https://orcid.org/0000-0002-4168-7190

UDC 623.764

P.S. SAPATY[*]

SPATIAL GRASP MODEL FOR DYNAMIC DISTRIBUTED SYSTEMS

[*]Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, Kyiv, Ukraine

Анотація. *Розробляються все більш складні розподілені та інтелектуальні системи, які використовуються в економіці, екології, зв'язку, безпеці та обороні й охоплюють як наземні, так і космічні середовища. Ефективне управління такими системами, особливо в динамічних та непередбачуваних ситуаціях, потребує серйозних досліджень та розвитку у науково-технічних сферах. Їх традиційне уявлення як таких, що складаються з окремих частин і працюють за певними алгоритмами й обмінюються повідомленнями, стає нераціональним, оскільки такі системи потребують набагато сильнішої інтеграції, щоб діяти як цілісні утворення, які орієнтовані на досягнення глобальних і часто змінних цілей. Ця стаття зосереджена на абсолютно іншій парадигмі організації та управління великими динамічними та розподіленими системами. Дана парадигма розширює і трансформує традиційне поняття алгоритму для опису логіки обробки знань. Завдяки цій парадигмі алгоритм може існувати, поширюватися і функціонувати як цілісне утворення у будь-яких розподілених просторах, що можуть постійно змінювати свої обсяги та структуру. Беручи до уваги деякі організаційні особливості моделі, які навіть схожі з небезпечними вірусами, також із пандемією, дана модель просторового захоплення (ПЗ) розглядається у статті як на філософському рівні, так і на рівні її реалізації. Особлива увага приділяється впровадженню спеціальних просторових діаграм або карт, які розширюють традиційні алгоритмічні блок-схеми і роблять їх придатними для застосування безпосередньо у розподілених просторах, що корисно для широкого використання і дослідження цієї моделі. Окрема увага приділяється шляхам використання моделі ПЗ для створення Технології просторового захоплення і її базової Мови просторового захоплення, детально описаних у попередніх численних публікаціях. У статті наводяться декілька простих прикладів керування розподіленими мережами, колективною поведінкою взаємодіючих людей і роботів, процесом знешкодження космічного сміття за допомогою угрупування «прибиральних» супутників, а також моделювання поширення вірусу та вакцинації проти нього, на яких пояснюються переваги ПЗ перед традиційними системами.*

Ключові слова: *алгоритм, блок-схема, розподілені системи, просторове захоплення, просторова діаграма, цілісні рішення, управління мережею, колективна поведінка, космічне сміття, глобальні віруси.*

Abstract. *More complex distributed and intelligent systems which relate to economy, ecology, communications, security and defense, and cover both terrestrial and celestial environments are being developed. Their efficient management, especially in dynamic and unpredictable situations, needs serious investigations and development in scientific and technological areas. Their traditional representations as parts operating by certain algorithms and exchanging messages are becoming inadequate as such systems need much stronger integration to operate as holistic organisms pursuing global and often varying goals. This paper is focused on a completely different paradigm for organization and management of large dynamic and distributed systems. This paradigm extends and transforms the notion of an algorithm for the description of knowledge processing logic. Moreover, it allows it to exist, propagate and operate as an integral whole in any distributed spaces which may constantly change their volumes and structures. Taking into consideration some organizational features related to dangerous viruses, as well as recent pandemics, this ubiquitous Spatial Grasp (SG) model is presented in the paper at philosophical and implementation levels, together with the introduction of special spatial charts for its exhibition and studies, which extend traditional algorithmic flowcharts towards working directly in distributed spaces. Utilization of this model for*

ISSN 1028-9763. Математичні машини і системи. 2021. № 3

the creation of resultant Spatial Grasp Technology and its basic Spatial Grasp Language, already described in details in numerous publications, is briefed as well. Elementary examples of dealing with distributed networks, collective human-robotic behavior, removal of space debris by a constellation of cleaning satellites and simulating the spread of virus and vaccination against it explain SG advantages over traditional system organizations.
Keywords: *algorithm, flowchart, distributed systems, spatial grasp, spatial chart, holistic solutions, network management, collective behavior, space debris, global viruses.*

DOI: 10.34121/1028-9763-2021-3-21

1. Introduction

This paper inherits some practical works concerning the creation of citywide computer networks in Kyiv, Ukraine, which integrated different institutes of the National Academy of Sciences and other organizations, from the end of the 60s and well before the appearance of the Internet. By spreading a fully interpreted scenario code in a wavelike mode between different computers it is possible to solve complex analytical and numerical problems on heterogeneous computer networks that were difficult to organize on individual computers. These works resulted in a new management concept, real distributed control methodology and technology which was further developed in different countries (including former Czechoslovakia, Germany, the UK, the US, Canada and Japan), with application in such areas as intelligent network management, industry, social systems, psychology, collective robotics, security and defense. A great number of successful implementations of this approach had been made in such languages as Analytic, Fortran, Lisp and C. There has been developed a special high-level recursive Spatial Grasp Language in which distributed, parallel and holistic algorithms could be expressed with resultant spatial scenarios up to a hundred time more compact and proportionally simpler than in other languages. All this activity resulted in a European patent, more than 200 international publications, including six books, with another one in process.

The aim of the current paper is to generalize mentioned above works and obtained experience in the form of a radically new computational, control and management model, as a natural extension of the traditional concept of algorithm and its exhibition by flowcharts, with potential applications in large distributed systems operating in combined terrestrial and celestial environments. This model may allow expressing complex solutions in distributed spaces with the feeling of direct staying and moving through them, as well as obtaining their overall vision and understanding in a holistic manner.

Section 3 describes the basics of Spatial Grasp model and explains how it differs from the conventional algorithm. It also introduces a new type of a chart – spatial chart as a further development of traditional flowcharts for the description and analysis of scenarios operating directly in distributed spaces. The section shows how a set of actions can be described in SG and exhibited by spatial charts, including the use of supervising repetition of control rules, sequencing and branching in spatial scenarios, as well expression and exchange of spatial dataflow with organizations that tend to be unlimitedly hierarchical and recursive.

Section 4 briefs the main elements of Spatial Grasp Technology (SGT) and its high-level recursive Spatial Grasp Language (SGL) based on SG philosophy with already existing numerous publications and books focused on the mentioned approach, its implementation and numerous applications. This includes different types of distributed worlds GGT operates with, various constants which may represent both information and physical matter, repertoire of spatial variables of SGL which can be either stationary or mobile, main types of SGL rules which can be nested, different control states provided by SGL scenarios propagation, as well as general organization of the distributed and networked SGL interpreter.

Section 5 provides examples of solution to different distributed problems under SG model which confirm its advantages in comparison with traditional parallel and distributed system organizations. These solutions include network management with finding and collecting a path be-

tween different nodes with issuing it at the final or starting node, organization of collective behavior of a mixed human-robotic team, using large constellation of debris-cleaning satellites operating together under a global goal, as well as simulation of the worldwide spread of the malicious virus and distributed vaccination used for the fight against it. In all these examples the solutions are provided by spatial SGL scenarios as self-evolving and matching distributed dynamic spaces which, unlike other approaches, do not require classification as systems in advance of these solutions.

