Emergent Engineering for the Management of Complex Situations

René Doursat
Institut des Systèmes Complexes, CNRS
and CREA, Ecole Polytechnique
57-59, rue Lhomond
75005 Paris, France
+33 1 42 17 09 99
http://doursat.free.fr

Mihaela Ulieru
Canada Research Chair
Director, Adaptive Risk Management (ARM Lab)
Faculty of Computer Science
University of New Brunswick
+1 (506) 458-7277
http://www.cs.unb.ca/~ulieru

ABSTRACT
Ubiquitous computing and communication environments connect systems and people in unprecedented ways, but also fundamentally challenge the mindset of traditional systems engineering. Complex techno-social systems exhibit spontaneous self-organization properties, based on decentralized interactions among a multitude of agents, that have preceded our ability as human designers to fully comprehend and control them. This should prompt us to steer away from managing details and, instead, focus on establishing the generic conditions for systems to *develop* and *evolve* under our guidance. In alignment with this paradigm shift we propose a methodological framework termed *emergent engineering* for deploying large-scale "eNetwork" systems, and illustrate it with self-organized security (SOS) scenarios. It involves an abstract model of *programmable* network self-construction in which nodes execute the same code, yet differentiate according to position. We illustrate these principles on a future application to SOS pointing to how this could lead to a new type of *controllable* self-organization, able to dynamically co-evolve the system with its environment.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *network topology, wireless networks.*

General Terms
Algorithms, Management, Design, Human Factors, Theory.

Keywords
Complex Systems, Emergent Engineering, Self-Organization, Dynamical Networks, Cyber-Physical Ecosystems, Security, Co-evolution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ACM-ICST Autonomics'08, September 23–25, 2008, Turin, Italy.

1. RETHINKING ENGINEERED SYSTEMS AS COMPLEX SYSTEMS
Information and communication technologies (ICT) are pervasive in today's world, making it a densely connected "eNetwork" at all scales, locally and globally. Fueled by strikingly rapid technological advances, wireless networks and nanomachines begin to blend the virtual and the physical worlds in creative ways. They merge single devices, departments, or enterprises into larger and more complex infrastructures that animate a great variety of "Cyber-Physical Ecosystems" (CPE) in many domains, e.g., industrial, artistic, educational, scientific. Every new ICT discovery and its application brings us closer to a global collaborative ecosystem that becomes more and more hybrid, inclusive, and virtually unlimited in its functionality.

This poses a new grand challenge to engineers: How can we harness and craft this "nervous system" in such a way that it evolves and adapts seamlessly to demanding circumstances—able to not only survive the unexpected, but even thrive on it? Every new wireless connection, sensor network, or router added to the communication infrastructure eventually comes at a price, reflected by an overall measure of "unmanageability" of the surrounding complexity. We continuously endow new artifacts with greater capabilities to infer our context and anticipate our desires and needs with increasing accuracy. They add more and more services to an already fast growing ambient intelligent environment. However, every such artifact improving our comfort and quality of life also invites unanticipated difficulties, which challenge our deep-seated way of thinking about how to design and manage systems that support human activities.

Imagine a self-configuring manufacturing plant [1], a self-stabilizing energy grid [2], or a self-deploying emergency taskforce [3] all relying on a myriad of mobile devices, software agents and human users building their own eNetwork on the sole basis of local rules and peer-to-peer communication. The past few years have seen a remarkable increase in research activities across many disciplines to realize systems that exhibit true *autonomy*. However, while we have just begun to work on conceptual and technological advances toward this goal, CPEs are already living a life of their own. Large-scale eNetworked systems have spontaneously grown and reached unanticipated levels of complexity, beyond the boundaries of the disciplines that conceived their components in the first place. The march toward

decentralization and self-organization has preceded our ability to fully master these phenomena.

If we can catch up and win the race, the stakes are high: *emergent architectures* based on decentralized automation, whether 3-D devices or *n*-D techno-social webs, promise to be the new paradigm of eNetworked systems engineering and control [4]. Traditionally, the role of an engineer is that of an active designer: she or he creates organization, system, or software blueprints with the end in mind. The system and its performance requirements are defined by hierarchical, top-down, linear thinking. By contrast, through their unplanned emergent behavior, eNetworked CPEs fundamentally defy this conceptualization approach. They push the engineer to become rather a passive observer of a "self-design" process, in which the system's organizational structure results from bottom-up interactions among a multitude of elementary components [5], [6]. Yet, *design-by-emergence* could be in fact more successful than classical design-by-planning as it *enables* the system to meet unexpected situations and dynamic goals in an anticipatory manner.

This work proposes an exploration beyond the top-down perspective on systems design and control imposing order exogenously. Can we find other ways than telling each element of the system what to do at every step through predetermined strategies, assuming that all possible situations the system might confront are knowable in advance? One major characteristic of the new generation of eNetworked systems is that, given their very nature and scale, *they cannot be a priori defined* but should rather emerge from the interactions between individual devices and users. We think that such systems could be managed by "riding the wave" of their complexity, instead of fighting it, i.e., by encouraging their endogenous growth, function, stability, and adaptation in a bottom-up fashion. Moving farther away from the system's profuse details, this approach looks rather for the *generic conditions* that will produce those details without dictating them.

We term here this approach *emergent engineering*: it shifts the role of humans toward machines from micromanagers to lawmakers. Inspired by evolution, it is less about direct design than developmental and evolutionary "meta-design" [7]-[9].

At the core of this quest, however, lie paradoxical questions: Can autonomy be planned? Can decentralization be controlled? Can evolution be designed? Can we expect specific characteristics from systems that we otherwise want to let free to assemble and possibly invent themselves? The resolution of these apparently inconsistent objectives will reside in the *change of scale* at which engineering operates. The generic conditions or "laws" of self-organization mentioned above should be instilled in every element of the system. No longer a separate global entity, control is broken down and distributed locally everywhere.

2. PRINCIPLES OF EMERGENT ENGINEERING

For clarity, we attempt to illustrate the major principles of this approach with a more concrete example involving emergency response to acute and developing disasters. In this type of scenario, called Self-Organizing Security (SOS) [10], several first responders come together in a collaborative endeavor and form joint teams, or "SOS networks", to contain and manage a crisis situation. These teams are dynamic, short-lived *meta-organizations* deployed on the fly from units belonging to different organizations such as military forces, police, firefighters, paramedics, or non-governmental organizations. SOS consists of networks of agents interacting intensely with each other and generating a collective behavior that co-evolves with the environmental dynamics.

