The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th ’12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Withstanding Asymmetric Situations in Distributed Dynamic Worlds

Peter S. Sapaty

Institute of Mathematical Machines and Systems
National Academy of Sciences
Glushkova Ave 42, 03187 Kiev, Ukraine
(Tel: +380-67-4199223; Fax: +380-44-5266457)
sapaty@immsp.kiev.ua, peter.sapaty@gmail.com

Abstract: A high-level ideology and technology will be revealed that can effectively convert any distributed system (manned, unmanned, or mixed) into a globally programmable spatial machine capable of operating without central resources. Compact mission scenarios in a special high-level language can start from any point, runtime covering & grasping the whole system or its parts needed, setting operational infrastructures, and orienting local and global behavior. The approach offered can be particularly useful for quick reaction on asymmetric situations and threats the world is facing, paving the way to massive use of cooperative robotics and gradual transition to unmanned systems for solving critical problems in unpredictable environments.

Keywords: distributed dynamic worlds, asymmetric situations and threats, Spatial Grasp Technology, Distributed Scenario Language, parallel networked interpretation, multi-robot systems.

1 INTRODUCTION

In our modern dynamic world we are meeting numerous irregular situations and threats where proper reaction could save lives and wealth and protect critical infrastructures. For example, no secret that world powerful armies with traditional system organizations are often losing to terrorists, insurgents or piracy with primitive gadgets but very flexible structures making them hard to detect and fight. And delayed reaction to earthquakes or tsunamis is a result of inadequacy of system organizations too.

A novel philosophy and supporting high-level networking technology will be described that can quickly react on irregular situations and threats and organize any available human and technical resources into operable systems providing global awareness, pursuing global goals and self-recovering from damages.

The approach allows us at runtime, on the fly, formulate top semantics of the needed reaction on asymmetric events in a special Distributed Scenario Language (DSL), shifting most of traditional organizational routines to automated up to fully automatic implementation, with effective engagement of unmanned systems.

The details of this technology, based on gestalt and holistic principles [1] rather than traditional multi-agent organizations [2, 3] will be revealed in this paper, with explanation of different DSL scenarios that can be effectively executed by self-organized robotic groups. These scenarios include collective navigation of distributed spaces, coastline patrols, outlining and impacting forest fire zones, and swarm-against-swarm solutions where highly organized robotic swarms can fight another, manned, groups related, say, to piracy intrusions.

The technology offered provides a unified solution to human-robot interaction and multi-robot behaviors just as a derivative of parallel and distributed interpretation of system scenarios in DSL.

2 TRADITIONAL SYSTEM ORGANIZATIONS AND THEIR PROBLEMS

2.1 From system structure to system function

The traditional approach to system design, development and management is when the system structure and system organization are primary, created in advance, and global function with overall behavior are secondary, as in Fig. 1.

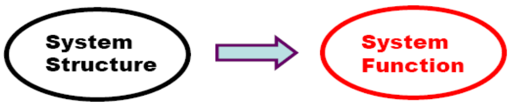

Fig. 1. Traditional approach to system design

Typical examples of the traditional approach are multi-agent organizations [2, 3], where global system behavior is the result of work and interaction of predetermined parts (agents). In this respect we can name the 4D/RCS Model Architecture [4], with its block diagram shown in Fig. 2.

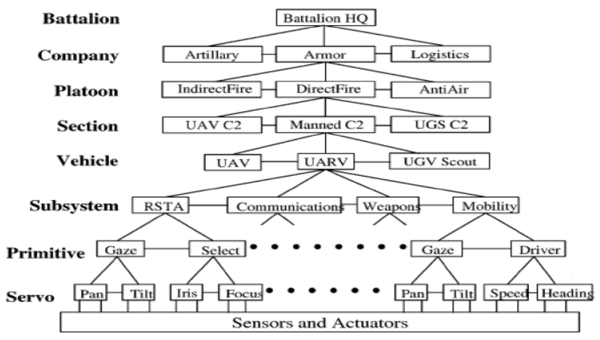

Fig. 2. 4D/RCS model architecture

4D/RCS prescribes a hierarchical control principle, where commands flow down the predefined hierarchy, and status feedback and sensory information flows up. Large amounts of communication may also occur between nodes at the same level, particularly within the same subtree of the command tree. Future Combat Systems (FCS) project [5] was ideologically and technologically based on this organizational (as well as artificial intelligence) hierarchical model.

