ISSN 2347 - 3983
Volume 8. No. 10, October 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter1278102020.pdf
https://doi.org/10.30534/ijeter/2020/1278102020

Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis

Igor Nevliudov[1], Murad Omarov[2], Vladyslav Yevsieiev[3], Artem Bronnikov[4], Vyacheslav Lyashenko[5]
[1]Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of RadioElectronics, Ukraine, igor.nevliudov@nure.ua
[2]Vice-Rector on International Cooperation, Kharkiv National University of RadioElectronics, Ukraine, murad.omarov@nure.ua
[3]Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of RadioElectronics, Ukraine, vladyslav.yevsieiev@nure.ua
[4]Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of RadioElectronics, Ukraine, artem.bronnikov@nure.ua
[5]Open Digital Group LVV, Kharkiv, Ukraine, lyashenko.vyacheslav@gmail.com

ABSTRACT

The management process of the new cyber-physical systems (CPPS) development is a complex task that requires the combination of physical and cybernetic components. Now there are no architectural solutions and approaches that allow to automate the management process of CPPS development based on the "end-to-end development" concepts, which will reduce the time and development cost. This article proposes a method for functioning algorithms synthesis. As a result of manipulations, the developer can get many options for algorithms synthesis into one, and the most optimal, according to the requirements that need to be achieved. This method can be implemented as a software in the form of an automation system for the CPPS development control process.

Key words: Industry 4.0, Cyber-Physical Production Systems, Digital Twins, Functioning Algorithms.

1. INTRODUCTION

Modern high-tech manufacturing imposes new requirements, whose implementation is not possible without the cyber-physical production systems (CPPS) use within the framework of Industry 4.0 concepts [1]-[4]. The development of a production system that takes into account all the factors of interaction between the physical and cybernetic CPPS components based on the concepts of "Digital Twins" will allow to fully automate the production process, which will ensure the Lean Production (LP) goals achievement [5]-[8].

Publications analysis showed that at the moment the proposed the CPPS architecture reference models, such as ISO-95, 5C, 8C and RAMI 4.0, do not have a single mathematical apparatus, but represent a set of general recommendations, which does not make it possible to automate the process of their development management [9]-[14].

In the publication [15], an architectural and logical model of the complex CPPS development management process automation was proposed, which, in contrast to the existing ones, is a “rigidly” hierarchical structure of logically interrelated mathematical concepts that makes it possible to implement an approach to automate the end-to-end CPPS development management process. During the experiments, it was revealed that this model and methods make it possible to obtain an algorithm for the functioning of each decomposition level, and not a general algorithm for the CPPS functioning, as a consequence of this, in order to solve this problem, it is necessary to develop a new synthesis method that will allow to combine all decomposition levels into a common algorithm for the CPPS functioning, which will automate its development management process.

2. THE DEVELOPMENT OF CPPS OPERATING ALGORITHMS GROUPING AND SYNTHESIS METHOD

2.1 CPPS development method grouping

The proposed architectural and logical model of the complex CPPS development management process automation analysis [8] showed some patterns: the possibility of using one method or another at different CPPS development levels and stages. As a result, it became necessary to develop a method for CPPS

functioning algorithms synthesis. In order to achieve this, it is necessary to minimize the number of executed operators, provided that the synthesized algorithm will meet all the conditions of the technical task completely. The proposed method will allow to optimize the CPPS functioning algorithm and achieve economic effect solving problems of management processes for the complex cyber-physical systems development automation.

To do this, it is proposed to group the proposed methods as follows:

- group the goal decomposition methods (Aim_i), tasks ($Task_j$), structures, algorithms of functioning (*AF*) top level to bottom level elements;

- group the methods: by operators ($Op_h \_ AF$), by tasks $Task_j$ and object structures (*InputCanal* , *OutCanal*).

Applying the method of transformation can be seen following pattern: goals are transformed to tasks into the task structure, the structure into a communication channel, communication into the object operation algorithm;

- group the methods for calculating the tactical and technical characteristics (*PCofI*) Aim_i , $Task_j$,structures, *InputCanal* , *OutCanal* , Op_h ;

- group the methods of a system model, target, functional, infological, informational and functioning algorithm development;

- group analytical methods of design solutions strictly according to design stages.

Each CPPS and its constituent parts differ in properties, but during the study, the presence of repetitions was highlighted, therefore it is possible to assert the existence of isomorphisms:

$$Aim_i \cong Task_j \cong StrE_q \cong IC_v \cong Op_h$$

This makes it possible to highlight elements, groups, subsystems and refer them to objects of system design.

2.2 Method for CPPS structural system models representing

Let introduce the following definitions: the system model is a graphical representation of the CPPS, where objects corresponding to the stages and levels of design can be represented as nodes, and edges as channels of connections between them.