Generally, this imposes certain structural and behavioral constraints on the network of agents: obviously, it should not act freely or randomly, but according to a high-level strategy, or set of "policies" (called *overall rules of the network* in [11]). To enable a response adapted to the crisis dynamics, these high-level policies have to materialize into concrete action plans, which are *compiled down into local rules*, or "protocols", and broadcast on the fly to all the agents involved in the complex situation at hand. Agents perform individual actions in response to both the local rules of the particular plan they have received, and the local events of the particular situation they are faced with. The overall impact of the individual agents is to create a *collective network behavior* (emergence), which in turn influences the agents (immergence). The key issue when deploying emergency operations in an SOS system is to find the right balance between individual protocols and network policies in order to achieve the best possible collective meta-organizational behavior.

2.1 Architecting from the Bottom-Up Without an Architect: Introducing Self-Organization into Engineering

As a prime example of *complex systems*, hybrid techno-social eNetworks are made of a large number of agents that interact locally and produce diverse types of emergent collective behavior. It is useful to examine the commonalities of this multi-agent framework [12] across different domains at an abstract level. It generally consists of

- a set of micro-instructions or *rules* on how to
  - *search* and *connect* to other agents,
  - *interact* with them over these connections,
  - *change* one's internal state and rules, and
  - carry out some specialized *function*;
- rules act upon an array of *variables* of two types:
  - internal *developmental* variables (dedicated to building the system—the architecting part),
  - internal *functional* variables (dedicated to making the system carry out tasks—the control part);
- the rules can also be modulated by *parameters*
- that can *evolve* over time, according to
- a global *fitness* that the system is exhibiting with respect to its function in the environment (co-evolution).

To be able to achieve this open-ended status, enabling continuous adaptation to the unexpected, individual agent rules cannot deterministically dictate what an agent must do, but rather provide *guidelines* about how to react to environmental stimuli and other agents' actions. Some rules also incite the agents to behave proactively and start communication or function of their own initiative. In all cases, agents only have a limited knowledge of their surroundings and no view of the global picture. With this, two different mechanisms of agent self-assembly can be envisioned:

- *Predefined* agents that reposition themselves in the network: these agents are looking for other specialized agents to connect and cooperate with [3]. This is the case of the SOS network alliance [10] involving various organizations with functional roles and protocols specifying how these roles are carried out.
- *Prepositioned* agents that redefine themselves under induction from other agents: these agents already form a structure through which they can modify each other by exchanging rules and variables. This is the case, for example, of biological cells [7]-[9], or computer hosts and routers in a distributed energy web application [5]. In SOS networks, this would also correspond to a firefighter providing paramedical care to a victim, etc.

Agent *proximity* is defined by the topology of the graph of interactions (agents are considered neighbors if they share a link), itself possibly depending on Euclidean spatial distance. Typical examples of spatial systems based on shortest-distance links are sensor networks and energy grids. For SOS networks, links are function of the space where the crisis occurs (e.g., an Olympic stadium). Examples of nonspatial systems are distributed software components or business collaboration networks. Many CPEs [6] inherently have a dual spatial/nonspatial nature, as they often include a physical infrastructure at a "lower" communication level, with a virtual overlay network at a "higher" application level [3], [13].

In summary, the complex systems viewpoint on autonomy introduces a distinction between the rules at the *microscopic* level of the agents, or "genotype", and the resulting overall *macroscopic* behavior of the collectivity, or "phenotype". In the SOS example, the former corresponds to the agents' individual protocols creating the appropriate action plans, while the latter corresponds to the network mediating global policies across all organizations that are hosts for the deployed individual agents. In essence, emergent engineering consists of indirectly designing the genotype (or letting it evolve toward an appropriate phenotype), rather than directly building the phenotype. This also mirrors the *developmental* dynamics of a system, as a prerequisite to evolution [14]. It envisions the architecture and the function as (rapid) emergent outcomes of the elements' capabilities of self-assembling and disassembling. This is the case with SOS network participants who come together to form barriers around attacks and contain the crisis at hand, then are sent back to rest after the situation is resolved.

2.2 Controlling Complexity without a Controller: Reintroducing a Program into Self-Organization

Because it promotes the transfer of decision-making to a swarm of autonomous agents, the new responsibility of complex systems engineering is also to establish the proper rules to *regain control* of these agents at the microlevel (individual agent protocols), relative to a desired behavior of the system at the macrolevel (SOS policies and action plans). Many families of self-organized systems based on simple rules exhibit a variety of complex orderly states, whether in pattern formation (e.g., [15]), spatially explicit evolution (e.g., [16]), neuronal synchronization (e.g., [17]), swarm intelligence (e.g., [18]), collective motion (e.g., [19])

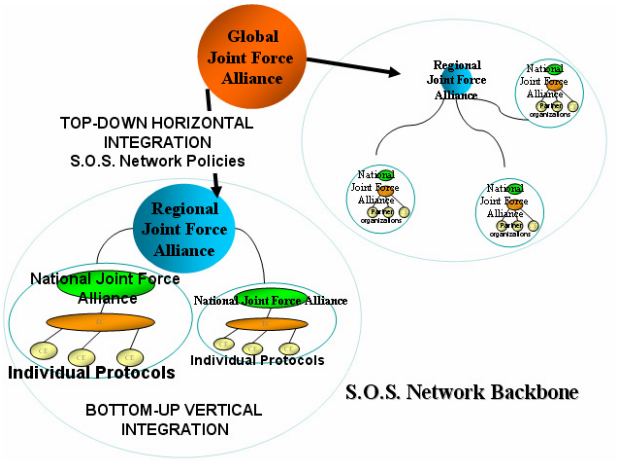

Figure 1. SOS as a "nervous system" controlling the crisis.

or statistical complex networks (e.g., [20]). However, a closer look reveals that this "complexity" often translates into emerging patterns that are either freely random or entirely determined by boundary conditions. What is much more challenging is how to harness and *guide* spontaneous complexity to form non-trivial *intrinsic structures* adapted to desired outcomes. Such guidance would require (a) preparing the generic mechanisms and rules acting at the microlevel that are capable of generating *reliable* (as opposed to random and unpredictable) self-assembly processes; and (b) steering these processes toward desired shapes and functions manifested at the macrolevel, by allowing the spontaneous generation of innovations through recombination and evolution.

2.2.1 C2 for Complex Systems: Growth by Positive Feedback, Control by Negative Feedback

The traditional view of control engineering is that the controller is a separate entity that monitors and affects the main system, generally by feedback from its output variables onto its input variables [21]. In the paradigm shift toward emergent engineering, this system/controller pair becomes fragmented into a myriad of micro-system/micro-controller pairs (the agents and their individual rules). In most classical examples of complex systems, agent rules can be decomposed into two parts: (1) a positive feedback that amplifies small local fluctuations (micro-system), and (2) a negative feedback that dampens or corrects the agent's response, and tunes its behavior more finely (micro-controller). At the emergent level, the tendency of the former is to *create* new mesoscopic or macroscopic structures, while the latter tends to *stabilize* them [13].

