http://www.moderntechno.de/index.php/meit/article/view/meit37-02-053
DOI: 10.30890/2567-5273.2025-37-02-053

UDC 165+004.2

A NOVEL APPROACH: LEVERAGING A GENERAL THEORY OF APPLIED ALGORITHMS FOR HIGH-LEVEL LANGUAGE DESIGN

НОВИЙ ПІДХІД: ВИКОРИСТАННЯ ЗАГАЛЬНОЇ ТЕОРІЇ ПРИКЛАДНИХ АЛГОРИТМІВ ДЛЯ ПРОЄКТУВАННЯ МОВ ВИСОКОГО РІВНЯ

Kolisnyk V.H. / Колісник В.Г.
s.r.s. / с.н.с.
ORCID: 0000-0002-2313-9852
Donbas State Engineering Academy, Ternopil, Fedkovycha, 9, 46001
Донбаська державна машинобудівна академія, Тернопіль, Федьковича, 9, 46001
Bodyk O.P. / Бодик О.П.
c.philol.s, as.prof. / к.філол.н., доц.
ORCID: 0000-0003-4642-8371
Mariupol State University, Kyiv, Preobrazhenska, 6, 03037
Маріупольський державний університет, Київ, Преображенська, 6, 03037

Abstract. *The software development industry has long recognized the need for an efficient theory to propel progress in programming technology. Despite continuous efforts, attaining a comprehensive theory has proven to be a formidable challenge. The study introduces a comprehensive theory of applied algorithms, aiming to fill this void. The proposed theory centers on the programming process as the most arduous component of software development and strives to enable a profound revolution. The theory is grounded on two fundamental concepts: G-properties and virtual component operations. These constructs serve as the foundational elements for a thorough decomposition scheme and a circuit for producing object attributes, collectively constituting a model for a universal or canonical algorithm. The primary objective of the theory is to develop a framework for accurately combining algorithms and application programs. The decomposition scheme and synthesis loop form the basis of this approach, allowing for the creation of software solutions that are more efficient and effective.*

The paper emphasizes the significance of conducting extensive empirical research to validate and enhance the proposed theory. We seek to overcome the theoretical obstacles in programming technology in order to facilitate substantial progress in Software Engineering. This, in turn, will improve the efficiency and excellence of software development processes, ultimately contributing to the advancement of the field.

Keywords: *decomposition scheme, object type, object part type, object properties, general programming concepts; algorithm.*

Abbreviations: *GTAA: General theory of applied algorithms; GTSE: General theory of software engineering; CO: Component operation; VCO: Virtual component operation; MCO: Material component operation; OPSC: Object properties synthesis circuit; EM: Executive member; SCF: Synthesis circuit fragment; DS: Decomposition scheme; PDS: Particular decomposition scheme; CDS: Complete decomposition scheme; DM: Decomposition mechanism; CM: Composition mechanism; DIM: Deductive inference mechanism; PCS: Particular composition scheme; BA: Basic action; E-property: Elementary property; A-property: Aggregate property; C-property: Cumulative property; G-property: General or generic name for an elementary property, an aggregate property and a cumulative property; E-data: Elementary data; A-data: Aggregate data; C-data: Cumulative data; G-data: General or generic name for an elementary data, an aggregate data and a cumulative data.*

I. Introduction.

Although specialists and scholars are currently engaged on developing a General Theory of Software Engineering [1], they have not yet accomplished significant progress. There are other factors contributing to this, specifically:

1. Computers are being integrated into every facet of human endeavor. Consequently, there exists a wide range of diverse and intricate software products. As a result, it is challenging to identify any substantial similarity between the software product and the development process [2].
2. Researchers must ascertain the direction of their search and investigation. The topic of software development as a research discipline is inundated with numerous facts and elements that must be considered. Consequently, researchers encountered the issue of how to divide the field of investigation:

"In summary, the 4th GTSE workshop grappled with the concern that a general theory would be too complicated to facilitate communication and scientific inquiry. Participants discussed several approaches to addressing this problem, including separating a general theory into multiple pieces (e.g., one for software artifacts and another for the process of developing software artifacts) and devising separability principles" [3].

3. Despite being written using abstract, mathematized signs, real computer programs and algorithms have not yet been subjected to mathematical analysis [4].
4. There is a lack of sufficient philosophical evaluation of the information and computer artefact.
5. Within the research community, there is no discussion about the notion of expansive notions and generalizations that should serve as the foundation for high-performance programming languages [5].

The absence of a GTSE can be attributed to these problems [6, 7].

One consequence of the current unsatisfactory state of programming technology is that, up to now, specific application algorithms have not been derived through logical deduction. Instead, they have been developed based on individual experience,

education, and intuition, which might be considered a craft style approach. While the craft style of programming can demonstrate talent and efficiency, it has inherent limitations in terms of production.

Software Engineering, similar to other branches of Engineering, encompasses several procedures and diverse sorts of work in addition to programming. However, the primary prerequisite for ensuring progress in Software Engineering is the alteration of programming itself, elevating it to the maximum level of development.

Algorithmic programming languages, such as ALGOL, Fortran, and COBOL, as well as modern languages, are known for being text-based and excessively comprehensive. Texts composed in such languages are characterized by their length, complexity, and challenging readability and handling. The software sector continues to provide challenges and, at times, exhibits a deficiency in project management. This is the exact rationale behind the absence of productivity increase in the programming sector.

It is imperative to develop programming languages that enable information compression and facilitate visual and tangible program creation[1]. However, it is highly improbable for such languages to emerge solely by speculation and intuition, similar to the languages that were developed during the early stages of the information age. The development of high-performance languages requires the application of theoretical principles. This hypothesis has the potential to facilitate the development of novel programming languages. Furthermore, this theory should establish the necessary conditions for the development and integration of algorithms for practical software applications[2] – the overarching general theory of applied algorithms (GTAA)[3].

Programming languages that will be developed based on this notion must incorporate operators that are more comprehensive and universal than the operators found in existing languages. Additionally, it is vital to utilize the resource of graphics. In order to utilize graphics, one must additionally employ operators that have a greater capacity than those found in conventional technology-focused programming

[1] Mathematical theorems, technical drawings in mechanical engineering, and other industrial documents are readily available for critical scrutiny by numerous specialists. As a result of this, the majority of errors are eradicated.

[2] We are discussing a theory distinct from the usual theory of algorithms, such as Turing machines and finite automata. This alternative theory aims to provide practical solutions for programming applications.

[3] This theory is also examined in the work [8].

languages.

The development of GTAA and algorithmic languages of a novel nature would enable the attainment of project manageability in Software Engineering and facilitate advances in enhancing labor productivity.

II. Articulating the problem and objectives.

This study aims to offer a comprehensive explanation of the fundamental principles behind applied algorithms. A theory description entails the delineation of its key characteristics, including the research methodology, research field, object and subject of research, fundamental assumptions (axioms), and the primary objective of the theory, a theoretical generalized model of the decomposition scheme as a means of addressing the main problem, research method, research goal, and research directions.

The following are the primary provisions or attitudes that form the foundation of GTAA. These requirements pertain to computer-based processes conducted in accordance with a program.

1. Initial installation. Knowledge is generated with the use of a computer by processing information and parameters. Precise data is produced that is essential for making particular judgements or carrying out specified actions. Data can be conveyed in various forms such as numerical, alphabetical, graphical, etc., either in digital or analogue format.
2. Subsequent installation. A computer, when executing a program, consistently begins and develops a process or actively engages in the activity. There are two fundamental categories of processes in their most comprehensive form:
   - The computer governs the procedure. The description of process fragments is either pre-prepared in the form of a written program or generated during computer operation in real-time ONLINE mode.
   - The process is conducted at one's own pace and relies on computer usage only sporadically. Process fragments are only generated using a computer. The computer executes process parameters according to the process's request.
3. Third installation. Every process and information system may be broken down into three distinct stages: input, processing, and output.

III. Explanation of decomposition schemes.

The first perspective posits that acquiring or constructing knowledge can be achieved by employing established frameworks derived from philosophy and epistemology. As part of this study, the decomposition method was taken into account[4].

