Article

Self-organization in multi-agent systems

DI MARZO SERUGENDO, Giovanna, GLEIZES, Marie-Pierre, KARAGEORGOS, Anthony

Abstract

This paper is the synthesis of joint work realised in a technical forum group within the AgentLink III NoE framework, which elaborated on issues concerning self-organization and emergence in multi-agent systems (MAS). The work concluded on a common definition of the concepts of self-organization and emergence in MAS and the associated properties and characteristics. Also it developed towards an approach for selecting self-organization mechanisms using a number of selected reference case studies and a set of evaluation criteria.

The Knowledge Engineering Review, Vol. 20:2, 165–189. © 2005, Cambridge University Press
doi:10.1017/S0269888905000494 Printed in the United Kingdom

Self-organization in multi-agent systems

GIOVANNA DI MARZO SERUGENDO[1], MARIE-PIERRE GLEIZES[2] and ANTHONY KARAGEORGOS[3]

[1]*University of Geneva, Department of Information Systems, 24, rue Général-Dufour, CH-1211 Geneva 4;*
e-mail: Giovanna.Dimarzo@cui.unige.ch
[2]*IRIT, UniversitéPaul Sabatier, France;*
e-mail: Marie-Pierre.Gleizes@irit.fr
[3]*University of Thessaly, Department of Computer & Communication Engineering, 382 21 Volos, Greece;*
e-mail: karageorgos@computer.org

Abstract

This paper is the synthesis of joint work realised in a technical forum group within the AgentLink III NoE framework, which elaborated on issues concerning self-organization and emergence in multi-agent systems (MAS). The work concluded on a common definition of the concepts of self-organization and emergence in MAS and the associated properties and characteristics. Also it developed towards an approach for selecting self-organization mechanisms using a number of selected reference case studies and a set of evaluation criteria.

1 Introduction

For several years, engineers of large-scale distributed systems or systems composed of a large number of elements have investigated solutions where those systems are composed of several (simple) elements which collectively produce 'something' more complex at the global level without explicit external control. It is not chaos, but an organization (a pattern, a process that produces this pattern, or a function that the system aims to fulfil). It is what is frequently referred to as *self-organization* and *emergent phenomenon.* There has recently been a great interest in the study, analysis and design of systems capable of producing a collective outcome resulting from local interactions among simple individual components. More precisely, what is interesting is the ability of such a system to have a complex collective response arising from interactions among relatively simple individual components.

The terms involved, for example, organization, self-organization and emergence, are been used in various disciplines, such as mathematics, physics, biology, and philosophy. These terms are increasingly used in computer science and in particular they are widely met in the multi-agent system (MAS) research community. We need to establish a common, consensual and operational meaning of these terms to enable researchers to know whether their artificial systems are self-organizing or whether there is emergent phenomenon.

This paper reports on the work carried out on these issues over a period of 18 months by a group of researchers organized into a technical forum group (TFG) on *self-organization for MAS*[1] hosted by the AgentLink III NoE European project.[2] This group elaborated heavily on the definitions of the two main concepts of self-organization and emergence, aiming to provide commonly agreed definitions and identification of the related properties. Furthermore, much attention was paid to the

[1] http://www.irit.fr/TFGSO/.
[2] http:///www.agentlink.org/.

fundamental notion of self-organization mechanisms, to the problem of assessing and characterizing such mechanisms, aiming to increase their reuse and facilitate selection of suitable mechanisms for given application requirements. To this purpose, three case studies from the representative domains of network management, manufacturing control and biology were selected and the TFG contributors followed a number of different approaches, concerning both modelling and implementation, to realise the necessary self-organizing and emerging behaviour that would solve the particular case study problem and meet its requirements. The case studies cover a wide variety of dynamic aspects, for example, both physical and logical system distribution and both small and large system sizes.

The objective of this exercise was to acquire a deeper understanding of self-organization mechanisms and to elaborate on their similarities and differences with an aim to assess their value. To facilitate evaluation, a unified framework for describing the case study problem, the mechanisms employed and the solutions provided was used. The self-organizing mechanism assessment was based on the existence of certain self-organization properties and on the satisfaction of evaluation criteria introduced for this purpose.

In this paper we only provide short descriptions of the selected case studies and the solutions provided and we focus on the analysis and comparison results; however, a more detailed description of the case studies and the mechanisms employed can be found in Di Marzo Serugendo *et al.* (2005). The present text starts with providing background definitions of the terms self-organization and emergence in Sections 2 and 3 respectively. The case study applications and the respective results are described in Sections 4–7. Section 8 concludes the paper.

2 Self-organization

2.1 Definitions

Self-organization is an attractive way to handle the dynamic requirements in software. It refers to a process where a system changes its internal organization to adapt to changes in its goals and the environment without explicit external control. Self-organization often results in emergent behaviour that can be either desirable or undesirable. Due to the dynamism and openness of contemporary agent environments and the ever increasing distribution, complexity and dynamic changes in application requirements, understanding the mechanisms that can be used to model, assess and engineer self-organizing behaviour in MAS is an issue of major interest.

> **Self-organization** is defined here as the mechanism or the process enabling a system to change its organization without explicit external command during its execution time.

It is also necessary to distinguish between systems where there is no internal and external explicit control from those where there is an internal centralized control. For example, in a termite society, the different arches are all located at the same distance from the queen due to a pheromone gradient. The queen broadcasts this information and this action is an internal control. By consequence the following definitions have been made.

> **Strong self-organizing systems** are those systems where there is no explicit central control either internal or external.

> **Weak self-organizing systems** are those systems where, from an internal point of view, there is re-organization maybe under an internal (central) control or planning. This kind of system can be illustrated by the example of the termites putting the bullet in a circle under the control of the queen.

https://doi.org/10.1017/S0269888905000494

2.2 Properties of self-organizing behaviour

Self-organizing behaviour is characterized by the following (mandatory or optional) properties.

- *Absence of explicit external control.* This is a mandatory property that states that the system is autonomous; it imposes and changes its organization based solely on internal decisions and without following any explicit external (re-)organization command. This property refers to the *self-* part of the self-organization definition.
- *Decentralized control.* A self-organizing system can work under decentralized control. In this case, there is no internal central authority or centralized information flow. As a result, access to global information is limited by the locality of interactions, which is governed by simple rules. This property is generally not mandatory, since we can observe internal central control in many natural self-organizing systems, such as the termite systems. However, in the context of the TFG efforts that concentrated on strong self-organization and MAS, the existence of decentralized control is also considered a mandatory property of self-organizing MAS.
- *Dynamic operation.* This mandatory property is linked to the system evolution in time. Since the organization evolves independently of any external control, this property implies continuity in the self-organization process.

2.3 Characteristics of self-organizing systems

In addition to the above defined properties that enable one to determine if a system is actually self-organizing, there are several characteristics that can appear in self-organizing systems, both natural and artificial, which alone are not sufficient to justify existence of self-organization.

- *Endogenous global order.* The system reaches some (stable) global state that is produced from *within* the system. This is also related to the notion of *emergent phenomenon.*
- *Emergence.* This refers to having emergent phenomena arising from local interactions. Such phenomena can be observed at global level but they cannot be accurately deduced from examining individual behaviours. Generally, emergent phenomena arise from local interactions occurring among the individual components, thus allowing the system to operate without any central control (see also Section 3 on Emergence).
- *Simple local rules.* The overall complex system behaviour can be based on simple individual behavioural rules. In this case, the information stored in rule descriptions is less than the information needed to describe global behavioural patterns. Local information describes the mechanism for producing the global behavioural pattern and not the pattern itself.
- *Dissipation.* In the absence of external perturbation, the system is expected to stabilize in some states in which emergent properties can be observed. This implies a kind of dissipation of some 'energy', otherwise the system would be continuously changing.
- *Instability.* Systems showing instability are mainly characterized by nonlinear dynamics which have the effect of small fluctuations in environmental conditions that result in significant variations in overall system behaviour. Furthermore, such systems exhibit increased sensitivity to initial conditions and parameter values resulting in small changes to parameters and producing different overall behaviour patterns. Therefore, the overall system properties cannot be understood by simply examining the individual component behaviours separately.
- *Multiple equilibria.* Multiple equilibria are observed when many possible attractors for stable states are present in the system.
- *Criticality.* This characteristic is related to the presence of threshold effects or phase changes.
- *Redundancy.* A system presents redundancy when it demonstrates insensitivity to damage due to replication of components.
- *Self-maintenance.* A system is self-maintained if it has the capacity to reproduce or repair itself, essentially by reproducing or repairing its components.

https://doi.org/10.1017/S0269888905000494

- *Adaptivity.* The re-organization capability of self-organizing systems implies adaptation to external (environmental) variations.
- *Complexity.* This characteristic usually arises from the irreducibility of the global properties to a combination of local behaviours.
- *Hierarchies.* Hierarchies are present in a system when multiple nested self-organized levels can be observed.