Based on the proposed decomposition method [8], the representation of system models can be written in the following form:

$$Aim_i \_ MS_0' = \Omega Aim_i \_ MS_0'' \quad (1)$$

where: $Aim_i \_ MS_0'$ – the main goal of CPPS design in accordance with the technical specifications; $Aim_i \_ MS_0''$ – goals at the level of decomposition MS_0'' .

Based on 1, you can imagine the decomposition $Aim_i \_ MS_0''$ on sub-goals $Sub\_S_k \_ MS_0''$:

$$Aim_i \_ MS_0'' = \Omega Aim_i \_ Sub\_S_k \_ MS_0'' \quad (2)$$

where $Aim_i \_ Sub\_S_k \_ MS_0''$ – goals at the decomposition level $Sub\_S_k \_ MS_0''$.

Thus, the goals of decomposition identities development are constructed throughout the tree:

$$Aim_i \_ Sub\_S_k \_ MS_0'' = \Omega Aim_i \_ G\{AEofS\}_j \_ MS_0'' , \quad (3)$$

where $Aim_i \_ G\{AEofS\}_j \_ MS_0''$ – goal for a group of atomic elements $G\{AEofS\}_j \_ MS_0''$.

Similarly (2-3), breaking the goals of an atomic element can be written as:

$$\begin{aligned} &Aim_i \_ G\{AEofS\}_j \_ MS_0'' = \\ &= \Omega Aim_i \_ AEofS_i \_ MS_0'' \end{aligned} \quad (4)$$

where: $Aim_i \_ AEofS_i \_ MS_0''$ – goal of an atomic element $AEofS_i$;

From 1-4, it can be seen that the main goal of CPPS design is a consequence of achieving goals at each decomposition level at each stage of development. Therefore, it can be argued about the existence of "inheritance" ($\rightarrow$) goals by the principle of decomposition "top-down" (5).

$$\begin{aligned} &Aim_i \_ MS_0' \rightarrow Aim_i \_ MS_0'' \rightarrow \\ &Aim_i \_ Sub\_S_k \_ MS_0'' \rightarrow \\ &\rightarrow Aim_i \_ G\{AEofS\}_j \_ MS'' \rightarrow \\ &Aim_i \_ AEofS_i \_ MS_0'' \end{aligned} \quad (5)$$

Based on expression 5, it is possible to construct a graph for achieving the main goal of CPPS design, taking into account its multilevelness (Figure 1), which allows to take into account the impact of each other goals achievement within one level of decomposition.

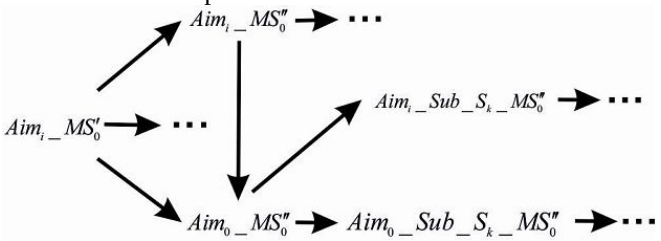

Figure 1: The CPPS design main goal achievement graph.

Similarly, you can build a graph of the functional level. In this case:

$$Aim_i\_MS_0' = \Omega Task_j\_MS_0'' \tag{6}$$

where $Aim_i\_MS_0'$ – the main goal of CPPS design; $Task_j\_MS_0''$ – tasks of MS_0'' level to reach $Aim_i\_MS_0'$.

Define the nodes of the graph as *Task* at each level of decomposition, and by the edges of the graph the connection between tasks to achieve the main $Task_j$ at this level. This makes it possible to fix connections between tasks using the task counter:

$$\begin{aligned} &Task_j \to Task_{j+1}; Task_j \to Task_{j+2}; \\ &Task_{j+1} \to Task_{j+3}; Task_{j+n-1} \to Task_{j+n} \end{aligned} \tag{7}$$

The proposed "heritage" allows to get a subgraph $Task_j$ representation model, an example of which is shown in Figure 2.

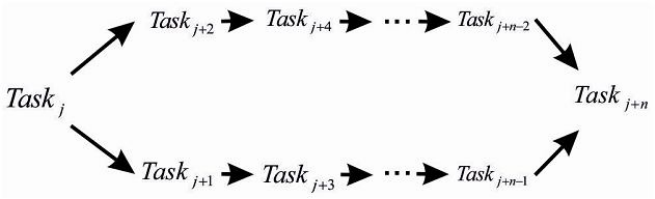

Figure 2: Subgraph $Task_j$ representation model.

Based on the subgraph $Task_j$ representation model, similarly, we can decompose the next sublevel $Sub\_S_k\_MS_0''$. Then, for this sublevel, the record is identical to:

$$Task_j\_MS_0'' = \Omega Task_j\_Sub\_S_k\_MS_0'' \tag{8}$$

where: $Task_j\_Sub\_S_k\_MS_0''$ – are the tasks at the functional stage of the level Sub_S_k.