Under the framework of GTAA, all objects in the physical world that pertain to issues addressed by computers and computer programs are classified into certain categories. All elements, constituents, and subordinate processes of the objects being examined are also exemplars of certain types. Individual information images are generated for each specific part type. The primary focus of the idea is to provide information images of various part types.

Images containing information about different types of parts are essential components of the decomposition scheme. The interdependencies between an object's type and the types of its parts, which can either be included or contain components themselves, are illustrated by a tree graph known as a part type tree or type tree. The root node, which represents the original object, is positioned at the top of the figure. The nodes in the type tree correspond to the types of incorporated parts. A tree edge is a connection between two nodes. The higher node represents the included kind of part, while the lower node represents the included type of part. A particular decomposition scheme (PDS) refers to a pair of nodes connected by an edge.

The type tree is a GTAA object that serves as both a carrier and holder of property types. Figure 1a displays the PDS. A random portion of a decomposition scheme is illustrated as a segment of a type tree in Figure 1b. The complete decomposition scheme (CDS) is depicted as a hierarchical structure of object pieces or a hierarchy of types in Figure 1c. The generic term for PDS, CDS, and a fragment of a decomposition scheme is "decomposition scheme" (DS).

Components of various types can serve as objects of decomposition, resulting in the possibility of an arbitrary number of levels in the type tree. Various decomposed parts can be reassembled as components to form a structure, and then decomposed

[4] Further elaboration on the decomposition procedure and decomposition scheme can be found in [9, 10].

again in a different manner. These components can then be combined into a more complicated structure, serving as a single component.

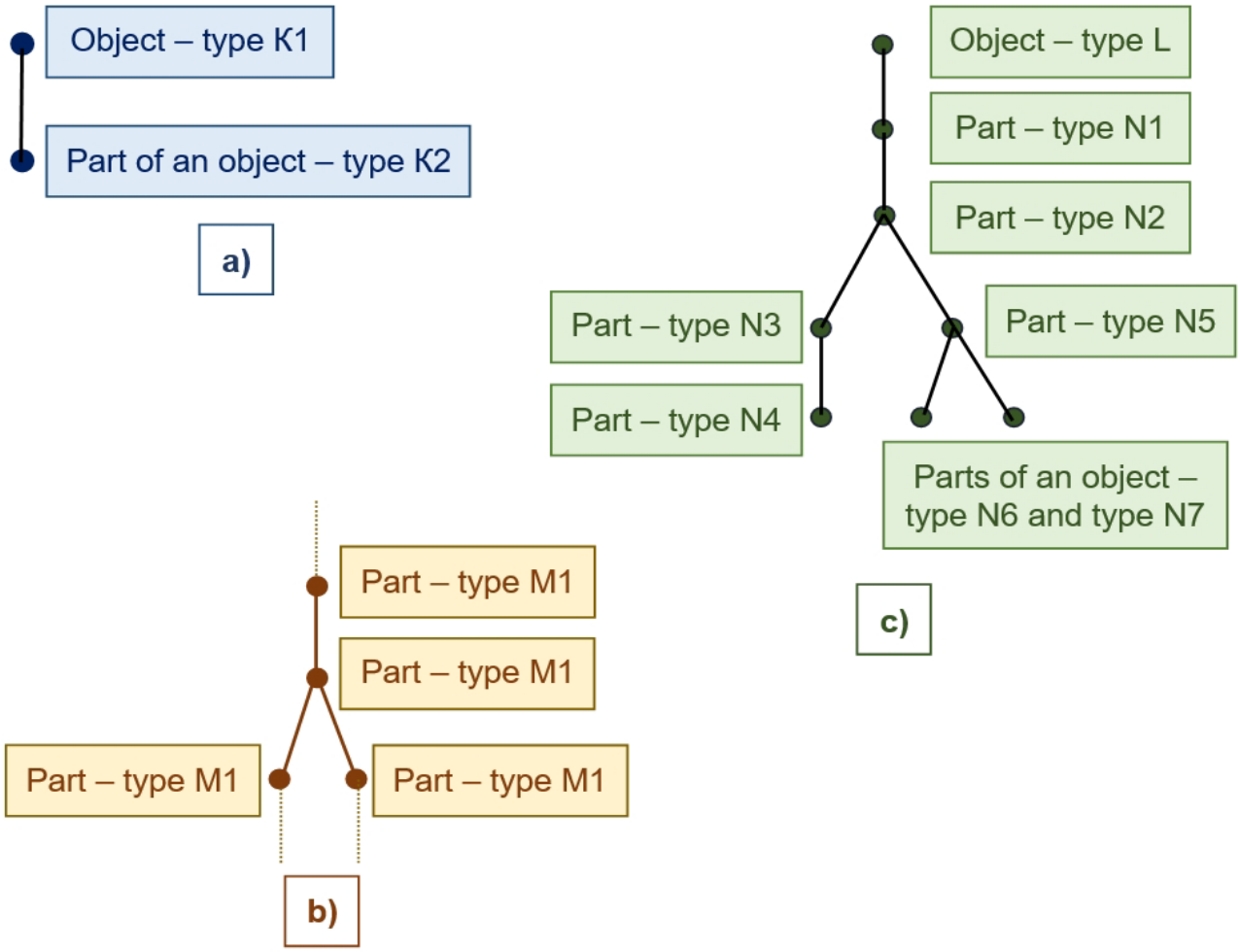

Figure 1: Illustration of parts of an object indicating their types

a) Particular decomposition scheme, b) Decomposition scheme fragment, c) Complete decomposition scheme

Authoring

Various components can be permanently located within the DS. Consequently, it is necessary to allocate a specific location for this type within the tree structure. Types may have conditional behavior. Therefore, certain vertices in the tree should be labelled as conditional. The types of two or more parts can be conditional. Data collections can be organized in a more varied manner when saved on storage media.

Once a description of the DS has been suggested, the fourth installation is established: each computer program is considered an executed DS or a portion that executes a part of the DS.

IV. Depiction of properties of object types and types of object parts.

Different qualities can be associated with each of types of object parts as well as with the object type itself[5]. As an illustration, the type *"part"* can be defined by its weight and size. The weight and size of the part are inherent characteristics of this type. However, it is possible for an object to be composed of components belonging to both type 1 and type 2. To differentiate between the characteristics of the components of these two types, it is essential to establish the properties of part type 1 as "weight of part type 1" and "size of part type 1", along with the properties of part type 2 as "weight of part type 2" and "size of part type 2". The term *"property"* will be used henceforth to refer to the concept of *"property type"*.

There are three main basic types of property types that are distinguished: elementary property (E-property), property-aggregate (A-property), and cumulative property (C-property). The C-property, together with the E-property and the A-property, have a common concept called "property" (G-property) that encompasses all of them. Figure 2 illustrates the interconnections between the various types of G-properties [9].

A *simple A-property* is defined as an A-property that contains many E-properties of various types. A C-property that consists of only one E-property is referred to as *simple C-property*. An *extended C-property* is formed by combining a *simple A-property*. An A-property that contains one or more simple A-properties or extended C-property is referred to be *extended A-property*. A *complicated C-property* is formed by combining an *extended A-property*. Neither the *extended C-property* nor the *extended A-property* are constrained by the quantity of nodes or the quantity of levels.

In the context of GTAA, the data pertaining to an object is taken into account in addition to its properties. G-data refers to information on an object or a specific quantity that describes the object. It is considered a G-property of the object, and it is associated with a sign that can be read or detected by a person or by technological devices. Additionally, there are methods available for storing or recording this sign on various media. This sign conveys the precise significance of this G-property. G-properties and

[5] The detailed descriptions of object types and property types can be found in [9, 10]

G-data pertain to both the entirety of the object and its parts. There are three main types of P-data: elementary data (E-data), data-aggregate (A-data), and cumulative data (C-data). The A-data may consist of both E-data and C-data, in addition to other information. The C-data may include A-data.

G-data types and G-property types are assigned names. A tree graph is utilized to represent all types of G-data (including G-properties) acquired during decomposition.

Each type of part of an object or the object itself is only connected with one A-property. The A-property in question is referred to as the *primary A-property* for a specific type of object part. The primary A-property may comprise *intermediate A-properties* that are encompassed inside it as constituent components. Intermediate A-properties can include A-properties that are also intermediate. In other words, there exists a hierarchical arrangement of A-properties.