2.2 The problems with classical organizations

The related systems, where we first formalize and build the system structure and organization and then try to get from these the global behavior needed, are usually static, and may often fail to adapt to highly dynamic and asymmetric situations. If the initial goals change, the whole system may have to be partially or even completely redesigned and reassembled. Adjusting the already existing system to new goals and functionality needed may result in a considerable loss of system's integrity and performance.

3 AN ALTERNATIVE: THE SPATIAL GRASP TECHNOLOGY (SGT)

3.1 SGT basic idea

Within the approach offered (also known as "overoperability" [6, 7] in contrast to the conventional *interoperability*) the global function and overall behavior are considered, as much as possible, to be primary. Whereas system structure and organization (command and control including) are secondary, with the latter as a derivative of the former, as in Fig. 3.

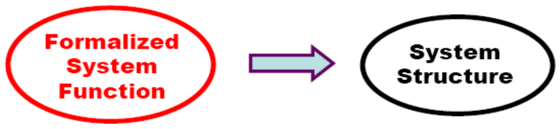

Fig. 3. SGT basic idea of system creation and organization

The advantages of this (actually, the other way round) approach include high potential flexibility of runtime system creation and organization, especially in quick responses to asymmetric events, and enhanced opportunities for automated up to fully automatic (unmanned) solutions.

3.2 Parallel spatial grasp of distributed worlds

SGT is based on a formalized wavelike seamless navigation, coverage, penetration, and grasping of distributed physical and virtual spaces, as shown in Fig. 4.

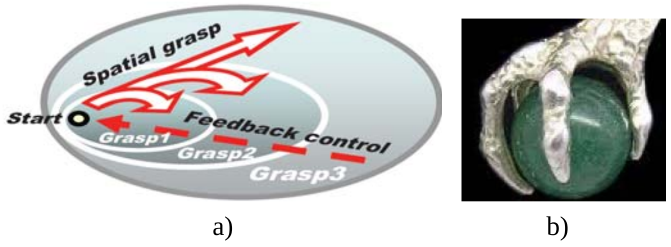

Fig. 4. Incremental integral grasping of distributed worlds: a) virtual interpretation, b) symbolic physical analogy

This top mode of system vision has strong psychological and philosophical background, reflecting, for example, how humans (top commanders) mentally plan, comprehend, and control complex operations in distributed environments.

3.3 Distributed scenario interpretation

The approach in practice works as follows. A network of universal control modules U, embedded into key system points, collectively interprets system scenarios expressed in DSL, as shown in Fig. 5. System scenarios, based on the spatial grasp idea (representing any parallel and distributed algorithms, spatial cycles including), can start from any node, subsequently covering the system at runtime.

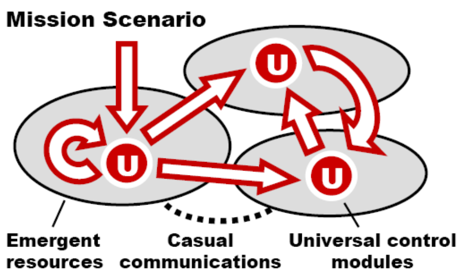

Fig. 5. Scenario execution in dynamic environments

DSL scenarios are very compact and can be created on the fly. Different scenarios can cooperate or compete in a networked space (depending on real control or distributed simulation mode) as overlapping fields of solutions. Self-spreading scenarios can also create runtime knowledge infrastructures distributed between system components (robots, sensors, humans). These infrastructures can effectively support distributed databases, command and control, situation awareness, and autonomous decisions, as well as any other computational or control models.

More details on the SGT, its core language DSL, and its distributed interpreter can be found elsewhere [8-16], with some key features (necessary for explanation of the chosen here applications) being briefed in the following sections.

