5

Supervisory Control of Industrial Processes

Alexander A. Ambartsumyan
Institute of Control Sciences RAS,
Russia

1. Introduction

Modern production is complex, integrated and is constantly being adapted to the market requirements by means of the reconfiguration of equipment structure and process alteration. The development of such production is performed based on evolutionary strategy by successively engaging (eliminating) stand-alone technological systems.

Evolutionary developed technical systems and facilities presently make up a considerable share of technical systems. It is typical both for high-tech industries, namely: aviation, space exploration, military equipment, machine-building (Sujeet, 2005), and for applications based on large-scale interconnected production complexes (e.g. oil- and gas-producing industry, oil and gas transportation, city economy engineering etc) (Gilard, 1999; Van Brussel et al., 1999; Jo, 1999; Ambartsumyan, Prangishvili, Poletykin, 2003; Ambartsumyan, Kazansky, 2008; Ambartsumyan, Potehin, 2003; Ambartsumyan, Branishtov, 2006).

Evolutionary developed technical systems and facilities are featured by complex control system availability. The latter integrates into a single whole different, as to the purposes, automatic control loops (automatic control and regulation of physical process parameters, automatic shielding and blocking, logical configuration control) as well as the functions of supervisory control mainly aimed at coordination of different processes in a technical system.

Supervisory control (SC) is intrinsically logical and is to provide the required operational sequence and exclude mutual blocking and deadlocks for stand-alone components (operating according to their internal rules time scale). SC is discrete and asynchronous by its nature and most commonly reveals itself as the change of event flow as required by certain application (technical system functionality).

It is important to consider two "event" aspects: first, everything happens as the result of a certain event; second, the change of states is regulated by events – there is no physical time though the system is dynamic.

Though control systems are widely spread in the technical systems of such kind (Sujeet, 2005; Gilard, 1999; Van Brussel et al., 1999; Jo, 1999; Ambartsumyan, Prangishvili, Poletykin, 2003; Ambartsumyan, Kazansky, 2008; Ambartsumyan, Potehin, 2003; Ambartsumyan, Branishtov, 2006), presently there is no appropriate theoretical base to solve such supervisory control tasks as local control loops coordination, configuration of material flows structure and interaction with operations staff.

Most spread concept of practical engineering of such systems is based on the model of interacting "black boxes": a "black box–control object" and symmetrically connected with it as to inputs and outputs a "black box–control system (device)". (Fig. 1).

Source: Process Management, Book edited by: Mária Pomffyová,
ISBN 978-953-307-085-8, pp. 338, April 2010, INTECH, Croatia, downloaded from SCIYO.COM

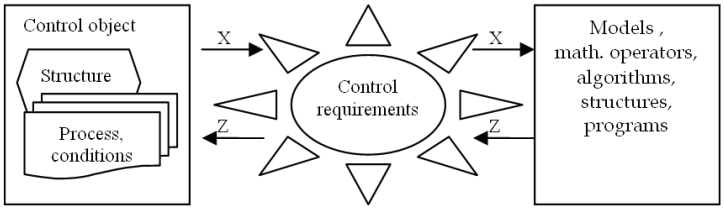

Fig. 1. The scheme of transfer from the object data base and control requirements to the mathematical description of the control

The first ″black box–control object″ is formed as a data base on the control object and technique at the stage of the object examination and includes the requirements of this object appropriate behaviour. The task of the required control search is tackled by the defining of a ″black box–control system″ able to monitor the behaviour – the event flow and, with the control purpose taken into account, to affect the object inputs in such a way that an appropriate behaviour of the object is achieved.

The question is how to search for a ″black box–control system″ with information on the first black box available. Common engineering practice shows that information on control object behaviour is only used indirectly.

What is the problem? We may speak about precise correspondence between a ″black box–control object″ and a ″black box–control system″ only as far as inputs and outputs are concerned, while behaviour is an approximate result of the designer's informal, speculative experiment with the initial data and limitations – the information the designer acquires considering the process physics peculiarities and the object structure properties. At that, there is not any confidence that a ″black box–control system″ can limit the behaviour of a ″black box–control object″ and provide its meeting the requirements since they, as a rule, are specified as models of another (not "event") nature and the extent they are taken into account depends on the designer's skills. The above leads to serious problems: designer's uncertainty in the fact that the designed system complies with the control tasks set; the necessity to make laborious verification of such compliance by computer simulation and the refinement of the designed system at facilities.

For the last 10–15, a sophisticated interaction among computer-driven actuating devices necessitates, when engineering, to analyze the design solutions safety and correctness, to validate technical systems implementation techniques, to take other approaches actually based on testing. It is a common knowledge that such approaches only can reveal a part of errors but cannot guarantee the system as a whole is error-free.

Different engineering approach than that based on two black boxes concept is declared in the theory of discrete event dynamic systems and supervisory control paradigm. The abbreviation is often simplified to DES. The distinctive features of supervisory control theory (all basic concepts and notions of this paper are borrowed from (Cassandras, Lafortune, 2008)) are as follows:

- The controlled object is represented in DES model by three components: generator G of $L(G)$ language – proper control object, specification language K – limitations and G functionality required, supervisor S – control component in DES;
- Setting and solving the task of formal synthesis of S on $L(G)$ and K.

The above, in its turn, creates a theoretical basis for machine control engineering fundamentally different from the deciphering of "black boxes" approximately fitting each

other. What does it give as compared with the classic procedure of discrete process control system synthesis according to two-black-boxes model?
First, the description of the object as *L(G)*-language generator *G*, limited by nothing, is more simple than the object description with all the admissible behaviour limitations taken into account. This work is performed as a separate stage – primary object examination and constructing a model "as it is".
Second, to form the required functionality (*K* specifications) basing on a generator *G* model already available is also easier than to consider all limitations and requirements in yet non-existing control system.
Third, control task is solved formally: a supervisor (provided the initial data is correct) is synthesized and does not require verification while the object and its behaviour are specified by object and know-how specialist and he is responsible for the data correctness, its verification and validation.
The present paper formulates the purpose of DES theory development, with the structural properties of technical systems taken into account, thus creating effective methods to synthesize a supervisor as an instrument to solve the task of consistency and co-ordination control of stand-alone components in a technical system.
Here below is given a brief survey of basic concepts and major noted results, as to DES and supervisory control, followed by the description of the present paper tasks and the results obtained.

