MEETING THE WORLD CHALLENGES
From Philosophy to Information Technology to Applications

Peter Simon Sapaty
Institute of Mathematical Machines and Systems, National Academy of Sciences
Glushkova Ave 42, 03187 Kiev, Ukraine
sapaty@immsp.kiev.ua

Keywords: World crises, Atomism, holism, Gestalt theory, System integrity, Waves, Distributed scenario language, Networked interpretation, Unmanned systems, Task level, Behavioral level, Crisis management, Electronic warfare, Directed energy systems.

Abstract: We have been witnessing numerous world crises and disasters—from ecological to military to economic, with global world dynamics likely to be increasing this century further. The paper highlights known holistic and gestalt principles mainly used for a single brain, extending them to any distributed systems which may need high integrity and performance in reaction to unpredictable situations. A higher organizational layer is proposed enabling any distributed resources and systems to behave as an organism having global "consciousness" and pursuing global goals. This "over-operability" layer is established by implanting into key system points the same copy of a universal intelligent module, which can communicate with other such modules and interpret collectively global mission scenarios presented in a special Distributed Scenario Language. The scenarios can be injected from any module, and then self-replicate, self-modify, and self-spread throughout the system to be managed, tasking components, activating distributed resources, and establishing runtime infrastructures supporting system's integrity. Numerous existing and prospective applications are outlined and discussed, confirming paradigm's usefulness for solving hot world problems.

1 INTRODUCTION

To understand mental state of a handicapped person, problems of economy and ecology, or how to win on a battlefield, we must consider the system as a whole -- not just as a collection and interaction of parts. The situation may complicate dramatically if the system is dynamic and open, spreads over large territories, comprises unsafe or varying components, and cannot be observed in its entirety from a single point. Numerous world crises we have been witnessing at the beginning of this century, including the current economic one, may have emerged, first of all, due to our inability of seeing and managing complex systems as a whole.

To withstand the unwanted events and their consequences (ideally: predict and prevent them) we need effective worldwide integration of numerous efforts and often dissimilar and scattered resources and systems. Just establishing advanced communications between parts of the distributed systems and providing the possibility of sharing local and global information from any point, often called "interoperability", is becoming insufficient (even insecure and harmful) for solving urgent problems in dynamic environments, in real time and ahead of it.

We may need the whole distributed system to behave as an integral organism, with parts not so interoperating but rather complementing each other and representing altogether an integral whole pursuing global goals and having a sort of global awareness and consciousness. This whole should be essentially more than the sum of its parts, with the latter having sense, possibly even existence, in the context of this whole, rather than vice versa.

This paper develops further the over-operability principle researched in Sapaty, 1993, 1999, 2002, 2005 and other works (the term "over-operability" coined in Sapaty, 2002), which can establish intelligent dominant layer over distributed resources and systems, and help solve urgent world problems in a parallel, distributed, and dynamic way.

The rest of this paper compares the dominant atomistic approach in system design, implementation and management with holistic and gestalt principles, and describes a novel ideology and technology for integral solutions in distributed

worlds, which can avoid many traditional management routines in solving global problems, with its numerous practical applications outlined and discussed.

2 ATOMISM, HOLISM, GESTALT

We used to exercise predominantly atomistic, parts-to-whole philosophy of the system design, comprehension and implementation, which extends even to the organization of management facilities themselves -- as a collection of interacting parts, or *agents*. (This philosophy actually being the same as a century ago.)

Originally a system or campaign idea and the functionality needed emerge in a very general form (in a single human mind or in a close collective of such minds). Then this general idea (shown symbolically in Fig. 1*a*) is partitioned into individual chunks, or "atoms", each detailed and studied further (Fig. 1*b*). This logical partitioning already causes swelling of the problem complexity (as indicated in Fig. 1*b*).

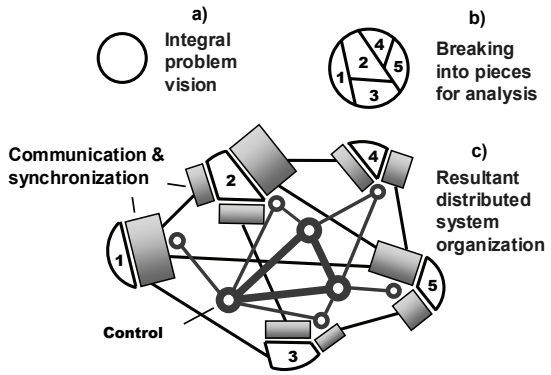


Figure 1: System overhead under atomistic organization.

The next step is materialization of the defined parts and their distribution in physical or virtual space. To make these parts work or behave together within the original idea of Fig. 1*a*, we may need a good deal of their communication and synchronization, also sophisticated control infrastructures, as depicted in Fig. 1*c*. This overhead may be considerable, outweighing and shadowing the original project definition.

The main problem is that the initial idea (Fig. 1*a*) and even its second stage (Fig. 1*b*) are usually non formalized, remaining in the minds of creators only, and the *real system description and implementation start from the already partitioned-interlinked stage, with its huge overhead* (as Fig. 1*c*).

This parts-to-whole approach also dominates in the controversial "society of mind" theory (Minsky, 1988), which is trying to explain even human thinking from the atomistic positions.

Holism (see, for example, Smuts, 2007) has quite an opposite vision of systems:

- Holism as an idea or philosophical concept is diametrically opposed to atomism.
- Where the atomist believes that any whole can be broken down or analyzed into its separate parts and the relationships between them, the holist maintains that the whole is primary and often greater than the sum of its parts.
- The atomist divides things up in order to know them better; the holist looks at things or systems in aggregate.

Gestalt theory (Koffka, 1913; Wertheimer, 1922) is based on the holistic principles too:

- For the gestaltists, "Gestalten" are not the sums of aggregated contents erected subjectively upon primarily given pieces.
- Instead, we are dealing with wholes and whole–processes possessed of inner intrinsic laws.
- *Elements* are determined as parts by the intrinsic conditions of their wholes and are to be understood *as parts* relative to such wholes."