2.4 Operational aspects for self-organization in MAS

When we design an artificial system, we generally want it to be self-organizing in order to benefit from self-organizing properties such as robustness, adaptivity, etc. This is a list of operational characteristics a MAS should demonstrate or have in order to be considered an artificial self-organizing system.

- Self-organization and emergence are present simultaneously.
- Several agents compose the system—trivial unless not MAS.
- Many local interactions occur among agents.
- Agents perceive other agents and global result.
- Agents perceive themselves.
- Local rules are defined at the agent level.
- Interactions occur between the MAS and the environment when they can be differentiated.
- MAS and environment: two kinds of MAS can be distinguished, those that are differentiated from the environment, and those that are combined with the system (also called eco-system). Two kinds of environment can be differentiated, the environment of individual agents and the environment of the whole MAS.
- The MAS works with decentralized internal control.
- The MAS environment is dynamically changing.
- Environmental constraints make the MAS converge to a stable state.
- Many interactions or many events occur in the system.

2.5 Measures for self-organization

The issue of measuring and assessing self-organization mechanisms has so far received little attention despite the fact that a proper assessment is fundamental for comparing such mechanisms with each other and with traditional approaches. As a first step towards deriving quantitative criteria for measuring self-organization effectiveness we argue that such measures would concern:

- an observed organizational *structure*;
- a *process* that produces and maintains that structure;
- a certain *function* or global goal that the system aims to fulfil at maximum using self-organization.

Furthermore, a self-organizing system can be studied from either local or global perspectives and more perspectives can be considered if self-organization spans multiple nested hierarchical levels. In all cases, measurements concerning the system structure, process or function can be taken at each level.

Structure based measurement focuses on assessing the system structure after it has been stabilized following a series of self-organization changes. Measures focusing on the self-organization process are related to the system's dynamics and its evolution over time. Finally, measurements focusing on the function that must be delivered by self-organization are related to how well the self-organizing system is able to fulfil its purpose. Therefore, measures in this case concern the characteristics of the problem the system is dedicated to solve and are similar to those used for performance assessment in classical systems.

https://doi.org/10.1017/S0269888905000494

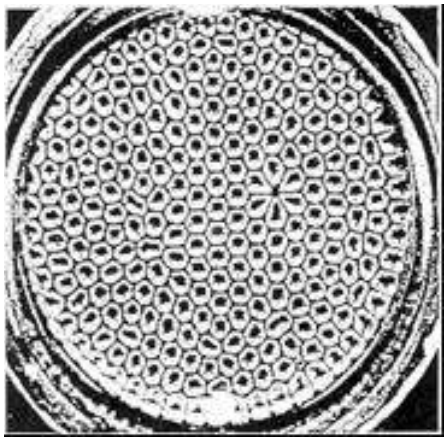

Figure 1 Bénard's cell

Examples of typical self-organization measures include:

- capacity to reach an organization able to fulfil the goal of the system as a whole once the system is started (success/failure/time required, convergence);
- capacity to reach a re-organization after a perturbing event (success/failure/time required);
- degree of decentralized control (central/totally decentralized/hybrid);
- capacity to withstand perturbations: stability/adaptability.

3 Emergence

3.1 What does emerge?

The concept of emergence was originally studied by the ancient Greeks and it appeared in various domains such as philosophy, mathematics, physics, thermodynamics, systemics and complex systems (Goldstein, 1999; Di Marzo Serugendo *et al.* 2006). Its description is often reduced to the phrase '*a whole that is more than its parts*' but there is currently no commonly accepted definition. Our aim is to analyse the emergence concept from two perspectives: one concerning the properties that when observed are sufficient to identify emergence and another focusing on those system characteristics that when all present the system have the capability to produce emergence. Subsequently, we attempt to provide an operational definition of emergence.

The outcome of emergence is often called *phenomenon* and it can be a particular organizational structure, framework, behaviour or function.[3] Let us illustrate this by a number of examples. In complex systems, the research very often centres on the emergent global dynamics of a whole system. It is usual in this approach to view the global properties of the system as emerging from the actions of its parts, rather than seeing the actions of the parts as being imposed from a dominant central source (Holland, 1995, 1998). The concept of the shortest path between an ant nest and a food source exists only for an observer that monitors the density of ants and pheromones in the biological environment of the ants.

In the Bénard's cells, the observer notices the emergence of a structure. The warmth is transmitted by a regular flux from the bottom towards the surface by conduction. When we continue to warm up, we move from equilibrium state towards an unstable state far from this equilibrium position. This phenomenon leads to the progressive apparition of chaotic circulation of the fluid that increases up until the boiling point. But before the boiling point there is a critical point at which circulation is insufficient to enable quick warmth dispersal inside the fluid. A

[3] The term function here does not refer to a concrete input-output mechanism such as the mathematical functions, but instead it refers to a piece of functionality the system provides towards fulfilling its goals, for example in an ant-based system the capability of rearranging the ant's path if it is obstructed or destroyed.

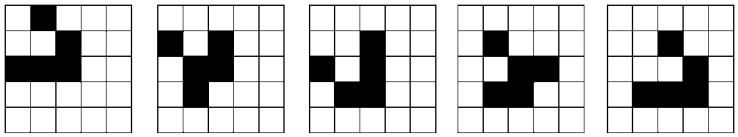

Figure 2 Game of life

self-organization phenomenon is observed in the system. The system leaves its chaotic state and provides a network of hexagonal running called *Bénard's cells* (see Figure 1). If the warmth increases, this phenomenon disappears.

Another example is provided by the game of life from Stanislas Ulam and John von Neumann (see Figure 2), where a moving pattern becomes apparent to an observer of the grid (Holland, 1998). The game of life is a grid with cells with a connectivity of eight. A cell can be alive or dead. The behaviour of a cell follows the three local transition rules:

- if a cell is alive and two or three of its neighbours are dead it becomes alive;
- if a cell is dead and two or three of its neighbours are alive it becomes alive;
- in other cases, the cell does not change its state.

When the system is running, the observer sees a glider, which corresponds to an emergent behaviour.

3.2 Properties of emergent phenomena

When a phenomenon appears, it is considered emergent if and only if there exists an observer capable of observing and characterizing it as such. Therefore, the role of the observer is fundamental in identification of emergent phenomena. Furthermore, to answer the question '*is a phenomenon emergent?*', an observer needs to analyse the phenomenon with respect to meeting certain criteria concerning the existence of essential properties of emergent phenomena. Several researchers have highlighted numerous such properties of emergent phenomena. In our point of view, an emergent phenomenon must exhibit six major properties. First, the phenomenon (a property, a structure, a pattern, a function) needs to be observable at a macro-level. When an observer perceives the phenomenon, they must be able to notice coherence and a strong relation, as well as a clear separation at the same time, with the constituent parts that cause the emergent phenomenon. Furthermore, the observer must be able to notice the following properties.

- *Novelty* (Lewes, 1875; Vijver, 1997). Novelty refers to the fact that although the resulting phenomenon is derived from a particular organization and interaction of the micro-level parts, it is radically different from their individual properties and it cannot be derived or predicted from them. In particular, to identify the emergent phenomenon different concepts and theories from those used to describe the micro-level activities are generally required.
- *Irreducibility* (Lewes, 1875; Ali *et al.*, 1998). Churchland (1984) defines emergence in terms of '*irreducibility of the properties of a high-level theory to properties of a lower level theory*'. Structures and/or functions appear at a macroscopic level and the observation of the component properties cannot enable to predict them (Atlan, 1993). For example, in complex systems, the complexity often results in features, emergent properties, which are properties of the system that the individual parts do not have.
- *Interdependency between levels* (Langton, 1990). In these systems there are at least two levels, a micro-level corresponding to the substrate where the emergent phenomena comes from and a macro-level enabling the observation of the emergent phenomena. The micro-level causes the emergent phenomena and the macro-level constrains the behaviour of the entities at the micro-level. Therefore, there is a strong dependency between the dynamics observed at both macro- and micro-levels.

https://doi.org/10.1017/S0269888905000494

- *Nonlinearity.* A chain of linear activities enables explanation and predictability of a collective phenomenon. On the other hand, an emergent phenomenon originates from nonlinear activities at the micro-level. Typical examples of nonlinear activities are loops of positive and negative feedback.