Substituting the subgraph (Figure 2) into the mathematical representation 8, you can get the decomposition graph (Figure 3.) of the main MS_0' level problem to Sub_S_k level.

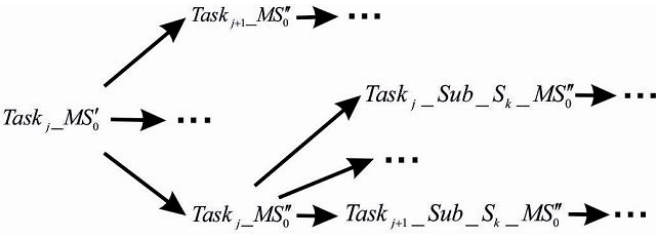

Figure 3: Functional stage graph decompositions to Sub_S_k level.

Similarly to expressions 6 and 8, can derive expressions for all levels of decomposition at the functional stage:

$$\begin{aligned} &StrE_q \to StrE_{q+1}; StrE_q \to StrE_{q+2}; \\ &StrE_{q+1} \to StrE_{q+3}; StrE_{q+n-1} \to SrtE_{q+n} \end{aligned} \tag{9}$$

To describe the system infological model, let us take the form of a graph node IC_v [8], and the edges of the graph will be *InputCanal* and *OutCanal* of connections. Based on the developed methodology for constructing a system model Aim_i, we describe the systemic infological model of the level MS_0'':

$$\begin{aligned} &I^{MS}\_MS_0' = \Omega IC_v\_MS_0'', \\ &InputCanal\_IC_v\_MS_0'', OutCanal\_IC_v\_MS_0'' \end{aligned} \tag{10}$$

where: $I^{MS}\_MS_0'$ – infological model of the MS_0'' system level; $IC_v\_MS_0''$ – information transformer at the MS_0'' level; $InputCanal\_IC_v\_MS_0''$ – input channels at the MS_0'' level; $OutCanal\_IC_v\_MS_0''$ – output channels at the MS_0'' level.

Similarly to expression 10 can present a systemic model of the infological stage at the following subsystem levels:

- system model of the $I^{MS}\_MS_0''$ level:

$$\begin{aligned} &I^{MS}\_MS_0'' = \Omega IC_v\_Sub\_S_k, \\ &InputCanal\_IC_v\_Sub\_S_k, \\ &OutCanal\_IC_v\_Sub\_S_k \end{aligned} \tag{11}$$

- system model of the $I^{MS}\_Sub\_S_k$ level:

$$\begin{aligned} &I^{MS}\_Sub\_S_k = \Omega IC_v\_G\{AEofS\}_j, \\ &InputCanal\_IC_v\_G\{AEofS\}_j, \\ &OutCanal\_IC_v\_G\{AEofS\}_j \end{aligned} \tag{12}$$

- system model of the $I^{MS}\_G\{AEofS\}_j$ level:

$$\begin{aligned} &I^{MS}\_G\{AEofS\}_j = \Omega IC_v\_AEofS_i, \\ &InputCanal\_IC_v\_AEofS_i, \\ &OutCanal\_IC_v\_AEofS_i \end{aligned} \tag{13}$$

- system model of the $I^{MS}\_AEofS_i$ level:

$$\begin{aligned} &I^{MS}\_AEofS_i = \\ &= \Omega AEofS_i(InputIP_p\_AEofS_i, OutIP_j\_AEofS_i), \end{aligned} \tag{14}$$

Based on the infological stage design results (11-14), the developer can begin to implement the system models of the CPPS development information stage - system tactical and technical characteristics of the physical information model ($PCofI\_PIM$) at all levels of CPPS decomposition. Based on the proposed methods, it is possible to present a system model of the information stage at the following subsystem levels:

- system model of the $PCofI\_PIM\_MS_0''$ level:

$$PCofI\_PIM\_MS_0'' = \\ = \Omega I^{MS}\_Sub\_S_k\_MS_0'', PCofI\_PIM\_Sub\_S_k \quad (15)$$

- system model of the $PCofI\_PIM\_Sub\_S_k$ level:

$$PCofI\_PIM\_Sub\_S_k = \Omega I^{MS}\_G\{AEofE\}_j\_MS_0'', \\ PCofI\_PIM\_G\{AEofE\}_j \quad (16)$$

- system model of the $PCofI\_PIM\_G\{AEofE\}_j$ level:

$$PCofI\_PIM\_G\{AEofE\}_j = \\ = \Omega I^{MS}\_AEofS_i\_MS_0'', PCofI\_PIM\_AEofS_i \quad (17)$$

- system model of the $PCofI\_PIM\_AEofS_i$ level:

$$PCofI\_PIM\_AEofS_i = \\ = \Omega PCofI\_InputIP_p\_AEofS_i, \\ PCofI\_OutIP_p\_AEofS_i \quad (18)$$