V. Factors related to the processing of general information.

Initial factor. Analysis and manipulation of visual representations of tangible objects in the physical environment.

GTAA, in its most comprehensive manifestation, encompass a range of basic actions (BA) that can be executed utilizing a computer.

1. The computer mirrors the real world by representing it through data.
2. The computer data is projected onto the physical environment.
3. Integration or merging of the initial and subsequent BA.
4. The articulation and synthesis of the initial BA, processing, and the subsequent BA.

Secondary factor. At every node of the CDS tree, the BA might be either intricate or streamlined. BAs can be created by combining or merging simpler BA.

The theory classifies two categories of tangible objects that BA are utilized for. These are objects, constructions, organisms, geographical or astrophysical objects – something that is a relatively stable, static object. The second category consists of processes that may be manipulated or seen by a computer – these are referred to as dynamic objects. The ultimate system, implemented by a computer, may comprise static and dynamic objects as its constituents.

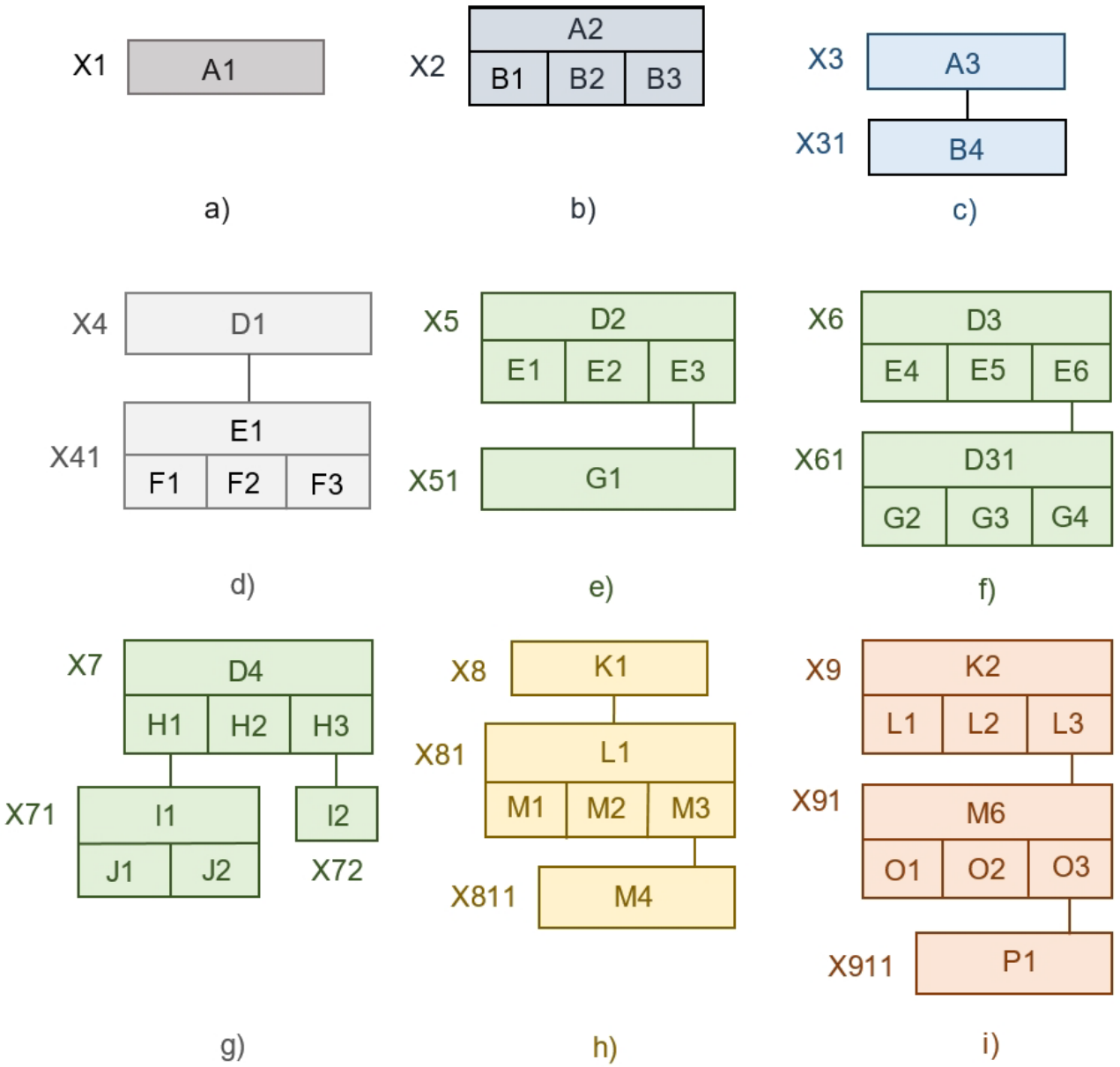

Figure 2: Objects, parts of objects and types of G-properties

X1, X2, X3, X4, X5, X6, X7, X8, X9 – types of objects

X31, X41, X51, X61, X71, X72, X81, X811, X91, and X911 – types of parts of objects

a) Elementary property (E-property) e, f, g) Advanced aggregate properties

b) Aggregate property (A-property) h) Advanced cumulative property

c) Cumulative property (C-property) i) Advanced aggregate property

d) Advanced cumulative property

Authoring

BAs are fundamentally structured and operated based on decomposition schemes.

The third factor. Through the utilization of a computer, novel information is generated. Even without active processing, new knowledge can still be produced through the mere reflection of information. The aforementioned four points can also be expressed in this manner.

1. Reflection of an information image of a static object into a computer.
2. The computer outputs the generated information image of the static object.
3. Projection of the dynamic information image into the computer.
4. The computer outputs the generated information image of the dynamic object.
5. The articulation or synthesis of acts in paragraphs 1 and 2.
6. The articulation or synthesis of actions in paragraph 1, information processing and actions in paragraph 2.
7. The articulation or synthesis of acts in paragraphs 3 and 4.
8. The articulation or synthesis of actions in paragraph 3, information processing and actions in paragraph 4.

The articulation or synthesis of BA and processing can be simultaneously more advanced and intricate. Here are some choices for making improvements to data processing.

Option 1. A computer generates and controls the information image of a dynamic object. The reflection that occurs from the computer involves the delivery of control signals to actuators. The control results can be reflected in the computer.

Option 2. A computer generates the information image of a dynamic object, but does not have control over it. The reflection emanates from the computer, which serves as an information model of the process for human observation.

G-properties are inputted into the computer, but upon input, they are transformed into G-data. The determination of the input involves identifying the appropriate means of both storing and accessing information.

The Internet, as well as corporate databases or repositories of practical / applied or scientific information, are regarded solely as stores of information and images pertaining to the material world. All the aforementioned operations can be executed on them. Information images and the structures they produce can serve as knowledge or

models for physical objects that are later created.

The process of choosing information objects in the physical world can result in success, failure, or partial success. The selection field may be extremely spacious. The input data possesses a hierarchical structure that is applied using proprietary methods.

In the context of GTAA, various forms of information, such as text, numbers, and graphics, are analyzed based on their G-properties and G-data. All additional information is structured as compositions of G-properties and G-data. Operators or procedures manipulate G-data as operands to execute the processing of information, resulting in input and output. The input-output operands consist of many sorts of G-data, including E-data, A-data (simple, extended, complex), and C-data (simple, extended, complicated).

VI. Operations of components.

There are four universally performed procedures or operations of a broad scientific or philosophical type that are carried out by individuals in various domains, including scientific, industrial, everyday, and artistic activities. The operations are as follows:

- Decomposition.
- Composition.
- Deductive inference.
- Inductive inference.

In theory, a collection of these operations is regarded as the foundation or model for more sophisticated programming language operators. GTAA conducts research, develops, and suggests a method for converting these processes into software structures. These operations are implemented on tangible objects or their components in practical tasks. Formally, these procedures are referred to as material component operations (MCO). Their virtual pictures or prototypes are referred to as virtual component operations (VCO) in theory. VCO theoretically utilizes information images of types of parts and types of properties of parts. The operands of the VCO are information images of types of parts and types of properties of parts. VCO serve as the primary essential tools for creating algorithms and programs. The overarching and inclusive term for MCO and VCO is component operations (CO). CO refers to operations performed on components that are the results of decomposition or are

operations on images of components.

