Appl. Math. and Comp. Sci., 1993, vol.3, No.4, 751–776

ON SELF-SYNCHRONISATION OF CASCADE-LIKE COUPLED CYCLIC PROCESSES†

ZBIGNIEW BANASZAK*, KRZYSZTOF JEDRZEJEK**

In this paper a unified framework is introduced for the study as well as for analysing the performance of concurrently interacting cyclic processes. Conditions that guarantee the periodicity of cascade–like interacting processes' behavior are provided. Formulae determining the parameters of the steady–state cyclic behaviors are proposed and are regarded as formal models of a self–synchronisation phenomenon. The parameters change and priority rules selection are examined. Procedures aimed at the control and performance evaluation are suggested.

1. Introduction

Modelling and performance evaluation problems for distributed control systems have recently been attracting considerable attention in the computer science community. As a result, there have been proposed specific discrete–event systems based methods of solving the problems in order to fulfil some extra requirements such as deadlock avoidance, priority rules' selection, job sequencing and resources allocation, structural and functional redundancy of systems design (Dembele *et al.*, 1992; Ramadge and Wonham, 1989; Righter and Walrand, 1989; Viswandham and Narahari, 1992).

A study of discrete event systems (DEVS) has been carried out directed forward to establish a mathematical description of performance evaluation oriented models (Scheuring and Wehlan, 1991; Zeigler, 1989). However, the models actually available are analytically intractable and computationally depended on the combinatorial explosion of the searched states amount. As a consequence, almost all the proposed computational or performance–oriented approaches to DEVSs incorporate stochastic effects as integral parts of their modelling, and the performance measures are mostly formulated in terms of continuous variables, such as average throughput, waiting time, profit, and inventory (Cao, 1989; Ho, 1989). On the other hand, there has been a growing interest in development of a conceptual–like sort of models aimed at examination of control theoretical ideas such as controllability, observability, and realizability (Brave and Heymann, 1990; Caspi, 1991).

The methods employed to deal with the modelling and performance evaluation problems come from Markov chains/automaton (including Petri nets) and generalised semi–Markov models as well as mini–max algebraic models and communicating sequential

† This work was supported by the Kuwait University Research Project No. SM 085 during the Authors' stay at Kuwait University, Dept. of Mathematics, P.O. Box 5969, 13060 Safat, Kuwait

* Technical University of Zielona Góra, Dept. of Robotics and Software Engineering

** Dept. of Mathematics, Kuwait University, P.O. Box 5969, 13060 Safat, Kuwait

processes framework (Braker, 1991; Cohen *et al.*, 1989; Hillion and Proth, 1989; Laftit *et al.*, 1992; Ramadge, 1990). Application of such methods to design and control problems has proved to be successful in a variety of systems ranging from the flexible manufacturing systems, through operating systems, to the computer/communication networks (Carpenter, 1987; Viswandham and Narahari, 1992; Willner and Heymann, 1991). These methods, however, in many cases become inadequate because of unrealistic assumptions and rough approximations they apply.

The shortcomings may be overcome by methods providing more general framework allowing both to encompass behaviors of continuous as well as discrete–time systems, and easily determine quantitative as well as qualitative properties of DEVS performance. Such a new perspective distinguishes itself by stating and examining performance evaluation problems in the context of more general problems and of structures of Loosely Coupled Systems (LCS) (Wedde, 1983; Weick, 1987). The approach should not only provide the DEVS theory with new results, but also may unify many specific results and offer a deeper insight into long–standing problems.

This paper may be viewed as a contribution to the problem of modelling and performance evaluation of LCSs that are composed of a set of cyclic processes interacting via common, shared and reusable resources. In contrast to the way the problem has been treated in DEVS theory until recently, this approach focuses on a self–synchronisation concept viewed as a relationship linking some structural and functional properties of concurrently interacting cyclic processes. Its main advantage lies in treating a periodicity of the steady–state behavior as a key factor of a performance measure. In other words, providing the self–synchronisation conditions which make it possible to determine a steady–state performance of interacting processes results in determination of dependencies linking the so called global system performance with the local, component process performances, e.g. cycle times.

The processes considered are typical for many man–made dynamic systems such as production or assembly lines, distributed computer networks, and traffic systems. Their evolution in time depends on the complex interactions of the timing of various discrete events, such as the arrival or departure of a job or the initiation and completion of a task or message. A system composed of a set of reusable resources such that each resource may be replaced with some other ones, usually less efficient (Banaszak *et al.*, 1992; Banaszak and Chudy, 1992), could be considered as the relevant example.

It is also assumed that a whole system behavior (its performance measure) is implied from its component subsystems, i.e. processes employing different subsets of available resources. As a consequence, because processes interact with each other via the commonly shared resources, the system performance depends on both the resources allocation and processes interaction. Then, assuming that a number of items produced per a time unit states a system performance measure, if a cycle is determined by a time elapsing between the delivery moments of the two subsequent items, the system efficiency depends on periodicity of component processes, their initial state, and the conflict–dispatching rules applied.

This approach allows us to consider different sorts of complex dynamic systems on the same modelling basis of concurrently interacting cyclic processes. The main question concerns the identification of analytical dependencies existing between structural and

behavioral aspects of systems design and operation, i.e. conditions allowing to determine the system performance according to its component characteristics and component interactions.

The proposed framework makes it also possible to consider the cascade–like interacting processes as a class of LCSs that are characterised by a connotation of impermanent and changing relationship. The LCSs are composed of a set of separate and in some sense autonomous components which retain their identity, but which are linked in a more or less informal manner. Their important characteristics include decentralisation and relative lack of a co–ordinator. Within this framework the analytical models relating the system dynamics to behaviors of component processes are mainly investigated. Some suggestions concerning the model implementation to the design and control problems are also proposed.

The paper is organised as follows. In section 2 we introduce illustrative examples of cyclic processes specification and concurrent processes interaction, and then we provide the model description and define related concepts such as a critical resource, critical processes, and conflict states. Section 3 is devoted to presentation of the main results, and formulates the problem and includes theorems providing self–synchronisation conditions for cascade–like interacting cyclic processes. Section 4 shows how some procedures aimed at performance evaluation of cyclic processes could be derived from obtained conditions. Section 5 concludes the paper.

2. Concurrently Interacting Cyclic Processes

Processes specification is introduced and a model description is proposed which provide the unified framework for the development of the self–synchronisation conditions.

2.1. Processes Specification

In order to introduce parameters that specify concurrent, cascade–like interacting processes let us consider a flexible Automated Measuring and Quality Testing System (MQTS) as an illustrative example. As shown in Figure 1, this system consists of the following resources: four inspection machines M_1, M_2, M_3, and M_4, three industrial robots R_1, R_2, and R_3, three input buffers B_{11}, B_{21} and B_{31}, and three output buffers B_{12}, B_{22}, and B_{32}.

Three product types are inspected in the system. Each of them needs two machine operations in order to complete the relevant inspection route. The inspection steps are described as follows: For products of type 1 (type 2, and type 3) robot R_1 (R_2, R_3) takes an item from B_{11} (B_{21}, B_{31}), approaches M_2 (M_4, M_3), and then performs the first testing operation if M_2 (M_4, M_3) is available, i.e. if the machine is not busy inspecting other products. Next, robot R_1 (R_2, R_3) moves the partially examined product from M_2 (M_4, M_3) and approaches M_1 (M_1, M_4) if it is available. Then, robot R_1 (R_2, R_3) moves the tested part from M_1 (M_1, M_4) and places it in B_{12} (B_{22}, B_{32}). Because robot R_1 (R_2, R_3) handles a product of type 1 (type 2, type 3) through all steps of the inspection route, only one item of type 1 (type 2, type 3) can be simultaneously examined. As a result, no more than three products of different types are concurrently tested.

There are four assumptions in the system:

(i) products of type 1 (type 2, type 3) are always available in B_{11} (B_{21}, B_{31}),

(ii) products of type 1 (type 2, type 3) in B_{12} (B_{22}, B_{32}) are carried away so that B_{12} (B_{22}, B_{32}) cannot be overflowed,

(iii) any machine of MQTS that does not inspect a part is assumed to be available,

(iv) any robot that does not handle a product is assumed to be available, and is ordered to take an available item from the relevant input buffer.

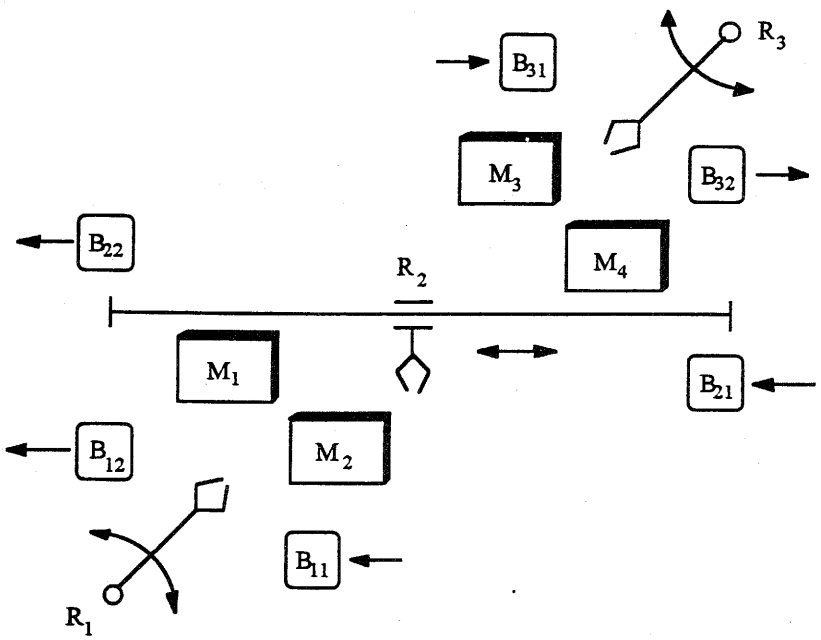

Fig. 1. A flexible automated measuring and quality testing system.

It may be easily noted that only one operation can be simultaneously executed on one resource, i.e. the robot or the measuring machine.