The collective dynamics resulting from the tensions between the overall goal of the network and the individual agent actions can be represented, in view of the cybernetics school [22], as an overarching "Command & Control" (C2, i.e., feed-forward & feedback) backbone (Figure 1, from [10]). It acts as a "nervous system" regulating individual behaviors by adapting the network policies in order to collectively achieve a desired action plan (e.g., containing the crisis at hand). C2 can be built into the architectural requirements of the SOS network by determining the local logic of the individual components (protocols) as well as

their interactions bound by the network (organizational policies and governance rules).

Insect colonies provide examples of *positive feedback* [18]: an ant deposits more pheromone where there is already pheromone; a termite brings more pellets of soil where there is already a heap of soil. In human societies, too, shoppers like buying lower priced products, traders like buying stock that goes up; the media likes to talk about what is currently talked about in the media. Starting from small initial fluctuations, positive-feedback agent behavior generally creates a single large homogeneous cluster characterized by some increasing quantity (concentration, density, price, etc.). More interesting structures can then emerge and be stabilized by adding *negative feedback*. For example, in collective motion [19], a bird follows the flock by continuously readjusting its speed and orientation. Each agent corrects small differences by sensing neighboring agents, and the collectivity converges, albeit temporarily, to a stable trajectory (the appropriate action plan).

Relatively complex behavior can therefore result from a balance between genotype (the simple agent-based rules that encode positive feedback) and phenotype (the overall rules of the system providing the negative feedback resulting in the adaptive action plans). This is done by adjusting the individual behavior to the overall goal of the network. Similar negative feedback is found in a *holonic enterprise* [23] that balances the autonomy of individual agents with the need to cooperate and compete to achieve the overall goal of the system.

In an SOS network, for example, via "orchestration and choreography" [24] of the processes that run across the multi-agent system, latest Web 2.0 advances can be used to balance the individual protocols at the agent level (positive feedback of the genotype) with the overall network policies (negative feedback of the phenotype). The C2 coordination mechanism separates process from execution, acting in the background according to the governance rules of the overall SOS network. Meanwhile, the individuals, coming together from their respective military and civilian units, follow their own specific protocols and aim toward a common goal-seeking, self-organizing swarm. It is this balance between bottom-up protocols and top-down policies that enables the flexible co-evolution of the SOS network with the crisis dynamics. It must keep the crisis under control, while also addressing new events and issues that come up on the path to resolution. To contribute to the overarching SOS network goal, the individual agents must be able to absorb the shared overall network policies by translating them into useful executable actions. Strategies for balancing individual holon autonomy with overall network goals have been proposed for emergent virtual organizations [3], [10].

2.2.2 Controlling Self-Organization: Crafting the "DNA" of eNetworks

So far, the only emergent *and* nontrivial architectures that we see around us are living organisms. This is because biological agents (cells) carry a set of rules (DNA) that endows them with a repertoire of non-trivial behaviours. Cells do not randomly mix but proactively position themselves. Regions of genetic expression are not randomly distributed but highly regulated in number and position. An organism's shape dynamically unfolds in time, on the basis of calculations and decisions carried out by each cell at every time step. The sophistication of the organism is a reflection of the relative sophistication of the DNA, and its associated cell dynamics, compared to elementary volumes of inert physical matter that only obey simple attraction/repulsion dynamics.

By analogy, our approach considers genetic-like regulation at the agent level (the eNetwork's "DNA" [6]) to harness large scale eNetworked techno-social systems. The new challenge is to reintroduce a certain dosage of *programmability* inside free self-organization, in the form of a developmental genotype. This important notion has been studied in particular in the field of artificial development [25], [7]-[9] and amorphous computing [26], [27]. Simple action-reaction rules are not sufficient in themselves to create complex architectures, such as growing adequate barriers to contain threats or attacks or guiding a crowd toward safety in the SOS case. More elaborate genotype logic is needed. In fact, the richer the information carried by the genotype (individual agent behaviours), the richer the variety of the overall phenotype (range of action plans that can emerge and be dynamically deployed). This is because a sophisticated genotype opens the door to agent *differentiation* via *positional information* – essential properties which enable programmability and evolution by combinations and recombinations of diverse agents into modules and hierarchical constructions. These properties are essential for enabling the SOS network's continuous adaptation to the dynamics of a chaotic situation, otherwise impossible to manage. We present in Part 3 an abstract model of autonomous network construction and dynamics in which nodes execute the same program in parallel however develop into different types according to their (limited) positional awareness.

2.2.3 Designing Evolution without a Designer: Varying the Rules and Selecting the System

The ultimate challenge of our enterprise is to understand how a complex eNetwork such as SOS can *evolve* on the longer time-scale, i.e., how individual agent protocols can be *varied* and the appropriate network policies *selected* to deploy the desired action plans.

First, in order to be selected, the network's functional success needs to be measured. After reaching structural maturation on a short deployment time scale, i.e., when the joint teams are deployed in response to a particular event, the SOS network should switch the bulk of its activity from executing the *developmental part* of its genotype to executing the *functional part* of its genotype. The former corresponds to dynamic architecting, which *positions* the actors within the network so that they can best perform their activity in teams. The latter corresponds to the adaptive control obtained by *executing* their roles within the teams to realize the most effective action plans.

Second, once the basic "eNetwork DNA" parameters have been set to achieve the SOS network growth (architecture) and function (control), the remaining question is how to make the SOS network *co-evolve* with the developing crisis. This can be done by specifying how the genotype (individual agent rules) may vary and how the phenotype (overall network policies that enable the selection and deployment of appropriate action plans) may be selected. For example, gradual optimization can rely on a distance of *performance* to predefined goals, instead of network structure, allowing the most successful candidates to reproduce faster and mutate.

3. AN ABSTRACT MODEL OF SELF-MADE E-NETWORK

In this part, we present preliminary results from an original model of autonomous network construction and dynamics, while we show its possible application to the emergent engineering of SOS networks in Part 4. Nodes can represent human agents who carry personal digital assistant (PDA) devices with wireless connectivity. These devices execute the same program in parallel, but gradually differentiate according to (limited) positional awareness. The self-assembly program includes routines for the exchange of messages and the dynamical creation or removal of links. It relies on a combination of "ports" and internal state variables derived from discrete "gradients". Ports and gradients guide the new nodes to specific attachment locations in the developing network. As the network expands and node positions change, nodes adapt by switching different subsets of rules on or off—similar to gene activation/inhibition in biological DNA—thus triggering the growth of specific structures such as chains, lattices, and more complicated composite topologies.