Although gestalt psychology and theory was a general approach, most of the work on gestalt was done in the area of perception. *In our research, we are trying to use the holistic and gestalt principles for the organization of distributed systems with highest possible integrity and performance* (see Sapaty, 2009).

3 WAVES, FIELDS, SCENARIOS

We describe here a novel organizational philosophy and model, based on the idea of spreading *interdependent parallel waves* (as shown in Fig. 2), as an alternative to the dominant atomistic approach briefed above, also under the influence of mentioned holistic and gestalt ideas.

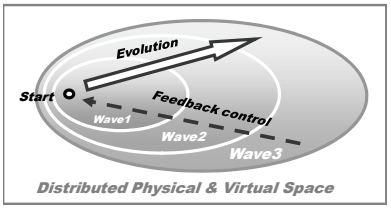


Figure 2: Grasping the entirety with spatial waves.

It allows us for an integral, parallel, and seamless navigation and coverage of virtual, physical or combined spaces where the solutions need to be found. Atomism emerges on the automatic implementation level only, which allows us to get high-level formal semantic definitions of systems and global operations in them, while omitting numerous organizational details (shown in Fig. 1*c*) and concentrating on global goals and overall performance instead.

An automatic materialization of this approach is carried out by the network of universal intelligent modules (U), embedded into important system points, which collectively interpret integral mission scenarios expressed in the waves formalism, which can start from any U, subsequently covering the distributed system at runtime.

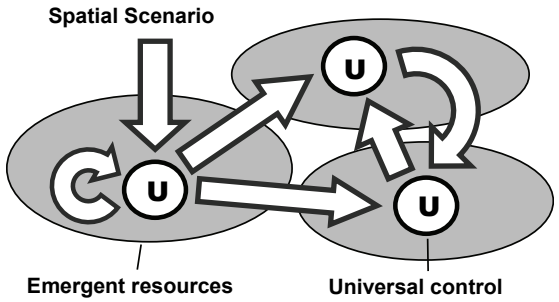


Figure 3: Self-spreading mission scenarios.

The wavelike scenarios are usually very compact and can be created and modified on the fly. They can cooperate or compete with each other in the distributed networked space as overlapping fields of parallel solutions.

Spreading waves can create knowledge infrastructures arbitrarily distributed between system components (robots, sensors, humans). These, subsequently or simultaneously navigated by same or other waves, can effectively support distributed databases, command and control, situation awareness, and autonomous decisions.

This paradigm is much in line with the existing abundant evidence that certain aspects of cognition, morals, needs, object relations, motor skills, and language acquisition proceed in developmental stages. These stages appear to be fluid, flowing, overlapping waves (Wilber, 2009), where also:

- Each stage has a holistic pattern that blends all of its elements into a structured whole;
- These patterns unfold in a relational sequence, with each senior wave transcending but including its juniors.

Our approach is also consistent with the ideas of self-actualization and person-centered approach (Rogers, 1978; Kriz, 2008), where the self is considered as an organized, consistent, conceptual gestalt exhibiting active forward thrust -- against tension reduction, equilibrium, or homeostasis (as in Freud, 2007, and others). In our case, instead of a single person we have the whole distributed system with high integrity and "active global thrust" behavior.

4 THE SCENARIO LANGUAGE

Distributed Scenario Language, or DSL (and its previous versions, WAVE including, as in Sapaty, 1999, 2005) reflects the waves model proposed, and allows us to directly express semantics of problems to be solved in distributed worlds, also the needed global system behavior in a non-atomistic manner. DSL operates with:

- *Virtual World (VW)*, which is discrete and consists of nodes and links connecting these nodes.
- Continuous *Physical World (PW)*, any point in which may be accessed by physical coordinates (taking into account certain precision).
- *Virtual-Physical World (VPW)*, which is an extension of VW where nodes additionally associate with certain coordinates in PW.

It also has the following key features:

- A DSL scenario develops as a transition between sets of progress points (or *props*) in the form of parallel *waves*.
- Starting from a prop, an action may result in one or more props (the resultant set of props may include the starting prop too).
- Each prop has a resulting *value* (which can be multiple) and a resulting *state* (being one of the four: *thru*, *done*, *fail*, and *abort*).
- Different actions may evolve independently or interdependently from the *same* prop, contributing to (and forming altogether) the resultant set of props.
- Actions may also *spatially succeed each other*, with new ones applied in parallel from all the props reached by preceding actions.
- Elementary operations can directly use local or remote values of props obtained from other actions (or even from the whole scenarios).
- Elementary operations can result either in open values that can be directly used as *operands* by other operations in an expression, or by the *next*

operations in a sequence. They can also be directly assigned to *local or remote variables* (for the latter case, an access to these variables may invoke scenarios of any complexity).

- Any prop can associate with a *node* in VW or a *position* in PW, or *both* -- when dealing with VPW.
- Any number of props can be simultaneously linked with the same points of the worlds.
- Staying with world points (virtual, physical, or combined) it is possible to *directly access* and update local data in them.
- Moving in physical, virtual or combined worlds, with their possible modification or even creation from scratch, are as routine operations as, say, arithmetic or logical operations of traditional programming languages.
- DSL can also be used as a usual universal programming language (like C, Java, or FORTRAN).

DSL has a recursive syntax, which on top level is as follows:

wave → ***phenomenon*** | ***rule*** **(** { ***wave*** **,** **}**)
phenomenon → ***constant*** | ***variable*** | ***special***
constant → ***information*** | ***matter*** | ***combined***
variable → ***heritable*** | ***frontal*** | ***environmental*** | ***nodal***
rule → ***movement*** | ***creation*** | ***elimination*** | ***echoing*** | ***fusion*** | ***verification*** | ***assignment*** | ***advancing*** | ***branching*** | ***transference*** | ***timing*** | ***granting***

Elementary programming examples in DSL are shown in Fig. 4 for: a) assignment of a sum of values to a variable; b) parallel movement into two physical locations; c) creation of a node in a virtual space, and d) extension of the latter with a new link and node.