Section 6 concludes the paper summarizing the obtained results and mentioning some new activities planned in this area, including new SGT implementation, fresh patenting of the SG model and an upcoming book describing the use of the SG model and the resultant technology for the management of integrated terrestrial and celestial systems.

2. Algorithm and Flowcharts

Algorithm is a finite sequence of well-defined, computer-implementable instructions which are usually used to solve a class of specific problems or to perform a computation [1–3]. Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, etc. In contrast, a heuristic is a used in problem solving technique that utilizes practical methods and/or various estimates in order to produce solutions that may not be optimal but are sufficient in the given circumstances [4].

A flowchart is a type of diagram that represents a workflow or a process [5, 6]. Flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach used to solve a task. Such diagrams show steps as boxes of various kinds whose order is defined with arrows. Flowcharts are used for analyzing, designing, documenting or managing a process or a program in various fields. Examples of simple flowcharts are shown in Fig. 1 (a processing step is usually depicted as a rectangular box and a decision as a diamond).

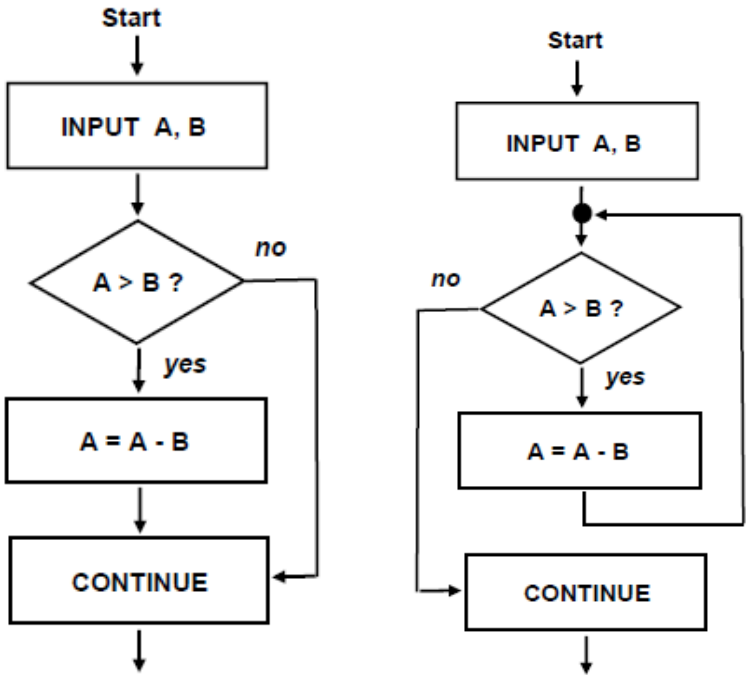

Figure 1 – Examples of flowcharts

3. Spatial Grasp versus Traditional Algorithm

3.1. Elementary Spatial Grasp explanation

SG model [7–13], which may be considered as a spatial extension and interpretation of the concept of algorithm, operates by recursive scenarios self-spreading in physical and virtual worlds while creating, matching, transforming, processing and managing them. The interpreted scenario text may not stay permanently in any point(s) (if it is not required) and can move in space while carrying its still unprocessed remainder and omitting utilized parts (if the latter is no more necessary). Below there are consistently presented and explained SG model on elementary operations, more complex solutions with many actions, universal recursive definition of any SG scenarios.

Single operation

Imagine yourself staying in some point of space (it can be a sheet of paper, computer memory, or any place in terrestrial or celestial environment) and writing:

```
44.55.
```

You will get this value in the current world point which may stay there indefinitely, without any name. Another example:

```
5+6.
```

This operation will produce the value 11 which will stay in this very point also without a name too. One more example:

```
R=5+6.
```

The result 11 will be assigned to a variable `R` and will stay in the starting point under this name. It may be subsequently accessed by name `R` if to come into this point again.

The next example is related to the physical space:

```
move(x55_y88).
```

From the current point in space you will move to another point with certain `x_y` coordinates and stay there. If you want to create a node named `John` in a virtual space staying indefinitely in this node just write:

```
create('John').
```

If the node `John` already exists, you may directly hop into it from the starting point and stay there as follows:

```
hop('John').
```

A single action may produce a multiple result, for example, by hopping simultaneously to virtual nodes `John`, `Peter` and `Alex` (if they already exist) and staying in all of them:

```
hop('John', 'Peter', 'Alex').
```

Such result can also be reached moving in parallel to a number of physical world locations from a starting point and staying there indefinitely, as follows:

```
move(x55_y88, x5_y12, x105_y92).
```

Summarizing stated above and other possible examples with a single action (named as `g` and applied in some `Start` point), we can get the result in some point of space (symbolically named also as `G`, but in capital) which may include the Start position as in first examples out of the given above. It is shown in Fig. 2 in the form of a special diagram or chart – a spatial chart (or

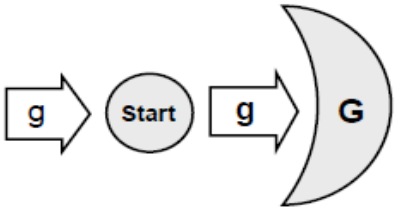

Figure 2 – The simplest spatial chart

simply spatiochart) – as an extension to a traditional flowchart used to represent operations and their groupings working directly in distributed spaces, what is pursued by SG philosophy and model.

Sequence of operations

Let us now consider an expression of a sequence of possible operations in space using semicolon as a delimiter between them.

Assigning it to a variable R, then changing its value and finally staying in the same position in space can be as follows:

```
R=15; R=R + 10.
```

Hopping to the virtual node `John` and then creating a new node `Peter`, building a relation between them `John` as Peter's `father`, and staying finally in the node `Peter`:

```
hop('John'); create(link('father'), node('Peter')).
```

Hopping to the virtual node `Peter` and then creating a `friend` relation to the already existing node `John`, staying finally in the node `John`:

```
hop('Peter'); linkup('friend', node('John')).
```

Moving to a physical location by its absolute coordinates and then twice shifting to other locations by given coordinate changes, staying finally in the node reached by the second shift:

```
move(x55_y88); shift(x11_y22); shift(x9_y45).
```

The mentioned examples and other variants with sequences of actions `gi` initially applied from a Start position, with the next action `gi+1` originating in all or some space positions reached by the previous action `gi`, can be represented the following way:

```
g1; g2; g3.
```

Their combined operation is shown by the spatial chart in Fig. 3 with regions (symbolically represented by crescent moon signs) reached by actions `gi` named as `Gi` too (which may generally include positions covered by the previous actions in the previous regions). It is also taken into account that the rest of the sequence has to propagate in space delivering descriptions of further actions and the already used operations are removed from the sequence as they are no longer necessary.