The aim of the work in self-organization is to be able to use the relevant concepts and techniques in engineering systems exhibiting emergence. Therefore, in the next section, in addition to properties that can be used to identify an emergent phenomenon, we describe characteristics that a system must possess in order to produce emergent phenomena.

3.3 System characteristics and emergent phenomena

The appearance of an emergent phenomenon implies the existence of a process that produces it. As a result, the system must support *dynamic processes* (Goldstein, 1999). Furthermore, since an emergent phenomenon is observable over time, some form of self-maintained and dynamic *equilibrium* is also needed. In addition, emergence occurs in a narrow possibility space lying between conditions that are either too ordered or too disordered. This boundary or margin is the *edge of chaos* (Kaufman, 1993), which is always far from equilibrium. Complexity theories are interested in studying this particular equilibrium using new forms of *attractors* (fixed point, limit cycle, strange attractor) explaining why emergence is not easily predictable. Near these equilibriums, a system has the ability to self-organize giving rise to an emergent phenomenon. Therefore, the appearance of emergence implies that the system also exhibits *self-organization* (see also Section 2). Finally, as mentioned previously, to produce emergence a system must be able to be described in at least two interdependent levels, a macro-level and a micro-level.

3.4 Operational aspects of emergence in computer science

For ten years new challenges appeared in engineering artificial systems including their ability to independently execute challenges, to adjust their behaviour due to various circumstances, to optimally manage their resources and self-repair when needed. Such requirements are commonly found in many new areas of research, such as autonomic computing, ubiquitous computing, emergent computation and ambient intelligence. Such systems exhibit emergent phenomena originating from numerous interacting components, for example, intelligent objects and agents. These components are characterized by dynamic changes in their population on run-time, inability to be regulated under a global control, an evolving and unpredictable environment and a functionally adequate behaviour[4] to be achieved in this environment.

Instead of attempting to eliminate emergent phenomena, it could be interesting to explore how this might be deliberately achieved and harnessed. That is, to elaborate on how to engineer artificial systems with desirable emergent properties. At the operational level, such systems must be build with the following properties.

- *Decentralized control.* There should be no entity which would control all the components of the system. In particular, each component must be autonomous and individually determine its behaviour.
- *Interacting part.* There are specific mechanisms to describe the emergent properties of a complex system. The mechanisms dealing with interactions can be classified in two types, *external* and *internal.* The external mechanisms enable modification of the system's behaviour to be initiated by its environment, for example, by imposing constraints, enforcing rules and delegating artifacts. The internal mechanisms are ways to change the interaction dimensions that are unfolded by

[4] A system is considered to have a functionally adequate behaviour when it realizes the behaviour expected by its environment, which can include end-users and is proven useful for its stakeholders.

https://doi.org/10.1017/S0269888905000494

processes within the system. Interaction is essential to this framework because the events of novelty and innovation within a system arise from the interactions of these agents with each other and with the environment.
- *Self-organization.* The system should be able to autonomously change its organization (see also Section 2).

The emergence in artificial systems is conceptually close to emergent computation defined by Forrest (1991) as follows:

- a collection of interactive agents: the process;
- an epiphenomenon produced by this process at the macro-level;
- a natural interpretation of this epiphenomenon as computation or computation results.

An operational definition is given by the SMAC team at IRIT (Capera *et al.*, 2003a; Georgé *et al.*, 2004). This technical definition of emergence has a strong computer science aspect and it is based on two points.

1. The subject. The goal of a computational system is to realize an adequate function, judged by a relevant user. It is this function, which may evolve during time, that has to emerge.
2. The condition. This function is emergent if the coding of the system does not depend in any way of the knowledge of this function. This coding has to contain the mechanisms allowing the adaptation of the system during its coupling with the environment, so as to tend anytime towards the adequate function.

Therefore, when we design an agent for a MAS, the code of the agent does not contain any knowledge of the collective function we want the MAS to compute. As a result, no agent controls the global system.

4 Case studies

Due to the proliferation of the Internet and the spread of mobile devices and progress in network and computer technologies, an increasing number of applications are suitable for using self-organization to fulfil their goals. Characteristic examples of relevant application domains include manufacturing control, network management, information retrieval and traffic management. There are currently two classes of approaches for designing self-organizing systems, the *bio-inspired* approach (Mano *et al.*, 2006) and the *social* approach (Hassas *et al.*, 2006). However, no approach is currently well-established or universally accepted as both approaches have weaknesses. Existing self-organizing applications differ widely in many respects including the self-organization mechanism employed, the method for designing component behaviour and the criteria for measuring and assessing the system's effectiveness.

To better understand self-organization, emergent properties and their underlying mechanisms, a framework to describe self-organizing systems has been proposed. This framework highlights the self-organization mechanism and the entities which self-organize. To gain insight on the strengths and weaknesses of self-organization mechanisms, three reference case studies and a number of representative self-organization approaches have been selected and implemented in the context of the self-organization in MAS Agentlink III TFG. The first case study concerns the *management of distributed computers.* It involves geographically distributed nodes that are composed of a huge amount of entities and it is well demonstrated that the control cannot be decentralized. The second case study is about the *manufacturing control.* This is a complex and a well-known problem in artificial intelligence. The system to be developed is open and medium scale. The constraints that can be added permit the problem to have several levels of complexity. The last case study is a problem-solving situation in biology. The problem is to find the *space conformation of a molecule.* This application has no environment, it can be classified in the small scale application and it is a complex problem.

https://doi.org/10.1017/S0269888905000494

The framework proposed to describe the designed systems is composed of the following six parts.

1. **The system description** presents the purpose and function of the system, behaviour at global level and the inputs of the system (the data to be given to the system at the initialization and/or during the functioning).
2. **The environment description** the system is situated in.
3. **The perturbations coming from the environment** is a list of the various types of stimuli the system can receive from the environment and how they can possibly vary over time.
4. **The entity description.** The designers have to provide a description of the characteristics of the system components (autonomous entities). This should include (but not be limited to) the following:

- goals;
- skills or capabilities (i.e. tasks that the entities can perform in the form of ⟨precondition, action, effect⟩);
- beliefs about the others entities;
- beliefs about itself;
- beliefs about the environment of the system.

5. **The interaction description** provides information on how the system entities interact with one another, i.e. by directly exchanging communication messages or indirectly by causing and observing particular changes in the state of their environment.
6. **The self-organization engine description** details the engine (mechanism) of self-organization processing:

- what triggers the re-organization?
- is this a local or a global mechanism?
- is this an internal or external mechanism?
- what are the steps (i.e. interaction sequences) involved in the various self-organization cases?
- when does the self-organization process stop?

Our final aim is to complete the framework given to describe the applications by a list of characteristics, which facilitate comparison of different applications:

- absence of external control (autonomy);
- decentralized control;
- dynamic operation (time evolution);
- global order endogenous;
- emergence from local interactions, emergent properties;
- dissipation (energy usage/far-from-equilibrium);
- instability (self-reinforcing choices/nonlinearity), parameter sensitivity;
- multiple equilibria (many possible attractors);
- criticality (threshold effects/phase changes);
- redundancy (insensitivity to damage);
- self-maintenance (repair/reproduction metabolisms);
- adaptivity (functionality/tracking of external variations);
- complexity (multiple concurrent values or objectives);
- hierarchies (multiple nested self-organized levels);
- simple local rules.

5 Case Study 1: managing computer networks

5.1 Case study description

This case study consists in a set of applications (processes) running on an open network of workstations, while dynamically satisfying the following set of constraints:

https://doi.org/10.1017/S0269888905000494

- dynamic load balancing of processors;
- minimal communications costs between highly interacting applications;
- optimal sharing of resources and data between process belonging to a same application;
- mutual exclusion between applications accessing a same set of resources.

The openness of the environment implies that during the processing, new workstations (respectively existing workstations) can join (respectively leave) the network. New applications (existing applications) could also be started (resume their activity). Furthermore, the network could be subject to perturbations originating from the environment, such as workstation breakdowns. In this case study problem, different self-organization goals can be considered, for example dynamic optimization of information routing, network topology and resource management.

Data available at the beginning. Network topology, application characterization, relations between applications, and relations to resources access.