The rules provided determine the execution of processes specified by the following inspection routes:

$$
\begin{aligned}
IR_1 = {} & (O_{1,1}; R_1, B_{11}), (O_{1,2}; R_1), (O_{1,3}; R_1, M_2), (O_{1,4}; R_1), (O_{1,5}; R_1, M_1), \\
& (O_{1,6}; R_1), (O_{1,7}; R_1, B_{12}), (O_{1,8}; R_1), \\
IR_2 = {} & (O_{2,1}; R_2, B_{21}), (O_{2,2}; R_2), (O_{2,3}; R_2, M_4), (O_{2,4}; R_2), (O_{2,5}; R_2, M_1), \\
& (O_{2,6}; R_2), (O_{2,7}; R_2, B_{22}), (O_{2,8}; R_2), \\
IR_3 = {} & (O_{3,1}; R_3, B_{31}), (O_{3,2}; R_3), (O_{3,3}; R_3, M_3), (O_{3,4}; R_3), (O_{3,5}; R_3, M_4), \\
& (O_{3,6}; R_3), (O_{3,7}; R_3, B_{32}), (O_{3,8}; R_3),
\end{aligned}
\tag{1}
$$

where $(O_{i,j}; R_i, B_k)$ is the j–th operation performed by the i–th robot taking (or placing) an item of the i–th sort product type from (in) the k–th buffer; $(O_{i,j}; R_i)$

– the j–th operation performed by the i–th robot: (i) carrying an item of the i–th product type between buffers and machines or between machines and buffers; (ii) moving its manipulator between output buffers B_{i1} to input buffers B_{i2}; $(O_{i,j}; R_i, M_k)$ – the j–th operation performed by the i–th robot and the k–th inspection machine.

Assumptions imposed encompass a quite realistic situation typical for the real–life Flexible Manufacturing Systems (Laftit *et al.*, 1992; Viswandham and Narahari, 1992). In the case under consideration some of MQTS resources, i.e. M_1 and M_4, are shared among the concurrently executed processes of the products testing. The processes competition leads to the occurrence of delays during which processes wait for the access to the commonly shared resources. This fact results in a process efficiency decrease. The processes efficiency η will be used as a performance measure and it can be determined by the following formula

$$\eta = 1 - \frac{\Delta}{nT} \tag{2}$$

where n is the number of all component processes, T – a cycle time of the interacting processes system, $\Delta = \sum_{i=1}^{n}(T - n_i T_i)$, T_i – a cycle time of the i–th component process, n_i – a number of times the i–th process occurs during the period T.

Efficiency close to unity means that the system resources are used effectively, low efficiency means that they are wasted. In other words the efficiency allows one to know how close he or she is getting to the best the system can do. Of course, in the case of elementary component processes (where no delays occur) the efficiency is equal to 1.

In order to analyse the behavior of the MQTS considered let us assume that each process specification is reduced to two stages only. Each stage contains preceding and succeeding auxiliary operations as well as the main inspection operations performed on machines M_1 – M_4, i.e. the only systems shared resources. Consequently, the time of the first stage is equal to a sum of the first handling operation, and the first transportation as well as the first inspection and the second transportation operation. In turn, the second stage time is determined by a sum of the second inspection, the third transportation, the second handling, and the input buffers approaching operations. In other words, each process can be considered as a process consisting of two stages that follow each other alternately. The graphical representation of the model considered is shown in Figure 2.

Before introducing the concept of the state space based processes representation let us focus for a moment on the periodicity of a class of elementary processes under consideration. According to rules (i) – (iv) each inspection process can be treated as cyclic, i.e. such that the following condition holds

$$\exists m, T \in N_o,\ \forall\, t > m : \boldsymbol{V}(t) = \boldsymbol{V}(t+T) \tag{3}$$

where $\boldsymbol{V}(t) = \{\mathrm{crd}_i \mathcal{V}(t) : i \in \{1, 2, ..., n+1\}\}$ is a vector $\mathcal{V}(t)$ restricted to its first $n+1$ parameters; $\mathcal{V}(t)$ – the process value at the t–th instant of time, i.e. $\mathcal{V}(t) = (v_1, v_2, ..., v_{n+1}, t)$, v_i – value of the i–th process parameter; T – the cycle time; $N_0 = N \cup \{0\}$, N – the set of natural numbers, $\mathrm{crd}_i A = a_i$ for $A = (a_1, a_2, ..., a_i, ..., a_n)$. Note that introduced assumption $t \in N$ results in treating time as a discrete variable whose value can be expressed as an integer. Such interpretation of the time nature is

very often applied in real–world problems. As a consequence, the current process value $V(t)$ should be distinguished from the relevant state concept. The following expression determines the relationship between both concepts

$$S_i \overset{\text{def}}{\Longleftrightarrow} \boldsymbol{V}(t) \quad \text{and} \quad t \in \tau_i \tag{4}$$

where $\tau_i = \bigcup\{\tau \subset N : \tau = \{k, k+1, ..., k+r\},\ k, r \in N\}$ is a union of a family of sets of times such that $\boldsymbol{V}(t) = \boldsymbol{V}(t')$, for $t, t' \in \tau_i$; S_i – the i–th state of the process.

The same representation of cyclic behavior can be applied in the case of modelling whole systems of cascade–like coupled elementary processes. In order to illustrate the behavior of concurrently interacting processes, let us return to the example of the system of interacting, simplified (i.e. reduced to two complex operations) processes shown in Figure 2.

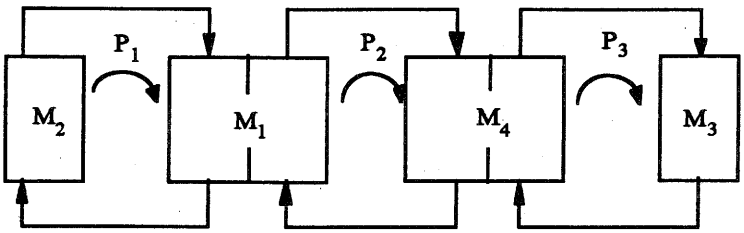

Fig. 2. Graphical representation of cascade like interacting cyclic processes.

Consider Gantt's charts (presented in Fig. 3) which are obtained for the following operation times

$$\begin{aligned} &T(\mathrm{P}_1, \mathrm{R}_1) = 2; \qquad T(\mathrm{P}_1, \mathrm{R}_2) = 1; \qquad T(\mathrm{P}_2, \mathrm{R}_2) = 1; \\ &T(\mathrm{P}_2, \mathrm{R}_3) = 4; \qquad T(\mathrm{P}_3, \mathrm{R}_3) = 1; \qquad T(\mathrm{P}_3, \mathrm{R}_4) = 3. \end{aligned} \tag{5}$$

where $T(\mathrm{P}_i, \mathrm{R}_j)$ is the operation time required for execution of the j–th stage in the i–th elementary process, e.g. $T(\mathrm{P}_1, \mathrm{R}_1)$ is the operation time required by the operations corresponding to resource R_1 standing for M_1.

Since both "process like" and "resource like" representations are equivalent to each other, thus they can be used alternately. In each particular case their use depends on the application context considered. The state representation may be directly derived from a given version of the Gantt's chart.

In the case of diagrams shown in Figure 3a and Figure 3c the following three–tuple provides a relevant description state $\mathcal{S}$

$$\begin{array}{ll} \textit{processes} & \overbrace{\quad}^{\mathrm{P}_1} \quad \overbrace{\quad}^{\mathrm{P}_2} \quad \overbrace{\quad}^{\mathrm{P}_3} \\ \textit{state} & \mathcal{S} = (\ s_1\ ,\ s_2\ ,\ s_3\ )\ , \\ \textit{resources} & \underbrace{\quad}_{\mathrm{R}_1} \quad \underbrace{\quad}_{\mathrm{R}_2} \quad \underbrace{\quad}_{\mathrm{R}_3} \quad \underbrace{\quad}_{\mathrm{R}_4} \end{array} \tag{6}$$

where

$$s_i = \begin{cases} i & \text{if process } P_i \text{ is executed on } R_i \\ i+1 & \text{if process } P_i \text{ is executed on } R_{i+1} \\ * & \text{if either process } P_i \text{ is suspended on } R_i \text{ and waits for } R_{i+1} \\ & \text{release or process } P_{i-1} \text{ is suspended on } R_i \text{ and waits} \\ & \text{for } R_{i-1} \text{ release} \end{cases} \quad (7)$$

However, according to the diagrams shown in Figure 3b and Figure 3d, the relevant state can be determined as the following four–tuple

$$\begin{array}{ll} \textit{resources} & \overbrace{R_1}\ \overbrace{R_2}\ \overbrace{R_3}\ \overbrace{R_4} \\ \textit{state} & S = (\ \underbrace{s_1\ s_2\ s_3\ s_4}\ ) \\ \textit{processes} & \quad P_1\ P_2\ P_3 \end{array} , \quad (8)$$

where

$$s_i = \begin{cases} 1 & \text{if process } P_{i-1} \text{ is executed on } R_i \text{ or process } P_i \text{ is executed on } R_i \\ 0 & \text{if process } P_i \text{ is executed on } R_{i+1} \text{ and process } P_{i-1} \text{ is executed} \\ & \text{on } R_{i-1} \\ * & \text{if either process } P_i \text{ is suspended on } R_i \text{ and waits for } R_{i+1} \\ & \text{release or process } P_{i-1} \text{ is suspended on } R_i \text{ and waits} \\ & \text{for } R_{i-1} \text{ release} \end{cases} \quad (9)$$

Note, that last state representation has a very simple and clear form. Moreover, it can be easily noted that both forms, i.e. (6) and (8) are equivalent to each other, both of them can be deduced from each other. For the sake of simplicity, whenever it will not lead to confusion, the former state notation S will be used.

According to rules (i) – (iv) a behavior of cascade–like coupled elementary (i.e. having simplified form) cyclic processes are determined by the following rules:

- each elementary cyclic process P_i can be treated as a sequence of alternately repeating operations executed on the two associated resources R_i, and R_{i+1}, respectively;
- on each system resource only one operation can be executed simultaneously;
- coupled processes that compete with access to the system shared resources can be either executed or suspended on one of the relevant (associated to them) resources;
- a process can be suspended on a given resource only if the other one is busy with its neighbor competing processes.

(10)

Thus, each state consists of unique $i \in \{1,2,3,4\}$ such that $s_i = 0$. Its structure easily expresses the encoded situations, e.g. the following state (1, *, 1, 0) (in order to simplify notation the following representation (1*10) will be further used) means that

processes P_1 and P_3 are executed on R_1 and R_3, respectively, while process P_2 being suspended on R_2 awaits for R_3 release.

