UDC 004.81
DOI: 10.15587/1729-4061.2026.356307

CREATING A POLYMODEL FRAMEWORK FOR THE CONSTRUCTION OF INTELLIGENT DECISION SUPPORT SYSTEMS

Nina Kuchuk
Corresponding author
Doctor of Technical Sciences, Professor
Department of Computer Engineering and Programming
National Technical University "Kharkiv Polytechnic Institute"
Kyrpychova str., 2, Kharkiv, Ukraine, 61002
E-mail: nina_kuchuk@ukr.net
ORCID: https://orcid.org/0000-0002-0784-1465
Leonid Artushin
Doctor of Technical Sciences, Professor, Chief Researcher*
ORCID: https://orcid.org/0000-0002-7488-7244
Yurii Zhuravskyi
Doctor of Technical Sciences, Professor
Department of Computer Technology in Medicine and Telecommunications
Zhytomyr Polytechnic State University
Chudnivska str., 103, Zhytomyr, Ukraine, 10005
ORCID: https://orcid.org/0000-0002-4234-9732
Iraida Stanovska
Doctor of Technical Sciences, Professor
Department of Advanced Mathematics and Systems Modelling
Odesa Polytechnic National University
Shevchenka ave., 1, Odesa, Ukraine, 65044
ORCID: https://orcid.org/0000-0002-5884-4228
Oleksii Kononov
Doctor of Technical Sciences, Associate Professor, Chief Researcher*
ORCID: https://orcid.org/0000-0003-2267-9109
Nadiia Protas
PhD, Associate Professor
Department of Information Systems and Technologies
Poltava State Agrarian University
Skovorody str., 1/3, Poltava, Ukraine, 36003
ORCID: https://orcid.org/0000-0003-0943-0587
Serhii Shostak
PhD, Associate Professor
Department of Higher and Applied Mathematics
National University of Life and Environmental Sciences of Ukraine
Heroiv Oborony str., 15, Kyiv, Ukraine, 03041
ORCID: https://orcid.org/0000-0003-1234-1024
Serhii Neronov
PhD, Senior Lecturer
Department of Computer Science and Information Systems
Kharkiv National Automobile and Highway University
Yaroslava Mudroho str., 25, Kharkiv, Ukraine, 61002
ORCID: https://orcid.org/0000-0003-2381-1271
Anton Nikitenko
PhD, Deputy Head of Department
Department of Operational Art
National Defence University of Ukraine
Povitrianykh Syl ave., 28, Kyiv, Ukraine, 03049
ORCID: https://orcid.org/0000-0003-0015-4440
Andrii Veretnov
PhD, Leading Researcher
Research Department
Central Scientific Research Institute of Armament and Military Equipment of Armed Forces of Ukraine
Povitrianykh Syl ave., 28, Kyiv, Ukraine, 03049
ORCID: https://orcid.org/0000-0003-0160-7325
*State Research Institute of Aviation
Kazarmenna str., 6, Kyiv, Ukraine, 01135

Intelligent decision support systems (IDSS) are the object of the study. The research problem is to improve the accuracy of the mathematical description of the process of processing heterogeneous data in IDSS. The subject of the study is a mathematical description of the processes of processing heterogeneous data in IDSS. The proposed polymodel complex for the construction of IDSS solutions, which allows:

– systemically represent the relationship between IDSS construction models in the course of their calculation and computing tasks;

– simulate the process of functioning of the IDSS, due to the use of an algebraic (formal) approach to object-oriented modeling and design of the IDSS;

– determine the rational tactical and technical indicators of the IDSS for solving specific calculation and computing tasks, due to the multilevel description of the order of construction of the IDSS;

– make the transition from one type of data representation in IDSS to another due to the presence of appropriate mathematical transformations;

– multidimensional to describe the process of processing heterogeneous data in IDSS, due to the use of a multidimensional matrix model of IDSS data;

– approach the solution of computational-calculation tasks in IDSS by using an interconnected set of mathematical models of IDSS construction;

– formalize the process of constructing IDSS, which allows combining IDSSs using different algorithmic and software. The disadvantages of the proposed polymodel complex include the need to adapt the mathematical apparatus to the specific operating conditions of the IDSS.

The proposed polymodel complex should be used for the construction of IDSS to solve general and specialized calculation tasks, as well as to solve the task of integrating various types of IDSS

Keywords: system modeling, data representation, decision-making, multidimensionality of data description

Received 16.01.2026
Received in revised form 19.03.2026
Accepted 30.03.2026
Published 30.04.2026

How to Cite: *Kuchuk, N., Artushin, L., Zhuravskyi, Y., Stanovska, I., Kononov, O., Protas, N., Shostak, S., Neronov, S., Nikitenko, A., Veretnov, A. (2026). Creating a polymodel framework for the construction of intelligent decision support systems. Eastern-European Journal of Enterprise Technologies, 2 (4 (140)), 26–38. https://doi.org/10.15587/1729-4061.2026.356307*

1. Introduction

Intelligent decision support systems (IDSS) are the basis for building modern information systems, automated systems, and computing systems for various functional purposes [1–3]. IDSS is used to solve a wide range of tasks in various areas [4, 5]:

– intellectualization of decision support processes by decision makers;

– increasing the reliability of decisions made due to multifactorial, multi-criteria, and multi-parameter assessment of available information for analysis;

– processing of large arrays of information circulating in information systems, automated systems of various functional purposes, of different origins and units of measurement;

– forecasting the development of events, as well as their consequences, etc.

The basis of any IDSS is mathematical support, which is later transformed into algorithmic and software for performing the tasks set before them [6–8].

This prompts the search for new solutions for the development of mathematical support for modern IDSS to increase the efficiency of their assigned tasks [9].

One of these approaches is the use of algebraic models of the functioning of the IDSS, which will allow to substantiate approaches to data approximation, optimize the use of IDSS resources, and carry out appropriate formalizations of heterogeneous data processing processes in the IDSS [10, 11].

Therefore, the study devoted to the development of new approaches to the construction of IDSS is relevant.

2. Literature review and problem statement

Work [12] presents an approach focused on searching for hidden information in large data sets. The method is based on analytical baselines, variable reduction, rarefied feature detection, and rule formation. The disadvantages of this method include the impossibility of taking into account different decision-making strategies, and the lack of taking into account the type of uncertainty of the initial data. In the study, not enough attention was paid to the principles and procedure of calculations, their effectiveness according to a certain criterion, and a comparative assessment of the specified approach was not carried out in comparison with the known ones.

Works [13, 14] provide an approach to the transformation of information models of objects to their equivalent structural models. This mechanism is designed to automate the necessary conversion, modification, and addition operations during such information exchange. The disadvantages of this approach include the impossibility of assessing the adequacy and reliability of the information transformation process, as well as carrying out appropriate corrections of the obtained models. In the study, not enough attention was paid to the principles and procedure of calculations, their effectiveness according to a certain criterion, and a comparative assessment of the specified approach was not carried out in comparison with the known ones.

In work [15], a method of fuzzy hierarchical evaluation is proposed, which allows for an assessment of the quality of library service. The disadvantages of the specified method include the impossibility of assessing the adequacy and reliability of the assessment and determining the assessment error accordingly. In the study, not enough attention was paid to the principles and procedure of calculations, their effectiveness according to a certain criterion, and a comparative assessment of the specified approach was not carried out in comparison with the known ones.

In work [16], an analysis of the 30 most common Big Data algorithms was carried out. It was established that the analysis of large data sets should be carried out layer by layer, take place in real time, and be able to self-learn, find a solution in different directions, and take into account the noise of the data.

Works [17, 18] present approaches for evaluating various types of data for support and decision-making systems based on the clustering of a basic set of input data, after which system training takes place based on the analysis. However, given the static architecture of artificial neural networks, an accumulation of error occurs.

In the work [19], a comparative analysis of existing decision support technologies was carried out, namely: the method of analyzing hierarchies, neural networks, the theory of fuzzy sets, genetic algorithms, and neuro-fuzzy modeling. The advantages and disadvantages of these approaches are indicated. For the tasks of assessing the state of hierarchical systems in conditions of risk and uncertainty, the use of the theory of artificial neural networks and gradient algorithms is justified.

In work [20], approaches to the structural-target analysis of the development of weakly structured systems are developed. At the same time, the problem is defined as the inconsistency of the existing state of the weakly structured system with the necessary one. At the same time, the disadvantages of the proposed approaches include the problem of local optimum, the lack of consideration of the system's computing resources, as well as the inability to conduct searches in several directions.

Analyzing scientific research [12–20] showed that the common shortcomings of the above-mentioned research are:

– not enough attention is paid to the effectiveness of the mathematical apparatus that forms the basis of their work;

– there is no systematic consideration of the mathematical apparatus, its advantages and disadvantages compared to known ones;

– there is no order of transition from one principle of data submission to another in the IDSS;

– using only one type of mathematical apparatus to solve computational tasks in IDSS.

Taking into account the above, all this allows to state that it is expedient to conduct a study devoted to the development of mathematical and algorithmic support for the construction of intelligent decision support systems.

3. The aim and objectives of the study

The aim of the study is to develop a polymodel complex for building intelligent decision support systems. This will make it possible to increase the reliability of the assessment of the state of objects with a given efficiency and the development of subsequent management decisions. This will make it possible to develop software for intelligent decision support systems.

To achieve the aim, the following objectives were set:

– cite the basic concepts and definitions of universal algebra and algebraic systems in IDSS and develop an intuitive approach to object-oriented IDSS modeling;

– propose an algebraic (formal) approach to object-oriented modeling and IDSS design;

– provide an object-oriented approach to the development of IDSS data models and to develop a multidimensional matrix model of IDSS data;

– establish the correspondence of set-theoretic and multidimensional-matrix models of IDSS data;

– develop an axiomatic approach to the formalization of data models for IDSS.

4. Materials and methods

The object of the study is IDSS. The subject of the study is a mathematical description of the processes of processing heterogeneous data in IDSS.

The hypothesis of the study is the possibility of increasing the accuracy of transformations during the processing of heterogeneous data in IDSS.

The study is based on algebraic models for formalizing the structure of heterogeneous data processing tasks in IDSS, finding optimal strategies for heterogeneous data processing, and forecasting the results of their processing under conditions of limitations.