Function of the system. The system has to provide the right services.

Perturbations. Breakdowns are generated along stochastic distribution including correlated occurrences of disturbances.

5.2 Approach based on holonic systems

System description. The artificial system representing the above described case study is a hierarchy of holons (holarchy) consisting of processing agents and nodes.

Input data. The case study considers a set of application (processes) running on an open network of workstations, while dynamically satisfying a set of constraints such as:

- dynamic load balancing of processors;
- minimizing communications costs between highly communicating applications;
- optimal sharing of resources and data between process belonging to a same application;
- mutual exclusion between applications accessing a same set of resources.

The input data are related to the network topology, node capacities, and number of applications running.

Expected results. The satisfaction of the set of constraints by the nodes and processes is the result of negotiations in order to assign to each process a node to be executed on. These negotiations modify the overall organization of nodes and processes without external control thus we can say that it is the result of a self-organization process.

Techniques used. An holonic perspective has been adopted to solve this case study and to prove the capacity of holarchies to self-organization. The notion of holon was originally introduced by Koestler (1976) to refer to natural or artificial structures that neither exist in whole or part in an absolute sense. According to Koestler, a holon must respect three conditions:

- being stable, it reacts when strong perturbations are applied;
- having the capability of autonomy, it is capable of self-organization to achieve its own goals;
- being capable of cooperation, it works in common projects according to shared goals with other holons and other layers of holons.

Entities that have to self-organize. Two main agent types have been identified, the processes and the nodes.

Environment description. The environment is composed of linked nodes and applications. Each node has a CPU which may execute processes and has resources available. The links between nodes allows communication and process migration.

Perturbations coming from the environment. The openness of the environment implies that during the processing, new workstations (respectively existing workstations) can join (respectively leave)

https://doi.org/10.1017/S0269888905000494

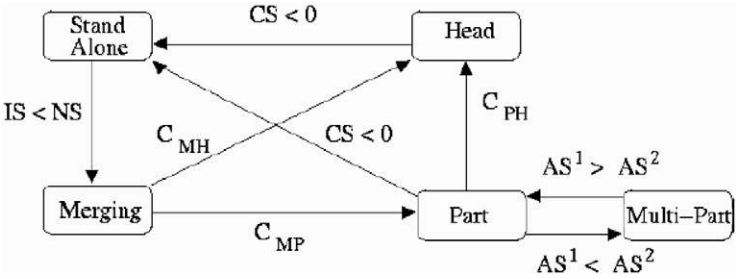

Figure 3 Role transition diagram

the network. New applications (respectively existing applications) could also be launched (respectively resume their activity). The network could also be subject to perturbations such as workstations breakdowns for example.

Entities description. Each entity composing a holarchy is called *holon.* We distinguish a particular kind of holons, which can not be decomposed into sub-holons, referred to as an atomic holon, classically called agent.

Process agent. A process agent owns a process and handles it in order to optimize the problem constraints. This agent can migrate to another node in order to better optimize the constraints. It can access information about the resources needed and the charge necessary to compute its process. It also knows the state of the current node and its neighbours.

Node agent. The node agent is situated on a specific node and communicates with the process agents. It knows the current load of its node and the one of its neighbours. The node agent is also in charge of managing the access to a resource (mutual exclusion). It also maintains an up-to-date history of modification of its resources. Before any process reaches its resources, it makes a saving point, to be able to re-generate the data in a coherent state in case an error arises. Every application may register with node agents managing the resources that it should use. This registration can be made either when the process agent is started, or for particular requests (e.g. edition of a file). The node will also maintain an up-to-date wish-list indicating the process agents able to reach the resources it manages. This list will enable agents to know with whom to group to generate shared accesses to the same resource.

Interaction description. Interactions between holons are defined by their *roles* in the holarchies. The role an holon plays may evolve as shown in Figure 3. We now sketch the interactions within each holarchy.

Communication holarchy. In this holarchy a process agent groups together with those who communicate the most with him. The regrouping phase proceeds in two steps: (a) a holon representing the group is created, process agents can leave this holon at any time; and (b) the link between process agents strengthens as their communications progress, process agents leave the holon only when the condition, which unites them, becomes false.

Execution holarchy. The execution holarchy possesses no regrouping criterion. The holons choose whether or not to grant to their representative, in another holarchy, a supplementary role as representative (Head) in this holarchy. They will then be considered by the outside and from the execution point of view as a single process.

Resource holarchy. In this holarchy the holons group together when they wish to read the same resource. There is a representative in charge of reading the resource. The read value is then redistributed among the various parts.

Self-organization engine. The satisfaction engine proposed in this work is inspired by the work on holonic systems of (Rodriguez *et al.*, 2004). The framework was formalized using the RIO Methodology (Hilaire *et al.*, 2002) and some self-organization properties have been proved (Hilaire *et al.*, 2005).

https://doi.org/10.1017/S0269888905000494

Holon satisfactions. In order to enable holons to dynamically change their roles, a satisfaction criterion based on the progress of its current task is defined. This satisfaction is called *instant satisfaction* and depends on the played role. When an holon is unsatisfied, two options are available. The holon may quit its current holon and look for another group, or it may join a second holon, while still remaining in the current holon. The satisfaction criteria guide the decision in this situation.

Satisfaction of a node agent. Concerning the node agent, its satisfaction depends on its personal load and the load averages of its neighbours. If its load is superior its satisfaction increases, otherwise it decreases. The introduction of the node satisfaction, *a priori*, accelerates the self-organization process for the load-balancing.

Satisfaction of a process agent. The satisfaction of a process agent depends on its holarchy. For an agent in the resource holarchy, the satisfaction depends on the time to obtain the desired resource. The more it has to wait, the more its satisfaction falls. For an agent in the execution holarchy, the satisfaction depends on the quotient between the allocated processor's time and the requested time and the satisfaction of the node where it is executed. For an agent in the communication holarchy, the satisfaction depends on the cost of the various communications that it has to realize. The communication cost depends on the time of packet routing.

Agents affinities. The affinity of a process agent is a boolean value depending on execution order/preferences, resources and communication threshold. The affinity of a node agent is defined by its immediate neighbourhood.

- More details on the mathematical formulas describing holon satisfaction as well as examples of self-organization processes in this holonic approach can be found in Di Marzo Serugendo *et al.* (2005).

5.3 Approach based on peer-to-peer systems

System description. Peer-to-peer (P2P) systems comprise of nodes connected in a network and functioning in a decentralized manner with the aim to increase the potential of the whole system to serve its 'customers' more efficiently and effectively. Each node has the potential to play any role from a set of roles (e.g. server, client, gateway or non-gateway), although the physical characteristics and processing capabilities of nodes may differ. These systems reside on the edge of *ad hoc* networks. In such a setting we study the self-organization of P2P systems at various levels such as communication, task and roles' levels (i.e. concerning the roles activated/deactivated in the system and their dependencies). At the communication level, the system organizes itself by computing the set of gateways through which all nodes communicate. At the task level, nodes exploit the communication infrastructure to establish groups of nodes that work jointly to achieve a goal or perform a task. At the roles level, agents are assigned roles either to achieve communication goals, or to perform tasks. In the latter case, roles are application/domain dependent. Each peer is considered to be an agent that may work in conjunction with other peers to serve a systems' 'clients'.

Environment description. Each node has a number of neighbours. The number of neighbours of a node may differ from time to time. We may consider that the whole system is 'surrounded' by clients that may request 'goods' (e.g. data, information items, meta-data, computing resources or just the routing of messages).

Perturbations coming from the environment. We distinguish the following perturbations:

- a new node joins the system;
- a node leaves the system;
- a node receives a request;
- a new communication link is established/destroyed.

https://doi.org/10.1017/S0269888905000494

Entity description. The functions of each node are realized by an agent. Agents, being minimal are able to:

- communicate with neighbours;
- set their communication status (gateway or no-gateway);
- form/propagate response/request messages;
- lock resources to clients.

Agents possess knowledge about:

- the status of their neighbours (e.g. roles that they play);
- routing of requests for locating 'goods';
- availability of its resources;
- availability of others' resources (e.g. using an aggregation method that allows it to apply a searching method—no node has complete knowledge of the system).

Furthermore, beyond minimal agents, we may consider agents that have more advanced abilities concerning:

- means-end reasoning (planning for achieving goal states);
- group decision making;
- deliberation (individual and social) (assessing options and deciding which one to pursue);
- collaboration with peers towards achieving a goal or performing a task.

