Grasping the Distributed Entirety

Peter Sapaty [1] Masanori Sugisaka [2] Nikolay Mirenkov [1] Minetada Osano [1] Robert Finkelstein [3]

[1] University of Aizu
Aizu-Wakamatsu, Fukushima-ken 965-8580 Japan
+81-242-37-2604, +81-242-37-2553 (fax)
{sapaty, nikmir, osano}@u-aizu.ac.jp

[2] Oita University
700 Oaza Dannoharu, 870-1192 Japan
+81-97-554-7831, +81-97-554-7841 (fax)
msugi@cc.oita-u.ac.jp

[3] Robotic Technology Inc.
11424 Palatine Drive, Potomac, Maryland 20854
+1-301-983-4194, +1-301-983-3921 (fax)
RobertFinkelstein@compuserve.com

Abstract. A flexible WAVE-WP ideology and technology, aimed at creation of new and penetration into other distributed systems of different natures and at different levels, are described. Supported by a high-level spatial programming language, the technology allows us to grasp large distributed systems as a whole, study and manage their behavior, and direct evolution. Programming examples combining swarm behavior of an unmanned group with a hierarchical command and control are presented, providing integrity, flexibility, and openness within the same solution. Other application areas of the paradigm are discussed too.

Keywords: parallel distributed processing, spatial navigation, WAVE-WP language, cooperative robotics, swarm behavior, distributed command and control, open systems, over-operability.

1 Introduction

To understand the mental state of a handicapped person, of what the life and soul really mean, problems of economy and ecology, or how to win market or battlefield, we must consider the system as a whole – not just as a collection of parts. The situation complicates dramatically if the systems are dynamic and open, spread over territories, comprise unsafe or changing components, and cannot be observed in their entirety from a single point.

Usually the whole can be comprehended and decisions made only by a single human brain -- that is why we still have queens, kings, presidents, prime-ministers and commanders-in-chief. However, due to physical limitations, the brain cannot perceive the entire system in detail, and uses simplifications. Hierarchical systems are common too, where decisions on different levels are made by single brains on restricted information, with averaging and abstracting on higher levels. This is usually inherited by automatic systems using computers and computer networks mimicking human hierarchies. The hierarchical systems with predetermined partitioning onto levels often become static and clumsy, and may not operate efficiently in changeable environments.

In this paper, as an alternative to existing manned or unmanned control systems, we are describing flexible WAVE-WP (or World Processing) ideology and technology aimed at creating new or penetrating other systems and their organizations. Supported by a high-level spatial programming language, WAVE-WP allows us to grasp large systems as a whole, study their behavior and direct evolution in the way required. On the implementation layer, the technology widely uses self-spreading mobile cooperative program code dynamically covering and matching distributed systems. This often allows us to get solutions orders of magnitude more compact than by other approaches. The internal system organization, including partitioning into components and specific command and control, can be a function of the environment and the mission scenario in WAVE-WP; it may change at runtime while preserving the overall system operability and goal orientation.

Practical applications (with code examples) will be presented in relation to collective behavior of a robotic group, which integrates a swarm-based movement with hierarchical command and control.

The proposed approach allows us to grasp and manage the integrity and wholeness of large dynamic systems in highly parallel and fully distributed mode, often contrary to the human experience, also to the systems modeling human behavior, with a real potential to outperform them.

2 The WAVE-WP Spatial Automaton

The WAVE-WP automaton [1,2,3] effectively inherits the integrity of traditional sequential programming over localized memory, but for working now with the real distributed world, while allowing its parallel navigation in an active pattern flow and matching mode, as a single spatial process. The automaton may start from any point of the distributed world to be controlled, dynamically covering its parts or

the whole, and mounting of a variety of parallel and distributed knowledge and control infrastructures. Implanting distributed "soul" into the system organization, the automaton increases system's integrity, capability of pursuing local and global goals, assessing distributed situations, making autonomous decisions, and recovering from indiscriminate damages. Many spatially cooperating or competing parallel WAVE-WP automata may evolve on the same system body serving, say, as deliberative, reactive, and/or reflective spatial processes.

One of the distinguishing features of WAVE-WP is the representation of distributed worlds it operates in. *Physical world* (or PW) is continuous and infinite in WAVE-WP. Existing at any its point, and possibly performing a job, is considered as residing in a *node* having physical coordinates. Such a node, reflecting only occupancy at the point, vanishes with the termination of all occupancies in it. *Virtual world* (or VW) is discrete and interlinked in WAVE-WP, being represented similar to WAVE [4] by a distributed Knowledge Network (KN). Its persistent nodes may contain established concepts or facts, and (also persistent) links (oriented and non-oriented, connecting the nodes) may reflect different relations between the nodes.