The limitations adopted in this study are that only the identified destabilizing factors, the distribution law of which is known, are taken into account during the modeling. To simplify modeling and calculations, it is accepted that IDSS performs typical calculations and computing operations under conditions of constant influence of destructive factors.

5. Polymodel complex for building intelligent decision support systems

5. 1. Basic concepts and definitions of universal algebraic systems, an intuitive approach to object-oriented modeling

As the basis of object-oriented models of IDSS, which will allow formalizing the representation of data and their transformation operations, concepts such as universal algebra and universal algebraic system are used in the future, the known definitions of which are given below.

Definition 1. Let A – some nonempty set. Partially defined function $y = F(x_1, ..., x_n)$, $(y, x_1, \ldots, x_n) \in A$ called n-ary partial operation on A. If the function is defined everywhere, they talk about n- ary operation.

Definition 2. System $U_A = <A, \Omega>$, consisting of the main set A and the set of partial operations defined on it $\Omega = \{F_s^{n_s}\}$ ($s = 1, 2, ...$), it is called a partial universal algebra with a signature Ω.

Definition 3. Universal algebras U_A and U_B, in which the signatures are specified, respectively Ω and Ω' they are called the same type. This becomes possible if such a one-to-one correspondence between signatures can be established Ω and Ω', in which any operation $F \in \Omega$ and the corresponding operation $F' \in \Omega'$ will be n-ary with the same n.

Let two universal algebras of the same type be given $U_A = <A, \Omega>$ and $U_B = <B, \Omega>$ with basic sets A and B.

Definition 4. Display $\varphi: A \to B$ it is called a homomorphic mapping of algebra U_A in algebra U_B, if for any elements of IDSS $a_1, \ldots, a_n \in A$ and arbitrary n-ary operations $F \in \Omega$ the ratio is performed $\varphi(F(a_1, ..., a_n)) = F(\varphi(a_1), ..., \varphi(a_n))$, where $\varphi(a_i) = b_i$ and $b_i \in B\left(i = 1, \ldots, n\right)$. Algebras U_A and U_B they are called homomorphic.

If between the main sets A and B an unambiguous correspondence is established, then the display φ it is called an isomorphic mapping, and algebras U_A and U_B they are called isomorphic.

Definition 5. Let $\pi(x_1, ..., x_n)$, $x_1, \ldots, x_n \in A$ – n-local predicate, $\Pi = \{\pi_s^{n_s}\}$ ($s = 1, 2, ...$) – predicate signature. System $U_A = <A, \Pi, \Omega>$ it is called a universal algebraic system.

Homomorphic and isomorphic dependencies can also be established between universal algebraic systems. For this, it is necessary to establish a correspondence between the signatures of the predicates, similar to the one established between the signatures of the IDSS operations.

Cases where a single main set can be dispensed with when constructing a formal model of a task solved by IDSS do not occur often, since objects in the subject area have a complex structure, for the description of which many different types of parameters are used. Multibasic algebras are used to model such subject areas.

Definition 6. System $U_M = <M, \Omega>$, consisting of a family of basic sets $M = \{A_\alpha\}$ ($\alpha = 1, 2, ...$) and signatures Ω operations defined on the family M. Each in this family n-ary operation with Ω is a mapping of the Cartesian product n sets from the family M in the plural from the same family $A_{\alpha_1} \times \ldots \times A_{\alpha_n} \to A_{\alpha_r}$, it is called a polybasic algebra.

Definition 7. System, $U_M = <M, \Pi, \Omega>$ where Π – signature n-local predicates $\pi : A_{\alpha_1} \times \ldots \times A_{\alpha_n} \to \{0,1\}$, it is called a polybasic algebra system.

Homomorphic and isomorphic mappings can be established between polybasic algebras and algebraic systems, just as between monobasic ones. The following are examples of some frequently encountered data types that can be represented as polybasic algebraic systems.

Example 1. Let $U_S = <S, Z_0; \Pi; \Omega>$ – a system consisting of a set of lines S and sets of nonnegative integers Z_0. Signature Ω consists of operations:

– Ⓛ: $S \to Z_0$ – line length;

– Ⓟ: $S \times S \to Z_0$ – substring position s_1 in line s_2;

– ?+?+: $S \times S \to S$ – concatenation (clutching) of rows s_1 and s_2;

– Ⓒ: $S \times Z_0 \times Z_0 \to S$ – selection from a line s_1, starting from the position i, substring s_2 lengths l.

Predicate signature Π may include predicates:

– π_1: "line s – empty" (does not contain any character);

– π_2: "line s_1 precedes the line s_2".

Example 2. Ordinary matrix algebra is very important from the point of view of mass data processing in IDSS for example, a universal two-basic algebraic system $U_M = <M, R, \Pi, \Omega>$, where M – a set of matrices, and R – a set of real numbers (matrix elements). Signature Ω universal two-basic algebraic system U_M it looks like this:

– $+: R \times R \to R$ – additive operation on matrix elements;

– $\times: R \times R \to R$ – multiplicative operation on matrix elements;

– $?'?': M \to M$ – matrix transpose;

– $\times: R \times M \to M$ – multiplying a matrix by a number;

– $+: M \times M \to M$ – matrix sum;

– $\times: M \times M \to M$ – product of matrices;

– Ⓓ: $M \to R$ – matrix determinant.

Predicate signature Π can contain predicates such as a "matrix M degenerate" or "matrix M_1 can be multiplied by the matrix M_2".

Next, polybasic algebraic systems will be used as an apparatus for building data models in IDSS, the properties of which satisfy the tasks set in the work.

The basis of the object-oriented approach in IDSS is the concept of abstract data type (ADT).

Definition 8. Intuitively, ADT is a construction of data representation in IDSS, in which the following elements are grouped into one concept:

– the set of operations (actions) in IDSS;

– the set (one or more) of objects to which these operations apply;

– protection, or the ability of the relevant IDSS to protect the internal representation of the ADT from actions not explicitly specified in its definition (that is, to hide it from the ADT user).

Definition 9. In IDSS ADT is a two-part design:

– interface containing the name of the conditioned ADT, the names of the operations indicating their argument types and values;

– descriptions of the operations and objects with which these operations work.

This description is called concrete, in contrast to the abstract description made by means of a higher level. A specific description is also called an ADT implementation or presentation. Thanks to protection, only the names listed in the interface are available; that is, they can be used by other components of the program external to the ADT.

In the future, algebraic definitions of ADT and specific descriptions for each ADT will be used in the creation of IDSS data models. The ADT interface is a list of variables that take values on the main ADT sets and operations defined on these sets and take values in one of them. In the implementation of ADT, variables are assigned names, and their types are determined. Operations are matched with software implementation (procedures and functions), consisting of two parts:

– declarative, in which each function implementing the operation is given a name and given a description of its interface (a list of formal parameters is given);

– imperative, which is the body of a partial algorithm that implements this function.

To solve the tasks set in the work, the conjugation of data models and calculation models by the method of establishing correspondence between these models, the intuitive definition of ADT is not enough. Therefore, a strict definition of ADT must be used.

5. 2. Algebraic (formal) approach to object-oriented modeling and design

From an intuitive definition, it follows that ADT combines data sets in IDSS and operations on them under one name with which it is possible to set a description of the properties and procedures of transformation of a real object from a certain subject area, that is, build its formal model. The concept of a multi-basic algebraic system is used to clearly and unambiguously define ADT.

Definition 10. Abstract data type IDSS (object or class) – a universal monobasic or multibasic algebraic system.

This definition allows to consider any, including monobasic, universal algebras and algebraic systems, as ADTs. Despite its brevity, it fully corresponds to the intuitive definition. The following example clearly illustrates this.

Example 3. When solving various computational and logical tasks, such as finding the shortest paths, determining the availability of graph vertices, and disassembly, the universal single-base algebra system – monoid is often used to determine the data.

A monoid is a set M, on which the binary operation is specified (*), and two conditions are met:

– for any three elements $x \in M,\ y \in M,\ z \in M\ (x*(y*z)) = ((x*y)*z$ – operation associativity;

– there is such an element $e \in M,$ what is called a neutral element that for any $x \in M,\ x*e = x = e*x.$

The monoid is designated as a triad of the species $U_M =< M;*;e >$. The monoid defined in this way can be considered as a basic ADT, that is, the highest level of abstraction, to which different implementations correspond, each of which corresponds to a specific task or a specific class of tasks:

– $U_R =< R;+;0 >$ – the set of real numbers with addition operation;

– $U_R =< R;\times;1 >$ – the set of real numbers with the multiplication operation;

– $U_R =< R;\min;+\infty >$ – a set of positive real numbers with the operation of finding the minimum of two numbers and a neutral element $(+\infty)$, which in software implementations usually corresponds to the largest value in the type;

– $U_B =< B;\vee;0 >$ – plural $B = \{0, 1\}$ with disjunction operation;

– $U_S =< S;+;\varnothing >$ – the set of character strings of arbitrary length with the operation of concatenating (coupling) strings and the neutral element "empty string" (a string that does not contain any character).

The encapsulation property is satisfied because in each implementation of the basic ADT, the main set is precisely indicated, and the operation is defined as algebraic, that is, closed on the main set.

All the above implementations of the monoid inherit the properties inherent in the basic ADT; that is, the inheritance property is also satisfied.

Finally, for the operations of two implementations of the monoid, the sign (+) is used to denote two operations of different content (compilation of real numbers and concatenation of strings). That is, the property of operations polymorphism is satisfied.

Example 4. Of the many operations for constructing an example of a universal polybasic algebraic system, only one – the compute method is chosen. This provides simplicity of example, but does not limit commonality. The operation is implemented by a function that is defined on the following sets:

– the set of all tables T;

– the set of lines of a special type S_E, containing expressions based on aggregate functions (sum, mean, maximum and similar);

– the set of lines of a special type S_F, containing logical expressions – table row filters.

Since the type of the result of the operation is determined by the type of the result of calculating the expression from S_E, to ensure commonality, it is given by type *object*. That is, the result can be given a value of any type specified by the user. It can be assumed that the type *object* corresponds to the concept of a universal set U, that is, a set fixed within the solvable problem and containing as elements all the objects considered in this theory. Then the operation can be defined as a function $Compute : T \times S_E \times S_F \to U.$ Class DataTable defined as a multibasic universal algebraic system $U_{DT} =< T;S_E;S_F;U;Compute;\Pi >$. The definition of predicates depends on the task being solved. For example, a "filter expression acquires a truth value on more than half of the rows of the table".