4 DISTRIBUTED SCENARIO LANGUAGE, DSL

DSL differs radically from traditional programming languages. It allows us to directly move through, observe, and make any actions and decisions in fully distributed environments. DSL directly operates with:

- *Virtual World* (VW), finite and discrete, consisting of nodes and semantic links between them.
- *Physical World* (PW), infinite and continuous, where each point can be identified and accessed by physical coordinates.
- *Virtual-Physical World* (VPW), finite and discrete, similar to VW but associating virtual nodes with certain PW coordinates.

4.1 DSL basic features

Any sequential or parallel, centralized or distributed, stationary or mobile algorithm operating with information and/or physical matter can be written in DSL on a high level. Its top level recursive structure is shown in Fig. 6.

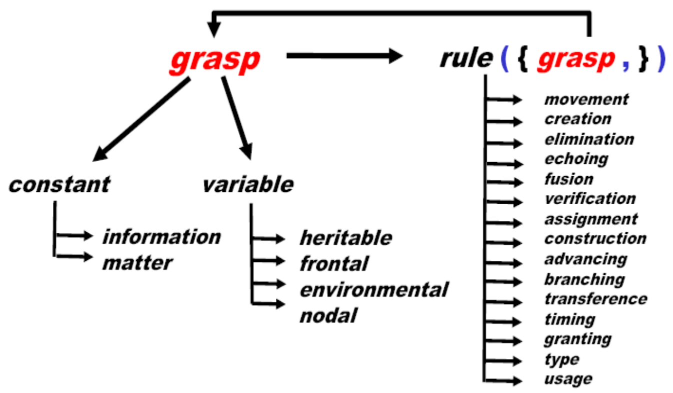

Fig. 6. DSL top level syntax

DSL main features may be summarized as follows:

- A DSL scenario develops as parallel transition between sets of progress points (*props*).
- Starting from a prop, an action may result in other props.
- Each prop has a resulting *value* and a resulting *state*.
- Different actions may evolve *independently or interdependently* from the same prop.
- Actions may also spatially *succeed each other*, with new ones applied in parallel from all props reached by the previous actions.
- Elementary operations may directly use *values of props obtained from other actions* whatever complex and remote.
- 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 simultaneously link with the same points of the worlds.
- Staying with the world points, it is possible to *directly access and impact* local world parameters, whether virtual or physical.

4.2. DSL rules

The basic construct, *rule*, of the language can represent any action or decision and can, for example, be as follows (this list is far from being complete):

- Elementary arithmetic, string or logic operation.
- Hop in a physical, virtual, or combined space.
- Hierarchical fusion and return of (remote) data.
- Distributed control, both sequential and parallel.
- A variety of special contexts for navigation in space, influencing operations and decisions.
- Type or sense of a value, or its chosen usage, guiding automatic interpretation.
- Creation or removal of nodes and links in distributed knowledge networks.

4.3 Spatial variables in DSL

Working in fully distributed physical or virtual environments, DSL has different types of variables, called *spatial*, effectively serving multiple cooperative processes:

- *Heritable variables* – these are starting in a prop and serving all subsequent props, which can share them in both read & write operations.
- *Frontal variables* – are an individual and exclusive prop's property (not shared with other props), being transferred between the consecutive props, and replicated if from a single prop a number of other props emerge.
- *Environmental variables* – are accessing different elements of physical and virtual words when navigating them, also a variety of parameters of the internal world of DSL interpreter.
- *Nodal variables* – allow us to attach an individual temporary property to VW and VPW nodes, accessed and shared by all props associated with these nodes.

These variables allow us to create spatial algorithms working *in between components* of distributed systems rather than *in* them, thus allowing for highly flexible, robust and capable of self-recovery solutions, even though different components may fail indiscriminately. Such algorithms can freely move in distributed processing environments (partially or as an *organized whole*), always preserving global integrity and overall control.

Traditional abbreviations of operations and delimiters can be used too, substituting some rules, as in following examples throughout this text, in order to shorten DSL programs (but always remaining within the general recursive syntactic structure shown in Fig. 6).

5 THE DSL INTERPRETER

5.1 Distributed interpreter organization

The DSL interpreter consists of specialized modules (which can work in parallel) handling and sharing specific language interpretation data structures [10, 13-16]. The network of the interpreters (the latter encircled as modules U in Fig. 7) can be mobile and open, changing the number of nodes and communication structure at runtime. Communicating copies of the interpreter can be concealed, if needed (say, for operation in hostile environments).