6.1. Decomposition.

Procedure VCO "Decomposition" is the process of mentally or virtually breaking down an object according to the PDS. The description of the PDS and the decomposition mechanism (DM) exhibit discrepancies. PDS refers to the process of breaking down an object into smaller components, which can be represented using a mathematical formula or written description. The PDS in a tree diagram of decomposition is coupled with a node that reflects the specific part that will undergo decomposition. DM is a method used to identify specific material or information components; it is an algorithm. The development of DM is consistently preceded by the construction of PDS. DM is specified if the following is known:

- The specific conditions, which are required for initiating decomposition.
- How the first step is executed.
- How the next steps are executed.
- Under what conditions the decomposition terminates.
- What operations are required prior to or subsequent to the final step.

In order to present the DM initially as a VCO "Decomposition" and then as an algorithm that may be executed as a fragment of a computer program, it is necessary to have knowledge of all this information.

Both the initial and last steps might hold significance. A precise execution can generally be found for each phase of decomposition.

Decomposition may result in parts of more than one type. An identifier is allocated to each part type. Each component inside a type is assigned a unique number or identifier. A vertex is created in the CDS tree for every new type of part, and it is coupled by an edge to a node that is obtained as a result of decomposition or to the original object. During the process of decomposition, certain components may emerge that are considered insignificant for further processing. The various components (together with their respective subassemblies) are appropriately marked. The number of decomposed components may deviate from the predicted value as shown by the DS.

During the process of decomposition, an object (or part of an object) can be

separated into components using many decomposition schemes sequentially – divided, then divided again, etc. In this scenario, the DS are analyzed to identify a circuit that will satisfy specific criteria. By implementing the subsequent DS, a distinct collection of parts will be acquired on each occasion. A collection of information images can be produced for each collection of material parts. If N decompositions were performed, N DS were also performed, resulting in N collections of their corresponding information images. It is possible to save all of these collections and then can be selected the ones that meet a specific criterion. The constituents of these collections, referred to as G-data, will possess varying types.

For the VCO "Decomposition" to be considered genuine, it is essential to ascertain the specific location and method of storing the processed information, known as the processed G-data. It is imperative to understand the means by which access to G-data is granted. If all of these characteristics are understood, then it may be concluded that the VCO "Decomposition" is clearly defined.

6.2. Composition.

The VCO "Composition" generates an image of the type of the inclusive (higher level) part. An information image of this part type is saved. Similar to the PDS, there exists a particular composition scheme (PCS) for decomposition. The description of the PCS and the composition mechanism (CM) are distinct. PCS refers to the fundamental concept of organizing an object by assembling its constituent parts according to a certain formula, written text, or algorithm. CM is a way of attaching another specific part to an enclosing part or to an object; it is an algorithm. In a tree diagram illustrating decomposition, the PCS is coupled with a specific node that symbolizes the type of part that will be assembled. The construction of the CM is consistently preceded by the construction of the PCS.

The specification of the CM is determined when it is known:

- The specific conditions, which are required for initiating composition.
- How the first step is executed.
- How the next steps are executed.
- Under what conditions the composition terminates.

- What operations are required prior to or subsequent to the final step.

Although not only the first or final steps might be exceptional. Each step of the composition can be determined by a particular execution.

In order to present the MC initially as a VCO "Composition" and then as an algorithm that may be executed as a computer program fragment, it is essential to have a comprehensive understanding of all this information.

The components utilized in the composition process can be chosen from a specific collection, meaning that the components can be selected or searched. There can be an excess of components required to generate a new instance of a part of the same type. A new node is added to the tree to represent the type of this new part, which is coupled with the component part type by an edge. Multiple composition schemes can be employed to constitute a part of one new type. By using the next scheme, a previously non-existent type of part will be generated. Out of all the varieties of types that are generated, only one will be actual, while the others will be hypothetical. The actual type of component will be chosen according to a certain condition. Hypothetical types of parts are either included in the CDS or they are neglected. However, if they are preserved, then they must be explicitly recorded in the CDS.

In order for the VCO "Composition" to be considered valid, it is necessary to determine the specific location and the manner in which the processed G-data is saved. Furthermore, it is imperative to understand how access to the G-data is provided.

6.3. Deduction.

Deductive inference, also known as VCO "Deduction," can be executed by the human intellect, on paper, on a computer, or by other means. The VCO "Deduction" can be described using words, and (or) formulae, and (or) algorithms. If deductive inference can be expressed algorithmically, it implies that the process for deductive inference has been precisely defined.

In the tree diagram representing the object being worked with, the deductive inference process is correlated with a node that possesses the initial properties for further inference. The VCO "Deduction" is deemed to be specified if it is known:

- Under what conditions to start inference.

- How the initial step is executed. This refers to how the value of the property (or properties) of the first part of the object is determined.
- How the next steps are executed.
- Under what conditions the inference is terminated.
- What operations are required to execute prior to or subsequent to the final step.

If information images of parts of an object were previously generated using the process of VCO “Decomposition”, then in the current processing, just the VCO “Deduction” is executed. If the properties of the components in the form of information images have not been previously generated, then they are composed simultaneously with the VCO “Deduction”.

Multiple VCO “Deduction” can be executed using the identical collection of information images. The subsequent iteration of the VCO “Deduction” process may produce a result that significantly deviates from the initial VCO “Deduction”, resulting in the generation of a part of a different type. In this scenario, a new type of object generates a property within the tree structure. Prior inference results are either saved or neglected. Only the final inference result is saved. Although nothing prevents to save the results of prior deductive inferences.

The following scenario is possible. The decomposition was completed earlier. Information images of types of parts and types of their properties have been composed. However, it is important to note that not all information images may be effectively utilized in deductive inference. It is presumed that the components chosen are those that generate an information image that fulfils specific conditions.

Both the outcome of decomposition and the outcome of deductive inference may fail to meet specific conditions and could be discarded. If the outcome of the decomposition is discarded, then the associated tree vertex is deleted. If the outcome of a deductive inference is discarded, then the accompanying E-data or A-data is zeroed.

In order for the VCO “Deduction” to be considered valid, it is necessary to determine the specific location and the manner in which the processed G-data are arranged, as well as it is imperative to understand how access to the G-data is provided.

6.4. Induction.

Inductive inference (VCO “Induction”), is a cognitive process that occurs in the human mind, as well as on paper or on a computer. The VCO “Induction” can be described using words, and (or) formulae, and (or) algorithm. If inductive inference can be expressed algorithmically, it implies that the process for inductive inference has been precisely defined. In the tree diagram representing the object being worked with, the VCO “Induction” is correlated with a node whose properties are output or resultant for subsequent output. The VCO “Induction” is deemed to be specified if it is known:

- Under what conditions to start inductive inference.
- How the initial step is executed. That is, how the property of the initial part is processed.
- How the next steps are executed.
- Under what conditions the inference is terminated.
- What operations are required to execute prior to or subsequent to the final step.

The following scenario is possible. The collection of data elements (C-property) to be examined and processed has been completed earlier. The composition or linkage has been completed earlier, and images of the information components are accessible for analysis. The components that are or were susceptible to composition may result in a smaller or larger quantity than anticipated for inductive inference. Consequently, there will be a decrease or increase in the number of information images. The VCO “Induction” should consider this situation.

The inductive inference for a specific part (type of part) can be repeated by either using the previous VCO “Induction” or by using another VCO “Induction”. Even the same VCO “Induction” may require multiple passes over the parts being processed. If the subsequent VCO “Induction” varies from the initial one applied, it can result in the generation of a new type and property, or a new property, or only a new value of an already existing property. In the first case, an additional edge and vertex are appended to the tree. The outcomes of the prior inference are either saved or neglected. It is possible to save all inference subproducts, but they must be appropriately labelled.

The following scenario is possible. An information image of the created part type and one of the property types of this part type has been generated. During the process of inductive inference, the C-data was utilized, which contained the properties of the components. Some components of the C-data may not be useful for inductive inference. That is, it is presumed to choose those elements that satisfy a specific condition. The VCO "Induction" should be capable of considering this particular situation.