Examples 3 and 4 clearly demonstrate the correspondence of two definitions of ATD: intuitive and strict (algebraic).

5. 3. An object-oriented approach to model development, a multidimensional-matrix data model

To develop IDSS data models that meet the requirements, namely: compliance with high-level data models and calculation models, procedurality, parallelism of algebraic operations, possibility of optimization of requests, and object orientation, it is necessary to choose basic ADTs.

The specified ADTs must have properties that will be sufficient so that, using the inheritance mechanism, it would be possible to build the necessary universal algebras or algebraic systems for use as data models.

Among the set of arbitrary ADTs, it is proposed to consider a specific ADT, hereinafter called an abstract algebraic machine.

Definition 11. An abstract algebraic machine is a two-basic algebraic system of the form $E = <S, T, \Omega, \Pi>$. Basis S it is called a structure, and the base T – type.

The structure is some construction composed of instances of this type. Examples of such structures are vectors, matrices, and graphs. The choice of structure and type is determined by the features of the solved task. Moreover, for some classes of tasks, several types can correspond to one structure. The following example illustrates this situation.

Example 5. To solve the IDSS of the tasks named in the example, finding the shortest paths, determining the accessibility of the vertices of the graph, and knotting a method based on the algorithm for calculating the transitive closure of a square matrix can be applied M.

Transitive matrix closure M it is calculated according to the following formula: $M^* = \sum_{i=1}^{K} M^i,\ M^i \neq Z$ for everyone $i \leq K$ and $M^{K+1} = Z$ where Z – zero matrix.

In this case, the abstract algebraic machine has the following form $E_M =< M; X; \Omega; \Pi >$, where X – type, and M – a set of square matrices composed of type elements.

Minimum type requirement X consists in on X two algebraic operations were defined, one of which is interpreted as additive and the other – as multiplicative.

That is, the type X there must be, according to each of these operations, at least an algebraic structure called a groupoid. In real problems, the types can be quite complex algebraic structures, such as rings and fields. The following are formal definitions and descriptions of signature operations Ω abstract algebraic machine E_M:

– $?+?+: X \times X \to X$ – additive operation on matrix elements;

– $?\times?\times: X \times X \to X$ – multiplicative operation on matrix elements;

– $?'?': M \to M$ – matrix transpose;

– $+: M \times M \to M$ – matrix sum;

– $\times: M \times M \to M$ – product of matrices;

– Ⓓ: $M \to X$ – matrix determinant.

In real tasks, IDSS in the role of type X there can be such sets as:

– in the "knotting task" – the set of nonnegative real numbers R^0, with additive addition operation and multiplicative multiplication operation of numbers, $\Omega = \{+, \times\}$;

– in the task of calculating the shortest paths in the graph –, the set of positive real numbers R^+, with the additive operation of calculating the minimum of two numbers, and the multiplicative operation of addition, $\Omega = \{\min, +\}$;

– in the task of determining the availability of vertices in the graph, the set {0, 1} with additive disjunction operation and multiplicative conjunction operation, $\Omega = \{\vee, \wedge\}$.

Operations on matrices included in the signature of operations Ω, are implemented by well-known sequential and parallel standard algorithms, which will be discussed later.

The importance of the considered example lies, first of all, in the fact that it clearly shows the presence of universal two-basic algebraic systems that have quality. Quality can be formulated as the possibility of replacing one main set, namely the type and operations on its elements, while preserving another set and algorithms that implement operations on its elements. This fact can be formulated in the form of an obvious statement.

Statement 1. Let S – structure, and $T_1, ..., T_n$ – types acceptable for this structure. Then $T_1, ..., T_n$ – universal systems of the same type (s).

The practical value of universal algebraic machines is that the essence of operations on elements of the structure S does not change when the essence of operations on type elements changes T. If types $T_1, ..., T_n$ – homomorphic or isomorphic universal algebraic systems, it becomes possible to adjust operations on the structure on the simplest type of data. The structure adjusted in this way becomes the basic ADT, from which specific ADTs (implementations) designed to solve IDSS tasks on complex data types can be generated. Next, the method of constructing binary operations on tuples is considered, without which it is impossible to construct binary operations on data aggregates that are used as elements of the structure (relationships, data, multidimensional matrices). In different data models, these tuple operations, given explicitly or implicitly, are interpreted either additively or multiplicatively. The proposed method made it possible to make explicit formal descriptions of binary operations on tuples. Next, the definition of a tuple generally accepted in mathematics is used.

Definition 12. Tuple – the final set $(t_1, ..., t_n)$ lengths n (where n – a non-negative integer), each element of which t_i belongs to some type T_i $(1 \leq i \leq n)$. Zero tuples play an important role in data models. In the future, the zero-tuple is perceived as a tuple consisting only of neutral elements of types $T_1, ..., T_n$.

Let $T_1, ..., T_n$ – the set of basic sets of universal algebras or algebraic systems, which are called simple types in programming languages. These are, for example, fixed or floating point numbers, strings, as well as types obtained from simple types by adding new operations. Let $x_1, ..., x_p, y_1, ..., y_q$, $(0 \leq p, q \leq n, p + q = n)$ – a set of variables, each of which acquires a value in one and only one of the sets $T_1, ..., T_n$.

On these sets, the system of functions is defined

$$f^j_{\alpha_1,...,\alpha_k,\beta_1,...,\beta_l}\begin{pmatrix} x_{\alpha_1},...,x_{\alpha_k}, \\ y_{\beta_1},...,y_{\beta_l} \end{pmatrix}: \begin{pmatrix} T_{\alpha_1}\times...\times T_{\alpha_k}\times \\ \times T_{\beta_1}\times...\times T_{\beta_l} \end{pmatrix} \to T_i.$$

Inequalities are performed here $1 \leq k \leq p$, $1 \leq l \leq q$, $j > 0$ and $1 \leq i \leq n$. Next, a shortened entry of these functions will be used – $f^j_{\alpha_1,...,\alpha_k,\beta_1,...,\beta_l}$.

Definition 13. Let tuple c_1 lengths p and tuple c_2 lengths q composed of variables $x_1, ..., x_p$ and $y_1, ..., y_q$ accordingly. Then the cortege c_3 lengths r, built according to the rule $(f^1_{\alpha_1,...,\alpha_k,\beta_1,...,\beta_l},..., f^r_{\alpha_1,...,\alpha_k,\beta_1,...,\beta_l})$, can be considered as the result of a binary operation on tuples c_1 and c_2 ($c_3 = c_1 * c_2$). If the function is defined on all elements of tuples c_1 and c_2, that is the designation used $f^j_{c_1,c_2}$.

The semantics of the operation on tuples (additivity or multiplicativity) is determined by the semantics of the operation, defined over a structure, the types of elements of which can be tuples c_1, c_2 and c_3.

The following example shows the construction of binary operations on tuples.

Example 6. Let S – the set of lines, a R^+ – the set of positive real numbers. Variables (attributes) A, B, C, D acquire meaning in the set S, and variables X, Y – in the plural R^+. Cortezhi c_1 and c_2 have schemes c_1 (A, B, C, X) and c_2 (B, C, D, Y). Functions $f^2_{c_1,c_2}(d) = d, (d \in D)$ they allow to build a motorcade $c_3 = c_1 \times c_2$ with scheme c_3 (A, D, Z), where $Z = X \times Y$, and takes values in the set R^+. In this way, the multiplicative operation on tuples is determined. If c_{31}, c_{32}, c_{33} – tuples with a scheme $c_3(A, D, Z)$, then functions:

$$f^1_{c_{31},c_{32}}(c_{31}.a) = c_{31}.a, (a \in A), \quad (1)$$

$$f^2_{c_{31},c_{32}}(c_{31}.d) = c_{31}.d, (d \in D), \quad (2)$$

$$f^3_{c_{31},c_{32}}\left(c_{31}.z,\, c_{32}.z\right) = c_{31}.z \times c_{32}.z, \left(z \in Z\right), \quad (3)$$

determine the additive operation on tuples $c_{33} = c_{31} + c_{32}$.

Similarly, the considered functions can be used to construct algebras of elements of IDSS structures (data and multidimensional matrices) in set-theoretic and multidimensional-matrix models.

In some models, when constructing binary operations on elements of the structure, a situation may arise when the result tuple is formed from only one operand tuple. This is impossible in a multidimensional matrix model, because there are all possible elements in the matrix, including zero tuples. But in the relational and set-theoretic model, relations and data, as a rule, do not contain strings and records consisting only of neutral elements. Therefore it is advisable to consider two more types of functions $f^j_{\alpha_1,\ldots,\alpha_k,0}(x_{\alpha_1},\ldots,x_{\alpha_k}):(T_{\alpha_1}\times\ldots\times T_{\alpha_k})\to T_j$ and $f^j_{0,\beta_1,\ldots,\beta_l}(y_{\beta_1},\ldots,y_{\beta_l}):(T_{\beta_1}\times\ldots\times T_{\beta_l})\to T_j$, which will allow to design operations that form a tuple-result from only one tuple-operand. The use of these operations ensures the unity of the formal recording of algorithms of binary operations on structures. An abbreviated entry of these functions looks like: $f^j_{0,\beta_1,\ldots,\beta_l}$, $f^j_{\alpha_1,\ldots,\alpha_k,0}$.

The proposed method defines various additive and multiplicative operations on tuples and, at the same time, obtains various universal algebras or algebraic systems. The properties of the constructed operations make up a list of axioms of these algebraic systems. Thanks to this, there is a possibility of formal construction of arbitrary universal algebraic systems of tuples in accordance with the requirements of real tasks.

The method of constructing a universal algebraic system for tuples will be used later to construct the second main set (type) in matrix algebraic machines. It will also be used to prove the isomorphism of the data models considered in the work.

The multidimensional-matrix data model considered below was proposed in order to solve the problems of increasing the efficiency of data processing in IDSS. Often, in practice, optimization of one step leads to deterioration of the characteristics of another. This happens because:

– it is difficult to efficiently construct a multidimensional structure even from data placed in analogous structures unless special algebraic operations on these structures are used;

– most modern methods of optimizing data processing processes in IDSS in various data models are based on a heuristic approach, which does not allow effective use of specific properties of multidimensional data structures and operations on them.

