A General Methodology for Designing Self-Organizing Systems

Carlos Gershenson
Centrum Leo Apostel, Vrije Universiteit Brussel
Krijgskundestraat 33 B-1160 Brussel, Belgium
Phone: +32-2-644.26.77, Fax: +32-2-644.07.44
cgershen@vub.ac.be http://homepages.vub.ac.be/~cgershen

May 4, 2005

Abstract

This paper presents a conceptual framework for speaking about self-organizing systems. The aim is to provide a methodology useful for designing and controlling systems developed to solve complex problems. A brief introduction to complexity and self-organization is given before introducing the conceptual framework and the methodology. A case study on self-organizing traffic lights illustrates the ideas presented in the paper.

Keywords: self-organization, complexity, systems, design, control, methodology

1 Introduction

Over the last half a century, much research in different areas has employed self-organizing systems to solve complex problems, e.g. [1, 2, 3, 4, 5]. Recently, particular methodologies using the concepts of self-organization have been proposed in different areas, such as software engineering [6, 7], electrical engineering [8], and collaborative support [9]. However, there is as yet no general framework for constructing self-organizing systems. Different vocabularies are used in different areas, and with different goals. In this paper, I present an attempt to develop a general methodology that will be useful for designing and controlling *complex* systems [10]. The proposed methodology, as with any methodology, does not provide ready-made solutions to problems. Rather, it provides a *conceptual framework*, a *language,* to assist the solution of problems. Also, many current problem solutions can be *described* as proposed. I am not suggesting new solutions, but an alternative way of thinking about them.

The paper is organized as follows: in the next section, notions of complexity and self-organization are discussed. In Section 3, original concepts are presented. These will be used in the Methodology, exposed in Section 4. In Section 5, a case study concerning self-organizing traffic lights is used to illustrate the steps of the Methodology. Discussion and conclusions follow in Sections 6 and 7.

2 Complexity and Self-organization

There is no general definition of *complexity*, since the concept achieves different meanings in different contexts. Still, we can say that a system is complex if it consists of several *interacting* elements, so that the behaviour of the system will be difficult to deduce from the behaviour of the parts. This occurs when there are many parts, and/or when there are many interactions between the parts. Typical examples of complex systems are a living cell, a society, an economy, an ecosystem, the Internet, the weather, a brain, and a city. These all consist of numerous elements whose interactions produce a global behavior that cannot be reduced to the behavior of their separate components [11].

Even when there is no general definition or measure of complexity, a relative *notion* of complexity can be useful: the complexity of a system scales with the number of its elements, the number of interactions between them, the complexities of the elements, and the complexities of the interactions [12][1].

The term *self-organization* has been used in different areas with different meanings, as is cybernetics [16, 17], thermodynamics [18], biology [19], mathematics [20], computing [21], information theory [22], synergetics [23], and others [24] (for a general overview, see [25]). However, the use of the term is subtle, since any dynamical system can be said to be self-organizing or not, depending partly on the observer [26, 17].

It is not necessary to enter into a philosophical debate on the theoretical aspects of self-organization to work with it, so a practical notion will suffice: a system *described* as self-organizing is one in which elements *interact* in order to achieve a global function or

[1]This can be confirmed mathematically in particular systems. As a general example, random Boolean networks [13, 14, 15] show clearly that the complexity of the network increases with the number of elements and the number of interactions.

behaviour. This function or behaviour is not imposed by one single or a few elements, nor determined hierarchically. It is achieved dynamically as the elements interact with one another. These interactions produce feedbacks that regulate the system. All the previously mentioned examples of complex systems fulfil the definition of self-organization. More precisely, the question can be formulated as follows: when is it useful to describe a system as self-organizing? This will be when the system or environment is very dynamic and/or unpredictable. If we want the system to solve a problem, it is useful to describe a complex system as self-organizing when the "solution" is not known beforehand and/or is changing constantly. Then, the solution is dynamically strived for by the elements of the system. In this way, systems can adapt quickly to unforeseen changes as elements interact locally. In theory, a centralized approach could also solve the problem, but in practice such an approach would require too much time to compute the solution and would not be able to keep the pace with the changes in the system and its environment.

In order to understand self-organizing systems, two or more *levels of abstraction* [12] should be considered: elements (lower level) organize in a system (higher level), which can in turn organize with other systems to form a larger system (even higher level). The understanding of the system's behaviour will come from the relations observed between the descriptions at different levels. Note that the levels, and therefore also the terminology, can change according to the interests of the observer. For example, in some circumstances, it might be useful to refer to cells as elements (e.g. bacterial colonies); in others, as systems (e.g. genetic regulation); and in others still, as systems coordinating with other systems (e.g. morphogenesis).

In the following section, further concepts will be introduced that will be necessary to apply the methodology .

3 The Conceptual Framework

Elements of a complex system interact with each other. The actions of one element therefore affect other elements, directly or indirectly. For example, an animal can kill another animal directly, or indirectly cause its starvation by consuming its resources. These interactions can have negative, neutral, or positive effects on the system [27].

Now, intuitively thinking, it may be that the "smoothening" of local interactions, i.e. the minimization of "interferences" or "friction" will lead to global improvement. But is this always the case? To answer this question, the terminology of multi-agent systems [28, 29, 30, 31] can be used. Every element, and every system, can be seen as an agent with goals and behaviours thriving to reach those goals. The behaviour of agents can affect (positively, negatively, or neutrally) the fulfilment of the goals of other agents, thereby establishing a relation. The satisfaction or fulfilment of the goals of an agent can be represented using a variable $\sigma \in [0,1]$[2]. Relating this to the higher level, the satisfaction of a system σ_{sys} can be recursively represented as a weighted function $f : \mathbb{R} \rightarrow [0..1]$ of the satisfaction of the n

[2]In some cases, σ could be seen as a "fitness" [27]. However, in most genetic algorithms [32] a fitness function is imposed from the outside, whereas σ is a property of the agents, that can change with time.

elements conforming it:

$$\sigma_{sys} = f\left(\sigma_1, \sigma_2, ..., \sigma_n, w_0, w_1, w_2, ..., w_n\right) \tag{1}$$

where w_0 is a bias and the other weights determine the importance given to each σ_i. If the system is homogeneous, then f will be the weighted sum of σ_i, $w_i = \frac{1}{n} \forall i \neq 0$, $w_0 = 0$. For heterogenous systems, the weights w_i's are determined *tautologically* by the importance of the σ of each element to the satisfaction of the system. Thus, it is a useful tautology to say that maximizing individual σ's, adjusting individual behaviours (and thus relations), will maximize σ_{sys}. If several elements decrease σ_{sys} as they increase their σ, we would not consider them as part of the system. An example can be seen with the immune system. It categorizes molecules and micro-organisms as akin or alien [33]. If they are considered as alien, they are attacked. Auto-immune diseases arise when this categorization is erroneous, and the immune system attacks vital elements of the organism. On the other hand, if pathogens are considered as part of the body, they are not attacked. Another example is provided by cancer. Cancerigenic cells can be seen as "rebel", and no longer part of the body, since their goals differ from the goal of the organism. Healthy cells are described easily as part of an organism. But when they turn cancerigenic, they can better be described as parasitic. The tautology is also useful because it gives a general mathematical representation for system satisfaction, which is independent of a particular system.

If the model of a system considers more than two levels, then the σ of higher levels will be recursively determined by the σ's of lower levels. However, the f's most probably will be very different on each level.