To describe the system information model, we take the form of this level node $PCofI\_PIM$, and the edges will be the tactical and technical characteristics of information links ($PCofI\_InputIP_p$, $PCofI\_OutIP_p$). The developer is obliged to comply with the condition for the existence of binary relations $PCofI\_OutIP_p\_AEofS_o$ and $PCofI\_InputIP_p\_AEofS_i$:

$$PCofI\_PIM\_AEofS_0 \cong PCofI\_PIM\_AEofS_i \quad (19)$$

The last stage of CPPS design in accordance with the proposed system design and control representation in the complex CPPS development is the functioning algorithm construction (AF) in this study is proposed to use the principle of graph-schemes due to their convenience of presentation. Therefore, the following method is proposed for their construction:

- the developer analyzes Aim_i, $Task_j$, $InputCanal$, $OutCanal$ of the system-level design;

- according to each $Task_j$ choose Op_h graph-scheme of the algorithm based on the proposed method [8]. Let's designate that the first task of the $Task_0\_MS_0'$ level, will be Op_0, therefore, for the next $Task_0\_MS_0'$ level of decomposition will be $Task_1\_MS_0''$ level, and will match the Op_1 operator and by analogy $Task_j\_MS_o''$ operator will be Op_h;

- analyzing the sequence of $Task_j\_MS_o''$ execution for a given level, the developer reaches $Aim_i\_MS_0''$, necessary to achieve the main goal of the CPPS design;

- the developer arranges Op_h according to the logical sequence to execute $Task_j\_MS_o''$. Let introduce the concept of conditional and unconditional transition from $Task_j\_MS_o''$ to $Task_{j+1}\_MS_o''$, if transition is without condition then Op_h connect to $\rightarrow$, if Op_h has connections with other operators Op_{h+1} and by condition, then it is necessary to maintain the concept of a conditional operator (CO_l), which works by analogy with the block "condition" from the basic theory of the algorithm construction. Based on this assumption AF can take into account not only the "linear view", but also implement the "disjunctive transition", "cycles" and "transition conditions" to achieve $Task_j$.

2.3 System models formalization

To formalize the system models developed in Section 2.2, in this research it was proposed to use the mathematical apparatus of regular algorithms and algebra of logic. Based on the theory of the regular schemes and algorithmic algebras apparatus, in this study we define the following notation:

OA_q – set of the algebra elements operators, are Op_h, and conduct additional operations for the convenience of presentation and are not operators of the information transformation. H – identity operator, $\varnothing$ – empty operator;

AP_r – a set of conditions that include all logical conditions CO_l, and can take the following values *true*, *false* or CO_l^x $x = [true, false, b, c, y, \ldots, r]$.

Therefore, it can be represented in the form of a tuple OA_q:

$$OA_q = (Op_h, CO_l, H, \varnothing, true, false, x) \quad (20)$$

For ease of manipulation OA_q (within algorithmic algebras) it is necessary to define the main types of operations:

Definition 1. Operator multiplication – strictly sequential execution of operators in the order of their queue.

$$OA = OA_i \cdot \underset{CO_l}{(} OA_k \vee OA_n) \cdot OA_m \quad (21)$$

where: $OA_i = Op_i, \ldots, Op_j$; $OA_k = Op_k \underset{CO_l}{(} Op_l \vee Op_m)$,

where CO_l – logical conditions; $OA_n = Op_n\{Op_p\}Op_t$;

$OA_m = Op_m, \ldots, Op_s$.

Definition 2. Operators Addition - is a conditional simple operation branching ($\tilde{O}p_h$) or nested within each other.

$$OA_q = \tilde{O}p_1 \cdot \tilde{O}p_2 \cdot, \ldots, \cdot \tilde{O}p_h \quad (22)$$

$$OA_{q-1} = (\underset{CO_1}{\tilde{O}p_1} \vee (\underset{CO_2}{\tilde{O}p_2} \vee (\underset{CO_3}{\tilde{O}p_3} \vee \ldots \ldots \vee (\underset{CO_l}{\tilde{O}p_h} \vee e))) \quad (23)$$

Using expression 23 as an example, this study suggests the following understanding: $(\underset{CO_l}{\tilde{O}p_h} \vee e)$ the results of addition of simple operations must satisfy the CO_l condition, after that, addition is performed to the next CO_3 condition, etc.

Definition 3. The addition process in the system model is the observance of conditional branching and connection of the algorithm paths forward depending on CO_l. The syntax for writing is represented as expression 24:

$$OA_q = (\underset{CO_l}{Op_{h-1} \vee Op_h}^{CO_l}) \quad (24)$$

An example of representing the addition process in the system model can be represented by the following expression:

$$OA_q = (\underset{CO_1}{Op_1 \vee Op_2 \cdot (\underset{CO_3}{Op_3 \vee Op_4}^{CO_1})} \vee Op_5^{CO_3}) \quad (25)$$

Based on studies by the expression 25 system models can be described iteration "forward" after checking conditions CO_l, in this case, it is necessary to characterize the frequently occurring "back" iteration, in which the return is carried out by Op_h, not to a condition CO_l. This requires a new definition in the developed formalization.