Solving these problems is possible based on the use of an algebra of multidimensional matrices, the properties of which allow solving the listed problems. Algebra operations of multivariate matrices are quite easily implemented on parallel computing complexes with different architectures. One of the most important properties of the algebra of multivariate matrices is the possibility of constructing optimal IDSS data processing processes based on the use of formal optimization methods, for example, fuzzy logic.

Matrices are usually understood as structured sets of elements of simple types. Next, multidimensional matrices are considered, the elements of which can belong to arbitrary data types. The main and only requirement is that two algebraic operations must be defined on these types: additive and multiplicative. This means that the types of matrix elements must be at least groupoids for each of the operations defined on them.

This approach allows to use such standard structural types as vectors and matrices to create matrices, although the use of multidimensional matrices deprives such constructions of practical meaning. In principle, different ADTs can be used as types of matrix elements. In order to achieve the goals, set in the work, it is especially important that in order to build a multidimensional-matrix data model, ADTs designed in accordance with the specified requirement for different types of tuples can be used to build a multidimensional-matrix data model.

Definition 14. Let – $i_1, \ldots, i_p$ a set of indices gaining values from 1 to n_α ($\alpha = 1, \ldots, p$) accordingly. Then p- the dimensional matrix is a collection $T = \{a_{i_1,\ldots,i_p}\}$ elements of some type, on which additive and multiplicative operations are defined.

Thus, p – the dimensional matrix contains $n_1 \times \ldots n_p$ elements. For multivariate matrices, notation is used $A = ||\, a_{i_1,\ldots,i_p}\, ||$. The algebra signature of multivariate matrices contains the unary operations of transpose, section, convolution, and binary operations of assembly and multiplication. The following are definitions of these operations.

Transposition. Matrix $A' = ||\, a_{i_{\alpha_1},\ldots,i_{\alpha_p}}\, ||$, which elements are related to those of the matrix $A = ||\, a_{i_1,\ldots,i_p}\, ||$ ratio $a_{i_{\alpha_1},\ldots,i_{\alpha_p}} = a_{i_1,\ldots,i_p}$, where $(i_1, \ldots, i_p)$ $(i_{\alpha_1},\ldots,i_{\alpha_p})$ – some permutation of the indices $(i_1, \ldots, i_p)$, it is called transposed with respect to the matrix A according to this permutation.

Visually p – the dimensional matrix can be represented as a p-dimensional parallelepiped or hypercube. Hence, the transposition operation can be interpreted as the rotation of this parallelepiped or hypercube.

Crossing. Two variants of this operation are possible. In the first, the dimension of the matrix decreases, while in the second, it remains.

Simple m-fold intersection. Let in m indices ($1 \le m \le p$) sets of indices $(i_1, \ldots, i_p)$ matrices are fixed by one value. For simplicity and without limitation of commonality, these will be assumed to be indices $(i_1, \ldots, i_m)$. $(p - m)$-dimensional matrix $||\, A_{(i_1^0,\ldots,i_m^0,i_{m+1}^0,\ldots,i_p)}\, ||$, consisting only of those elements of the matrix $A = ||\, a_{i_1,\ldots,i_p}\, ||$, in which indices $(i_1, \ldots, i_p)$ have a single fixed value $(i_1^0,\ldots,i_m^0)$, it is called simple m-multiple cross-section of the matrix A orientations $(i_1, \ldots, i_p)$.

Example 7. Let $A = ||\, a_{i_1 i_2 i_3 i_4}\, ||$ – a four-dimensional matrix, all indices of which acquire the value 1, 2. If to fix the value of two indices $i_1 = 2$ and $i_2 = 1$, then a two-time simple cross-section of the matrix is obtained A orientations (i_1, i_2), which is a two-dimensional matrix of the form:

$$A_{(i_1,i_2)} = \begin{Vmatrix} a_{2111} & a_{2112} \\ a_{2121} & a_{2122} \end{Vmatrix} \begin{pmatrix} i_1 = 2 \\ i_2 = 1 \end{pmatrix}.$$

If to fix the value of one index $i_2 = 2$, then a one-time simple cross-section of the matrix is obtained A orientations (i_2), which is a three-dimensional matrix of the form:

$$A_{(i_2)} = \begin{matrix} & i_4 \rightarrow \\ i_1, i_3 & \\ \downarrow & \end{matrix} \left\| \begin{array}{cc} a_{1211} & a_{1212} \\ a_{1221} & a_{1222} \\ \hline a_{2211} & a_{2212} \\ a_{2221} & a_{2222} \end{array} \right\| \left(i_2 = 2\right). \quad (4)$$

Intersection with fixed index values. Let be in the set of index values $(i_1, \ldots, i_m)$ $(1 \le m \le p)$ sets of indices $(i_1, \ldots, i_p)$ matrices $A = ||\, a_{i_1,\ldots,i_p}\, ||$ recorded in more than one value.

This means that any index i_k $(1 \le k \le m)$ accepts t_k $(1 < t_k < n_k)$ values from the set $(1, \ldots, n_k)$. p-dimensional matrix $A_{(i_1,\ldots,i_m)}$, consisting only of those elements of the matrix $A = ||\, a_{i_1,\ldots,i_p}\, ||$, in which indices $(i_1, \ldots, i_m)$ accepted accordingly $t_1, \ldots, t_k$ values,

called m-multiple intersection of orientation (i_1, ..., i_m) with fixed values of matrix indices A.

Obviously, that m-multiple cross-section of the matrix A can be considered as a matrix built from simple m-multiple sections.

Example 8. Let $A=||a_{i_1i_2i_3i_4}||$ – a four-dimensional matrix in which all indices take values (1, 2, 3). With the help of double sections, this matrix can be presented in the form:

$$A_{(i_2)}=\overset{i_1}{\downarrow}\left\|\begin{array}{ccc|ccc|ccc} & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ \hline & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ \hline & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{array}\right\|. \quad (5)$$

$$i_2 \rightarrow$$

Matrix A is a four-dimensional hypercube. If to fix the values of the indices $i_1=(1, 2)$ and $i_2=(2, 3)$ then as a result of performing the section operation of the matrix A turns into a four-dimensional parallelepiped of the form:

$$A_{(i_1,i_2)}=\overset{i_1}{\downarrow}\left\|\begin{array}{cc|cc} & & & \\ & & & \\ & & & \\ \hline & & & \\ & & & \\ & & & \end{array}\right\|\begin{pmatrix} i_1=(1,2) \\ i_2=(2,3) \end{pmatrix}. \quad (6)$$

$$i_2 \rightarrow$$

A combined version of this operation is also possible, when a simple cross-section is performed in the index part, and an intersection with a fixed set of index values is performed in the part. In this case, the dimension of the result matrix is less than the dimension of the operand matrix by the number of indices, according to which a simple cross-section is made.

Convolution. Let the partition of the set of matrix indices be given $A=\left\|a_{i_1,\ldots,i_p}\right\|$ on totality $l=(l_1, ..., l_k)$ and $c=(c_1, ..., c_\mu)$, $\kappa+\mu=p$. Matrix ${}^{\mu}A=\left\|a_l\right\|$, which elements are related to those of the matrix $A=\left\|a_{lc}\right\|$ ratio $a_l=\sum\limits_{(c)}a_{lc}$, called μ-rolled up matrix and is denoted ${}^{\mu}A=\left\|\sum\limits_{(c)}a_{lc}\right\|$. Breakdown indices $l=(l_1, ..., l_k)$ they are called free indices, and partition indices $c=(c_1, ..., c_\mu)$ – cell indices.

Example 9. Let $A=\left\|a_{i_1i_2i_3i_4}\right\|$ – a four-dimensional matrix, all indices of which take the value 1, 2. If $\kappa=\mu=2$, $l_1=i_1$, $l_2=i_2$ – free indices and $c_1=i_3$, $c_2=i_4$ – Kelly indices, then the matrix ${}^{2}A=\left\|\sum\limits_{(c)}a_{lc}\right\|$ looks like:

$${}^{2}A=\left\|\begin{array}{cc} \sum\limits_{i_3=1}^{2}\sum\limits_{i_4=1}^{2}a_{11i_3i_4} & \sum\limits_{i_3=1}^{2}\sum\limits_{i_4=1}^{2}a_{12i_3i_4} \\ \sum\limits_{i_3=1}^{2}\sum\limits_{i_4=1}^{2}a_{21i_3i_4} & \sum\limits_{i_3=1}^{2}\sum\limits_{i_4=1}^{2}a_{22i_3i_4} \end{array}\right\|. \quad (7)$$

Adding. The sum of two p-dimensional matrices $A=||a_{i_1,\ldots,i_p}||$ and $B=||b_{i_1,\ldots,i_p}||$ with the same sets of indices i_1, ..., i_p called p-dimensional matrix $C=||c_{i_1,\ldots,i_p}||$ with the same set of indices, the elements of which are calculated according to the formula $c_{i_1,\ldots,i_p}=a_{i_1,\ldots,i_p}+b_{i_1,\ldots,i_p}$.

Multiplication. Let the matrices $A=||a_{i_1,\ldots,i_p}||$ and $B=||b_{i_1,\ldots,i_p}||$ p and q-measured accordingly. The sets of indices of these matrices i_1, ..., i_p and i_1, ..., i_q they are divided into four groups containing, respectively κ, λ, μ and ν indices (κ, λ, μ, $\nu\geq 0$). And $\kappa+\lambda+\mu=p$, and $\kappa+\lambda+\mu=q$.

Designations are used for the obtained index groups: $l=(l_1, ..., l_k)$, $s=(s_1, ..., s_\lambda)$, $c=(c_1, ..., c_\mu)$ and $m=(m_1, ..., m_\nu)$. Then matrices A and B can be submitted in the form $A=\left\|a_{lsc}\right\|$ and $B=\left\|b_{scm}\right\|$. Group indices s and c in matrices A and B completely match. Just like in the convolution operation, partition indices c they are called Kelly. Breakdown indices s they are called scott's, and the division indices m, as are the breakdown indices l, – free.

Matrix $C=\left\|c_{lsm}\right\|$, the elements of which are calculated according to the formula $c_{lsm}=\sum\limits_{(c)}a_{lsc}\times b_{scm}$ it is called the product of matrices A and B.