The same model can also operate with the *united (or PVW) world*, in which any element may have features of the both worlds. A variety of effective access mechanisms to nodes, links and their groups, say, by physical coordinates, electronic addresses, by names, via traversing links, etc. (classified as *tunnel* and *surface* navigation) abound in the model, using both selective and broadcasting access modes.

Solutions of any problems in this formalized world in WAVE-WP are represented as its coordinated parallel *navigation* (or *exploration*, *invasion*, *grasping*, *coverage*, *flooding*, *conquest*, etc.) by some higher-level forces, or *waves* [2,4] These bring local operations, control and transitional data directly into the needed points of the world. The obtained results, together with the same or other operations may invade the other world parts, and so on.

During the world navigation, which may be loose and free or strictly (both hierarchically and horizontally) controlled, waves can modify the very world they evolve in and move through, as well as create it from scratch (including any distributed structures and topologies). Waves may also settle persistent cooperative processes in world's different points, subsequently influencing its further development and evolution in the way required.

3 WAVE-WP Language

The system language expressing full details of this new control automaton has been developed. Having recursive space-navigating and space-penetrating nature, it can operate with both information and physical matter. The language can also be used as a traditional one, so no integration with (and/or interfaces to) other programming models and systems may be needed for solving complex distributed knowledge processing and control problems.

Very compact syntax of the language, as shown in Fig.1, see also [1,2], makes it particularly suitable for direct interpretation in distributed environments, being supported by effective program code mobility in computer networks.

```
wave      →  { advance ; }
advance   →  { move , }
move      →  constant | variable | { move act } |
             [ rule] ( wave )
constant  →  information | physical-matter
variable  →  nodal | frontal | environmental
act       →  flow-act | fusion-act
rule      →  forward-rule | echo-rule
```

Fig.1. Syntax of WAVE-WP language.

In this description, braces set up zero or more repetitions of a construct with a delimiter at its right; square brackets identify an optional construct; semicolon allows for sequential, while comma for parallel invocation of program parts; and parentheses are used for structuring of WAVE-WP programs (or *waves*). Successive program parts, or *advances*, develop from all nodes of the set of nodes reached (SNR) by the previous advance, whereas parallel or independent parts, *moves*, constituting the advances, develop from the same nodes, while splitting processes and adding their own SNRs to the resultant SNR of the advance.

Elementary *acts* represent data processing, hops in both physical and virtual spaces, and local control. *Rules* establish non-local constraints and contexts over space-evolving waves like, for example, the ability to create networks, also allowing WAVE-WP to be used as a conventional language. *Variables*, called *spatial* (as being dynamically scattered in space), can be of the three types: *nodal,* associated with virtual or physical nodes and shared by different waves; *frontal,* propagating with waves as their sole property; and *environmental,* accessing elements of internal and external environments navigated by waves.

This recursive navigational structure of the language allows us to express highly parallel and fully decentralized, albeit strongly controlled and coordinated, operations in distributed worlds in a most compact way – in the form of integral space processing and transformation formulae. These resemble data processing expressions of traditional programming languages, but can now *operate in and process the whole distributed world*.

4 Implementation Basics

On the implementation layer, the automaton widely uses high-level mobile cooperative program code self-spreading and replicating in networks, and can be easily implemented on any existing software or hardware platform. As the automaton can describe direct movement and processing in physical world, its implementation may need to involve multiple mobile hardware, with or without human participation. A network of (hardware or software) communicating WAVE-WP language interpreters (or WIs), which can be mobile if installed in manned or unmanned vehicles, should be embedded into the distributed world to be controlled, in most sensitive points, see Fig. 2.

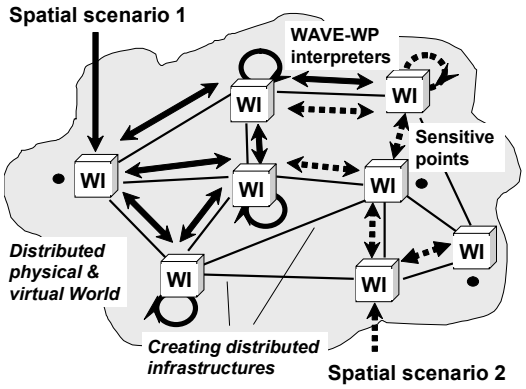

Fig.2. A network of WAVE-WP interpreters.