In order for the VCO "Induction" to be considered valid, it is necessary to determine the specific location and the manner in which the processed G-data are arranged. It is imperative to understand how access to the G-data is provided.

6.5. Supplementary remarks.

DM, CM, formula VCO "Deduction", and formula VCO "Induction" can be specified as a list. When executing any of the specified VCO, the processed components may vary in quantity (more or less) compared to the items in the list. Clarity is required in this matter. Both VCO "Decomposition" and VCO "Composition" can be repeated multiple times as specified in the list. That is, there are many decomposed parts, and the list describes the decomposition or composition only for a limited number of them. In this case, the VCO "Decomposition" or VCO "Composition" are reiterated for the subsequent components as per the provided list. The conditions for completing any of the VCOs in the case of a list may include the following:

- Executed a certain number of iterations.
- Completed all the tasks on the list.
- During the execution process, a specific condition has been met. It is assumed that the parameter being checked in the condition changes during the execution of operations in this case. The condition can be intricate – the outcome of simpler conditions synthesis.
- Stopped by external influence.

The generic name for VCO "Induction" and VCO "Deduction" is synthetic operations that execute synthetic dependencies.

VII. Arranging data.

The CDS is depicted as a hierarchical structure like a tree. Every node in this tree

corresponds to a distinct part type. Every type of part and, correspondingly, every node of the tree is coupled with the primary A-property. In order to performance CO, it must be known how and where the G-data, which compose the primary A-data, are arranged. There are numerous possible modifications for the arrangement of G-data.

The components of the A-property, that is, the G-data, can be stored on several types of media. G-data can serve as both an input and an output. G-data can be stored on either local or corporate networks, as well as on the Internet. G-data can be processed from both durable media and online mode. Additionally, the output of G-data can be directed to both durable media and to terminal device. Data, that is, the main A-property, can be arranged in one mode for one node, and in another mode for another node.

The listing of components in the description of the main A-property can have a specific order. Components may be accessible under conditional or alternative basis. Components can be arranged either sequentially based on their listing or in relation to one another. The placement of G-data while describing the main A-data can indicate the sequential arrangement of G-data on storage media and throughout the retrieval process from networks or terminal devices. An algorithm can determine the sequence in which G-data is accessed.

VIII. Building a synthesis circuit.

CO are utilized to ensure the building of an object properties synthesis circuit (OPSC). The properties of parts of an object of the same type can be coupled in a manner where certain properties serve as the basis for determining or calculating other properties. These kind of dependencies are referred to as analytical. Analytical dependencies can be represented using a mathematical equation, textual explanation, or algorithmic procedure. An algorithm fragment and a program fragment represent the dependencies. During the execution of a program, the value of one property is transformed into the value of another property – a property transformation operation or executive member (EM) is executed. The OPSC composes of EM. EM is the implementation of both synthetic and analytical dependencies between properties; it is the implementation of VCO and operations of property values transformation.

Each EM implements the computing or calculating of one or multiple G-properties[6] based on the dependency that couples the given properties with the sought ones. An individual EM or a cluster of neighboring EM form a synthesis circuit fragment (SCF).

A circuit fragment is correlated with the original object (zero level of CDS). In this scenario, the OPSC consists of one or many EM (as shown in Figure 3a), which can vary in complexity. These EM implement the dependency between the properties that describe the original object (are correlated with the original object). The tree consists of a single node. The algorithm that implements this OPSC in this particular scenario follows the structure: input – EM – output. If this fragment does not include any VCO, then this fragment will serve as the entire synthesis circuit (Fig. 3a) – (F_1, ..., F_m).

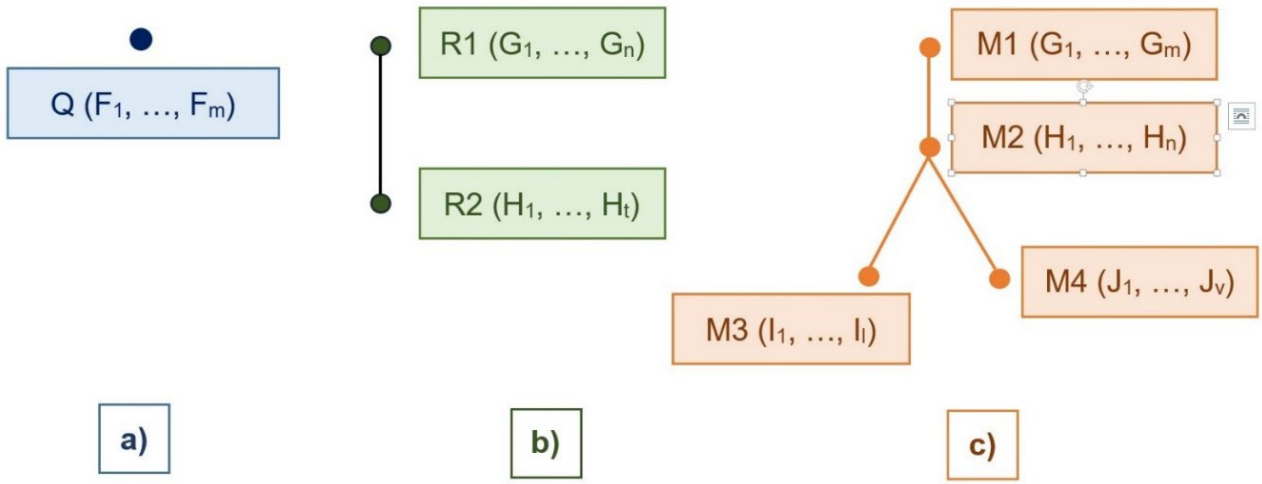

Figure 3: Synthesis circuit alternatives

a) The synthesis circuit does not involve any component operations

b) Synthesis circuit fragment c) General OPSC

Authoring

A more algorithmically complex construction of OPSC includes either a synthetic dependency or a VCO. If there is at least one VCO present, it indicates that the CDS will include parts of a subordinate type (Fig. 3b) – R2. Thus, this type is coupled with the specific SCF (H_1, ..., H_t). In terms of trees, this implies the presence of an outgoing

[6] The detailed descriptions of synthesis circuit can be found in [9, 10].

leg that mates two nodes: a node that denotes the original object in the tree (level zero), and a node that denotes the descendant type of the part (level one). The synthesis circuit for this fragment of the decomposition scheme will have the mode: (G_1, ..., (H_1, ..., H_l), ..., G_n).

If the implementation of a synthetic dependency involves deductive inference, then the algorithm may also contain decomposition. The parts can be decomposed earlier and information about them is saved on media. The tree will consist of two vertexes. The algorithm appears as follows:

- For the object itself (for the root vertex): input – implementation of a synthetic dependency (one EM) – output.
- For a part of an object (for an end vertex): input – processing of properties (C-data) of one part (one EM) – output.

The synthesis circuit for CDS can be represented as follows:

(G_1, ..., (H_1, ... (I_1, ..., I_l), ..., (J_1, ..., J_v), ..., H_n), ..., G_m).

If there is more than one VCO in a circuit fragment that corresponds to the original object, each VCO will generate a unique part type. A new leg with a node will be added to the tree. There may be instances where VCO "Decomposition" and VCO "Deduction" have a common type of part. In this scenario, there will be a single shared leg in the tree. There may be instances where the VCO "Composition" and the VCO "Induction" correlate with the same type of part.

Each fragment of OPSC at the first level may contain VCO processes. Consequently, they result in the generation of part types of level two. There are no restrictions on the quantity of VCO that can exist within any SCF, nor on the number of levels that a decomposition tree can encompass.

A general OPSC connects all SCF that are included in CDS. SCF levels with a greater numerical value are recursively run-in within SCF levels with a lower numerical value.

CDS can contain an arbitrary number of types of parts. Regarding trees, this implies that a decomposition tree can contain an arbitrary number of nodes. Each part type, like the original object, can contain an arbitrary number of constituent part types.

In tree terms, this indicates that an arbitrary number of legs can emanate from each node. There are no restrictions on the nesting of part types. Regarding trees, this implies that a decomposition tree can contain an arbitrary number of levels.

Each type of part can be correlated to a SCF. This fragment can contain both synthetic and analytical dependencies. A fragment can compose any dependencies.