The algorithm for implementing this operation is as follows:

– multiplies all pairs of elements in which the values of the indexes of the groups completely coincide s and c;

– sum all products with the same values of group indices c.

The product of multidimensional matrices is called (λ, μ)-collapsed product is denoted by ${}^{\lambda,\mu}(A\times B)$. From definition ($\lambda$, μ)-collapsed product follows that for any pair of multidimensional matrices, many different products can be constructed by selecting different values λ and μ.

The number of all possible (λ, μ)-collapsed products p-dimensional matrix A on q-dimensional matrix B it is calculated according to the formula

$$N_{p,q}=\sum_{\lambda+\mu=0}^{\min(p,q)}\frac{p!}{\lambda!\mu!(p-\lambda-\mu)!}\cdot\frac{q!}{\lambda!\mu!(q-\lambda-\mu)!}. \quad (8)$$

Example 10. The example demonstrates different options (λ, μ) – the convolved product of two three-dimensional matrices. Let a set of indices be given i_1, i_2, i_3, i_4, dimensions which ones n_1, n_2, n_3, $n_4=2$. Then three-dimensional matrices $A=\left\|a_{i_1i_2i_3}\right\|$ and $B=\left\|b_{i_2i_3i_4}\right\|$ look like

$$i_1(1,2)\rightarrow$$

$$A=\left\|\begin{array}{cc|cc} a_{111} & a_{112} & a_{211} & a_{212} \\ a_{121} & a_{122} & a_{221} & a_{222} \end{array}\right\|$$

and

$$i_1(1,2)\rightarrow$$

$$B=\left\|\begin{array}{cc|cc} b_{111} & b_{112} & b_{211} & b_{212} \\ b_{121} & b_{122} & b_{221} & b_{222} \end{array}\right\|. \quad (9)$$

Among the possible (λ, μ)-convolved products of matrices A and B will be as follows: four-dimensional matrix $C={}^{2,0}(A\times B)|^{i_2,i_3}=||c_{i_1,i_2,i_3,i_4}||$, the elements of which are calculated according to the formula $c_{i_1,i_2,i_3,i_4}=a_{i_1,i_2,i_3}\times b_{i_2,i_3,i_4}$ index values match for all i_2, i_3.

This matrix has the form:

$$C = \left\| \begin{array}{cc|cc} a_{111}\times b_{111} & a_{111}\times b_{112} & a_{121}\times b_{211} & a_{121}\times b_{212} \\ a_{112}\times b_{121} & a_{112}\times b_{122} & a_{122}\times b_{221} & a_{122}\times b_{222} \\ \hline a_{211}\times b_{111} & a_{211}\times b_{112} & a_{221}\times b_{211} & a_{221}\times b_{212} \\ a_{212}\times b_{121} & a_{212}\times b_{122} & a_{222}\times b_{221} & a_{222}\times b_{222} \end{array} \right\|. \quad (10)$$

In this case, matrix multiplication is performed without convolution, because there are no Kelly indices.

Three-dimensional matrix $C = {}^{1,1}(A\times B)\Big|_{i_2}^{i_3} = \left\| c_{i_1,i_3,i_4} \right\|$, the elements of which are calculated according to the formula $c_{i_1,i_3,i_4} = \sum_{i_2=1}^{2} a_{i_1,i_2,i_3} \times b_{i_2,i_3,i_4}$.

This matrix has the form:

$$C = \left\| \begin{array}{cc|cc} a_{111}\times b_{111} + a_{121}\times b_{211} & a_{111}\times b_{112} + a_{121}\times b_{212} & a_{211}\times b_{111} + a_{221}\times b_{211} & a_{211}\times b_{112} + a_{221}\times b_{212} \\ a_{112}\times b_{121} + a_{122}\times b_{221} & a_{112}\times b_{122} + a_{122}\times b_{222} & a_{212}\times b_{121} + a_{222}\times b_{221} & a_{212}\times b_{122} + a_{222}\times b_{222} \end{array} \right\|. \quad (11)$$

The three-dimensional matrix is obtained because the cell index i_2 when folded, it is removed from the set of indices common to both matrices.

Two-dimensional matrix $C = {}^{0,2}(A\times B)\Big|_{i_2,i_3} = \left\| c_{i_1,i_4} \right\|$, the elements of which are calculated according to the formula: $c_{i_1,i_4} = \sum_{i_2=1}^{2}\sum_{i_3=1}^{2} a_{i_1,i_2,i_3} \times b_{i_2,i_3,i_4}$.

This matrix has the form:

$$C = \left\| \begin{array}{cc|cc} a_{111}\times b_{111} + a_{111}\times b_{112} + a_{121}\times b_{211} + a_{121}\times b_{212} & & a_{211}\times b_{111} + a_{211}\times b_{112} + a_{221}\times b_{211} + a_{221}\times b_{212} & \\ a_{112}\times b_{121} + a_{112}\times b_{122} + a_{111}\times b_{221} + a_{122}\times b_{222} & & a_{212}\times b_{121} + a_{212}\times b_{122} + a_{222}\times b_{221} + a_{222}\times b_{222} & \end{array} \right\|. \quad (12)$$

The operations on multivariate matrices defined in this way make it possible to construct a two-basic algebraic system or ADT, which is called an abstract algebraic multivariate machine.

The peculiarity of building operations on matrix elements is that neutral elements of the type are clearly present in matrices.

Therefore, these operations are built based on a defined system of functions

$$f^{j}_{\alpha_1,\ldots,\alpha_k,\beta_1,\ldots,\beta_l}\left(x_{\alpha_1},\ldots,x_{\alpha_k},y_{\beta_1},\ldots,y_{\beta_l}\right).$$

Let M – be the set of multidimensional matrices, and E – be the set of their elements, which is determined for each specific subject area. Next, a list of IDSS operations included in the signature of operations of this algebraic system is given. Unary operations specify mapping $M \to M$, and binary – displays $M \times M \to M$.

Structure operations: T – transposition; S – intersection; C – convolution; + – adding; × – (λ, μ)-collapsed product.

Type operations (over structure elements): ?+?+ – additive; ?×?× – multiplicative.

In predicate signatures Π includes the following predicates:

– z: $M \to \{0, 1\}$; matrix X contains only neutral elements of the type;

– ac: $M \times M \to \{0, 1\}$; matrices X and Y compatible by addition;

– mc: $M \times M \to \{0, 1\}$; matrices X and Y compatible by multiplication.

In addition, in cases where the elements of the matrices – tuples, predicates are set on many elements of the matrices (type) $pred$, eq, $succ$: $E \times E \to \{0,1\}$, which specify binary relations of warning, equivalence and following.

Then the abstract multidimensional matrix machine is given as follows:

$$M_{am} = \left\langle \begin{array}{l} M, E; T, S, C, +, \times, ?+?+, ?\times?\times; \\ z,\ ac,\ mc, pred, eq, succ \end{array} \right\rangle. \quad (13)$$

Further, the work considers the methods of building an abstract multidimensional machine, based on the generalization of parallel algorithms that implement operations on matrices into multidimensional matrices.

5. 4. Establishing the correspondence of set-theoretic and multidimensional matrix data models

The correspondence of the operations of the set-theoretic and multidimensional model for IDSS can be established as shown in the Table 1.

Definition 15. Multidimensional matrix $A = \left\| a_{i_1,\ldots,i_p} \right\|$ it is called a logical multidimensional matrix (LMM), if its elements belong to the set {0,1} and over these, the additive disjunction operation and the multiplicative conjunction operation are defined.

Transpose and cross-section operations are independent of the type of matrix elements, and therefore their definitions remain unchanged for LMM. In the definitions of other operations, the formulas that calculate the values of the LMM elements of the operation result change.

Convolution: if $B = {}^{\mu}A$, that's it $b_l = \bigvee\limits_{(c)} a_{lc}$, where l – a set of free, and c – Kelly indices.

Addition: if $C = A + B$, that's it $c_{i_1,\ldots,i_p} = a_{i_1,\ldots,i_p} \circ b_{i_1,\ldots,i_p}$, where $\circ$ – a binary logical operation that is perceived as an additive operation; (λ, μ)-collapsed product: if $C = {}^{\lambda,\mu}(A\times B)$, that's it $c_{lsm} = \bigvee\limits_{(c)} a_{lsm} \wedge b_{scm}$.

Table 1

Correspondence of algebraic operations in set-theoretic and multidimensional-matrix models of IDSS data

Theoretically, a multiple model (data algebra)	Multidimensional matrix model (algebra of multidimensional matrices)	Operation type
Sorting	Transposition	unary
Sampling	m-multiple section	unary
Compression	Convolution	unary
Fusion of strictly ordered data	Adding	binary
Merging of unordered data	(λ, μ)-collapsed product	binary

Statement 2. Each type of data corresponds to a single LMM.

Let XK – be data strictly ordered by the set of keys $K = \{K_1, \ldots, K_p\}$. Since the set of key values is finite, it is possible to number all the values of each key and thereby match each key K_α index $i_\alpha = (1, \ldots, n_\alpha)$. Then each instance of the set of

keys $K^* = \{K_1^*, \ldots, K_p^*\}$ one and only one set of index values corresponds $(i_1^0, \ldots, i_p^0)$.

This means that between the set of all instances of the set of keys K file X_K and a set of all sets of index values $(i_1, \ldots, i_p)$ mutually unambiguous correspondence has been established. Let the instance of the set of keys $K^* = \{K_1^*, \ldots, K_p^*\}$ corresponds to a set of index values $(i_1^0, \ldots, i_p^0)$. Then the equivalence class X_{K^*} it is possible to match the LMM element $X = \left\| x_{i_1, \ldots, i_p} \right\|$, the value of which is determined by the formula:

$$x_{i_1^0, \ldots, i_p^0} = \begin{cases} 0, \text{ if } X_{K^*} = \Theta, \\ 1, \text{ if } X_{K^*} \neq \Theta. \end{cases} \quad (14)$$

Since the data X_K strictly ordered, each of its equivalence classes contains either a single definite record or a universal indefinite record. Therefore, for this method of constructing a LMM, each strictly ordered array of data corresponds to a single LMM. However, a situation is possible in which the same LMM corresponds to several different data arrays, each containing data from a different subject area. That is, the constructed mapping of the set of strictly ordered data to the set of LMM is unambiguous.