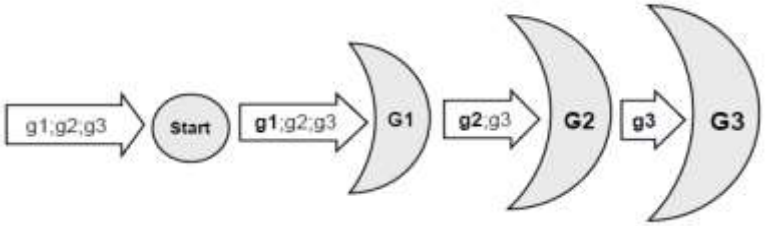

Figure 3 – Spatial chart for a sequence of actions in space

In Fig. 4, which provides a more detailed sequence of operations presented in Fig. 3, it is shown that the movement from regions `Gi` to `Gi+1` can be generally made in parallel from different points of `Gi` (potentially from the same points as well because `gi` can represent parallel

operations themselves), so operational sequences in reality can be replicated at any stage of their development.

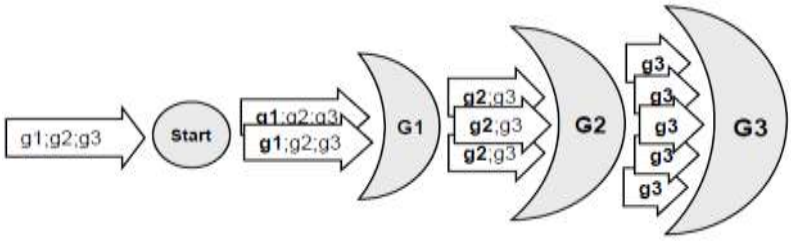

Figure 4 – Possible code replications during parallel space navigation

3.2. The use of rules

Each operation is usually based on certain *rules* defining its origin, sense and details of implementation, management and control. The concept of the rule as a certain entity and mechanism, which may either influence or even completely represent different operations, is considered below in a broader sense.

In sequencing of operations

In a more advanced organization of the sequence of operations there can be used different rules which can provide additional (often nonlocal) control, functionality and some more advanced processing and coverage of distributed spaces, as it follows for the rule `r1` and the operational sequence considered before:

```
r1(g1; g2; g3).
```

The rule will be activated in the position where the whole sequence is applied, like Start as before. Then it may influence the whole sequence of embraced operations receiving a feedback from its entire development (if functionality of the rule requires such a feedback), as shown in Fig. 5.

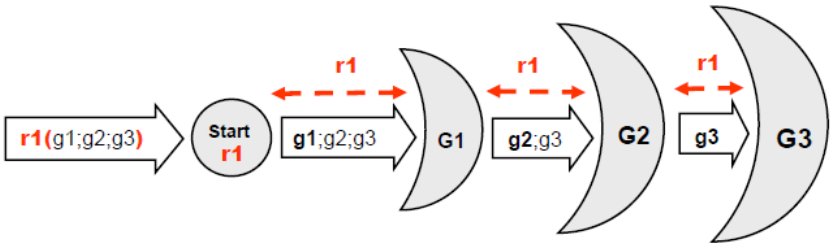

Figure 5 – Spatial chart with a control rule

The rule, for example, may represent such functionality as print, create, repeat and many other cases of nonlocal management and control. Let us consider a few examples in more details.

```
r1➔print.
```

According to this rule, the results obtained by the embraced sequence of operations (not only by its last operation `g3`, but also by `g1` and `g2` if they provide final results too) can be returned to the Start position and printed there finally staying in the Start.

```
r1➔create.
```

This rule can supply the space propagating operations, especially those describing movement in virtual spaces, with global power of creating these spaces or their individual elements if they are absent during this movement and therefore do not allow proceeding further. This means

that the same written sequence of actions `gi` can work in both space navigation and space creation modes, depending on circumstances.

`r1➔repeat.`

Under this rule, the sequence of operations `gi` at first is processed as usual, step by step, until the rest of it becomes empty, but after this it starts to work from the very beginning again, as shown in Fig. 6. The `repeat` rule always saves the already processed operations in their sequence (which could be removed without it), with the whole sequence repeatedly propagating and working in space until possible.

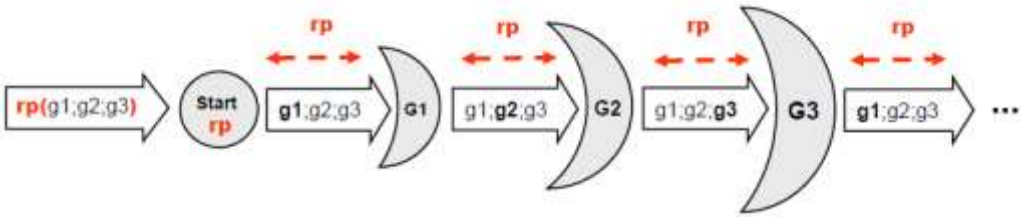

Figure 6 – Repeated navigation of space

The operational sequence can be embraced by any number of control rules, which can be nested, as it follows for the rules `r1` and `r2`, and shown in Fig. 7 (with `r1` activated in the Start position, and `r2` in the positions in space which belong to `G1` (there can be many of them) and reached by operation `g1`.

`r1(g1; r2(g2; g3)).`

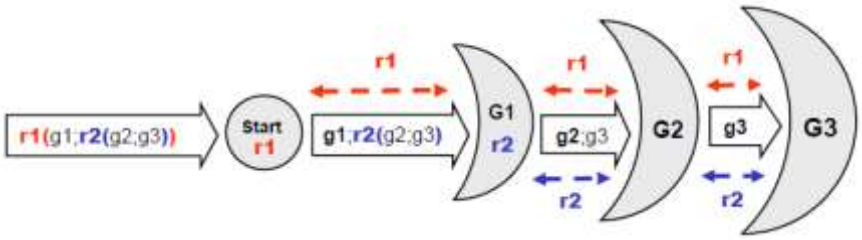

Figure 7 – Nested control rules

A few examples of combination of such nested rules:

`r1 ➔ create  r2 ➔ repeat`
`r1 ➔ print  r2 ➔ create`
`r1 ➔ repeat  r2 ➔ repeat`
`r1 ➔ repeat  r2 ➔ print`

For the last case, printing the results obtained by actions `g2` and `g3` will be organized in all positions of the regions `G1` reached by `g1` (which can be repeated by rule `r1` at the higher level).

In branching operations

Other rules may allow and supervise branching in space, with different branches (separated by comma) developing from the same positions in space (as shown in Fig. 8 where `r1` is used to control two branches and `r2` – to manage the sequence of operations belonging to the second branch).