SCF, which correlates to one type of object part (with one tree node), can contain an arbitrary number of EM. EM can possess a relational sequence, which is determined in relation to one another. Within the SCF, the outcome of executing a single EM might serve as the source data for another EM. In this instance, they are catenated by a sequential catenation. A sequential catenation can be used to catenate and non-interconnected EM.

EM themselves can have attributes. EM can be either conditional or unconditional. Conditional EM can be aggregated into clusters using an alternative catenation. This group is executed exclusively for a certain value of a particular parameter (which is also a G-property). Conditional constructions or alternative catenations may be nested. The depth of nesting is not limited. EM may have fixed places inside the OPSC. EM can be allocated within the OPSC in a certain sequence relative to one another. EM can be aggregated into a constellation, which can function as a unified rung and be connected based on its attributes.

Each synthetic operation presupposes the existence of parts (and, accordingly, types of parts) of a deeper dividing (decomposition). As a result, additional levels will appear in the tree.

From the fact that more than one synthetic operation is implemented in OPSC, there will be multiple outgoing legs in the tree. The legs will stem from the current node. Due to the possibility of synthetic operations occurring for tree nodes (and, consequently, types of parts) at any level, the tree can possess an arbitrary number of levels.

The following operations on DS are possible. Multiple circuits are computed across. The circuit may branch. A branched circuit can interflow into one circuit.

A synthetic operation can replace the analytical one, and then decomposition is

executed and a new node in the tree is generated. Decomposition can lead to the appearance of one or more new nodes in the tree. An analytical dependency can substitute for the synthetic one. Then one node and any potential nodes branching from it are removed from the tree.

In the process of studying a practical problem and designing a solution, new G-properties may appear. Regarding trees, this implies that trees have the ability to expand in width and the number of levels within a tree can also rise.

The order of calculations specified by the OPSC can set the order of accessing the components (G-data) of the main A-data and vice versa. The order in which the components of the main A-data are accessed can impact the architecture of the OPSC.

IX. VCO synthesis.

The types of parts in CDS can be correlated with different numbers of VCO – more than one. Certain tasks must be executed concurrently within a running program. In order for this to occur, it is necessary to execute a synthesis operation. The most basic form of synthesis is executed throughout all types of VCO, including decomposition, composition, deductive inference, and inductive inference. All operations are correlated with the dominant node; however, they can be executed in both the dominant and subordinate nodes. Synthesis can connect any two VCO in any order.

- Two or more VCO “Decomposition”.
- Two or more VCO “Composition”.
- Two or more VCO “Deduction”.
- Two or more VCO “Induction”.
- An arbitrary number of VCO “Decomposition” and VCO “Deduction”.
- An arbitrary number of VCO “Decomposition” and VCO “Induction”.
- An arbitrary number of VCO “Deduction” and VCO “Induction”, etc.

Prior to the completion of CDS building, synthesis can be executed. Synthesis can be executed simultaneously with algorithm generation.

Various synthesis options are possible:

A. PDS or DM can be synthesized using either terms or algorithm operators.

B. Program fragments that execute two analytical dependencies, that is, two

transformation operations of property values, can be synthesized.

C. Decomposition and composition can be synthesized. Therefore, the subsequent component is extracted by decomposition and subsequently utilized as a component in the composition.

D. When decomposing, processing of the decomposition outcomes may be necessary before deductive inference. When composing, before making inductive inference, it may be necessary to process the outcomes of the composition.

E. It is possible to synthesize Any VCO and transformation operations of property values of any level of complexity. VCO and transformation operations of property values of any complexity can be synthesized.

Example 1. Two VCO “Induction” are synthesized. Each of them is executed based on a unique algorithm. The first one entails working with each individual part, while the second method focuses just on every second part (specifically, every second G-data).

Example 2. Both VCO “Decomposition” and VCO “Composition” are synthesized. When composing, only a subset of the parts received after decomposition are utilized, namely those parts that meet a specific condition.

F. VCO “Decomposition” can generate divisions that are incompatible with the VCO “Deduction”. VCO “Deduction” may necessitate obtaining parts of the object acquired by PDS. These two schemes can be synthesized. A similar mismatching may occur between the VCO “Composition” and the VCO “Induction”.

DS can also be synthesized. The synthesis of VCO leads to increased saturation and/or lengthening of the OPSC.

In order for synthesis to be performed prior to commencing the algorithm generation process, it is imperative that the tool environment includes synthesis tools as one of its capabilities.

During the examination of an application domain and an existing process that requires automation, the necessity for analysis may emerge. We are discussing analysis as

a procedure that is the inverse of the synthesis action. In order to achieve this, it is necessary to express the current process, its surroundings, and the objects involved in the process using the concepts and terminology of the theory, namely the notions of DS.

As shown in [11] a detailed explanation of how DS synthesis procedures are modelled. Although as shown in [12] a detailed explanation of how application domain analysis activities are represented in DS terms.

X. VCO and in-out operations synthesis.

The concept of G-data diverges from the conventional understanding of "Data" in the realm of information processing. In this concept, all information is regarded as G-data structures, which include E-data, A-data (simple, extended, complex), C-data (simple, extended, complex), and host A-data. In-out operations involve G-data as operands. VCO precisely synthesizes such operations. Any operation involving the input or output of G-data from / to an arbitrary medium is inherently a form of synthesis.

G-data can be inputted either from conventional, machine-readable media or from diverse sensors that transform analog data into digital data. Similarly, the G-data output is expected both to conventional, machine-readable media and devices that transform digital signals to analog.

G-data, upon input, is mirrored on a specific device, and thereafter generated for further processing or for storage / recording on media. Both the algorithm and the program in this scenario are products of synthesis. This synthesized program construction requires a minimum of three operations as operands. *Example:* inputting G-data from the keyboard and simultaneously displaying G-data on the monitor screen, then outputting G-data for subsequent recording or processing. Afterwards, the G-data is outputted for further recording or processing. Similarly, along with the output, G-data is duplicated or displayed on the monitor. If a program implements this, then it is also the product of synthesis. Synthesis involves three essential operations: data input, reflection (or duplication) of output G-data, and outputting (recording) G-data to media.

G-data on media can be packaged in blocks, clusters, containers, and other similar

formats. Upon input, it is necessary to unpack them in order to proceed with further processing. When outputting to media, it is necessary to package the P-data. In fact, these actions can also be classified as decomposition and composition. The parts with which these operations work are information objects. Blocks, clusters, or containers are ways of organizing information based on the inherent structure of the medium. That is what is correlated with machine processing. The decomposition and composition operations utilized are not only unrelated to the fundamental nature of information and its application domains, but they also necessitate synthesis.

The G-data is outputted to a medium where one coordinate is utilized to define the storage position. Media can possess a structure that necessitates the use of either two coordinates (such as on a screen) or three coordinates (such as on a 3D printer or in relation to the surrounding material world) to specify the storage place.

Output G-data on a two-coordinated screen is also considered as packaging. The output of G-data (text) on a two-coordinated sheet to be printed is also the result of synthesis. Similarly, it is possible to simultaneously output G-data to three-coordinated devices and other storage media. In this scenario, the operations of decomposition and composition (or packaging and unpacking) of information will be more intricate, resulting in more intricate synthesis procedures.

The conventional computational scheme using a computer “input – processing – output” with the simplest operations that constitute this scheme, is already a complex (multicomponent or multi-place) synthesis procedure. The “processing” component in this scheme may also be the outcome of synthesis. Furthermore, there is no limit to the number of components involved in the synthesis.

Every form of media must possess a minimum of one way of access. If there are multiple accesses, they can (or should) be synthesized or determined on a case-by-case basis.

The design of G-data input and output synthesis is contingent upon the nature of the interaction between the present process and the computer. This is an outcome resulting from the implications of setting 2.

Output may involve simultaneous composition or decomposition of information.

Output may involve simultaneous composition or decomposition of information across several DS. Schemes can be synthesized into one CDS. Upon output, information can be decomposed in accordance with one PDS and simultaneously composed in another PDS. In this scenario, there can be more than one of both the PDS and the PCS. In this scenario, DS can be synthesized. Furthermore, data can be decomposed and composed by repeated passes.

Multiple output streams can exist. Output can be copied (duplicated) and occur simultaneously on many mediums. In this scenario, packets containing G-data can be generated concurrently, albeit using distinct schemes.