Definition 4. The iterative addition process is the rules for writing and reading sequential and concentrically nested loops. Based on this definition, we represent the existing types of loops:

- simple sequence of loops

$$CY = CY_1 \cdot CY_2 \cdot \ldots CY_n \quad (26)$$

where: $CY_n = \{Op_h\}_{CO_l}$;

- concentric of nested loops:

$$CY_n = \{\{\{\ldots\{Op_1\}_{CO_1} Op_2\}_{CO_2} Op_3\}_{CO_3} \ldots Op_h\}_{CO_l} \quad (27)$$

- a special case of a loop CY_n with return to operator Op_h when the condition CO_l performed:

$$CY_n = \{Op_{h-2} \ldots Op_{h-1}\}_{CO_l} Op_h \quad (28)$$

Based on definitions 1-4, the developer can provide the following description of the designed algorithm iterative paths:

- execution of an interactive way by the condition $CO_{l-1} = false$, until the closing curly brace and repeat, then return to the check if $CO_l = true$, then the action of the algorithm is carried over the closing brace to the next operator.

$$CY_{n-3} = \{Op_1 \ldots Op_{h-1}\}_{CO_{l-1}}^{CO_l} Op_h \quad (29)$$

- execution of an interactive path conditionally $CO_{l-1} = false$, return to open brace and repeat. By condition $CO_l = true$ the action wraps behind the closing brace.

$$CY_{n-m} = \{Op_1 \ldots Op_{h-1}\}_{CO_l}^{CO_{l-1}} Op_h \quad (30)$$

Condition CO_l presented below the braces, determines the check of it depending on its parameter (*false,true*) and depending on the given parameter, make a conditional jump. Condition CO_{l-1} presented above the brace indicate where to return when the parameter *false* and continue the algorithm. An example of this type of the algorithm description is presented in 31.

$$OA_q = \{Op_{h-5} (Op_{h-4} \vee Op_{h-3})_{CO_{l-3}}^{CO_{l-3}} \{(Op_{h-2}\}_{CO_{l-2}} \vee Op_{h-1})_{CO_{l-1}}^{CO_{l-2}}\}_{CO_l}^{CO_l} Op_h \quad (31)$$

Definition 5: The conjunction of the algorithm is an unconditional branching into the execution of several parallel paths of the algorithm.

$$OA_q = [Op_{h-2} \wedge Op_{H-1} \wedge Op_h] \quad (32)$$

In this study, the following assumption was made that the actions of the algorithm operators enclosed in square brace start in parallel after the opening brace. For this record to be correct, all paths must be equal to exp. 33, therefore we define P – path length (number of operators) in this branch of the algorithm.

$$POp_{h-2} = POp_{H-1} = POp_h \quad (33)$$

When a condition occurs $POp_{h-2} \neq POp_{H-1}$ it is necessary to add n number of operators to the shorter length of the algorithm before condition 33.

To determine the algorithm branches is proposed the following recording method. CO_l^x is indicated at the bottom of the opening square bracket, which means checking the

condition and the beginning of the parallel operation of the algorithm when x, at the top above the closing brace is the exit condition CO_{l-1}^{x} from the algorithm. It should be noted that when executing parallel algorithms, it is necessary to take into account the x parameter at CO_l must be determined by the dimension of conditions and must always be determined for each specific condition $x=[true, false, b, c, y, \ldots, r]$, fulfillment of conditions b, c, r, which satisfies the requirements of the exit conditions CO_{l-1}, we can assume that the parallel algorithm has fulfilled its function. An example recording is presented in expression 34.

$$OA_q = \underset{CO_l^x}{[} Op_{h-4}^{\ b} \wedge \ldots \wedge Op_{H-1}^{\ c} \wedge Op_h^{\ r} \overset{CO_{l-1}^x}{]} \tag{34}$$

Similarly, the possibility of existence CO_l^x in $x=[true, false, b, c, y, \ldots, r]$ for the operation of the adding algorithms iterative process. For the convenience of writing the system model, the following syntax rules are proposed:
- check condition CO_l written below the closing curly brace;

- define the following notation as defining superscripts for curly braces $CO_l^{true, false, b, c, y, \ldots, r}$, and place it in the place where the action of the algorithm returns depending on the values condition $x=[true, false, b, c, y, \ldots, r]$

$$OA_q = \overset{CO_{l-1}^b}{\{} Op_{h-4} \overset{CO_{l-2}^{true}}{\{} Op_{h-3} \{Op_{h-2} \cdot Op_{h-1}^{\ CO_{l-3}^y}\}_{CO_i} \tag{35}$$

On the basis of the developed system models formalizations language, the developer can carry out an algebraic description of all operators and the conditions of their interaction in the form of an algorithm of functioning (AF) at all stages and levels of CPPS decomposition.