`r1(g1, r2(g2; g3)).`

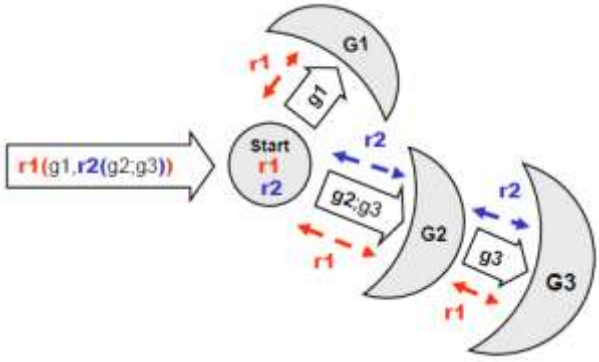

Figure 8 – Branching under control rule

Possible meaning of r1 and r2 may be as follows, with both rules activated in the Start position.

r1➔or r2➔create.

Rule r1, activating two branches in any order or in parallel from the same Start position, selects the first one in time replying with a positive termination result, considering its space locations and data obtained there as the final result, while ignoring all achievements of another branch. Rule r2 covers the sequence of operations g2 and g3 of the second branch, supplying them with creative power during space navigation (it will be completed and accepted if classified as the resultant branch by r1).

r1➔if r2➔print.

Rule r1 first launches branch g1 and, in case it terminates with a positive result, activates the second branch with g2 and g3 embraced by rule r2 which organizes printing their final results in the Start position with staying there (the latter is considered as holding the final result). If g1 results with failure, the final stay will be in the Start too but without any new result.

r1➔and r2➔repeat.

Rule r1 activates two branches in any order or in parallel, and only after both branches reply with their final success, r1 can confirm the whole success of this mission (with r2 organizing repetitive development of the sequence of two embraced operations which will eventually terminate). The successful positions reached in space by both branches will be considered as holding the final results of the scenario, with subsequent indefinite staying in them. If one of the branches replies with failure, the development of the other one will be terminated as soon as possible, as it is no longer necessary. After r1 fails as a whole, there will be no position to stay in space further, with Start just abandoned.

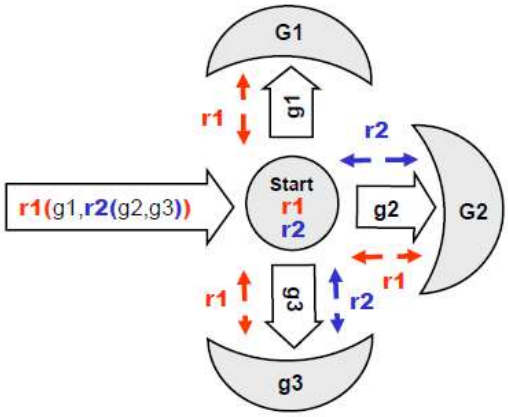

Figure 9 – Nested branching

We can also consider the development of operations g2 and g3 not in a sequence but in the branching mode as well (Fig. 9), with both rules activated in the Start position, where r1 embraces r2 and r2 coordinates all operations, i.e. g2 and g3, as follows:

r1(g1, r2(g2, g3)).

Possible examples of combinations of such rules:

```
r1 ➔ or     r2 ➔ or
r1 ➔ if     r2 ➔ and
r1 ➔ and    r2 ➔ or
r1 ➔ and    r2 ➔ and
```

3.3. Recursive hierarchy of scenarios

In Figures 2 to 9 there were shown operations `g1`, `g2`, `g3` which can be of any complexity. This means that each of them may itself be substituted by the whole scenarios like the ones shown in all these figures, and for which each operation could be substituted by any scenario over and over again, with such recursion potentially available to any depth. If, for example, to substitute operation `g2` of the previous scenario `r1(g1, r2(g2, g3))` shown in Fig. 9 with the scenario `(g4; r3(g5, g6))` enclosed in parentheses as a whole unit, we will receive a detailed combined scenario as follows (Fig. 10):

```
r1(g1, r2((g4; r3(g5; g6)), g3)).
```

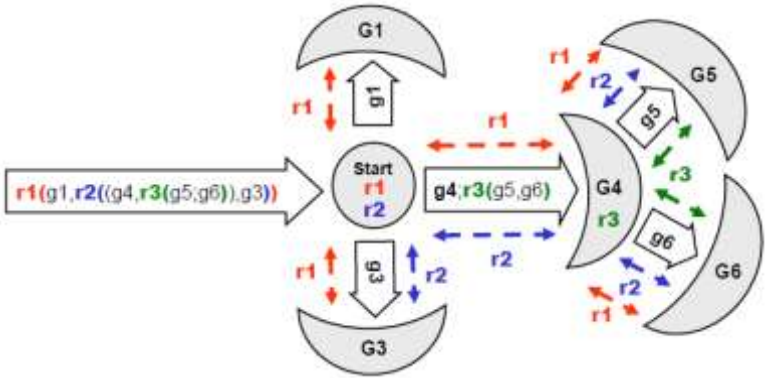

Figure 10 – Recursive extension of the scenario

3.4. Treating any operations as rules too

Collecting data for local processing

We may also consider all mentioned above operations `gi` as rules which can have certain parameters as local constants, variables or arbitrarily remoted items. For example, we may have a rule providing a `sum` of values, like:

```
sum(g1, g2).
```

Parameters of the rule, represented by operations `g1` and `g2` (which can also be treated as rules), may have these values directly or they may first need to obtain them remotely after the network search of any complexity and depth, as shown in Fig. 11. The result will be obtained in the point where the rule started which will also be the final space point for the whole rule from which any further developments may take place (with the reached positions `G1` and `G2` subsequently abandoned).

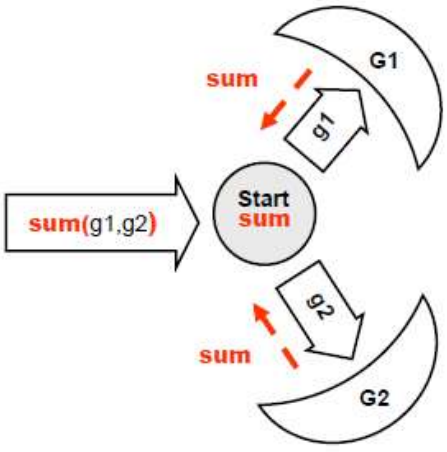

Figure 11 – Expressing the operation of summation according to the rule

Further development after this rule may be as follows (Fig. 12):

```
sum(g1, g2); g3; g4.
```

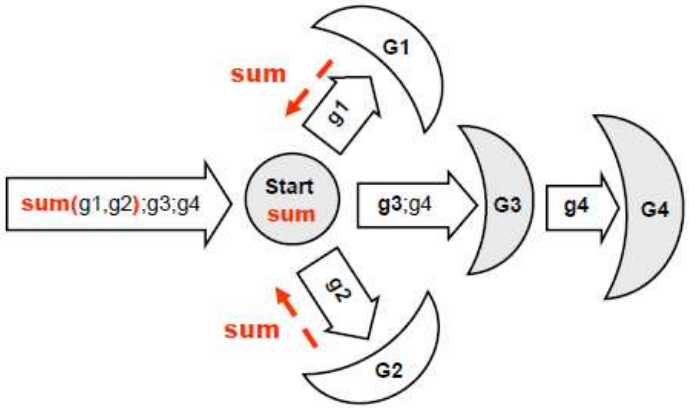

Figure 12 – Example of further development from the Start position of the rule

Local processing with the remote leaving of results

Instead of the rule summing values we may have any other ones, like, for example, finding maximum from the values received by `g1` and `g2` which will also produce the result in the rule starting position, the latter will also represent the final space position. But another rule like `maxdestination` (or `mxd`) may find the maximum value as before, but leave it in the space position where it was originally found, with declaring this space position as the final one in the rule, like, for example, `G2` in Fig. 13.