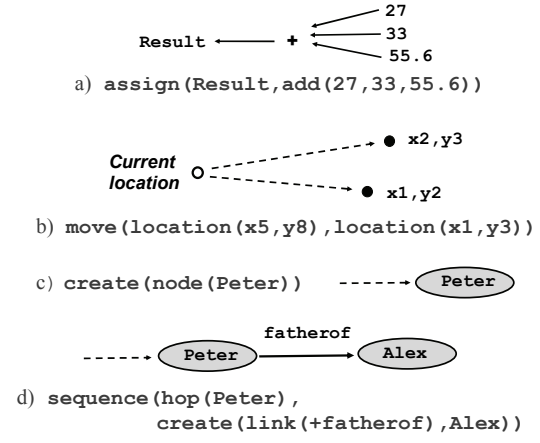


Figure 4: Elementary examples in DSL.

Traditional abbreviations of operations and delimiters can also be used, as in many further examples throughout this text, to simplify and shorten DSL programs, remaining however within the general recursive syntactic structure shown above.

5 COMPOSITION OF WAVES

The language allows for an integral parallel navigation of distributed worlds in a controlled *depth and breadth mode*, with any combinations of the two. We will highlight here key possibilities of doing this by composition of DSL scenarios, or waves.

5.1 Single Wave Features

Single wave (let it be *W1*) development features are shown in Fig. 5. Starting from a prop, which may be associated with a point in the world, the related scenario evolves, grasps, and covers certain region in it, performing any operations needed in the distributed space.

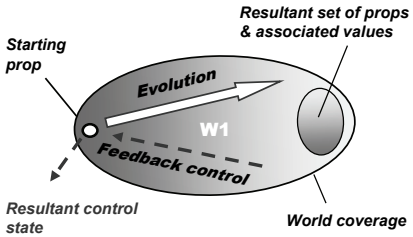


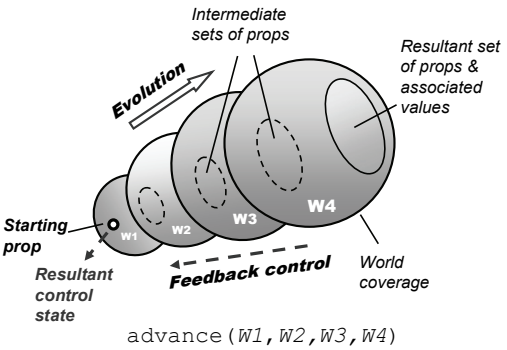
Figure 5: Single wave features.

The result of this spatial evolution may be multiple, and may lie in a (final) sub-region of the region covered, being represented by a set of resultant props (each linked to world points) and associated with them values. After termination of the wave, its resultant control state (which, in a parallel feedback process, merges termination states throughout the region covered) is available in the starting prop, and may be taken into account for decisions at higher levels. Also, if requested from higher levels, the values associated with the resultant props (which may be remote) can be lifted, spatially raked, and returned to the starting prop for a further processing.

5.2 Advancing in Space

The depth mode development of waves is shown in Fig. 6. For this type of composition, each subsequent

wave is applied in parallel from all props in space reached by the previous wave, with the resultant set of props (and associated values) on the whole group being the one of the last applied wave (i.e. *W4* in the figure).



`advance(W1,W2,W3,W4)`

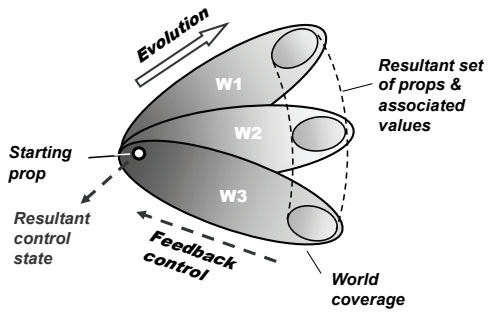
Figure 6: Depth mode composition of waves.

This spatial advancement of waves returns the resultant control state which is available at the starting prop, and the values of the resultant set of props can also be echoed to the starting prop if requested. Examples of other advancing rules:

- `advance synchronized` – the one where any new wave is applied only after all invocations of the previous wave have been terminated;
- `repeat` – where the same wave is applied repeatedly from all props reached by its previous invocation;
- `repeat synchronized` – where in the repeated invocation of a wave each new invocation starts only after full completion of the previous one.

5.3 Branching in Space

The branching breadth mode composition of waves is shown in Fig. 7, where all waves in the group are evolving from the same starting prop, and each wave, with its own resultant set of props and associated values, contributes to the final result.



`sequence(W1,W2,W3)`

Figure 7: Breadth mode composition of waves.

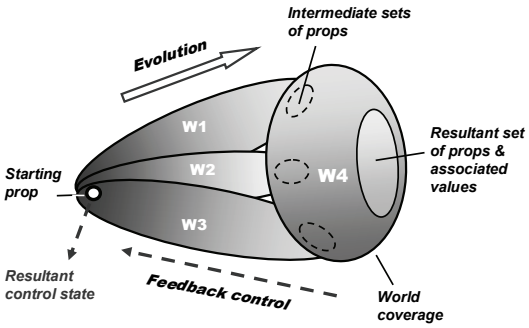
The merge of results from different waves depends on the branching rule used, with their repertoire (besides the `sequence` in Fig. 7) including:

`if, while, parallel, or, parallel or, and, parallel and, cycle, loop, and sling.`

(More details on these and other rules can be found, say, from Sapaty, 1999, 2005.)

5.4 Combined Branching-Advancing

Any combination of advancing and branching modes in a distributed space can be expressed and implemented in DSL (as shown in Fig. 8).



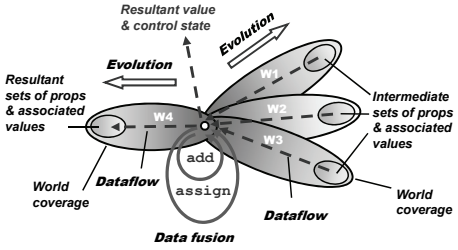
`advance(sequence(W1,W2,W3),W4)`

Figure 8: Breadth–depth composition mode.