In the following three statements, it is assumed that X – the set of strictly ordered data, M – the LMM set, $\varphi: X \to M$ – unambiguous mapping of a set of strictly ordered data to a set of LMM.

Statement 3. If X_K – data strictly ordered by a set of keys $K = \{K_1, \ldots, K_p\}$, and $A = \left\| a_{i_1, \ldots, i_p} \right\|$ – its corresponding LMM $(\varphi(X_K) = A)$ and $\begin{pmatrix} K_1, \ldots, K_p \\ K_{\alpha_1}, \ldots, K_{\alpha_p} \end{pmatrix}$, $\begin{pmatrix} i_1, \ldots, i_p \\ i_{\alpha_1}, \ldots, i_{\alpha_p} \end{pmatrix}$ the same permutations of data keys and LMM indices, then φ (sort $(X_K)) = A'$, provided that sorting and transposing are performed according to given permutations of keys and indices.

This statement follows directly from the definitions of data sorting and multidimensional matrix transposition operations.

Statement 4. Let X_K – data strictly ordered by a set of keys $K = \{K_1, \ldots, K_p\}$, $A = \left\| a_{i_1, \ldots, i_p} \right\|$ – their corresponding LMM $(\varphi(X_K) = \text{A})$ and predicate $\pi(K)$ fixes the value of the keys $K_1, \ldots, K_m \in K\left(m < p\right)$, which indices correspond to $(i_1, \ldots, i_m)$ LMM A.

Then the data obtained from the data X_K as a result of applying to it a sampling operation according to the predicate $\pi(K)$, a single LMM corresponds $A_{(i_1, \ldots, i_m)}$, obtained from the matrix A m-multiple intersection of orientation $(i_1, \ldots, i_m)$, i.e. $\varphi(sel(X_K, \pi(K))) = A_{(i_1, \ldots, i_m)}$.

It is enough to consider the validity of this statement for the case when the predicate $\pi(K)$ fixes a single instance of each key $K_1, \ldots, K_m$. Then to each fixed instance K_j^* key K_j corresponds to a single value i_j^* index i_j $(j = 1, \ldots, m)$, and there is a simple one m-multiple section $A_{(i_1, \ldots, i_m)} = \left\| a_{i_1, \ldots, i_p} \right\| \begin{pmatrix} i_1 = i_1^* \\ \cdots \\ i_m = i_m^* \end{pmatrix}$ LMM $A = \left\| a_{i_1, \ldots, i_p} \right\|$.

If X_{K^*} – data equivalence class X_K, such that in an instance of the set of keys K^* key instances $K_1^*, \ldots, K_m^*$ fixed predicate $\pi(K)$, and instances of other keys – arbitrarily, this equivalence class corresponds to the LMM element A

$$a_{i_1^*, \ldots, i_m^*, i_{m+1}^0, \ldots, i_p^0} = \begin{cases} 0, \text{ if } X_{K^*} = \Theta_{K^*}, \\ 1, \text{in the other case.} \end{cases} \quad (15)$$

But this element will also belong to the simple m-multiple section $A_{(i_1, \ldots, i_m)}$. So, to all the selected data equivalence classes X_K the only elements of this section of the LMM will correspond A. However, m-multiple section can be constructed from the corresponding simple ones m-multiple section. If predicate $\pi(K)$ fixes at least one instance of each of the keys $K_1, \ldots, K_m$, then it is possible to get all the necessary simple sections, and then build the necessary LMM from them using the LMM addition operation. From what has been said, the validity of statement 4 follows.

Statement 5. Let X_K – data strictly ordered by multiple keys $K = \{K_1, \ldots, K_p\}$, $A = \| a_{i_1, \ldots, i_p} \|$ – its corresponding LMM $(\varphi(X_K) = A)$, $M = \{K_1, \ldots, K_m\}$, $(m < p)$ – the subset of a set of keys M, by what data X_M not strictly ordered. Then given $Y_M = quant(X_M)$ a single convolution of LMM corresponds A, i.e. $\varphi(quant(X_M)) = {}^{\mu}A$.

According to the definition of the compression operation, equivalence classes of strictly ordered data $quant$ (X_M) they come out according to the formula $Y_{M^*} = f(X_{M^*})$, where $f(X_{M^*})$ – a function implementing a group operation on each equivalence class corresponding to an instance of a set of keys M^*. That is, the set of data records X_K, what keys $K_1, \ldots, K_m$ have the same values, turns into a single data record Y_M. Each record from this population corresponds to an LMM element A $a_{i_1^*, \ldots, i_m^*, i_{m+1}^0, \ldots, i_p^0} = 1$. Then $a_{i_1^*, \ldots, i_m^*} = \bigvee\limits_{(i_{m+1}, \ldots, i_p)} a_{i_1^*, \ldots, i_m^*, i_{m+1}^0, \ldots, i_p^0} = 1$. So, $\varphi(quant(X_M)) = {}^{\mu}A$.

Statement 6. Let X_K and Y_K – data strictly ordered by multiple keys $K = \left\{K_1, \ldots, K_p\right\}$, $A = \| a_{i_1, \ldots, i_p} \|$ and $B = \| b_{i_1, \ldots, i_p} \|$ – their corresponding LMM $(\varphi(X_K) = A$ and $\varphi(Y_K) = B)$. Then the result of the strictly ordered data merger operation X_K and Y_K a single LMM corresponds $C = \| c_{i_1, \ldots, i_p} \|$ such that $C = A + B$.

Depending on the task to be solved for the formation of equivalence classes (records) of data – of the result of merging strictly ordered data X_K and Y_K functions are constructed that generate an equivalence class containing either a real or a universal indefinite entry. Similarly, as an additive operation on LMM elements A and B one of sixteen logical operations is selected. It forms the LMM element of the result based on the equivalence class of the result data and according to the rules for forming the LMM – image of the data when displayed φ.

Thus, the data – of the result of the merger operation is strictly ordered data X_K and Y_K a single LBM corresponding to, equal to the sum of the LMM A and B with additive operation $\circ$ above the elements of the matrices. That is $\varphi\left(ms\left(X_K, Y_K\right)\right) = A +_{(\circ)} B$. Operation sign $+_{(\circ)}$ read as LMM addition operation with additive operation $\circ$ above their elements.

Example 11. The operations of merging strictly ordered data, which implement set-theoretic union and intersection operations, correspond to the operations of assembling multivariate matrices with additive operations on elements: disjunction $(\vee)$ that conjunction $(\wedge)$.

Statement 7. Let X_L and Y_M – data strictly ordered by multiple keys $L = \{L_1, \ldots, L_p\}$ and $M = \{M_1, \ldots, M_p\}$ and the condition is fulfilled $L \cap M \neq \varnothing$, and let $K = \{K_1, \ldots, K_r\}$, $(r < p + q)$ – set of keys associated with sets L and M ratio: $K \subseteq L \cup M$, $K \cap L \neq \varnothing$, $K \cap M \neq \varnothing$ (data $X_{K \cap L}$ and $X_{K \cap M}$ not strictly ordered by their sets of keys). $A = \| a_{i_1, \ldots, i_p} \|$ and $B = \| b_{i_1, \ldots, i_r} \|$ – data-relevant $X_{K \cap L}$ and $X_{K \cap M}$ LMM $(\varphi(X_{K \cap L}) = A$ and $\varphi(X_{K \cap M}) = B)$. Then given Z_K – the result of the merger operation is not strictly ordered by the set of keys K given X_K and Y_K a single multidimensional matrix corresponds $C = \| c_{i_1, \ldots, i_r} \| = {}^{\lambda, \mu}(A \times B)$.

According to the definition of the merge operation of non-strictly ordered data, the set of keys K consists of three subsets:

– L' – keys belonging only to a set of keys L;

– M' – keys belonging only to a set of keys M;

– T – keys belonging to both sets of keys L and M.

When building an operation $^{\lambda,0}(A\times B)$ it can be assumed that the keys of subsets L' and M' correspond to free LMM indices A and B (breakdown indices l and m). The keys correspond to subsets T – scott indices of LMM A and B (breakdown indices s). Equivalence class $Z_{K^*}\neq\Theta_{K^*}$ only if the classes corresponding to it equivalence $X_{(L'\cup T)^*}\neq\Theta_{(L'\cup T)^*}$ and $Y_{(M'\cup T)^*}\neq\Theta_{(M'\cup T)^*}$.

Then elements LMM A and B, corresponding to these equivalence classes, have a value of 1. So the result of the conjunction of these elements will also have a value of 1. That is, the equivalence class corresponding to this element $Z_{K^*}\neq\Theta_{K^*}$.

If the function $f(X_{(K\cap L)^*},X_{(K\cap M)^*})$ implements a group operation, then the operation of merging ordered non-strictly data includes a data compression operation Z_K, as a result of which data is obtained $Z_{K'}$ (plural keys K' there is a subset of the set of keys K, i.e $K'\subset K$). In this case, the parts of the keys of the subset T LMM cell indices correspond A and B (breakdown indices c), and merge operations of not strictly ordered data X_L and Y_M the operation corresponds (λ, μ)-collapsed product of the matrices corresponding to them. That is, $\varphi(mns(X_{K\cap L},X_{K\cap M}))=^{\lambda,\mu}(A\times B)$.

5. 5. Axiomatic approach to formalizing data models

The axiomatic method of formalization of mass data processing (MDP) for IDSS is considered below. The axiomatic method is used to formalize an object transformation operation by describing the resulting object from transforming the original objects. That is, it provides a description of the objects and their transformation operations according to the principle: "which should be the result of operations on operand objects, regardless of what the reality of these objects is and how this operation is performed".

The algebraic method allows for constructive descriptions of object structures and operation algorithms. The structure of objects is defined either strictly, using mathematical terminology, or by analogy with known structures. For example, in a file model, a file is defined in the language of set theory, and in relational terms, a similar object looks to be a set according to a table.

Algorithms are also defined either by natural language description or by using formalization tools inherent in theoretical and practical programming. That is, the principle is inherent to the algebraic method: "after applying an operation according to a known algorithm to an object (s) with a known structure, an object result with a given structure" is obtained.

