mathematics

MDPI

Article

An Approach for Mathematical Modeling and Investigation of Computer Processes at a Macro Level

Radi Romansky

Department of Informatics, Faculty of Applied Mathematics and Informatics, Technical University of Sofia, 1000 Sofia, Bulgaria; rrom@tu-sofia.bg

Received: 23 August 2020; Accepted: 14 October 2020; Published: 19 October 2020

Abstract: In the digital age, the role of information technology and computer processes is growing. This requires refining the development of software by optimizing the communications between program components and seeking effective interaction in the implementation of processes. Complex module structures are usually developed, which require high compatibility between components and their proper functioning. The purpose of this article is to propose an approach for investigation of a set of connected computer processes executed on a macro level by using deterministic modelling. A formal technological procedure for conducting a deterministic investigation of the interaction between processes was developed. It allows for the transition from the object-original to an adequate mathematical model with its program realization. The core of the constructed procedure is the phases "mathematical formalization", "mathematical description", and "program realization". The goal was to present an application of the procedure to investigate all possible realizations of connected processes, presented as nodes in a directed graph scheme of algorithms by determining the reachability. The program language APL2 was used as a tool for program description of the defined mathematical models, which were realized in the software system TryAPL2 for research automation. A preliminary mathematical formalization of interacting processes was made by presenting an example graph scheme and its transformation into an ordered structure. On the basis of the mathematical description, we developed two program models for automation of the transition to an ordered graph scheme and determination of all possible paths in it for activation of sequences of processes. The proposed models are part of a generalized environment for program investigation of the computer processing organization.

Keywords: computing; mathematical modelling; deterministic formalization; processes investigation; assessments

1. Introduction

It is known that computer processing is a mathematical (functional) transformation of a set of input data (D) into output results (R), summarized by a global function $f: D \rightarrow R$. In this sense, computer data processing is a form of concrete realization of a complex mathematical function in an environment of hardware and software tools and means in different versions—traditional, parallel [1], real-time [2], learning scheduling [3], etc. Regardless of the computer environment realization, the organization of computer processes covers two hierarchical levels related to its program management at a high (macro) level and its machine implementation at low (micro) level. The micro-level presents the organization of elementary operations for machine calculations generated by firmware (concrete sequence of microinstructions from the activated microprogram). An approach for organization of low-level model investigation is discussed in [4], proposing a unified framework for the joint conduct of the classification process and the modelling. The goal is to ensure the effectiveness of model investigation on a micro level (firmware). The firmware analysis permits the determination of correct

functionality of processing at the micro level, with the generalized approach being presented in [5], which is directed to identification of firmware modules, its extraction, disassembling, modification, and reprogramming. This approach is particularly relevant in the development of embedded devices for the purpose of the Internet of things (IoT). In this respect, a software technological framework for the continuous execution of binary firmware together with independent testing is proposed in [6], which is not affected by the peripherals (without ignoring them) and processes the input–output data for the firmware on the basis of automatically generated models. An evaluation of this framework is made and good assessment for the effectiveness is obtained. Another actual example of the importance of the investigation of the processing in micro-level (firmware) is the proposed in the method of [7] for security analysis of a memory, which stores micro algorithms for 3G modem control. Testing the method for different versions of the modem has shown good results.

The macro level (as opposite of micro level) in general is a sequence of separate (generally independent) procedures for data stream processing, which are realized by activation of separate algorithms integrated in the frame of symbolic general algorithm, conditionally marked as A_G. In both cases, the organization of the processes at each of the two levels is subject to probabilistic parameters and conditions in the computer environment [8]. Formally, from the point of view of formalization, any investigation of processes at the macro level and communications between them is subject to a defined procedure. In this aspect, a generalized scheme of a procedure for analysis and optimization of complex-structured objects during the modelling and algorithms is proposed in [9]. Numerical methods and algorithms for finding the optimal solution are used. Analysis of the features of the management process is carried out to increase the efficiency of using these components of the analyzed system.

The research of information processing in the processes of program module design at a high (macro) level has different applications due to the modern architectures, including distributed computer systems. One of the directions is the formalization of the relationship between the two levels—micro and macro—with the article in [10] proposing an abstract basis for field calculations, which contain several syntactic constructions, and the relationship with the micro level (actions of individual devices and their interaction when performing collective processing) is analyzed. The goal is to facilitate the design of a user interface (API—Application Program Interface) in complex software systems.

Another direction for the application of the analysis of high-level programming processes is the automatic correction of programs, which must ensure reliable implementation of the applied actions through the use of modern tools. In this sense, the study in [11] discusses the problem of locating faults in program codes at the macro level and interactions between programs, including their impact on automatic correction systems. The goal is to analyze the activated procedures, the possibilities for error correction, and the reliable investigation of system performance parameters through analysis at the macro level. The practical use of formal techniques for analysis is discussed in [12], where a pragmatic application of formal method technologies is a purpose of the research. This formal analysis was focused on the controller components of the software implementation for error management and the logic was analyzed by using a technique of model investigation. This helped to analyze the possible situation of a risk and for verification of risk control measures relating to the software components.

The relationships between different software entities can be used for evaluation of the quality of a program, which determines the importance of the tools for computation of their metrics. In this respect, a classification (made by a systematic literature review in software engineering) of the different kinds of coupling relations, together with the metrics to measure them, is presented in [13]. The result of this research is a proposal of suitable tools for software engineering for extracting program metrics united in four groups—structural, dynamic, semantic, and logical. As an addition, this research retrieves tools that extract the metrics belonging to each coupling group. The problem of object monitoring, including software components, is an important one in the processes of relationship evaluation. The communication time between software objects is a critical parameter and is subject to minimization. A set of procedures for effective communication between territorially connected systems is proposed in [14] and an analysis of the effectiveness is made.

One of the applied approaches for investigation of computer processes is mathematical modelling by using formalized descriptions of the investigated object through abstract language (mathematical formal system) or through mathematical relations describing functional behavior. This approach is applicable in different areas—for example, an application in the field of complex electronic systems for various purpose is discussed in [15]. The article proposes a hierarchical method of mathematical and computer modelling of interval-stochastic thermal processes that take into account various physical phenomena in in the considered systems. Different mathematical means can be used for their creation (algebraic, differential, and integral calculus; theoretical systems; etc.). The research presented in [15] is realized by systems of stochastic, unsteady, and nonlinear partial differential equations, and their computer simulation is made by using supercomputer. Another approach is applied in [16], when "*a new rumour spreading model in social networks*" is proposed that was realized by using discrete mathematical modelling. An optimal control strategy to fight against the spread of the rumor is recommended in the article. Five different perspectives on mathematical modelling are reviewed in [17] because, according to the authors, "*there is not a single agreed-on-definition of what mathematical modelling is or how it should be done*". The classification discussed in the article is directed to "*realistic modelling, educational modelling, models and modelling perspective, socio-critical modelling, and epistemological modelling*". Finally, a statistical approach for mathematical modelling or solving the problem with minimization of the search time for the necessary information is proposed in [18].