These combinations, when embraced by the existing variety of composition rules, can provide any imaginable and even so far unimaginable spatial algorithms that can solve distributed problems in highly integral and compact ways, without explicit descending to the traditional atomistic level shifted to the automatic implementation only.

5.5 Operations on Remote Values

Due to fully recursive organization of DSL, it is possible to program in it arbitrary complex expressions directly operating not only on local but also arbitrarily remote values, where any programs (scenarios) can happen to be operands of any operations (expressed by rules). This gives an enormous expressive power and compactness to complex spatial scenarios evolving in distributed environments. An example of such compact expression of spatial operations on remote values and variables is shown in Fig. 9.




```
assign(W4,add(W1,W2,W3))
```

Figure 9: Direct operations on remote values.

6 DISTRIBUTED INTERPRETER

DSL interpreter, as from the previous language version called WAVE (Sapaty, 1993, 1999, 2005), has been prototyped in different countries on various platforms. Its public domain version (financed in the past by Siemens/Nixdorf) is being used for applications like intelligent network management or simulation of distributed dynamic systems. The DSL interpreter basics include:

- It consists of a *number of specialized modules* working in parallel and handling and sharing specific data structures, which are supporting persistent virtual worlds and temporary hierarchical control mechanisms.
- The whole network of the interpreters can be *mobile and open*, changing at runtime the number of nodes and communication structure between them.
- The heart of the distributed interpreter is its *spatial track system* enabling hierarchical command and control and remote data and code access, with high integrity of emerging parallel and distributed solutions.

The DSL interpreter structure is shown in Fig. 10.

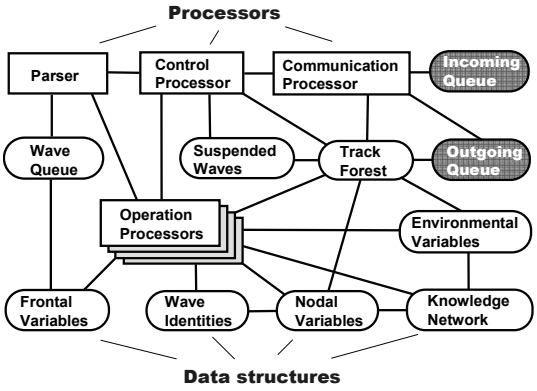


Figure 10: Structure of DSL interpreter.

It can be easily implemented in both software and hardware on any platforms, where the intelligent "wave chip" can be implanted into a great variety of devices, making them working together as an integral unit under the spatial DSL scenarios.

7 PROGRAMMING EXAMPLES

We will show here examples of solution in DSL of some important problems on networks and graphs in a fully distributed way, where each node may reside in a separate computer.

7.1 Shortest Paths

The solution for finding a path between two nodes by navigating the network with parallel waves is sown in Fig. 11, and the scenario that follows.

```
sequence(
 (direct # a; Ndist = 0; repeat(
  any #; Fdist += LINK;
  Ndist == nil, Ndist > Fdist;
  Ndist = Fdist; Npred = BACK))
 (direct # e; repeat(
  Fpath &= CONT; any # Npred);
  USER = Fpath))
```

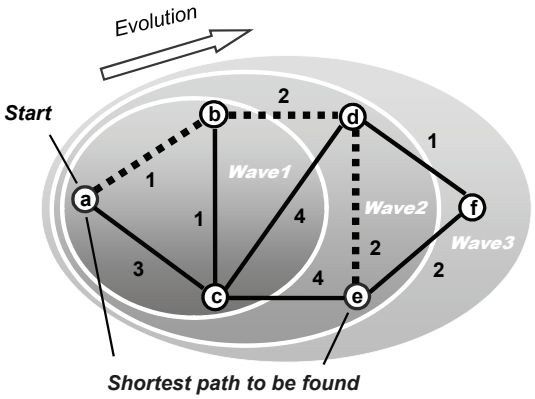


Figure 11: Finding shortest path with waves.

Many problems of optimization and control may be expressed as finding shortest paths in a distributed solution space.

7.2 Spatial Topology Analysis

DSL allows us to directly analyze and process distributed topologies in a parallel and extremely concise way.

7.2.1 Articulation Points

To find the weakest nodes in a network (called *articulation points*) which, when removed, split it into disjoint parts, as in Fig. 12 for node d, we need only the program that follows.

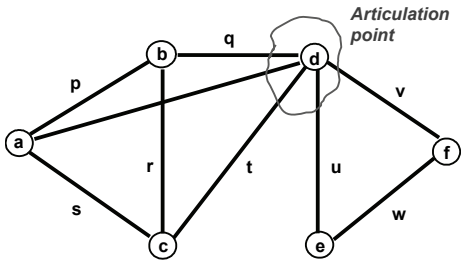


Figure 12: Articulation points.

```
direct # all; ID = CONT; Nm = 1;
and((random(all #);
 repeat(Nm ==; Nm = 1; all #)),
 (all #; Nm ==), USER = CONT)
```

Result: d.

7.2.2 Cliques

Cliques (or fully connected sub-graphs of a graph, as in Fig. 13), on the opposite, may be considered as strongest parts of a system. They can be found in parallel by the program that follows.