The heart of the distributed interpreter is its spatial *track system*. The dynamically crated track forests are used for supporting (or removing) spatial variables and echoing and merging different types of control states and remote data.

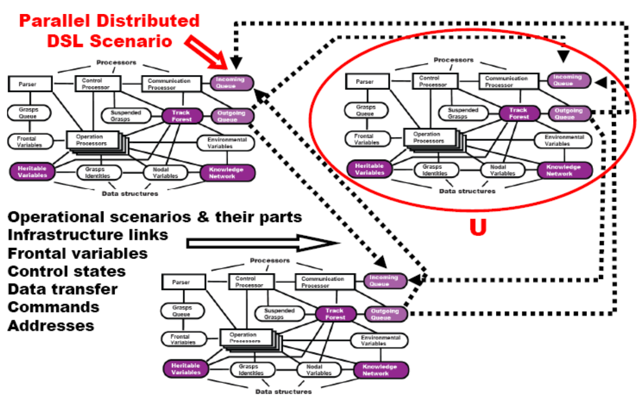

Fig. 7. Networked DSL interpreter organization

Being self-optimized in the echo processes, the track forests are dynamically covering the systems in which DSL scenarios evolve, keeping the overall parallel and distributed process integrity as well as local and global control. They also route further grasps to the positions in physical, virtual or combined spaces reached by the previous grasps, uniting them with the frontal variables left there by preceding grasps.

5.2 Integrating interpreter with usual robotic functionality

Installing DSL interpreters (as universal modules U, see Fig. 8) into mobile robots (ground, aerial, surface, underwater, space, etc.) allows us to organize effective group solutions (incl. any swarming) of complex problems in distributed physical spaces in a clear and concise way, effectively shifting traditional management routines to automatic levels. Human-robot interaction and gradual transition to fully unmanned systems are essentially assisted too.

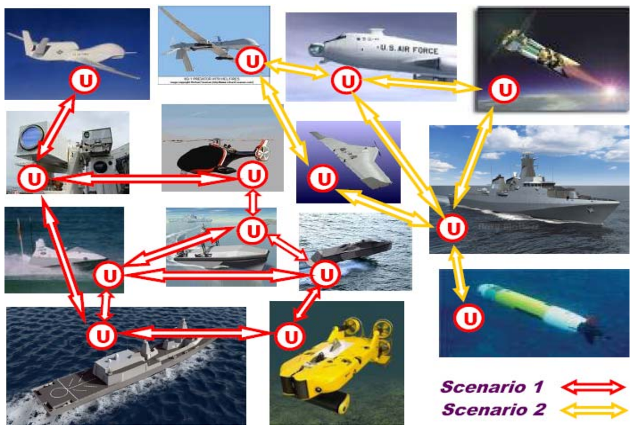

Fig. 8. Examples of cooperative robotic scenario skeletons

Any groups of manned-unmanned devices with DSL interpreters implanted into them, with any communication networks in between, can serve as universal spatial machines capable of doing any jobs together, under a unified control automatically emerging from high-level DSL scenarios.

6 EXAMPLE OF SEMANTIC, TASK LEVEL

By embedding DSL interpreters into robotic vehicles we can task them on a higher, *semantic* level, skipping numerous traditional details of management of them as a group -- fully delegating these to an automatic solution. An exemplary semantic level tasking may be as follows.

Go to physical locations of the disaster zone with coordinates: (x1, y1), (x2, y2), (x3, y3); evaluate radiation level at each location; return its maximum value with attached exact coordinates of the respected location to the headquarters; and launch from the latter a massive cleanup operation at this location.

The DSL program will strictly follow this scenario:

```
Location = maximum(move(x1_y1, x2_y2, x3_y3);
  attach(evaluate(radiation), WHERE));
move(Location[2]); massive_cleanup(radiation)
```

This (inherently parallel and fully distributed) scenario can be executed with any available number of mobile robots (practically from one to four), and the number of robots may change at runtime. Distributed DSL interpreter automatically creates the needed operational and command and control infrastructures of the robotic group and guarantees full task execution under any variations [8, 9].