Computer modelling is mathematical modelling in which computer tools are used to create the model and to conduct experiments with it. Mine principles of computer modelling are discussed in [19] and a formal technological scheme for investigation by modelling is proposed. The basic phases of this process are determined: (a) preliminary functional decomposition of the modelled object and a conceptual model formulation, (b) mathematical formalization of structure and relations based on the conceptual model, (c) defining of the functional algorithm, (d) design of the mathematical model, (e) realization of a program model in suitable software system, and (f) organization of experiments with results analysis and conclusions. In addition, a hierarchical procedure for evaluation based on obtained results from modelling with three levels (empirical, analytical, and evaluation) is proposed.

Different language tools (universal and specialized) can be used to create a program model, with thousands of languages having been developed for various applications. In this regard, article [20] proposes an overview of the set of modelling languages, specifically in Industry 4.0. On the basis of this extensive review, the researchers updated the systematic mapping study of modelling languages and modelling techniques. A total of 408 relevant publications were identified on the basis of the study of 3344 candidate publications. One of the article's goals was to determine the modelling languages in system engineering and knowledge representation. The program language, which can be used for mathematical formal description and model preparation in the area of computing, is APL2 (Array Program Language), which allows for the creation of additional user workspaces with models formed as a separate function or as sequentially called functions. Each defined function (enclosed by symbol ∇) can be executed independently or when calling from another function. Each function has its own definition (header part), which contains its name and its possible arguments (variables), through which data can be exchanged with other functions. There are rules for defining (explicitly or implicitly) local and global variables. Three basic components are connected with the loading—definition of virtual vector (array) for storing items, definition of a relative storage address where the array could be stored, and definition of the virtual distance between subsequent elements [21].

The investigation of processes in data processing is essential for improving the overall organization, both at the macro level and at the low (micro) level. This would allow pre-ensuring the effectiveness of the software and firmware, which is important to minimize the time parameters of execution and communications. The task of ensuring efficient processing is relevant in today's digital age due to the increased amount of processed data and communications between programs, including cloud computing, Internet of things, and big data analytics. In this respect, the article presents an approach for automation of the process interactions on a macro level on the basis of mathematical

description and software realization. A technological scheme of the formal procedure for investigation is proposed. Three phases form the core of this procedure on the basis of which the investigation was conducted—mathematical formalization, mathematical description, and program realization. As a result of their implementation, we performed the following: (1) defining an initial conceptual model by using graph theory, (2) producing a deterministic mathematical formalization for analytical description of the proposed graph scheme and its transformation into an ordered structure, and (3) program realization of the developed mathematical models in the software system TryAPL2 by using the program language APL2. The aim was to propose a formal mathematical approach and tool for evaluation of characteristics of macro-level computer processing. The proposed models are part of a generalized environment for program investigation of the computer processing organization at macro and micro levels.

2. Materials and Methods

The main essence of a model study is to replace the original object Ω_O with another object-model Ω_M through which the properties or behavior in certain situations of the original are studied by experimenting with the model [22].

The *object-original* Ω_O can be an arbitrary system or process that may not actually exist. Nevertheless, its system properties can be described by finite sets, such as S_O—system parameters characterizing the internal state of the real system, its structure, and functioning; Y_O—quantitative characteristics of system parameters, describing mainly resultant behavioral features that are important in the interaction with other systems; and X_O—external actions influencing the behavior of system parameters. In this way, a formal representation of the object-original as a class of finite discrete sets $\Omega_O = \{S_O, Y_O, X_O\}$ can be made. The peculiarity is that the main sets in the formed class may be too large, which will require the selection of adequate subsets. In this reason, when studying a system, a subset $\{y_o\} \in Y_O$ is usually chosen to be analyzed under the influence of external factors $\{x_o\} \in X_O$, and each individual characteristic y_{oi} ($i = 1 \div K$, where K is the total number of elements in the subset of individual characteristics) depends on some subset $\{s_o\} \in S_O$ of system parameters (usually the influence of the other parameters is neglected). This subset of selected system parameters determines the spice of the system, and the characteristics are the data describing its organization and behavior according to the goal of the research. In this sense, each object of study should be considered as a complex of two related parts—static (independent of time) and dynamic (system parameters and characteristics that depend on time).

The *object-model* Ω_M must reproduce with sufficient accuracy the real object under study, being in accordance with the selected goal of the concrete research. The model should comply with the selected conditions for the analysis of the behavior of the original system or process. This determines the need to coordinate the subsets of the class Ω_O with appropriate components used for realization of Ω_M. In the concrete case, this coordination requires correct transformation of the selected subset in mathematically presented subsets for the model realization. As a result, the following components can be defined: $\{s_m\} \in S_M$—parameters of the object-model, $\{y_m\} \in Y_M$—characteristics of the object-model, and $\{x_m\} \in X_M$—external factors of influence for the object-model. The replacement $\Omega_O \rightarrow \Omega_M$ is admissible if the determined model characteristics $\{y_m\} \in Y_M$ sufficiently reflect the respective quantitative characteristics $\{y_o\} \in Y_O$, defined in the modelling process. In this sense, the modelling is a replacement of a real functional dependence $\{y_o\} = \Phi_O[\{s_o\}, \{x_o\}, T_o]$, describing the behavior of the original object in time, with a corresponding equivalent dependence $\{y_m\} = \Phi_M[\{s_m\}, \{x_m\}, T_m]$, where usually the model time T_m is related to the real time T_o by scaling. The main requirement in modelling is to find such an analytical dependence that describes with sufficient accuracy the behavior of the original in terms of the objectives of the study.

A summary of the information above is made by the formalized scheme of model investigation presented in Figure 1.

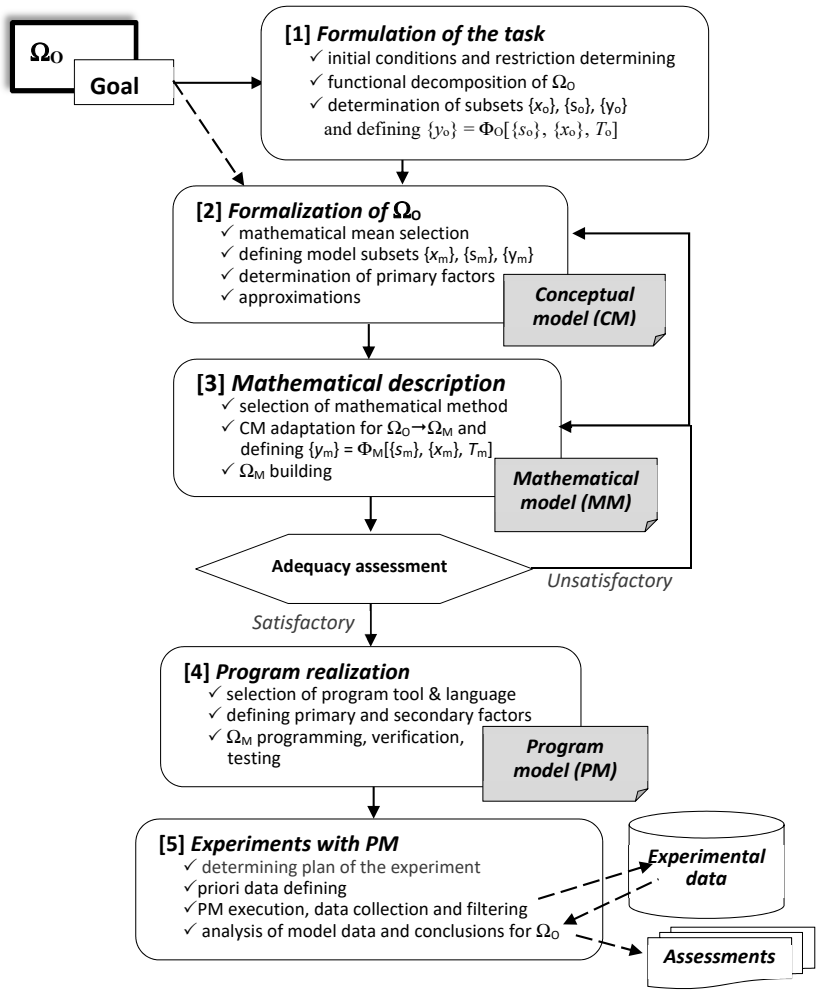


Figure 1. Generalized formal procedure for organization of model investigation.

