Contents

Chapter 1
Cooperation

Jean-Pierre Georgé and Marie-Pierre Gleizes and Valérie Camps

"Great discoveries and improvements invariably involve the cooperation of many minds"
Alexander Graham Bell

"Thank you for your cooperation and vice versa"
Eugene Ormandy

Abstract This chapter aims at providing the reader with a thorough understanding of the notion of cooperation and its use as a self-organising mechanism in artificial systems. As the complexity and scope of applications increase, the need for self-adaptation must be addressed by software engineers. This chapter describes why and how cooperation can be used for this. An intuitive understanding of the concept will be provided, as well as definitions. As computer scientists, the readers will be introduced to the translation of the concept in artificial systems through the Adaptive Multi-Agent Systems (AMAS) theory. The importance of adaptation and emergence will be presented, as well as how cooperation plays the role of the engine for self-organisation. Technically, a multi-agent system approach is used and the architecture of a cooperative agent in this theory is described.

For a concrete understanding of this approach, two case studies are described in detail. The first is a dynamic and open service providing MAS where all the providers and customers (the agents) need to be put in relation with one another. This relationship needs to be constantly updated to ensure the most relevant social network (by being cooperative one with another). The second is a multi-robot resource transportation problem where the robots (the agents) have to share the limited routes to efficiently transport the resources (by choosing cooperatively how to move). Each description focuses on how cooperation can be applied, what Non Cooperative Situation are for the agents and how it enables them to self-organise towards the adequate emergent function (and these concepts will also be explained).

Jean-Pierre Georgé and Marie-Pierre Gleizes and Valérie Camps
University Paul Sabatier - IRIT, 118 route de Narbonne, 31062 Toulouse Cedex 9 - France
e-mail: {george, gleizes, camps}@irit.fr

Objectives This chapter aims at providing the reader with a thorough understanding of the notion of cooperation and its use in artificial systems. In this chapter the reader will:

- understand cooperation both on an intuitive level and as a definition;
- see an illustration of cooperation in natural systems and understand its importance;
- learn about the AMAS (Adaptive Multi-Agent Systems) theory which states how to use cooperation as an engine for self-organisation, effectively building adaptive multi-agent systems;
- be guided as he will try to apply it in artificial systems and discover how it has been done in existing applications;
- see how cooperation can be used as a self-organising mechanism in artificial system to produce emergent functionalities.

1.1 Introduction

The new applications that software engineers have to develop become more and more complex (see "History and Definition" chapter). The different events encountered by the system cannot all be known at the specification phase of the system design. Therefore, designers need new approaches to design adaptive systems, i.e. enabling the system to adapt itself to unexpected events. One powerful way to obtain this is to rely on the emergence of the required functionalities in a given environment or context.

To obtain emergent phenomena, different methods or mechanisms have been studied by researchers (see the other chapters in this book). In the approach presented in this chapter, we assume that to change the function of a system, the system only has to change the organisation of its agents. For example, a common definition of agent organisation in the agent literature is that it is defined by the lines of communication of the agent components, the authority relationships between them, and the individual agent functionality. Based on this definition if any if these three aspects changes for some reason then the agent organisation also changes.The behaviour rules enabling this self-organisation are based on **cooperation** which is the heart of this emergence-based bottom-up approach.

Everybody has an intuitive understanding of what cooperation is about. Readers can see an illustration in Fig. 1.1. This chapter aims at giving a better understanding of this simple notion so that it can be applied in complex artificial systems when needed to improve their functioning or for facilitating their design.

Cooperation is classically defined by the fact that two agents cooperate if they need to share resources or competences [8, 33]. We add to this definition, the fact

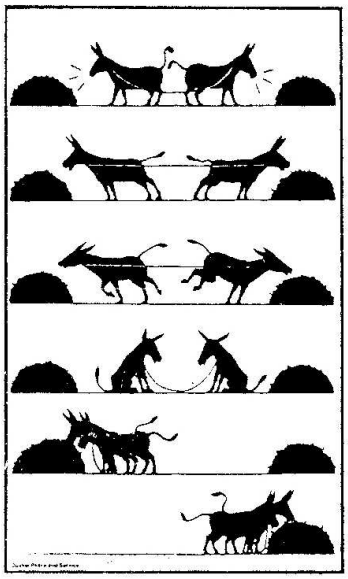

Fig. 1.1 Cooperation increases mutual benefit of parties involved. Even for mules !

that an agent tries on one hand, to anticipate cooperation problems and on the other hand to detect cooperation failure and try to repair these non cooperative situations [29]. To anticipate, the agent always chooses the actions which perturb other agents it knows as little as possible (another agent is perturbed when the action hinders it for its own goals or if the action results in any king of cost for it).