3.1 Simple Chaining

Chains are the simplest self-assembling structures that can be realized with two ports, X and X', and two corresponding gradient values x and x' in each node (Figure 2). Ports can be "free" (not linked to other ports from other nodes) or "occupied" (linked), while free ports can be "open" (available for a connection) or "closed" (disabled). New nodes that just arrived in the system's space, or nodes that are not yet connected, have both ports open and gradients set to 0. A node i can create a link with another node j only through a pair of complementary open ports, here X and X', with one link per port. As soon as a new link is made, ports are occupied and gradients are immediately updated according to the following rules: (a) a free port always maintains its value at 0 (gradient source), and (b) x is sent out through occupied port X' to port X of the neighbor node with an increment of +1 (resp. x', X, X'). Discrete counter increments are also the method of choice for spreading positional information in amorphous and spatial computing systems, e.g., [9], [27]. A new agent can connect to any available port at random, including the most recent and oldest nodes of the chain (Figure 2a). The gradient counters keep track of the nodes' positions in the chain allowing, for example, to build chains of a fixed length n by closing any remaining open ports as soon as $x + x' = n - 1$.

Figure 2b is a step-by-step illustration of the gradient update routine after a new node (dashed) has connected to the system. In general, the new x value of a node i, denoted by $x_i(t+1)$, is set to $x_j(t) + 1$ if j is the neighbor attached to i_X (same with x' and $i_{X'}$). This ensures a natural propagation of gradient value corrections and converges after $O(n)$ time steps. Note that, in order to avoid inconsistencies, gradient update has to be much faster than node addition and connectivity changes. This difference of time scales is simulated here by alternating gradient update and link formation in two embedded loops: first, nodes are visited several times until gradient values have converged (fast inner loop); then, new nodes and links are added ("slower" outer loop).

Thus all nodes carry the same program (their genotype or "DNA"), which consists of three main routines: *gradient update* (G), *port management* (P), and *link creation* (L). The gradient update routine G is the generic code that provides nodes with the positional information they need to make further decisions. Figure 2c is an example of port management routine P, which contains the heart of the logic specific to a target structure. For example, in the case of a 5-node chain, it orders a node to shut its ports whenever $x + x' = 4$ (the "open" and "close" commands apply only to free ports, and are ignored on occupied ports). Routines G and P are executed by the nodes already involved in the network, and prepare the way for new nodes to execute L (Figure 2a). Routine L provides the generic logic that prompts new nodes to pick one of the open ports of the network at random to make a new connection.

3.2 Lattice Formation by Guided Attachment

With *two* pairs of ports, (X, X') and (Y, Y'), and two pairs of associated gradient variables, (x, x') and (y, y'), also set to 0 when the node is new, lattices can grow (Figure 3). Two nodes i and j

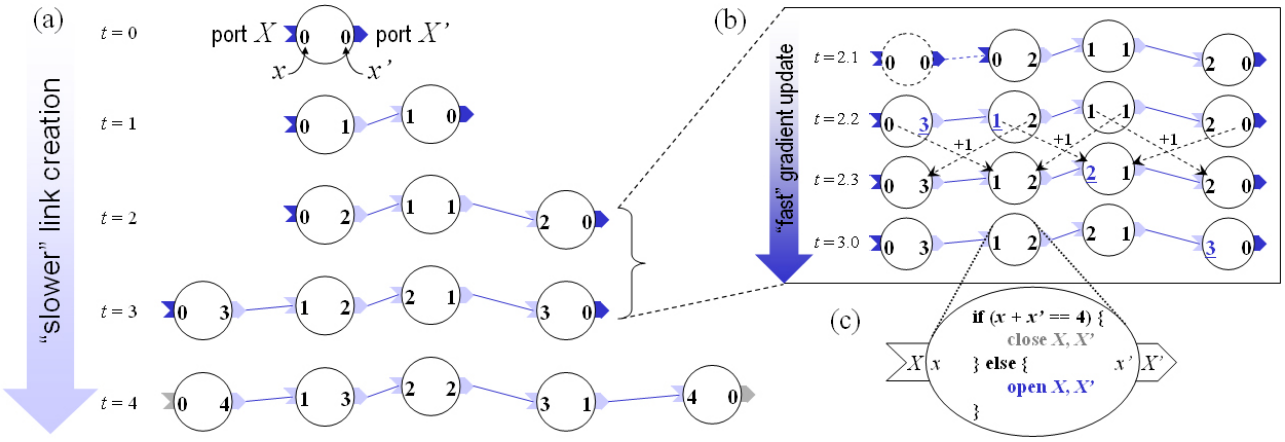

Figure 2: Self-assembly of a simple chain. (a) The five main steps leading to a 5-node chain. Through the *link creation* routine, incoming nodes attach to either open ports, X or X' (dark blue), of the forming chain. When a link is created, its ports become "occupied" (light blue) and gradient values are updated in all nodes (see b). When chain length is 5 (i.e., $x + x' = 4$), all open ports are closed (gray; see c). (b) Detailed substeps of the value-passing *gradient update* routine (see text). (c) *Port management* routine of the "DNA" program in each agent: ports close when length is 5.