For example, relational computation is nothing more than an axiomatic description of objects (relationships) and operations of transforming one relation into another. This allows to determine what relation should be obtained as a result of applying a given operation on operand relations. In relational algebra, an object is structured as a comparison in the form of tables, and operations on relations are usually given as verbal (intuitive) descriptions. In multivariate-matrix algebra, which is homomorphic to relational algebra, each relation corresponds to a multivariate matrix, which can be considered a highly structured object. Strict formal algorithms are given for all operations.

Next, the formalization of MDPs for set-theoretic (file) relational and multidimensional matrix data models is considered. In the future, all of them will be considered a single formal (axiomatic) theory T. It will be considered defined if the following conditions are met:

– given defined finite set of symbols called theory symbols, T;

– defines the finite sequences of the symbols of the theory forming the set T^*, which is called expressions of the theory T.

The set of symbols of the axiomatic theory T for MDP [17, 19] contains two types of symbols:

1. Common symbols:

– digits: 0, ..., 9;

– upper and lower case letters: a, b, ..., A, B, ...;

– letters with indexes: a_1, a_2, ..., A_1, A_2, ...

2. Special symbols of two classes:

– therms (objects) of the theory T, which are defined as letters or sequences of letters;

– formulas (ratio) of the theory T.

Intuitively, the terms are the notation of objects, and the formulas are the notation of statements in what can be done with these objects. There is usually an effective procedure that allows each expression to determine whether it is a formula. In this theory, this procedure is given by a recursive relation. The formulas thus defined can be interpreted differently in different models of theory T. These are set-theoretic, (file), multidimensional-matrix and relational models of IDSS data.

In these data models, there are the terminological differences, but also different ways of representing objects and different algorithms of their transformation operations. Table 2 shows interpretations in different data models of some formulas that will be used in the construction of axioms.

Table 2

Examples of interpretation of formulas

Formulas and their values in data models		
Set theory	Multidimensional-matrix	Multidimensional-relational
A_i, a_i		
Record field, field value	Set (type of elements), value of the element	Attribute, attribute value
$(a_{1,\ ...,}\ a_n)$		
Instance of data recording	Multidimensional matrix element	Cortege
$\mathscr{K}\left(A_1,\ldots,A_k\right)R\left(A_1,\ldots,A_k,A_{k+1},\ldots,A_n\right)$		
$X_K=(F_1, ..., F_n)$ – data with type recording $(F_1, ..., F_n)$ and plural keys $K=(F_1, ..., F_k)$;	M – multidimensional matrix with indices $i_1, ..., i_k$	Relationship $R(A_1, ..., A_k)$ with scheme $(A_1, ..., A_n)$ and the constituent key $A_1, ..., A_k$
$f(A_1, ..., A_n)$		
n-ary operation on data recording fields	Additive operation on elements of a multidimensional matrix	n-ary operation defined on attribute domains $A_1, ..., A_n$
$f_{a_1,\ldots,a_k}\left(A_{k+1},\ldots,A_n\right)$		
n-ary an operation on fields of a group of data records with the same instance of a set of keys	Multiplicative operation on multidimensional matrices with Kelly indices $i_{1,\ ...,}\ i_k$	n-ary group operation defined on attribute domains $A_{k+1}, ..., A_n$
$^{so}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R\left(A_1,\ldots,A_k,A_{k+1},\ldots,A_n\right)\right)$		

A defined set of formulas called axioms (or schemes of axioms) of theory T is highlighted. Formulas that are axioms for mass data processing are discussed later:

A1) Strict order axiom

$$^{so}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R\left(A_1,\ldots,A_k,A_{k+1},\ldots,A_n\right)\right)=$$
$$=\left\{\begin{pmatrix}a_1,\ldots,a_k,\\ a_{k+1},\ldots,a_n\end{pmatrix}\mid\forall\begin{pmatrix}a_1,\ldots,a_k,\\ a_{k+1},\ldots,a_n\end{pmatrix}\neg\left(\exists\begin{pmatrix}a_1,\ldots,a_k,\\ a'_{k+1},\ldots,a'_n\end{pmatrix}\right)\right\}. \quad (16)$$

This axiom means that objects of type (a_1, ..., a_k, a_{k+1}, ..., a_n) are in the facility R no more than once. It gives a description of the data aggregate as a set.

A2) The axiom is not strict orderliness

$$^{nso}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R\left(A_1,\ldots,A_k,A_{k+1},\ldots,A_n\right)\right)=$$
$$=\left\{\begin{pmatrix}a_1,\ldots,a_k,\\ a_{k+1},\ldots,a_n\end{pmatrix}\mid\forall\begin{pmatrix}a_1,\ldots,a_k,\\ a_{k+1},\ldots,a_n\end{pmatrix}\exists\begin{pmatrix}a_1,\ldots,a_k,\\ a'_{k+1},\ldots,a'_n\end{pmatrix}\right\}. \quad (17)$$

This axiom means that objects of type (a_1, ..., a_k, a_{k+1}, ..., a_n) may occur in an object R no more than once, that is, the set of data is defined as a factor of a set (multiple). Interpretations of axioms A1 and A2 in the data models, it is given in the Table 2.

A3) Sample axiom

$$^{sel}R\left(A_1,\ldots,A_k,A_{k+1},\ldots,A_n\right)\Pi\left(a_1,\ldots,a_k\right)=$$
$$=\left\{\begin{pmatrix}a_1,\ldots,a_k,\\ a_{k+1},\ldots,a_n\end{pmatrix}\in\begin{pmatrix}A_1,\ldots,A_k,\\ A_{k+1},\ldots,A_n\end{pmatrix}\mid\Pi\left(a_1,\ldots,a_k\right)\right\}. \quad (18)$$

This axiom gives a description of a subset of a data aggregate, which elements satisfy the condition defined by the predicate $\Pi(a_1, ..., a_k)$. This subset is formed as a result of the application of the sampling operation $^{(sel)}$ to the data aggregate $R(A_1, ..., A_k, A_{k+1}, ..., A_n)$. In the data model, it corresponds to the sampling operation, the multidimensional-matrix data model, the cross-section operation of the multidimensional matrix, in the relational data model – operator SELECT.

A4) Compression axiom

$$\left(^{nso}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R\left(A_1,\ldots,A_k,B_{k+1},\ldots,B_m\right)\right)\right)=$$
$$=\left\{\begin{pmatrix}a_1,\ldots,a_k,\\ f_{a_1,\ldots,a_k}\left(B'_{k+1},\ldots,B'_m\right)\end{pmatrix}\mid B'_{k+1}\subset B_{k+1},\ldots,B'_m\subset B_m\right\}. \quad (19)$$

This axiom gives a description of the data aggregate to be derived from the data aggregate $R(A_1, ..., A_k, B_{k+1}, ..., B_m)$ after applying the operation $f_{a_1,\ldots,a_k}(B'_{k+1},\ldots,B'_m)$. This function applies to all groups of its elements in which the values of the variables A_1, ..., A_k coincide.

A new data aggregate is formed as a result of applying an operation to the data aggregate $R(A_1, ..., A_k, B_{k+1}, ..., B_m)$. In the set-theoretic data model, this axiom corresponds to the operation of compressing non-strictly ordered data, in the multivariate-matrix – convolution, in the relational one – *select ... group by ...* to the relation in the first normal form.

A5) Merge axiom with strict ordering of data aggregates

$$^{so}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R_1\left(A_1,\ldots,A_k,B_{k+1},\ldots,B_m\right)\right)\oplus$$
$$\oplus\,^{so}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R_2\left(A_1,\ldots,A_k,C_{k+1},\ldots,C_n\right)\right)=$$
$$=\left\{\left(a_1,\ldots,a_k,f\left(b_{k+1},\ldots,b_m,c_{k+1},\ldots,c_n\right)\right)\right\}. \quad (20)$$

This axiom gives a description of the data aggregate to be derived from the data aggregates $R_1(A_1, ..., A_k, B_{k+1}, ..., B_m)$ and $R_2(A_1, ..., A_k, C_{k+1}, ..., C_n)$ after applying the operation $f(b_{k+1}, ..., b_m, c_{k+1}, ..., c_n)$ to elements with the same values $a_1, ..., a_k$.

A new data aggregate is formed as a result of the operation ($\oplus$) to these data aggregates.

A6) Merge axiom with non-strict ordering of data aggregates

$$^{nso}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R_1\left(A_1,\ldots,A_k,B_{k+1},\ldots,B_m\right)\right)\otimes$$
$$\otimes\,^{nso}\left(\mathscr{K}\left(A_1,\ldots,A_k\right)R_2\left(A_1,\ldots,A_k,C_{k+1},\ldots,C_n\right)\right)=$$
$$=\left\{\begin{pmatrix}a_1,\ldots,a_k,\\ f_{a_1,\ldots,a_k}\begin{pmatrix}B'_{k+1},\ldots,B'_m,\\ C'_{k+1},\ldots,C'_n\end{pmatrix}\\ C'_{k+1}\subset C_{k+1},\ldots,C'_n\subset C_n\end{pmatrix}\mid B'_{k+1}\subset B_{k+1},\ldots,B'_m\subset B_m,\right\}. \quad (21)$$

This axiom gives a description of the data aggregate to be derived from the data aggregates $R_1(A_1, ..., A_k, B_{k+1}, ..., B_m)$ and $R_2(A_1, ..., A_k, C_{k+1}, ..., C_n)$ after applying the operation $f_{a_1,\ldots,a_k}\left(B'_{k+1},\ldots,B'_m,C'_{k+1},\ldots,C'_n\right)$ to groups of elements with the same values $a_1,\ldots,a_k$. A new data aggregate is formed as a result of the application of operation ( ) to these data aggregates.

Thus, using these axioms constructed using definitions, property ops of data aggregates and operations with them are constructed.

6. Discussion of the results on the development of a polymodel complex for the construction of intelligent decision support systems

The proposed polymodel complex for building intelligent decision support systems allows:

– systemically represent the relationship between IDSS construction models in the course of their calculation and computing tasks;

– carry out the simulation of the process of functioning of the IDSS, due to the use of an algebraic (formal) approach to object-oriented modeling and design of the IDSS;

– determine the rational tactical and technical indicators of the IDSS for solving specific calculation and computing tasks, due to the multi-level description of the IDSS construction order;

– make the transition from one type of data representation in IDSS to another due to the presence of appropriate mathematical transformations;

– multidimensional to describe the process of processing heterogeneous data in IDSS, due to the use of a multidimensional matrix model of IDSS data;

– approach universally the solution of computational-calculation tasks in IDSS by using an interconnected set of mathematical models for constructing IDSS;

– formalize the process of constructing IDSS, which allows combining IDSSs using different algorithmic and software.