In this chapter, we present an approach to design self-organising systems based on multi-agent [33] systems and cooperation (see also the chapters of this book describing the ADELFE methodology). The next section clarifies the concept of cooperation by giving a background context, explaining related notions and defining it. In Sect. 1.3 and 1.4, the theoretical notions and the resulting technology constituting the adaptive multi-agent systems (*AMAS*) theory are expounded. In this theory, systems' self-organisation capabilities are based on the social inspired notion of cooperation. To illustrate that approach, two applications are then presented, one concerning a service providing multi-agent system and one elaborating on a multi-robot resource transportation problem. The focus is on the implementation of cooperative behaviours and how they enable the self-organisation which solves the problems constituting each application. Finally, the readers will be able to test and train their skills with practical exercises asking to implement cooperation in various situations.

1.2 Understanding Cooperation

This section will rapidly clarify the underlying concepts which support cooperation such as the notions of collective activity in social systems, interaction and communication. It will then give an intuitive understanding of cooperation and definitions illustrated by examples in natural systems, both animal and human. It will show that multi-agent systems are quite fond of cooperation, that computer science provides a more formal definition of cooperation for artificial systems and that it seems to be a promising mechanism for achieving self-organisation and emergence.

1.2.1 Underlying Concepts

As soon as an activity involves more than one lone entity, interaction is bound to appear eventually. This interaction can take multiple forms, from simple pushing, racing, giving something, to more elaborate exchanges, negotiations, deals, and finally complex organised social structures as seen with social insects or human groups. Communication (directly of indirectly through the environment) plays of course an essential role in enabling this interaction, as well as a representation of the others, their nature and their goals.

Computers and software are build as powerful Input/Output systems with well-known communication and interaction means when considering computer-peripheral or computer-computer interaction. Things get more complicated as complexity grows, entities get mode heterogeneous, resources get scarce, goals vary and the whole is of course expected to produce relevant and optimal results. At this point, an agent model facilitates further analysis of the systems. Ferber [8] produced a first clear interaction typology of situations depending on the goals of the agents (compatible or not), resources availability (sufficient for all or not) and competences of the involved agents (does each agent possess all its needed competences ?). This is summarised in Table 1.1.

A brief explanation of the different situation types is presented below:

- **Situation 1.** Since each agent is self-sufficient, there is no need for cooperation. Agents can still benefit from cooperating to get optimised results. Example: *"I can set up my tent on my own, but if you help me and then I help you, we might set both up before the storm. Or at least, there will be one already usable when the rain hits and we can share it. Isn't it nice ?"*
- **Situation 2.** The agents do not have all the competences needed to be self-sufficient. This is the classic case, for instance, where robots have to move boxes which are to heavy for one robot to carry. Two robots need to cooperate to move one box together, then another.
- **Situation 3.** Resources are insufficient and the agents have to share these resources. For instance, a one-way bridge situation is optimally handled when

	Goals	Resources	Competences	Situation type	Interaction Category
1	Compatible	Sufficient	Sufficient	Independence	Indifference
2			Insufficient	Simple collaboration	Simple cooperation
3		Insufficient	Sufficient	Cluttering	
4			Insufficient	Coordinated collaboration	
5	Incompatible	Sufficient	Sufficient	Pure individual competition	Antagonism (requires negotiated cooperation)
6			Insufficient	Pure collective competition	
7		Insufficient	Sufficient	Individual conflicts for resources	
8			Insufficient	Collective conflicts for resources	

Table 1.1 Interaction types depending on goals, resources and competences as done by Ferber

agents agree to wait to let some agents cross on one direction, then on the other and so on.