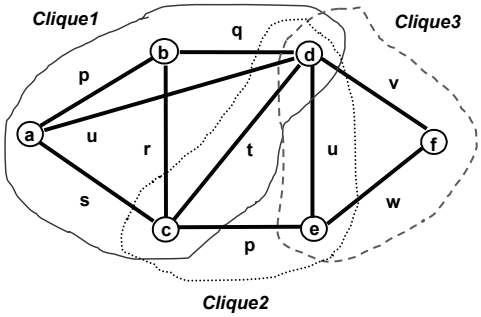


Figure 13: Cliques.

```
direct # all; Fclique = CONT;
repeat(all #; CONT !~ Fclique;
 and(andpar(any # Fclique; done !),
  or((BACK > CONT; done !),
   Fclique &= CONT))); USER = Fclique
```

Result: (a,b,c,d), (c,d,e), (d,e,f)

7.2.3 All Triangles

Any topological patterns can be found in any distributed network. For example, finding all triangles in a graph in Fig. 13 needs a simple code:

```
direct # all; Ftr = CONT;
2(all#; BACK > CONT; Ftr &= CONT);
 any # Ftr : 1; USER = Ftr
```

Result: (a,b,c), (b,c,d), (c,d,e), (d,e,f), (a,b,d), (a,c,d)

7.2.4 Network Creation

Any network can be created in a distributed space, and in parallel mode, by a very simple code too, as follows, as for the network in Fig. 13.

```
create(direct#a; p#b; q#d; u##a,(v#f;
w#e; u##d,(p#c; s##a, r##b, t##d)))
```

Arbitrary infrastructures can be created at runtime, on the fly, which can become active by putting certain procedures into their nodes and links. Any other existing models (incl. Petri nets, neural nets, contract nets, etc.) can also be implemented in a fully distributed and parallel way in DSL. Many related examples can be found in Sapaty, 1999.

8 COLLECTIVE ROBOTICS

Installing DSL interpreter into mobile robots (ground, aerial, or underwater) may allow us to organize any group solutions of complex problems in distributed physical spaces in a concise and effective way, shifting traditional management routines to automatic level. It is possible to express tasks and behaviors on different levels, as follows.

8.1 Task Level

Heterogeneous groups of mobile robots (as in Fig. 14) can be tasked at a highest possible level, just telling what they should do together, without detailing how, and what are the duties of every unit. An example task may be formulated as follows.

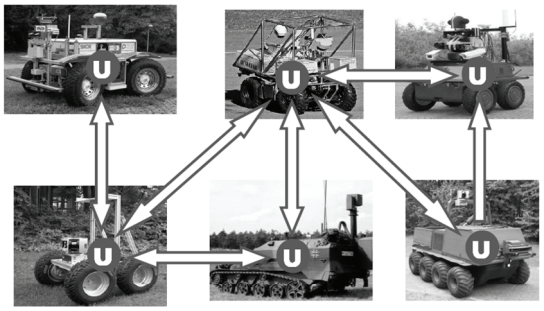


Figure 14: Grouping ground vehicles.

Go to physical locations of the disaster zone with coordinates (50.433, 30.633), (50.417, 30.490), and (50.467, 30.517). Evaluate damage in each location, find and transmit the maximum destruction value, together with exact coordinates of the corresponding location, to a management center.

The DSL program will be as follows:

```
transmit(maximum(
  move((50.433, 30.633),
       (50.417, 30.490),
       (50.467, 30.517));
  evaluate(destruction)& WHERE))
```

Details of automatic implementation of this scenario by different numbers of mobile robots are discussed in (Sapaty, 2009c).

8.2 Behavioral Level

After embedding DSL interpreters into robotic vehicles (like the aerial ones in Fig. 15), we can also provide any needed detailed collective behavior of them (at a lower than top task level, as before)—from *loose swarms* to a *strictly controlled integral unit* obeying external orders. Any mixture of different behaviors within the same scenario can be easily programmed too.

The following DSL scenario combines loose, random swarm movement in a distributed space with periodic finding/updating topologically central unit, and setting runtime hierarchical infrastructure between the units. The latter controls observation of distributed territory, collecting potential targets, distributing them between the vehicles, and then impacting potential targets by them individually. More on the implementation of this scenario can be found in Sapaty, 2008.

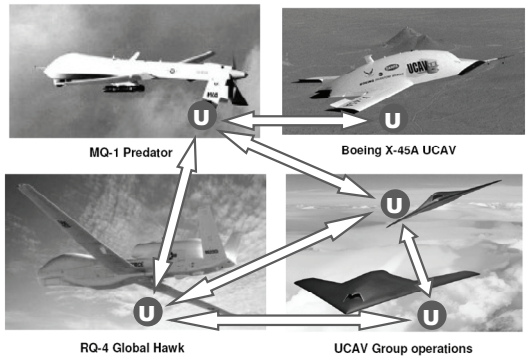


Figure 15: Grouping aerial vehicles.

```
(hop(allnodes); Range = 500;
 Limits = (dx(0,8), dy(-2,5));
 repeat(Shift = random(Limits);
  if(empty(hop(Shift, Range),
     move(Shift)))),
(repeat(hop(
  Faver =average(hop(allnodes);WHERE);
  min(hop(allnodes);
  distance(Aver, WHERE)& ADDRESS):2));
  stay(hop(nodes,all);rem(links,all);
  Frange = 20; repeat(
   linkup(+infra, firstcome, Frange));
  orpar(
   loop(nonempty(Fseen =
    repeat(free(detect(targets)),
    hoplinks(+ infra));
    repeat(
     free(select_move_shoot(Fseen),
     hoplinks(+ infra))),
   sleep(360)))
```

9 OTHER APPLICATIONS

9.1 Distributed Avionics

Distributed communicating DSL Interpreters, embedded into aircraft's key mechanisms (as in Fig. 16), can provide highest possible integrity of the aircraft that may continue to *function as a whole* even under *physical disintegration* -- which may help find critical runtime solutions saving lives and equipment (see also Sapaty, 2008a).

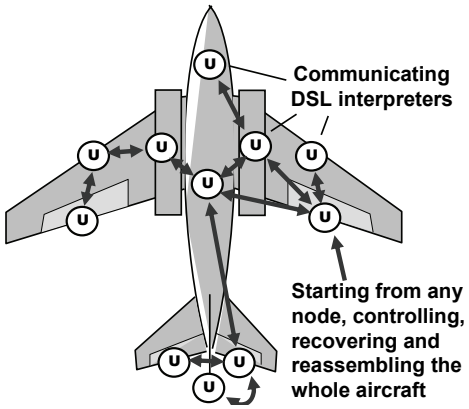


Figure 16: Distributed control infrastructure.