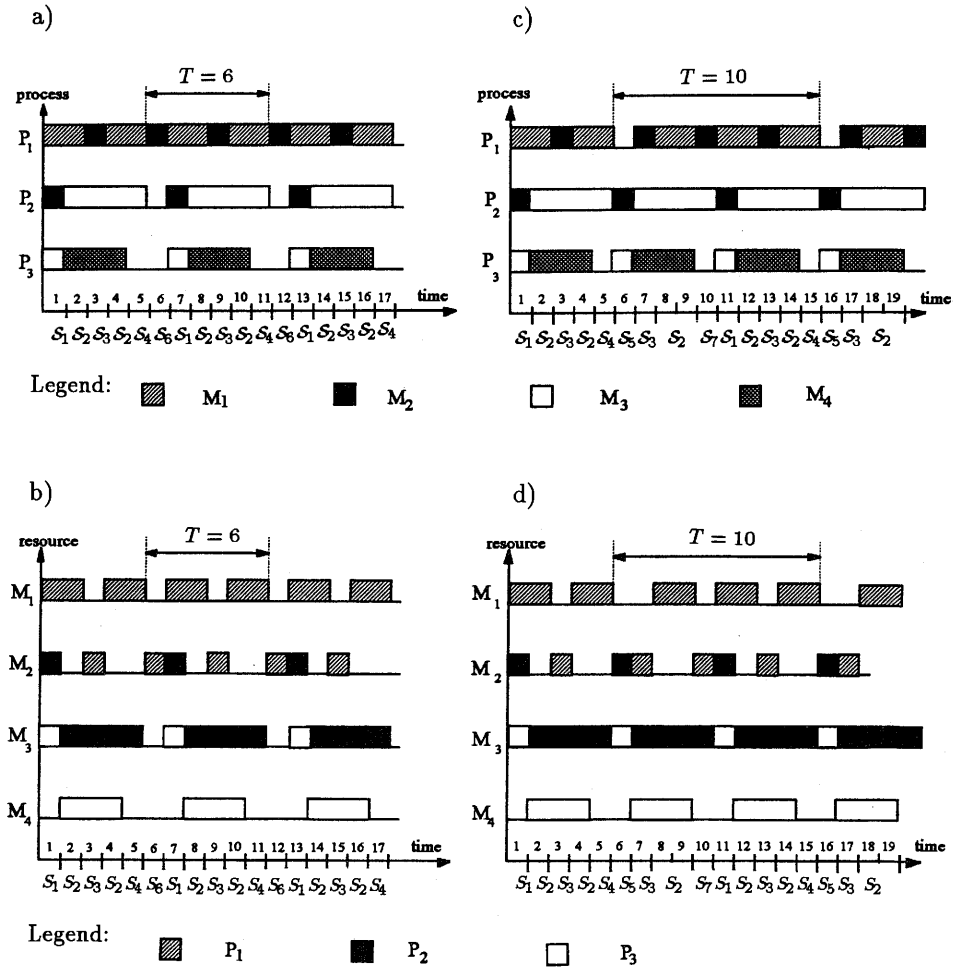

Fig. 3. The Gantt's chart of a process flow:
a) the "process version" diagram for dispatching rule #1;
b) the "resource version" diagram for dispatching rule #1;
c) the "process version" diagram for dispatching rule #2;
d) the "resource version" diagram for dispatching rule #2.

Note that according to the above assumed rules (10) the states such as (0***) or (1111) cannot occur. The reachability graph encompassing the case considered is shown in Figure 4. Note that states of diagrams shown in Figure 3b and Figure 3d correspond to the following (1110), (0111), (1011), (101*), (*110), (01**) four–tuples of Figure 4.

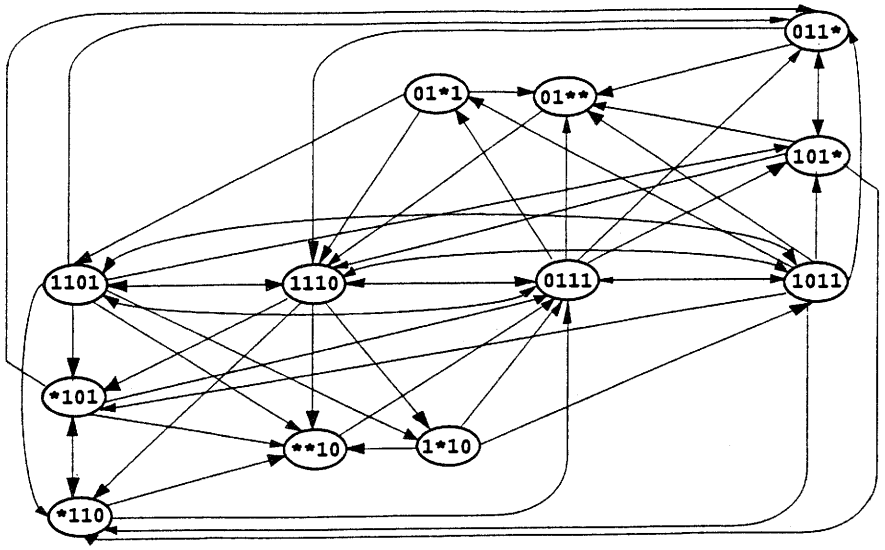

Fig. 4. Reachability graph of the MQTS model from Figure 2.

Moreover, comparing the considered reachability digraph with diagrams in Figure 3b and Figure 3d it becomes evident that assumption of particular values of operation times $T(\mathrm{P}_i, \mathrm{R}_j)$ limits the whole state space to some of its subspaces. In the case considered, it leads to the subgraph shown in Figure 5.

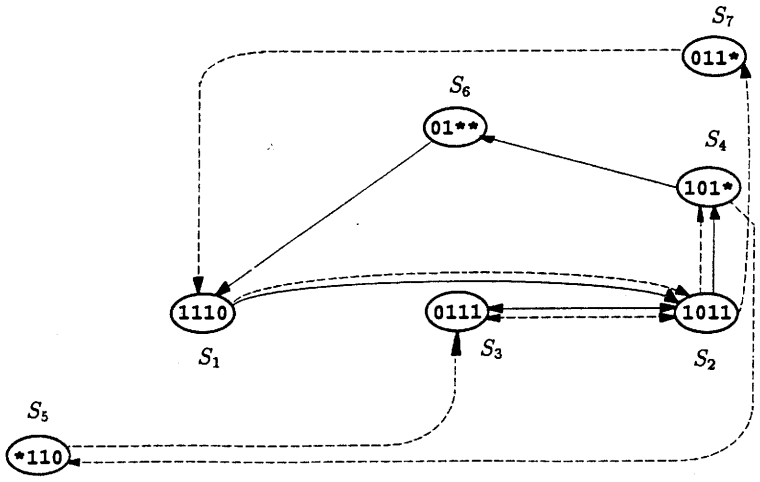

Fig. 5. The subgraph obtained from the reachability graph shown in Figure 4, for $T(\mathrm{P}_i, \mathrm{R}_j)$ shown in the charts in Figure 3b and Figure 3d.

Charts in Figure 3b and Figure 3d describe the ways of executing two different processes. Their alternative representations, encompassing the relevant processes flow are shown in Table 1 and Table 2, respectively.

Tab. 1. Alternative representation of the processes realisation shown in Figure 3b.

Time	t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\boldsymbol{V}(t)$	v_1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
	v_2	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1
	v_3	1	1	1	1	1	*	1	1	1	1	1	*	1	1	1
	v_4	0	1	1	1	*	*	0	1	1	1	*	*	0	1	1
State	S_i	S_1	S_2	S_3	S_2	S_4	S_6	S_1	S_2	S_3	S_2	S_4	S_6	S_1	S_2	S_3

Tab. 2. Alternative representation of the processes realisation shown in Figure 3d.

Time	t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\boldsymbol{V}(t)$	v_1	1	1	0	1	1	*	0	1	1	0	1	1	0	1	1
	v_2	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0
	v_3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	v_4	0	1	1	1	*	0	1	1	1	*	0	1	1	1	*
State	S_i	S_1	S_2	S_3	S_2	S_4	S_5	S_3	S_2		S_7	S_1	S_2	S_3	S_2	S_4

According to expressions (3) and (4), the following two sorts of cycles can be considered:

- the cycle time T_v associated with the sequence $\mathcal{V}(t)$ of process current values
- the cycle–state time T_s associated with the process states sequence $(S_{j_i} : i \in N)$

For the process determined by charts in Figure 3b and Figure 3d, the relevant cycles have the following values: $T_v = 6$, $T_s = 6$, and $T_v = 10$, $T_s = 9$, respectively. It means that for the same maximal efficiency of the component processes (i.e. equal to 1), the resultant executions of cascade–like process have different coefficients of resource utilisation. In the first case, according to the performance measure determined by formula (2) the efficiency of cascade–like interacting processes is equal to 0.833 while in the other case it is equal to 0.9. So, system resources are better utilised in the case of the other than in the case of the first execution of processes.

It is worth noting that ways of process realisation under consideration result in different implementations of dispatching rules. The dispatching rules determine the process flow in the case of occurrence of conflict events. Conflict events are defined as pairs of states of the same first component and such that there exist instants of time that all the states directly succeeding the common state are simultaneously available. For instance, events (S_4, S_5), (S_4, S_6) are conflict events in the digraph shown in Figure 5.

It means that the dispatching rules select a sequence of alternately following states and events. So, the ways that dispatching rules are used influence the resultant process which can be either cyclic or not. Moreover, the dispatching rules determine the parameters of system steady-state behavior, especially its efficiency.

The dispatching rules applied are usually assumed to be deterministic ones, i.e. either they are constant or their selection mechanism is determined by the explicitly given recurrent procedure. The deterministic character of dispatching rules as well as a finite state space guarantees a cyclic steady-state functioning of concurrently interacting processes.

2.2. Model Description

A system of processes that follows rules (10) and such that its structure matches the scheme shown in Figure 6 is said to be a system of cascade-like interacting cyclic processes.

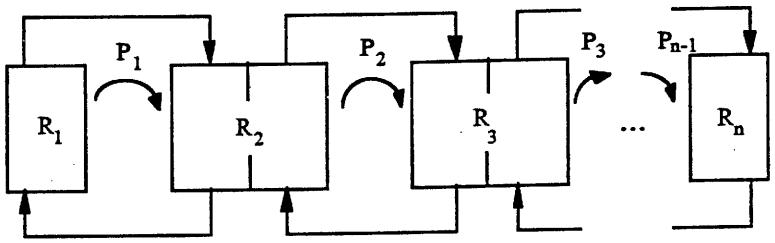

Fig. 6. Model of cascade-like interacting cyclic processes: P_i – the i-th simple process; R_i – the i-th system resource.

Let us remind that at any moment of time each simple (composed of two operations) process P_i is either executed or suspended on one of the following two resources R_i and R_{i+1}.