Interaction description. Agent send messages directly to other agents.

Self-organization engine. At the communication layer, agents gather information of their one- or two-hops away nodes to determine their gateway status. This is done in a totally distributed way and dynamically, the nodes may change their status (role) according to the perturbations from the environment.

At the task layer, agents that receive clients' requests try to locate agents that match capability requirements towards serving the request. Agents that fulfil the capability requirements are determined to play specific task-dependent roles. This is a dynamic process, due to perturbations from the environment the system may re-organize itself by employing new roles that a new set of agents may play.

5.4 Approach based on routing and delegation concept

System description. The solution is freely inspired by routing techniques. We expect good properties of these techniques to be inherited here. However, such conjectures have to be pragmatically verified by implementing and experimenting what we propose on a shared testbed. This proposal is at the design stage only, without any underlying implementation. An 'agent' is associated to every processor present in the network and the key feature is to design a way to route new computational needs. By routing, we mean that each agent may commit to a task and/or delegate it, i.e. 'route' the task to itself or neighbours. Therefore, it needs to maintain preferential delegation targets within its direct neighbours. It is conjectured that this proposal leads to a self-organization result, since it consists in a constant adaptation of delegation trails and delegation trails are considered as an organization between agents. In addition, when a new computational need is inserted at a specific point in time, its result must arrive at some recipients and we consider the problem is pretty close to the routing issue with the following constraints added:

- the task must be computed on the way;
- some special resources must be on the way.

Fixed agents have a strategy to keep memories of good and bad direct-delegation experiences. This memory determines the delegation activity through the network that then constitutes the

https://doi.org/10.1017/S0269888905000494

organization. Here we described a simplified scheme where only atomic computational needs are inserted.

Environment description. Each agent perceives only a part of the environment:

- messages from others are received via the network interface;
- the local processor's schedule which contains atomic tasks.

Each processor executes the following loop:

1. unqueue next task with needed resources available;
2. execute it;
3. ask the agent to send result to the recipients list, i.e. where the result is needed in the network.

Perturbations coming from the environment. The following kinds of disturbances are considered:

1. addition/removal of needs;
2. addition/removal of computers;
3. communication cut;
4. resource vanishment.

Entity description. The agent here simply maintains a delegation willingness table that can be interpreted as its belief about delegation benefit. At the message level, special treatment is applied to incoming results and needs, in addition to maintaining routing tables to agents and resources.

Interaction description. A message between agents is a tuple consisting of:

- *sender* contains the sender identifier;
- *type* contains either: insertion, if the message contains a task to execute, or result, if the message contains the result of the task;
- *task* contains the task description;
- *result* contains the result if computed;
- *recipients* contains the list of all recipients (identifiers) for the result.

Self-organization engine. The 'engine' consists of two steps, one achieved at delegation time and the other when a result is perceived.

Details of the relevant algorithms can be found in Di Marzo Serugendo *et al.* (2005).

5.5 Approach based on stigmergy mechanism

System description. The solution proposed here is inspired by ant foraging behaviour; mobile agents, representing ants, permanently roam the network performing management tasks. Electronic pheromone guides mobile agents' behaviour.

Input data. Criteria to satisfy are:

- load distribution;
- membership of processes, for example same application, strongly communicating applications and competitive applications.

Expected results. Emergence of spatial aggregate of processes on a set of processors, satisfying the specified criteria (a spatial organization).

Self-organization. The result is obtained just by applying local rules for behaviours, without any external control. It is expressed as a situation where the systems dynamics guide it to reach an attractor where the specified criteria are balanced. The attractor corresponds to the spatial organization of the processes in such a way that the multiple criteria are satisfied.

Techniques used. The approach uses the stigmergy mechanism, it is an ant-like based approach (foraging behaviour). Multiple electronic pheromones represent the different criteria to be

https://doi.org/10.1017/S0269888905000494

satisfied. Ant-like (mobile) processes agents move over the network following the gradient of their specified pheromone (Foukia & Hassas, 2004).

Entities which have to self-organize. Agents associated to processes.

Environment description. The environment is a dynamic network of processors (a dynamic graph).

Perturbations coming from the environment. Considered perturbations are:

- processor breakdowns;
- sudden network overloading;
- arrival of new nodes (processors);
- departure of existing nodes (processors).

Entity description. Ant agents are of the following form.

- Agents have no individual goal.
- Agents perform tasks of the following form:
  (if perception of an appropriate pheromone field, follow the gradient field with a high probability, reinforce the pheromone amount);
  (if no perception of an appropriate pheromone field, move randomly, no effect).
- Agents have no beliefs about other agents, about themselves, or about the environment.

Interaction description. Interactions are mediated by the environment (pheromone field).

Self-organization engine. The engine is based on the self-catalytic mechanism of ant-like foraging behaviour:

- perturbations of the environment trigger re-organization;
- the mechanism is global to the system;
- re-organization is internal to the system;
- interaction sequences are; pheromone diffusion, evaporation, enrolling mechanism (self-catalytic behaviour);
- the self-organization process stops when the system is completely stable, and no perturbation occurs.

5.6 Characterization of the self-organization approaches

In order to compare the different approaches, we will now consider the different characteristics listed in Section 4, point 6.

Holarchies. The holon approach clearly demonstrates all the characteristics, except:

- emergence, multiple equilibrium and criticality need to be tested to actually be demonstrated;
- endogenous global order, redundancy and self-maintenance there is a conjecture in favour of these characteristics;
- instability is dependent on affinity and satisfaction functions;
- dissipation is not present in this approach.

Peer-to-Peer. The P2P approach clearly demonstrates self-maintenance, adaptation and hierarchies. The remaining characteristics are present in the following way:

- instability is not demonstrated in this approach;
- criticality cannot be determined;
- emergence comes from the fact that there is no node that has complete knowledge of the systems, the tasks performed and the network topology;
- dissipation comes from the fact that new requests, new nodes and communication links make the system behave in a dynamic way;

https://doi.org/10.1017/S0269888905000494

- multiple equilibria arise from the fact that each node may play more than one role concurrently;
- redundancy occurs through continuous update of node status;
- complexity occurs in case of defined requirements;
- rules are local essentially at the communication layer.

Routing. The routing approach clearly demonstrates an absence of external control and simple local rules. The remaining characteristics are present in the following manner:

- endogenous global order, emergence, dissipation, instability, multiple equilibrium, criticality and adaptation can not be proved as present;
- redundancy and self-maintenance are not provided by this approach;
- decentralized control occurs as decisions are taken locally and no agent has a special behaviour;
- dynamic operation is inferred as it is conjectured that the solution exhibits constant adaptation through organization.

Stigmergy. The stigmergy approach demonstrates all the characteristics, except hierarchies.
A detailed table describing these 4 approaches against characteristics can be found in Di Marzo Serugendo *et al.* (2005).

We observe that the routing and delegation approach only clearly demonstrates a few characteristics. The others are difficult to assess. The other three approaches demonstrate most of the characteristics, the stigmergy approach being the one that clearly demonstrates the majority.

As a first conclusion, it seems that the more the approach is naturally-inspired the more it demonstrates the expected characteristics. In these examples, the routing and delegation approach is clearly an *ad hoc* artificial mechanism for networking, but not naturally-inspired. Similarly, even though the P2P approach ably demonstrates most of the characteristics, it demonstrates less characteristics than the holon or stigmergy approaches. However, the routing and delegation approach has been not implemented and no experimentation have been realized. This explains the difficulty of whether or not to assess a given characteristic.

6 Case Study 2: manufacturing control

6.1 Case study description

The benchmark comprises the coordination and control of the internal logistics of a manufacturing department. Finished parts have to be delivered against a given due date for assembly in another department. Parts are transported in containers. Typically, a container holds about ten parts, but this may vary. The system comprises of a grid of container storage spaces, distributed across the manufacturing department. The system also comprises of ten workstations with varying properties and capabilities. Workstations have two or three locations at which a container can be placed. An operator picks parts out of one container, processes them in a pipelined fashion on the machines in their workstation and places processed parts in another container at their workstation. An automated transporter, moving over rails, normally transports the containers. It can carry maximally two containers at any given time. The parts enter the system in a container through the storage system. The system produces a mix of products, imposing varying loads on the workstations. The transport system has sufficient capacity on average but intermittently experiences rush hour complications. Operators have varying qualifications (having received training in function of shortages that occurred historically). The operator schedule is given.