Certainly, an important question remains: how do we determine the function f and the weights w_i's? To this question there is no complete answer. One method consists of *lesioning* the system[3]: removing or altering elements of the system, and observing the effect on σ_{sys}. Through analysing the effects of different lesions, the function f can be reconstructed and the weights w_i's obtained. If a small change $\Delta\sigma_i$ in any σ_i produces a change $\Delta\sigma_{sys} \geq \Delta\sigma_i$, the system can be said to be *fragile*.

What could then be done to maximize σ_{sys}? How can we relate the σ_i's and avoid conflicts between elements? This is not an obvious task, for it implies bounding the agents' behaviours that reduce other σ_i's, while preserving their functionality. Not only should the interference or friction between elements be minimized, but the synergy [23] or "positive interference" should also be promoted. Dealing with complex systems, it is not feasible to tell each element what to do or how to do it, but their behaviours need to be constrained or modified so that their goals will be reached, blocking the goals of other elements as little as possible. These constraints can be called *mediators* [34]. They can be imposed from the top down, developed from the bottom up, be part of the environment, or be embedded as an *aspect* [35, Ch. 3] of the system. An example can be found in city traffic: traffic lights, signals and rules mediate among drivers, trying to minimize their conflicts, which result from the competition for limited resources, i.e. space to drive through. The role of a mediator is to arbitrate among the elements of a system, to minimize interferences and frictions and maximize synergy. Therefore, the efficiency of the mediator can be measured directly using

[3]This method has been used widely to detect functions in complex systems such as genetic regulatory networks and nervous systems.

σ_{sys}. Individually, we can measure the "friction" $\phi_i \in [-1, 1]$ for each agent i, relating the change in satisfaction $\Delta\sigma_i$ of element i and the change in satisfaction of the system $\Delta\sigma_{sys}$:

$$\phi_i = \frac{-\Delta\sigma_i - \Delta\sigma_{sys}\,(n-1)}{n}. \tag{2}$$

Friction occurs when the increase of satisfaction of one element causes a decrease in the satisfaction of some other elements that is greater than the increase. Note that $\phi_i = 0$ does imply that there is no conflict, since one agent can "get" the satisfaction proportionally to the "loss" of satisfaction of (an)other agent(s). Negative friction would imply synergy, e.g. when $\Delta\sigma_i \geq 0$ while other elements also increase their σ. The role of a mediator would be to maximize σ_{sys} by minimizing ϕ_i's. With this approach, friction can be seen as a type of *interaction* between elements.

Thus, the problem can be put in a different way: how can we find/develop/evolve efficient mediators for a given system? One answer to this question is the methodology proposed in this paper. The answer will not be complete, since we cannot have precise knowledge of f for large evolving complex systems. This is because the evolution of the system will change its own f [36], and the relationships among different σ_i's. Therefore, predictions cannot be complete. However, the methodology proposes to follow steps to increase the understanding (and consequently the control) of the system and the relations between its elements. The goal is to identify conflicts and diminish them without creating new ones. This will increase the σ_i's and thus σ_{sys}. The precision of f is not so relevant if this is achieved.

It should be noted that the timescale chosen for measuring $\Delta\sigma_i$ is very important, since at short timescales the satisfaction can decrease, while on the long run it will increase. In other words, there can be a short term "sacrifice" to harvest a long term "reward". If the timescale is too small, a system might get stuck in a "local optimum", since all possible actions would decrease its satisfaction on the short term. But in some cases the long term benefit should be considered for maximization. A way of measuring the slow change of σ_i would be with its integral over time for a certain interval Δt:

$$\int_{t}^{t+\Delta t} \sigma_i dt. \tag{3}$$

Another way of dealing with the local optima is to use neutral changes to explore alternative solutions [37].

Before going into further detail, it is worth noting that this is not a reductionist approach. Smoothing out local interactions will not provide straightforward clues as to what will occur at the higher level. Therefore, the system should be observed at both levels: making local and global changes, observing local and global behaviours, and analysing how one affects the other.

Concurrently, the *dependence* $\epsilon \in [-1, 1]$ of an element to the system can be measured by calculating the difference of the satisfaction σ_i when the element interacts within the system and its satisfaction $\overline{\sigma_i}$ when the element is isolated.

$$\epsilon = \sigma_i - \overline{\sigma_i}. \tag{4}$$

In this way, full dependence is given when the satisfaction of the element within the system σ_i is maximal and its satisfaction $\overline{\sigma_i}$ is minimal when the element is isolated. A

negative ϵ would imply that the element would be more satisfied on its own and is actually "enslaved" by the system. Now we can use the dependences of the elements to a system to measure the *integration* $\tau \in [-1, 1]$ of a system, which can be seen also as a gradual measure of a meta-system transition (MST) [38].

$$\tau = \frac{1}{n} \sum_{i=1}^{n} \epsilon_i. \tag{5}$$

A MST is a gradual process, but it will be complete when elements are not able to reach their goals on their own, i.e. $\overline{\sigma_i} \to 0$. Examples include cells in multi-cellular organisms and mitochondria in eukaryotes.

In an evolutionary process, natural selection will tend to increase τ because this implies higher satisfaction both for the system and its elements (systems with a negative τ are not viable). Relations and mediators that contribute to this process will be selected, since higher σ's imply more chances of survival and reproduction. Human designers and engineers also select relations and mediators that increase the σ's of elements and systems. Therefore, we can see that evolution will tend, in the long run, towards synergetic relationships [39], even if resources are scarce.

In the next section, the steps suggested for developing a self-organizing system are presented, using the concepts described in this section.

4 The Methodology

The proposed methodology meets the requirements of a system, i.e. what the system should do, and enables the designer to produce a system that fulfils the requirements. The methodology includes the following steps: Representation, Modelling, Simulation, Application, and Evaluation, which will be exposed in the following subsections. Figure 1 presents these steps. These steps should not necessarily be followed one by one, since the stages merge with each other. There is also backtracking, when the designer needs to return to an earlier stage for reconsideration.

This methodology should not be seen as a recipe that provides ready-made solutions, but rather as a guideline to direct the search for them.

4.1 Representation

The goal of this step is to develop a *specification* (which might be tentative) of the components of the system.

The designer should always remember the distinction between model and modelled. A model is an abstraction/description of a "real" system. Still, there can be several descriptions of the same system [12, 11], and we cannot say that one is better than another independently of a context.

There are many possible representations of a system. According to the *constraints* and *requirements*, which may be incomplete, the designer should choose an appropriate vocabulary (metaphors to speak about the system), abstraction levels, granularity, variables, and interactions that need to be taken into account. Certainly, these will also depend on the

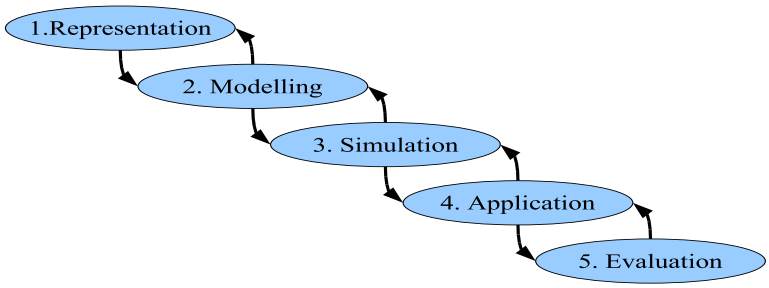

Figure 1: Diagram relating different stages of methodology

experience of the designer. The choice between different approaches can depend more on the expertise of the designer than on the benefits of the approaches.