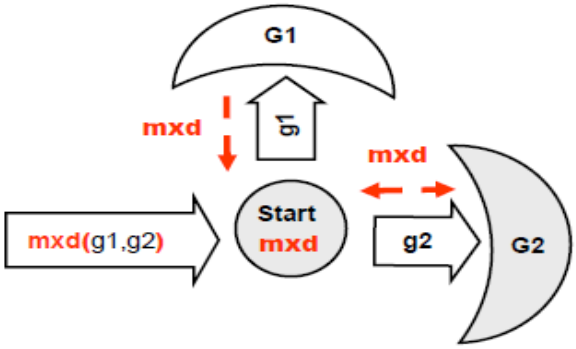

Figure 13 – Example of maximum destination rule

From the position `G2`, which may be remote, any further developments should take place (while abandoning `G1`), as follows (Fig. 14):

```
mxd(g1, g2); g3; g4.
```

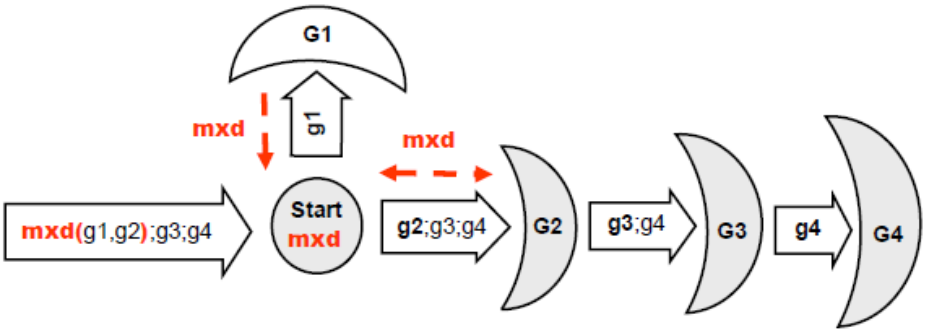

Figure 14 – Example of further development from the G2 position

3.5. Expressing sequences of operations with the rule

Above there were shown only few examples of structuring of spatial scenarios and rules coordinating their collections. In the provided examples semicolon was used to separate operations succeeding each other, comma – to separate branches starting from the same points. The same separator can also be utilized for any collections of actions, embracing their sequences by the rule `advance` (or `ad`) with comma as a separator, as follows (see Fig. 15 which can be used instead of Fig. 3):

```
g1; g2; g3 ➔ ad(g1, g2, g3).
```

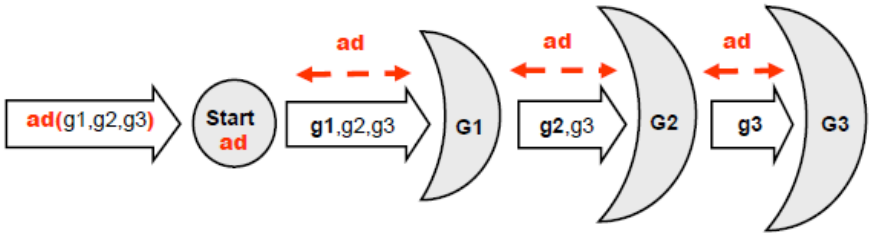

Figure 15 – Using the rule to represent a sequence of operations

3.6. The resultant unified recursive syntax of SG scenarios

The discussed above options and possibilities of structuring distributed scenarios with multiple operations allow us to obtain a clear uniformity of all possible spatial scenarios by the SG model with the use of recursion, as shown in Fig. 16, where the overall SG scenario is called a grasp, syntactic categories are in italics, vertical bar separates alternatives and the part in braces indicates zero or more repetitions using comma as a delimiter.

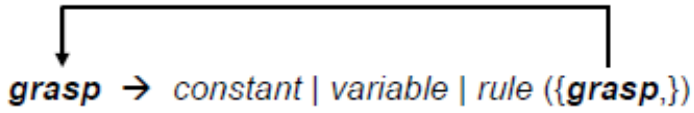

Figure 16 – Universal recursive representation of the SG model

In general, the Spatial Grasp model under the top syntax of Fig. 16 is much more diverse, complex and advanced than the one that was shown above by the restricted number of simple examples, with capability of dynamic covering and matching of any distributed spaces and returning the obtained results and control states whatever remote and multiple they might be. It also allows us to make any decisions for a further space navigation, creates dynamic operational infrastructures capable of solving any distributed problems, effectively creates, implements or simulates any other models and approaches (including Petri nets and neural nets), etc. For example, SG has different types of variables which propagate when navigating distributed spaces, which together with returning remote results and data exchanges between different locations can be explicitly mentioned in spatial charts shown above. An example of extension of the chart of Fig. 15 with such DFE (Data Flow and Exchange) capability is shown in Fig. 17.

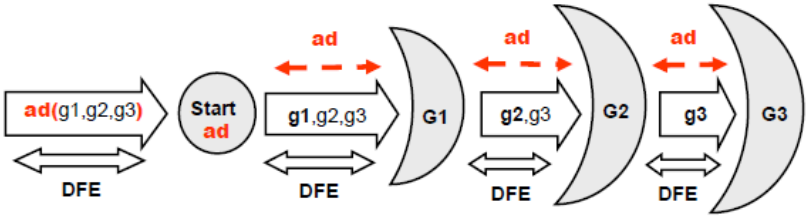

Figure 17 – Extension of a spatial chart with Data Flow and Exchange capability

For simplification and shortening, SG scenarios, depending on their implementation, may also utilize some constructions of traditional languages and use, for example, a more convenient form R=R+10 instead of `assign(R, sum(R, 10`. Moreover, semicolon is traditionally used to separate following one other constructs without embracing them with a rule. Branches can also be separated with a comma without embracing them with a rule, if they are independent from each other, with comma being superior to semicolon in the expressions, as follows:

`g1; (g2, g3, g4); (g5, g6)` ➔ `g1; g2, g3, g4; g5, g6.`

However, in the next case the use of parentheses will be absolutely important:

`(g1; g2; g3), g4, (g5; g6).`

But with such simplifications, the general organizational structure of the SG model will always follow the one shown in Fig. 16.