The *model investigation* is related to three basic phases, realizing the successive components of a computer model (conceptual model (CM), mathematical model (MM), program model (PM)) on the basis of mathematical formalization, mathematical description, and program realization of the designed mathematical model in a program source using a suitable software environment. The phase of *mathematical formalization* is an initial stage, allowing us to make an adequate transition from the formulated task to achieve the goal in the investigation of the object-original (Ω_O) to the actual development of a mathematical model (Ω_M). The phase of *mathematical description* is the basic stage in the investigation because it must build a sufficiently adequate model of the real object (process or system). This requires an appropriate mathematical environment to be selected (formal, deterministic, probabilistic, empirical, etc.). The third phase of *program realization* transforms the mathematical description into a suitable software environment and creates the final result of the modelling, which will be subjected to experiments. A large number of program languages (universal and specialized) can be used to develop the program model, but the choice must take into account the type of mathematical description (Ω_M) and the nature of the model investigation—deterministic, probabilistic, simulation, statistical, or heterogeneous. A good opportunity is provided by the APL2 language and the TryAPL2 operating environment, which is relevant to the deterministic model investigation of computer process organization presented in the following sections.

APL is a high-level program language that is suitable for analytical descriptions and for working with vectors and matrices. There is a possibility for parallel execution of several operations, which surpasses in these respects many of the modern universal algorithmic languages. Characteristic

features are the specific alphabet, the maintenance of various data structures, the ability to work in dialog and program mode, a wide range of mathematical operations, and the possibility for laconic expression of complex transformations through composite operators. The order of operations in the expressions is from right to left. It is suitable for expressing the subordination between the different parts of the algorithm, allowing parallel execution of several operations. In a sequence of included functions, a variable, determined by a value in an external function, remains with the same value in the internal functions if it is not re-defined in them. The last specified value is actual. The functions use labels interpreted as local, and it is possible to apply recursive calculation.

The language version of APL2, implemented in the TryAPL2 operating environment, allows the creation of additional user workspaces from models formed as a separate function or as sequentially called functions. The function (enclosed by the symbol ∇) can be performed independently or when it is called by another function. Each function has its own definition (header part), which contains its name and its possible arguments (variables), through which data can be exchanged with other functions. There are rules for defining (explicitly or implicitly) local and global variables. A function is executed by calling its name, which can be done on its own in an expression or in the body of another function. Execution requires that the arguments (if any) be real variables or constants. In the environment, system variables and functions with reserved name names (starting with the symbol □) are maintained and used. The organization of work is performed through system commands that serve to manage the work session, store and edit copies of workspaces, and transfer data from one working space to another (file exchange management).

An illustration of the applicability of APL2 in deterministic model investigation is made in Figure 2, which is a program description of a formalized functionality of an idealized computing environment supporting traditional processing in a processor (intensity f_1 and number of tasks N_1) and k calls to external memory (intensity f_2 and number of tasks N_2). It is accepted that $N = N_1 + N_2 = const$; the task falls in the processor to $(k + 1)$; and the bandwidth p coincides with the intensity for falling f_0 into a passive state, which allows for the presentation of the following simple analytical dependences:

$$f_2(N_2) = \frac{k}{k+1} \cdot f_1(N_1);$$
$$P = f_0(N) = f_1(N_1) - f_2(N_2) = \frac{f_1(N_1)}{(k+1)}.$$
$$f_1(N_1) = \begin{cases} 0 & ;N_1 = 0 \\ (k+1)/t & ;N_1 \geq 1 \\ N_1(k+1)/t & ;0 < N_1 < 1 \end{cases}$$

```
    ∇ WORKLOAD  N1
[1]  →(N1≠0)/ET1
[2]  F1←0
[3]  →ET3
[4]  ET1:F1←(K+1)+T
[5]  →(N1≥1)/ET3
[6]  F1←F1xN1
[7]  ET3: P←F1+(K+1)
[8]  'N1 = ',N1
[9]  'F1 = ',F1
[10] 'P = ',P  ∇
```

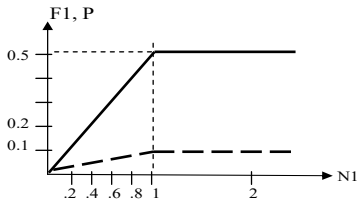
(a)

```
   T←10
   K←4

   WORKLOAD  N1←.2
N1=0.2
F1=0.1
P=0.02

   WORKLOAD  1
N1=1
F1=0.5
P=0.1
```

(b)



(c)

Figure 2. A simple example for analytical model investigation by using APL2—(**a**) program code, (**b**) model execution, (**c**) results interpretation.

To solve the mathematical model, it is necessary to determine the intensity $f_1(N_1)$, which is based on the following: (a) there is zero intensity for the CPU (Central Processor Unit) load when it does not process a task ($N_1 = 0$), for example, due to an infinitely long time for servicing the tasks in state S_2; (b) the service is performed only in one processor and if number of tasks is $N_1 \geq 1$, the analytical dependence $f_1(N_1 \geq 1) = const$ will be saturated and limited to maximum bandwidth $(k + 1)/t$; (c) to solve the situation $0 < N_1 < 1$, it is necessary to know $f_2(N_2)$, which is usually a nonlinear dependence.

The following analytical description can be provided as a result:

$$f_1(N_1) = \begin{cases} 0 & ;N_1 = 0 \\ (k+1)/t & ;N_1 \geq 1 \\ N_1(k+1)/t & ;0 < N_1 < 1 \end{cases}$$

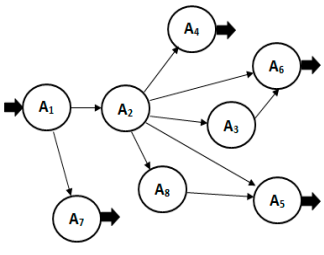
The program model is defined as a separate function WORKLOAD for calculating $f_1(N_1)$ and p, requiring the presentation of of the controllable model parameters (T—fixed processing time for processor, K—number of external memory access). To construct a functional dependence, successive experiments are performed according to a randomized factor plan, for example at values N = 5; T = 10; K = 2, 3, 4, 5. The results of execution of the function at a fixed value for K = 4 and different values (0; 0.2; 0.4; 0.6; 0.8; 1; 2; 4) of the argument N1 are graphically presented in Figure 2c.