Let us introduce now a formal definition of the class of systems under consideration which is based on a concept of a simple cyclic process.

Definition 1. A sequence of vectors $SP = \{\mathcal{V}(t) : \mathcal{V}(t) = (v_1, v_2, t),\ t \in N,\ v_i \in \{0,1\}\}$, such that the following conditions hold is said to be a simple cyclic process specified by a cycle time T (SCP(T) – for short).

(i) $\mathrm{crd}_1\mathcal{V}(t) = 1$ and $\mathrm{crd}_2\mathcal{V}(t) = 0$ or
$\mathrm{crd}_1\mathcal{V}(t) = 0$ and $\mathrm{crd}_2\mathcal{V}(t) = 1$, $\quad \forall\ \mathrm{crd}_3\mathcal{V}(t) \in N$,

(ii) $\exists\, m, T \in No,\ \forall\, t > m : \boldsymbol{V}(t) = \boldsymbol{V}(t+T)$, ($\boldsymbol{V}$ defined in (3))
where $T = T(1) + T(2)$ is the cycle time of SCP(T), and

$$T(1) = \sum_{t=m+1}^{T} \mathrm{crd}_1\boldsymbol{V}(t), \qquad T(2) = T - T(1).$$

Suppose that components v_1 and v_2 of vector $\mathcal{V}(t)$ reflect the operations performed on the resources R_1 and R_2, respectively. So, the SCP(T) can be interpreted as a sequence of alternately executed operations. Therefore, the times required by the operations are determined by $T(1)$ and $T(2)$, and their sum specifies a cycle time T of the process considered.

Further, for the sake of simplicity, the following notations P_i and SCP(T_i) of the i–th simple elementary component process will be used equivalently (whenever it will not lead to confusion).

Definition 2. The system composed of $R_1, R_2, ..., R_{n+1}$ resources which are accessed by $P_1, P_2, ..., P_n$ simple cyclic processes SCP(T_i) in the way following the scheme of Figure 6 is said to be a system of cascade–like coupled cyclic processes specified by sequence of time cycles $T^n = (T_1, T_2, ..., T_i, ..., T_n)$ (CP(T^n) – for short).

The processes executed in CP(T_n) are determined by the following definition.

Definition 3. The sequence of vectors

$$P_v = \{\mathcal{V}(t) : \mathcal{V}(t) = (v_1, v_2, ..., v_j, ..., v_{n+1}, t),\ t \in N,\ v_j \in \{0, 1, *\}\},$$

such that the following conditions hold is said to be a process of CP(T^n) (PCP – for short):

(i) $\forall\, t \in N, \quad \exists!\ j \in \{1, 2, ..., n+1\}, \quad v_j = 0, \quad$ and

$$\begin{array}{ll} (v_{j-1} = v_{j+1} = 1 & \text{for } j \in \{2, 3, ..., n\} \text{ or} \\ v_{j+1} = 1 & \text{for } j = 1 \text{ or} \\ v_{j-1} = 1 & \text{for } j = n+1), \end{array}$$

(ii) if $\mathrm{crd}_k \mathcal{V}(t) = \mathrm{crd}_k \mathcal{V}(t+1) = 0$, then $\forall\, t \in N$,

$$\mathrm{crd}_r \mathcal{V}(t+1) = \begin{cases} * \text{ if } \mathrm{crd}_r \mathcal{V}(t) \in \{*, 1\} & \text{for } r \in \{1, 2, ...n+1\} \setminus \{k-1, k, k+1\} \\ 1 \text{ if } \mathrm{crd}_r \mathcal{V}(t) = 1 & \text{for } r \in \{1, 2, ...n+1\} \setminus \{k\} \end{cases}$$

(iii) if $\mathrm{crd}_k \mathcal{V}(t) = \mathrm{crd}_r \mathcal{V}(t+1) = 0$ and $k \neq r$, then $\mathrm{crd}_k \mathcal{V}(t+1) = 1$, for $t \in N$, and $\forall\, t \in N_0$

$$\mathrm{crd}_j \mathcal{V}(t+1) = \begin{cases} * \text{ if } \mathrm{crd}_j \mathcal{V}(t) \in \{*, 1\} & \text{for } j \in \{1, 2, ...n+1\} \setminus \{k, r-1, r, r+1\} \\ 1 \text{ if } \mathrm{crd}_j \mathcal{V}(t) \in \{*, 1\} & \text{for } j \in \{1, 2, ...n+1\} \setminus \{k, r\} \end{cases}$$

The relationship between simple cyclic processes $\mathcal{V}_i(t)$, $i \in \{1, 2, ..., n+1\}$, and the resultant process $\mathcal{V}(t)$ is determined by formula (9). It means that at each instant t for $\mathcal{V}(t)$ and $\mathcal{V}_i(t)$, for $i \in \{1, 2, ..., n+1\}$, the following condition holds:

$$\mathrm{crd}_i\mathcal{V}(t) = \begin{cases} 1 & \text{if} \quad \mathrm{crd}_1\mathcal{V}_i(t) = 1 \quad \text{or} \quad \mathrm{crd}_2\mathcal{V}_{i-1}(t) = 1 \\ 0 & \text{if} \quad \mathrm{crd}_1\mathcal{V}_{i-1}(t) = 1 \quad \text{and} \quad \mathrm{crd}_2\mathcal{V}_i(t) = 1 \\ * & \text{if} \quad \text{process } \mathrm{P}_i \text{ is hanged on } \mathrm{R}_i \text{ and} \\ & \quad (\,\mathrm{crd}_1\mathcal{V}_{i-1}(t) = 1 \text{ and } \mathrm{crd}_1\mathcal{V}_{i+1}(t) = 1 \quad \text{or} \\ & \quad\ \mathrm{crd}_1\mathcal{V}_{i-1}(t) = 1 \text{ and } \mathrm{crd}_{i+1}\mathcal{V}(t) = * \quad \text{or} \\ & \quad\ \mathrm{crd}_{i-1}\mathcal{V}(t) = * \text{ and } \mathrm{crd}_1\mathcal{V}_{i+1}(t) = 1 \quad \text{or} \\ & \quad\ \mathrm{crd}_{i-1}\mathcal{V}(t) = * \text{ and } \mathrm{crd}_{i+1}\mathcal{V}(t) = *\,) \end{cases}$$

According to formula (4) the above mentioned concept of a PCP process can be also considered in terms of a state space.

Definition 4. Let P_v be a given PCP process. The following sequence of states is said to be a state–process of CP(T^n) (SPC – for short).

$$\mathrm{P}_s = \{S : S = \boldsymbol{V}(t),\ V(t) \in \{\mathrm{crd}_i\mathrm{P}_v : i \in N\}\},$$

where $\boldsymbol{V}(t) = (v_1, v_2, ..., v_{n+1})$ is a vector $\mathcal{V}(t) = (v_1, v_2, ..., v_{n+1}, t)$ restricted to $v_1, v_2, ..., v_{n+1}$.

It is easy to note that P_s does not allow one to determine the "source process" P_v. As a consequence, P_s cannot be used for the quantitative evaluation of system performance. However, because of its natural interpretation in terms of theory of automata, P_s representation seems to be a well suited tool for the interacting processes analysis and the control procedures design.

2.3. Basic Concepts

In order to obtain formulae that provide quantitative evaluation of P^v processes, the following concepts of a conflict event, a critical resource, and a critical component process are introduced.

Let $\boldsymbol{P}_s(S^*) = \{\mathrm{P}_s : \mathrm{crd}_1\mathrm{P}_s = S^*\}$ be a set of all alternative P_s processes specified by a PCP process being, in turn, determined by the initial process value $\mathcal{V}(1)$ (that corresponds to $S^* = \boldsymbol{V}(1)$), and a set of operation times

$$OT = \{T(\mathrm{P}_i, \mathrm{R}_i), T(\mathrm{P}_i, \mathrm{R}_{i+1}) : i \in \{1, 2, ..., n\},\ \ T(\mathrm{P}_i, \mathrm{R}_i),\ T(\mathrm{P}_i, \mathrm{R}_{i+1}) \in N\}$$

Consider the graph $G(S^*) = (S\mathrm{P}^*, \langle)$ such that

$$S\mathrm{P}^* = \{\mathrm{crd}_j\mathrm{P}_s : \mathrm{P}_s \in \boldsymbol{P}_s(S^*),\ \ j \in N\},\ \ \langle\ \subset S\mathrm{P}^* \times S\mathrm{P}^*,$$

where

$$S \langle S' \overset{\text{def}}{\Longleftrightarrow} \exists\ \mathrm{P}_s \in \boldsymbol{P}_s(S^*),\ \exists\ k \in N : S = \mathrm{crd}_k\mathrm{P}_s,\ S' = \mathrm{crd}_{k+1}\mathrm{P}_s,\ \forall\ S, S' \in S\mathrm{P}^*.$$

It is worth noting that

(i) from Definition 4 and assumptions concerning $\boldsymbol{P}_s(S_i)$ it follows that besides of S^*, the graph $G(S^*) = (S\mathrm{P}^*, \langle)$ is also determined by a set OT;

(ii) according to assumed rules (10) and formula (9) it follows that a finite number of component processes implies a finite set of states $S\mathrm{P}^*$;

(iii) the graph shown in Figure 5 is specified by $G(S_1)$ which in turn provides two possible process realisations to be shown in Table 1 and Table 2.

The following definition providing a concept of a Flow Graph enables us to introduce a definition of conflict events.

Definition 5. Consider $G(S^*) = (S\mathrm{P}^*, \langle)$. The graph $\boldsymbol{\Gamma}(S^*) = (\boldsymbol{X}^*, A)$ is said to be a Flow Graph (FG – for short) of possible and initialised at S^* realisations of processes $\boldsymbol{P}_s(S^*)$, if the following conditions hold:

(i) $\boldsymbol{X}^* = X^*/_\sim$ is a set of equivalence classes, where the equivalence relation $\sim \subset X^* \times X^*$, $X^* = \{(S,n) : S \in S\mathrm{P}^*, \quad n \in N\}$, is defined as follows:
$x \sim x' \overset{\text{def}}{\Longleftrightarrow} \mathrm{crd}_1 x = \mathrm{crd}_1 x'$ and $\mathrm{crd}_2 x = \mathrm{crd}_2 x' + T$, $\forall\, x, x' \in X^*$;

(ii) $x\, A\, x' \overset{\text{def}}{\Longleftrightarrow} \mathrm{crd}_1 x \langle \mathrm{crd}_1 x'$ and $\mathrm{crd}_1 x [> \mathrm{crd}_1 x'$, where $\mathrm{crd}_1 x [> \mathrm{crd}_1 x'$ means that $\mathrm{crd}_1 x'$ is directly available from $\mathrm{crd}_1 x$ at the moment $\mathrm{crd}_2 x$ and $\mathrm{crd}_2 x' = \mathrm{crd}_2 x + 1$ holds;

(iii) $(\mathrm{crd}_1 x \langle \mathrm{crd}_1 x') \overset{\text{def}}{\Longleftrightarrow} (\mathrm{crd}_2 x = \mathrm{crd}_2 x' + 1$ or $\mathrm{crd}_2 x' - \mathrm{crd}_2 x = nT)$, $\forall\, x, x' \in X^*$, $\forall\, n \in N$.