Collecting availability of aircraft's basic mechanisms, and establishing overall aircraft control from any available DSL interpreter, may be organized as follows:

```
Available =
 repeat(free(belong(CONT,
  (left_aileron, right_aileron,
   left_elevator,right_elevator,
   rudder, left_engine,right_engine,
   left_chassis, right_chassis, …));
```

```
   CONT), hop(firstcome, neighbors));
if(sufficient(Available),
   control(Available), set(alarm))
```

9.2 Objects Tracking

In a large distributed space, each embedded (or moving) sensor can handle only a limited part of space, so to keep the whole observation continuous, the mobile object seen should be handed over between neighboring sensors during its movement, along with the data accumulated on it (see also Sapaty, 1999, 2007, 2008).

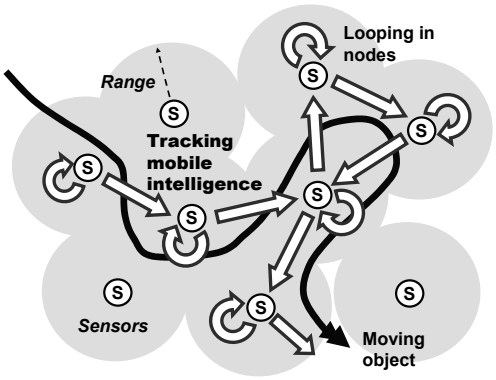


Figure 17: Tracking mobile objects.

The following program, starting in all sensors, *catches the object* it sees and *follows* it wherever it goes, if not observable from this point any more.

```
hop(allnodes); Fthr = 0.1;
Fobj = search(aerial);
visibility(Fobj) > Fthr; repeat(
 loop(visibility(Fobj) > Fthr);
 maxdest(hop(neighbors); (Seen =
 visibility(Fobj)) > Fthr; Seen))
```

9.3 Emergency Management

Embedded communicating DSL Interpreters can convert any post-disaster wreckage into a universal spatial machine capable of self-analysis and self-recovery under integral management scenarios (as in Sapaty, Sugisaka, Finkelstein, Delgado-Frias, Mirenkov, 2006; Sapaty, 2006). For example, all casualties counting program may be as follows (with its distributed operation shown in Fig. 18):

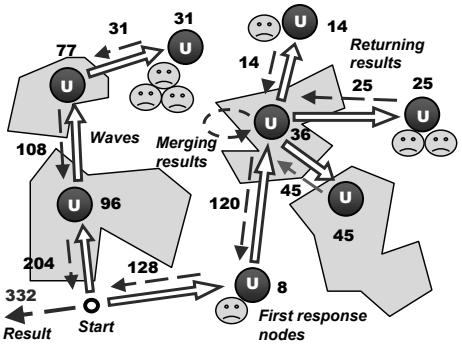


Figure 18: Counting all casualties.

```
Farea = disaster area definition;
output(sum(hop(Farea);
 repeat(free(count(casualties)),
 hop(alllinks, firstcome, Farea))))
```

Counting casualties in each region separately and organizing proportional relief delivery to each of them, may be expressed as follows:

```
Frea = disaster area definition;
split(collect(hop(Farea));
 repeat(done(count(casualties)&WHERE),
   hop(anylinks, firstcome, Farea))));
Fsupply = replicate("package", VAL:1);
move(VAL:2); distribute(Fsupply)
```

9.4 Directed Energy Systems

Directed energy systems and weapons are of rapidly growing importance in many areas, and especially in critical infrastructure protection, also on advanced battlefields (as shown in fig. 19). With the hardware equipment operating with the speed of light, traditional manned C2 is becoming a bottleneck for these advanced technical capabilities. With the technology offered, we may organize any runtime C2 infrastructures operating automatically, with the "speed of light" too, fitting the hardware capabilities and excluding men from the loop in time critical situations.

Figure 19: DEW in an advanced battlespace.

The following is an example of setting an automatic runtime C2 in a system with direct energy (DE) source, relay mirror (RM), and a target discovered, with an operational snapshot shown in Fig. 20.

```
sequence(
  parallel(
    (hop(DE); adjust(RM)),
    (hop(RM); adjust(DE, Target))),
  (hop(DE); activate(DE)))
```

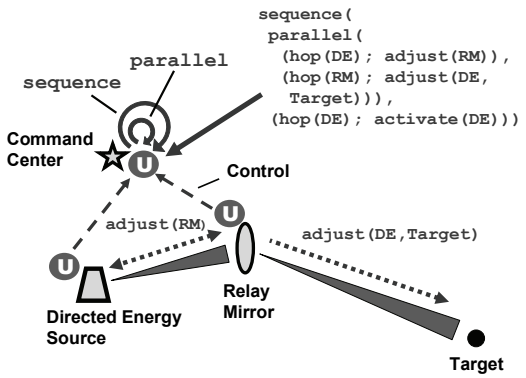


Figure 20: DE-RM-target operational snapshot.

There also exist advanced projects of global dominance with transference of directed energy, like the Boeing's Advanced Relay Mirror System (ARMS) concept. It plans to entail a constellation of as many as two dozen orbiting mirrors that would allow 24/7 coverage of every corner of the globe. When activated, this would enable a directed energy response to critical trouble spots anywhere.

We can use the distributed shortest path solution shown in section 7.1 for providing a runtime path in a worldwide distributed *dynamic* set of relay mirrors (as some of which may happen to be out of order) -- between the DE source and destination needed. This will enable optimal directed energy transfer, as shown in Fig. 21 (see also Sapaty, Morozov, Sugisaka, 2007).