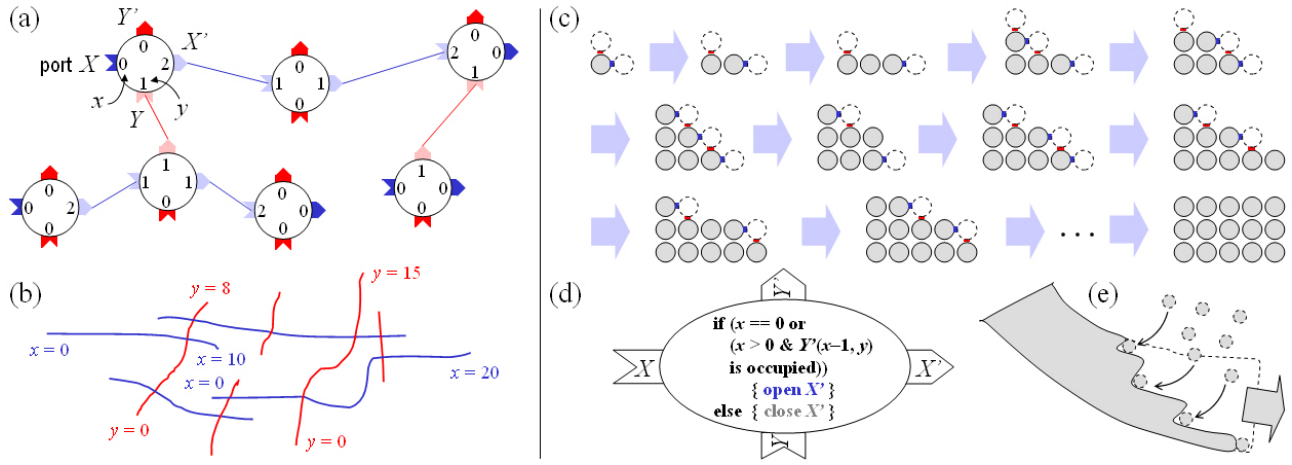

Figure 3: Self-assembly of a lattice. (a) Nodes have four ports, *X*, *X'*, *Y*, and *Y'*, and can form either *X*↔*X'* or *Y*↔*Y'* links. (b) Without any port management routine *P*, node chains (schematized by curved lines) form and intersect in a random manner. (c) Condensed view of an example of 5×3 lattice self-assembly in oderly "waves" of node attachment: the only available spots offered by open ports are internal "corners". (d) An excerpt of the *P* routine in every node (rules P_2 and P_3 explained in the text). (e) A generic illustration of lattice-building attachment waves.

can now form four possible links involving pairs of complementary ports (Figure 3a). If left without additional constraints, i.e., only routines *G* and *L*, but no *P*, the networking process will grow branches that criss-cross randomly, where each branch maintains its own gradient values along its length (Figure 3b). More orderly network, such as a regular square lattice of fixed size $n \times m$, can be programmed by endowing *P* with specific port-shutting commands that strictly regulate the pool of open ports at any time in the life of the structure (Figure 3d). Therefore, *P* guides node attachment toward a few locations of interest, similar to blinking beacons on a landing runway (Figure 3c). A new node can then randomly choose among one of these few locations, according to *L*. The general building principle of this example is that a row or column cannot be augmented by a new node if it leaves a hole in a nearby column or row. In essence, ports *X* and *Y* are permanently shut, while ports *X'* and *Y'* are opened only in the inner "corners" of the forming lattice (Figure 3c,e). For further implementation details, see [28].

3.3 Cluster Chains and Lattices

In biological development, the position and number of individual cells is very imprecise, while the structures and organs they form are reliably placed. Similarly, programmed network self-assembly could also be irregular at the microscopic level of the nodes, while retaining an orderly arrangement at the higher, "mesoscopic" levels of *clusters* of nodes. To introduce this element of *variability* and *redundancy* in the system, in addition to the fundamental *programmability* of an emerging structure, one can "thicken" chains and lattices by replacing single nodes with clusters (Figure 4). This can be done through one additional port, *C* (as in "cluster" or "clique") that allows multiple nodes with identical *x* and *y* gradient coordinates to form random connections with each other, thus cluster into families according to their gradient values. The *C* port represents an extra "nonlinear" dimension added to the single pair of ports (*X*, *X'*) of 1-D chains, or the two pairs of ports (*X*, *X'*), (*Y*, *Y'*) of 2-D lattices. Another new feature is that nodes are also allowed to make multiple connections per port, whether *X*, *Y* or *C* (Figure 4a). Thus, in the case of a chain, a new node has two possibilities of attachment: it can either *thicken* or *lengthen* the chain. Either it connects to an existing node through *C*, in which case it inherits the coordinates of that node's cluster, or it connects as before via the *X* or *X'* port, in which case it pioneers the creation of a new cluster at one end of the chain and all coordinates are updated according to the usual gradient dynamics. After their first link, new nodes may also establish a few supplementary connections through any of their ports, under the constraint of coordinate consistency (–1 and +1 via ports *X*, *Y* or *X'*, *Y'*, equal coordinates via port *C*).

Similar to cellular proliferation in morphogenetic tissues and organs, this proliferation of nodes in structured networks introduces redundancy and "failover" safety. Unlike single-node chains, the failure of one link in a cluster chain does not imply the failure of the whole structure. Yet, while relying on a fluctuating swarm of agents for its robustness, the *overall topology* of a programmed network is still not left to chance but narrowly guided by the genotype of the attachment rules *G*, *P* and *L* to grow desired structures.

3.4 Modular Structures by Local Gradients

More complicated structures can be developed by composing multiple chains and lattices. To allow the creation of *modules* with their own identities and local positional information, one can find again inspiration from biology, in particular the concepts of modularity and homology central in evo-devo [29]. Modules are similar to "limbs" that have distinct morphologies and geographies. This is modeled here by different coordinate systems based on tags *a*, *b*, *c*, etc. Gradient ports in one part of the system, e.g., a chain, are denoted by (X_a, X'_a), while ports in other branches will be (X_b, X'_b), (X_c, X'_c), and so on. Accordingly, routine *L* is amended so that links cannot be created between ports with different tags.

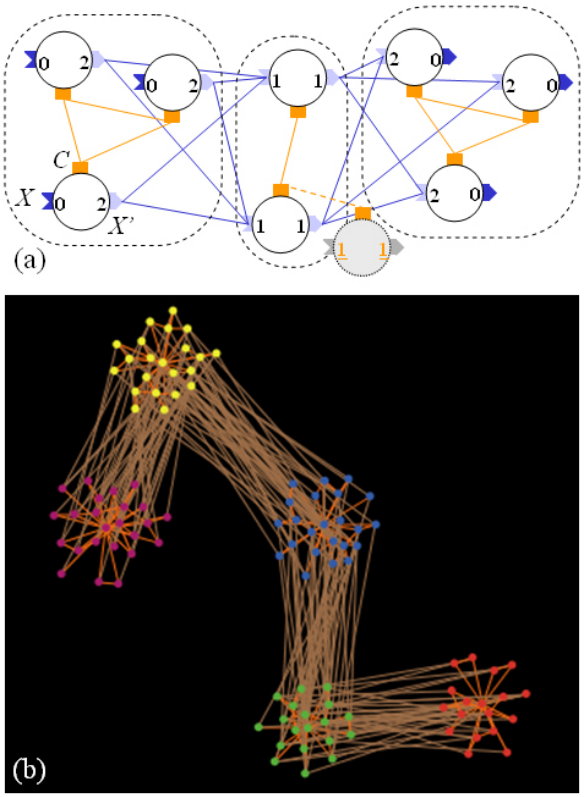

Figure 4: Cluster chain. (a) Detailed 3-cluster chain: internal (orange) links connect the *C* ports of nodes with same (*x*, *x'*) values, while (blue) links *between* clusters form the chain. A new node (gray) connecting through *C* adopts the cluster's values. (b) Simulation with 5 clusters and ~20 nodes/cluster.