As one can see, the $G(S^*)$ shown in Figure 5 may serve as an illustrative example of the definition provided.

In the case of the example considered earlier (see Tab. 1 and Tab. 2), the following set of processes is identified $\boldsymbol{P}_s(S_1) = \{\mathrm{P}(S_1), \mathrm{P}(S_1')\}$
where

$$\mathrm{P}(S_1) = S_1, S_2, S_3, S_2, S_4, S_6, S_1, S_2, S_3, S_2, S_4, S_6, S_1, S_2, S_3, \ldots$$

$$\mathrm{P}(S_1') = S_1, S_2, S_3, S_2, S_4, S_5, S_3, S_2, S_7, S_1, S_2, S_3, S_2, S_4, \ldots$$

Thus, according to Definition 5, the relevant FG has its graphical representation as shown in Figure 7.

Definition 6. Consider $\boldsymbol{\Gamma}(S^*) = (\boldsymbol{X}^*, A)$ being a FG where each pair (x, x'), $x, x' \in X^*$, such that $x\, A\, x'$, is said to be an event. Two events $e = (x, x')$ and $e' = (x'', x^*)$ are said to be the conflict events, i.e. $e, e' \in E$ if and only if $\mathrm{crd}_1 e = \mathrm{crd}_1 e'$ holds, where $E \subset A$ denotes a subset of conflict events.

The occurrence of conflict events indicates the possibility of alternative process execution. In the case considered (see FG shown in Fig. 7) there exist only two conflict events, i.e. $e = (x_5, x_7)$ and $e' = (x_5, x_6)$.

Because alternative process executions lead, in the general case, to different values of performance measure, thus the selection problem of an appropriate realisation plays a primary role. For its solution, the dispatching rules are applied. It means that a problem of optimal control of PCP processes can be treated as a problem of the relevant dispatching rule selection.

In order to consider the quantitative properties of 'PCP processes behavior let us introduce the following definitions.

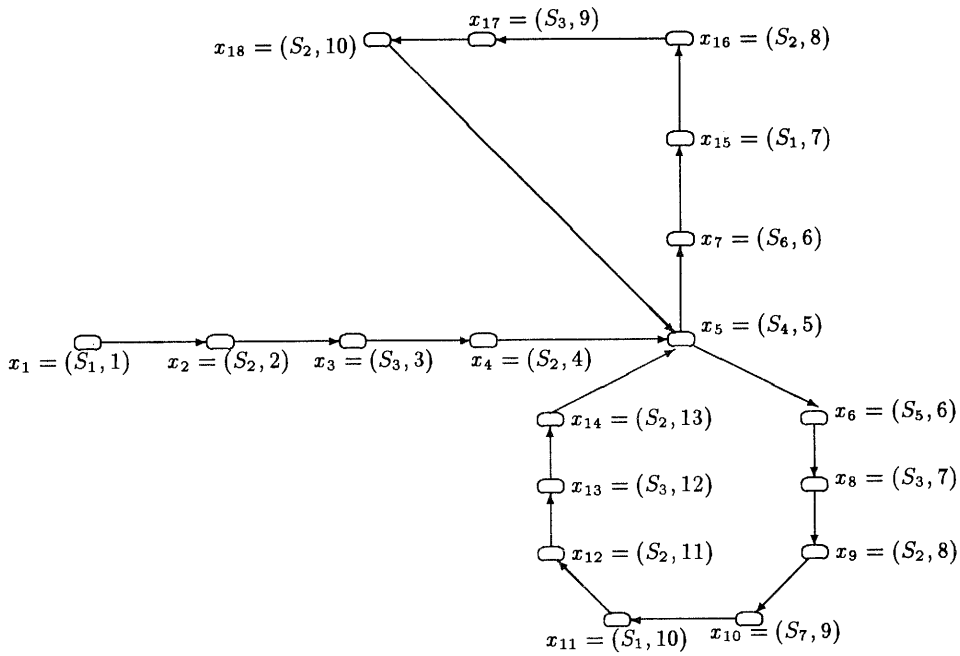

Fig. 7. The flow graph obtained from the graph shown in Figure 5.

Definition 7. Consider CP(T^n) system. Let $P_s = \mathcal{S}_{j_1}, \mathcal{S}_{j_2}, \mathcal{S}_{j_3}, ..., \mathcal{S}_{j_i}, ...,$ be a SPC process such that each state $\mathcal{S}_{j_i}$ has the form determined by rule (7). The k–th component process P_k is said to be a critical component process (CCP–for short) if and only if the following condition holds

$$\mathrm{crd}_k(\mathrm{crd}_m P_s) \neq *, \quad \forall\, m \in N$$

Definition 8. Consider CP(T^n) system. Let $P_s = S_{j_1}, S_{j_2}, S_{j_3}, ..., S_{j_i}, ...$ be a SPC process such that each state S_{j_i} has a form determined by rule (9). The k–th resource R_k is said to be a critical resource (CR – for short) if and only if the following condition holds

$$\mathrm{crd}_k(\mathrm{crd}_m P_s) = 1, \quad \forall\, m \in N$$

The process P_2 in Figure 2c and the resource M_3 in Figure 2d can be considered as the relevant illustrative examples of the concepts provided.

The concepts provided reflect the fundamental features of the class of processes under consideration, which are the subject of the following proposition.

Proposition 1. *Every* CP(T^n) *system contains either a critical component process or a critical resource.*

In order to prove this thesis let us note that the simultaneous absence of a critical resource and a critical process violates rules (10) which determine the CP(T^n) system behavior. The component processes compete with the access to common system resources

in order to reach their performance before their connection. This results in occurrences of either the most busy resource or the slowest process which, in turn, acts as "bottlenecks" in the P_v execution. In other words, the concepts provided in Definition 7 and Definition 8 reflect the nature of CP(T^n) system behavior which is aimed at the best utilisation of system resources.

Now let us introduce two new concepts of a steady–state and transient–state behaviors of CP systems. In the remainder of this paper, the following abbreviations SSB and TSB will be used instead of the terms: the steady–state and transient–state behavior, respectively.

In the case of the example considered earlier (see Tab. 1 and Tab. 2), the following processes were identified

$P_s = S_1, S_2, S_3, S_2, S_4, S_6, S_1, S_2, S_3, S_2, S_4, S_6, \ldots$ – obtained for dispatching rule #1,

$P_s = S_1, S_2, S_3, S_2, S_4, S_5, S_3, S_2, S_7, S_1, S_2, S_3, \ldots$ – obtained for dispatching rule #2.

These processes are cyclic and contain the states that belong to steady–state behavior of CP(T^n), i.e. the sequence of states which follow formula (3) for $m = 0$. Usually, the steady–state behavior is preceded by a transient–state behavior, i.e. the sequence of states determined by formula (3) for $m > 0$.

As it can be seen, different operation times $T(P_i, R_j)$ lead to different cycles. For example, the following operation times

$$T(P_1, R_1) = 1, \quad T(P_2, R_2) = 3, \quad T(P_3, R_3) = 1,$$

$$T(P_1, R_2) = 1, \quad T(P_2, R_3) = 1, \quad T(P_3, R_4) = 1$$

lead to the process $P_s = S_2, S_7, S_1, S_3, S_4, S_5, S_3^*, S_4, \ldots$, where the sequence S_2, S_7, S_1 determines the transient–state behavior, and the relevant steady–state behavior is described by the sequence $S_3, S_4, S_5, S_3, S_4, S_5, \ldots$.

Also, different initial stages lead to different transient–state behaviors. For instance, assuming the initial stage $U_1(1) = 2$ of the component process V_1 (see former example) leads to the following process

$$P_s = S_1, S_3, S_2, S_7, S_1, S_2, S_3, S_2, S_4, S_6, S_1, S_2, S_3, S_2, S_4, S_6, \ldots$$

where the transient–state and steady–state behaviors are determined by the sequence S_1, S_3, S_2, S_7 and $S_1, S_2, S_3, S_2, S_4, S_6, S_1, S_2, S_3, S_2, \ldots$, respectively.

Here we assume that $U_i(j)$ means a stage of the j–th operation executed in the i–th component process at the initial moment $t = 1$; for example, the first operation of the first component process P_1 has two stages $U_1(1) \in \{1, 2\}$, and the second operation of the process considered has just one stage $U_1(2) \in \{1\}$. Note that sets $\{1, 2\}$, $\{1\}$ contains all natural numbers limited by the relevant operation times $T(P_1, R_1) = 2$ and $T(P_1, R_2) = 1$.

Many questions arise regarding the modelling and performance evaluation of the systems composed of cascade interacting cyclic processes. For example: what are the conditions that determine the CP(T^n) system steady–state behavior? Is a cycle of a CP(T^n) system an initial state invariant? What are the conditions of the transient–state

occurrence? How does the cycle time of a steady–state behavior of CP(T^n) depend on the periodicity of its component processes? What are the conditions that guarantee the absence of conflict events? How are the steady–state and transient–state behaviors influenced by parameters determining values of initial stages of component processes?

3. Self–Synchronisation Conditions

The questions mentioned earlier inspire and motivate our further considerations that are aimed at providing the conditions determining a self–synchronisation mechanism that plays a primary role in the course of a CP(T^n) control problem solving.

3.1. Problem Formulation

The objective of these investigations is two folded. The providing of the analytical formulae that renders it possible to calculate an efficiency of CP(T^n) system is our first goal. The other objective is to provide the conditions that are responsible for a self–synchronised behavior of CP(T^n) systems.

In order to fulfil the requirements of the first objective formula (2) is taken into account as a performance measure. Its parameters, such as a cycle time $T = T_v$ and n_i determining a number of times the i–th component process occurs during the cycle time T_v, have to be calculated. The problem considered is to find out an analytical model describing the relationship existing between a CP(T^n) system behavior (evaluated by η) and cycle times T_i of component processes SCP(T_i) as well as an initial state. It means that the problem considered consists of the following two subproblems.