- **Situation 4.** Here both resources and competences are limited. As an example, we can mix both previous examples and have plenty of our robots needing to cross the bride in one direction or another (see also the application presented in Sect. 1.5.2).
- **Situation 5,6,7 and 8.** These situations have the same characteristics as the previous ones with one huge exception: the agents do not have the same goals. For instance, each robots aims at gathering *all* the boxes. They can start attacking each other, avoid each other if possible or, more rationally, negotiate. Such negotiation commonly originates from the belief that acquiring at least some of the available boxes is still preferable to acquiring none. These situations usually require that agents either actively communicate to reach an agreement, or have some means to exclude or punish agents which are not cooperating, at least if the aim is to reach a global optimum or to limit risk for an individual agent.

1.2.2 What is Cooperation ?

The reader cetainly already has at least an intuitive understanding of cooperation, for basically, one only has to look at their own everyday cooperation. The above table, explanations and examples also give an intuitive understanding. Dictionaries simply define it as:

Definition 1.1. Cooperation: the act of cooperating, or of operating together to one end; joint operation; concurrent effort or labor. [1913 Webster]

Definition 1.2. Cooperate: To act or operate jointly with another or others; to concur in action, effort, or effect. [1913 Webster]

Furthermore, the following explanation can be found on *Wikipedia*[1]: *"Cooperation is the process of working or acting together, which can be accomplished by both intentional and non-intentional agents. In its simplest form it involves things working in harmony, side by side, while in its more complicated forms, it can involve something as complex as the inner workings of a human being or even the social patterns of a nation. It is the alternative to working separately in competition. [...] cooperation may be coerced (forced), voluntary (freely chosen), or even unintentional, and consequently individuals and groups might cooperate even though they have almost nothing in common qua interests or goals. Examples of that can be found in market trade, military wars, families, workplaces, schools and prisons, and more generally any institution or organisation of which individuals are part (out of own choice, by law, or forced)."*

We can find a multitude of cooperation examples in nature and social systems, whatever the size of the group or its aims. A usual example is how an anthill manages to quite efficiently gather food, or how termites build complex structures using stigmergy [15]. The ant, by leaving pheromones on the ground when returning to the nest after finding food ensures that other ants looking for food have a better chance to quickly find food sources. Moreover, these pheromones accumulate and evaporate in a way that enables the emergence of collective patterns, for instance one can observe in specific laboratory set-ups that after a while all ants take the shortest path [4].

The most notorious and well studied social experiment involving cooperation is the *prisoner's dilemma* [2]. Two prisoners face specific different sentences depending on if they denounce each other or not as having committed the crime, and they can't speak with each other. If they both denounce each other, they face heavy sentences, if they both deny, they face light sentences, and if only one denounces the other, he walks while the other pays for the crime alone. The rational action is to both deny (they cooperate) so that as a team, the cost is small. A more realistic set-up is the *iterated prisoner's dilemma* where the situation occurs more than once and where trust and reputation enter the game.

It is important to note that not all activity involving at least two agents can be seen as cooperation. All prejudicial activity of one agent on another is of course the contrary of cooperation. But even a fully altruistic behaviour is not cooperative in the sense that when some agents sacrifice themselves for the others, it might not be the best for the group as a whole. Cooperation implies that both parties benefit from the activity, at least in the long run.

It is also interesting to note that agents can act cooperatively but with very different social strategies. The most simple strategy, which could be called *benevolent cooperation*, is to suppose that every agent is also cooperative and so, the agent

[1] www.wikipedia.org

[2] http://en.wikipedia.org/wiki/Prisoner's_dilemma

always spontaneously cooperates when asked. This is an assumption which can easily be taken when building an artificial system where each agent is designed to be cooperative. Open heterogeneous systems where agents are free to enter and no normative structure exists to enforce cooperation forces the agents to be more prudent. The agents rely on negotiation, trust, reputation and so on, leading to a *tit-for-tat cooperation*.

1.2.3 Using Cooperation in Artificial Systems

Cooperation was extensively studied in computer science by Axelrod [1] and Huberman [19] for instance. "*Everybody will agree that cooperation is in general advantageous for the group of cooperators as a whole, even though it may curb some individual's freedom*" [17]. Relevant biological inspired approaches using cooperation are for instance *Ant Algorithms* [7] which give efficient results in many domains.