Use one space after periods and colons. Hyphenate complex modifiers: “zero-field-cooled magnetization.” Avoid dangling participles, such as, “Using (1), the potential was calculated.” [It is not clear who or what used (1).] Write instead, “The potential was calculated by using (1),” or “Using (1), we calculated the potential.”

3. METHOD FOR THE CPPS OPERATING ALGORITHMS SYNTHESIS

System models formalization makes it possible to develop an algorithm for the functioning of a certain stage at the CPPS design chosen level, but at the same time it does not solve the general problem of their interaction as a whole to solve $Aim_i \_ MS_0'$. Therefore, the problem arises of synthesizing private algorithm of functioning AF into a general. Based on the above, this study proposed the following synthesis methods:

Definition 6. Functioning algorithms combining – if the set Op_h to execute private algorithms is general and CO_l is also common, therefore they can be combined by CO_l.

$$AF = \underset{OC_l}{(} AF_r \vee AF_m) \tag{36}$$

provided that $CO_l = (true, false)$

$$AF_r + AF_m = \underset{OC_l}{(} AF_r \vee AF_m) = \\ = \begin{cases} AF_r \text{ при } CO_l = true \\ AF_m \text{ при } CO_l = false \end{cases} \tag{37}$$

Based on 36-37, the developer can apply the system of algorithms identical transformations axioms, which will make it possible to simplify the implemented algorithm logical structure.

Definition 7. Decomposition of functioning algorithms is a breakdown of the algorithm into simple algorithms without losing the identities of its operation conditions. An example of the functioning algorithm 38 decomposition is presented in 39-40.

$$AF_i = \underset{OC_{l-2}}{(} AF_j \vee \underset{OC_{l-1}}{(} AF_m \vee AF_n \overset{OC_{l-1}\, OC_{l-2}}{)}) \cdot \\ \cdot \underset{OC_l}{(} AF_p \vee AF_g \overset{OC_l}{)} \tag{38}$$

- enlarged example of decomposition:

$$AF_i = \begin{cases} AF_1 = AF_j \underset{OC_l}{(} AF_p \vee AF_g) \\ \text{when } OC_{l\text{-}2} = true \\ AF_2 = \underset{OC_{l-1}}{(} AF_m \vee AF_n) \underset{OC_l}{(} AF_p \vee AF_g) \\ \text{when } OC_{l\text{-}2} = false \end{cases} \tag{39}$$

- path is a fragment AF_i which contains a specific sequence Op_h, by the algorithm will be $Op_{h-1} \to Op_h$. Hence it follows that the path of the algorithm is simple, which have no branching, and which is complex, contains: iterative process (definition 4), conjunction (definition 5), etc.

- path lenth – time characteristic of the algorithm and is used to prove the identity of the algorithm for the structural minimization of the same paths.

- detailed example of decomposition:

$$AF_i = \begin{cases} AF_1 = \begin{cases} AF_1' = AF_j \cdot AF_p \\ \text{when } (OC_{l\text{-}1} \wedge OC_l) = true \\ AF_2'' = \underset{OC_{l-1}}{(} AF_m \vee AF_n) AF_g \\ \text{when } (OC_{l\text{-}1} \wedge OC_l) = false \end{cases} \\ AF_2 = \begin{cases} AF_j \underset{OC_l}{(} AF_p \vee AF_g) \\ \text{when } (OC_{l\text{-}2} \wedge OC_{l\text{-}1}) = true \\ AF_n \underset{OC_l}{(} AF_p \vee AF_g) \\ \text{when } (OC_{l\text{-}2} \wedge OC_{l\text{-}1}) = false \end{cases} \end{cases} \quad (40)$$

Denote $\vec{A}F_i$ as a partial algorithm or algorithm path.

$$\vec{A}F_i = \vec{A}F_1 \cdot \vec{A}F_2 \cdot \vec{A}F_3 \cdot \ldots \cdot \vec{A}F_{i-1} \quad (41)$$

To combine the original algorithms, it is necessary to decompose it in accordance with Definition 7 and determine simple paths, then analyze Op_h and OC_l for each $\rightarrow$ from AF_i to identify the same.

Axiom 1. Suppose any two algorithms $\vec{A}F_i$ and $\vec{A}F_{i-1}$ будут тождественно эквивалентны, will be identically equivalent if they consist of the same $\rightarrow$ path lengths, Op_h and OC_l in $\rightarrow$ will be equivalent, OC_l – is identical.