Simplified hypothesis. All the parts in a container are the same. A container must be manipulated by several given workstations according to the process the parts have to follow.

https://doi.org/10.1017/S0269888905000494

This process consists of a series of jobs to be done and it can be known by the container. In a workstation, we need to have one empty container in which to put the treated part. A workstation consists in N given machines able to perform N tasks. An operator is affected to a workstation in function of their capabilities and had constraints applied, such as their breaks, holidays, or work duration. The operator can be ill and not available.

Data available at the beginning. The system has to start from an initial description of the factory:

- number of workstations;
- description of the workstations capabilities;
- number of operators;
- description of these operators in terms of capabilities and constraints;
- number of containers;
- the process associated to each container;
- data files on order arrival of containers.

Function of the system. The system has to find the right organization of these elements in the factory to have an efficient functioning.

Perturbations. Machine breakdown, process failures, operator unavailability, operator switch (other skills), process time variations, order arrival, rush order arrival, etc. Breakdowns, failures are generated along stochastic distribution including correlated occurrences of disturbances.

6.2 Approach based on ant algorithms

System description. The system proposed has two levels; the top level is a holonic architecture composed of product holons, resource holons and order holons, and the low level is composed of ant agents produced by the resource holons in order to find the expected available resources for the different orders.

Input data

- The manufacturing system is composed of:
  - workstations (type, number, location, capabilities);
  - storage system (type, number of spaces, location of spaces);
  - transport system (type, connections, capabilities);
  - diverse operators skills and availability.
- Product type definitions: recipes.
- Order book definition: arrival time and product type.
- Manufacturing scenario:
  - break down statistics;
  - process failure statistics;
  - process time variation statistics.

Expected results

- Effective supervision of the manufacturing activities.
- Availability of reliable short-term prediction of the state of the production system, usable to plan secondary activities.
- Minimization of the required workforce to perform the work on time.

Self-organization

- None of the agents can have a global view or data model.
- No centralized supervisor is allowed.

https://doi.org/10.1017/S0269888905000494

- It is essential not to expose individual software entities (agents) beyond their own manageable and stable scope. For instance, an agent managing a workstation is reusable wherever such a workstation is present in whatever manufacturing system. Software maintenance is only required when the workstation itself changes and the maintenance effort should be proportional to the chance. Yet, the interactions amongst such agents yield an effective manufacturing control that anticipates the immediate future as a means to coordinate the activities.

Techniques used

- Stigmergy;
- a holonic system as a specific agent society structure;
- delegation whenever something is outside an agents' scope;
- low and late commitment (but early preparation when feasible/doable).

Entities which have to self-organize

- Orders amongst each other;
- orders with resources.

Environment description. The environment is currently implemented as a MAS, following PROSA architecture (Can Brussel *et al.*, 1998). Analysis shows that resource agents can be split into an environment entity (choice-free) and an accompanying (pure) resource agent (handling choices and decisions concerning the resource). The implementation environment is a software toolkit/library on top of Java/Eclipse.

Perturbations coming from the environment. Considered perturbations are:

- machine breakdowns;
- process failures: possible rework and/or rush orders
- test outcomes: possible rework and/or rush orders
- late deliveries;
- order cancellations;
- rush orders;
- variable process duration;
- system reconfigurations (new equipment, equipment removal);
- degrading equipment and inverse;
- operator (un)availability.

Entity description. The resource holons are able to accept or reject tasks assigned to them. The product holons realize the task of production. The order holons are responsible for the scheduling, deadlock handling, progress monitoring and managing the process on a resource (start, suspend, resume, abort, stop). The ant agents are generated to find the available, adequate resources for a given order.

Interaction description. Interactions are either direct P2P messages between agents, or indirect communications through stigmergy.

Self-organization engine. There is a self-reinforcing process that builds load forecasts for resources and route/processing forecasts for orders.

1. Orders inform the resources about their intentions (make the necessary bookings) allowing these resources to predict their near-future loads;
2. Resources inform ant agents acting on behalf of the orders about the expected availability of processing capacity, duration, result, etc. allowing these orders to optimize and select intentions. When the orders inform the resources, the forecast accuracy of the resource improves, allowing the resource to give more accurate answers to queries by the orders, which improves the forecast accuracy of the order, informs the resources more accurately about its intentions and thus

https://doi.org/10.1017/S0269888905000494

closing a loop that improves accuracy until uncertainty, noise and disturbances start to impose an upper bound.

6.3 *Approach based on cooperative agents (AMAS Theory)*

System description. The system will be developed according to the Adaptive Multi-Agent System (AMAS) theory. This theory, based on self-organizing MAS, enables the design of systems in which agents only pursue a local goal while trying to keep cooperative relations with other agents embedded in the system. The agents are cooperative in the sense that they have to anticipate non-cooperative situations in order to avoid them. But also when they encounter non-cooperative situations they have to act to return to cooperative interactions. From the designer point of view, to find the non-cooperative situations in a MAS and to remove them can be viewed as an exceptions treatment in classical programs. The different agents to design are those representing operators and containers. The workstations are considered as resources disseminated on a grid. Agents have to move on this grid to find the right partner and right resources (an operator has to find a container on a workstation and a container has to find an operator on a workstation). The result of this problem solving gives rise to a given organization on the grid.

Input data. Input data that can be found in the system description are the following:

- the workstations' properties and capabilities;
- the operators' qualification and schedule;
- the containers (capacity, location, contents), we suppose that all containers have heterogeneous parts;
- the parts (characteristics, process, state), we do not consider them separately from the container;
- the automated transporter (location, contents, properties), we do not consider it in the first approach of the problem;
- some global constraints, we do not take them into account in a first approach;
- human-handled carts (location), we do not take them into account in this first approach.

Expected results. The aim of the system is to maximize the outputs (the number of containers processed) of the factory while minimizing the production time. The global solution must respect the time constraints given to all containers.

Self-organization. During the process, there are many perturbations coming from unexpected events (machine breakdown, process failures, operator unavailability). Faced with such perturbations, the workstations and the containers have to reorganize their activities without an external centralized control.

Techniques used. We use cooperative agents (according to the AMAS theory) (Capera *et al.*, 2003a, 2003b; George *et al.*, 2004) to solve this problem. In this approach, we want to build artificial systems having a coherent collective behaviour (Bernon *et al.*, 2004) (such systems are termed '*functionally adequate*') whereas agents only seek to reach an individual objective. To obtain such systems every agent is autonomous, has a local view of its environment and follows a classical lifecycle (perceive, decide, act) while keeping cooperative relations both with each other and with its environment. Definition of cooperation is not a classical one, it relies on three conditions: (1) all perceived signals must be understood, (2) the information provided must be useful for the agent's reasoning and (3) this reasoning leads to useful actions towards other agents. So, when an agent perceives a cooperative situation it executes the function for witch it has been designed and when the agent perceives a non-cooperative situation, it acts in the world in order to come back to a cooperative state by a self-organization process.

Entities that have to self-organize. Containers with parts they contain (the number of parts in a container can be null) and operators have to self-organize in order to get an organization allowing an efficient (optimized) supply chain. In our approach, workstations are only considered as resources in the system, they do not have a goal to pursue and are not autonomous.

Environment description. The environment of the system is only composed of the factory manager.

Perturbations coming from the environment. In this case study, several perturbations can be taken into account, such as process failures, process time variation, order arrival, rush order arrival, unavailability of a workstation (operator illness, strikes, machine breakdown), etc.

Entity description.
We have two different types of agents in this case study: container agents and operator agents, which have the following characteristics.

Individual goals

- Container agents: each container has to find a right operator on a right workstation according to constraints (time, etc.).
- Operator agents: each operator has to find a right container on a right workstation according to constraints.

Agents' perceptions. Each agent has a local view of the system according to its location.

Agents' skills or capabilities

- Container agents: a container agent is able to determine which type of workstations can process the parts it contains to find workstations that has the right operator to book a workstation to have a partnership with an operator agent.
- Operator agents: an operator has its own abilities.

Beliefs about other agents or acquaintances. At the beginning agents (containers or operators) do not have knowledge about others. When they encounter other agents, they learn about them (with a limited size memory).

Beliefs about themselves.

- Container agents: a container agent knows the process needed to treat the parts it contains.
- Operator agents: an operator agent knows its capabilities and its availability.

Beliefs about the environment of the system. Agents do not have any knowledge about the environment of the system.

Interaction description. In a workstation, agents can communicate directly by message sending. They also can communicate in an indirect way (by putting booking intentions in the environment).