2. Basic concepts and definitions

DES behaviour is considered generally as behaviour of a certain generator (source) of strings (sequences) of the events from a finite set of events *E*. The event $e \in E$ is an abstraction for a multitude of facts associated with DES "life". Events are instantaneous, occur spontaneously in unpredictable moments, therefore the only thing that can be observed is their sequences that are represented by strings. Event examples are: the facts of change in position and state of separate object components; commands to which the object reacts by the change of its state (position); characteristics of normal and abnormal states etc.
The main operation of strings forming is concatenation (we would like to remind that concatenation is the appending of separate events or entire strings of events on the right to the string, including ε – a space character). For the string, an integral function $\mu(s) = n$ is defined, where n is the number of characters in string s. If $n = 0$, $s = \varepsilon$. A set of all string of any finite length is designated by E^* (it is endless but countable). Let a string s consist of three parts: r, u, $t \in E^*$ connected by concatenation in such a way that $s = rut$, where r – a prefix, t – a suffix, and u – a substring of string s. Any subset of strings $L \subseteq E^*$ is called a language over E. If L includes ε and, jointly with any string s, contains all its prefixes, L is a prefix–closed language. As usual, conventional language operations are defined, namely: concatenation, prefix-closure and Kleene-closure.
In many constructions of DES theory, a couple of very important operations over languages are used: a projection P and a back projection P^{-1}. Let E_1, $E_2 \subset E$ be such that $E_1 \cup E_2 = E$ (possibly $E_1 \cap E_2 \neq \varnothing$). Projection P_i of any string from E^* on E_i is defined in three steps:
1. $P_i(\varepsilon) = \varepsilon$; 2. $P_i(e) = \varepsilon$ if $e \notin E_i$, otherwise $P_i(e) = e$; 3. $P_i(se) = P_i(s)\, P_i(e)$ for $s \in E^*$ and $e \in E$. Conceptually, a projection of strings from larger alphabet E on smaller one E_i deletes from the string all characters from $E \setminus E_i$ (all characters outside E_i). Inverse function $P_i^{-1}(s) = \{t \in$

E^*: $P_i(t) = s\}$. $P_{i-1}(s)$ correlates every string $s \in E_i$ with some subset of strings E^* the projects of which on E_i equal s. Both operations are in natural manner extended to the languages $L \subseteq E^*$ and $L_i \subseteq E_i^*$. $P_i(L)=\{t \in L_i: (\exists s \in L)\ [P_i(s) = t]\}$; $P_{i-1}(L_i) := \{s \in E^*: (\exists t \in L_i)\ [P_i(s) = t]\}$.

In projection operation definition, instead of set indexes, for the sets, the events of which are excluded from the result of this operation, we shall use the designation of the set itself: P_{E_i} or $P_{E_i}^{-1}$.

Languages are a good instrument to observe DES behaviour but in order to perform analytical study and to set the task of providing the required dynamics (off-line behaviour), it is necessary to present a countable string set as a mathematical operator. There are many ways to present languages in the form of mathematical operators that generate or recognise the language. In DES theory, for these purposes, as a rule, finite state machines are used. A finite state machine is defined as $G = (Q, E, \delta, \Gamma, Q_m, q_0)$, where Q – a set of states; E – a set of events; δ – a transition function $Q \times E \to Q$; $\Gamma : Q \to 2^E$ – a function of admissible events in each state; Q_m – a set of marked states; q_0 – an initial state. We would like to note that in this definition the function of outputs is missing. For every state q_i the function of transitions is specified for the events admissible in this state (e.g. for $q_i \in Q$ and $e \in \Gamma_i$ the function $\delta(q_i, e) := q_j$). This definition can be naturally extended also for the following event strings: $\delta(q_i, \varepsilon) := q_i$, $\delta(q_i, se) := \delta(\delta(q_i, s), e)$ for $s \in E^*$ and $e \in E$. Let's denote by $\delta(q_i, s)!$ the fact that the function $\delta(q_i, s)$ is defined.

The function $\Gamma : Q \to 2^E$ is excessive in a model definition but it simplifies many examination schemes and algorithms development when analysing the languages presented by finite state machines, e.g. consistency definition. $Q_m \subset Q$ is a subset of marked states – the states corresponding to a certain functionality of G, with one of them necessarily being initiated in a specific variant of G use.

The language generated by G machine is designated as $L(G) := \{s \in E^* : \delta(q_0, s)!\}$. This is a set of all strings from E^* admissible in the initial state q_0. It is evident that $L(G) \subseteq E^*$. If the machine is completely defined, $L(G) = E^*$. It G is represented by a weighed graph of transitions, $L(G)$ is presented as a set of strings of the events weighing the edges of all the paths originated from the initial state q_0.

When a sophisticated DES is defined via components, two more operations on machines are often applied: Cartesian product and parallel composition. Product definition

$$G_1 \times G_2 = (Q_1 \times Q_2,\ E_1 \cap E_2,\ \delta_{1,2},\ \Gamma_{1\times 2},\ Q_{m1} \times Q_{m2},\ (q_0 := q_{01,\,02}))$$

is conventional but there is one nuance: a function of transitions is defined on common events for every pair of states. Isolated pairs and those unattainable from the initial state are discarded together with their associated transitions. From the definition it follows that the language $L(G_1 \times G_2)$ of the Cartesian product of two machines is equal to $L(G_1) \cap L(G_2)$ – the intersection of these machines languages.

Parallel composition (or just composition, let it be designated as $\oplus$) is defined on the union of events of both machines $G_1 \oplus G_2 = (Q_1 \times Q_2,\ E_1 \cup E_2,\ \delta_{1,2},\ \Gamma_1 \oplus_2,\ Q_{m1} \oplus Q_{m2},\ (q_{01}, q_{02}))$. At this, it is possible that $E_1 \cap E_2 \neq \varnothing$, then on common events, transition synchronization takes place in both components. If the event is individual, transition takes place in one component (provided for this pair this event belongs to the value area of the corresponding function Γ).

Formally:

$\delta((q_1, q_2), e) = \{(\delta_1(q_1, e), \delta_2(q_2, e))$ if $e \in \Gamma_1(q_1) \cap \Gamma_2(q_2)$ | $(\delta_1(q_1, e), q_2)$ if $e \in \Gamma_1(q_1) \setminus E_2$ | $(q_1, \delta_2(q_2, e))$ if $e \in \Gamma_2(q_2) \setminus E_1$ | and indeterminate in other cases}.

It is obvious that both operations are associative and, provided parentheses are places accordingly, may be easily generalized for n machines: a product – $G = \bigtimes_1^n G_i = G_1 \times ... \times G_n$; a composition – $G = \bigoplus_1^n G^i = G^i \oplus ... \oplus G^n$.

The initial stage of object study (modelling) is dedicated to prognostication of possible physical behaviour of the entire object or its subsystems, i.e. consideration of possible actions and possible variants of behaviour in the absence of any control and restrictive actions. At this stage, DES is represented by machine G as a language $L(G)$ generator. Thus, G generates event sequences of any kind reflecting control-free DES behaviour. In order to specify and provide control in DES, a set of events E is subdivided into two disjoint subsets: E_c – a subset of controllable events corresponding to the commands and E_{uc} – a subset of uncontrollable events for which the moments they occur are unpredictable.