The first one concerns finding out a formula that provides T_v as a function of initial state S^* and a set of operation times OT. The other problem can be stated as a task of developing a procedure that expresses the dependence existing between n_i, where $n_i \in N$, and a set of a given data including a cycle T_v, a set OT, and CCP and/or CR as well. Solution of the above mentioned subproblems provide the components of the performance measure assumed.

As opposed to the goal discussed above the other objective concerns the evaluation of qualitative measures of CP(T^n) performance. The investigations of relationships existing between the introduced concepts of PCP periodicity, conflict events, CCP, CR, SSB, and TSB7 are the main subjects of further considerations. So, the problem considered is how to find the conditions to predict the qualitative features of CP(T^n) behavior. Among others, the conditions ensuring the cyclicity of PCP executed in CP(T^n) are the conditions on which we will focus further attention.

Because of the space limits, the scope of the above declared investigations will be restricted to the CP(T^n) having the unique CCP. We believe, however, that the results obtained will provide a formal framework to investigate the whole class of CP(T^n) as well as other classes of concurrently interacting cyclic processes.

3.2. Periodicity

Let us consider a set of all states reachable in CP(T^n) for $T^n \in N^n$ and determined as $\boldsymbol{S}(n) = \{S = (s_1, s_2, ..., s_i, ..., s_n) : s_i \in \{0, 1, *\}\}$ such that the following conditions hold

(i) $\forall S \in \boldsymbol{S}(n), \quad \exists!\ i \in \{1,2,...,n\} : s_i = 0;$

(ii) if $s_i = 0$ and $i \in \{2,3,...,n-1\}$, then $s_{i+1} = s_{i-1} = 1$, and if $s_1 = 0$, then $s_2 = 1$, and if $s_n = 0$, then $s_{n-1} = 1$.

Note that the above conditions directly follow from condition (i) of Definition 3 and lead to conclusion that value $\mathrm{crd}_i S = 0$ occurs uniquely in each state $S \in \boldsymbol{S}(n)$.

As an illustration of the above introduced notion, the reachability graph $G = (\boldsymbol{S}(4), \langle)$ shown in Figure 4 can be considered. The relation $\langle$ is defined in section 2.3.

The concept introduced here of the set of states $\boldsymbol{S}(n)$ which are potentially reachable in a class of $\mathrm{CP}(T^n)$, $T^n \in N^n$, as well as the following descriptions provide a formal basis for further considerations:

- P_v that provides a P_s (according to Definition 4) is said to be a source process of P_s,
- Let $\boldsymbol{\Gamma}(S^*) = (\boldsymbol{X}^*, A)$ be a given FG. A set $X_c \subset \boldsymbol{X}^*$ is said to be a set of cycle states, i.e. a set of nodes $x \in X_c$ such that for each of them there exists a closed path of nodes $\alpha = x, x', ..., x^r$ and such that $(x^r, x) \in A$, $x, x', ..., x^r \in X_c$.

Proposition 2. *Consider* $\mathrm{CP}(T^n)$. *Let* $\boldsymbol{\Gamma}(S^*) = (\boldsymbol{X}^*, A)$ *be a* FG *determined by a given* P_s *such that* $\mathrm{crd}_1 \mathrm{P}_s = S^*, S^* \in \boldsymbol{S}(n)$.
$E = \emptyset$ *if and only if* $\|\boldsymbol{P}(S^*)\| = 1$ *and* $\mathrm{crd}_1 \mathrm{P}_s = S^*$, $\mathrm{P}_s \in \boldsymbol{P}(S^*)$.

Proof. First, we assume that the set of conflict events $E = \emptyset$. Thus, from Definition 6 it follows that the corresponding FG (see Definition 5) contains a unique cycle. Moreover, each $x \in X$ has a unique successor x' such that $(x, x') \in A$ holds. Since there exists a unique sequence $\lambda = x, x', ..., x^*$ such that either $x \in X_c$ or $(x^*, x) \notin A$, $\forall\, x^* \in X$, hence there exists a unique, infinite sequence of states $\mathrm{P}_s = (S : S = \mathrm{crd}_1 x \text{ and } x \in X)$ such that $\mathrm{crd}_1 \mathrm{P}_s = S^*$ and $S^* = \mathrm{crd}_1 x$. Consequently, unique P_s yields $\|P_s(S^*)\| = 1$ and $E = \emptyset$ holds for $\boldsymbol{\Gamma}(S^*)$ obtained from $G(S^*)$. $\|\boldsymbol{P}(S^*)\| = 1$, $\mathrm{crd}_1 \mathrm{P}_s = S^*$, $\mathrm{P}_s \in \boldsymbol{P}(S^*)$ implies $E = \emptyset$ (see Definition 5 and Definition 6). ■

Corollary 1. *Consider the assumptions of Proposition 2. If* $E = \emptyset$, *then* P_s *is a cyclic process.*

Since $\boldsymbol{S}(n)$ is finite, the proof directly follows from Definition 4, Definition 3, and formula (4).

The following lemma leads to a theorem that provides the conditions allowing one to search for the qualitative properties of $\mathrm{CP}(T^n)$ processes.

Lemma 1. *Consider a given system of cyclic processes* $CP(T^n)$ *and the relevant set of the potentially reachable states* $\boldsymbol{S}(n)$.
If there exists P_s *such that* $\|\boldsymbol{P}(S^*)\| = 1$, $S^* = \mathrm{crd}_1 \mathrm{P}_s$, $S^* \in \boldsymbol{S}(n)$, $\mathrm{P}_s \in \boldsymbol{P}(S^*)$, *then for each* P'_s *such that* $S^* \neq S^{**}$, *where* $S^{**} = \mathrm{crd}_1 \mathrm{P}'_s$, *the following condition holds*

$$\|\boldsymbol{P}(S^{**})\| = 1, \quad \mathrm{P}'_s \in \boldsymbol{P}(S^{**}). \tag{11}$$

Proof. Assume that S^{**} is such that there exists $x \in X^*$, $\boldsymbol{\Gamma}(S^*) = (\boldsymbol{X}^*, A)$ and $S^{**} = \mathrm{crd}_1 x$. Thus, either $X^{**} \cap X^* = X_c{}^*$ or $X^{**} \cap X^* \neq X_c{}^*$, i.e. the intersection of FGs generated by P_s and P'_s is either equal to a set of cyclic states or not. In

the first case, because $G(S^*)$ has a unique cycle, then $G(S^{**})$ has to have also one (the same) cycle. In turn, the assumption that there exist more than one cycle leads to contradiction with $||\boldsymbol{P}(S^*)|| = 1$. The second case yields contradiction with assumption $||\boldsymbol{P}(S^*)|| = 1$. Note, that the following case $X^{**} \cap X^* = 0$ cannot occur because the intersection of a set of states which are contained in a cyclic process, with a set of any other process is not an empty set. This is because the finite operations times imply that any process realisation must include at least two states S and S' such that for each one the following condition holds

$$\exists\, k \in \{1,2,...,n\}\ \forall\, j \in \{1,2,...,n\} \setminus \{k\} : \mathrm{crd}_k S = 0 \ \text{ and } \ \mathrm{crd}_j S = 1, \quad (12)$$

where n is a number of all the component processes P_i of a given $\mathrm{CP}(T^n)$. Moreover, any cyclic process contains at least n states such that condition (12) holds.

■

The above lemma provides the conditions that are sufficient for the existence of a unique cycle in $\mathrm{CP}(T^n)$. This fact is expressed in the following corollary.

Corollary 2. *Consider a given system of cyclic processes* $\mathrm{CP}(T^n)$ *and the relevant set of states* $\boldsymbol{S}(n)$.
If there exists P_s *such that* $||\boldsymbol{P}(S^*)|| = 1$, $S^* = \mathrm{crd}_1 \mathrm{P}_s$, $S^* \in \boldsymbol{S}(n)$, $\mathrm{P}_s \in \boldsymbol{P}(S^*)$, *then every* P'_s *such that* $S' \in \boldsymbol{S}(n)$ *have the same unique cycle specified by a cycle time* T_v.

The following theorem provides the generalisation of the results obtained so far.

Theorem 1. *Consider a given system of cyclic processes* $\mathrm{CP}(T^n)$ *and a relevant set of potentially reachable states* $\boldsymbol{S}(n)$.
If there exists P_s *such that* $||\boldsymbol{P}(S^*)|| = 1$, $S^* = \mathrm{crd}_1 \mathrm{P}_s$, $S^* \in \boldsymbol{S}(n)$, $\mathrm{P}_s \in \boldsymbol{P}(S^*)$, *then* $\mathrm{CP}(T^n)$ *has a unique cycle which is an initial state and an initial phase independent.*

Proof. The uniqueness of a cycle and its initial state independence follows directly from Lemma 1 and Corollary 2.
In order to prove that the cycle is also the initial phase independent let us note that the different possible initial phases of a given state $S' \in \boldsymbol{S}(n)$ that influence the process execution are included in the reachability graph $G = (\boldsymbol{S}(n), \langle\rangle)$, i.e. they are encompassed by a set of events $\{(S', S'') : (S', S'') \in \langle\ \rangle$. Because for any initial phase of an initial state S^* the relevant $\boldsymbol{P}_s(S^*)$ is such that for the associated $\boldsymbol{\Gamma}(S^*)$ the following condition $E = \emptyset$ holds, then the assumptions of Lemma 1 are satisfied. Thus, according to Corollary 2, all processes P_s beginning with S'' have the same cycle. ■

The above theorem provides the conditions sufficient for self–synchronised functioning of $\mathrm{CP}(T^n)$ systems. It means that, if for $\mathrm{CP}(T^n)$, there exists P_s such that $S^* = \mathrm{crd}_1 \mathrm{P}_s$ which has a unique realisation (the relevant graph $G(S^*)$ has a unique cycle), then all the possible processes initialised at any state of $\boldsymbol{S}(n)$ and at any possible initial phase will have a unique cycle specified by a cycle time T_v. The considered property reflects the so–called self–synchronised behavior of $\mathrm{CP}(T^n)$, i.e. behavior according to which any phase of any initial state results in a cyclic steady–state performance characterised by T_v.

Lemma 2. *Consider assumptions of Proposition 2. Let a unique* CCP *component* P_c *exist for a given* $\mathrm{CP}(T^n)$ *such that* $E = \emptyset$.