4. Spatial Grasp Technology (SGT) Basics

4.1. The Spatial Grasp Language

The mentioned above and many other SG model capabilities can be expressed by the recursive high level Spatial Grasp Language (SGL) in which all spatial scenarios are represented, with its top level syntax following.

grasp → *constant* | *variable* | *rule* [({ ***grasp***,})]
constant → *information* | *matter* | *custom* | *special* | ***grasp***
variable → *global* | *heritable* | *frontal* | *nodal* | *environmental*
rule → *type* | *usage* | *movement* | *creation* | *echoing* |
verification | *assignment* | *advancement* | *branching* |
transference | *exchange* | *timing* | *qualifying* | ***grasp***

Main features of SGL and its distributed networked implementation are briefed below.

4.2. The worlds SGT operates with

SGT allows us to directly operate with the following world representations: Physical World (PW), considered as continuous and infinite, where each point can be identified and accessed by physical coordinates; Virtual World (VW) which is discrete and consists of nodes and semantic links between them; and Executive world (EW) which consists of active «doers» with some communication possibilities between them. Different kinds of combination of these worlds can also be possible within the same formalism: Virtual-Physical World (VPW) where individually named VW nodes can associate with coordinates of certain PW points or any its regions; Virtual-Execution World (VEW) where doer nodes may have special names assigned to them and semantic relations in between, similarly to pure VW nodes; Execution-Physical World (EPW) can have doer nodes associated with certain PW coordinates; and Virtual-Execution-Physical World (VEPW) combining all features of the previous cases.

4.3. SGL constants

Different types of constants can be used in SGL. Information can be represented by number in its traditional syntax or by any string of characters which may also reflect a full scenario text or its part suitable for direct automatic interpretation; it can correspond to physical matter (physical objects including) as well. A repertoire of self-identifiable special and custom constants can be used as standard parameters (or modifiers) in different language rules. Constants can also be compound and use recursive grasp definition in SGL syntax which allows us to represent any nested and hierarchical structures consisting of multiple (elementary or compound) objects.

4.4. SGL variables

Spatial variables, stationary or mobile, which can be used in fully distributed physical, virtual or executive environments, effectively serve multiple cooperative processes under the unified control. These are Global variables (the most expensive) which can serve any SGL scenarios and be shared by them or by their different branches; Heritable variables appearing within a scenario step and serving all subsequent, descendent steps; Frontal variables serving and accompanying the scenario evolution, being transferred between subsequent steps; Environmental variables allowing us to access, analyze and possibly change different features of physical, virtual and executive words during their navigation; and last but not least Nodal variables as a property of the world positions reached by scenarios and shared with other scenarios in the same positions.

4.5. SGL rules

SGL rules, which are capable of representing any actions or decisions, belong to the following main categories: (a) hierarchical fusion and return of potentially remote data; (b) distributed control, sequential and/or parallel, in both breadth and depth of the scenario evolution; (c) атvariety of special contexts detailing navigation in space, clarifying character and peculiarities of the embraced operations and decisions; (d) type and sense of a value or its chosen usage for guiding automatic language interpretation; (e) individual/massive creation, modification or removal of nodes and connecting links in distributed knowledge networks, allowing us to effectively work with arbitrary knowledge structures. All rules are pursuing the same unified ideology and organizational scheme, as follows: (1) they start from a certain world position, being initially linked to it; (2) perform or control the needed operations in a distributed space which may be branching, stepwise, parallel and arbitrarily complex, also local and remote; (3) produce or supervise concluding results of the scenario embraced and expressed by control states and values in different points.

4.6. Control states

The following control states can appear after completion of different scenario steps. Indicating local progress or failure, they can be used for effective control of multiple distributed processes with proper decisions at different levels. These states are: true – reflects full success of the current scenario branch with capability of further development; done – indicates success of the current scenario step with its planned termination; fail – indicates non-revocable failure of the current branch, there is no possibility of further development from the reached location; and fatal – reports terminal failure with nonlocal effect while triggering massive abortion of all currently evolving scenario processes and removal of associated temporary data with them. These control states, appearing in different branches of parallel and distributed scenario at bottom levels, can be used to obtain generalized control states at higher levels, up to the whole scenario, in order to make proper decisions for the further scenario evolution.

4.7. Networked SGL interpreter

There may be from a million to several billions SGL interpreters which can be effectively integrated into any existing systems and communications, and their dynamic networks can represent powerful spatial engines capable of solving any problems in terrestrial and celestial environments. Such collective engines can simultaneously execute many cooperative or competitive tasks without any central resources or control, as symbolically depicted in Fig. 18 (SGL interpreters are marked as U, as universal computational and management nodes).

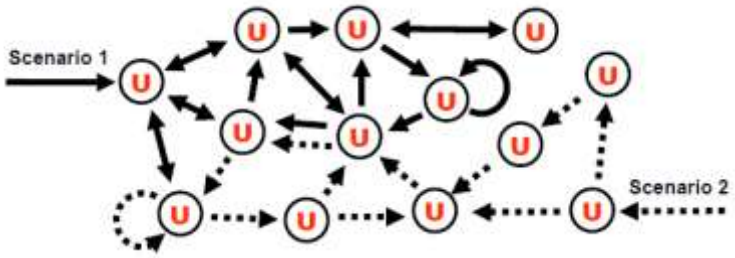

Figure 18 – SGL interpretation networks as a global world computer

As both backbone and nerve system of the distributed interpreter, its hierarchical spatial track system dynamically spans in the worlds where SGL scenarios evolve and provide automatic control of multiple distributed processes. Self-optimizing in parallel echo processes, this distributed structure (which is generally a forest-like one) provides hierarchical command and control, as well as remote data and code access. It supports spatial variables and merges distributed control states for making proper decisions at different organizational levels. The track infrastructure is automatically distributed between different active components (humans, robots, computers, smartphones, satellites, etc.) during scenario spreading in distributed environments.

Detailed information on SGT, SGL and its networked interpreter, as well as solutions to numerous problems from different classes under such an approach, can be obtained from many existing publications, including [7–13].

5. Examples of Spatial Scenarios in SGL

Some solutions to practical problems from different areas, which are entirely based on the Spatial Grasp model described in this paper, as well explanation of its advantages are shown below.

5.1. Network management

Below there is presented the process of finding a path between nodes `a` and `f` in a network of Fig. 12.