An example of Axiom 1. Let any three algorithms 42-44 be given:

$$\vec{A}F_1 = Op_1 \cdot Op_2 \underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \vec{A}F_{i-1} \quad (42)$$

$$\vec{A}F_2 = \underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} Op_1 \cdot Op_2 \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \vec{A}F_{i-1} \quad (43)$$

$$\vec{A}F_3 = \vec{A}F_{i-1} \cdot Op_1 \cdot Op_2 \underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \quad (44)$$

As you can see that the algorithms $\vec{A}F_1$, $\vec{A}F_2$ and $\vec{A}F_3$ have the same operators ($Op_1, Op_2, Op_3, Op_4, Op_5, Op_6$) and conditions (OC_1,OC_2), are the same $\rightarrow$. To obtain a combined algorithm and simplify data manipulation, we use the following abbreviations.

$$Op_1 \cdot Op_2 = W \quad (45)$$

$$\underset{OC_1}{(} Op_3 \vee Op_4 \overset{OC_1}{)} = M \quad (46)$$

$$\underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} = K \quad (47)$$

$$\vec{A}F_{i-1} = Z \quad (48)$$

Substitute the accepted abbreviations from 45-48 to 42-44, and we get the following form of recording the algorithms presented by 49-51:

$$\vec{A}F_1 = WMKZ \quad (49)$$

$$\vec{A}F_2 = MWKZ \quad (50)$$

$$\vec{A}F_3 = ZWMK \quad (51)$$

Let us define additional conditions for the algorithms existence $\vec{A}F_1$, $\vec{A}F_2$ и $\vec{A}F_3$ using systems:

$$\hat{S}_1 = \begin{cases} \vec{A}F_1 \text{ when } s_1 = true \\ \vec{A}F_1 \; \vec{A}F_2 \text{ when } s_1 = false \end{cases} \quad (52)$$

$$\hat{S}_2 = \begin{cases} \vec{A}F_2 \text{ when } s_2 = true \\ \vec{A}F_3 \text{ when } s_2 = false \end{cases} \quad (53)$$

At the first step of the algorithms synthesis $\vec{A}F_1$, $\vec{A}F_2$ and $\vec{A}F_3$ the developer needs to determine the same paths according to the criterion of least repeatability and make a record according to the conditions of existence $\vec{A}F_1$, $\vec{A}F_2$ and $\vec{A}F_3$. As a result of manipulations, the developer can get many options for synthesizing algorithms into one, and depending on the requirements that need to be achieved, the most optimal is selected. Expression 54 gives an example of an algorithm synthesis by the minimizing the number of operators criterion using the introduced abbreviations from 49-51, taking into account additional conditions 52-53.

$$\vec{A}F_{1-3} = \underset{s_2}{(} M \vee Z) \cdot W \cdot \underset{s_1 \vee \bar{s}_2}{(} M \vee e) \cdot K \cdot \underset{s_1 \vee s_2}{(} Z \vee e) \quad (54)$$

Let us substitute 54 accepted abbreviations into 45-48 and obtain the synthesized algorithm $\vec{A}F_{1-3}$ as a set of operators and a condition.

$$\vec{A}F_{1-3} = \underset{s_2}{(} \underset{OC_1}{(} Op_3 \vee Op_4) \vee \vec{A}F_{i-1} \overset{s_2}{)} \cdot Op_1 \cdot Op_2 \underset{s_1 \vee \bar{s}_2}{(} \underset{OC_1}{(} Op_3 \vee Op_4) \vee e \overset{s_1 \vee \bar{s}_2}{)} \underset{OC_2}{\{} Op_5 \cdot Op_6 \overset{OC_2}{\}} \underset{s_1 \vee s_2}{(} \vec{A}F_{l-1} \vee e) \quad (55)$$

By analyzing the synthesis result of the combined algorithm 55 and the original algorithms 42-44, the developer can see

that in the algorithms $\vec{A}F_1$, $\vec{A}F_2$ and $\vec{A}F_3$ there are 21 operators and 6 conditions for a given criterion about the need to minimize operators, 10 operators were obtained while maintaining the number of conditions. The number of operators in this synthesis was reduced by 2 times, while the algorithm fulfills all the conditions set initially. Figure 4 graphically shows a comparison of the execution parameters of the synthesized and non-synthesized algorithm from the example with the number of conditions = 6.

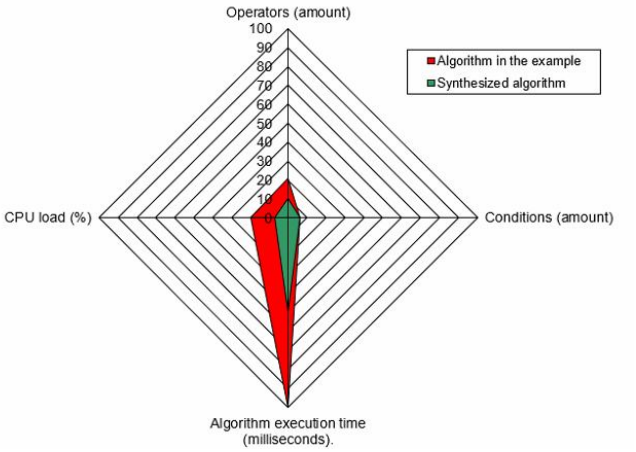

Figure 4: Comparison of the synthesized execution parameters and non-synthesized algorithm.