Self-organization engine. The engine (or mechanisms) of self-organization is the cooperation (see AMAS theory). Every agent tries to find a workstation and to have a relevant partnership (container–operator) in the organization. Re-organization is triggered by the fact that an agent (operator or container) is not satisfied because its constraints are not satisfied. It then tries to find a partnership. This is a local criterion. This mechanism is internal to the system. Interaction sequences involved are dependent on the fact that every agent (operator or container) has to find a suitable workstation and partnership that satisfies its constraints. The self-organization process stops when every agent has found a partnership in a particular workstation with a minimum constraint relaxation.

6.4 Characterization of the self-organization approach

We have compared the different approaches against the same characteristics as case study 1.

Stigmergy. The stigmergy approach clearly demonstrate all the characteristics, except:

- criticality and self-maintenance cannot be determined without further investigations;
- hierarchies are not present in this case study;
- instability would need a social control to be provided.

Cooperative agents. The cooperative agents approach demonstrates all the characteristics, except:

https://doi.org/10.1017/S0269888905000494

- instability, criticality, self-maintenance and hierarchies are not present in this case study;
- redundancy is present provided there are more workstations than necessary in order to face machine breakdowns.

A detailed table describing these two approaches against characteristics can be found in Di Marzo Serugendo *et al.* (2005).

The stigmergy and cooperative approaches provide the characteristics in a very similar way. This result can be explained by the fact that the stigmergy approach is naturally inspired and the cooperative approach has been thought for self-organization and emergence. Note that the concepts agentified and implied in the self-organization mechanism are not the same in the two solutions. In the stigmergy approach, ant agents represents orders and are in charge of finding the adequate and available resource under given constraints. In the cooperation approach, the agents represent operators and containers and they have to negotiate each other with local information in using direct or indirect communication to reach a global solution. The solution provided by stigmergy provides the shortest paths to reach all resources needed. These shortest paths can be qualified as emergent. The solution provided by cooperation corresponds to a structure that is the organization between operators, containers and workstations and the optimal organization is an emergent phenomenon in this application.

7 Case study 3: space conformation of molecules

7.1 Case study description

The content of this case study deals with computational biology in order to find new drugs in pharmacology. The case study concerns the modecule level and the aim is to find the space conformation of a molecule that minimizes its potential energy. This section ends with the proposition of a solution using self-organization based on cooperative agents.

Search for drugs. Structural biology is interested in the relationship between the structure of molecules and their biological function. In a general way, questions more often depend on the field of pharmacology. Thus for a known membrane receiver, finding a ligand, i.e. a complementary medicamentous molecule is problematic. It is the docking problem, for example, the whole set of mechanisms and interactions that play a part during molecular complexes' formation. Simulation study of the docking, rather than an experimental approach, has a long history and most algorithms are used daily in the academic environment as well as in pharmaceutical laboratories. Molecular docking is the centre of practical applications like protein engineering, drug design and screening of potentially relevant molecules. It has thus already contributed to new ligands design for anti-AIDS and anti-cancer agents and for the treatment of the diabetes. To obtain significant progress, we need to develop new methods to facilitate the techniques of docking by improving our capacity to understand and analyse docking interactions and to develop assumptions for which molecules could have better interactions. Knowledge of the process of space conformation of interacting molecules (folding) is essential and is the subject of this case study.

Atomic interactions. A molecule is an assembly of atoms by covalence connections. In fact, strong connections define quasi-stable inter-atomic distances (in the region of the Angstroms) for a particular pair of atoms. Such strong connections are supplemented by strict directional impositions due to weak connections. These weak connections, to have an unspecified effectiveness, can act only at short distances, and in great number. A molecule's space conformation results from the inter-atomic distances defined by the strong connections and from interactions due to weak connections. To simplify, a space conformation consists in minimizing the residual weak energy on the whole molecule, while respecting constraints of strong connections. In inter-atomic relations, the fundamental role is held by outer-shell electrons, pertaining to the outermost orbital of bonding atoms. A union between atoms results from modifications of the distribution in space of their outer-shell electrons. A covalence connection is a connection due to a bilateral pooling of electrons.

https://doi.org/10.1017/S0269888905000494

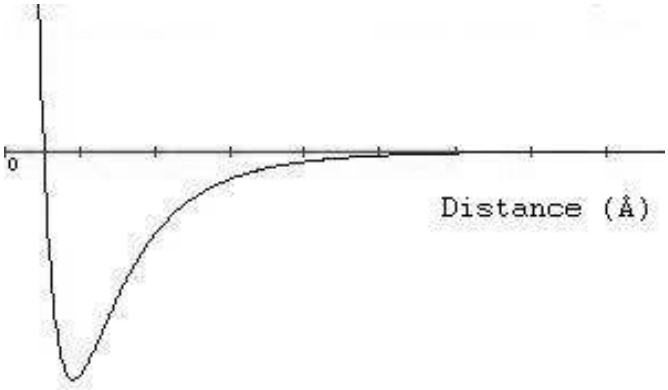

Figure 4 Resolution of space conformation

When two atoms approach one another, an attraction between the nucleus of the one and the electrons of the other (and vice versa) appears starting from a certain distance (~1 nm). This attraction sensibly increases when the distance separating the nuclei is about 500 pm. But, starting from this distance, electronic obstruction and repulsion forces between charges of the same sign tend to compensate for the attraction. A balance is established at a defined distance, characteristic of each atom called the Van der Waals' ray. This value is, for example 120 pm for hydrogen, 140 pm for oxygen and 285 pm for sulphur. Van der Waals' forces induce a weak energy (about 1 kcal/mol). Thus they are interesting for the cohesion of molecular structures only if they are numerous and applied between neighbouring atoms. The energy function for weak connections follows a law as indicated on the curve shown in Figure 4, the x-coordinate represents inter-atomic distance (in Angstroms).

Constraints of the space conformation resolution. Molecule's space conformation research (to find the minimum of residual energy) is an NP-complete problem and hence the solution cannot be found in a polynomial time according to the number of atoms in the molecule. Resolution of this type of problem must satisfy the following constraints:

- calculation must be distributed (physically or logically) in such a way that the local algorithm does not have all the information about the molecule;
- calculation must be emergent, i.e. local calculations are unaware about information on the total residual energy of the molecule;
- ideally we could show (by experimentation or formally) that the algorithm finds a solution in a time limited by a polynomial (depending on the number of atoms).

The system has to start from an initial structure of the molecule, a given space conformation. The input data is a description of a molecule in the protein data bank (PDB) format. A PDB file includes as many lines as atoms in the described molecule, each line provides information about an atom, for instance, its name and its identification number in the molecule. The data available at the beginning are the distance between the atoms with strong connections and the energy functions between two atoms. In a first step, there is no need for perturbations. In a second step, when the problem will be solved, the system can be perturbed by moving the location of one or more atoms in the molecule. In any case, the system has to find the right space conformation, a conformation for this molecule that minimizes its global potential energy.

7.2 Approach based on cooperative agents (AMAS Theory)

System description. In the system proposed, each atom is an agent. Agents have local perceptions, each agent perceives all atoms in a given area. This area concerns its covalent atoms (strong

https://doi.org/10.1017/S0269888905000494

interactions) and the atoms with which it has non-bonded interactions (weak interactions). We call the agents it perceives its neighbours.

Environment description. The environment of the system (the molecule) is composed of the physical laws that rule the interactions between atoms, these laws are static during a session. The atoms have to self-organize in order to get an organization that reflects the spatial conformation we want to obtain and the molecule is a self-organizing system because its inner constitutive atoms (the agents) are going to modify their relative location.

Entity description. The goal of an agent is to reduce its potential energy and possesses the following skills. It is able to modify its location within the molecule, it moves. It knows how to evaluate its potential energy and it can evaluate the potential energy of its neighbours, for a given move. It knows the following information about its neighbours:

- the position and the current energy level of other agents in its perception area;
- their beliefs about themselves;
- an agent knows its position, its neighbours, its covalent atoms;
- their beliefs about the environment of the system (so without the other agents);
- the physical laws concerning the atoms (coming from the environmental database).

Note that an atom ignores the value of the global energy of the molecule.