The present-day view on DES was first worded in (Ramadge, Wonham, 1987) though then the term "discrete event systems" was not used but a new technique of discrete process modelling and control was stated. The term "discrete event systems (DES)" appears already in (Ramadge, Wonham, 1989), where DES is represented by generator G of different sequences of events from E. G is limited by nothing and therefore the sequences reflect the behaviour $L(G) \subseteq E^*$ unbounded by control. Any DES has some functionality to implement which are required not all possible sequences but only those providing this functionality and meeting the limitations specified. In order only to provide the required event sequences, G is term "supplemented" by supervisor S, built-in a "feedback" manner (Fig. 2).

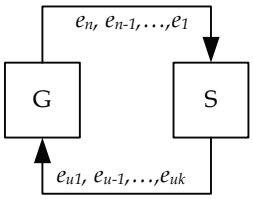

Fig. 2. The scheme of object – supervisor interaction

The scheme in Fig. 2 is no different from the conventional structure "control object – control system" but the behaviour is absolutely different. First, a generator event sequence covers all events in the system; second, a supervisor sequence includes only controlled events and third, controlled event e_k is incorporated into G output sequence conditioned to its presence also in S sequence. This allowed to define S transparently enough as a function of strings from the set $L(G)$: $S : L(G) \to 2^E$.

Supervisor S is equipped with a mechanism of G sequences blocking provided they do not meet limitations. For this purposes, S structure comprises one more component allowing for G "free" behavior restriction – a specification K. For the real object, a certain functionality (depending on G destination) must consider a multitude of all types of requirements and limitations $R = \{r_i \mid i=1,..,n\}$. As a rule, R is formed reasoning from physical, process and

design limitations imposed on joint behaviour of separate *G* components. The allowance for all restrictions *R* gives rise to $K \subseteq L(G)$ – a language of specifications – a subset of sequences dictated by *G* functionality. Actual control scheme stated in (Van Brussel et al., 1987) is presented in Fig. 3. It took the name of "Supervisory control theory" or RW approach (named after its authors J. Ramadge J. and W. Wonham W).

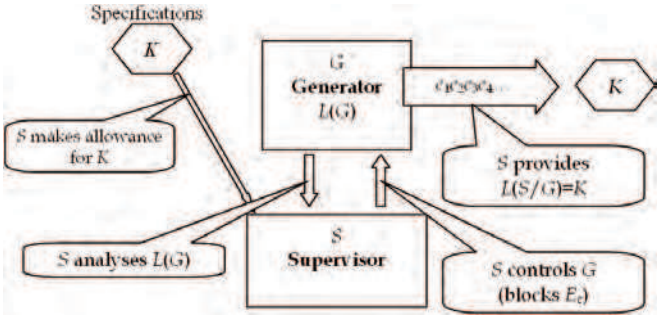

Fig. 3. Interrelationship of supervisory control components in DES

The functioning of *G* in the presence of *S* is denoted by *S/G* and a corresponding language – *L(S/G)*. The scheme symbolically shows that specification *K* is involved in *S* forming and in providing blocking. Supervisor is designed, with *K* taken into account, in such a way that, in accordance with *L(G)* observation results, *S* blocking mechanism provide the language *L(S/G)* = *K* at DES output. We would like briefly to dwell upon the way *L(S/G)* generation is realized. *G* is supposed to have its own controller that generates control events while a supervisor blocks the events the occurrence of which runs counter to the specification (Fig.4).

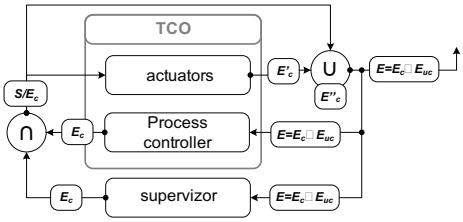

Fig. 4. Control scheme proposed in the paper (Ramadge, Wonham 1987)

Supervisor *S* monitors *G* output events and permits all E_{uc} events, while as to E_c events, it is "entitled" to permit or not permit them (to block by imposing limits on transition function $\delta(q_i, e_c) \coloneqq q_j$). For every string $s \in L(G)$ generated by *G* under *S* control, a supervisor only permits a set $\{S(s) \cap \Gamma(\delta(q_0, s))\}$ – a set of events admissible in *G* current state $\delta(q_0, s)$ and not conflicting with *K*. Hereinafter, $\delta(q_0, s)$ will mean a state *G* transfers to from q_0 as affected by *s*. In other words, *G* cannot realize the event from its current active event subset $\Gamma(\delta(q_0, s))$ unless this event is contained also in $S(s)$. However, making allowance for the fact that *E* is subdivided into **controllable** and **uncontrollable** subsets and the appearance of the latter is limited by nothing, supervisor *S* is called **admissible** if for all $s \in L(G)$, always $E_{uc} \cap \Gamma(\delta(x_0, s)) \subseteq S(s)$, i.e. *S* is specified in such a way that in all states it is impossible to block an

uncontrollable event and vice versa: *S* blocks the events not meeting limitations (irrelevant to *K*). Further on, only admissible supervisors will be considered.
For the modelling of DES with passive actuators in paper (Chalmers, Golaszewski, Ramadge, 1987) it is suggested that the model should be expanded with forced controllable events and a new control scheme (Fig. 5), with controllable events generated by supervisor, is developed. For such model, the terms of controllability for specification language are also defined.

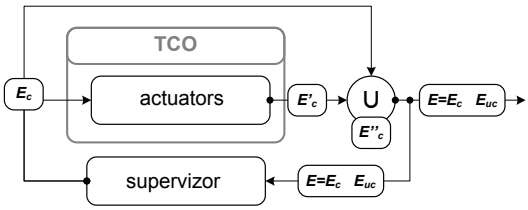

Fig. 5. Control scheme for DES with forced controllable events

For both models were developed the methods of supervisor synthesis as a finite state machine (FSM) with output converters regulating blocking (or generation) of E_c events. However, for the methods proposed the number of supervisor S states is less or equal to the product of the number of states for *G* and *K* (Cassandras, Lafortune, 2008).
DES dynamics is interpreted in the sense that the system (a pair of *G* and *S*), once set to the initial state, operates off-line, reacting to internal and external events, and provides a resulting flow relevant to *G* structure and *S* control.
Since 1987, there have been a lot of publications on DES subject-matter. At three last world IFAC Congresses, three sections on DES theory were working; IFAC Committee on DES theory was established; symposiums on this subject-matter are held. The paper scope limitation does not allow to survey the results on DES theory so we shall confine ourselves to listing the basic research trends. They are as follows:

- Study of DES as a dynamic system with a certain range of states and a structure of event transitions; the study of properties of the languages generating DES from the position of general control theory and the definition, in terms of language properties, of controllability, observability, attainability, safety (avoiding blocking situation) and some others;
- Study of different models of *G* and *K* specification (finite state machines, Petry nets etc) and the development of synthesis (engineering) methods for supervisor *S* on *G* and *K*;
- Assessment of supervisor complexity at synthesis with FSM models of *G* and *K* involved;
- Study of different modular presentations of supervisor *S* in the form of parallel generators of sub-languages with their subsequent combining via product operation (conjunctive scheme), via parallel composition operation (disjunctive scheme) and others;
- Development of programming methods for logical controllers in industrial systems with supervisor control theory applied;
- Creation of program verification methods for industrial systems with DES, as simulation instrument, applied;
- Development of the methods of industrial system state diagnostics using DES as a modelling instrument.