Multi-Agent Systems [33] are a perfect paradigm to apply cooperation: several to numerous possibly heterogeneous entities, each with its own local view, knowledge and goals striving to achieve a collective function as effectively as possible. The need to cooperate is inherent in these kind of systems, not surprisingly since they are essentially a social systems metaphor. All imaginable means to implement cooperation can be used since the only limit is what functionality a designer can put into an agent. Well known and studied mechanisms that can be used to realise cooperation include negotiation protocols, trust, reputation, gossip, normative structures, stigmergy, etc. (refer to corresponding chapter for more detail).

The next section will give a more formal and specific definition of cooperation when used in MAS. It is intended as a guide for the design of the behaviour of the agents. The main aim of this guide is to ensure that the system as a whole, by having the agents locally and cooperatively self-organise, behaves as expected (or as best as it is possible) in any situation.

1.3 The Philosophy of the *AMAS* Theory

This section presents an informal and intuitive approach of the AMAS theory in order to show the process followed to construct this theory and to highlight the motivation that lead to the development of this theory. First, the objectives of the AMAS theory are expounded. For this, we start by presenting examples of the kind of complex adaptive systems we want to develop and the characteristics tackled by this theory. Then, three main concepts of the AMAS theory are presented. The first concerns the adaptation and explains how a system can adapt. The second concept focuses on emergence and describes why we can qualify the global behaviour of the artificial system as emergent. The last concept is about cooperation, which plays a fundamental role in the AMAS theory.

1.3.1 Objectives

Example systems are briefly presented here to give the reader a better idea of the kind of systems this approach is used on. Specificities about the agents and the MAS they constitute are given, as well as the more general or philosophical aim of this approach.

Examples of Targeted Adaptive Complex Systems

The adaptive multi-agent systems we want to design are dedicated to solve complex problems. These problems exclude simple brute force solvers and lead us to design complex systems in order to solve them. Some illustrative examples include:

- a time tabling with numerous and dynamic constraints in which when a new constraint is changed the new solutions must minimise the numbers of changes in the time tabling,
- manufacturing control which is a problem of production planning and control. Its concern is the application of scientific methodologies to problems faced by production management, where materials and/or other items, flowing within a manufacturing system, are combined and converted in an organised manner to add value in accordance with the production control policies of management [36].
- molecule folding which consists in finding the organisation between the atoms constituting the molecule that minimises the global energy of the molecule. It is a difficult problem because of the lack of knowledge about the inter-atom influences (see the TFGSO website[3]).

Many more of these kind of problems exists, which generally share one or more of the characteristics presented in the following sub-section.

Context and Scope of AMAS

The agents have to collectively solve the problem and thus all the agents participate in the solving without deliberatively lying or being malicious. The agents can be compared to sub-programs which contribute to the design of the global program in classical computer science. In this case, designer do not conceive a sub-program which would hide a result or provide a false result when they find the right one. This approach focuses on MAS in which autonomous agents have to solve a common task or reach a common objective. By consequence, the AMAS theory cannot be fully used to design all the adaptive complex systems and all kind of simulation of these systems in the same MAS. For example, a system used to simulate in a same system an economic system with layers and malicious agents would require other approaches.

[3] www.irit.fr/TFGSO

The application field is concerned by applications with the following characteristics:

- the application is complex in the sense of complex systems (see "Definition" chapter)
- the control and the knowledge can (and often has to) be distributed,
- there is a problem to solve. The problem can be expressed as a task or a function to realise, a structure to be observed...,
- the application objective can be very precise such as the optimisation of a function or more diffuse such as the satisfaction of the system end-users,
- the system has to adapt to an endogenous dynamic (with the add-ons or removing of parts of the system) or exogenous (with the interaction with its environment),
- the system is underspecified. In this case the adaptation is a mean to design it (see "Methodologies" chapter).

