The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th '10),
B-Con Plaza, Beppu,Oita, Japan, February 4-6, 2010

Countering Asymmetric Situations with Distributed Artificial Life and Robotics Approach

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

Masanori Sugisaka
Department of Mechanical and Electrical Engineering, Nippon Bunri University
1727 Oaza Itiki, Oita, 870-0397, Japan, ms@alife-robotics.co.jp

Abstract

A novel control model and technology, creating distributed virtual systems with artificial life features, is discussed. They become capable of runtime reshaping, adapting to unknown environments, and pursuing global goals. The approach is based on known holistic and gestalt principles, where the whole is first and parts are treated in the context of the whole. Distributed Scenario Language, DSL, the core of the approach, and its spatial interpretation in networked systems will be revealed. Mission scenarios in DSL, covering, integrating, tasking, and controlling distributed resources (robotic swarms including), can effectively fight world disasters and crises.

Keywords: irregular situations, integrity, overoperability, distributed scenario language, networked interpretation, smart structures, self-recovery, artificial life and robotics.

1 Introduction

The world dynamics is increasing due to global warming, numerous natural and manmade disasters, military conflicts, and international terrorism. New approaches to organization of distributed systems, especially for solving irregular and "asymmetric" problems, are needed [1]. The approach offered, symbolically called *overoperability*, allows us to create, modify, analyze, process, simulate, and manage any distributed systems by establishing advanced global control over them [2].

Within the overoperability philosophy, an integral mission scenario expressed in a special wave-like formalism (see Fig. 1*a*) is executed in a parallel manner by dynamically networked universal control modules U embedded into distributed worlds (Fig. 1*b*). This scenario, written in a special high-level Distributed Scenario language (DSL), can start from any unit and dynamically cover the whole system, setting its internal organization, and orienting behavior [2,3].

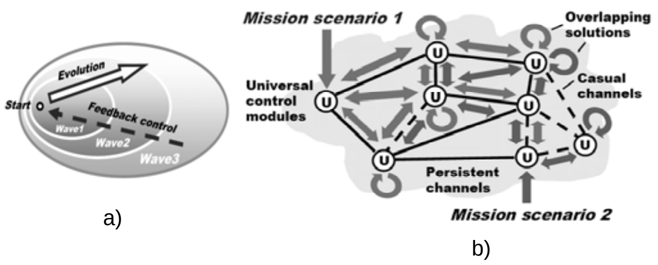

Figure 1. The overoperability paradigm.

During the scenario evolution, any operations can be accomplished in the distributed world, causing, if needed, movement of code and equipment and creation and maintenance of physical and virtual infrastructures supporting the missions. Different spatial scenarios can cooperate or compete in the networked space (as in Fig. 1*b*), allowing for effective distributed simulation of complex dynamic systems or their live management and control, with any combination of the two.

This paradigm has been extensively studied, discussed, and published elsewhere [4-6], and here we will be concentrating only on the latest updated version of DSL and its application for the creation of smart distributed structures, with their use in irregular situations.

2 DSL: The Scenario Language

DSL allows us to directly express semantics of problems to be solved in distributed worlds and, if needed, implicit or explicit system behavior to solve these problems.

2.1 DSL Key Features

The language operates with:

- Virtual World (VW), which is discrete and consists of nodes and links connecting the nodes.
- Continuous Physical World (PW), points in which are accessible by physical coordinates.
- Virtual-Physical World (VPW), as an extension of VW, where nodes additionally associate with coordinates in PW.

Other key DSL features include:

- A scenario in it develops as a transition between sets of progress points (*props*), as *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* (full success, also allowing us to proceed further), *done* (success with planned termination), *fail* (regular failure, with local termination), and *abort* (emergency failure, terminating the whole distributed process, associated with other props too).
- Different actions may evolve independently or interdependently from the same prop, forming altogether the resultant set of props.
- Actions may also spatially succeed each other, with new ones applied in parallel from all props reached by the preceding actions.

- Elementary operations can directly use local or remote values of props obtained from other actions (the latter including the whole scenarios).
- Elementary operations can result in open values that can be used by other operations in an expression or by the following operations in a sequence. They can also be directly assigned to local or remote variables (an access to which may invoke scenarios of any complexity).
- Any prop can associate with a node in VW or a position in PW, or both.
- Any number of props can be simultaneously linked with the same points of the worlds.
- Staying with world points (virtual, physical, combined) it is possible to access and update local data in them.
- DSL can be used as a universal programming language.

2.2 DSL Syntax and Main Constructs

DSL has a recursive syntax, which on top level may be expressed as follows (programs are called *waves*, braces show repetition, and vertical bar delimits alternatives):

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

The basic construct, *rule*, can represent any definition, action or decision, being for example:

- 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.