1. *If* $T_c > T_i$, $i \in \{1,2,...,n\} \setminus \{c\}$, *then the cycle time* T_v *of the corresponding source process* P_v *is equal to* T_c,
2. *If there exists* T_k *such that* $k \in \{c-1, c+1\}$ *and* $T_k > T_i$, $i \in \{1,2,...,n\} \setminus \{k\}$, *then the cycle time* T_v *of the corresponding source process* P_v *is equal to* nT_c, *where* $(n-1)T_c < T_k < nT_c$.

Proof. From Corollary 2 it follows that the corresponding source process P_v has a unique cycle time T_v. Thus, according to Definition 7 and Proposition 1, $T_v = mT_c$, $m \in N$, holds. The following two cases can be considered. First, the assumption $T_v = T_c$ yields the condition $T_c > T_i$, $\forall\, i \in \{1,2,...,n\}$, because $T_c = n_i T_i + x_i$, $x_i \neq 0$, $i \in \{1,2,...,n\} \setminus \{c\}$ holds. In turn, the assumption $m > 1$ yields $T_v = mT_c$. Moreover, because $mT_c = n_i T_i + x_i$, $x_i \neq 0$, $i \in \{1,2,...,n\} \setminus \{c\}$, then $\exists\, k \in \{1,2,...,n\}$, $\forall\, i \in \{i,2,...,n\} \setminus \{k\} : T_k \geq T_i$ holds. Thus, the assumptions $T_k > T_i$, $\forall\, i \in \{1,2,...,n\} \setminus \{k\}, k \in \{c-1, c+1\}$ implies that $mT_c = T_k + x_k$, $x_k \neq 0$ such that $x_k < T_c$ holds. That is because either $T(P_{c+1}, R_{c+1}) < T(P_c, R_c)$ or $T(P_{c-1}, R_{c-1}) < T(P_c, R_c)$ holds. Consequently, the following condition is true: $(m-1)T_c < T_k < mT_c$. ■

Corollary 3. *Consider assumptions of Proposition 2. Let a unique* CCP *component* P_c *exist in a given* $CP(T^n)$ *such that* $E = \emptyset$.
If there exists T_k *such that* $k \in \{c-1, c+1\}$ *and* $T_k > T_i$, $\forall\, i \in \{1,2,...,n\} \setminus \{k\}$, *then a cycle time* T_v *of the corresponding source process* P_v *is determined by the formula* $T_v = nT_c$, *where* $n = T_k$ div T_c, *and* a div b *is an integer result of divide operation between* a *and* b.

The proof follows directly from Lemma 2.

The results provided so far present the conditions sufficient for $CP(T^n)$ cyclic behavior as well as for calculation of a cycle time T_v. In the next section their extension to the conditions that allow one to design $CP(T^n)$ systems of a guaranteed presumed cyclic behavior are presented.

3.3. Performance Measure

Besides the conditions making it possible to calculate a cycle time T_v in order to apply the evaluation measure η the formulae determining n_i, $i \in \{1,2,...,n\}$, are required. First of all let us consider the following proposition stating that the number n_i describing an amount of the component process P_i occurrences during the period T_v is an integer.

Proposition 3. *Consider a given* $CP(T^n)$. *Let* P_s *be a* SPC *specified by a cycle time* T_v *of the corresponding source process* P_v. *Each component process* P_i *occurs in* $CP(T^n)$, *during the period* T_v, $n_i \in N$ *times.*

Proof. We need to prove that $n_i \in N$, $i \in \{1,2,...,n\}$. From Definition 4, Definition 2, and formula (3) it follows that the cycle is finite. Thus, let us consider a matrix C of size $(m \times n)$ such that each row describes the subsequent state of the form (7) and where m determines the number of states occurring in the cycle, and n the number of component processes, i.e. the length of a state vector $\mathcal{S}$.

Note that the i–th column of matrix C presents an execution flow of the relevant component process P_i. Thus, in the i–th column sequences composed either of i or

composed of $i+1$ or composed of $*$ can occur. Consequently, because P_s is cyclic, then for the i-th column the number n_i determining the occurrences of subsequences composed of i is the same as the number determining the occurrences of subsequences composed of $i+1$. ■

The above proposition provides the results enabling us to consider an analytical model of the performance measure η. The conditions which support the procedure aimed at calculation of n_i are presented in the following lemma.

Lemma 3. *Consider a given* CP(T^n). *If* $T^n = (T_1, T_2, ..., T_n)$ *is such that* $\forall\, i \in \{1, 2, ..., n\}$,

(i) $T_1 > T_2 > T_3 > .... > T_{n-1} > T_n$,

(ii) $T(P_i, R_{i+1}) < T_{i+1}$,

(iii) $T(P_i, R_{i+1}) + T(P_{i+1}, R_{i+1}) < T(P_i, R_i) + T(P_{i+1}, R_{i+2})$, *and*

(iv) P_1 *is a* CCP *process*,

then $T_v = T_1$ and $n_1 = 1$, $n_2 = T_v \operatorname{div} T_2$, and for $i > 2$, the n_i can be calculated from the following recurrent formula

$$n_i = n_{i-1} \times n'_i, \quad \textit{where} \quad n'_i = T_{i-1} \operatorname{div} T_i \quad \forall\, i \in \{1, 2, ..., n\}$$

Proof. From the assumption the following equations hold

$$n_i T_i + x_i = n_{i+1} T_{i+1} + x_{i+1}, \quad \text{for} \quad i \in \{1, 2, ..., n-1\} \tag{13}$$

Because P_1 is a CCP process, then $x_1 = 0$ and consequently $n_1 = 1$. Thus, $T_1 = T_v$ holds. From $T_1 = n_2 T_2 + x_2$, $x_2 \neq 0$ and Proposition 3 it follows that $n_2 = T_v \operatorname{div} T_2$ holds.

In order to calculate n_i for $i > 2$, besides formula (13) the following set of equations should be considered

$$T_i = n'_{i+1} T_{i+1} + x'_{i+1}, \qquad \text{for} \quad i \in \{1, 2, ..., n-1\} \tag{14}$$

The solution of both sets (13) and (14) follows from the following formulae

$$n_i = n'_1 \times n'_2 \times ... \times n'_i, \quad x_i = x'_1 + x'_2 + ... + x'_i \tag{15}$$

Note that because $n'_1 = n_1 = 1$ and $x'_1 = x_1 = 0$ the following formula holds:

$$\underbrace{x'_1 + x'_2 + ... + x'_i}_{x_i} + \underbrace{n'_1 \times n'_2 \times ... \times n'_i}_{n_i} T_i = \underbrace{x'_1 + x'_2 + ... + x'_{i+1}}_{x_{i+1}} + \underbrace{n'_1 \times n'_2 \times .. \times n'_{i+1}}_{n_{i+1}} T_{i+1}$$

Finally, $n'_1 = T_{i-1} \operatorname{div} T_i$ follows from the assumption $T(P_i, R_{i+1}) < T_{i+1}$ and $T(P_i, R_{i+1}) + T(P_{i+1}, R_{i+1}) < T(P_i, R_i) + T(P_{i+1}, R_{i+2})$ for $\forall\, i \in \{1, 2, ..., n\}$. ■

Corollary 4. *Consider a given* CP(T^n). *If* $T^n = (T_1, T_2, ..., T_n)$ *is such that* $T_1 > T_2 > ... > T_n$ *and* P_1 *is a* CCP *process, then* $n_1 \leq n_2 \leq n_3 \leq ... \leq n_n$, *where* $n_1 = 1$, *holds.*

The proof directly follows from Lemma 3.

Corollary 5. *Consider a given* CP(T^n). *If* $T^n = (T_1, T_2, ..., T_n)$ *is such that* $T_1 < T_2 < ... < T_{k-1} < T_k > T_{k+1} > ... > T_n$ *and* P_k *is a* CCP *process and other assumptions of Lemma 3 hold, then* n_i *is determined by the following formula*

$$n_i = \begin{cases} n_{i-1} \times n'_i & \text{where } n'_i = T_{i-1} \operatorname{div} T_i \quad \text{for } i > k \\ n_{i+1} \times n'_i & \text{where } n'_i = T_{i+1} \operatorname{div} T_i \quad \text{for } i < k \end{cases}$$

The above corollary simply generalises the condition provided in Lemma 3.

Theorem 2. *If* $\{n_i : i = \{1, 2, ..., n\}\}$ *such that* $n_1 \leq n_2 \leq ... \leq n_n$ *and* T_v *are given, then there exists a* CP(T^n) *such that* $T^n = (T_1, T_2, ..., T_n), T_1 > T_2 > ... > T_n, T(P_i, R_{i+1}) < T_{i+1}$ *and* $T(P_i, R_{i+1}) + T(P_{i+1}, R_{i+1}) < T(P_i, R_i) + T(P_{i+1}, R_{i+2})$ *for* $i \in \{1, 2, ..., n\}$, $T_1 = T_v$, *and the cycle times* T_i, $i \in \{1, 2, ..., n\}$, *are the solutions of the following set of equations*

$$n'_i = T_{i-1} \operatorname{div} T_i,$$

$$n_i = n_{i-1} \times n'_i, \qquad \forall\, i \in \{2, 3, ..., n\}$$

Proof. Note that the considered cycle times T_i, $i \in \{1, 2, ..., n\}$ under consideration, are the solutions of the following set of equations, provided by Lemma 3,

$$x'_1 + x'_2 + ... + x'_i + n'_1 \times n'_2 \times ... \times n'_i T_i = x'_1 + x'_2 + ... + x'_{i+1} + + n'_1 \times n'_2 \times ... \times n'_{i+1} T_{i+1}$$

where $x'_1 + x'_2 + ... + x'_i = x_i$ and $n'_1 \times n'_2 \times ... \times n'_i = n_i$ for a given n_i, $i \in \{1, 2, ..., n\}$ (following the above assumption), $T_1 = T_v$, $n_1 = 1$, and $x'_i = T_{i-1} - n'_i T_i$, $i \in \{1, 2, ..., n\}$. ■

The above theorem provides the conditions that are sufficient for the calculation of η.