Objectives of the AMAS Theory

Specifying *a priori* an organisation for a system that will have to deal with unexpected events constrains (maybe inopportunely) the space of possibilities. It is commonly admitted that it is very difficult to predict how a real complex system will behave in a dynamic environment. Because there is a huge number of states in the space search, it is not possible to explore all of them in a reasonable time. A goal of the AMAS theory is to provide means to enable the system to find its right configuration in a given environment in order to be in adequacy with this environment (this will be explained in the next sub-section). In these terms, the AMAS theory can be used to produce systems with the same aims as those produced with optimisation methods such as simulated annealing [20], genetic algorithms [18, 14], swarm algorithms [4, 7].

Since von Bertalanffy [3], many authors [30, 16, 22, 24] have studied systems of different order that cannot be apprehended by studying their parts taken separately: *"We may state as characteristic of modern science that this scheme of isolable units acting in one-way causality has proven to be insufficient. Hence the appearance, in all fields of science, of notions like wholeness, holistic, organismic, gestalt, etc., which all signify that, in the last resort, we must think in terms of systems of elements in mutual interaction"* [3].

By consequence, scientists who are interested in these complex systems, must propose new models and new approaches to study and design such systems. The AMAS theory is one of these theories which aim to assist in designing adaptive complex (multi-agent) systems. Complying to the previous point of view, it focuses on the elements of the system that are the agents. The main concern is to design interacting autonomous agents with local knowledge which would collectively provide the global system with a coherent behaviour. The question is: what are the local rules of behaviour for these agents? Cooperation is the key as the reader could easily guess.

1.3.2 Adaptation

Adaptation is commonly defined by the capability a system has to change its internal structure in order to modify its behaviour to reach adequacy with its environment [37, 9] (see "Definition" chapter). Adequacy means that the behaviour of the system fits well inside its environment. For example, if we observe a crisis management system during a forest fire, the system is composed of the rescue team (such as fire men, doctors...) and autonomous artificial resources (such as robots). The environment of the system is the forest and the people present in the forest (wounded or not). We can say that the system is adequate from the environment point of view if it can stop the fire, rescue the wounded and save the other people.

A multi-agent system is a system composed of several interacting and autonomous agents. The relationships between the agents provide the organisation of the global system. These relationships can be expressed in terms of:

- tasks relations between the agents, for example an agent A provides to an agent B the result of a task or an information;
- beliefs relations about others agents; for example, an agent can have a point of view and a confidence on others agents;
- physical relations between agents, for example two atoms are linked in a molecule by a covalent link.

For example, the EPE (Emergent Programming Environment) system [10] shows an organisation between the following agents: the two "+" ("plus") and "*" ("multiply") operators and 3 numbers (120, 2 and 10) which can also be seen as simple agents. The aim of these agents is to form an organisation to exchanges values and calculate a final result. With the organisation in Fig. 1.2 the global behaviour of the system is to compute and provides the value: 120. The system can then be required to produce a different value.

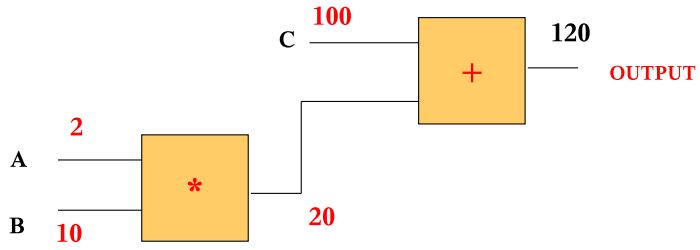

Fig. 1.2 Simple calculus agents in a first configuration representing a function with a result of 120.

An intuitive idea (applied in the AMAS theory) to realise the adaptation of a complex system composed of several interacting and autonomous agents is to change its organisational structure. Continuing with the example, you can see on Fig. 1.3 a new organisation of the EPE system. This new organisation provides a new final result

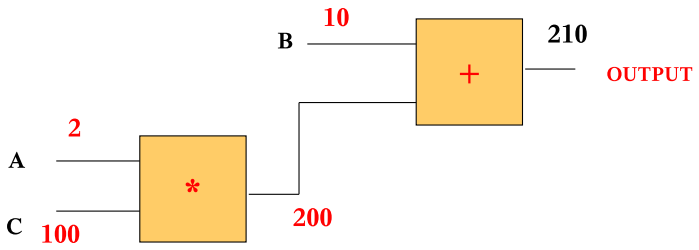

Fig. 1.3 Simple calculus agents after reorganising, representing a function with a result of 210.

which is now 210. The adaptation capability of this system lies on its capability to change and find its right organisation.