The advantages of the proposed polymodel complex are due to the following:

– system representation of the relationship between IDSS construction models during their performance of calculation and calculation tasks (expressions (1)–(21)), compared to works [12, 19];

– the ability to carry out simulations of the process of IDSS functioning, through the use of an algebraic (formal) approach

to object-oriented IDSS modeling and design, in comparison with works [11, 14];

– to determine the rational tactical and technical indicators of the IDSS for solving specific calculation and computing tasks, due to the multi-level description of the order of construction of the IDSS, (expressions (1)–(21), Tables 1, 2), compared to works [9, 13];

– the possibility of making a transition from one type of data representation in IDSS to another due to the presence of appropriate mathematical transformations (Tables 1, 2), compared to works [9, 11];

– by multidimensional representations of the process of processing heterogeneous data in IDSS, due to the use of a multidimensional matrix model of IDSS data in comparison with works [13, 15];

– versatility to approach the solution of computational-calculation problems in IDSS by using an interconnected set of mathematical models for constructing IDSS (expressions (1)–(21), Tables 1, 2), in comparison with works [13, 18];

– a high degree of formalization of the IDSS construction process, which allows combining IDSS s using different algorithmic and software (expressions (16)–(21), Table 2) in comparison with works [12, 14];

– by the adequacy of the obtained results (expressions (1)–(21), Tables 1, 2), compared to works [16, 17].

The disadvantages of the proposed polymodel complex include the need to adapt the proposed mathematical apparatus to the specific conditions of the functioning of the IDSS.

The proposed polymodel complex should be used for the IDSS construction to solve general and specialized calculation tasks, as well as to solve the task of integrating various types of IDSS.

The limitations of the study are the need to have an initial database on the peculiarities of the IDSS functioning.

Areas of further study should be directed to reducing computing costs when processing various types of data in IDSS.

7. Conclusions

1. The basic concepts and definitions of universal algebra and algebraic systems in IDSS are given and an intuitive approach to object-oriented modeling of IDSS is developed. This allows to outline the list of basic concepts and definitions used in IDSS, and also allows to simplify the modeling of the process of functioning of IDSS.

2. An algebraic (formal) approach to object-oriented modeling and design of IDSS is proposed. This approach allows to formulate a list of basic mathematical procedures to the object-oriented simulation of the IDSS life cycle, regardless of the type and tasks performed by them.

3. An object-oriented approach to the development of IDSS data models is presented and a multidimensional matrix model of IDSS data is developed. This makes it possible to form a scientifically based approach to the order of development of IDSS data models, including in a multidimensional-matrix form.

4. Correspondences of set-theoretic and multidimensional-matrix models of IDSS data have been established. The proposed correspondences make it possible to flexibly make the transition from one form of data presentation in the IDSS to another, which significantly increases the degree of convergence of various types of IDSS.

5. An axiomatic approach to the formalization of data models for IDSS has been developed. This approach makes it possible to increase the unambiguity of data processing in IDSS when formalizing models of IDSS functioning. The proposed polymodel complex should be used for the IDSS construction to solve general and specialized calculation tasks, as well as to solve the task of integrating various types of IDSS.

Conflict of interest

The authors declare that they have no conflict of interest in this study, including financial, personal, authorship or other nature that could affect the study and its results presented in this article.

Financing

The study was conducted without financial support.

Data availability

The manuscript has related data in the data warehouse.

Use of artificial intelligence tools

The authors confirm that they did not use artificial intelligence technologies when creating the presented work.

Authors' contributions

Nina Kuchuk: Conceptualization; Methodology; Project administration; Writing – original draft; Writing – review & editing; **Leonid Artushin**: Methodology; Writing; Writing – review & editing; **Yurii Zhuravskyi**: Writing – original draft; **Iraida Stanovska**: Writing – review & editing; **Oleksiy Kononov**: Resources; Data Curation; **Nadiia Protas**: Validation; Data Curation; **Serhii Neronov**: Software; Validation; Data Curation; **Serhii Shostak**: Methodology; Formal analysis; Visualization; **Anton Nikitenko**: Software: Programming, software development; designing computer programs; implementation of the computer code and supporting algorithms; testing of existing code components; Validation; Data Curation; **Andrii Veretnov**: Software; Validation; Data Curation.

References

1. Kuchuk, N., Shyshatskyi, A., Radchenko, V., Andrusenko, Y., Klivets, S. (2026). Design of a multilevel architecture for optimizing virtual machine migration. Advanced Information Systems, 10 (2), 35–43. https://doi.org/10.20998/2522-9052.2026.2.04
2. Dudnyk, V., Sinenko, Y., Matsyk, M., Demchenko, Y., Zhyvotovskyi, R., Repilo, I. et al. (2020). Development of a method for training artificial neural networks for intelligent decision support systems. Eastern-European Journal of Enterprise Technologies, 3 (2 (105)), 37–47. https://doi.org/10.15587/1729-4061.2020.203301

3. Sova, O., Shyshatskyi, A., Salnikova, O., Zhuk, O., Trotsko, O., Hrokholskyi, Y. (2021). Development of a method for assessment and forecasting of the radio electronic environment. EUREKA: Physics and Engineering, 4, 30–40. https://doi.org/10.21303/2461-4262.2021.001940
4. Pievtsov, H., Turinskyi, O., Zhyvotovskyi, R., Sova, O., Zvieriev, O., Lanetskii, B., Shyshatskyi, A. (2020). Development of an advanced method of finding solutions for neuro-fuzzy expert systems of analysis of the radioelectronic situation. EUREKA: Physics and Engineering, 4, 78–89. https://doi.org/10.21303/2461-4262.2020.001353
5. Zuiev, P., Zhyvotovskyi, R., Zvieriev, O., Hatsenko, S., Kuprii, V., Nakonechnyi, O. et al. (2020). Development of complex methodology of processing heterogeneous data in intelligent decision support systems. Eastern-European Journal of Enterprise Technologies, 4 (9 (106)), 14–23. https://doi.org/10.15587/1729-4061.2020.208554
6. Li, Z., Xiong, J. (2024). Reactive Power Optimization in Distribution Networks of New Power Systems Based on Multi-Objective Particle Swarm Optimization. Energies, 17 (10), 2316. https://doi.org/10.3390/en17102316
7. Lee, B. M. (2025). Efficient Resource Management for Massive MIMO in High-Density Massive IoT Networks. IEEE Transactions on Mobile Computing, 24 (3), 1963–1980. https://doi.org/10.1109/tmc.2024.3486712
8. Gasimov, V., Mammadov, J., Islamov, I., Hashimov, E. (2026). Evaluation of alternative solutions for the effective structure of the cyber security system in critical information infrastructures by the hierarchical analysis method. Advanced Information Systems, 10 (2), 87–99. https://doi.org/10.20998/2522-9052.2026.2.10
9. Folorunso, O., Mustapha, O. A. (2015). A fuzzy expert system to Trust-Based Access Control in crowdsourcing environments. Applied Computing and Informatics, 11 (2), 116–129. https://doi.org/10.1016/j.aci.2014.07.001
10. Mohammad, A. (2020). Development of the concept of electronic government construction in the conditions of synergetic threats. Technology Audit and Production Reserves, 3 (2 (53)), 42–46. https://doi.org/10.15587/2706-5448.2020.207066
11. Bodin, L. D., Gordon, L. A., Loeb, M. P., Wang, A. (2018). Cybersecurity insurance and risk-sharing. Journal of Accounting and Public Policy, 37 (6), 527–544. https://doi.org/10.1016/j.jaccpubpol.2018.10.004
12. Cormier, A., Ng, C. (2020). Integrating cybersecurity in hazard and risk analyses. Journal of Loss Prevention in the Process Industries, 64, 104044. https://doi.org/10.1016/j.jlp.2020.104044
13. Hoffmann, R., Napiórkowski, J., Protasowicki, T., Stanik, J. (2020). Risk based approach in scope of cybersecurity threats and requirements. Procedia Manufacturing, 44, 655–662. https://doi.org/10.1016/j.promfg.2020.02.243
14. Perrine, K. A., Levin, M. W., Yahia, C. N., Duell, M., Boyles, S. D. (2019). Implications of traffic signal cybersecurity on potential deliberate traffic disruptions. Transportation Research Part A: Policy and Practice, 120, 58–70. https://doi.org/10.1016/j.tra.2018.12.009
15. Isong, A., Stephen, B. U.-A., Asuquo, P., Ihemereze, C., Enang, I. (2026). Machine learning based cloud computing intrusion detection. Advanced Information Systems, 10 (1), 115–125. https://doi.org/10.20998/2522-9052.2026.1.13
16. Zarreh, A., Saygin, C., Wan, H., Lee, Y., Bracho, A. (2018). A game theory based cybersecurity assessment model for advanced manufacturing systems. Procedia Manufacturing, 26, 1255–1264. https://doi.org/10.1016/j.promfg.2018.07.162
17. Zhuravskyi, Y. (Ed.) (2026). Intelligent decision support systems: methods for optimizing and supporting management decisions. Kharkiv: TECHNOLOGY CENTER PC. https://doi.org/10.15587/978-617-8360-23-8
18. Koval, M., Sova, O., Shyshatskyi, A., Artabaiev, Y., Garashchuk, N., Yivzhenko, Y. et al. (2022). Improving the method for increasing the efficiency of decision-making based on bio-inspired algorithms. Eastern-European Journal of Enterprise Technologies, 6 (4 (120)), 6–13. https://doi.org/10.15587/1729-4061.2022.268621
19. Shyshatskyi, A. (Ed.) (2024). Information and control systems: modelling and optimizations. Kharkiv: TECHNOLOGY CENTER PC. https://doi.org/10.15587/978-617-8360-04-7
20. Voznytsia, A., Sharonova, N., Babenko, V., Ostapchuk, V., Neronov, S., Feoktystov, S. et al. (2025). Development of methods for intelligent assessment of parameters in decision support systems. Eastern-European Journal of Enterprise Technologies, 4 (4 (136)), 73–82. https://doi.org/10.15587/1729-4061.2025.337528