4. Performance Evaluation

Let us consider once again the performance measure introduced in Section 2. Now, according to the conditions submitted in Lemma 1, Lemma 2, and Lemma 3 the analytical formulae allowing us to calculate components such as T_v and n_i can be used to provide an analytical model of the performance measure η. Because of the recurrent character of its component formulae, the finally proposed model has the following form

$$\eta = f_1(T_1, T_2, ..., T_n) \text{ for } T_1 < T_2 < ... < T_k > T_{k+1} > ... > T_n \text{ and } T_k = T_c = T_v \qquad (16)$$

where f_1 is determined from the following expression

$$\eta = 1 - \frac{\sum_{i=1}^{n} (T_v - n_i T_i)}{nT_v} = 1 - \frac{nT_v - \sum_{i=1}^{n} n_i T_i}{nT_v} = \frac{\sum_{i=1}^{n} n_i T_i}{nT_v}.$$

Function f_1 that is written in a Pascal–like pseudocode has the following form

```
Function f1(T1, T2, ..., Tn)                                        (17)
    function nl(i) { for i < k}
    begin
        if i ≠ k then nl = nl(i + 1) × (T_{i+1} div T_i)
                 else nl = T_v div T_n
    end
    function nr(i) { for i > k}
    begin
        if i ≠ k then nr = nr(i − 1) × (T_{i−1} div T_i)
                 else nr = T_v div T_k
    end
begin
    for j = 1 to k do WT = WT + nl(j) × T_j
    for j = n downto k do WT = WT + nr(j) × T_j
    WT = WT + T_v { for j = 1}
    f1 = WT / (k × T_v) {WT = Σ_{i=1}^{n} n_i T_i}
end {f1}
```

It should be remainded that the model developed is valid only in the case when assumptions of Lemma 3 hold.

In order to illustrate the application of procedure (17), let us consider the following example. Let P_v be a SCP process executed in $CP(T^n)$ such that $T^n = (T_1, T_2, T_3, T_4, T_5)$ where

$$
\begin{array}{lll}
T_1 = 7, & T(P_1, R_1) = 5, & T(P_3, R_4) = 4 \\
T_2 = 9, & T(P_1, R_2) = 2, & T(P_4, R_4) = 2 \\
T_3 = 12, & T(P_2, R_2) = 6, & T(P_4, R_5) = 3 \\
T_4 = 5, & T(P_2, R_3) = 3, & T(P_5, R_5) = 1 \\
T_5 = 3, & T(P_3, R_3) = 8, & T(P_5, R_6) = 2
\end{array}
$$

Consider P_3 as a CCP process. Because assumptions of Lemma 3 and Corollary 6 hold, then T_v and η are determined as $T_v = 12$ and $\eta = 0.6$. The Gannt's chart of the above considered process is shown in Figure 8.

In general case, the value of η can be determined by using computer simulation methods. Let us recall, for instance, the processes specified by Table 1 and Table 2. Because the processes violate the conditions provided in Lemma 3, the efficiency is calculated using simulation techniques. The values obtained, viz. $\eta_1 = 0.833$ and $\eta_2 = 0.9$, correspond to realisations determined by dispatching rule #1 and dispatching rule #2, respectively. That means that the measure introduced may be used directly as a tool for evaluation of dispatching rules. Moreover, it may serve as a criterion in the process of decision making that follows from the optimal control problems.

Besides the performance measure η other ones may also be considered. Because of the space limits let us focus on two of them, i.e. an average cycle time AT_i of the i–th

component process P_i, and the system stability margin SM. The first of them is defined as follows

$$AT_i = \frac{T_v}{n_i} \tag{18}$$

where T_v is a cycle time, n_i – a number determining the amount of occurrences of the component processes P_i in the period T_v.

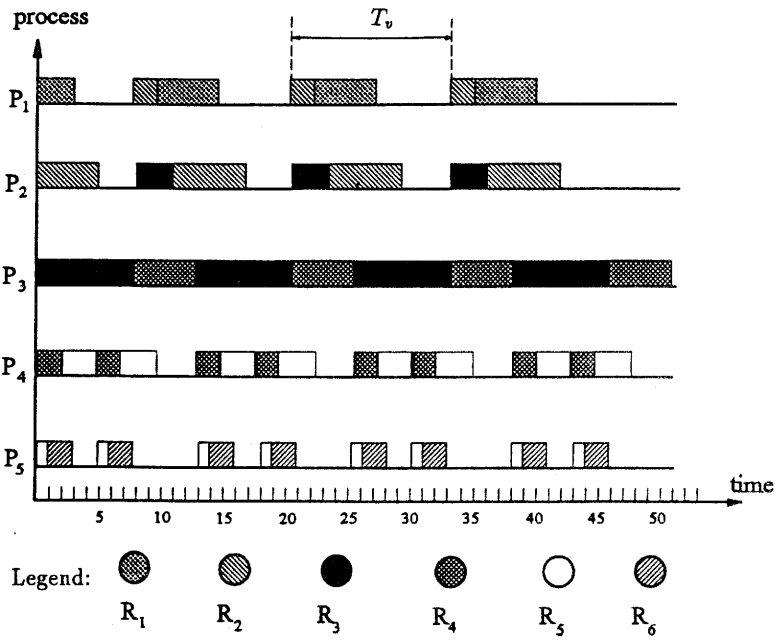

Fig. 8. The illustration of possible processes with one CCP.

The other measure is defined by the following formula

$$\mathbf{SM} = \min_{i \in \{1,2,\ldots,n\}} \{x_i : x_i \neq 0\} \tag{19}$$

where $x_i = T_v - n_i T_i$, and n_i, T_i follow from the assumptions of Lemma 3.

The average cycle time AT_i provides information about the "disturbance" of the nominal cycle time T_i, i.e. about the difference $AT_i - T_i$. Such information can be valuable for the efficiency evaluation of P_i usage. In turn, SM provides information about the possible changes of the cycle time T_i that may eventually lead to new critical processes, i.e. to the new cycle time T_v and η values.

The performance measures provided will be the subject of further investigations, particularly from the view point of more general classes of CP(T^n), i.e. classes including simultaneous occurrence of more than one critical process and/or critical resource.

5. Concluding Remarks

The paper presents a new approach to the modelling and performance evaluation of systems of discrete processes, sharing a set of reusable resources. The approach is illustrated on the basis of the cascade–like coupled processes. It can, however, be also applied to create models of more complex systems either belonging to or which can be reduced to the class considered.

The analysis of the behavior of the concurrently interacting cyclic processes leads to the conditions determining the periodic performance of the whole system. It has been proven that a unique P_v belonging to CCP processes is the sufficient condition for CP(T^n) self–synchronised functioning. It means that the functioning of a system is specified by the same cyclic steady–state behavior being independent of an initial state and/or its initial phase.

Besides the results having a qualitative character, the conditions sufficient for a cyclic performance, i.e. having a quantitative character, have been provided. Thus, we proposed an analytical model aimed at performance evaluation of a system composed of a set of cyclic processes as well as we provided the conditions sufficient for the synthesis of the systems encompassing a presumed cyclic performance.

The work presented in this paper is of great interest for evaluation and synthesis of a wide range of systems composed of a set of independent and loosely interacting cyclic processes. The manufacturing systems, computer/communication network systems, as well as ecology and sociological systems that are specified by their cyclic steady–state behavior, can be modelled and analysed using the approach proposed above.

References

Banaszak Z., Chudy C. and Skoczylas G. (1992): *On periodicity of concurrent processes.* — Technical Report No.3/92, Dept. of Computer Sci., Technical University in Zielona Góra, (in Polish).

Banaszak Z. and Chudy C. (1992): *Modelling and performance evaluation of cyclic processes.* — Technical Report No. 16/92, Inst. of Technical Cybernetics, Wrocław Technical University, Wrocław, (in Polish).

Braker J.G. (1991): *Max–algebra modelling and analysis of time–table dependent transportation networks.* — Proc. ECC'91 *European Control Conference*, Grenoble, France, pp.1831–1836.

Brave Y. and Heymann M. (1990): *Stabilization of discrete –event processes.* — Int. J. Control, v.51, No.5, pp.1101–1117.

Cao X-R. (1989): *A comparison of the dynamics of continuous and discrete event systems.* — Proc. IEEE, v.77, No.1, pp.7–13.

Carpenter G.F. (1987): *The use of Occam and Petri nets in the simulation of logic structures for the control of loosely coupled distributed systems.* — Proc. UKSC Conf. *Computer Simulation*, (Ed.) R.N. Zobel, UK, pp.30—35.

Caspi P. (1991): *Models of discrete event systems.* — Proc. ECC'91 *European Control Conference*, Grenoble, France, pp.503–511.

Cohen G., Moller P., Quadrat J-P. and Viot M. (1989): *Algebraic tools for the performance evaluation of discrete event systems.* — Proc. IEEE, v.77, No.1, pp.39–58.

Dembele S., Lhote F. and Bourrieres J.P. (1992): *Products and equipments united modelling: making cycles and flows generation in evidence.* — Proc. IFAC/INCOM'92, pp.545–550.

Hillion H.P. and Proth J-M. (1989): *Performance evaluation of Job–shop systems using timed event–graphs.* — IEEE Trans. Automatic Control, v.34, No.1, pp.3–9.

Ho Y-C. (1989): *Dynamics of discrete event systems.* — Proc. IEEE, v.77, No.1, pp.3–6.

Laftit S., Proth J-M. and Xie X-L. (1992): *Optimization of invariant criteria for event graphs.* — IEEE Trans. Automatic Control, v.37, No.5, pp.547–555.

Ramadge P.J.G. and Wonham W.M. (1989): *The control of discrete event systems.* — Proc. IEEE, v.77, No.1, pp.81–98.

Ramadge P.J.G. (1990): *On the periodicity of symbolic observations of piecewise smooth discrete–time systems.* — IEEE Trans. Automatic Control, v.35, No.7, pp.807–813.

Righter R. and Walrand J.C. (1989): *Distributed simulation of discrete event systems.* — Proc. IEEE, v.77, No.1, pp.99–113.

Scheuring R. and Wehlan H. (1991): *On the design of discrete event dynamic systems by means of the Boolean differential calculus.* — Prep. 1st IFAC Symp. *Design Methods of Control Systems*, ETH Zurich, Switzerland, (Eds.) D. Franke, F. Kraus, Pergamon Press, Oxford, N.Y., Tokyo, v.2, pp.723–728.

Viswandham N. and Narahari Y. (1992): *Performance Modelling of Automated Manufacturing Systems.* — New York: Prentice Hall.

Wedde H. (Ed.), (1983): *Adequate Modelling of Systems.* — Berlin: Springer–Verlag.

Weick K. (1987): *Management of organisational change among loosely coupled elements.* — In: Goodman P. *et al*, Change of Organisation. Jossey Bass.

Willner Y. and Heymann M. (1991): *On supervisory control of concurrent discrete–event systems.* — Proc. ECC'91 *European Control Conference*, Grenoble, France, pp.1460–1465.

Zeigler B.P. (1989): *DEVS representation of dynamical systems: event–based intelligent control.* — Proc. IEEE, v.77, No. 1, pp.72–80.

Received June 3, 1993
Revised December 21, 1993