In the AMAS approach, we consider that each part P_i of a system S achieves a partial function f_{P_i} of the global function f_S (cf. Fig. 1.4). f_S is the result of the combination of the partial functions f_{P_i}, noted by the operator "$\circ$". The combination being determined by the current organisation of the parts, we can deduce $f_S = f_{P_1} \circ f_{P_2} \circ ... \circ f_{P_n}$. As generally $f_{P_1} \circ f_{P_2} \neq f_{P_2} \circ f_{P_1}$, by transforming the organisation, the combination of the partial functions is changed and therefore the global function f_S changes. This is a powerful way to adapt the system to the environment. A pertinent technique to build this kind of systems is to use adaptive MAS. As in Wooldridge's definition of *multi-agent systems* [35], we will be referring to systems constituted by several autonomous agents, plunged in a common environment and trying to solve a common task.

Since we want the system to find by itself its organisation, we characterise it as self-adaptive. Note that we have to distinguish a system which is adapted because the designer stops it, changes something in it and launches it again, from systems which adapt themselves to react to their environment. For example, in the first case, a car which does not start is not well adapted but after a mechanic has repaired it the car starts and is adapted to the need of its owner. In the second case we can take an example in natural systems with the Darwinian evolution where animals evolve to adapt to their environment. This book is dedicated to the second type of systems.

In the AMAS approach, the organisation changes are done autonomously by the agents of the system and it is a process of self-organisation (as it has been defined in the "Definition" chapter). This self-organisation is the origin of emergent global properties at the system level which cannot be predicted when given only the agents behaviours.

1.3.3 Emergence

As you can see in the "Definition" chapter, emergence is a widely studied concept and numerous definitions exist. Because you are computer scientists, we provide

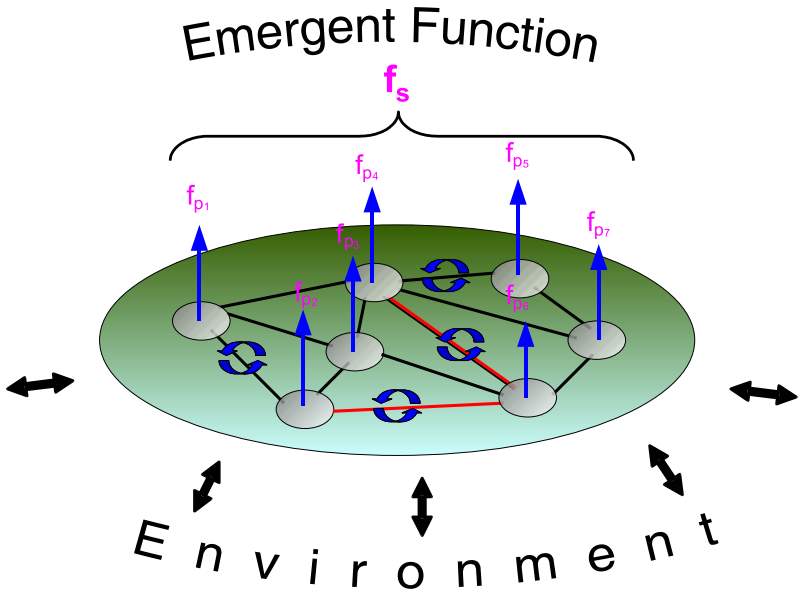

Fig. 1.4 Adaptation: changing the function of the system by changing the organisation.

a "technical" definition of emergence, relying on computer science concepts. It is based on three points:

1. **The subject.** The goal of a computational system is to realise an adequate function, judged by an external observer. It is this function, which may evolve during time, that has to emerge. The term function must be taken in a general meaning and not in a strict mathematical sense. A global function here can be a problem solving, a coherent behaviour, a structure... In software engineering it is simply what the system has to do.
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. Still, this coding has to contain the mechanisms allowing the adaptation of the system during its coupling with the environment, so as to tend any time towards the adequate function.
3. **The method.** To change the function in the AMAS theory, the system only has to change the organisation of its components. The mechanisms which allow the changes are specified by cooperative self-organisation rules providing autonomous guidance of the components behaviour without any knowledge on the collective function.