During the spatial execution of system scenarios in WAVE-WP, individual interpreters can make local information and physical matter processing, as well as physical movement in space. They can also partition, modify and replicate program code, sending it to other interpreters (along with local transitional data), dynamically forming *track-based distributed interpretation infrastructures*.

The automaton can exploit other systems as computational and control resources too, with or without preliminary consent, i.e. in a (remotely controlled) virus-like mode. For example, many existing network attacks (especially DDoS) may be considered as a possible malicious, simplified and degenerated implementation of the automaton. WAVE-WP can also effectively integrate with other advanced systems managing distributed resources, like, for example, J-UCAS [5], within their orientation on rescue and crisis relief missions.

5 Programming Example: Integration of Swarm Behavior with Distributed Command and Control

Effective integration of swarm behavior [6] with strict command and control for robotic teams may help fulfill complex objectives and survive in dynamic environments.

Different forms of group behavior can coexist within WAVE-WP model of parallel and distributed processing and control. Written in a higher-level spatial language, with considerable code reduction, the combined system scenarios can start from any component, covering at runtime the whole system that may be dynamic and open.

A distributed organization has been programmed which, for example, makes all units of a scout platoon move in a swarm, but at the same time regularly redefining the topologically central unit and creating a fresh, most efficient, neighborhood-based hierarchical infrastructure covering all units. It fuses targets seen by the units, distributing them back to all units for an individual selection and impact.

We will consider here only some very simplified examples of the WAVE-WP code expressing different distributed operations of the platoon, along with their unity.

5.1 Swarm Movement

The initial, casual, distribution of mobile units in space may be as shown in Fig. 3. The simple program below activates all units in the group (say, unmanned scout platoon), making them move in a swarm.

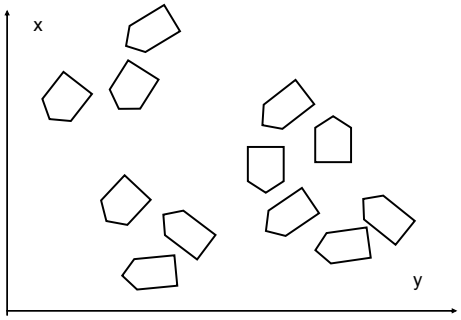

Fig. 3. Distributed group of mobile units: initial order.

Each unit randomly chooses next step within a given global direction, if the planned new position is not too close to other units, otherwise the next step is being redefined unless a suitable move is found. The program (let us call it ***swarm-move***) may start from any node, making all units fully autonomous and independent, and communicating only locally with other units:

```
Flimits = (dx0_dy-2, dx8_dy5);
Frange = r5; direct # all;
repeat(
  [Fshift = Flimits ? random;
   (direct # Fshift;
    direct ## Frange)== nil;
   WHERE += Fshift
  ];
)
```

A possible snapshot of the group during the work of this distributed program is shown in Fig. 4.

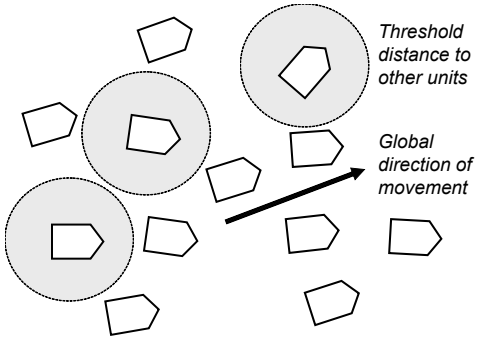

Fig. 4. Moving in a swarm with a threshold distance between units.

5.2 Finding Topologically Central Unit

Let us consider now the finding of a topologically central unit of the group for a certain moment of time (as the units may be constantly moving and changing positions to each other). Starting from *any* unit, this can be done by the following program (calling it ***find-center***):

```
Faver = average(direct # all; WHERE);
Ncenter =
  min(
   direct # all;
   (Faver, WHERE) ? distance _ ADDRESS
  ) : 2
```

5.3 Creating a Hierarchical Infrastructure

Let us create a hierarchical infrastructure starting from the central unit found and covering all other units. It can be most efficient if based on a physical neighborhood principle, with the next layer nodes lying from a current node, say, within a certain physical range. This can be accomplished by the following program (calling it ***infra-build***):

```
Frange = r20;
repeat(
 direct ## Frange;
 grasp(
  (all #) == nil; [create(-infra # BACK)]
 )
)
```

An example of such an infrastructure built over a swarm of Fig. 4 is shown in Fig. 5.