3. Mathematical Formalization and Model Construction

In the general case, computer processing can be determined as a set of sequentially executed procedures over data streams. The procedures are implemented on the base of a corresponding algorithm A_j, which can be considered as components of a global functional algorithm A_G. Theoretically, the complete algorithm A_G can be described logically if the individual algorithms A_j and the conditions for their activation at a specific input information flow are known. This undermines its formal description by a directed graph representing a graph scheme of algorithm (GSA), in which the matrix of connections $\{c_{ij}\}$ describes the existence of an information connection $A_i \rightarrow A_j$ between two individual algorithms. In an investigation of the computer processes organization, it is possible to apply both approaches—stochastic (if the relationships are defined as probabilities) or deterministic (the matrix of relationships is Boolean).

3.1. Mathematical Formalization

A preliminary formalization of the investigated object Ω_O is made on the basis of the requirement of the phase [2] of the general technological procedure presented in Figure 1. In the current case, the object Ω_O of model investigation is a global algorithm A_G of exemplary computer processing presented as a GSA (Figure 3a). It is a generalized structure of communicating program modules, each of which can be activated by another, depending on the development of the generalized process. In practice, the interactions between different modules and the possible activations of a concrete sequence of them is a probabilistic process, but the task determined for the research allows a deterministic approach to be applied. In this case, the goal is to determine the number and lengths of all possible paths representing the possible realizations of information processing.



(**a**)

L	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
A_1		1					1	
A_2			1	1	1	1		1
A_3						1		
A_4								
A_5								
A_6								
A_7								
A_8					1			

(**b**)

Figure 3. Mathematical formalization of the investigated object Ω_O (**a**) graph scheme of algorithm (GSA), (**b**) matrix of connections (conceptual model).

In the deterministic approach, a Boolean matrix of connections $L = \{l_{ij}\}$ is applied, where $l_{ij} = 1$ (if there is a connection) and $l_{ij} = 0$ (in the absence of a connection), as seen in Figure 3b.

The formalization will allow for the construction of a conceptual model (CM) for the model investigation organization, which will formulate the basis for the next mathematical descriptions and the creation of a mathematical model (MM—phase [3] in the Figure 1) on the basis of a graph structure $G(A,L)$.

The object of the investigation is a set of independent algorithms (processes), presented as a final discrete set $A = \{A_1, \ldots, A_n\}$, which are executed in different sequences with possible input/output interactions. This allows for the determination of a tree of relations between sequential processes in GSA on the basis of the consequence matrix $L—l_{ij}{:}A_i \rightarrow A_j$ $(i,j \in \{1,2, \ldots, n\})$, as shown in Figure 3b. Usually, the investigation is connected with determining the reachability of a given final task (algorithm) from one or several initial tasks. For this purpose, a system of algebraic equations is compiled, describing the presence of edges a_{ij} between individual algorithms A_i and A_j:

$$A_j = \left\{ \sum_{i=1}^{n} a_{ij} . A_i \right\}; j = 1, 2, \ldots, n$$

Solving this system of equations allows for the construction of all directed paths, as well as for the determination of the equivalent paths that lead to the same event in computer processing. The dependence presented above allows to define a graph model (GSA) of the complete processing algorithm $G(\alpha A)$, which will be transformed in an ordered GSA (OGSA) with nodes numbered on the basis of the following condition:

$$\text{IF } \exists\, pass(A_i \rightarrow A_j\} \Rightarrow A_i < A_j; \text{ FOR } \forall\, A_i, A_j \; (i \neq j).$$

The ordering forms layers of information-independent nodes so that (a) the nodes of the first layer have no predecessors and the nodes of the last layer have no successors, and (b) the nodes included in a common layer have no connecting arcs. The rules for the OGSA formation are as follows:

1. All initial nodes (without predecessors) are numbered first and included in layer (1).
2. A node is numbered and included in the current layer if all its predecessors are already numbered (included in previous determined layers).
3. Nodes to be numbered are successors of already numbered nodes.

3.2. Deterministic Mathematical Description

The phase 3 of the procedure shown in Figure 1 recommends making a transition from the formal model to a mathematical description. On the basis of the chosen approach for deterministic mathematical description, we can make the construction of the model Ω_M on the basis of the following steps.

Step 1. Formation of a transposed matrix LT by transforming the initial L, where the columns are vectors V_{Aj}, describing the successors of each node, fulfilling the condition $LT[i,j] \neq 0$ $(j = 1 \div n)$, shown in Figure 4a. This can be realized by constructions LT←L & N←ρL [1;] and LT←⌀⊃L.

LT	V_{A1}	V_{A2}	V_{A3}	V_{A4}	V_{A5}	V_{A6}	V_{A7}	V_{A8}
A_1								
A_2	1							
A_3		1						
A_4		1						
A_5		1						1
A_6		1	1					
A_7	1							
A_8		1						

(**a**)

V_1	V_2	V_3	V_4
0			
1	0		
1	1	0	
1	1	0	
2	2	1	0
2	2	1	0
1	0		
1	1	0	

AL

AL_1	AL_2	AL_3	AL_4
A_1	A_2	A_3	AL_5
	A_7	A_4	AL_6
		A_8	

(**b**)

Figure 4. Transposed matrix ***LT*** (**a**) and determining the matrix of layers ***AL*** (**b**).

Step 2. Calculation of the elements of a new vector $V_1 = \sum V_{Aj}$ ($j = 1 \div n$; $n = 8$) from LT:

$$V_1[j] = LT[j,1] + LT[j,2] + \ldots + LT[j,n]; \text{ for } j = 1 \div n.$$

If $V_1[j] = 0 \Rightarrow A_j \in Layer(1)$, node A_j must be included in the layer (1), which is marked in the matrix of layers AL. In this case, it is only $V_1[1] = 0$, which determines that the algorithm A_1 must be included in $Layer(1)$ (see column AL_1 in Figure 4b).

Step 3. Calculation of elements of the next vector V_2 on the basis of vector V_1 and row j of LT:

$$V_2[j] = V_1[j] - \sum_{A_k \in V_1} LT[j,k]; for\ j = 1 \div n$$

Elements $V_2[j] = 0$ determine algorithms included in the $Layer(2)$—in our case, algorithms A_2 and A_7, column AL_2 in Figure 4b.

Steps 4, 5, etc. Similar calculation of the next vectors V_q ($q = 3, 4, \ldots$) to determine algorithms included in the $Layer(q)$:

$$V_q[j] = V_{q-1}[j] - \sum_{A_k \in V_{q-1}} LT[j,k]; for\ j = 1 \div n$$

For each layer "q", an added element is marked by $AL[j,q] = 1$. The steps are performed to obtain a vector with only zero elements $\forall V_q[j] = 0$ ($j = 1 \div n$).

The result of applying all steps of the procedure is the distribution of the analyzed processes into separate ranks and defining the complete set of final processes (algorithms).