A detailed survey of the results obtained on DES can be found in (Cassandras, Lafortune, 2008); herein the major results on controllability from (Ramadge, Wonham, 1987; Ramadge, Wonham, 1989) are set forth:

- Is formulated the condition of controllability for the language: $K \subseteq L(G)$ is controllable if $\overline{K}E_{uc} \cap L(G) \subseteq \overline{K}$
- It is proved that if K is controllable, there exists a non-blocked S such that $L(S/G) = K$
- Are developed the methods to design supervisor S as a function of strings (Ramadge, Wonham, 1987; Cassandras, Lafortune, 2008).

However, the direct practical application of the proposed models and methods is confined to lab examples of dynamic DES engineering and supervisor synthesis. Such constraint is explained by high dimensionality of the object states set. To analyze for controllability, a complete DES specification of generator G is required. Even in the simple example given here below (a machine with four mechanisms) the number of states equals 4356. (The number can be considerably reduced with DES structural features taken into account).

Main direction of works focused on overcoming supervisor synthesis complexity is based on different kind of modularity. Methods of modular supervisor synthesis for G, as a single entity, are elaborated. At this, different control schemes are explored (disjunctive, conjunctive, hierarchical, generalized). Pioneer work (Ramadge, Wonham, 1989) that initiated the development of modularity, as applied to DES theory, was evolved and generalized in (Yoo, Lafortune, 2002). Later, different authors (De Queiroz, Cury, 2000; Gaudin, Marchand, 2003) developed the methods of modular supervisor synthesis on modular description $G=<G_1, G_2, \ldots, G_n>$ and modular specification $K=<K_1, K_2, \ldots, K_n>$ of modular S. However, the complexity of such synthesis and weak correspondence of the initial specification structure to the resulting supervisor make the methods proposed scantily attractive for practical implementation. Besides, controllability properties are verified on language models K and $L(G)$ defined for the object (Plant) as a whole, which makes it difficult to apply these results to real industrial facilities.

The present paper sets the task to develop a prototype of structured dynamic DES by structuring the object components according to their functionality. To operate the model, the paper proposes the methods that will allow to raise the dimension of supervisor control tasks and form a theoretical basis for a new supervisor control engineering technique. Structured are all three DES components but mainly object model and specification.

3. Structured Discrete Event Systems (SDES)

3.1 Base concept – the structuring of events and specifications

The author considers it promising to develop a supervisory control theory in the direction of structuring the events according to their role in production operations and in the required object behaviour specification. This research is based on two specific machinery features from DES-modelling point of view. The first feature relates to the fact that for discrete machinery a set of events is usually subdivided into three sets. These are sets of controllable and uncontrollable events E_c and E_{uc} (typical for DES theory) and E_w is a set of expected events. The events from E_w simulate states (positions) of actuator(s) or object components. Supervisor cannot block E_w events as those controllable from E_c and thus E_w events are traditionally referred to uncontrollable events as per Wonham's classification (Ramadge and Wonham, 1987). However, E_w events are expected to occur as a response to E_c events – a

confirmation of the fact that the commands sent to actuators were executed. So, the foregoing gives the ground to mark out E_w events as a separate set. The second specific feature is as follows: the behaviour of every actuator G^i is simulated by the language $L(G^i)$ of words over $E^i = \{E^i_w \cup E^i_c\}$ and the specification of desired behaviour is formulated as a language K over events $E_d = E_c \cup E_{uc}$, a totality of commands and conditions of their use. Making the allowance for these specifics, makes it possible to get numerous advantages both in defining DES and formulating controllability conditions and supervisor synthesis.

3.2 SDES definition

Definition 1: If the structure of DES is defined by: a collection of components $G=<G^1,G^2,\ldots,G^n>$; sets of E_i events, each being structured on $E^i = \{E^i_w \cup E^i_c\}$, and a set E_{uc} of general uncontrolled events; the behaviour of each DES component being defined by FSM $G^i = \left\langle Q^i, E^i, \delta_i, \Gamma^i, Q^i_m, q^i_0 \right\rangle$ and $L(G^i)$ language, then the DES with the above structure is called well structured.

A set of common events for $G=<G^1,G^2,\ldots,G^n>$ is defined through the union of subsets $E = \{E_w \cup E_c \cup E_{uc}\}$, where E_w and E_c each are the unions of appropriate component subsets.

Note 1: Sets E_w and E_c for various mechanisms do not intersect, since various mechanisms have their own actuators and their states are individual.

Note 2: Components of G^i define the behaviour of G that is not limited (controllable) by anything, e.g. from the successive operation of $<G^1,G^2,\ldots,G^n>$ in any order up to their independent work in parallel.

According to the theory of supervisory control, a parallel composition of all object components is implemented, and, as the result, a model of uncontrollable object behaviour is created (Ramadge & Wonham, 1987). The narrowing of free behaviour is carried out with the constraints of purposeful joint behaviour considered. This, in essence, is the procedure of adapting the initial unlimited behaviour i.e specifying the behaviour as required by application. We would like to remind that the implementation of all restrictions generates a language $K \subseteq L(G)$ called a language of specifications. Establishing the restrictions is a creative process that requires an experimental approach to achieve a reliable result. Such experiment is quite difficult to carry out as the number of states is increasing in the course of composing. There is a collision.

On the one hand, a system analyst needs to get a general picture of all the transitions to analyze their admissibility.

On the other hand, it is unreal to do it for complete composition, since the number of states in it is too high (for practical applications this number is about $n \cdot 103$). Sequent revealing of restrictions in the process of pair-wise composing, gives a ground to doubt of such restrictions completeness or, on the contrary, of their extreme strictness. At the same time, there is no possibility to consider the joint action of components with those absent in the composition.