The condition is perhaps the more difficult part to understand. As a designer, we know what the system has to do and we want to "control" what the system will do. Designers want to control the emergence which seams completely antinomic with the meaning of emergence. Therefore, most of the time, the phenomenon observed is not a surprise for the designer. But we can qualify it as emergent, from an engi-

neering point of view, because if only the code of the agents is accessible and can be studied, the study cannot explain and predict the global function realised by interacting agents. In this sense, the global function is emergent. It is the reason why we need to code local criteria to guide the agents behaviour and the agents does not know the global function. The way to obtain this global function is not coded inside the agent code. In the AMAS theory, these local criteria are based on the aim to maintain cooperation.

1.3.4 Cooperation as the Engine for Self-Organisation

For a MAS, implementing this adaptation with an emergent global function implies that the designer only has to take care of the agent by giving it the means to decide autonomously to change its links with the other agents. As you have seen, we start from the principle that, to have a relevant behaviour, the elements that constitute a system have to be "at the right place, at the right time" in the organisation. To achieve this, each agent is programmed to be in a cooperative situation with the other agents of the system. Only in this case does an agent always receive relevant information for it to compute its function, and always transmit relevant information to others. The designer provides the agents with local knowledge to discern between Cooperative and Non Cooperative Situations (NCS[4]).

Cooperation can be summarised in the following attitude: an agent tries to help and not hinder the other agents. So an agent has to detect and eliminate NCS he encounters and it has to avoid to create new NCS. This behaviour constitutes the engine of self-organisation. Reader can compare this with how stigmergy (described in the "Stigmergy" chapter) is used in ant colonies (even if stigmergy can also be seen as a means to cooperate).

Depending on the real-time interactions the MAS has with its environment, the organisation between its agents emerges and constitutes an answer to the aforementioned difficulties in complex systems (cf. Sect. 1.1): indeed, there is no global control of the system. In itself, the emergent organisation is an observable organisation that has not been given first by the designer of the system. Each agent computes a partial function f_{P_i}, but the combination of all the partial functions produces the global emergent function f_S.

By principle, the emerging purpose of a system is not recognisable by the system itself, its only criterion must be of strictly local nature (relative to the activity of the parts which make it up). By respecting this, the *AMAS* theory aims at being a theory of emergence.

[4] The concept of NCS will be precisely defined in section 1.4.1, Definition 1.4

1.4 The *AMAS* Theory: Underlying Principles and Implementation

The two first parts of this section expound the main definitions and theorem the AMAS theory is based on. This theory can be applied at the system and also at the agent level. Because the global system behaviour is the result of interacting agents, the architecture and the general algorithm of an agent are described in the two last parts.

1.4.1 The Theorem of Functional Adequacy

In order to show the theoretical improvement coming from cooperation, the *AMAS* (Adaptive Multi-Agent System) [12] theory has been developed, which is based upon a specific theorem which is described bellow. This theorem describes the relation between cooperation in a system and the resulting functional adequacy[5] of the system. For example, let's consider a car as a system and the driver plus the real world as its environment. If the driver wants to move forward with the car and acts on the car to do this and if the car goes backward, the system behaviour is not adequate from the environment point of view.

Theorem 1.1. *For any functionally adequate system, there exists at least one cooperative internal medium system that fulfils an equivalent function in the same environment.*

Definition 1.3. A cooperative internal medium system is a system where no *Non Cooperative Situations* exist.

Note that the cooperative internal medium system can be either built so as to be functionality adequate or this system is given the capabilities to reach the adequacy on its own.

In a cooperative internal medium system, the components composing the system (which are in the internal medium) are always in cooperative situations. For example, if we consider a manufacturing control problem (briefly described in Sect. 1.3), it has to be constantly in a solved state inside. This means that the products are always made with all the constraints satisfied. From the environment point of view the system is functionally adequate because all constraints are satisfied. Inside the system, this means that the products, the work stations and the operators have no problems to work together, they cooperate well, they have no NCS. In other terms, we can say that all components are in the right location at the right time. The configuration of the system is the right one.