Even when there is a wide diversity of possible systems, a general approach for developing a Representation can be abstracted. The designer should try to divide a system into elements by identifying semi-independent modules, with internal goals and dynamics, and with few interactions with their environment. Since interactions in a model will increase the complexity of the model, we should group "clusters" of interacting variables into elements, and then study a minimal number of interactions between elements. The first constraints that help us are space and time. It is useful to group variables that are close to each other (i.e. interacting constantly) and consider them as elements that relate to other elements in occasional interactions. Since the proposed methodology considers elements as agents, another useful criterion for delimiting them is the identification of goals. These will be useful in the Modelling to measure the satisfaction σ of the elements. We can look at genes as an example: groups of nucleotides co-occur and interact with other groups and with proteins. Genes are identified by observing nucleotides that keep close together and act together to perform a function. The fulfilment of this function can be seen as a goal of the gene. Dividing the system into modules also divides the problem it needs to solve, so a complex task will be able to be processed in parallel by different modules. Certainly, the integration of the "solutions" given by each module arises as a new problem. Nevertheless, modularity in a system also increases its robustness and adaptability [40, 41, 42].

The representation should consider at least two levels of abstraction, but if there are many variables and interactions in the system, more levels can be contemplated. Since elements and systems can be seen as agents, we can refer to all of them as x-agents, where x denotes the level of abstraction relative to the simplest elements. For example, a three-layered abstraction would contemplate elements (0-agents) forming systems that are elements (subsystems, 1-agents) of a greater system (meta-system, 2-agents). If we are interested in modelling a research institute, 0-agents would be researchers, 1-agents would be research groups, and the research institute would be a 2-agent. Each of these have goals and satisfactions (σ^x)

that can be described and interrelated.

If there are few elements or interactions in the Representation, there will be low complexity, and therefore stable dynamics. The system might be better described using traditional approaches, since the current approach might prove redundant.

4.2 Modelling

In science, models should ideally be as simple as possible, and predict as much as possible [22]. These models will provide a better understanding of a phenomenon than complicated models. Therefore, a good model requires a good Representation. The "elegance" of the model will depend very much on the metaphors we use to speak about the system. If the model turns out to be cumbersome, the Representation should be revised.

The Modelling should specify a Control mechanism that will ensure that the system does what it is required to do. Since we are interested in self-organizing systems, the Control will be *internal* and *distributed*. If the problem is too complex, it can be divided into different subproblems. The Modelling should also consider different trade-offs for the system.

4.2.1 Control mechanism

The Control mechanism can be seen as a *mediator* [34] ensuring the proper interaction of the elements of the system, and one that should produce the desired performance. However, one cannot have a strict control over a self-organizing system. Rather, the system should be *steered* [43]. In a sense, self-organizing systems are like teenagers: they cannot be tightly controlled since they have their own goals. We can only attempt to steer their actions, trying to keep their internal variables under certain boundaries, so that the systems/teenagers do not "break" (in Ashby's sense [44]).

To develop a Control, the designer should find aspect systems, subsystems, or constraints that will prevent the negative interferences between elements (friction) and promote positive interferences (synergy). In other words, the designer should search for ways of maximizing the satisfaction σ of elements that will result in maximization of the global satisfaction σ_{sys}. The performance of different mediators can be measured using equation (1).

The Control mechanism should be *adaptive*. Since the system is dynamic and there are several interactions within the system and with its environment, the Control mechanism should be able to cope with the changes within and outside the system. An adaptive Control will be efficient in more contexts than a static one. In other words, the Control should be *active* in the search of solutions. A static Control will not be able to cope with the complexity of the system. There are several methods for developing an adaptive Control, e.g. [45]. But these should be applied in a distributed way, in an attempt to reduce friction and promote synergy.

Different methods for reducing friction in a system can be identified. In the following cases, an agent A negatively affected by the behaviour of an agent B will be considered:

- **Tolerance**. This can be seen as the acceptance of others and their goals. A can tolerate B by modifying itself to reduce the friction caused by B, and therefore increase σ_A. This can be done by moving to another location, finding more resources, or making internal changes.

- **Courtesy**. This would be the opposite case to Tolerance. B should modify its behaviour not to reduce σ_A.

- **Compromise**. A combination of Courtesy and Tolerance: both agents A and B should modify their behaviours to reduce the friction. This is a good alternative when both elements cause friction to each other. This will be common when A and B are similar, as in homogeneous systems.

- **Imposition**. This could be seen as forced Courtesy. The behaviour of B could be changed by force. The Control could achieve this by constraining B or imposing internal changes.

- **Eradication**. As a special case of Imposition, B can be eradicated. This certainly would decrease σ_B, but can be an alternative when either σ_B does not contribute much to σ_{sys}, or when the friction caused by B in the rest of the system is very high.

- **Apoptosis**. B can "commit suicide". This would be a special case of Courtesy, where B would destroy itself for the sake of the system.

The difference between Compromise/Apoptosis and Imposition/Eradication is that in the former cases, change is triggered by the agent itself, whereas in the latter the change is imposed from the "outside" by a mediator. Tolerance and Compromise could be generated by an agent or by a mediator.

Different methods for reducing friction can be used for different problems. A good Control will select those in which other σ's are not reduced more than the gain in σ's. The choice of the method will also depend on the importance of different elements for the system. Since more important elements contribute more to σ_{sys}, these elements can be given preference by the Control in some cases.

Different methods for increasing synergy can also be identified. These will consist of taking measures to increase σ_{sys}, even if some σ's are reduced:

- **Cooperation**. Two or more agents adapt their behaviour for the benefit of the whole. This might or might not reduce some σ's.

- **Egoism**. An agent can choose to increase its σ if it increases σ_{sys}. A mediator should prevent increases in σ's if these reduce σ_{sys}, i.e. friction.

- **Altruism**. An agent can choose to sacrifice an increase of its σ or to reduce its σ in order to increase σ_{sys}. This would make sense only if the *relative* increase of σ_{sys} is greater than the decrease of the σ of the altruistic agent. A mediator should prevent wasted altruism.

- **Exploitation**. This would be forced Altruism: an agent is driven to reduce its σ to increase σ_{sys}.

In general, the Control should explore different alternatives, trying to constantly increase σ_{sys} by minimizing friction and maximizing synergy. This is a constant process, since a self-organizing system is in a dynamic environment, producing "solutions" for the current situation.

4.2.2 Dividing the problem

If the system is to deal with many parameters, then the main problem, i.e. what the elements should do, could be divided into the problems of communication, cooperation, and coordination [46].

For a system to self-organize, its elements need to *communicate*: they need to "understand" what other elements, or mediators, "want" to tell them. This is easy if the interactions are simple: sensors can give *meaning* to the behaviours of other elements. But as interactions turn more complex, the *cognition* [47] required by the elements should also be increased. New meanings can be learned [48, 49] to adapt to the changing conditions. These can be represented as "concepts" [50], or encoded, e.g., in the weights of a learning neural network. The precise implementation and philosophical interpretations are not relevant if the outcome is the desired one.

The problem of *cooperation* has been widely studied using game theory [51]. There are several ways of promoting cooperation, especially if the system is designed. To mention mention only two of them: the use of tags [52, 53] and multiple levels of selection [54] have proven to yield cooperative behaviour. This will certainly reduce friction and therefore increase σ_{sys}.

Elements of a system should *coordinate* while reducing friction, not to obstruct each other. An important aspect of coordination is the *division of labour*. This can promote synergy, since different elements can specialize in what they are good at and *trust* others to do what they are good at [55, 56]. This process will yield a higher σ_{sys} compared to the case when every element is meant to perform all functions independently of how well each element performs each function. A good Control will promote division of labour by mediating a balance between *specialization* and *integration*: elements should devote more time doing what they are best at, but should still take into account the rest of the system.