7 PATROLLING COASTAL WATERS

This scenario may be suitable for both surface and varying depth underwater search of intrusions in the coastline zone, but for simplicity we will be assuming here only two dimensional space to be navigated.

At the beginning let us create a coastal waypoint map in the form of a semantic network, as in Fig. 9 (where `r` is chosen as an arbitrary name of links between the nodes-waypoints). The corresponding DSL solution is as follows.

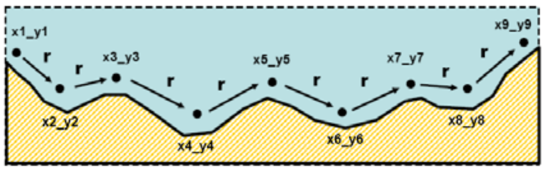

Fig. 9. Coastal waypoint map

```
create(
 #x1_y1; +r#x2_y2; +r#x3_y3; ... +r#x9_y9)
```

A single USV (or UUV) solution repeatedly navigating all coastal area by the map created is shown in Fig. 10 and DSL program that follows (searching the water space for alien objects by the `depth` available by vehicle's sensors).

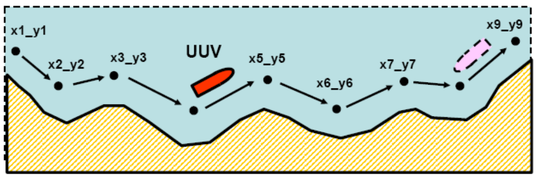

Fig. 10. Patrolling coastal waters with a single vehicle

```
move(hop(x1_y1)); R = +r;
repeat(repeat(move(hop(R));
         check_report(depth)); invert(R))
```

Two-vehicle parallel solution is shown in Fig. 11 and by the following program, with vehicles are moving according to the coastal map independently, assuming each having embedded automatic procedures for avoiding possible collisions with the other vehicle.

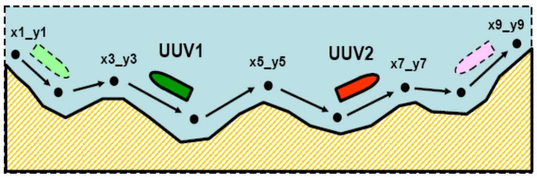

Fig.11. Patrolling coastal waters with two vehicles

```
(move(hop(x1_y1)); R = +r),
(move(hop(x9_y9)); R = -r);
repeat(repeat(move_avoid(hop(R));
          check_report(depth)); invert(R))
```

Another solution for the two-vehicle case may be when each vehicle turns back if discovers another patrol vehicle on its way, checking for this its vicinity by `depth2`).

```
(move (hop(x1_y1)); R = +r),
(move (hop(x9_y9)); R = -r);
repeat(repeat(none(depth2); move(hop(R));
           check_report(depth)); invert(R))
```

For the both latter cases, the whole coastline will always be searched in full if at least a single vehicle remains operational.

8 BATTLING FOREST FIRES

We will consider a solution where distributed physical space is randomly searched by simultaneous propagation of multiple reconnaissance units, which when discover irregularities (e.g. forest fires) move further and encircle respected zones, collect their perimeter coordinates, transfer them to the headquarters (HQ), and ultimately initiate massive impact on the zones under fire.

The zones with fires and initial positions of reconnaissance units are shown in Fig. 12*a*, and intermediary positions of robotic unites moving randomly-oriented (repeatedly shifting their positions within certain coordinate sector) are in Fig. 12*b*.

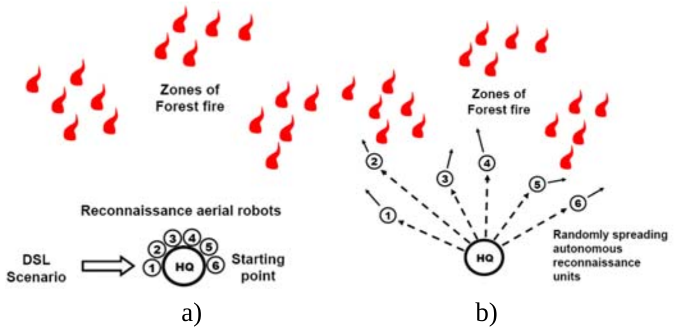
Zones of Forest fire
Reconnaissance aerial robots
DSL Scenario
HQ
Starting point
a)
Zones of Forest fire
Randomly spreading autonomous reconnaissance units
HQ
b)