The G-data stream may consist of packets containing G-data of different levels of the original (or main) decomposition scheme implemented by the program. In this scenario, packets containing G-data can be mixed on the media.

XI. Complete decomposition scheme.

CDS and OPSC serve as the primary source of information for generating algorithms and programs[7]. The CDS design by the dividing method assumes the existence of the following EM. CDS can be saturated with CO. This is a generalized algorithm 1:

1. Decomposition of the original object.
2. VCO “Deduction”.
3. Operation of transforming property values.
4. VCO “Induction”.
5. Composition of the output object.

The CDS design by the inclusion method[8] assumes the existence of the following EM. This is a generalized algorithm 2:

1. The VCO “Induction”.
2. Composition of the inclusive object.
3. Operation of transforming property values.
4. VCO “Deduction”.

[7] To formulate an algorithm and develop a program, it remains essential to delineate the procedures for storing data on storage media and the methods for accessing by this data.
[8] The division method and the inclusion method are explicated within the context of [10].

5. Decomposition of the inclusive object.

These two algorithms provide general or ideal formal algorithms for solving issues in any industry using methods of division decomposition or inclusion decomposition[9].

Numerous iterations of these two algorithms are conceivable.

A. OPSC is divided into two SCF. A distinct algorithm and program have been built for every SCF. These programs are executed at different times with different schedules. The sequence of executing these SCF must ensure that the OPSC is not violated. However, the temporal interval between the completion of each part may differ. It is possible that the first SCF may be executed once in a practical scenario, but the second SCF might be executed multiple times.

B. OPSC can be divided at any level of CDS. Even a fragment that is coupled to a certain part type can be divided. The respective programs are executed at distinct time, however address a shared issue. The OPSC can be divided into parts of more than two.

C. OPSC is divided into two SCF. Throughout the calculating process, a multitude of scenarios can arise:

- The initial SCF is executed multiple times, while the subsequent SCF is executed only once.
- The initial SCF is executed multiple times throughout the workday, whereas the subsequent one is only executed once at the conclusion of the workday.
- The initial SCF is executed multiple times, while the subsequent one is executed on an as-needed basis.
- The initial SCF is necessarily executed once, and then both the initial and subsequent SCF are executed many times on an as-needed basis.
- Both the initial and subsequent generalized algorithms can be separated in terms of G-data input and output. The available options include: data input into the computer and output; input and only processing, and output later; only processing (input was executed earlier, and output will be later);

[9] This assertion holds validity within the domain of Software Engineering.

processing and output, and so on. In this scenario, the computation, which is executed sequentially (input, processing, and output), is the outcome of synthesis.

CDS can be the result of the synthesis of any multi-level schemes. The same statement in terms of trees: CDS can be the result of any previously constructed trees.

Only OPSC (text and graphics) can be taken into account for DS analysis. In order to do OPSC analysis, the inclusion and arrangement of part types, property types, main A-data, input G-data, and output G-data can be extracted and organized into a tree structure.

XII. Research methodology.

GTAA examines computer-processed information from three distinct perspectives or contexts:

- Application information is provided in terms of specific application areas.
- Abstract information is conveyed using DS and G-data.
- Traditional computer processing information is handled using conventional programming languages.

There are three types of calculations:

- Particular application calculations that can be executed without relying on a computer.
- Abstract calculations are conveyed in DS terms.
- Computer-based computations – utilizing contemporary algorithmic languages to express terms and concepts.

In essence, the scheme of transitioning from problem representation within one domain to another unfolds as follows: domain experts propose application computations, system designers document these computations as G-data and DS, and software systems are subsequently created using algorithmic languages[10]. To ensure coherence in terminology and facilitate shared comprehension among analysts, the cultivation of suitable terminology becomes imperative.

[10] Upon the development of GTAA, the ultimate phase of implementation could involve the creation of a program written in machine code.

A theoretical model of the decomposition scheme is essential[11], also referred to as a canonical or universal decomposition scheme. This model facilitates empirical study aimed at describing, analyzing, and organizing OPSC and DS, along with their attributes, within applied domains beyond computer-mediated contexts. The goal is to identify commonalities between diverse OPSC and DS, refining them to the level of algorithms and practical application programs. The canonical decomposition scheme can also serve as a theoretical canonical model for algorithms, enabling the formal representation of described application algorithms and subsequent exploration of their attributes.

The objects of research may encompass operational application programs and software systems. Systematic and comprehensive research should commence by theoretically modelling real, previously executed programs, thereby enabling the identification of theoretical deficiencies. Additionally, algorithms facilitating access to G-data on media are also the objects of research.

The research approach will involve constructing rows of algorithms that implement access methods for placing G-data. The task will involve attempting to discover a universal (canonical) access algorithm. If one is discovered, it will enable the compilation of a list of connections: algorithm's uniqueness + factor.

Uniqueness refers to deviations from the canonical algorithm in this series. The variations in algorithmic constructions establishing distinctiveness will determine the factor. When analyzing a series of algorithms, it is important to identify a common generalized algorithmic construction, and then, identify the factors that generate variations or modifications in this canonical algorithm.

This approach is evident in two specific instances.

Example 1. A task generated by inputting G-data into computer memory using a keyboard. Similar to the development and programming of operations for manipulating abstract data using a keyboard: character input, position return, character deletion, jumping forward a certain number of characters, character searching, reverse character searching, and forward searching etc., it is essential to develop and program operations

[11] A theoretical model serves as a hypothetical construction that functions similarly to models such as those of atomic nuclei, the human body, the periodic table of chemical elements, and pivotal models within various branches of mathematics.

for manipulating G-data: inputting E-data, A-data, and C-data; searching for E-data, A-data, and C-data, and navigating forward or backward through G-data while viewing it on the screen. The execution of input operations will be the outcome of building a row of algorithms.

Example 2. A task generated by outputting G-data on a two-coordinate display. Outputting a character string on the display requires a series of actions: moving the cursor to the end of the string, the cursor to the beginning of the string, moving the cursor at a particular spot on the screen, etc. Just like these operations, it is imperative to build G-data management operations:

- Output E-data, A-data, and C-data to the current location on the screen.
- Output E-data, A-data, and C-data to any location on the screen. Including to the beginning, to the end of the screen or to the end of the string.
- Displaying G-data on the screen while considering the dimensions of the G-data.

These and other operations should be developed through the construction and analysis of various algorithms[12].

The proposed methodology enables the conduct of empirical research independently of practical programming and specific projects.

XIII. Research directions.

Additional research is required to enhance the practicality and efficiency of the proposed theory in the field of programming technology. These studies encompass conducting research in the following domains:

1. Enhancing and optimizing the generalized algorithms 1 and 2 to be effectively utilized in describing problems in practical fields. Given that these algorithms are known as OPSC, it is important to take into account the diversity of OPSC. Real OPSC can be both incomplete and saturated with EM, and manifest themselves in various variants.
2. Examination of various modes of PDS in their multitude. It is important to take into account situations where an object or object type possesses multiple types of properties. Examine the PDS conditional, alternative, and executed in

[12] Comparable work is documented in [13, 14, 15]

a certain order.

3. Examination of various modes of PCS.
4. Research and description of the different modes of MD and CM, as well as the mechanism of deductive inference and the mechanism of inductive inference from the point of view of the variety of algorithmic solutions.
5. Investigation into the feasibility of utilizing VCO multiple times.
6. Investigation of more intricate forms of G-data.
7. Investigation of techniques for manipulating G-data. Classification of media types and methods of input and output of G-data.
8. Investigation of various methods of formatting (packing and unpacking) G-data.
9. Examination of various types of synthesis.
10. Examination of various modes of intricate (synthesized) input.
11. Examination of various modes of intricate (synthesized) output.
12. Enhancing and refining visual representations for CDS.
13. Development of translators of CDS (tree) into an algorithm (and into text in an algorithmic language), and in the future into a program in machine code.
14. Development of graphics for the algorithm, which is the result of converting (generating) CDS into the algorithm. This graphic should not be a traditional flowgraph, but a tree.
15. Creation of a synthesis procedure of CDS. A synthesis theory is required to provide the theory of applied algorithms with classification and potential generalisations of DS, novel attributes of DS, and synthesis schemes.
16. Creation of an updating procedure of CDS.
17. Creation of a protocol for utilising previously prepared CDS.