4.2.3 Trade-offs

A system needs to be able to cope with the complexity of its domain to achieve its goals. There are several trade-offs that can be identified to reach a balance and cope better with this complexity:

- **Complexity of Elements/Interactions**. The complexity of the system required to cope with the complexity of its domain can be tackled at one end of the spectrum by complex elements with simple interactions, and at the other by simple elements with several/complex interactions.

- **Quality/Quantity**. A system can consist at one extreme of a few complex elements, and at the other of several simple elements.

- **Economy/Redundancy**. Solving a problem with as few elements as possible is economical. But a minimal system is very fragile. Redundancy is one way of favouring the *robustness* of the system [57, 42, 58]. Still, too much redundancy can also reduce the speed of adaptation and increase costs for maintaining the system.

- **Homogeneity/Heterogeneity**. A homogeneous system will be easier to understand and control. A heterogenous system will be able to cope with more complexity with less elements, and will be able to adapt more quickly to sudden changes. If there is a system of ten agents each able to solve ten tasks, a homogeneous system will be able to solve more than ten tasks robustly. A fully heterogeneous system would be able to solve more than a hundred tasks, but it would be fragile if one agent failed.

- **System/Context**. The processing and storage of information can be carried out internally by the system, or the system can exploit its environment "throwing" complexity into it, i.e. allow it to store or process information [59].

- **Ability/Clarity**. A powerful system will solve a number of complex problems, but it will not be very useful if the functioning of the system cannot be understood. Designers should be able to understand the system in order to be able to control it [31].

- **Generality/Particularity**. An abstract Modelling will enable the designer to apply the Modelling in different contexts. However, particular details should be considered to make the implementation feasible.

There are only very relative ways of measuring the above mentioned trade-offs. However, they should be kept in mind during the development of the system.

In a system, the trade-offs will become clearer once the Simulation is underway. They can then be reconsidered and the Modelling updated.

4.3 Simulation

The goal of this stage is to build computer simulation(s) implementing the model(s) developed in the Modelling stage, and test different scenarios and mediator strategies.

The Simulation development should go in stages: from abstract to particular. First an abstract scenario should be used to test the main concepts developed during the Modelling. Only when these are tested and refined, details should be included in the Simulation. This is because particular details take time to develop, and there is no sense in investing before knowing that the Modelling is on the right path. Details can also influence the result of the Simulation, so they should be preferably left for a stage when the main mechanisms are understood.

The Simulation should compare the proposed solutions with traditional approaches. This is to be sure that there is a benefit in applying self-organization in the system. Ideally, the designer should develop in the Modelling stage more than one Control to test its different qualities in the simulation. A rock-scissors-paper situation could arise, where no Control is best in all situations (also considering classic controls). The designer can then adjust or combine the Controls and then compare again in the Simulation.

Experiments made with the aid of the Simulation should go from simple to extensive. Simple experiments will show proof of concepts, and their results can be used to improve the Modelling. Once this is robust, extensive studies should be made to be certain of the performance of the system under different conditions.

Based on the Simulation results and insights, the Modelling and Representation can be improved. A Simulation should be mature before taking the implementation into the real world.

4.4 Application

The role of this stage is basically to use the developed and tested model(s) in a real system. If this is a software system, the transition will not be so difficult. On the other hand, the transition to a real system can expose artifacts of a naive Simulation. A useful way to develop robust Simulations consists in adding some noise into the system [60].

Good theoretical solutions can be very difficult/expensive/impossible to implement (e.g. if they involve instantaneous access to information, mind reading, teleportation, etc.). The feasibility of the Application should be taken into account during the whole design process. In other words, the designer should have an *implementation bias* in all the Methodology stages. If the proposed system turned out to be too expensive or complicated, all the effort of the designer would be fruitless. If the system is developed for a client, there should be feedback between developers and clients during the whole process [61], to avoid client dissatisfaction once the system is implemented. The *legacy* of previous systems should also be considered for the design [62]: if the current implementation is to be modified but not completely replaced, the designer is limited by the capabilities of the old system.

If constraints allow, a pilot study should be made before engaging in a full Application, to detect incongruences and unexpected issues between the Simulation or Modelling stages and the Application. With the results of this pilot study, the Simulation, Modelling, and Representation can be fine-tuned.

4.5 Evaluation

Once the Application is underway, the performance of new system should be measured and compared with the performances of the previous system(s).

If the constraints allow, keep on trying to improve the system, since the requirements for it will certainly change with time (e.g. changes of demand, capacity, etc.). The system will be more adaptive if it considers that its solution might not be the best indefinitely, and is enabled to change itself based on its performance and the demands set on it.

4.6 Notes on the methodology

- All the returning arrows in Figure 1 are given because it is not possible to predict the outcome of strategies before trying them. All the information and eventualities cannot be abstracted, nor emergent properties be predicted before observing them. Emergent properties are *a posteriori*.

- The proposed Methodology will be useful for unpredictable problem domains, where all the possible situations of the system cannot be considered beforehand.

- Most methodologies in the literature apply to software systems, e.g. [63, 64]. The present one is more general, since it is domain independent. General principles can be

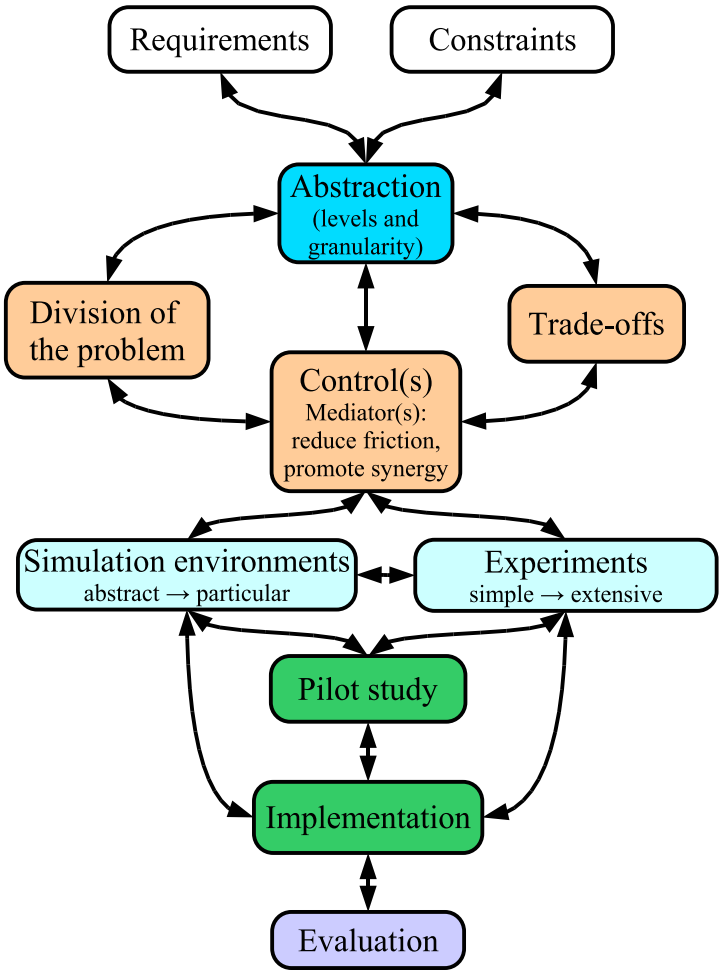

Figure 2: Detailed diagram of Methodology

applied to any domain for developing a functioning self-organizing system. The AMAS methodology [65] has similar goals, but the approaches differ.