The techniques used are based on the AMAS theory (in which cooperation is the local criterion that makes components of a system reorganise, see `www.irit.fr/SMAC`) (Capera *et al.*, 2003a; George *et al.*, 2004) and involve cooperative agents. These agents are autonomous entities that follow a classical lifecycle (perceive, decide and act) and have a social attitude based on cooperation. The cooperation has to respect three conditions: (1) all perceived signals must be understood, (2) the information provided must be useful for the agent's reasoning, and (3) this reasoning leads to useful actions towards other agents. A cooperative agent knows how to detect situations that infringe these laws and acts to remove them. Those situations are called non-cooperative situations (NCS). When an agent detects a NCS, which can be called a failure at the local level, it tries to come back to a cooperative one. In addition, whenever possible, it always acts in a cooperative way, the cooperation guiding the local behaviour of an agent.

Self-organization engine. In the current problem, the engine of the self-organization process is the cooperation, an agent always tries to reduce its own potential energy and its worst neighbour's energy level (non-cooperative situation action).

The algorithm of an agent is:

```
While (True)
  Begin  Perceive the potential energy of all its neighbours
  Compute its own local potential energy
  Spatially move to decrease the worst potential energy while
   respecting the covalent bonds
End
```

The process at the agent level could not stop because an atom has no information about the value of the energy at the global level. The designer can choose to stop all the process when:

- the variation of the global energy is small (given an epsilon);
- or a number of simulation steps is reached;
- or requested by the user.

The system has been developed in 2003–2004 (`www.irit.fr/SMAC`) and cannot be compared to other systems based on self-organization mechanisms as currently no other attempt has been made to model space conformation of interacting molecules using this approach. Few algorithms have been created to tackle this problem only one organization, called folding@home, has proposed

https://doi.org/10.1017/S0269888905000494

a solution. Their technique is based on the space search distance covered, which is very large. To compute this distance they distribute a part of the computation on a huge number of computers. It is very difficult to compare our results with theirs as their time to solve the problem follows an exponential curve that is balanced by the number of machines. In contract, our computation time follows a linear curve enabling us to comment on the efficiency of the self-organizing solution. Besides, our results show that the AMAS theory enables us to tackle complex problems and consequently that self-organization is a new way to solve very difficult problems.

8 Conclusion

This paper provided an overview of the work done in the context of the Agentlink NoE TFG on self-organization in MAS. The paper focused on providing a common definition of the concepts of self-organization and emergence in MAS and the associated properties and characteristics. Furthermore, it described work done towards a systematic approach for the selection of self-organization mechanisms using a number of selected reference case studies and a set of evaluation criteria. The results revealed many similarities in self-organization mechanisms that can be used towards providing a systematic approach for developing self-organization in MAS. Furthermore, the need for systematic methodologies and tools for developing such systems has been emphasized and therefore work is planned to continue in these directions.

Acknowledgements

The authors would like to thank the following people who contributed to this paper by providing substantial input for the case studies descriptions or through fruitful discussions: Carole Bernon, Camille Besse-Patin, Markus Bongard, Christine Bourjot, Paul Burghardt, Valerie Camps, Davy Capera, Cristiano Castelfranchi, Vincent Chevrier, Joris Deguet, Michel Fabien, Eduardo Fernandez, Luca Gardelli, Nicolas Gaud, Pierre Glize, Bdelkader Gouaich, Zahia Guessoum, Salima Hassas, Vincent Hilaire, Tom Holvoet, Gordan Jezic, Mario Kusek, Gabriel Lopardo, Marco Mamei, Jean-Pierre Mano, Paul Marrow, Sebastien Picault, Eric Platon, Alois Reitbauer, Sebastian Rodriguez, Paulo Sousa, Paul Valckenaers, Paul Verstraete, Giuseppe Vizzari, George Vouros, Danny Weyns, Franco Zambonelli.

This work is partly supported by Agentlink III and by the Swiss NSF grant 200020–105476/1.

References

Ali, S *et al.*, 1998, The question concerning emergence: implication for artificiality. In Dubois, DM (ed.), *First Computing Anticipatory Systems Conference CASYS'97 Conference*, Liege Belgium, CHAOS.

Atlan, H, 1993, *Enlightenment to Enlightenment: Intercritique of Science and Myth.* Albany: SUNY Press.

Bernon, C *et al.*, 2004, Tools for self-organizing applications engineering. In Di Marzo Serugendo, G, Karageorgos, A, Rana, OF and Zambonelli, F (eds.), *Engineering Self-Organizing Systems, Nature-Inspired Approaches to Software Engineering* (*Lecture Notes in Artificial Intelligence, 2977*). Berlin: Springer, pp. 283–298.

Capera, D *et al.*, 2003a, The AMAS theory for complex problem solving based on self-organizing cooperative agents. In *International Workshop on Theory and Practice of Open Computational Systems (TAPOCS). Twelfth International IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2003)* Johannes Kepler Universitaet, Linz, Austria. Piscataway, NJ: IEEE Press, pp. 383–388.

Capera, D *et al.*, 2003b, Emergence of organizations, emergence of functions. In *AISB'03 Symposium on Adaptive Agents and Multi Agent Systems*, University of Wales, Aberystwyth. pp. 103–108.

Churchland, P, 1984, *Matter and Consciousness.* Cambridge, MA: MIT Press.

Di Marzo Serugendo, G *et al.*, 2005, Self-organization in multi-agent systems. Research Report IRIT/2005–18–R, IRIT.

Di Marzo Serugendo, G *et al.*, 2006, Self-organization and emergence in MAS: an overview. *Informatica Journal, Special Issue about Agentlink TFG* (in press).

Forrest, S (ed.), 1991, Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing network. In *Proceedings of the 9th Annual CLNS Conference.* Cambridge, MA: MIT Press.

Foukia, N and Hassas, S, 2004, Managing computer networks security through self-organization: complex system perspectives. In Di Marzo Serugendo, G, Karageorgos, A, Rana, OF and Zambonelli, F (eds.), *Engineering Self-Organizing Systems, Nature-Inspired Approaches to Software Engineering* (*Lecture Notes in Artificial Intelligence, 2977*). Berlin: Springer, pp. 124–138.

Georgé, J-P *et al.*, 2004, Making self-organizing adaptive multi-agent systems work—towards the engineering of emergent multi-agent systems. In Bergenti, F, Gleizes, M-P and Zambonelli, F (eds.), *Methodologies And Software Engineering For Agent Systems.* New York: Springer, pp. 321–340.

Goldstein, J, 1999, Emergence as a construct: history and issues. *Emergence* **1**(1), 49–72.

Hassas, S *et al.*, 2006, Self-organizing mechanisms from social and business/economics approaches. *Informatica Journal, Special Issue about Agentlink TFG* (in press).

Hilaire, V *et al.*, 2002, A mechanism for dynamic role playing. In *Agent Technologies, Infrastructures, Tools and Applications for E-Services* (*Lecture Notes in Computer Science, 2592*). Berlin: Springer.

Hilaire, V *et al.*, 2005, Formal specification of a holonic multiagent system framework. In *Intelligent Agents in Computing Systems—The Agent Days in Atlanta*, vol. to appear of *Lecture Notes in Computer Science.* Berlin: Springer.

Holland, JH, 1995, *Hidden Order: How Adaptation Builds Complexity.* Reading, MA: Addison-Wesley.

Holland, JH, 1998, *Emergence: From Chaos to Order.* Reading, MA: Addison-Wesley.

Kaufman, S, 1993, *The Origin of Order: Self-Organization and Selection in Evolution.* New York: Oxford University Press.

Koestler, A, 1976, *The Ghost in the Machine*, the danube edn. London: Hutchinson & Co.

Langton, CG, 1990, Computation at the edge of chaos: phase transitions and emergent computation. *Physica D* **42**(1–3), 12–37.

Lewes, J, 1875, *Problems of Life and Mind.* London: Kegan Paul, Trench, Turbner & Co.

Mano, J-P *et al.*, 2006, Bio-inspired mechanisms for artificial self-organized systems. *Informatica Journal, Special Issue about Agentlink TFG* (in press).

Rodriguez, S *et al.*, 2004, Towards a methodological framework for holonic multi-agent systems. In Omicini, A, Petta, P and Pitt, J (eds.), *Engineering Societies in the Agents World 3rd International Workshop, ESAW 2003, London, UK, October 19–23, 2003.* (*Lecture Notes in Computer Science, 3071*). Berlin: Springer.

Van Brussel, H *et al.*, 1998, Holonic manufacturing systems: PROSA. *Computers In Industry, Special Issue on Intelligent Manufacturing Systems* **37**(3), 255–276.

Vijver, GVD, 1997, Emergence et explication. *Intellectica: Emergence and Explanation* **2**(25), 185–194.