The aforementioned translator should incorporate the subsequent characteristics:

- conversion of CDS into an algorithm;
- synthesis of two or more CDS;
- synthesis of CDS and procedures for the input and output of G-data;
- synthesis of procedures solely for the input and output of G-data.

G-data are relatively simple information objects. Information constructions built in terms of decomposition schemes can be more complex due to the intricacies of real-world objects, which involve a greater number and variety of components as well as more connections between them. Consequently, new modes of G-data may appear.

The components or parts of real-world objects, their types, and the connections between them generate information images. A single design of a real object can generate multiple information images, leading to increased complexity and a higher number of images. Therefore, a practical methodology is needed to represent various real-world information objects as G-data and DS.

Conclusion.

The study presents a comprehensive exposition of the general theory of applied algorithms. The description provides a comprehensive account of the theory's attributes, which encompass the research methodology, field of research, object of research, subject of research, basic assumptions (axioms), main task of the theory, theoretical generalized model of the algorithm, purpose of research, and direction of research.

The theory encompasses two primary interconnected concepts: G-properties and virtual component operations. These concepts are the core innovation of the theory. They incorporate the concept of programming language operators with a significant level of universality. The research outlines a complete decomposition scheme and a circuit for generating properties. These two constructions are regarded as model of a universal or canonical algorithm.

The primary objective of the theory is to construct an accurate synthesis circuit. The algorithm and application program are generated based on the initial data of the complete decomposition scheme and synthesis circuit.

The designated areas for research investigation are delineated. The proposed methodology facilitates the conduct of empirical research in the field of algorithms, separate from the practical aspects of designing and programming applied software. Such a distinction is crucial for progress in the fields of programming and software engineering.

References.

[1] Ivar Jacobson and Bertrand Meyer. 2009. Methods need theory. *Dr. Dobb's J.,* 6 (August 2009). Retrieved June 25, 2024 from http://www.drdobbs.com/architecture-and-design/219100242

[2] David L. Parnas. 2010. Really rethinking "formal methods". *Comp.*, 43, 1 (Jan. 2010), 28–34. DOI: https://doi.org/10.1109/MC.2010.22

[3] Iaakov Exman, Dewayne E. Perry, Balbir Barn, and Paul Ralph. 2016. Separability principles for a general theory of software engineering. *ACM SIGSOFT Soft. Eng. Not.*, 41 (Feb. 2016), 25–27. DOI: https://doi.org/10.1145/2853073.2853093

[4] Dick Hamlet. 2012. Science, Mathematics, Computer Science. *Comp. J.*, 55, 1 (Jan. 2012), 99–110. DOI: https://doi.org/10.1093/comjnl/bxr090

[5] Edmund M. Clarke, Jeannette M. Wing et al. 1996. Methods: State of the Art and Future Directions. *ACM Comp. Surv.*, 28, 4 (Dec. 1996), 626–643. Retrieved June 25, 2024 from https://www.cs.columbia.edu/~wing/publications/ClarkeWing96.pdf

[6] Steve Adolph and Philippe Kruchten. 2013. Generating a useful theory of software engineering. In *Proceedings of 2013 2nd SEMAT Workshop on a general theory of software engineering (GTSE)*. (May 2013). IEEE Computer Society, San Francisco, CA, USA, 47–50. DOI: https://doi.org/10.1109/GTSE.2013.6613870

[7] Klaas-Jan Stol and Brian Fitzgerald. 2013. Uncovering Theories in Software Engineering. In *Proceedings of 2013 2nd SEMAT Workshop on a general theory of software engineering (GTSE)*. (May 2013). IEEE Computer Society, San Francisco, CA, USA, 5–14. DOI: https://doi.org/10.1109/GTSE.2013.6613863

[8] Victor R. Basili. 1996. The role of experimentation in software engineering: past, current, and future. In: *Proceedings of IEEE 18th International conference on software engineering*. (March 1996). Berlin, Germany: IEEE, 442–449. DOI: https://doi.org/10.1109/ICSE.1996.493439

[9] Valery G. Kolesnyk. 2012. DS-teoriya kak prototip teorii prikladnykh algoritmov [DS-theory as a prototype of the theory of applied algorithms]. *Probl. of*

program., 1 (Jan.2012), 17–33. Russian. Retrieved June 25, 2024 from https://pp.isofts.kiev.ua/index.php/ojs1/article/view/35/41

[10] Valery H. Kolisnyk. 2020. Formalizatsiia protsesu dekompozytsii [Formalization of the decomposition process]. *Epistemolog. Stud. in Philos., Soc. and Polit. Scien.*, 3, 2 (Dec. 2020), 26–43. Ukrainian. DOI: https://doi.org/10.15421/342030

[11] Valery Kolisnyk and Ostap Bodyk. 2023. Synthesis theory in action: deductive derivation of algorithms for application software. *Epistemolog. Stud. in Philos., Soc. and Polit. Scien.*, 6, 2 (Dec. 2023), 11–26. DOI: https://doi.org/10.15421/342321

[12] Valery H. Kolisnyk and Ostap Bodyk. 2024. Algorithmic approaches to universal analysis models. *Contr. Sys. and Comp.*, 2 (July 2024), 35–47. DOI: https://doi.org/10.15407/csc.2024.02.035

[13] Valery G. Kolesnyk. 2015. DS-teoriya. Issledovaniye faktorov deleniya r-dannykh dlya generatsii prikladnykh algoritmov. Chast 1 [DS-theory. Research of R-data division factors in order to generate applied algorithms. Part 1]. *Probl. of program.*, 3 (Sept. 2015), 3–12. Russian. Retrieved June 25, 2024 from https://pp.isofts.kiev.ua/index.php/ojs1/article/view/143/136

[14] Valery G. Kolesnyk. 2015. DS-teoriya. Issledovaniye faktorov deleniya r-dannykh dlya generatsii prikladnykh algoritmov. Chast 2 [DS-theory. Research of R-data division factors in order to generate applied algorithms. Part 2]. *Probl. of program.*, 4 (Dec. 2015),3–13. Russian. Retrieved June 25, 2024 from https://pp.isofts.kiev.ua/index.php/ojs1/article/view/155/149

[15] Valery G. Kolesnyk. 2016. DS-teoriya. Issledovaniye faktorov formatirovaniya r-dannykh [DS-theory. The research of R-data factors formatting]. *Probl. of program.*, 4 (Dec. 2016), 14–26. Russian. Retrieved June 25, 2024 from https://pp.isofts.kiev.ua/index.php/ojs1/article/view/209/201

Анотація. *Індустрія розробки програмного забезпечення вже давно визнала потребу в ефективній теорії для просування прогресу в технології програмування. Незважаючи на постійні зусилля, створення комплексної теорії виявилося величезним викликом. Дослідження представляє комплексну теорію прикладних алгоритмів, яка має на меті заповнити цю*

прогалину. Запропонована авторами теорія зосереджується на процесі програмування як найскладнішому компоненті розробки програмного забезпечення та прагне здійснити глибоку революцію. Теорія базується на двох фундаментальних концепціях: G-властивості та операції віртуальних компонентів. Ці конструкції служать основоположними елементами для ретельної схеми декомпозиції та схеми для створення атрибутів об'єкта, разом становлячи модель для універсального або канонічного алгоритму. Основною метою теорії є розробка основи для точного поєднання алгоритмів і прикладних програм. Схема декомпозиції та цикл синтезу складають основу цього підходу, дозволяючи створювати програмні рішення, які є ефективнішими та ефективнішими.

У статті наголошується на важливості проведення обширних емпіричних досліджень для перевірки та вдосконалення запропонованої теорії. Автори прагнуть подолати теоретичні перешкоди в технології програмування, щоб сприяти значному прогресу в розробці програмного забезпечення. Це, у свою чергу, підвищить ефективність і досконалість процесів розробки програмного забезпечення, що зрештою сприятиме розвитку галузі.

Ключові слова: *схема декомпозиції, тип об'єкта, тип частини об'єкта, властивості об'єкта, загальні поняття програмування; алгоритм.*

Статтю надіслано: 24.02.2025 р.