At the same time, it is known from the practice of discrete process engineering that the efficient behaviour of discrete systems is achieved by solving two control tasks, namely: operation control and control of operation sequence. Operation control is provided by the execution of a certain command and monitoring the corresponding object response. Commands and their reactions once defined, are iterated in various places of the sequence of operations. In process modelling, it is important to set up the sequence of commands and

to evaluate the completeness and correctness of conditions. With the above in view, herein is proposed to create a specification of a well-structured DES with the events $E_d = \{E_c \cup E_{uc}\}$, i.e. combination of commands and conditions for their execution in sequence.

Definition 2: The language $K \subseteq E_d^*$ defined by FSM $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ as a set of strings defining the required specifications, is called a directive specification language (a process specification tapes language).

It is assumed that FSM *H* has no deadlocks (Fig. 6) and livelocks (liveloops, within which *H* fails to go out of a certain state subset and does not reach Q_m and then q_0), i.e. *H* is non-blocking.

It is worthy to be noted that if a graph is strongly connected and $q_0 \notin Q_m$, then q_0 transitions only as shown in Fig. 6 are possible.

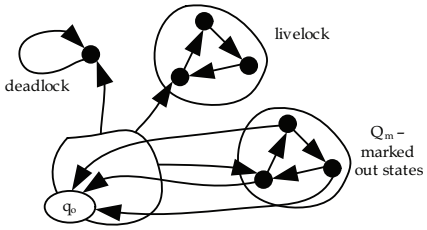

Fig. 6. Types of fragments in the machine *H*

The fact of non-blocking is easily verified. Contrary to the general DES theory (Cassandras, Lafortune, 2008), where deadlocks and livelocks result from the excessive general description via the product and composition, in SDES, there should be no hurry in cutting down "bad" states and transitions but, vice versa, it is necessary to check if any transition is missed to avoid deadlock or livelock situations.

Let's define a supervisor for *G* and *K*. It is conceptually evident, that supervisor is an operator that defins, for every string s, which of possible events, admissible for *G*, are suitable as the next event not conflicting with *K*. At this, supervisor remains admissible in terms of (Van Brussel et al., 1998) since it in no way limits E_{uc} occurrence and affects only E_c.

Definition 3. Supervisor *S* is a converter of strings admissible for the system $G = \langle G^1, G^2, ..., G^n \rangle$ initial state to the events $S(s) = \{\varepsilon \cup E_{uc} \cup \{e_c\}\}$ such that: first, these are any of uncontrollable events E_{uc} (i.e. S is admissible for *G*); second, these are controllable events e_c admissible for the current *G* state; third, these events do not cause blocking of *S* and $G = \langle G^1, G^2, ..., G^n \rangle$ composition.

Let's denote, as agreed, by *L(S/G)* the language generating *G* under *S* control. It is evident that $L(S/G) \subseteq L(G)$. Let's also give a definition of *L(S/G)* language generating *S/G*, that is consistent to the conventional definition of language generating *G* under *S* control.

Definition 4. The language *L(G/S)* generating $G = \langle G^1, G^2, ..., G^n \rangle$ under *S* control contains the following strings:

1. $\varepsilon \in L(S/G)$;
2. $\forall s, e(s \in L(S/G) \wedge e \in S(s)) : se \in L(G) \Leftrightarrow se \in L(S/G)$

In other words, any string *se* belongs to *L(S/G)* provided it also belongs to *L(G)* being at the same time the extension of string *s* which also enters *L(S/G)* by event e such that $e \in S(s)$. Possibly, $s = \varepsilon$.

Definition 5. A well-structured DES, for which the uncontrollable part is set up by definition 1, the desired behaviour is set by specification language $K \subseteq E_{d^*}$ $(K \neq \emptyset)$, and which is supplied with a supervisor S such that K is fulfilled, is called a structured dynamic discrete event system (SDES).

K fulfilment means that $P_{E_d}(L(S/G)) = K$, i.e. that K will be equivalent to the projection on E_d of $L(S/G)$ language that is generated by S/G.

3.3 Technical object modelling by structured DES

The events associated with real industrial objects, as a rule, are easily divided into groups (types) as proposed herein. Such event grouping is typical for process systems of many industrial spheres. Here below is the example which refers to the field of mechanical metal-working. We consider this example most interesting since it is close to illustrative examples frequently used in publications on DES (Ramadge, Wonham, 1987; Ramadge, Wonham, 1989; Chalmers, Golaszewski, Ramadge, 1987; Ambartsumyan, 2009).

The structuring of technical object (the first phase of study) includes as follows:

- enumerating actuators;
- defining for each of them the set of events necessary and sufficient for the outer supervisor to identify actuators behaviour;
- defining the classification of marked out events;
- defining the components and object behaviour in the compact-form languages, e.g. finite machine models.

In Fig. 7 a kinematical scheme of a small milling machine is presented. The machine consists of 4 mechanisms: "workpiece clutch" - G^1, "turntable" – G^2, "spindle" – G^3 and "cutter" – G^4

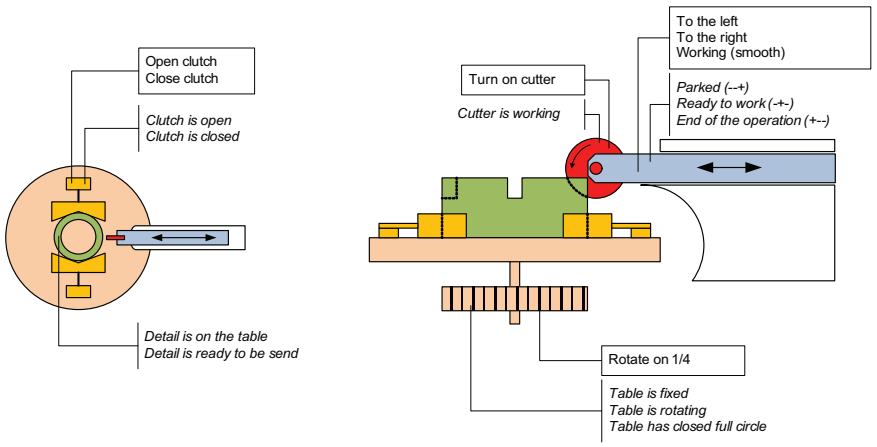

Fig. 7. Kinematical model of the machine

Let's enumerate the events and their semantics in the liveloop (behaviour) of each mechanism.