There are different types of variables in DSL:

- *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 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; they can be accessed and shared by any props associated with these nodes.

These variables, especially when used together, allow us to create efficient spatial algorithms working *between* components of distributed systems rather than *in* them.

Elementary examples in DSL may look like follows.

- assignment of a sum of values to a variable: `assign(Result, add(27,33,55.6))`
- parallel hop into two given physical locations: `move(location(x5,y8), location(x1,y3))`
- creating isolated node `Peter` in a virtual space: `create(node(Peter))`
- extending the previous single-node network with a new link-node pair (`father of`, `Alex`): `hop(Peter); create(+fatherof, Alex))`

Traditional abbreviations of operations and delimiters can be used too, as in further examples throughout this text, to shorten the DSL programs.

3 Distributed DSL Interpreter

The DSL interpreter ([6], Fig. 2), with the following features, has been prototyped on different platforms.

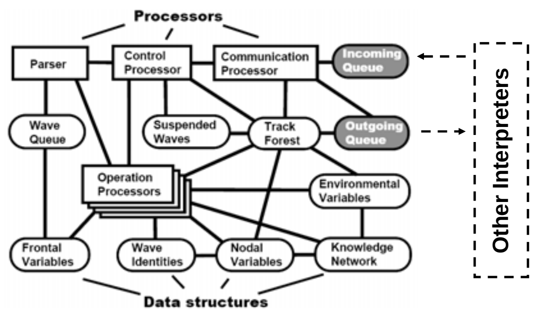

Figure 2. Organization of DSL interpreter.

- It consists of a number of specialized modules working in parallel and handling and sharing specific data structures 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*, with its parts kept in the Track Forest memory of local interpreters; these being logically interlinked with such parts in other interpreter copies, forming altogether *indivisible space coverage*. This allows for hierarchical command and control and remote data and code access, with high integrity of parallel distributed solutions.
- Copies of the interpreter can be concealed, say, in hostile systems, allowing us to impact them globally.

The dynamically crated track trees, spanning the systems in which DSL scenarios evolve, are used for supporting spatial variables and echoing and merging different types of control states and remote data, being self-optimized in the echo mode. They also route further waves to the positions in physical, virtual, or combined spaces reached by the previous waves, uniting them with the frontal variables delivered there by preceding waves.

4 Operating with Distributed Structures

Network creation. To create the virtual network of Fig. 3*a* in a distributed environment, and in a fully distributed way, the following DSL program will be sufficient (expressing the directed graph template of Fig. 3*b* based on depth-first spanning tree of the network, with all links named r, for simplicity):

```
create(hop(a); r#b; r#c; r##a,(r#d; r##b))
```

Starting with node a, it gradually self-evolves in a distributed space while creating the network topology needed (in the navigation mode shown in Fig. 1*b*).

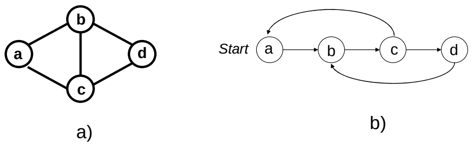

Figure 3. Distributed creation of a network structure.

Network modification. Another DSL scenario can make any modification of this network, also in a fully distributed and parallel way, say, substituting all existing triangles in the network with stars (naming additional central star nodes with combination of fringe node names), as in Fig. 4*a-b*:

```
hop(allnodes); F=C; twice(#; P>A; F&=C);
#; C==F:1; create(hop(unite(F)); r##F);
remove(##F)
```

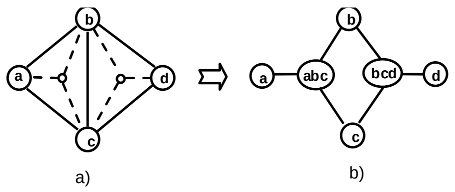

Figure 4. Distributed structure modification.

Network self-recovery. Applying another special DSL scenario to any network can make it capable of self-recovery after any indiscriminate damages of nodes and links, where missing elements can be restored by remaining neighboring nodes. The restored nodes can, in their turn, restore the other nodes (including the ones that restored them), and so on. A simplified example of such a program converting the whole network into a *self-regenerating live creature* may be as follows (the lost nodes a and d and links to them are restored by nodes b and c, as in Fig. 5*a-b*).

```
Fp={repeat(split(diff(Fi,(#; C))); Fn=V;
[or((direct#Fn; create(r##P)),
    (create(r#Fn); Fi=(#; C); run(Fp)))])};
hop(allnodes); Fi=(#; C); run(Fp)
```

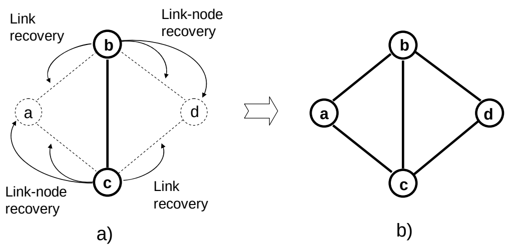

Figure 5. Distributed network self-recovery.