In the elementary scenario of Figure 5, only the X'_a port is open in the beginning (Figure 5a-b). When the third node has attached, another pair of ports (X_b, X'_b) is created on that node and only port X'_b stays open (Figure 5c). This particular event is triggered by the positional information carried by the node: in this example, the *P* routine (Figure 5g) stipulates that when $x_a = 2$, a node must differentiate into a bifurcation node, i.e., create another pair of ports and their corresponding gradient variables. After this event, new nodes can attach to either open port, X'_a or X'_b (Figure 5d), i.e., either choose to first augment the original chain or its branch. As in the previous structure formations, however, the order of node attachment does not influence the final structure. New nodes carry an untagged pair of ports (X, X') and acquire the tag of their first contact. The same "stop-rule" of chains (Figure 2c) applies here when the *b* branch reaches length $n_b = 3$, i.e., $x_b + x'_b = 2$, closing the only open port X'_b (Figure 5e). Independently, another branch *c* grows from the fifth node of chain *a* and stops at $n_c = 4$ nodes, while chain *a* stops at $n_a = 6$ nodes.

Another example of branching structure based on lattices instead of chains is shown in Figure 6a: here, once a 3×3 lattice tagged *a* has finished self-assembling, its last node (x_a, x'_a, y_a, y'_a) = (2, 0, 2, 0) creates a new quartet of ports (X_b, X'_b, Y_b, Y'_b) and spins off a new 3×3 lattice tagged *b*, and so on.

Finally, whether based on 1-D chains or 2-D lattices, modular structures can also be "thickened" with clusters of nodes by adding a *C* port to each node, as explained in the previous section. An example of complex programmed network made of a branching chain (including a cycle) of clusters is shown in Figure 6b.

4. GUIDELINES FOR A CONCRETE SCENARIO

The previous section described abstract principles of self-made networks that have a purely endogenous ability to form precise configurations. It established new foundations for the emergence of non-random, programmable patterns exhibiting *intrinsic structures* that are neither repetitive nor imposed by the environment. Starting from these premises, the model must now be completed with other important features in order to be applicable to concrete problems such as the deployment of an SOS intervention taskforce. We identify and discuss here four notions: (1) physical space, (2) external events, (3) agent functionality, and (4) action plans.

4.1 Physical Space

As mentioned in the introduction, most real-world eNetworks such as SOS combine non-spatial graph topologies (e.g., connecting organizations and entities), with Euclidean graph topologies (e.g., connecting people and equipment on the field) at different degrees. The abstract mechanisms of programmed attachment described in Part 3 create purely non-spatial graphs that are displayed in 2-D figures only for convenient viewing. So far, nodes do not have any location and can potentially "see" all other nodes. The discrete positional information provided by the gradients is purely internal to the link structure.

On the other hand, if nodes represent agents and devices interacting in real space, the dynamics must be modified to take into account the effects of metric distance. In addition to the internal gradient values (x, x', y, y', ...) that they carry, nodes are now also labeled by a real vector $\mathbf{r} = (r_x, r_y, r_z)$. Space can then intervene at two levels: by limiting the scope of pre-attachment detection (nodes can connect only to nearby nodes, within a certain radius), and by giving a mechanical meaning to the nodes and links. For example nodes can be interpreted as electrically charged particles, and links as elastic springs. Through attraction/repulsion forces, connected nodes would tend to position themselves at an optimal distance from each other without colliding. These mechanisms are typically implemented in force-based layout algorithms [30] to make a graph unfold and self-arrange for best visualization of its structure. The same principles could also serve in real-world situations to guide users with mobile devices containing GPS or local positioning functions (e.g., embedded in "augmented-reality" glasses) toward specific precalculated locations.

In SOS intervention scenarios, an important part of the network of agents is essentially spatial, and most of the intervention efforts will be focused on *placing* people, units, vehicles, equipment where they are needed to form effective *functional patterns*. For example, during the evacuation of a stadium, space could be partitioned into different sectors organized around the nearest exits and the center of the field to direct the flow of the crowd

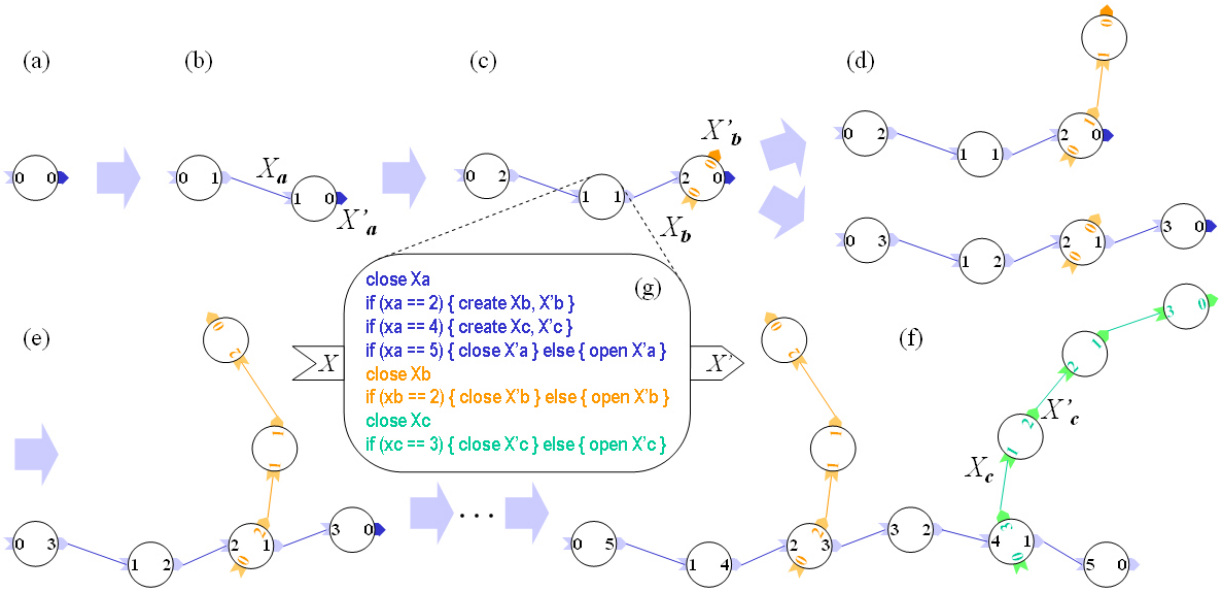

Figure 5: Branching scenario (see text). (a,b) Beginning of chain *a*. (c) Branch *b* starts. (d) Two possible next steps. (e) Chain *b* stops at length 3. (f) Final outcome, including a 4-node branch *c*. (g) This exact structure is prescribed by the port management program *P* carried by each node.

more efficiently (Figure 7). Military or law-enforcement personnel could form human chains and security cordons in complex but targeted branching structures serving multiple purposes: encircling the scene of a threat or accident, guiding people toward the exits, transporting victims to emergency vehicles, and building specific local formations such as enclosed areas containing equipment or medical field units (rectangle in Figure 7).