3.3. Possibility for Application of Formalization in Process Dispatching

An example of the proposed approach can be made by extension of the mathematical formalization in the direction of dispatching independent processes in a closed system S for which the processes of the set A do not require additional resources beyond it. It is accepted that the time vector $T = \{t_1, \ldots, t_n\}$ for realization of processes A_i ($i = 1 \div n$) is known. If the system of resources has a heterogeneous nature, it is possible for a given process A_i to occupy several devices in succession, staying in each for different times $\{t_{i1}, t_{i2}, \ldots, t_{ik}\}$. Then, $t_i = \sum t_{ik}$ and the vector T can be modified in a matrix $T^* = \{t_{ij}\}$ with $n = |A|$ rows and $m = |S|$ columns. This will allow for the formalization of the process of creating a dispatching plan for the analyzed processes in the system, which can be presented as a ordered discrete structure $p = <S, A, G, T, F>$, where S is a set of system resources with $|S| = m$, A is a final discrete partially ordered set of processes in the environment S with $|A| = n$, $G(A,L)$ is a directed graph for describing GSA of the processes from set A, $T = \{t_1, \ldots, t_n\}$ presents the vector of execution times for all processes in the environment, and F determines a formal criteria (strategy) for process dispatching.

This formalization allows for the presentation of the dispatching as a function $D(t) = \{d_1(t), \ldots, d_m(t)\}$ defined in the interval $(0, \tau)$ and accepting integer values from the set of indices $(1, 2, \ldots, n)$ of A. Thus, the elements $d_i(t) = j$ ($1 \le i \le m$; $1 \le j \le n$) of the function D will represent the occupation of a resource S_i in the moment t during the execution of process A_j.

When defining a static parallel plan, it is usually assumed that the GSA does not take into account the logical connections, but only the information dependence on data transmission. This is mainly related to the requirement to minimize the total execution time of the set A in a fixed structure S. In this case, the nodes of GSA determine times for realization of the processes. An example for application of this approach for formalization is presented in the next section.

4. Program Realization and Experimental Results

4.1. Program Model Realization

This subsection refers to the implementation of phase 4 of the technological procedure from Section 2, with the goal of developing program modules for the formation of a program model based on the performed mathematical description from Section 3. Two program functions were realized by using language APL2 in the operation environment TryAPL2, which are presented below. The model investigation was performed by their execution.

Function GSA—program realization of the deterministic mathematical model Ω_M is presented in Figure 5a. The results of the model execution are shown in Figure 5b and include ✓ the Boolean matrix L, describing the initial graph-scheme of the algorithm; ✓ the transposed matrix LT; and ✓ the calculated vectors $V[q]$ (q = 1,2, ...) and the matrix of the layers AL (Table 1), which corresponds to the ordered graph-scheme OGSA (Figure 6). Each of the four defined layers includes procedures that can be performed independently of each other.

Table 1. Separate layers in the matrix AL with the dependence equations for nodes.

Layer 1	Layer 2	Layer 3	Layer 4
$A_1 = f(0)$	$A_2 = a_{12} \cdot A_1$ $A_7 = a_{17} \cdot A_1$	$A_3 = a_{23} \cdot A_2$ $A_4 = a_{24} \cdot A_2$ $A_8 = a_{28} \cdot A_2$	$A_5 = a_{25} \cdot A_2 + a_{85} \cdot A_8$ $A_6 = a_{26} \cdot A_2 + a_{36} \cdot A_3$

```
[0]  S←GSA  L
[1]  'MATRIX  L:'
[2]  □←LT←L
[3]  N←ρL[1;]
[4]  LT←∅⊃L
[5]  'MATRIX  LT:'
[6]  □←LT
[7]  V←AL←(N,N)ρ0
[8]  K←I←1
[9]  QLIST←Q←ρ0
[10] ET1:V[I;K]←(+/LT[I;J← ιN])
[11] →(V[I;K]≠0)/ET2
[12] Q←Q,I
[13] AL[I;K]←1
[14] ET2: →(N≥I←I+1)/ET1
[15] ET3:'VECTOR V[',K,'] : ',V[;K]
[16] '  TASKS IN LEVEL ',K,' : ',Q
[17] QLIST←QLIST,Q
[18] →(0=(+/V[;K]))/OUT
[19] K←K+I←1
[20] QNEW←ρ0
[21] ET4:VA←0
[22] J←1
[23] LOOP:→((+/(J=Q))=0)/JUMP
[24] VA←VA+LT[I;J]
[25] JUMP:→(N≥J←J+1)/LOOP
[26] V[I;K]←V[I;K-1]-VA
[27] →(V[I;K]≠0)/ET5
[28] →((+/I=QLIST) ≠0)/ET5
[29] QNEW←QNEW,I
[30] AL[I;K]←1
[31] ET5: →(N≥I←I+1)/ET4
[32] Q←QNEW
[33] →ET3
[34] OUT:'MATRIX AL OF TASK I IN LEVEL J:'
[35] AL
```

(a)

```
MATRIX  L:
0  1  0  0  0  0  1  0
0  0  1  1  1  1  0  1
0  0  0  0  0  1  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
0  0  0  0  1  0  0  0
MATRIX  LT:
0  0  0  0  0  0  0  0
1  0  0  0  0  0  0  0
0  1  0  0  0  0  0  0
0  1  0  0  0  0  0  0
0  1  0  0  0  0  0  1
0  1  1  0  0  0  0  0
1  0  0  0  0  0  0  0
0  1  0  0  0  0  0  0
VECTOR V[1]: 0 1 1 1 2 2 1 1
  TASKS IN LEVEL 1 : 1
VECTOR V[2]: 0 0 1 1 2 2 0 1
  TASKS IN LEVEL 2 : 2 7
VECTOR V[3]: 0 0 0 0 1 1 0 0
  TASKS IN LEVEL 3 : 3 4 8
VECTOR V[4]: 0 0 0 0 0 0 0 0
  TASKS IN LEVEL 4 : 5 6
MATRIX AL OF TASK I IN LEVEL J:
1  0  0  0  0  0  0  0
0  1  0  0  0  0  0  0
0  0  1  0  0  0  0  0
0  0  1  0  0  0  0  0
0  0  0  1  0  0  0  0
0  0  0  1  0  0  0  0
0  1  0  0  0  0  0  0
0  0  1  0  0  0  0  0
```

(b)

Figure 5. Program function GSA—(**a**) program code of the model Ω_M; (**b**) execution.

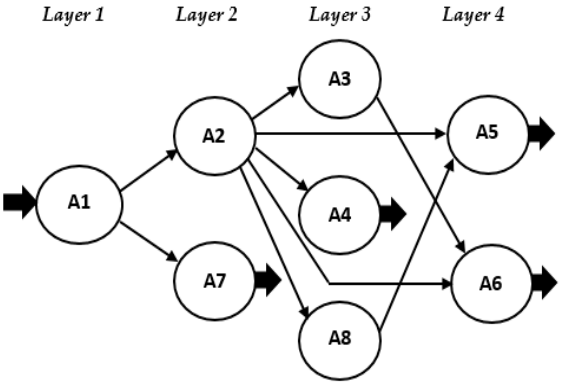


Figure 6. Ordered graph-scheme OGSA.