"Workpiece clutch" mechanism: $e_{1\text{-}1}$ – to clamp, $e_{1\text{-}2}$ – clutch closed, $e_{1\text{-}3}$ – to unclamp, $e_{1\text{-}4}$ – clutch closed, $e_{1\text{-}5}$ – clutch is moving.
"Turntable" mechanism: $e_{2\text{-}1}$ – to lock the table, $e_{2\text{-}2}$ – table locked, $e_{2\text{-}3}$ – to unlock the table, $e_{2\text{-}4}$ – table unlocked, $e_{2\text{-}5}$ – locker is moving, $e_{2\text{-}6}$ – to make a ¼ turn, $e_{2\text{-}7}$ – table is moving, $e_{2\text{-}8}$ – table is turned, $e_{2\text{-}9}$ – to switch off turning gear, $e_{2\text{-}10}$ – table stopped.
"Spindle" mechanism: $e_{3\text{-}1}$ – to move spindle fast to the left, $e_{3\text{-}2}$ – feed zone, $e_{3\text{-}3}$ – working position, $e_{3\text{-}4}$ – to move spindle to the left, $e_{3\text{-}5}$ – working zone, $e_{3\text{-}6}$ – operation finished, $e_{3\text{-}7}$ – to move spindle to the right, $e_{3\text{-}8}$ – to move spindle fast to the right, $e_{3\text{-}9}$ – parked.
"Cutter" mechanism: $e_{4\text{-}1}$ – to turn on cutter, $e_{4\text{-}2}$ – cutter working, $e_{4\text{-}3}$ – to turn off cutter, $e_{4\text{-}4}$ – cutter stopped, $e_{4\text{-}6}$ – cutter unstable spinning.
Mechanisms behaviour, as agreed here above, will be considered as sequences (strings) of possible events. These sequences will be defined as finite state machines (Fig. 8–11). Hereinafter they are called component finite machines (CFM). It is easily seen that CFM transition graphs and graph edges weighed by events, specify operation of each mechanism quite transparently.

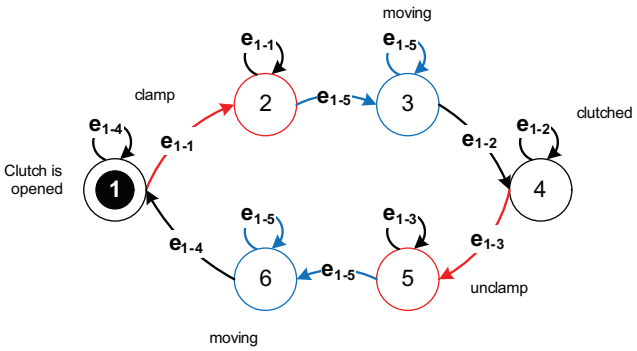

Fig. 8. G^1 CFM – a model of "Workpiece clutch" mechanism

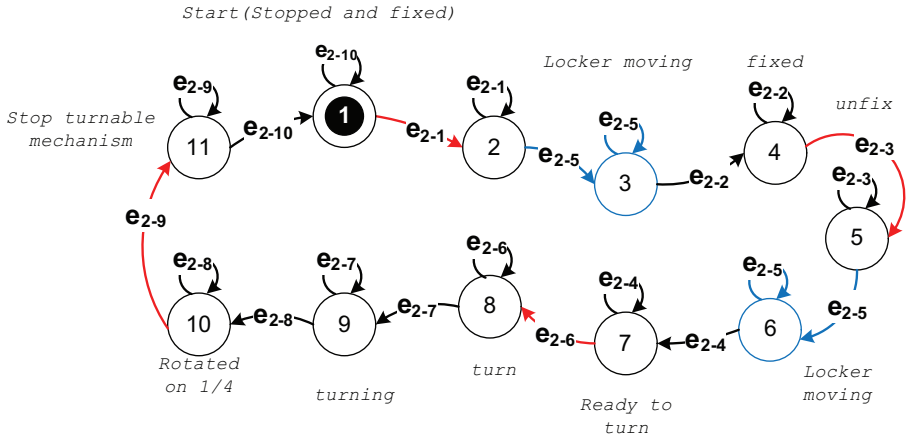

Fig. 9. G^2 CFM – a model of "Turntable" mechanism

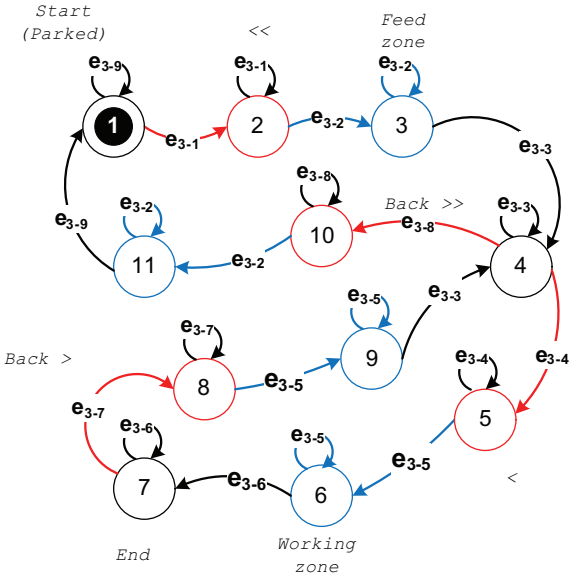

Fig. 10. G^3 CFM – a model of "Spindle" mechanism

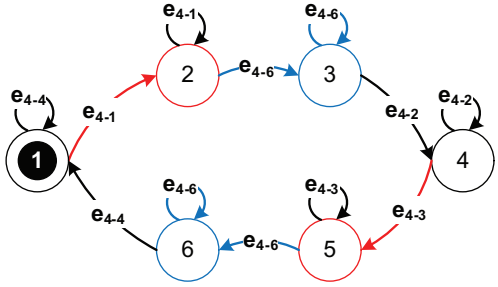

Fig. 11. G^4 CFM – a model of "Cutter" mechanism

It is easy to make natural event grouping in all the CFM, namely:
G^1 – $E_c^1 = \{e_{1-1}, e_{1-3}\}$, $E_w^1 = \{e_{1-2}, e_{1-4}, e_{1-5}\}$; G^2 – $E_c^2 = \{e_{2-1}, e_{2-3}, e_{2-6}, e_{2-9}\}$,
$E_w^2 = \{e_{2-2}, e_{2-4}, e_{2-5}, e_{2-7} e_{2-8}, e_{2-10}\}$; G^3 – $E_c^3 = \{e_{3-1}, e_{3-4}, e_{3-7}, e_{3-8}\}$,
$E_w^3 = \{e_{3-2}, e_{3-3}, e_{3-5}, e_{3-6}, e_{3-9}\}$; G^4 – $E_c^4 = \{e_{4-1}, e_{4-3}\}$, $E_w^4 = \{e_{4-2}, e_{4-4}, e_{4-6}\}$ and to see the events $E_{uc} = \{e_{ex-1}, e_{ex-2}, e_{ex-3}, e_{ex-4}, e_{ex-s}, e_{ex-w}\}$ common for all components (respectively: a workpiece is on the table; a workpiece is removed from the table; processing is over, clutch of s type , clutch of w type).