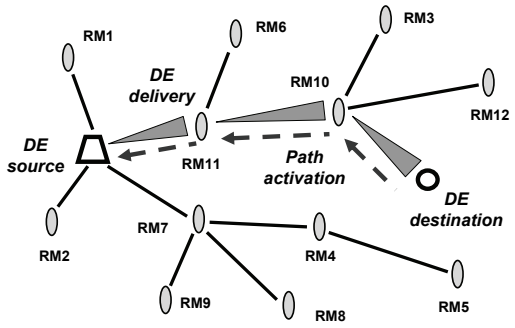


Figure 21: DE delivery via network of relay mirrors.

9.5 Electronic Warfare

Often the picture in Fig. 22 is shown as a typical example of electronic warfare. But this may rather be the last chance to survive from a missile attack. Involvement of many diverse and interlinked systems, especially for preventing and anticipating the attacks, which may be multiple and simultaneous, should be of paramount importance. All existing and being developed electronic support, attack, and protection measures have very limited scope and effect if used alone. But taken together they may provide a capability for fulfilling the objectives required. And the technology offered can readily organize this (as in Sapaty, 2007a, 2009a).

Figure 22: A Lockheed plane releasing decoy flares.

Instead of *physical flares* thrown from a plane in the final moments, we may throw, throughout the region in danger, which may be worldwide, the "*DSL scenario flares"* that can dynamically unite any available DE facilities and systems in an overwhelming electronic response to any threats.

9.6 Robotized Armies

Distributed robotized systems are of rapidly growing importance in defense (Singer, 2009, 2009a), where robotic swarming on asymmetric battlefields is becoming a major dimension of the new military doctrine for 21st century. But, as admitted by Singer, 2009, swarming, along with its simple rules of individual behavior and fully distributed nature, agility, and ubiquity, may also result in unpredictability of behavior for both sides of the conflict.

The approach briefed in this paper, also investigated in previous publications on this paradigm, is very much in line with these modern trends. Moreover, we are offering a unified solution that can harness loosely coupled swarms, always

guaranteeing their global-goal-driven behavior, where the watershed between loose swarming and strict hierarchical control may be situation dependent and changing over time (as programmed in Section 8.2).

These new doctrine trends will inevitably influence the role and sense of communications on battlefields, as with the planned drastic reduction of centralized C2 much more emphasis will be paid to intelligent tactical communications, where the scenario mobility in networked systems, offered by the approach proposed, may constitute an effective solution, with the key points (as in Sapaty, 2009b):

- Dramatic shift of global organization to intelligent tactical communications;
- Self-spreading and self-recovering mission scenarios and emergent command and control;
- Embedding intelligent protocol module into existing communication equipment;
- Situation-dependent watershed between global control and local communications.

In relation to the said above, different (including new) types of commands and control strategies for distributed robotized systems were investigated in DSL (Sapaty, Morozov, Sugisaka, Finkelstein, Lambert, 2008).

10 THE FIRST COMPUTERS

The approach offered may be compared with the invention of the first world computers (Rojas, 1997) and first high-level programming languages (Zuse, 1948/49). In our case, this computer may not only operate with data stored in a localized memory, but can cover, grasp, and manage any distributed system, the whole world including, and can work not only with information but with physical matter or objects too.

If compared with the Turing computational model, instead of the head moving through tape in performing calculations, we have a recursive formula that unwraps, replicates, covers and matches the distributed world in parallel, scanning it forth and back, bringing operations and data directly to the points of their consumption, automatically setting distributed command and control infrastructures, and organizing local and global behaviors needed.

The term "computer" first referred to the people who did scientific calculations by hand (Grier, 2005). In the end, they were rewarded by a new electronic machine that took the place and the name of those who were, once, the computers.

We can draw the following symbolic parallel with this. Despite the overwhelming automation of human activity (in both information and matter processing) the world as a whole may still be considered as remaining a *human machine*, as main decisions and global management still remain the human prerogative.

With the approach offered, we can effectively automate this top-level human supervision, actually converting the whole world into a *universal programmable machine* spatially executing global scenarios in DSL or a similar language. Despite certain science fiction flavor of this comparison, we can find numerous applications for such a global approach, some mentioned above, where top level decision automation could withstand unwanted events and save lives, and where timely human reaction may not be possible, even in principle.

11 CONCLUSIONS

We have developed and tested a novel system approach, which can describe what the system should do and how to behave on a higher level, while delegating traditional management details (like partitioning into components, their distribution, interaction and synchronization) to the effective automatic layer.

A DSL scenario is not a usual program -- it is rather a recursive active spatial pattern dynamically matching structures of distributed worlds. It has a hierarchical organization, which is grasping, by means of spreading parallel waves, the whole of the system to be comprehended and impacted.

The DSL scenarios can also create, in a parallel and fully distributed way, active distributed worlds, which become persistent and operate independently. They may spatially intervene into operation of these and other worlds and systems, changing their structures and behaviors in the way required, also self-recover from indiscriminate failures and damages, as well as repair and recover the systems managed.

Prospective applications of this work can also be linked with economy, ecology and weather prediction—by using the whole networked world as a spatial supercomputer, self-optimizing its performance. Also, for terrorism and piracy fight, where the powerful parallel ability of analyzing distributed systems and finding strong and weak patterns in them, as well as any structures (as shown in Section 7.2) may be the key to global solutions.

Crises may spark anywhere and anytime like, say, birds or swine flu or the current global economic disaster. We must be ready to react on them quickly and asymmetrically, withstanding and eradicating them -- in a "pandemic" way too, highly organized and intelligent, however.

We already have technical capabilities for this, as for example, the number of mobile phone owners in the world is approaching 3bn, and installing DSL interpreter in at least a fraction of them, can allow us to organize collective runtime (and ahead of it) response to any world events.

ACKNOWLEDGEMENTS

This work has been sponsored by the Alexander von Humboldt Foundation in Germany. Special thanks to Anatoly Morozov, Vitaly Klimenko, Yuri Korovitsky, Vladimir Voloboev, Eugene Bondarenko, and Anatoly Belyaev from the National Academy of Sciences of Ukraine for years of invaluable support and productive, often hot, discussions of these ideas. Recent chats with Juergen Kriz at the 16th GTA Convention in Osnabrueck, Germany, strengthened author's admiration of the gestalt psychology and theory, influencing general orientation of this paper.

