf{M} = \mathbf{M}_1\ \cap\ \mathbf{M}_2$$
$$\mathbf{M}_2 = ({}^2\{\mathbf{X}_0\}, {}^2\{\mathbf{X}_1\}, \ldots, {}^2\{\mathbf{X}_{N-1}\}, \mathbf{X}_N);\ \mathbf{M} = (\mathbf{X}_0, \{\mathbf{X}_1\}, \ldots, \{\mathbf{X}_{N-1}\}, \mathbf{X}_N)$$

where the matrices $\mathbf{M}_1$, $\mathbf{M}_2$ represent, respectively, the SLRT and the BTRT. The special intersection both of the trees is performed by means of the column-to-

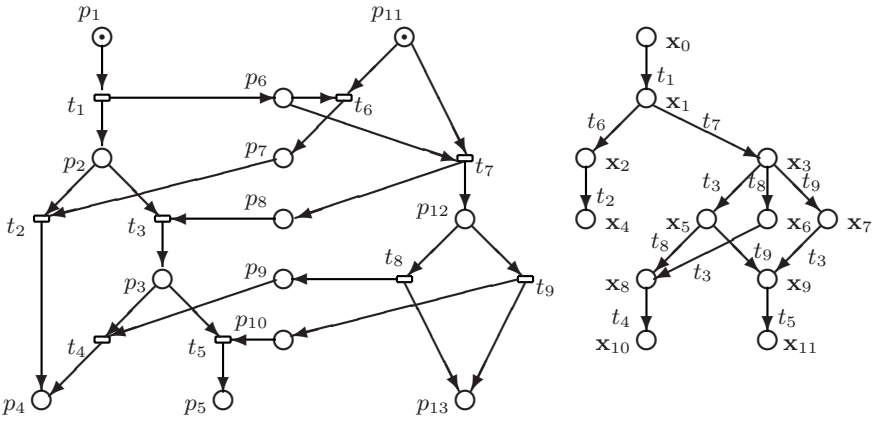


**Fig. 3.** The PN-based model of two agents cooperation (on the left) and the corresponding reachability graph (on the right)

column intersection both of the matrices. Thus, $\{\mathbf{X}_i\} = \min({}^1\{\mathbf{X}_i\}, {}^2\{\mathbf{X}_i\})$, $i = 0, ..., N$ with ${}^1\{\mathbf{X}_0\} = \mathbf{X}_0$, ${}^2\{\mathbf{X}_N\} = \mathbf{X}_N$.

To illustrate the approach let us model a simple cooperation of two agents $A$ and $B$ by PN. $A$ needs to do an activity $P$, however, it is not able to do this. Therefore, $A$ requests $B$ to do $P$ for him. On the base of the conversation between the agents $P$ is either done (if $B$ is willing to do $P$ and it is able to do $P$) or not (when $B$ refuses to do $P$ or it is willing to do $P$, however, it is not able to do $P$). The PN places express the activities of the agents and the messages being routed between them: $p_1$ = the agent $A$ wants to do $P$, however, he is not able to do $P$, $p_2 = A$ waits for an answer from $B$, $p_3 = A$ waits for a help from $B$, $p_4$ = the failure of the cooperation, $p_5$ = the satisfaction of the cooperation, $p_6 = A$ requests $B$ to do $P$, $p_7 = B$ refuses to do $P$, $p_8 = B$ accepts the request of $A$ to do $P$, $p_9 = B$ is not able to do $P$, $p_{10}$ = doing $P$ by $B$, $p_{11} = B$ receives the request of $A$, $p_{12} = B$ is willing to do $P$, $p_{13}$ = the end of the work of $B$. The transitions correspond either to synchronization due to the receipt of a message or to conditions of the application of actions. The PN-based model is given on the left in Fig. 3 while the corresponding RG is given on the right. When the initial state vector $\mathbf{x}_0 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)^T$ and the structural matrices are

$$\mathbf{F}^T = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 0&1&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&0&1 \end{pmatrix}$$

the output matrices $\mathbf{A}$, $\mathbf{X}_{reach}$, and $\mathbf{M}$ are, respectively, the following