Fig.12. Initial scenario injection (a) and robots movement (b)

After detecting fire locations, the reconnaissance units that reached them begin moving around the fire zones, having initially randomly chosen the encirclement orientation (i.e. clockwise or anticlockwise). In each step they accumulate coordinates of the periphery of fire zones, and upon termination of the encirclement send the completed zone coordinates to the headquarters (HQ). Getting the latter, the HQ is launching a massive direct impact on the zones outlined, as shown in Fig. 13, which may be manned, unmanned, or mixed. The full DSL scenario for this task may be as follows.

```
move(HQ); create_nodes(1,2,3,4,5,6);
repeat(shift(random(limits));
 if(check(fire),
    (Zone = WHERE; Direction =
      random(clockwise, anticlockwise);
      repeat(
       move_around(fire, Direction, depth);
       append(Zone, WHERE);
       if(distance(WHERE,Zone[1])<threshold,
          (hop(HQ); impact(Zone); done)))))
```

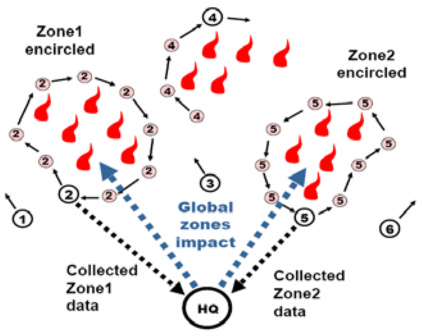

Fig.13. Encircling fire zones followed by global impact

Other interpretations of this scenario may be dealing with radiation zones, environment pollution, zones of terrorist activities, zones of fish concentration, etc., with aerial, ground, surface or underwater robots engaged.

9 SWARM AGAINST SWARM SCENARIO

As a more complex scenario example in DSL we will consider here the case where an unmanned swarm is opposing other (possibly, manned) group/swarm, as in Fig. 14. This may relate, for example, to fighting piracy in maritime environment where aerial, surface and underwater unmanned vehicles, working cooperatively under a unified control, can be used for withstanding this negative activity taking place worldwide.

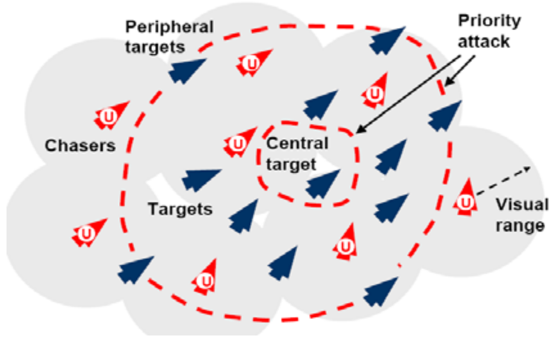

Fig. 14. Fighting group targets with unmanned swarms

Main features of this scenario are as follows:

- Initial launch of the swarmed chasers (shown in red in Fig. 14, with DSL interpreters embedded, which can communicate with each other) into the expected piracy area.
- Discovering targets and forming their priority list by their positions in physical space where maximum priority is assigned to topologically central targets as potential control units of the intruders.
- Other targets are sorted out by their distance from the topological center of their group, estimated previously.

- Most peripheral targets (those in maximum distance from the topological center, as potentially having more chances to escape, are being of higher priority too.
- Assigning available chasers to targets, classifying them as engaged, with chasing and neutralizing targets, and subsequently returning them into status free after performing mission.
- The vacant chasers are again engaged in the targets selection & impact.