- The proposed Methodology is not quite a spiral model [66], because the last stage does not need to be reached to return to the first one. Rather, it is a "*backtracking* model", where the designer can always return to previouis stages.

- It is not necessary to understand a solution before testing it. In many cases understanding can come only after testing, i.e., the global behaviour of the system is irreducible. Certainly, understanding the causes of a phenomenon will allow the modellers to have a greater control over it.

A detailed diagram of the different substeps of the Methodology can be appreciated in Figure 2.

5 Case Study: Self-organizing Traffic Lights

Recent work on self-organizing traffic lights [67] will be used to illustrate the flow through the different steps of the Methodology. These traffic lights are called self-organizing because the global performance is given by the local rules followed by each traffic light: they are unaware of the state of other intersections and still manage to achieve global coordination.

Traffic modelling has increased the understanding of this complex phenomenon [68, 69, 70, 71, 72, 73]. Even when vehicles can follow simple rules, their local interactions generate global patterns that cannot be reduced to individual behaviours. Controlling traffic lights in a city is not an easy task: it requires the coordination of a multitude of components; the components affect one another; furthermore, these components do not operate at the same pace over time. Traffic flows and densities change constantly. Therefore, this problem can be tackled by self-organization. A centralized system could also perform the task, but in practice the amount of computation required to process all the data from a city is too great to be able to respond in real time.

Requirements. The goal is to develop a feasible and efficient traffic light control system.

Representation. The traffic light system can be modelled on two levels: the vehicle level and the city level. These are easy to identify because vehicles are objects that move through the city, establishing clear spatiotemporal distinctions. The goal of the vehicles is to flow as fast as possible, so their "satisfaction" σ can be measured in terms of their average speed and average waiting time at a red light. Cars will have a maximum σ if they go as fast as they are allowed, and do not stop at intersections. σ would be zero if a car stops indefinitely. The goal of the traffic light system on the city level is to enable vehicles to flow as fast as possible, while mediating their conflicts for space and time at intersections. This would minimize fuel consumption, noise, pollution, and stress in the population. The satisfaction of the city can be measured in terms of the average speeds and average waiting times of all vehicles (i.e. average of σ_i, $\forall i$), and with the average percentage of stationary cars. σ_{sys} will be maximum if all cars go as fast as possible, and are able to flow through the city without stopping. If a traffic jam occurs and all the vehicles stop, then σ_{sys} would be minimal. Since all vehicles are considered equal, with no preferences, then all σ_i's should contribute evenly to σ_{sys}. The precise f should not yet be declared, since it would be advisable to identify in the Simulation how variables differ using diverse Control methods and how these reflect the performance of the system.

Modelling. Now the problem for the Control can be formulated: find a mechanism that will coordinate traffic lights so that these will mediate between vehicles to reduce their friction (i.e. try to prevent them from arriving at the same time at crossings). This will maximize the satisfactions of the vehicles and of the city (σ_i's and σ_{sys}). Since all vehicles contribute equally to σ_{sys}, ideally the Control should minimize frictions via Compromise.

Simulation. A simple simulation was developed in NetLogo [74], a multi-agent modelling environment. The "Gridlock" model [75] was extended to implement different traffic control strategies. It consists of an abstract traffic grid with intersections between cyclic single-lane arteries of two types: vertical or horizontal (similar to the scenario of [76]). Cars only flow in a straight line, either eastbound or southbound. Each crossroad has traffic lights that allow traffic flow in only one of the intersecting arteries with a green light. Yellow or red lights stop the traffic. The light sequence for a given artery is green-yellow-red-green. Cars

simply try to go at a maximum speed of 1 "patch" per timestep, but stop when a car or a red or yellow light is in front of them. Time is discrete, but not space. A "patch" is a square of the environment the size of a car. The simulation can be tested at the URL http://homepages.vub.ac.be/~cgershen/sos/SOTL/SOTL.html . At first, a tentative model was implemented. The idea was unsuccessful. However, due to a programming error, an efficient method was discovered, named *sotl-request*.

Modelling. In the *sotl-request* method, each traffic light keeps a count κ_i of the number of cars times time steps ($c * ts$) approaching *only* the red light, independently of the status or speed of the cars (i.e. moving or stopped). κ_i can be seen as the integral of waiting/approaching cars over time. When κ_i reaches a threshold θ, the opposing green light turns yellow, and the following time step it turns red with $\kappa_i = 0$, while the red light which counted turns green. In this way, if there are more cars approaching or waiting before a red light, the light will turn green faster than if there are only few cars. This simple mechanism achieves self-organization as follows: if there is a single or just a few cars, these will be made to stop for a longer period before a red light. This gives time for other cars to join them. As more cars join the group, cars will be made to wait shorter periods before a red light. Once there are enough cars, the red light will turn green even before the oncoming cars reach the intersection, thereby generating "green corridors". Having "platoons" or "convoys" of cars moving together improves traffic flow, compared to a homogeneous distribution of cars, since there are large empty areas between platoons, which can be used by crossing platoons with few interferences. The *sotl-request* method has no phase or internal clock. Traffic lights change only when the above conditions are met. If no cars are approaching a red light, the complementary light can remain green.

Representation. It becomes clear now that it would be useful to consider traffic lights as agents as well. Their goal is to "get rid" of cars as quickly as possible. To do so, they should avoid having green lights on empty streets and red lights on streets with high traffic density. Since the satisfactions of the traffic lights and vehicles are complementary, they should interact via Cooperation to achieve synergy. Also, σ_{sys} could be formulated in terms of the satisfactions of traffic lights, vehicles, or both.

Modelling. Two classic methods were implemented to compare their performance with *sotl-request*: *marching* and *optim*.

Marching is a very simple method. All traffic lights "march in step": all green lights are either southbound or eastbound, synchronized in time. Intersections have a phase φ_i, which counts time steps. φ_i is reset to zero when the phase reaches a period value p. When $\varphi_i == 0$, red lights turn green, and yellow lights turn red. Green lights turn yellow one time step earlier, i.e. when $\varphi == p - 1$. A full cycle of an intersection consists of $2p$ time steps. "Marching" intersections are such that $\varphi_i == \varphi_j, \forall i, j$.

The *optim* method is implemented trying to set phases φ_i of traffic lights so that, as soon as a red light turns green, a car that was made to stop would find the following traffic lights green. In other words, a fixed solution is obtained so that *green waves* flow to the southeast. The simulation environment has a radius of r square patches, so that these can be identified with coordinates (x_i, y_i), $x_i, y_i \in [-r, r]$. Therefore, each artery consists of $2r + 1$ patches. In order to synchronize all the intersections, red lights should turn green and yellow lights

should turn red when

$$\varphi_i == round(\frac{2r + x_i - y_i}{4}) \tag{6}$$

and green lights should turn to yellow the previous time step. The period should be $p = r + 3$. The three is added as an extra margin for the reaction and acceleration times of cars (found to be best, for low densities, by trial and error).

These two methods are *non-adaptive*, in the sense that their behaviour is dictated beforehand, and they do not consider the actual state of the traffic. Therefore, there cannot be Cooperation between vehicles and traffic lights, since the latter ones have fixed behaviours. On the other hand, traffic lights under the *sotl-request* method are sensitive to the current traffic condition, and can therefore respond to the needs of the incoming vehicles.

Simulation. Preliminary experiments have shown that *sotl-request*, compared with the two traditional methods, achieves very good results for low traffic densities, but very poor results for high traffic densities. This is because depending on the value of θ, high traffic densities can cause the traffic lights to change too fast. This obstructs traffic flow. A new model was developed, taking this factor into account.