Note 3. Sets E_w and E_c for different mechanisms do not intersect.
It is evident, since different mechanisms have their own drivers and their positions for each mechanism are individual.
The next stage of a technical system SDES-modelling is the defining of the system behaviour specification based on the requirements to the system functionality and limitations. It is

done by forming the behaviour of G as an uncontrollable system, as a whole, followed by putting in limitations, thus "narrowing" G behaviour up to that required.
The traditional approach being applied, uncontrollable G behaviour is defined by component machines combination. Let's use two mechanisms of the above milling machine (Turntable and workpiece Clutch) to illustrate this.

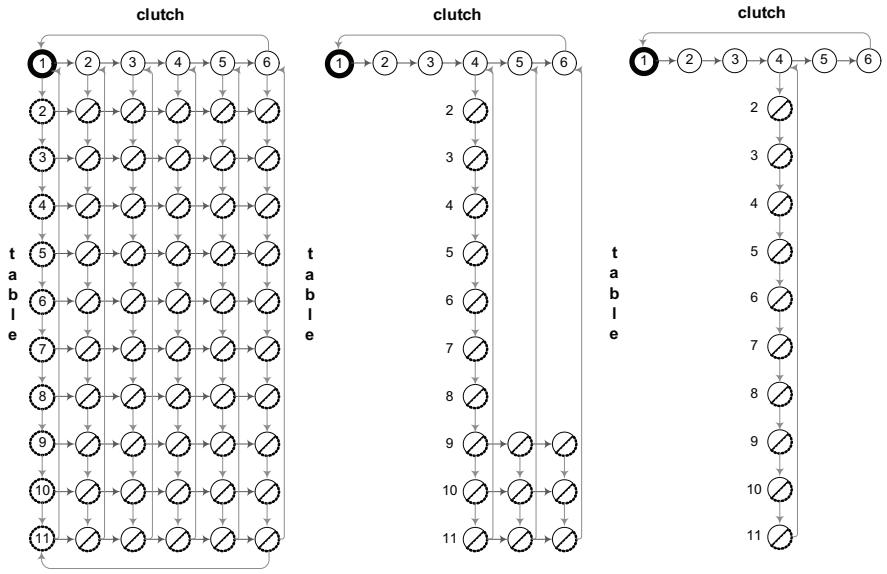

Fig. 12. CFM composition for G^1 and G^2: a) complete; b) with allowance for limitations r_1 and r_2; c) with allowance for limitations r_1, r_2, r_3

Pursuant to SC theory, we should make a composition of all machines to achieve "uncontrollable" G behaviour. DES, modelling "uncontrollable" behaviour of the first two mechanisms, is represented by $G^1 \oplus G^2$ composition, with relevant transition graph structure illustrated in Fig. 12-a. Here a structure of initial components transitions is shown: across - G^1 structure, down - G^2 structure, and relevant pairs are represented by nodes at arrows intersection. Edges weighing corresponds to weighing of transitions in the initial components.
Machine $G^{1\otimes 2}$ represents unlimited by anything, parallel operation of mechanisms G^1 and G^2 originating $L(G^{1\otimes 2})$ language.
In our example, the following restrictions as to joint behaviour of the mechanisms take place: r_1: "turning of G^2 "Turntable" mechanism is possible if a workpiece is clutched"; r_2: "if in the course of the table turning a workpiece unclasping begins , "Turntable" will only terminate turning".
The implementation of these technological restrictions are formally realized by banning the following state compositions: 1, 2, 3 of G_1 CFM and 2-9 of G_2 CFM. With these limitations applied, all pairs of states under verticals 1, 2, 3 and a number of pairs under verticals 5, 6 are excluded (Fig. 12-b). The same refers to their incident transitions. As the result, we get the machine K_1 as shown in Fig. 12-b. More detailed analysis of admissible transitions results in the necessity of one more limitation: r_3 – "at table turning, a workpiece unclasping is inadmissible", which makes specification more strict (K_2) as shown in Fig. 12-c.

Thus, we have DES of $G^{1\otimes 2}$ and it's necessary to provide its operation within the framework of language *K*. In what way is it possible to regulate a path choice in $G^{1\otimes 2}$ graph? In our example, for $G^{1\otimes 2}$ $E_c^{1\otimes 2} = \{e_{1-1}, e_{1\text{-}3}, e_{2-1}, e_{2\text{-}3}, e_{2-6}, e_{2-9}\}$, $E_w^{1\otimes 2} = \{e_{1-2}, e_{1\text{-}4}, e_{1-5}, e_{2-2}, e_{2\text{-}4}, e_{2-5}, e_{2-7} e_{2-8}, e_{2-10}\}$.

Graph transition trajectory can be regulated by a function of transitions $G_{1\otimes 2}$ by blocking or accepting the events from E_c set with the help of supervisor *S* (outer to *G*) which dynamically interacts with *G* in a feedback manner. The way it can be realized is illustrated by our example. In state, $q_{1,4}$ in cycles 1, 2 and 3 of the table operation, a supervisor each time enables $e_{2\text{-}1}$ and disables $e_{1\text{-}4}$, and, after the table returns to its initial position for the 4-th time, it is $e_{1\text{-}4}$ that is admitted and $e_{2\text{-}1}$ that is banned.

So, CFM sequential merging and the detection of limitations for CFM joint operation are quite a complicated procedure even in our case. We have already noted that the detection of limitations in the course of pairwise component combination, gives the ground to doubt about the completeness of such limitations or vice versa in their excessive strictness. Besides, there is no possibility to predict the consequences of joint operation with the components still absent in the composition. For example, should we start CFM merging with "Spindle" and "Turntable" mechanisms, it will in no way possible to make allowance for the fact that between their "activities" a locker actuation will take place.

At the same time, for technical objects, their required behaviour is always defined by their functionality that is specified, for example, by text description. The required machine behaviour is presented by informal specification in table 1.

1) on arrival, the piece is locked by clutch; 2) after clenching, the spindle moves from park position to work position (to the left); 3) the cutter is switched on; 4) smooth feed to the left utmost position (operation is over); 5) the spindle moves to the right back to work position;	6) positioner makes a ¼ table rotation; 7) after the table is fixed, the next operation is carried; 8) after the table makes a turnover, the spindle is parked, the clutch is unclamped, the signal of the piece readiness is sent; 9) prior to parking, to switch off the cutter and wait for a stop.

Table 1. Text description of initial specification

At SDES-modelling, at this stage, a specification of joint behaviour in *K* language is applied. A specification, compliant with the text specification, is presented by machine $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ shown in Fig. 13.