This entire advanced swarm-against-swarm scenario may be expressed in DSL in a very compact form, as follows.

```
frontal(Next);
sequence(
 start_launch(all_free_chasers,targets_area),
 repeat(
  hop(any_free_chaser);
  All_targets = merge(hop(all_free_chasers);
  coordinates(targets_seen));
  nonempty(All_targets);
  Center = average(All_targets);
  List = min_max_sort(split(All_targets);
      attach(distance(VALUE,Center),VALUE);
  List = append(withdraw(List, last), List);
  loop(nonempty(List); Next =
   element(withdraw(List, first), second);
   Chaser =
    element(min(hop(all_free_chasers);
      attach(distance(WHERE,Next),ADDRESS),
       second);
   release(hop(Chaser); STATUS = engaged;
    pursue_investigate_neutralize(Next);
    STATUS = free))));
```

It is worth noting that all the chaser swarm management expressed or automatically induced by the above program is done exclusively within the swarm itself, without any external intervention, which dramatically simplifies external control of this multi-robot operation.

10 CONCLUSION

A brief summary of advantages of the approach offered may be as follows.

- The Spatial Grasp ideology and technology can dramatically simplify application programming in distributed dynamic systems.
- Setting multi-robot solutions in DSL may often be comparable in complexity to routine data processing in traditional languages.
- External management of multi-robot systems may not depend on the number of components in them due to their internal self-organization and automatic command and control inside robotic groups.
- Formalization of mission scenarios in DSL can make human-robot interaction and transition to fully unmanned systems natural and straightforward.
- Spatial swarm intelligence in DSL can successfully compete with human collective intelligence, outperforming the latter in time critical situations.

In addition to the features listed above, we can state that in comparison with other systems, DSL interpreter represents an *embedded universal intelligence* common to all applications. Any scenario can be executed by a network of such intelligences. In other approaches, most of the system intelligence has to be programmed *explicitly for each application,* thus enormously complicating mission planning and management.

All communications among unmanned units, also between manned and unmanned ones, are on a high, semantic level in DSL. They are *very compact* (often hundreds times shorter than in other languages) which may be essential for maritime (especially underwater) operations with casual and limited data channels.

REFERENCES

[1] Wertheimer M (1924), Gestalt theory, Erlangen, Berlin
[2] Minsky M (1988), The society of mind, Simon and Schuster, New York
[3] www.wikipedia.org/wiki/Multi-agent_system
[4] 4D/RCS: A reference model architecture for unmanned vehicle system, version 2.0 (2002), Report NIST
[5] Feliciano CN (2009), The army's future combat system program (Defense, Security and Strategy Series), Nova Science Pub Inc.
[6] Sapaty PS (2002), Over-operability in distributed simulation and control, The MSIAC's M&S Journal Online, Winter Issue, Volume 4, No. 2, Alexandria, VA, USA
[7] Sapaty P (2009), The over-operability organization of distributed dynamic systems for asymmetric operations, Proc. IMA Conference on Mathematics in Defence, Farnborough, UK
[8] Sapaty P (2011), Spatial grasp technology for high-level management of distributed unmanned systems, Unmanned Systems Asia 2011, Singapore
[9] Sapaty PS (2011), Seeing and managing distributed spaces using maritime unmanned systems, GLOBAL OPV and Maritime Unmanned Systems Summit, Dedeman Hotel, Istanbul, Turkey
[10] Sapaty P (2011), Meeting the world challenges with advanced system organizations, Informatics in Control Automation and Robotics, Lecture Notes in Electrical Engineering, Vol. 85, 1st Edition, Springer
[11] Sapaty P, Kuhnert D, Sugisaka M, Finkelstein R (2009), Developing high-level management facilities for distributed unmanned systems, Proc. Fourteenth International Symposium on Artificial Life and Robotics (AROB 14th'09), B-Con Plaza, Beppu, Oita, Japan
[12] 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, Numbers 1-2, ISSN: 1433-5298 (Print) 1614-7456 (Online), Springer Japan
[13] Sapaty P (2008), Distributed technology for global dominance, Proc. of SPIE, Volume 6981, Defense Transformation and Net-Centric Systems 2008, 69810T
[14] Sapaty P (2005), Ruling distributed dynamic worlds, John Wiley & Sons, New York.
[15] Sapaty P (1999), Mobile processing in distributed and open environments, John Wiley & Sons, New York
[16] P. Sapaty (1993), A distributed processing system, European Patent No. 0389655, Publ. 10.11.93, European Patent Office.