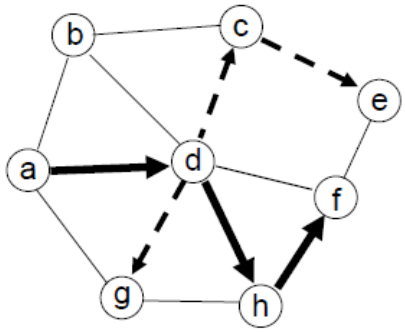

Figure 19 – Finding a path between two network nodes

```
hop(a); repeat(hopfirst(links(all));
               if(NAME == f, done)).
```

We may decide to collect the passed path and organize its output at the destination node, as follows:

```
hop(a); frontal(Path = NAME);
repeat(hopfirst(link(any)); Path &&= NAME;
       if(NAME == f, (output(Path); done))).
```

Arbitrary path found between these two nodes (may not be optimal like the one in Fig. 12) is printed in node `f`, like `(a, d, h, f)`. This path can also be returned and issued in the starting node `a` as:

```
hop(a); frontal(Path = Name);
output(repeat(hopfirst(link(any)); Path &&= NAME;
              if(NAME == f, done(Path)))).
```

SG solution analysis

The described solutions on a distributed network are entirely based on its spatial navigation by SGL self-evolving scenarios with finding a path between the necessary nodes, as well as collecting and printing this path in its final or starting nodes. Such integral parallel and fully distributed solutions are much superior to traditional methods of representing distributed computations in the form of multiple parts or agents that exchange messages.

More information about the network management under SGT can be found in [7, 8, 12, 14].

5.2. Human-robotic collectives

Any collectives from human, robotic or mixed units operating under spatial scenarios can be easily organized, as symbolically shown in Fig. 13.

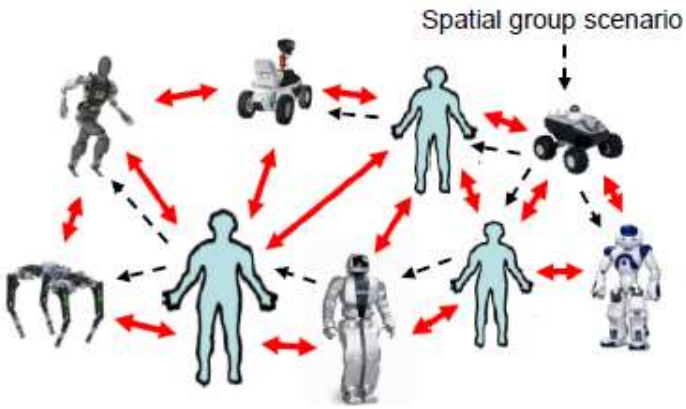

Figure 20 – Unified human-robotic teams with collective scenario injected from any unit

Randomized collective group movement, starting in any node, with minimal `Range` distance, allowed between units during movement, can be organized as follows, where each unit after discovering some dangerous objects or situations issues a corresponding `alarm` message or sound.

```
hop(any_unit);
repeat(
  hop_first(all_neighbors);
  free(nodal(Limits = (dx(0,8), dy(-2,5)), Range = 100, Shift);
       repeat(Shift = random(Limits);
              if(empty(Shift, Range), WHERE += Shift);
              if(analyze(Seen), output(alarm));
              sleep(delay))))
```

SG solution analysis

This scenario starts from any human or robotic unit and then covers all reachable units by flooding them in parallel, regardless of their number which may be arbitrarily large and unknown in advance. It immediately tasks each reached unit with the collective movement and observation functionality, without waiting for full completion of the scenario distribution. This holistic self-evolving spatial solution is also superior to traditional parallel computations representing the system as a collection of agents that exchange messages.

More information about collective behavior under SGT can be found in [7–10].

5.3. Space debris collection

Dealing with such complex problem as huge amount of space debris can be possible only by using large constellations of special cleaning (like de-orbiting) satellites working together, see Fig. 21. The following scenario is launched from a ground station `G2` and enters any currently reachable satellite, after which moves to the whole constellation by direct inter-satellite links in an attempt to find a suitable cleaner-satellite for the removal of junk initially given by its parameters detected by `G2`.

```
hop(G2);
frontal(Details) = find_select(radar, junk, TIME);
hop_first(any_cleaner, radar);
repeat(
   Snapshot = parameters(closest_junk, seen);
   if(match(Details, Snapshot),
      (deorbit(Snapshot); abort));
   update(Details, TIME);
   hop_first(all_cleaners, direct_links))
```

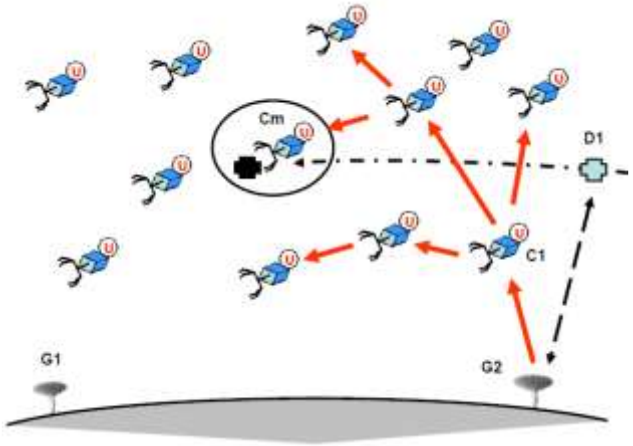

Figure 21 – Debris removal by self-organized network of cleaning satellites

SG solution analysis

This solution, where satellites move at high speed and possibly in different orbits or even directions, is highly dynamic and fully distributed. The self-spreading holistic scenario can operate with any constellation topologies which may rapidly change in time, as well as with any number of currently available satellites which may not be known in advance. This SG solution is also superior to traditional representation of a distributed system in the form of collection of agents exchanging messages where such a system in this dynamic multiple satellite case cannot be defined in advance at all.

More information about debris and management of multiple satellite architectures can be found in [15–21].

5.4. Spreading and fighting viruses

Below there is shown a sketch in SGL of how to model massive spread of Covid-like virus, distribution and use of antivirus vaccine with its influence on further virus development. In Fig. 22 the virus is assumed originating from a source `S` and initial vaccination starting from locations `V1` and `V2`, with both processes spreading worldwide in parallel.

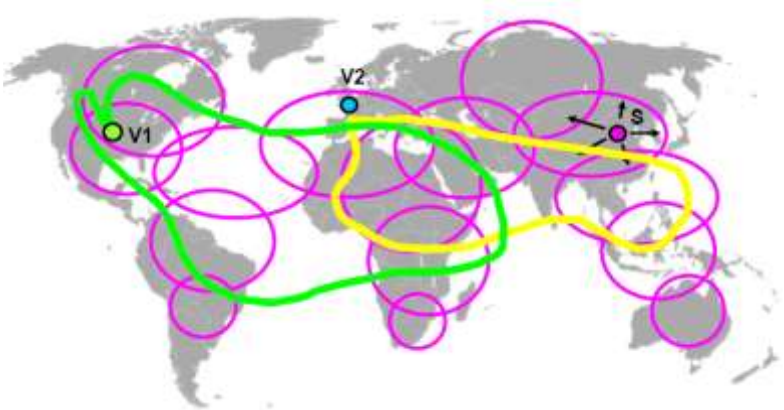

Figure 22 – Spreading and fighting viruses worldwide

The following scenario describes unlimited virus spread in a random mode with certain `Breadth` from any reached point. With this spread being chaotic and capable of returning to previous areas along with covering new regions, while being halted in the already visited points or where the vaccination took place.

```
move(S); nodal(Status);
frontal(Breadth = number1, Limits = (Xmin1_Xmax1, Ymin1_Ymax1));
repeat(
  parallel(replicate(Breadth, shift(random(Limits))));
  if(Status == not(protected), Status = infected))
```