$$\begin{pmatrix}
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&6&7&0&0&0&0&0&0&0&0&0\\
0&0&0&0&2&0&0&0&0&0&0&0&0\\
0&0&0&0&0&3&8&9&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&8&9&0&0&0\\
0&0&0&0&0&0&0&0&0&3&0&0&0\\
0&0&0&0&0&0&0&0&0&0&3&0&0\\
0&0&0&0&0&0&0&0&0&0&0&4&0\\
0&0&0&0&0&0&0&0&0&0&0&0&5\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{pmatrix};
\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&1&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&1&1&1&1
\end{pmatrix};
\begin{pmatrix}
1&0&0&0&0&0\\
0&1&0&0&0&0\\
0&0&0&0&0&0\\
0&0&1&0&0&0\\
0&0&0&0&0&0\\
0&0&0&1&0&0\\
0&0&0&0&0&0\\
0&0&0&1&0&0\\
0&0&0&0&0&0\\
0&0&0&0&1&0\\
0&0&0&0&0&0\\
0&0&0&0&0&1
\end{pmatrix}$$

The approach can be tested very quickly in Matlab. The trajectory given in the matrix $\mathbf{M}$ represents the situation when the desired terminal state is the successful cooperation $\mathbf{x}_{11} = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1)^T$. It is graphically demonstrated on the left in Fig. 4. When the terminal state represents the failure of the cooperation $\mathbf{x}_{10} = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1)^T$ (when $B$ is not able to do $P$) the result is different - see the trajectory on the right in Fig. 4.

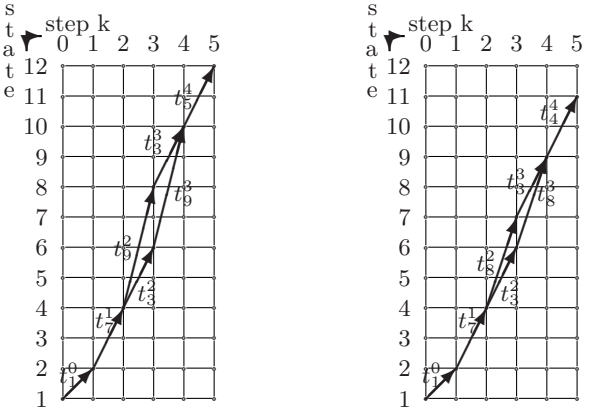


**Fig. 4.** The resulting trajectories - in case of the successful cooperation (on the left) and in case of the failure of the cooperation when $B$ is not able to do $P$ (on the right)

## 3 The Adaptivity

As to the applicability of the above described approach we can distinguish two kinds of adaptivity. On the one hand it is the adaptivity concerning the modifying of the system dynamics development by means of choosing a suitable trajectory

from the set of feasible state trajectories obtained in the control synthesis process. Such a kind of the adaptivity can be clear e.g. from the left picture in Fig. 4 where two different feasible trajectories (expressing possibilities of the system behaviour) are presented. Because no other trajectory exists, either the trajectory corresponding to the sequence of enabled transitions $\{t_1^0, t_7^1, t_9^2, t_3^3, t_5^4\}$ or the trajectory corresponding to the sequence $\{t_1^0, t_7^1, t_3^2, t_9^3, t_5^4\}$ can be chosen in order to adapt the system behaviour to the actual demand. On the other hand it is the adaptivity of the system behaviour by means of a structural fragment added to the original system structure. Such a fragment is able to accept demands (given from without) on the system behaviour and realize them. The adaptivity is illustrated in Fig. 5. On the left the system model consisting of two processes and the structural fragment containing the place $p_4$ is given. The fragment is able to influence the performance of the processes (the mutual exclusion, sequencing, re-run). In the centre of Fig. 5 the RG is presented while the result of the control synthesis process transforming the system from $\mathbf{x}_0 = (1, 0, 0, 1, 1, 0, 0)^T$ to $\mathbf{x}_6 = (0, 0, 1, 0, 0, 1, 0)^T$ (when $p_3$ has priority ahead of $p_7$) is put on the right.