4.2 External Events

Naturally, the propensity to create structured network formations must also be influenced and modified by the environment in which those formations will function. In Part 3, node attachment was based on port availability driven only by positional gradient values. This internal dynamics must now interact with the external dynamics of the system's context, along with its boundary conditions and events occurring unexpectedly. Environmental landmarks can play different roles in the self-structuring process: they can act as *triggers*, *attractors* or *obstacles*.

- In the SOS example, a located threat or accident can be a starting point that triggers the aggregation of security agents into circular chains around it. This could be implemented through special event-driven ports, searching for external stimuli to virtually "connect" to.
- Then, other human cordons can branch off from this initial structure and grow like trails aiming toward other attractor points, e.g., exits and emergency vehicles. This behavior could be implemented by new "tropism" rules: mobile positioning devices with a knowledge of a facility's map, or with remote sensing capabilities, could bias the attachment routines toward target points of interest, "pulling" a chain formation toward them.

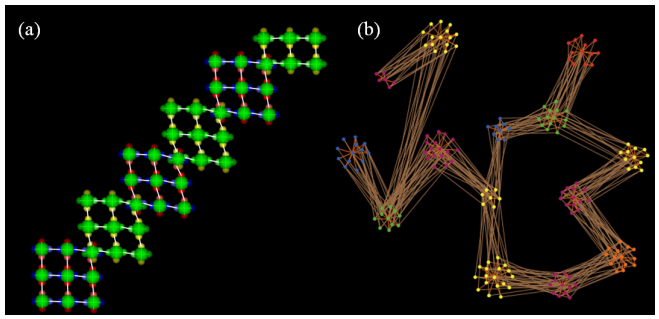

Figure 6: Two simulations of programmed modular networks. (a) Branching 3×3 lattices attached by their corners. (b) Complex branching chain of node clusters, including a cycle.

- Finally, without environmental constraints, this type of deployment would appear like a star shape with a central hub and a few spokes radiating from it symmetrically. However, once immersed in the geography of the stadium, this ideal structure must adapt and bend around obstacles, e.g., turn around corners and stretch across aisles. Such flexibility would naturally result in part from the interaction of human agents with their physical surroundings, but it could also be facilitated by force-based layout algorithms of the type mentioned above.

4.3 Agent Functionality

Another important aspect not included in the abstract model is the diversity of functional roles that agents may take on, in addition to their self-assembly capabilities. In real-world situations such as SOS, the problem is in fact reverse: police officers, paramedics, firefighters, military, etc., already come from different professional trainings and experiences and the new challenge is to

make them interact in a less centralized and more autonomous way, i.e., by carrying automated positioning devices that guide them toward optimal places. In any case, the model must now mix various *predefined* agent identities (the first case of two seen in the introduction), before they even further differentiate by gradient position inside the structure. This natural heterogeneity of agents could be reflected in the model by a heterogeneity of ports and gradients, and diversified attachment rules that depend on agent types. This would result in various *subnetworks* of two kinds: "intra-category" subnetworks linking agents of the same profession (police subnetwork, medical subnetwork) and "inter-category" subnetworks combining agents of different professions together, e.g., into smaller intervention units containing a few members of each type.

4.4 Action Plans

Finally, as discussed in the introduction, the adequacy or "fitness" of the deployed network to the crisis situation, both in its structure and function, will also critically depend on a two-way communication between the agents and command headquarters. Effective emergency deployment cannot rely exclusively on peer-to-peer self-organization at the local level. Techno-social networks still need global monitoring and orchestration. Dynamical adaptation to an evolving crisis basically happens at two levels: (a) quick adaptation to local circumstances at the level of the human agents (collision avoidance, common sense reactions, etc.) *under the same rules of deployment*, and (b) major changes of strategy at the command level that *change the rules of deployment*. High-level C2 action plans would set only the global course of the action, based on symbolic codenames ("raid", "evacuation", "withdrawal", etc.), while the low-level implementation details are carried out by individual agent protocols (real-time positioning). Action plans are compiled down into local rules of attachment and broadcast to all agents. Thus, the network can adapt to new incidents and episodes of an evolving crisis by reprogramming the agents' PDAs on the fly to create new formations.

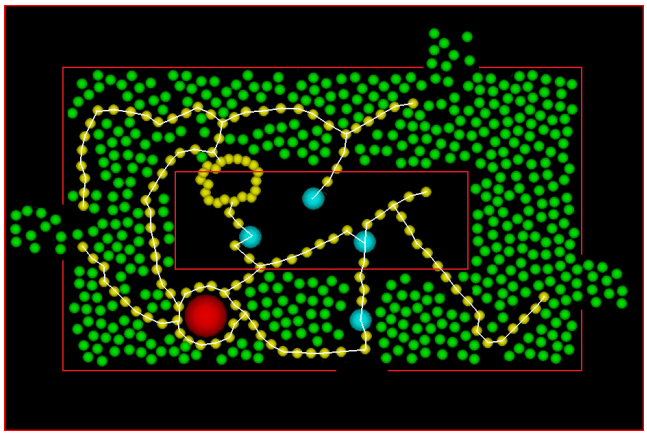

Figure 7: Schematic view (not a simulation) of a possible SOS scenario within the space of a stadium, that would combine programmed networking and dynamic interaction with the environment. Growing cordons of security agents (orange) encircle the threat (red), guide the crowd (green) toward the exits, carry victims to emergency vehicles (blue, driving in and out through gates under the bleachers), and create special enclosed spaces on the field (cycle).

In summary, future work can expand the abstract algorithmic rules (gradient update *G*, port management *P*, and link creation *L*) to take into account spatial extension, external events, agent diversity, and hierarchical command. By implementing these four principles, in addition to intrinsic self-connectivity, the SOS scenario discussed here as an illustration of a self-organized *and* structured eNetwork could become functional. It would involve teams that create specific, but adaptive, spatial architecture to contain the hazard, while directing the crowd toward safety. This dynamical process would be continuously adjusting to the crisis dynamics, including its unexpected new events and effects.