Since the verbal behaviour description, as a rule, is inaccurate, the resulting specifications may vary. The example of another interpretation of verbal description is presented in Fig. 14. The specification is described in conformity with verbal description. Basing on the information from table 1, it is possible to assume that at the beginning of operation, the table is fixed, since otherwise is not specified and thus, the operation relevant to the transition graph node 3 is omitted. However, should the order of operations as shown in Fig. 14 be accepted, already the processing of the second workpiece will start with the table unfixed since in the beginning of the large loop locker is not considered. The necessary operation is missing.

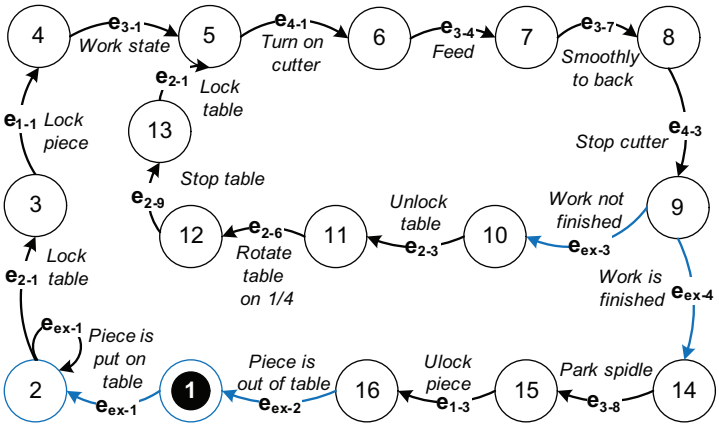

Fig. 13. The required machine behaviour in terms of directive specifications. The semantics is as follows: $e_{ex\text{-}1}$ – a workpiece is on the table, $e_{ex\text{-}2}$ – a workpiece is removed from the table, $e_{ex\text{-}3}$ – processing is not over, $e_{ex\text{-}4}$ – processing is over (other events semantics was given here above in the mechanisms description).

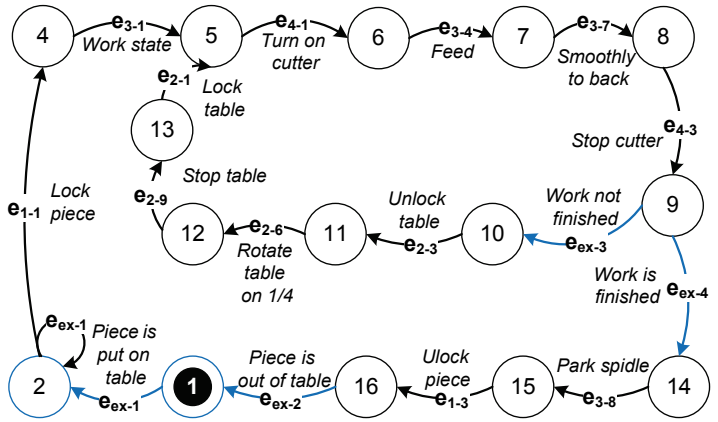

Fig. 14. *H* specification with erroneous missing of "Table locking" operation

Operation omission is far from being the only inconsistency in the required behaviour specification. Here below (Fig. 15) another text description interpretation is given. The specification is elaborated in accordance with the text but a "cutter halt" operation (node 8 of Fig. 15) is performed prior to cutter parking in the "large" loop, which follows from item 9 of the text description from Table 1. Cutter halt is performed in the "large" loop but on the processing termination, therefore, while processing the second piece position, the attempt will be made to switch on a working cutter.

Note3. The composition of modular hierarchic DES description of solely unblocked modules may result in DES blocked operation.

This stage of SDES-modelling reveals a principle difference of discrete control engineering with supervisor *S* on G and *K* given, as compared with a "black box" technique.

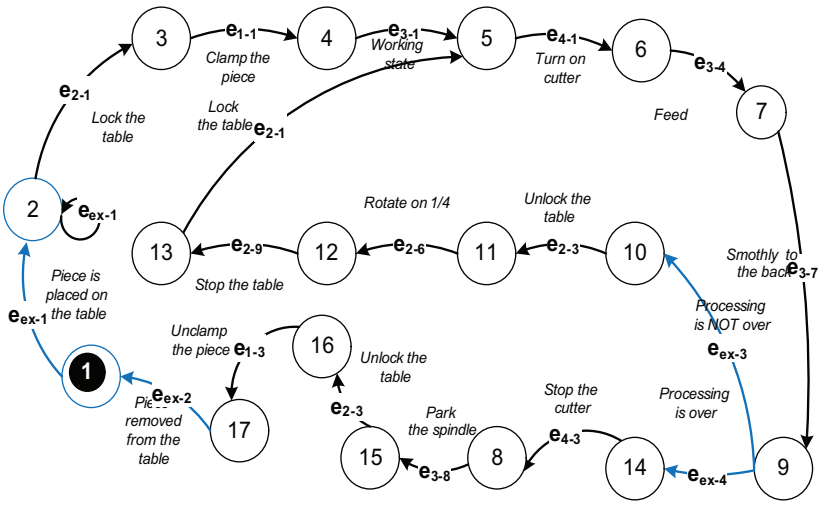

Fig. 15. Machine behaviour as described in the language of directive specifications, with a "Cutter halt" operation moved to the large loop

Indeed, if we make quite a transparent substitution of CFM operations in the transition graph of *H* specification and properly apply the functions of outputs (to be shown here below), we shall get a controlling finite state machine. This machine, provided inputs are independent (this being an indispensable condition for conventional logical control according to the "black box" scheme), will precisely perform the operation sequences specified. Note that substitutions can be made for each of three specifications and, thus, three different controlling machines will be obtained. Later on, it will be possible to carry out arbitrarily profound optimization applying all the methods used in the finite machine theory and logical synthesis. However, at the attempt to unite a control object and $G = \langle G^1, G^2, ..., G^n \rangle$ machines, obtained as per specifications presented in Fig. 15, 16, the errors, mentioned here before, will reveal themselves in blocking (non-fulfilment) of some commands and a "hanging" – an unforeseen cyclic operation interruption will occur. At the same time, with DES theory analytic methods applied, possible blocking situation will be revealed analytically. It is evident that once DES theory methods are applied, a "dimension damnation" will manifest itself: CFM parallel composition of the example in question already gives a machine with the number of states equal to 4356 and its composition with *H* machine results in the machine with dozens of thousands states.

So, we face the following problem: how to predict blocking situation without composition of G_i in *G* followed by general composition with *K*. To tackle this problem, let's continue considering the theory of SDES-modelling.

4. Features of the models of *G* components and H specification

We would like to point out a number of important features of the models of $G = \langle G^1, G^2, ..., G^n \rangle$ components and specifications of industrial objects. Model components, as a rule, simulate the behaviour of different actuators able to "perceive" events-commands,