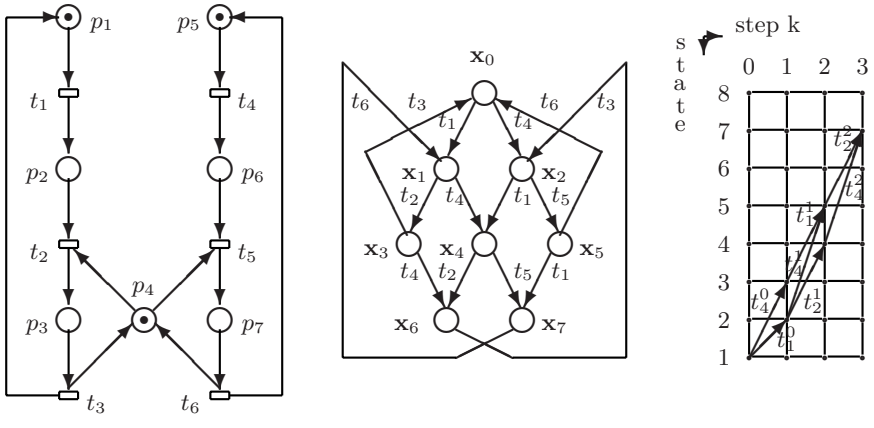


**Fig. 5.** The PN-based model of the system behaviour (on the left), its reachability graph (in the centre), and the feasible trajectories from $\mathbf{x}_0$ to $\mathbf{x}_6$ (on the right)

## 4 Conclusions

The approach to the modelling and control synthesis of DEDS was presented in this paper. Its applicability to the special communication systems was demonstrated. The approach is suitable for DEDS described by PN with the finite state space. In order to automate the control synthesis process the graphical tool GraSim was developed. The input of the tool is represented by the RG created by means of icons. On its output the tool yields (in the graphical form) the system trajectories from a given initial state to a prescribed terminal one.

## References


1. Čapkovič, F.: The Generalised Method for Solving Problems of DEDS Control Synthesis. In: Chung, P.W.H., Hinde, C., Ali, M. (eds.): Developments in Applied Artificial Intelligence. Lecture Notes in Artificial Intelligence, Vol. 2718. Springer-Verlag, Berlin Heidelberg New York (2003) 702–711
2. Čapkovič, F., Čapkovič, P.: Petri Net-based Automated Control Synthesis for a Class of DEDS. In: ETFA 2003. Proceedings of the 2003 IEEE Conference on Emerging Technologies and Factory Automation. IEEE Press, Piscataway, NJ, USA (2003) 297–304
3. Čapkovič, F.: Control Synthesis of a Class of DEDS. Kybernetes. The International Journal of Systems and Cybernetics **31** No. 9/10 (2002) 1274–1281
4. Čapkovič, F.: A Solution of DEDS Control Synthesis Problems. Systems Analysis Modelling Simulation **42** No. 3 (2002) 405–414
5. Čapkovič, F., Čapkovič, P.: Intelligent Control Synthesis of Manufacturing Systems. In: Monostori, L., Vancza, J., Ali, M. (eds.): Engineering of Intelligent Systems. Lecture Notes in Computer Sciences, Vol. 2070. Springer-Verlag, Berlin Heidelberg New York (2001) 767–776
6. Čapkovič, F.: Intelligent Control of Discrete Event Dynamic Systems. In: Koussoulas, N.T., Groumpos, P.P., Polycarpou, M. (eds.): Proc. of IEEE International Symposium on Intelligent Control. IEEE Press, Patras, Greece (2000) 109–114
7. Čapkovič, F.: Modelling and Control of Discrete Event Dynamic Systems. BRICS Report Series, RS-00-26. University of Aarhus, Denmark (2000) 58 p.
8. Čapkovič, F.: A Solution of DEDS Control Synthesis Problems. In: Kozák, Š., Huba, M. (eds.): Control System Design. Proceedings of IFAC Conference. Pergamon, Elsevier Science. Oxford, UK (2000) 343–348
9. Čapkovič, F.: Automated Solving of DEDS Control Problems. In: El-Dessouki, A., Imam, I., Kodratoff, Y., Ali, M. (eds.): Multiple Approaches to Intelligent Systems. Lecture Notes in Computer Science, Vol. 1611. Springer-Verlag, Berlin Heidelberg New York (1999) 735–746
10. Čapkovič, F.: Knowledge-Based Control Synthesis of Discrete Event Dynamic Systems. In: Tzafestas, S.G. (ed.): Advances in Manugacturing. Decision, Control and Information Technology. Chapter 19. Springer-Verlag, London (1998) 195–206
11. Diestel, R.: Graph Theory. Springer-Verlag, New York (1997)
12. Peterson, J.L.: Petri Net Theory and Modeling the Systems. Prentice Hall, New York (1981)
13. Tzafestas, S.G., Čapkovič, F.: Petri Net-Based Approach to Synthesis of Intelligent Control for DEDS. In: Tzafestas, S.G. (ed.): Computer Assisted Management and Control of Manufacturing Systems, Chapter 12. Springer-Verlag, Berlin Heidelberg New York (1997) 325–351