The effectiveness of the SOS network would depend on how the genotype is designed (i.e., how individual roles are specified through protocols) in such a way as to obtain maximal synergy under the overarching constraints imposed by the phenotype (reflected in the SOS network policies). It is this continuous "balancing act" between individual agent autonomy and overall goals (previously explored in holonic enterprises [31]) that would enable the emergence of effective barriers, growing when and where needed, to contain the unexpected developing threats and attacks. This could ensure a continuous adaptation and co-evolution with a crisis dynamics by making an SOS network (as the controller) "weave itself" into the situation to control like a nervous system, growing new connections and "nerves" around important events and locations.

5. ACKNOWLEDGMENTS

This collaborative work has been made possible in part by the Scientific Services of the French Embassy in Ottawa, through their funding of R. Doursat's visit to M. Ulieru's laboratory at UNB. We thank Adam MacDonald for his excellent work in producing the computer simulations.

6. REFERENCES

[1] Ulieru, M., and Cobzaru, M. 2005. Building holonic supply chain management systems: An e-logistics application for the telephone manufacturing industry. IEEE Transactions on Industrial Informatics 1, 1 (Feb. 2005), 18-31.

[2] Grobbelaar, S., and Ulieru, M. 2006. Self-organizing cyber-systems as infrastructure for optimizing power distribution networks. Annual Conference of the South African Institute of Computer Scientists and Information Technologists (Capewinelands, South Africa, October 9-11, 2006). SAICSIT'06.

[3] Ulieru, M., and Unland, R. 2004. Emergent e-Logistics infrastructure for timely emergency response management. In Engineering Self-Organising Systems: Nature Inspired Approaches to Software Engineering, G. Di Marzo Serugendo et al., Eds. Springer-Verlag, Berlin, 139-156.

[4] Ulieru, M. 2004. Emerging computing for the industry: Agents, self-organization and holonic systems. Workshop on Industrial Informatics (Busan, South Korea, November 2-6, 2004). IECON'04.

[5] Carreras, I., Miorandi, D. and Chlamtac, I. 2007. From biology to evolve-able ICT systems. 1st International Workshop on eNetworks Cyberengineering, IEEE Systems

Man and Cybernetics Conference (Montreal, Canada, October 7-10, 2007). IEEE SMC'07.

[6] Ulieru, M. 2007. Evolving the "DNA blueprint" of eNetwork middleware to control resilient and efficient cyber-physical ecosystems. 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (Budapest, Hungary, December 10-14, 2007). BIONETICS'07.

[7] Doursat, R. 2006. The growing canvas of biological development: Multiscale pattern generation on an expanding lattice of gene regulatory networks. InterJournal: Complex Systems 1809.

[8] Doursat, R. 2008. Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. In Organic Computing, R. P. Würtz, Ed. Springer-Verlag, Berlin, 2008, 167-200.

[9] Doursat, R. 2008. Programmable architectures that are complex and self-organized: From morphogenesis to engineering. 11th International Conference on the Simulation and Synthesis of Living Systems (Winchester, UK, August 5-8, 2008). ALIFE XI.

[10] Ulieru, M. 2008. Enabling the SOS network, Proceedings of the IEEE Systems, Man and Cybernetics Conference (SMC 2008), October 12-15, 2008, Singapore.

[11] Ulieru, M. and Verdon, J. 2008. IT revolutions in the industry: From the command economy to the eNetworked industrial ecosystem. 1st International Workshop on Industrial Ecosystems, IEEE International Conference on Industrial Informatics (Daejoen, Korea, July 13-17, 2008).

[12] Macal, C. M. and North, M. J. 2006. Tutorial on agent-based modeling and simulation, Part 2: How to model with agents. In Proceedings of the 2006 Winter Simulation Conference. L. F. Perrone et al., Eds., 73-83.

[13] Grobbelaar, S. and Ulieru, M. 2007. Complex networks as control paradigm for complex systems. 1st International Workshop on eNetworks Cyberengineering, IEEE Systems Man and Cybernetics Conference (Montreal, Canada, October 7-10, 2007). IEEE SMC'07.

[14] Kauffman, S. 2008. Reinventing the Sacred. Basic Books.

[15] Gierer, A. and Meinhardt, H. 1972. A theory of biological pattern formation. Kybernetik 12, 30-39.

[16] Hoelzer, G., Drewes, R., Meier, J. and Doursat, R. 2008. Isolation-by-distance and outbreeding depression are sufficient to drive parapatric speciation in the absence of environmental influences. PLoS Computational Biology 4, 7, e10001262008.

[17] von der Malsburg, C. 1999. The what and why of binding: The modeler's perspective. Neuron 24 (Sep. 1999), 95-104.

[18] Bonabeau, E., Dorigo, M. and Theraulaz, G. 1999. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press.

[19] Grégoire, G. and Chaté, H. 2004. Onset of collective and cohesive motion. Physical Review Letters 92, 025702.

[20] Barabási, A.-L. and Albert, R. 1999. Emergence of scaling in random networks. Science 286, 5439, 509-512.

[21] Isermann, R. 1996. Digital Control Systems. Springer-Verlag, New York.

[22] Heims, S. J. 1991. The Cybernetics Group. MIT Press.

[23] Ulieru, M., Brennan, R. and Walker, S. 2002. The holonic enterprise: A model for Internet-enabled global supply chain and workflow management. International Journal of Integrated Manufacturing Systems 13, 8, 538-550.

[24] Peltz, C. 2003. Web services orchestration and choreography. Computer 36, 10 (Oct. 2003), 46-52.

[25] Bentley, P. and Kumar, S. 1999. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In Proceedings of the Genetic and Evolutionary Computation Conference (Orlando, Florida), W. Banzhaf et al., Eds. Morgan Kaufmann, vol. 1, 35-43.

[26] Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, Jr., T., Nagpal, R., Rauch, E., Sussman, G. and Weiss, R. 1999. Amorphous computing. MIT Artificial Intelligence Laboratory memo 1665 (Aug. 1999).

[27] Nagpal, R. 2002. Programmable self-assembly using biologically-inspired multi-agent control. 1st International Conference on Autonomous Agents (Bologna, Italy, July 15-19, 2002).

[28] Doursat, R. and Ulieru, M. 2009, IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Engineering Cyber-Physical Ecosystems, to appear July 2009 (submitted).

[29] Callebaut, W. and Rasskin-Gutman, D., Eds. 2005. Modularity. MIT Press

[30] Fruchterman, T. M. J. and Reingold, E. M. 1991. Graph drawing by force-directed placement. Software—Practice and Experience 21, 11 (Nov. 1991), 1129-1164.

[31] Ulieru, M. and Grobbelaar, S. 2006. Holonic stigmergy as a mechanism for engineering self-organizing applications. 3rd International Conference of Informatics in Control, Automation and Robotics (Setubal, Portugal, August 1-5, 2006). ICINCO'06.