When considering the case for dispatching independent processes 1 (Section 3.3), we made an assumption for system resources homogeneity $S = \{S_1, ..., S_m\}$ and equal labor intensity of the processes forming the set $A = \{A_1, \ldots, A_n\}$. This means that the mathematical expectation of the times for the realization of the processes were the same, i.e., $E[t(A_i)] \equiv E[t_i] = const$, which allowed for the application of a binary graph to represent the GSA and a fixed time vector $T = \{t_i = \tau/i = 1 \div n\}$. In this case, the formed ordered scheme from Figure 6 determined six paths, as the maximum length was 4, which allowed for the definition of a maximum parallel plan with a minimum execution time for this set of eight processes (minimum number of layers). In this case, Table 1 can be transformed into an optimal parallel plan, requiring three independent processor nodes for realization, as shown in Table 2 (u is the total time for parallel form realization).

Table 2. Determined parallel plan based on the result of GSA execution.

Time Resource	1	2	3	4
S_1	A_1	A_2	A_8	A_5
S_2	-	A_7	A_3	A_6
S_3	-	-	A_4	-
	⟷			
			u	

Function PATHS—evaluation of the reachability and determining all paths in the investigated algorithmic structure with argument matrix L (the program code and experimental results are shown in Figure 7).

```
[0]   PATHS  L
[1]   'ENTER THE VECTOR OF FINAL TASKS:'
[2]   F←⍴(Q←0)
[3]   F←⍞
[4]   N←(⍴L)[1]
[5]   SA←(N  N)⍴ 0
[6]   NF←⍴F
[7]   VL←(N)⍴0
[8]   K←1
[9]   ET1:VL←L[;F[K]]
[10]  ET2: →((+/VL)=0)/ET6
[11]  J←P←F[K]
[12]  SA[Q←Q+1;J]←1
[13]  I←0
[14]  ET3: →(VL[I←I+1]=0)/ET3
[15]  VL[I]←0
[16]  ET4: P←I,P
[17]  SA[Q;I]←1
[18]  J←I
[19]  I←1
[20]  ET5: →(L[I;J]≠0)/ET4
[21]  → (N≥I←I+1)/ET5
[22]  'PATH ',⍕Q,' IS :',⍕P
[23]  →ET2
[24]  ET6: →(NF≥K←K+1)/ET1
[25]  'MATRIX IN FORMAT SA[PATH;TASK]:'
[26]  SA
```

(a)

```
   PATHS  L
ENTER THE VECTOR OF FINAL TASKS:
⍞:
   4   5   6   7
PATH   1   IS :   1   2   4
PATH   2   IS :   1   2   5
PATH   3   IS :   1   2   8   5
PATH   4   IS :   1   2   6
PATH   5   IS :   1   2   3   6
PATH   6   IS :   1   7

MATRIX IN FORMAT SA[PATH;TASK]:
1   1   0   1   0   0   0   0
1   1   0   0   1   0   0   0
1   1   0   0   1   0   0   1
1   1   0   0   0   1   0   0
1   1   1   0   0   1   0   0
1   0   0   0   0   0   1   0
```

(b)

```
S1 :   <A1 , A2 , A4 >
S2 :   <A1 , A2 , A3 >
S3 :   <A1 , A2 , A8 , A5>
S4 :   <A1 , A2 , A5 >
S5 :   <A1 , A2 , A3 , A6>
S6 :   <A1 , A7 >
```

(c)

Figure 7. Program function PATHS—(**a**) program code; (**b**) execution; (**c**) defined paths.

Let us consider a deterministic homogeneous model for the set $A = \{A_1, \ldots, A_{14}\} = \{1, \ldots, 14\}$, whose GSA is described below by the existing arcs $<A_i \rightarrow A_j> \equiv$ "*i-j*":

1-2, 1-3, 1-4;	4-7, 4-10;	8-11;
2-5, 2-8;	6-9;	9-11;
3-2, 3-6, 3-12;	7-9, 7-12;	10-9; 10-13; 10-14.

After applying the ranking procedure, we determined five layers (height of the parallel form) for the ordered GSA, which are

$$G^* = \{\{1\}^1; \{3, 4\}^2; \{2, 6, 7, 10\}^3; \{5, 8, 9, 12, 13, 14\}^4; \{11\}^5\}$$

The maximum width of the parallel form was 6, determined by the maximum power of a layer in G * (layer '4'). These parameters determined the optimal environment for the realization of the parallel form with $|S| = m = 6$ processors with a minimum total time for full implementation $u = 5\tau$. The possible paths in ordered graph scheme G * (possible realizations of information processing) are summarized in Table 3.

Table 3. Defined layers and paths in the ordered GSA.

Layer Path	I	II	III	IV	V	Clock Path	1	2	3	4	5
1	1	–	2	5	–	1	1	2	5	–	–
2	1	–	2	8	11	2	1	2	8	11	–
3	1	3	2	5	–	3	1	3	2	5	–
4	1	3	2	8	11	4	1	3	2	8	11
5	1	3	6	9	11	5	1	3	6	9	11
6	1	3	–	12	–	6	1	3	12	–	–
7	1	4	7	9	11	7	1	4	7	9	11
8	1	4	7	12	–	8	1	4	7	12	–
9	1	4	10	9	11	9	1	4	10	9	11
10	1	4	10	13	–	10	1	4	10	13	–
11	1	4	10	14	–	11	1	4	10	14	–

4.2. Experimental Result Discussion and Examples for Application

The set of final tasks {A4, A5, A6, A7} in GSA and OGSA allowed us to determine all paths from the node A1 presenting possible realization of computer processing. The function performed success transformations to obtain an equation describing the dependence of a final node on one or more initial nodes. For example, two paths, PATH 2 and PATH 3, were determined from the node A1 to the final node A5 (see Figure 7b, which is based on the transformation).

$$A_5 = a_{25} \cdot A_2 + a_{85} \cdot A_8 = a_{25} \cdot [a_{12} \cdot A_1] + a_{85} \cdot [a_{28} \cdot A_2] = a_{25} \cdot a_{12} \cdot A_1 + a_{85} \cdot a_{28} \cdot a_{12} \cdot A_1$$

determining the paths $<A_1 \rightarrow A_2 \rightarrow A_5>$ and $<A_1 \rightarrow A_2 \rightarrow A_8 \rightarrow A_5>$. These two paths were equivalent—$\{S_2, S_3\}$, and another couple of equivalent paths were $\{S_4, S_5\}$. The presence of equivalent paths is marked by more than one "1" in the columns for the final tasks in the matrix SA (Figure 7b).

The investigated case of dispatching on the basis of the formalization discussed in the end of Section 3.3 considered homogeneous computer systems with times $t(A_i) \equiv t_i = const$ for separate resources $S_j \in S$. Communication times for exchange between processors can be ignored if there is shared memory. The weights of the nodes can be transferred along the outgoing arcs of the GSA, which allows for the use of the procedure for finding a path in a graph and defining the optimal plan to relate to the maximum path and the corresponding critical time in the study of parallel planning (maximum path length in GSA; Figure 7c). Weight or binary graphs can be applied, depending on the specific environment for the realization of the parallel processes.