Distributed topology self-analysis. DSL allows us to directly analyze and process distributed topologies in a parallel way. For example, to find the weakest nodes in a network (like *articulation points*, Fig. 6*a*), which when removed split the network into disjoint parts, we only need the following program (resulting in articulation node d).

```
hop(allnodes); ID=C;
and((random(#); repeat(firstcome(#))),
    (firstcome(#)), out(C))
```

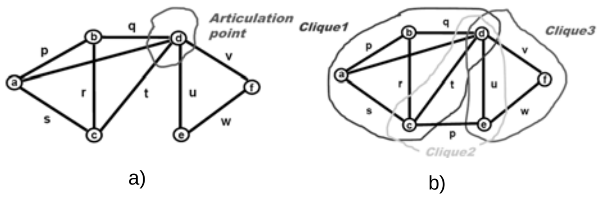

Figure 6. Distributed discovery of topological features.

Cliques (or fully connected sub-graphs of a graph, as in Fig. 6*b*), on the contrary, may be considered as strongest parts of a system. They can be found in parallel by the following program, resulting for the network in Fig. 6*b* in cliques: (a,b,c,d),(c,d,e),(d,e,f).

```
hop(allnodes); Fc=C;
repeat(#; notbelong(C, Fc);
 and(done(andparallel(#Fc)),
     or(done(B>C), Fc&=C))); out(Fc)
```

More on distributed topology operations are in [4,5].

5 Researched Applications

Overlaying the obtained integral virtual solutions onto networked hardware, which may be heterogeneous and open, allows us to work in the following modes:

- Simulation mode, where using computer networks for parallel simulation of large systems provides realistic results, both functional and behavioral.
- Live control mode, causing and guiding the needed hardware behavior in solving complex problems.
- Combined mode, where distributed simulation serves as intelligent look-ahead part of live control.

Collective Robotics. Installing the interpreter in mobile robots of different types (as in Fig. 7) allows us to organize effective group solutions (including any swarming) in distributed physical spaces.

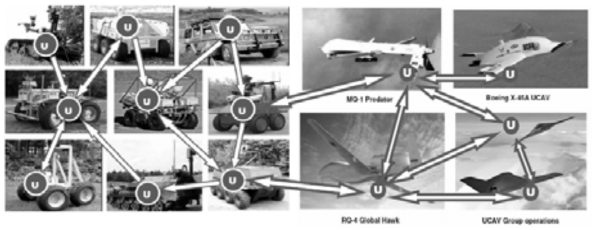

Figure 7. Grouping unmanned vehicles.

A simple example task here may be formulated as:

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));
  append(evaluate(destruction), WHERE)))
```

Details of automatic implementation of this scenario, as well as of many others, by different numbers of mobile robots and at different system levels are discussed in details elsewhere [7]. Under the technology developed, loose robotic swarming may be combined with strong hierarchical control to make quick global solutions and withstand unexpected situations.

Terrorism and Piracy Fight. No secret that mightiest world armies with classical organizations are often powerless against poorly armed terrorists and pirates, who are using flexible asymmetric tactics (see in Fig. 8 the 2009 world piracy map with possible information leakages, forming altogether a sophisticated distributed network). The ideology and technology discussed here can dynamically organize the whole world to withstand such activities, offering runtime spatial solutions--from global network search to managing unmanned swarms for asymmetric responses to asymmetric attacks [3].

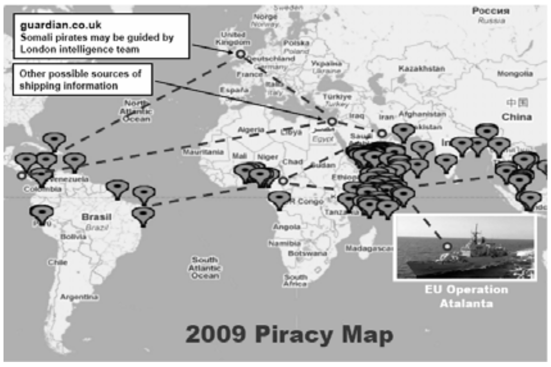
guardian.co.uk
Somali pirates may be guided by London intelligence team
Other possible sources of shipping information
EU Operation Atalanta
2009 Piracy Map

Figure 8. Global piracy fight.

Other applications. These are presented, discussed and published at numerous world events--from philosophy [8] to information technologies [1] to artificial life and robotics [9] to sensor networks [10] to crisis management [11] to defense [12-14]. Some of these researched applications are shown in Fig. 9.

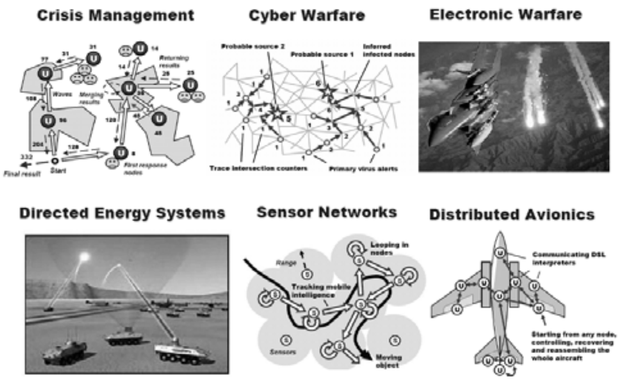

Figure 9. Other researched applications.

7 Conclusions

A distributed processing and control model and technology has been discussed, allowing us to obtain integral albeit fully distributed systems with artificial life features. These systems are capable of self-reshaping at runtime, changing networked structures, and adapting to unknown environments. The approach is based on holistic and gestalt philosophical principles, where the whole is first, greater than parts, and the parts are treated in the context of the whole rather than vice versa, as usual.

The approach, challenging conventional atomistic and agent-based philosophies in the system design and management, puts the artificial life concept, empowered with advanced distributed robotics, to the forefront of fight with numerous world disasters and crises.

Providing smooth transition from simulated to live solutions, with the watershed gradually shifting from the former to the latter, it can also support *a unified conversion from manned to fully unmanned advanced systems* within the same organizational concept.

Acknowledgments. Special thanks to the Alexander von Humboldt Foundation (AvH) in Germany and the Japanese Society for the Promotion of Science (JSPS) who supported (both morally and financially) these ideas, research, implementation, and related international cooperation throughout the years of their development.

References

[1] Sapaty, P. S., "Meeting the World Challenges: From Philosophy to Information Technology to Applications", Keynote lecture, Proc. 6th International Conference in Control, Automation and Robotics ICINCO 2009, Vol.1, Milan, Italy, IS-31--IS-43, 2009.
[2] Sapaty, P., "The Over-operability Organization of Distributed Dynamic Systems for Asymmetric Operations", Proc. IMA Conference on Mathematics in Defence, Farnborough, UK, 2009.
[3] Sapaty, P., "Gestalt-based Integrity of Distributed Networked Systems", SPIE Europe Security + Defence, bcc Berliner Congress Centre, Berlin Germany (2009).
[4] Sapaty, P. S., Mobile Processing in Distributed and Open Environments, John Wiley & Sons, 1999.
[5] Sapaty, P. S., Ruling Distributed Dynamic Worlds, John Wiley & Sons, New York, 2005.
[6] Sapaty, P., A distributed Processing System, European patent No. 0389655, European Patent Office, 1993.
[7] Sapaty, P. S., "Providing Spatial Integrity for Distributed Unmanned Systems", Proc. 6th International Conference in Control, Automation and Robotics ICINCO 2009. Milan, Italy, 2009.
[8] P. Sapaty, "Gestalt-Based Ideology and Technology for Spatial Control of Distributed Dynamic Systems", International Gestalt Theory Congress, 16th Scientific Convention of the GTA, University of Osnabrück, Germany, March 26 - 29, 2009.
[9] P. Sapaty, K.-D. Kuhnert, M. Sugisaka, R. Finkelstein, "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, Feb. 5-7, 2009.
[10] Sapaty, P., "Intelligent Management of Distributed Sensor Networks", In Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense VI, Proc. of SPIE, Vol. 6538, 653812, 2007.
[11] Sapaty, P., Sugisaka, M., Finkelstein, R., Delgado-Frias, J., Mirenkov, N., "Advanced IT Support of Crisis Relief Missions", Journal of Emergency Management, Vol.4, No.4, 2006.
[12] Sapaty, P., Morozov, A., Sugisaka, M., "DEW in a Network Enabled Environment", Proc. International conference Directed Energy Weapons, Le Meridien Piccadilly, London, UK, 2007.
[13] Sapaty, P., "Grasping the Whole by Spatial Intelligence: A higher level for Distributed Avionics", Proc. International Conference Military Avionics 2008, Cafe Royal, London, UK, 2008.
[14] Sapaty, P., "Distributed Capability for Battlespace Dominance", In Electronic Warfare 2009 Conference & Exhibition, Novotel London West Hotel & Conference Center, London, 2009.