REFERENCES

Freud, S., 1997. *General Psychological Theory*. Touchstone.

Grier, D. A., 2005. When Computers Were Human, Princeton University Press.

Koffka, K., 1913. Beiträge zur Psychologie der Gestalt. *von F. Kenkel. Zeits. f.Psychol*., 67.

Kriz, J., 2008. *Self-Actualization: Person-Centred Approach and Systems Theory*. PCCS Books.

Minsky, M., 1988. *The Society of Mind.* Simon and Schuster, New York.

Rojas, R., 1997. Konrad Zuse's Legacy: The Architecture of the Z1 and Z3. *IEEE Annals of the History of Computing*, Vol. 19, No. 2.

Rogers, C. R., 1978. *Carl Rogers on Personal Power: Inner Strength and Its Revolutionary Impact*. Trans-Atlantic Publications.

Sapaty P., 2009. Gestalt-Based Ideology and Technology for Spatial Control of Distributed Dynamic Systems, *Proc. International Gestalt Theory Congress, 16th Scientific Convention of the GTA*. University of Osnabrück, Germany.

Sapaty, P., 2009a. Distributed Capability for Battlespace Dominance. In *Electronic Warfare 2009 Conference & Exhibition*, Novotel London West Hotel & Conference Center, London.

Sapaty, P., 2009b. High-Level Communication Protocol for Dynamically Networked Battlefields. *Proc. International Conference Tactical Communications 2009 (Situational Awareness & Operational Effectiveness in the Last Tactical Mile)*. One Whitehall Place, Whitehall Suite & Reception, London, UK.

Sapaty, P. S., 2009c. Providing Spatial Integrity For Distributed Unmanned Systems". *Proc. 6th International Conference in Control, Automation and Robotics ICINCO 2009*. Milan, Italy.

Sapaty, P., 2008. Distributed Technology for Global Dominance. *Proc. of SPIE* -- Volume 6981, Defense Transformation and Net-Centric Systems 2008, Raja Suresh, Editor, 69810T.

Sapaty, P., 2008a. Grasping the Whole by Spatial Intelligence: A Higher Level for Distributed Avionics. *Proc. International Conference Military Avionics 2008*, Cafe Royal, London, UK.

Sapaty, P., Morozov, A., Finkelstein, R., Sugisaka, M., Lambert, D., 2008. A new concept of flexible organization for distributed robotized systems. *Artificial Life and Robotics*, Volume 12, Nos. 1-2/ March, Springer Japan.

Sapaty, P., 2007. Intelligent management of distributed sensor networks. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense VI*, edited by Edward M. Carapezza, *Proc. of SPIE*, Vol. 6538, 653812.

Sapaty, P., 2007a. Global Management of Distributed EW-Related Systems. *Proc. International Conference Electronic Warfare: Operations & Systems*. Thistle Selfridge, London, UK.

Sapaty, P., Morozov, A., Sugisaka, M., 2007. DEW in a Network Enabled Environment. *Proc. International Conference Directed Energy Weapons 2007*. Le Meridien Piccadilly, London, UK.

Sapaty, P., 2006. Crisis Management with Distributed Processing Technology. *International Transactions on Systems Science and Applications*. Vol. 1, no. 1.

Sapaty, P., Sugisaka, M., Finkelstein, R., Delgado-Frias, J., Mirenkov, N., 2006. Advanced IT Support of Crisis Relief Missions. *Journal of Emergency Management.* Vol.4, No.4, July/August.

Sapaty, P. S., 2005. *Ruling Distributed Dynamic Worlds*. John Wiley & Sons, New York.

Sapaty, P. S., 2002. Over-Operability in Distributed Simulation and Control. *The MSIAC's M&S Journal Online*. Winter Issue, Volume 4, No. 2, Alexandria, VA, USA.

Sapaty, P. S., 1999. *Mobile Processing in Distributed and*

Sapaty, P. S., 1993. A distributed processing system, *European Patent No. 0389655*. European Patent Office.

Singer, P. W., 2009. Wired for War. Robots and Military Doctrine. *JFQ / issue 52*, 1st quarter.

Singer, P. W., 2009a. *Wired for War: The Robotics Revolution and Conflict in the 21st Century*. Penguin.

Smuts, J. C., 2007. *Holism And Evolution*. Kessinger Publishing, LLC

Wertheimer, M., 1924. *Gestalt Theory*. Erlangen, Berlin.

Wilber, K. 2009. *Ken Wilber Online: Waves, Streams, States, and Self--A Summary of My Psychological Model (Or, Outline of An Integral Psychology)*. Shambhala Publications.

Zuse, K., 1948/49. Uber den Plankalk, als Mittel zur Formulierung schematisch kombinativer Aufgaben. In *Archiv Mathematik*, Band I.

BRIEF BIOGRAPHY

Dr Peter Simon Sapaty, chief research scientist and director of distributed simulation and control project at the Institute of Mathematical Machines and Systems, National Academy of Sciences of Ukraine, is with networked systems for more than four decades. A power network engineer on education, he created citywide heterogeneous computer networks from the end of the sixties, implemented a multiprocessor macro-pipeline supercomputer in the seventies-eighties, and since then used distributed computer networks for solving complex problems of most different natures—from distributed knowledge bases to intelligent network management to road traffic control to simulation of battlefields. He also worked in Germany, UK, Canada, and Japan as Alexander von Humboldt Foundation fellow, project leader, research professor, department head, and special invited professor; created and chaired a SIG on mobile cooperative technologies within Distributed Interactive Simulation project in the US. Peter invented a higher-level distributed networking technology used in different countries and resulted in a European Patent and two John Wiley books. His interests include coordination and simulation of large distributed dynamic systems under the holistic and gestalt principles, with application in advanced command and control, cooperative robotics, infrastructure protection, crisis management, and especially for finding asymmetric solutions in unpredictable and hostile environments.