The next scenario describes the world coverage by vaccination which distributes in a more ordered way by extending the vaccinated space, in contrast to the chaotic manner of the virus.

```
move(V1, V2); nodal(Status, Distance, Shift););
frontal(Start = WHERE, Faraway = 0, Breadth = number2,
        Limits = (Xmin2_Xmax2, Ymin2_Ymax2));
repeat(parallel(replicate(Breadth,
  (Shift = random(Limits);
   Distance = distance(Start, WHERE + Shift);
   Distance > Faraway; Faraway = Distance; shift(Shift);
   if(Status == not(protected), Status == protected)))));
```

SG solution analysis

These are highly dynamic and fully distributed solutions where the system in which parallel and concurrent virus and vaccination processes take place is outlined and formed dynamically in SGL, where fully interpreted SG model develops itself as an almighty virus which can do every-

thing in fully distributed environments. Such processes may also relate to global warming, weather crises, dynamics of market economies, military conflicts, cultural and religious clashes, spreading different cultures and beliefs, etc.

More information about virus and antivirus vaccine simulation and management, as well as other global self-evolving processes, can be found in [7, 8, 9, 22].

6. Conclusions

This section concludes the paper by summarizing and generalizing the experience obtained during many years of dealing with distributed networked systems and their applications in various areas, as well as the development of distributed control technology with its Spatial Grasp Language. This generalization has been made in the form of radically new computational and control model as an extension of traditional concepts of algorithm and its flowcharts, allowing direct operation of systems in combined terrestrial and celestial environments. In the article there was introduced a new concept of flowcharts called spatial charts which help to exhibit, analyze and develop complex solutions for distributed systems operating on Earth and in outer space. Instead of representing distributed systems as a collection of parts or agents exchanging messages, there was presented an integral, holistic and semantic level solutions as self-evolving and self-matching spatial patterns on a high semantic level which can often simplify and shorten global management code up to a hundred times in comparison with other approaches and languages. Among the activities planned in this area, there are new SGT implementation which can be done even within traditional university environments as for the previous technology versions, new patenting of the SG model as a further development of the previous patent [13] and a new book about the utilization of SG model and the resultant technology for the management of integrated terrestrial and celestial systems. The mentioned book is currently in preparation under the contract with reputable publishers, with its draft contents presented in [23] and the latest chapters-related journal papers already appearing [15, 16, 21], including the current paper.

REFERENCES

1. Algorithm. URL: https://en.wikipedia.org/wiki/Algorithm.
2. Definition of ALGORITHM. Merriam-Webster Online Dictionary. URL: https://www.merriam-webster.com/dictionary/algorithm.
3. Goodrich M.T., Tamassia R. Algorithm Design: Foundations, Analysis, and Internet Examples. New York: John Wiley & Sons, Inc., 2002. URL: https://web.archive.org/web/20150428201622/http://ww3.algorithmdesign.net/ch00-front.html.
4. Heuristic. URL: https://en.wikipedia.org/wiki/Heuristic.
5. Flowchart. URL: https://en.wikipedia.org/wiki/Flowchart.
6. Lynch A. Flow Chart Design – How to design a good flowchart, 04/22/2021. URL: https://www.edrawsoft.com/flowchartdesign.html?gclid=Cj0KCQjw5auGBhDEARIsAFyNm9H7cWvesxjTsposJTaD890zBspEUDA18dHi_9R_GChiyzitwgpYCu4aAjutEALw_wcB.
7. Sapaty P.S. Symbiosis of Real and Simulated Worlds under Spatial Grasp Technology. Springer, 2021. 251 p.
8. Sapaty P.S. Complexity in International Security: A Holistic Spatial Approach, Emerald Publishing, 2019. 160 p.
9. Sapaty P.S. Holistic Analysis and Management of Distributed Social Systems, Springer, 2018. 234 p.
10. Sapaty P.S. Managing Distributed Dynamic Systems with Spatial Grasp Technology, Springer, 2017. 284 p.
11. Sapaty P.S. Ruling Distributed Dynamic Worlds. New York: John Wiley & Sons, 2005. 255 p.
12. Sapaty P.S. Mobile Processing in Distributed and Open Environments. New York: John Wiley & Sons, 1999. 410 p.
13. Sapaty P.S. A distributed processing system, European Patent N 0389655, Publ. 10.11.93, European Patent Office. 35 p.

14. Sapaty P.S. Global Network Management under Spatial Grasp Paradigm, *Global Journal of Researches in Engineering: J General Engineering.* 2013. Vol. 20, Issue 5, Version 1.0 Year 20. P. 58–69. URL: https://engineeringresearch.org/index.php/GJRE/article/view/2082/2013.
15. Sapaty P.S. Spatial Management of Large Constellations of Small Satellites. *Mathematical machines and systems*. 2021. N 2. P. 3–14.
16. Sapaty P.S. Global Management of Space Debris Removal Under Spatial Grasp Technology. *Acta Scientific COMPUTER SCIENCES.* 2021. Vol. 3, Issue 7. URL: https://www.actascientific.com/ASCS/pdf/ASCS-03-0135.pdf.
17. Space Debris. URL: https://en.wikipedia.org/wiki/Space_debris
18. Space Debris, NASA Headquarters Library. URL: https://www.nasa.gov/centers/hq/library/find/bibliographies/space_debris.
19. Deorbit Systems, National Aeronautics and Space Administration. 2020. Nov. 28. URL: https://www.nasa.gov/smallsat-institute/sst-soa-2020/passive-deorbit-systems.
20. Chen Y. et al. Optimal mission planning of active space debris removal based on genetic algorithm, IOP Conf. Series: Materials Science and Engineering. 2020. N 715, P. 012025. URL: https://iopscience.iop.org/article/10.1088/1757-899X/715/1/012025/pdf.
21. Sapaty P.S. Managing multiple satellite architectures by spatial grasp technology. *Mathematical machines and systems*. 2021. N 1. P. 3–16. http://www.immsp.kiev.ua/publications/eng/2021_1/.
22. Sapaty P.S. Fighting global viruses under spatial grasp technology. *Transactions on Engineering and Computer Science. 2020.* Vol. 1 Issue 2. URL: https://gnoscience.com/uploads/journals/articles/118001716716.pdf.
23. Sapaty P.S. Spatial Grasp as a Model for Space-based Control and Management Systems. *Mathematical machines and systems*. 2021. N 1. P. 135–138. URL: http://www.immsp.kiev.ua/publications/articles/2021/2021_1/Sapaty_book_1_2021.pdf.

Стаття надійшла до редакції 18.06.2021