Modelling. The *sotl-phase* method takes *sotl-request* and only adds the following constraint: a traffic light will not be changed if the time passed since the last light change is less than a minimum phase, i.e. $\varphi_i < \varphi_{\min}$. Once $\varphi_i \geq \varphi_{\min}$, the lights will change if/when $\kappa_i \geq \theta$. This prevents the fast changing of lights[4].

Simulation. *Sotl-phase* performed a bit less effectively than *sotl-request* for very low traffic densities, but still much better than the classic methods. However, *sotl-phase* outperformed them also for high densities. An unexpected phenomenon was observed: for certain traffic densities, *sotl-phase* achieved *full synchronization*, in the sense that no car stopped. Therefore, speeds were maximal and there were no waiting times nor sopped cars. Thus, satisfaction was maximal for vehicles, traffic lights, and the city. Still, this is not a realistic situation, because full synchronization is achieved due to the toroidal topology of the simulation environment. The full synchronization is achieved because platoons are promoted by the traffic lights, and platoons can move faster through the city modulating traffic lights. If two platoons are approaching an intersection, *sotl-phase* will stop one of them, and allow the other to pass without stopping. The latter platoon keeps its phase as it goes around the torus, and the former adjusts its speed until it finds a phase that does not cause a conflict with another platoon.

Modelling. Understanding the behaviour of the platoons, it can be seen that there is a favourable system/context trade-off. There is no need of direct communication between traffic lights, since information can actually be sent via platoons of vehicles. The traffic lights communicate *stigmergically* [78], i.e. via their environment, where the vehicles are conceptualized as the environment of traffic lights.

Simulation. With encouraging results, changes were made to the Simulation to make it more realistic. Thus, a scenario similar to the one of [79] was developed. Traffic flow in four directions was introduced, alternating streets. This is, arteries still consist of one lane, but the directions alternate: southbound-northbound in vertical roads, and eastbound-westbound in

[4]A similar method has been used successfully in the United Kingdom for some time, but for isolated intersections [77].

horizontal roads. Also, the possibility of having more cars flowing in particular directions was introduced. Peak hour traffic can be simulated like this, regulating the percentages of cars that will flow in different roads. An option to switch off the torus in the simulation was added. Finally, a probability of turning at an intersection P_{turn} was included. Therefore, when a car reaches an intersection, it will have a probability P_{turn} of reducing its speed and turning in the direction of the crossing street. This can cause cars to leave platoons, which are more stable when $P_{turn} = 0$.

The results of experiments in the more realistic Simulation confirmed the previous ones: self-organizing methods outperform classic ones. There can still be full synchronization with alternating streets, but not without a torus or with $P_{turn} > 0$.

Modelling. Another method was developed, *sotl-platoon*[5], adding two restrictions to *sotl-phase* for regulating the size of platoons. Before changing a red light to green, *sotl-platoon* checks if a platoon is not crossing through, not to break it. More precisely, a red light is not changed to green if on the crossing street there is at least one car approaching at ω patches from the intersection. This keeps platoons together. For high densities, this restriction alone would cause havoc, since large platoons would block the traffic flow of intersecting streets. To avoid this, a second restriction is introduced. The first restriction is not taken into account if there are more than μ cars approaching the intersection. Like this, long platoons can be broken, and the restriction only comes into place if a platoon will soon be through an intersection.

Simulation. *Sotl-platoon* manages to keep platoons together, achieving full synchronization commonly for a wide density range, more effectively than *sotl-phase* (when the torus is active). This is because the restrictions of *sotl-platoon* prevent the breaking of platoons when these would leave few cars behind, with a small time cost for waiting vehicles. Still, this cost is much lower than breaking a platoon and waiting for separated vehicles to join back again so that they can switch red lights to green before reaching an intersection. However, for high traffic densities platoons aggregate too much, making traffic jams more probable. The *sotl-platoon* method fails when a platoon waiting to cross a street is long enough to reach the previous intersection, but not long enough to cut its tail. This will prevent waiting cars from advancing, until more cars join the long platoon. This failure could probably be avoided introducing further restrictions. In more realistic experiments (four directions, no torus, $P_{turn} = 0.1$), *sotl-platoon* gives on average 30% (up to 40%) more average speed, half the stopped cars, and seven times less average waiting times than non-responsive methods. Complete results and graphics of the experiments discussed here can be found in [67].

Representation. If priority is to be given to certain vehicles (e.g. public transport, emergency), weights can be added to give more importance to some σ_i's.

A meso-level might be considered, where properties of platoons can be observed: their behaviours, performance, and satisfaction and the relationships of these with the vehicle and city levels could enhance the understanding of the self-organizing traffic lights and even improve them.

Simulation. Streets of varying distances between crossings were tested, and all the self-organizing methods maintained their good performance. Still more realistic simulations should be made before moving to the Implementation, because of the cost of such a system.

[5]Curiously, this method was the result of misinterpreting a suggestion by Bart De Vylder.

At least, multiple-street intersections, multiple-lane streets, lane changing, different driving behaviours, and non homogeneous streets should be considered.

Application. The proposed system has not been implemented yet. Still, it is feasible to do so, since there is the sensor technology to implement the discussed methods in an affordable way. A pilot study should be made before applying it widely, to fine tune different parameters and methods. External factors, e.g. pedestrians, could also affect the performance of the system.

Pedestrians could be taken into account considering them as cars approaching a red light. For example, a button could be used to inform the intersection of their presence, and this would contribute to the count κ_i.

A mixed strategy between different methods could be considered, e.g. *sotl-platoon* for low and medium densities, and *sotl-phase* or *marching* for high densities.

Evaluation. If a city deploys a self-organizing traffic light system, it should be monitored and compared with previous systems. This will help to improve the system. If the system is a success, its implementation in other cities would be promoted.

6 Discussion

As could be seen in the case study, the backtracking between different steps in the Methodology is necessary because the behaviour of the system cannot be predicted from the Modelling, i.e. it is not reducible. It might be possible to reason about all possible outcomes of simple systems, and then to implement the solution. But when complexity needs to be dealt with, a mutual feedback between experience and reasoning needs to be established, since reasoning alone cannot process all the information required to predict the behaviour of a complex system.

For this same reason, it would be preferable for the Control to be distributed. Even when a central supercomputer could possibly solve a problem in real time, the information delay caused by data transmission and integration can reduce the efficiency of the system. Also, a distributed Control will be more robust, in as much as if a module malfunctions, the rest of the system can still provide reliable solutions. If a central Control fails, the whole system will stop working.

Now the reader might wonder whether the proposed Methodology is a *top-down* or a *bottom-up* approach. And the answer is: it is both and neither, since (at least) higher and lower levels of abstraction need to be considered simultaneously. The approach tests different local behaviours, and observes local and global (and meso) performances, for local and global (and meso) requirements.

Since "conflicts" between agents need to be solved at more than one level, the Control strategies should be carefully chosen and tested. A situation as in the prisoner's dilemma [51] might easily arise, when the "best" solution on one level/timescale is not the best solution on another level/timescale.