5. CONCLUSION

On the basis of the functioning algorithms synthesis developed method, theoretical studies of the three algorithms synthesis were carried out, which showed: at the initial stage there were 21 operators and 6 conditions under a given constraint conditions - minimization of operators. As a result of the proposed synthesis method approbation 10 operators were obtained while maintaining the number of conditions, which will make it possible to minimize the general CPPS functioning algorithm by 1.5 - 2 times, while the synthesized algorithm fulfills all the initially set conditions. Summing up, it can be argued that the developed synthesis method makes it possible to automate the process of creating a generalized algorithm for the CPPS functioning, which will reduce the development time and achieve the maximum economic effect.

REFERENCES

1. V.Alcácera, and V. Cruz-Machado. **Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems**, *Journal of Engineering Science and Technology*, Vol. 22, no. 3, pp. 899-919, 2019. https://doi.org/10.1016/j.jestch.2019.01.006
2. Y. Lu. **Industry 4.0**, *Journal of a Survey on Technologies, Applications and Open Research*, Vol. 6, pp. 1-10, 2017. https://doi.org/10.1016/j.jii.2017.04.005
3. P. Leitão, A. W. Colombo, and S. Karnouskos. **Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges**, *Journal of Computers in Industry*, Vol. 81, pp. 11-25, 2016. https://doi.org/10.1016/j.compind.2015.08.004
4. Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu. **Review on cyber-physical systems**, *IEEE/CAA Journal of Automatica Sinica*, Vol. 4 , no. 1, pp. 27 – 40, 2016. https://doi.org/10.1109/JAS.2017.7510349
5. T. Wagner, Ch. Herrmann, and S. Thiede. **Industry 4.0 Impacts on Lean Production Systems**, *Journal of Procedia CIRP*, Vol. 63, pp. 125-131, 2017. https://doi.org/10.1016/j.procir.2017.02.041
6. M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann. **Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0**, *IEEE Transactions on Industrial Informatics*, Vol. 14, no. 4, pp. 1722 – 1731, 2018. https://doi.org/10.1109/TII.2018.2804917
7. Y. Cai, B. Starly, P. Cohen, and Y.-Sh. Lee. **Sensor Data and Information Fusion to Construct Digital-twins Virtual Machine Tools for Cyber-physical Manufacturing**, *Procedia Manufacturing*, Vol. 10, pp. 1031-1042, 2017. https://doi.org/10.1016/j.promfg.2017.07.094
8. F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee. **Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison,** *Engineering*, Vol. 5, no. 4, pp. 653-661, 2019. https://doi.org/10.1016/j.eng.2019.01.014
9. M. A. Pisching, M. A. O. Pessoa, F Junqueira, D. J. S. Filho, and P. E. Miyagi. **An architecture based on RAMI 4.0 to discover equipment to process operations required by products**, *Computers & Industrial Engineering*, Vol. 125, pp. 574-591, 2018. https://doi.org/10.1016/j.cie.2017.12.029
10. O. Kuzomin, M. A. Ahmad, H. Kots, V. Lyashenko, and M. Tkachenko. **Preventing of technogenic risks in the functioning of an industrial enterprise**, *International Journal of Civil Engineering and Technology*, Vol. 7, no. 3, pp. 262-270. 2016.
11. E. P. Garina, A.P. Garin, E. V. Romanovskaya, N. S. Andryashina, and Zh. V. Smirnova. **The study of approaches for the coordination of product development systems and production systems in the stage of conceptual design**, *International Journal of Emerging Trends in Engineering Research*, Vol. 8, no. 9, pp. 5746-5749, 2020. https://doi.org/10.30534/ijeter/2020/135892020
12. E. Egorova, and A. Akhmadiev. **Formation of Engineering Capability Matrix of Production Procedure**, *International Journal of Emerging Trends in Engineering Research*, Vol. 8, no. 9, pp. 6537-6540, 2020. https://doi.org/10.30534/ijeter/2020/258892020

13. J.-R. Jiang. **An improved cyber-physical systems architecture for Industry 4.0 smart factories**, *Journal of Advances in Mechanical Engineering*, Vol. 10, no. 6, pp. 1-15, 2018. https://doi.org/10.1177/1687814018784192
14. A. M. Babker, A. E. A. Altoum, I. Tvoroshenko, and V. Lyashenko. **Information technologies of the processing of the spaces of the states of a complex biophysical object in the intellectual medical system health**, *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 8, no 6, pp. 3221-3227. https://doi.org/10.30534/ijatcse/2019/89862019
15. I. Nevliudov, V. Yevsieiev, S. Maksymova, and I. Filippenko. **Development of an architectural-logical model to automate the management of the process of creating complex cyber-physical industrial systems**, *Eastern-European Journal of Enterprise Technologies*, Vol. 4, no. 3 (106), pp.44-52, 2020. https://doi.org/10.15587/1729-4061. 2020.210761