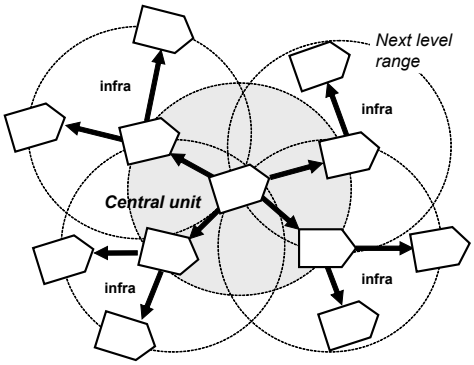

Fig. 5. Creating a neighborhood-based infrastructure from the most central unit.

Such an infrastructure can be effectively used for different purposes within the distributed command and control, with a possible one discussed below.

5.4 Hierarchical Fusion and Distribution of Targets

Starting from the same root node, the created hierarchical infrastructure can be repeatedly used for collecting targets discovered by the sensors of all units (ascending the hierarchy in parallel), with subsequent distribution of the collected list of targets back to all the units (descending the hierarchy, in parallel too, with the target list replicated in nodes). The units may be allowed to choose suitable targets individually, impacting them by the available means. All this can be achieved by the following spatially-recursive program (named ***collect-distribute***):

```
F1={(+infra#; ^F1), ?detect};
F2={(+infra#; ^F2), Fseen?selectImpact};
repeat([Fseen=(^F1); Fseen!=nil; ^F2])
```

The work of this program is explained in Fig. 6.

5.5 The Combined Scenario

All these programs can be effectively combined within a single scenario, with the center constantly migrating when units move in the swarm, and a new hierarchical infrastructure being rebuilt each time and frequently used for parallel and distributed vision and impacting targets.

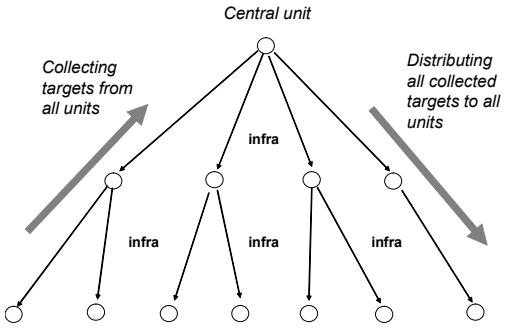

Fig. 6. Hierarchical fusion and distribution of targets.

To achieve this, we will also need removing of the previous infrastructure each time before creating of a new one from, possibly, a new central unit, which can be easily done by the following program (symbolically called ***infra-remove***):

```
direct # all; all #; LINK = nil
```

The united program, combining all the previous programs within a single distributed scenario, which can originally be injected from any mobile unit, will be as follows.

```
swarm-move,
repeat(
  find-center; direct # Ncenter;
  [infra-remove]; [infra-build];
  orparallel(
    collect-distribute,
    TIME += 300
  )
)
```

The named constituent programs, discussed before, can participate in it directly by their full texts, or by calls to them if represented as procedures. The program allows the regularly updated infrastructure to be used for fusing-distributing targets for some period of time (here 300 sec.), after which it finds the topologically central unit and new infrastructure from it again, after removing the previous infrastructure, and so on. All units continue moving in a swarm (with the details given before) independently of the infrastructure updating, targets collecting, distributing, and impacting processes.

As can be seen from the programming examples above, WAVE-WP is a completely different language from conventional terms, allowing us to express complex operations and control directly in distributed dynamic spaces, with programs often orders of magnitude more compact than in other known languages.

6 Other WAVE-WP Applications

The technology has numerous practical applications in other areas too, summarized in [1-4]. Some exemplary projects are as follows.