Many frictions between agents are due to faulty communication, especially in social and political relations. If agents do not "know" the goals of others, it will be much more difficult to coordinate and increase σ_{sys}. For example, in a social system, knowing what people or corporations need to fulfil their goals is not so obvious. Still, with emerging technologies,

social systems perform better in this respect. Already in the early 1970s, the project Cybersin in Chile followed this path [80]: it kept a daily log of productions and requirements from all over the country (e.g. mines, factories, etc.), in order to distribute products where they were needed most; and as quickly as possible. Another step towards providing faster response to the needs of both individuals and social systems can be found in e-government [81]. A company should also follow these principles to be able to adapt as quickly as possible. It needs to develop "sensors" to perceive the satisfactions and conflicts of agents at different levels of abstraction, and needs to develop fast ways of adapting to emerging conflicts, as well as to changing economic environment. A tempting solution might be to develop a homogeneous system since, e.g., homogeneous societies have fewer conflicts [82]. This is because all the elements of a homogeneous system pursue the same goals. Thus, less diversity is easier to control. However, less diversity will be less able to adapt to sudden changes. Nevertheless, societies cannot be *made* homogeneous without generating conflicts since people are already diverse, and therefore already have a diversity of goals. The legacy [62] of social systems gives less freedom to a designer, since some goals are already within the system. A social Control/mediator needs to satisfy these while trying to satisfy those of the social system.

7 Conclusions

This paper suggests a conceptual framework and a general methodology for designing and controlling self-organizing systems. The Methodology proposes the exploration for proper Control mechanisms/mediators/constraints that will reduce frictions and promote synergy so that elements will dynamically reach a robust and efficient solution.

Even if this paper is aimed mainly at engineers, it is rather philosophical. It presents no concrete results, but *ideas* that can be exploited to produce them. It should be noted that in order to achieve this goal, the abilities of the engineer will be more important than any methodology. Yet, the proposed Methodology can increase these abilities.

The backtracking ideology is also applicable to this Methodology: it will be improved once applied, through learning from experience. The more this Methodology is used, the more potentially useful its abstractions will be. For example, would it be a good strategy to minimize the standard deviation of σ's? This might possibly increase stability and reduce the probability of conflict, but the strategy needs to be tested before it can be properly understood.

Any system is liable to make mistakes (and *will* make them in an unpredictable environment). But a good system will *learn* from its mistakes. This is the basis for adaptation. It is pointless to attempt to build a "perfect" system, since it is not possible to predict future interactions with its environment. What should be done is to build systems that can adapt to their unexpected future.

8 Acknowledgements

I should like to thank Hugues Bersini, Marco Dorigo, Erden Göktepe, Francis Heylighen, and Marko Rodriguez for interesting discussions and comments. I also wish to thank Michael

Whitburn for proof-reading the manuscript. This research was partly supported by the Consejo Nacional de Ciencia y Teconolgía (CONACyT) of México.

References

[1] W. R. Ashby, *An Introduction to Cybernetics*. London: Chapman & Hall, 1956.

[2] S. Beer, *Decision and Control*. New York: John Wiley and Sons, 1966.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity, New York: Oxford University Press, 1999.

[4] G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, eds., *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering*, vol. 2977 of *Lecture Notes in Computer Science*, Springer, 2004. Revised and extended papers presented at the Engineering Self-Organising Applications Workshop, ESOA 2003, held at AAMAS 2003 in Melbourne, Australia, in July 2003 and selected invited papers from leading researchers in self-organisation.

[5] F. Zambonelli and O. F. Rana, "Self-organization in distributed systems engineering: Introduction to the special issue," *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 35, pp. 313 – 315, May 2005.

[6] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285–312, 2000.

[7] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Trans on Software Engineering and Methodology*, vol. 12, no. 3, pp. 317–370, 2003.

[8] P. Ramamoorthy, S. Zhang, C. Fubao, and D. Ramachandran, "A new paradigm for the design of nonlinear dynamical systems and self-organizing systems," in *Intelligent Control, 1993., Proceedings of the 1993 IEEE International Symposium on*, pp. 571–576, 1993.

[9] P. M. Jones, N. Contractor, B. O'Keefe, and S. C.-Y. Lu, "Competence models and self-organizing systems: Towards intelligent, evolvable, collaborative support," in *Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology'., 1994 IEEE International Conference on*, vol. 1, pp. 367 – 372, vol. 1, 1994.

[10] Y. Bar-Yam, *Dynamics of Complex Systems*. Studies in Nonlinearity, Westview Press, 1997.

[11] C. Gershenson and F. Heylighen, "How can we think the complex?," in *Managing Organizational Complexity: Philosophy, Theory and Application* (K. Richardson, ed.), ch. 3, Information Age Publishing, 2005.

[12] C. Gershenson, "Complex philosophy," in *Proceedings of the 1st Biennial Seminar on Philosophical, Methodological & Epistemological Implications of Complexity Theory*, (La Habana, Cuba), 2002.

[13] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[14] S. A. Kauffman, *The Origins of Order*. Oxford University Press, 1993.

[15] C. Gershenson, "Introduction to random boolean networks," in *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)* (M. Bedau, P. Husbands, T. Hutton, S. Kumar, and H. Suzuki, eds.), (Boston, MA), pp. 160–173, 2004.

[16] H. von Foerster, "On self-organizing systems and their environments," in *Self-Organizing Systems* (M. C. Yovitts and S. Cameron, eds.), pp. 31–50, Pergamon, 1960.

[17] W. R. Ashby, "Principles of the self-organizing system," in *Principles of Self-Organization* (H. V. Foerster and J. G. W. Zopf, eds.), pp. 255–278, Pergamon, 1962.

[18] G. Nicolis and I. Prigogine, *Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order Through Fluctuations*. Wiley, 1977.

[19] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*. Princeton University Press, 2003.

[20] G. G. Lendaris, "On the definition of self-organizing systems," *Proceedings of the IEEE*, March 1964.

[21] F. Heylighen and C. Gershenson, "The meaning of self-organization in computing," *IEEE Intelligent Systems*, pp. 72–75, July/August 2003.

[22] C. R. Shalizi, *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin at Madison, 2001.

[23] H. Haken, "Synergetics and the problem of selforganization," in *Self-Organizing Systems: An Interdisciplinary Approach* (G. Roth and H. Schwegler, eds.), pp. 9–13, Campus Verlag, 1981.

[24] J. Skår and P. V. Coveney, eds., *Self-Organization: The Quest for the Origin and Evolution of Structure*, Phil. Trans. R. Soc. Lond. A 361(1807), June 2003. Proceedings of the 2002 Nobel Symposium on self-organization.

[25] F. Heylighen, "The science of self-organization and adaptivity," in *The Encyclopedia of Life Support Systems*, EOLSS Publishers, 2003.

[26] C. Gershenson and F. Heylighen, "When can we call a system self-organizing?," in *Advances in Artificial Life, 7th European Conference, ECAL 2003 LNAI 2801* (W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, eds.), pp. 606–614, Springer-Verlag, 2003.

[27] F. Heylighen and D. T. Campbell, "Selection of organization at the social level: Obstacles and facilitators of metasystem transitions," *World Futures: the Journal of General Evolution*, vol. 45, pp. 181–212, 1995.

[28] P. Maes, "Modeling adaptive autonomous agents," *Artificial Life*, vol. 1, no. 1&2, pp. 135 – 162, 1994.

[29] M. Wooldridge and N. R. . Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[30] M. Wooldridge, *An Introduction to MultiAgent Systems*. Chichester, England: John Wiley and Sons, 2002.

[31] F. Schweitzer, *Brownian Agents and Active Particles. Collective Dynamics in the Natural and Social Sciences*. Springer Series in Synergetics, Berlin: Springer, 2003.

[32] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[33] N. M. Vaz and F. J. Varela, "Self and non-sense: An organism-centered approach to immunology," *Medical Hypothesis*, vol. 4, no. 3, pp. 231–267, 1978.