An example graphical description of a dispatch plan for basic parameters S = {1,2,3}, A = {1,2,3,4,5,6,7}, and T = {4,2,2,5,4, 8.5} is given in Figure 8. The applied criteria for dispatching were "general minimization of the time for completion of the set of tasks". The figure shows the ordered graph scheme formed by the procedure GSA (Figure 5). Four layers were defined, forming information-independent subsets of nodes with corresponding weights t_i (i = 1 ÷ 7): {$1^{(4)}$; $2^{(2)}$}, {$3^{(2)}$}, {$4^{(5)}$; $5^{(4)}$, $6^{(8)}$}, {$7^{(5)}$}, where the notation is $i^{(ti)}$. The maximum processor environment for the implementation of the plan was determined by the maximum power of the layer in the ordered GSA (in this case, three processors).

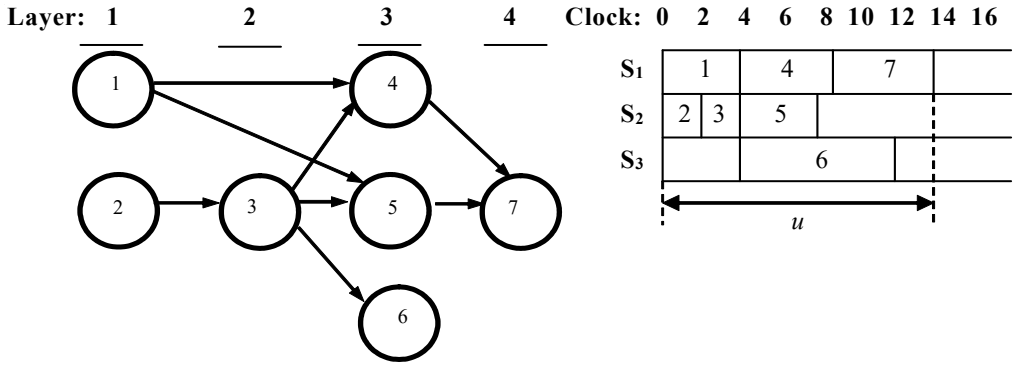


Figure 8. Graph interpretation of the dispatching plan based on results obtained from GGSA and PATH execution.

The experimental results from the case of 14 processes (Table 3) permitted the analysis of possible realizations of the parallel form at different dispatching strategies, summarized in Table 4.

Table 4. Possible generated parallel plans for realization of the graph scheme G*.

	D1(t)							**D3(t)**				
S1	1	3	2	8	11	**S1**	1	3	2	6	5	14
S2	–	4	6	9	–	**S2**	–	4	10	8	12	11
S3	–	–	7	5	–	**S3**	–	–	7	9	13	–
S4	–	–	10	12	–							
S5	–	–	–	13	–							
S6	–	–	–	14	–							
	$u = 5\tau$						$u = 6\tau$					
	D2(t)							**D4(t)**				
S1	1	3	2	8	11	**S1**	1	3	2	6	5	14
S2	–	4	6	9	13	**S2**	–	4	10	8	12	11
S3	–	–	7	5	14	**S3**	–	–	7	9	13	–
S4	–	–	10	12	–							
	$u = 5\tau$						$u = 6\tau$					
	D5(t)											
S1	1	3	2	7	8	5	13	11	1			
S2	–	4	6	10	9	12	14	–	–			
	$u = 8\tau$											

D1(t)—plan for realization of G* with priority of maximum paths;

D2(t)—modification of the plan D1(t) by taking empty clocks and being reduced while maintaining the priority of the maximum path;

D3(t)—plan for realization of G* with limit m = 3 and sequential traversal of the layers;

D4(t)—plan for realization of G* with limit $m = 3$ and selection of a process from a given layer according to its maximum repeatability in different paths;

D5(t)—plan for realization of G* with limit $m = 2$.

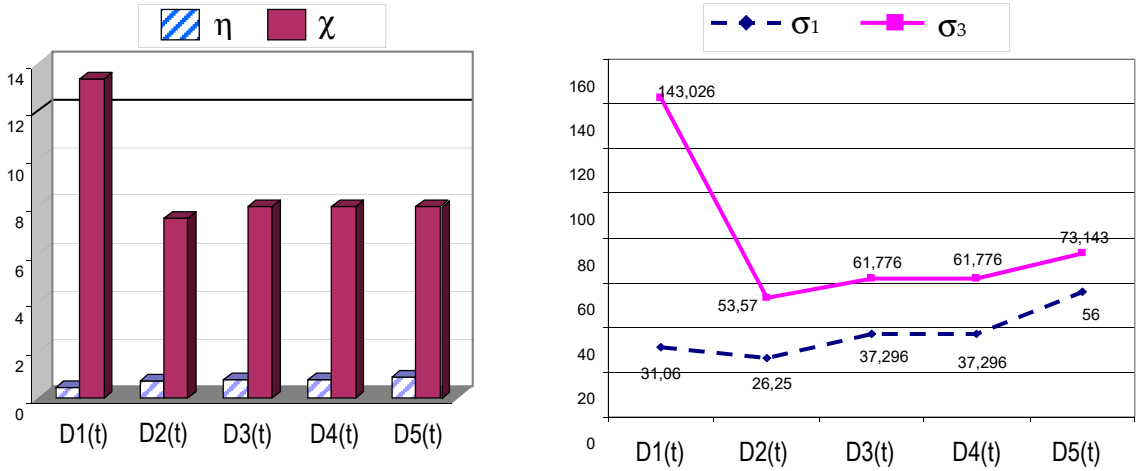
A summary of the results of the analysis is presented in Table 5, and a graphical interpretation is given in Figure 9. In addition to the two parameters $m = |S|$ and u presented above, two new characteristics η and χ were defined on the basis of the following expressions:

- Relative average resource load factor: $\eta = \frac{\sum \tau'}{\sum \tau} = \frac{\sum \tau'}{m.u}$;
- Relative weight of inefficient work (stay): $\chi = u\left(\frac{\sum \tau''}{m}\right)$;

where τ is a separate measure from the plan, which can be effective (τ') or passive (τ''). The application of different dispatching strategies allowed for the determination of the best result for different plans (marked positions in the table), but in a global aspect as an optimal plan can be determined D2(t).

Table 5. Evaluation of the parameters of determined dispatching plans.

Plan	*m*	*u*	η	χ	$\sigma_1 = u{\cdot}\eta{\cdot}\chi$	$\sigma_2 = m{\cdot}\sigma_1$	$\sigma_3 = (u{\cdot}\chi)/\eta$
D1(t)	6	**5**	0.466	13.33	31.06	186.36	143.026
D2(t)	4	**5**	0.7	**7.5**	**26.25**	**105**	**53.57**
D3(t)	3	6	0.777	8	37.296	111.88	61.776
D4(t)	3	6	0.777	8	37.296	111.88	61.776
D5(t)	**2**	8	**0.875**	8	56	112	73.143



(**a**) assessments for η and χ

(**b**) assessments for σ1 and σ3

Figure 9. Graphical interpretation of assessments.