- *Distributed knowledge representation and processing.* Dynamically creating arbitrary knowledge networks in distributed spaces, which can be modified at runtime, WAVE-WP can implement any knowledge processing and control systems in parallel and fully distributed way. A program package had been developed for basic problems of the graph and network theory, where each graph node could reside on a separate computer.
- *Operating in physical world under the guidance of virtual world.* Operating in the unity of physical and virtual worlds, the WAVE-WP model can effectively investigate physical worlds and create their reflection in the form of distributed virtual worlds. The latter can guide further movement and search in the distributed PW, and so on.
- *Intelligent network management.* Integrating traditional network management tools and systems, and dynamically extracting higher-level knowledge from raw data via them, WAVE-WP establishes a higher, intelligent layer allowing us to analyze varying network topologies, regulate network load and redirect traffic in case of line failures or congestions. It also can be used for essentially new, universal and intelligent network protocols.
- *Advanced crisis reaction forces.* Smaller, dynamic armies, with dramatically increased mobility and lethality, represent nowadays the main direction in the development of advanced crisis reaction forces, which may effectively use multiple unmanned units. The WAVE-WP technology can quickly assemble a highly operational battle force from dissimilar (possibly casual) units, setting intelligent command and control infrastructures over them.
- *Distributed road and air traffic management.* Distributed computer networks working in WAVE-WP, covering the space to be controlled, can be efficiently used for both road and air traffic management. The model provides simultaneous tracking of multiple objects in PW by mobile intelligence spreading in VW, via computer networks.
- *Autonomous distributed cognitive systems.* While cognitive systems include reactive and deliberative processes, they also incorporate mechanisms for self-reflection and adaptive self-modification. The WAVE-WP paradigm allows for the description of interacting deliberative, reactive, and reflective processes on a semantic level, representing the whole mission rather than individual robots. This provides new, important degrees of freedom for autonomous robotic teams.
- *Distributed interactive simulation.* The technology allows for highly efficient, scalable distributed simulation of complex dynamic systems, like battlefields, in open computer networks. Due to full distribution of the simulated space and entities operating in it, there is no need to broadcast changes in terrain or positions of entities to other computers, as usual. Each entity operates in its own part of the simulated world, communicating locally with other entities. Entities can move freely through the simulated space (and between computers) if needed.
- *Intelligent global defense and security infrastructures.* WAVE-WP can also be used in a much broader scale, especially for the creation of intelligent international infrastructures widely using automated and fully automatic control and advanced robotics. The global system may effectively solve problems of distributed air defense, where multiple hostile objects penetrating the air space can be simultaneously discovered, chased, analyzed, and destroyed using computerized radar networks as a collective brain.

7 Conclusions

The WAVE-WP technology allows for a more rational and universal integration, management and simulation of large complex systems. This is being achieved by establishing a higher level of their vision and coordination, symbolically called "over-operability" [2] versus (and in supplement) to the traditional "interoperability".

Distributed system creation and coordination scenarios in WAVE-WP are often orders of magnitude simpler and more compact than usual, due to high level and spatial nature of the model and language.

This helps us to effectively grasp and manage large, dynamic and open systems and solutions in them as a whole, often avoiding tedious partitioning into pieces (agents) and setting their communication and synchronization.

These and other routines are effectively shifted to the automatic implementation by dynamic networks of WAVE-WP interpreters. Traditional software or hardware agents may have sense within this approach only when required, during the spatial development of parallel system scenarios.

A detailed description of the WAVE-WP model and its extended applications can soon be available from [7].

References

[1] P. Sapaty, N. Mirenkov, M. Sugisaka, and M. Osano, "Distributed Artificial Life Using World Processing Technology", Proc. of the Fifth Int. Conference on Human and Computer (HC-2004), September 1-3, 2004, The University of Aizu, Japan, 2004.

[2] P. S. Sapaty, "Over-Operability in Distributed Simulation and Control", The MSIAC's M&S Journal Online, Winter 2002 Issue, Volume 4, No. 2, Alexandria, VA, http://www.msiac.dmso.mil/journal/WI03/sap42_1.html.

[3] P. Sapaty, M. Sugisaka, "WAVE-WP (World Processing) Technology", Proc. 1st International Conference on Informatics in Control, Automation and Robotics, 25-28 August 2004, Setubal, Portugal.

[4] P. S. Sapaty, "Mobile Processing in Distributed and Open Environments", John Wiley & Sons, ISBN: 0471195723, New York, February 1999, 436 p.

[5] "Joint Unmanned Combat Air Systems (J-UCAS)", www.darpa.mil.

[6] R. Finkelstein, "Swarm Intelligence: Application to the 4/DRCS and the Scout Platoon Mission", White Paper, Robotic Technology Inc., USA, 2004.

[7] P. S. Sapaty, "Ruling Distributed Dynamic Worlds", John Wiley & Sons, ISBN: 0471655759, New York, June 2005, 256p., www.wiley.com.