[34] F. Heylighen, "Mediator evolution," tech. rep., Principia Cybernetica, 2003.

[35] W. Ten Haaf, H. Bikker, and D. J. Adriaanse, *Fundamentals of Business Engineering and Management, A Systems Approach to People and Organisations*. Delft University Press, 2002.

[36] S. A. Kauffman, *Investigations*. Oxford University Press, 2000.

[37] M. Kimura, *The Neutral Theory of Molecular Evolution*. Cambridge: Cambridge University Press, 1983.

[38] V. Turchin, *The Phenomenon of Science. A Cybernetic Approach to Human Evolution*. New York: Columbia University Press, 1977.

[39] P. A. Corning, *Nature's Magic: Synergy in Evolution and the Fate of Humankind*. Cambridge University Press, 2003.

[40] H. A. Simon, *The Sciences of the Artificial*. MIT Press, 3rd ed., 1996.

[41] R. A. Watson, *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, May 2002.

[42] P. Fernández and R. Solé, "The role of computation in complex regulatory networks," in *Power Laws, Scale-Free Networks and Genome Biology* (E. V. Koonin, Y. I. Wolf, and G. P. Karev, eds.), Landes Bioscience, 2004.

[43] N. Wiener, *Cybernetics; or, Control and Communication in the Animal and the Machine.* New York: Wiley and Sons, 1948.

[44] W. R. Ashby, "The nervous system as physical machine: With special reference to the origin of adaptive behavior," *Mind*, vol. 56, pp. 44–59, January 1947.

[45] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, 1989-1994.

[46] C. Gershenson and F. Heylighen, "Protocol requirements for self-organizing artifacts: Towards an ambient intelligence," in *Proceedings of International Conference on Complex Systems ICCS2004* (Y. Bar-Yam, ed.), (Boston, MA), 2004. Also AI-Lab Memo 04-04.

[47] C. Gershenson, "Cognitive paradigms: Which one is the best?," *Cognitive Systems Research*, vol. 5, pp. 135–156, June 2004.

[48] L. Steels, "Synthesising the origins of language and meaning using co-evolution, self-organisation and level formation," in *Approaches to the Evolution of Language* (J. R. Hurford, M. Studdert-Kennedy, and C. Knight, eds.), pp. 384–404, Cambridge University Press, 1998.

[49] E. D. de Jong, *Autonomous Formation of Concepts and Communication*. PhD thesis, Vrije Universiteit Brussel, 2000.

[50] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought*. Bradford Books, MIT Press, 2000.

[51] R. M. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1984.

[52] R. Riolo, M. D. Cohen, and R. M. Axelrod, "Evolution of cooperation without reciprocity," *Nature*, vol. 414, pp. 441–443, 2001.

[53] D. Hales and B. Edmonds, "Evolving social rationality for MAS using "tags"," in *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems* (J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, eds.), pp. 497–503, ACM Press, 2003.

[54] T. Lenaerts, *Different Levels of Selection in Artificial Evolutionary Systems: Analysis and Simulation of Selection Dynamics*. PhD thesis, Vrije Universiteit Brussel, 2003.

[55] B. R. Gaines, "The collective stance in modeling expertise in individuals and organizations," *Int. J. Expert Systems*, vol. 71, pp. 22–51, 1994.

[56] G. Di Marzo Serugendo, "Trust as an interaction mechanism for self-organising systems," in *International Conference on Complex Systems (ICCS'04)* (Y. Bar-Yam, ed.), 2004.

[57] J. von Neumann, *The Theory of Self-Reproducing Automata*. University of Illinois Press, 1966. Edited by A. W. Burks.

[58] A. Wagner, "Distributed robustness versus redundancy as causes of mutational robustness," Tech. Rep. 04-06-018, Santa Fe Institute, 2004.

[59] C. Gershenson, J. Broekaert, and D. Aerts, "Contextual random Boolean networks," in *Advances in Artificial Life, 7th European Conference, ECAL 2003 LNAI 2801* (W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, eds.), pp. 615–624, Springer-Verlag, 2003.

[60] N. Jakobi, "Evolutionary robotics and the radical envelope of noise hypothesis," *Adaptive Behavior*, vol. 6, no. 2, pp. 325–368, 1997.

[61] T. Cotton, "Evolutionary fusion: A customer-oriented incremental life-cycle for Fusion," *Hewlett Packard Journal*, vol. 47, no. 4, 1996.

[62] P. Valckenaers, H. Van Brussel, Hadeli, O. Bochmann, B. Saint Germain, and C. Zamfirescu, "On the design of emergent systems: An investigation of integration and interoperability issues," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 4, pp. 377–393, 2003.

[63] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Addison-Wesley Object Technology Series, Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[64] N. R. Jennings, "On agent-based software engineering," *Artificial Intelligence*, vol. 117, no. 2, pp. 277–296, 2000.

[65] D. Capera, J.-P. Georgé, M.-P. Gleizes, and P. Glize, "The AMAS theory for complex problem solving based on self-organizing cooperative agents," in *1st International Workshop on Theory and Practice of Open Computational Systems TAPOCS 2003 at IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises WETICE 2003*, p. 383, 2003.

[66] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[67] C. Gershenson, "Self-organizing traffic lights," Tech. Rep. 2005-02, ECCO, 2005.

[68] I. Prigogine and R. Herman, *Kinetic Theory of Vehicular Traffic*. New York: Elsevier, 1971.

[69] D. E. Wolf, M. Schreckenberg, and A. Bachem, eds., *Traffic and Granular Flow '95*, (Singapore), World Scientific, 1996.

[70] M. Schreckenberg and D. E. Wolf, eds., *Traffic and Granular Flow '97*, (Singapore), Springer, 1998.

[71] D. Helbing, H. J. Herrmann, M. Schreckenberg, and D. E. Wolf, eds., *Traffic and Granular Flow '99: Social, Traffic, and Granular Dynamics*, (Berlin), Springer, 2000.

[72] D. Helbing, *Verkehrsdynamik*. Berlin: Springer, 1997.

[73] D. Helbing and B. A. Huberman, "Coherent moving states in highway traffic," *Nature*, vol. 396, pp. 738–740, 1998.

[74] U. Wilensky, "NetLogo," 1999. http://ccl.northwestern.edu/netlogo.

[75] U. Wilensky and W. Stroup, "NetLogo HubNet Gridlock model," 2002. http://ccl.northwestern.edu/netlogo/models/HubNetGridlock.

[76] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing traffic lights in a cellular automaton model for city traffic," *Physical Review E*, vol. 64, p. 056132, 2001.

[77] R. A. Vincent and C. P. Young, "Self optimising traffic signal control using microprocessors - the TRRL MOVA strategy for isolated intersections," *Traffic Engineering and Control*, vol. 27, pp. 385–387, July/August 1986.

[78] G. Theraulaz and E. Bonabeau, "A brief history of stimergy," *Artificial Life*, vol. 5, pp. 97 – 116, Spring 1999.

[79] B. Faieta and B. A. Huberman, "Firefly: A synchronization strategy for urban traffic control," Tech. Rep. SSL-42, Xerox PARC, Palo Alto, 1993.

[80] E. Miller Medina, *The State Machine: Politics, Ideology, and Computation in Chile, 1964-1973*. PhD thesis, MIT, 2005.

[81] K. Layne and J. Lee, "Developing fully functional E-government: A four stage model," *Government Information Quarterly*, vol. 18, p. 122136, 2001.

[82] É. Durkheim, *The Division of Labor in Society*. New York: The Free Press, 1893 (1984). Translated by George Simpson.