5. Conclusions

This article discusses problems related to the mathematical formalization of objects, processes, and structures in the computer field for the purposes of research, analysis, and evaluation of parameters of information services. The proposed approach allows automation of the evaluation of certain characteristics of computer processing, presented as a sequence of relatively independent processes. To solve the set goals, we proposed a formalized technological procedure for mathematical model investigation, which had a sufficiently wide field of application, both in studying the implementation of complex software environments and in dispatching various independent processes (homogeneous and heterogeneous). In particular, the basis of the carried out investigation included two main parts—developing an approach for transforming a graph-formalized scheme of structure of algorithms (processes) and automation of the determination of possible sequences of executed processes (all paths

in the ordered graph scheme). This allowed for the analysis of selected characteristics of execution and of the dispatching of a set of processes in computer structure—for example, time parameters such as minimum and maximum execution time, and consecutive calls of procedures (algorithms). The choice of the program language was made due to its possibility for parallel calculations and direct work with two-dimensional structures as variables. Program modules for the TryAPL2 operating environment were developed, allowing for the organization of experiments for investigation of developed formal mathematical models. Program experiments were performed, and some experimental results were presented and discussed. The applicability of the designed program tools was extended by additional examples for investigation of dispatching in a computer environment for execution of a sequence of processes. Characteristics for evaluation of the efficiency of the dispatching were defined and used for calculation and comparison of assessments to select the best plan in each of the evaluated dispatch strategies.

The research presented in this article is mainly related to homogeneous processes developing in a homogeneous environment. This can be continued in further research by analyzing processes in micro and macro levels, including in heterogeneous computer spaces, parallel structures, and distributed computer environments. For example, when studying process dispatching in a heterogeneous structure, it will be necessary to determine the time parameters $t(A_i)$ for execution, which in the formalization can be presented by scalar weights of the graph nodes. In this respect, the further research will be directed to an extension of the model investigation to application of a probabilistic approach, which is typical for computer processes. This will be well supported by the capabilities of the software environment TryAPL2 for presentation and executions of stochastic processes.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Bhimani, J.; Ningfang, M.; Leeser, M.; Yang, Z. New performance modeling methods for parallel data processing applications. *ACM Trans. Model. Comput. Simul.* **2019**, *29*, 15. [CrossRef]
2. Habeeb, R.A.A.; Nasaruddin, F.; Gani, A.; Hashem, I.A.T.; Ahmed, E.; Imrah, M. Real-time big data processing for anomaly detection: A survey. *Int. J. Inf. Manag.* **2019**, *45*, 289–307. [CrossRef]
3. Mao, H.; Schwarzkopf, M.; Venkatakrishnan, S.B.; Meng, Z.; Alizadeh, M. Learning scheduling algorithms for data processing clusters. In Proceedings of the ACM Special Interest Group on Data Communication (SIFCOMM'19), Beijing, China, 19–23 August 2019; pp. 270–288. [CrossRef]
4. Lagrange, A.; Fauvel, M.; May, S.; Dobigeon, N. Hierarchical Bayesian image analysis: From low-level modeling to robust supervised learning. *Pattern Recognit.* **2019**, *85*, 26–36. [CrossRef]
5. TechPats. Firmware Analysis. Available online: https://www.techpats.com/technology/systems-and-software/firmware-analysis/ (accessed on 5 October 2020).
6. Feng, B.; Mera, A.; Lu, L. P2IM: Scalable and Hardware-independent Firmware Testing via Automatic Peripheral Interface Modeling. In Proceedings of the 29th USENIX Security Symposium, Boston, MA, USA, 12–14 August 2020; Available online: https://www.usenix.org/system/files/sec20spring_feng_prepub_0.pdf (accessed on 5 October 2020).
7. Zhukoskyy, V.; Zhukovska, N.; Vlasyuk, A.; Safonyk, A. Method of forensic analysis for compromising carrier-lock algorithm on 3G modem firmware. In Proceedings of the 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), Lviv, Ukraine, 2–6 July 2019; pp. 1179–1182. Available online: https://ieeexplore.ieee.org/abstract/document/8879941/authors#authors (accessed on 5 October 2020). [CrossRef]
8. Baron, M. *Probability and Statistics for Computer Scientists*, 3rd ed.; CPC Press, Taylor & Francis Group: Boca Raton, FL, USA, 2019; pp. 135–170.
9. Lvovich, Y.E.; Tishukov, B.N.; Preobrazhenskiy, A.P.; Kravets, O.J. Complex-structured objects optimization during modelling on the population algorithms adaptation basis. *Int. J. Inf. Technol. Secur.* **2019**, *11*, 41–50.

10. Audrito, G.; Viroli, M.; Damiani, F.; Pianini, D.; Beal, J. A higher-order calculus of computational fields. *ACM Trans. Comput. Logic.* **2019**, *20*, 5. [CrossRef]
11. Liu, K.; Koyuncu, D.; Bissyandé, T.F.; Kim, D.; Klein, J.; Le Traon, Y. You cannot fix what you cannot find! An investigation of fault localization bias in benchmarking automated program repair systems. In Proceedings of the 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), Xi'an, China, 22–27 April 2019; pp. 102–113. Available online: https://ieeexplore.ieee.org/document/8730164 (accessed on 5 October 2020). [CrossRef]
12. Harrison, M.D.; Freitas, L.; Drinnan, M.; Campos, J.C.; Masci, P.; Costanzo, M.; Whitaker, M. Formal techniques in the safety analysis of software components of a new dialysis machine. *Sci. Comput. Program.* **2019**, *175*, 17–34. [CrossRef]
13. Fregnan, E.; Baum, T.; Palomba, F.; Bacchelli, A. A survey on software coupling relations and tools. *Inf. Soft Technol.* **2019**, *107*, 159–178. [CrossRef]
14. Goryachko, V.V.; Choporov, O.N.; Preobrazhenskiy, A.P.; Kravets, O.J. The use of intellectualization management decision-making in the interaction of territorially connected systems. *Int. J. Inf. Technol. Secur.* **2020**, *12*, 87–97.
15. Madera, A.G. Hierarchical method for mathematical modeling of stochastic thermal processes in complex electronic systems. *Comput. Res. Model.* **2019**, *11*, 613–630. [CrossRef]
16. El Bhih, A.; Ghazzali, R.; Ben Rhila, S.; Rachik, M.; El Alami Laaroussi, A. A discrete mathematical modeling and optimal control of the rumor propagation in online social network. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 4386476. [CrossRef]
17. Abassian, A.; Safi, F.; Bush, S.; Bostic, J. Five different perspectives on mathematical modeling in mathematical education. *Inv. Math. Learn.* **2020**, *12*, 53–65. [CrossRef]
18. Atlasov, I.V.; Bolnokin, V.E.; Kravets, O.J.; Mutin, D.I.; Nurutdinov, G.N. Statistical models for minimizing the number of search queries. *Int. J. Inf. Tech. Secur.* **2020**, *12*, 3–12.
19. Romansky, R. A formal approach for modelling and evaluation in the field of computing. *Int. Trans. Electr. Electron. Comm. Eng.* **2012**, *2*, 1–7.
20. Wortmann, A.; Barais, O.; Combemale, B.; Wimmer, M. Modeling languages in Industry 4.0: An extended systematic mapping study. *Soft Syst. Model.* **2020**, *19*, 67–94. [CrossRef]
21. Engel, S. Writing circuit histories. *Fast Capital.* **2018**, *15*, 19–30. [CrossRef]
22. Romansky, R.P. Mathematical Formalization and Investigation by Modeling of Structures and Processes for Information Servicing. Ph.D. Dissertation, Technical University of Sofia, Sofia, Bulgaria, 12 March 2013. (In Bulgarian).