[5] "Functional" refers to the "function" the system is producing, in a broad meaning, i.e. what the system is doing, what an observer would qualify as the behaviour of a system. And "adequate" simply means that the system is doing the "right" thing, judged by an observer or the environment. Therefore, "functional adequacy" can be seen as "having the appropriate behaviour for the task".

We can apply this theorem to design complex adaptive systems composed of agents. To obtain an internal cooperative medium, we have to guarantee cooperative states between the agents inside the system and between the agents and the system environment (if the agents can perceive it of course). We call this kind of agents: cooperative agents.

To design cooperative agents, it is necessary to provide for each of them a behaviour to be continually in cooperative interactions. But in a dynamic environment and open systems, this status cannot be always guaranteed. The objective is by consequence to design systems that do the best they can when they encounter difficulties. These difficulties can be viewed as exceptions in traditional programming. From an agent point of view, we call them Non Cooperative Situations (NCS, see definition below) or cooperation failures.

The designer has to describe not only what an agent has to do in order to achieve its goal but also which locally detected situations must be avoided and if they are detected how to suppress them (in the same manner that exceptions are treated in classical programs). The agent design concerns on one part to provide the agent its nominal behaviour (the capabilities to play its role in the system) but also a cooperative behaviour to avoid and/or to remove the NCS. The NCS are defined in a very general and high level way with the above meta-rules at each step of the agent life cycle (perception, decision, action). These meta-rules have to be instantiated in each application by the designer.

Definition 1.4. An agent is in a Non Cooperative Situation (*NCS*) when:

$(\neg c_{perception})$ a perceived signal is not understood or is ambiguous; Here, *signal* is a general term to point out something received or perceived by the agent (a message, a video feed...).
$(\neg c_{decision})$ perceived information does not produce any new decision;
$(\neg c_{action})$ the consequences of its actions are not useful to others.

Let us give examples from everyday life to better understand these situations. For the first situation, a signal is not understood when a person speaks to you in Chinese and if you don't understand Chinese. For the second situation, in the human world, quite often a person overhears a conversation between other people and this conversation doesn't concern him, or he already knows the information. In the third situation, if in a robot world, robots have to clean a room and if a robot R_a prevents another robot R_b from moving, R_a does not act cooperatively.

We can identify seven NCS subtypes that further specify these situations:

- *incomprehension* ($\neg c_{per}$): the agent cannot extract the semantic contents of a received stimulus,
- *ambiguity* ($\neg c_{per}$): the agent extracts several interpretations from a same stimulus,
- *incompetence* ($\neg c_{dec}$): the agent cannot benefit from the current knowledge state during the decision,
- *unproductiveness* ($\neg c_{dec}$): the agent cannot propose an action to do during the decision,

- *concurrency* ($\neg c_{act}$): the agent perceives another agent which is acting to reach the same world state,
- *conflict* ($\neg c_{act}$): the agent believes the transformation it is going to operate on the world is incompatible with the activity of another agent,
- *uselessness* ($\neg c_{act}$): the agent believes its action cannot change the world state or it believes results for its action are not interesting for the other agents.

1.4.2 Consequence of the Functional Adequacy Theorem

This theorem means that we only have to use (and hence understand) a subset of particular systems (those with cooperative internal mediums) in order to obtain a functionally adequate system in a given environment. We concentrate on a particular class of such systems, those with the following properties [12]:

- The system is cooperative and functionally adequate with respect to its environment. Its parts do not *know* the global function the system has to achieve via adaptation.
- The system does not use an explicitly defined goal, rather it acts using its perceptions of the environment as a feedback in order to adapt the global function to be adequate. The mechanism of adaptation is for each agent to try and maintain cooperation using their skills, representations of themselves, other agents and environment.
- Each part only evaluates whether the changes taking place are cooperative from its point of view – it does not know if these changes are dependent on its own past actions.

This approach has been successfully applied in the engineering of self-organising agent-based systems in various application contexts with different characteristics, such as autonomous mechanisms synthesis [6], flood forecast [11], electronic commerce and profiling [13]. On each, the local cooperation criterion proved to be relevant to tackle the problems without having to resort to an explicit knowledge of the goal and how to reach it.

1.4.3 Architecture and Behaviour of an AMAS *Agent*

A cooperative agent in the *AMAS* theory has the four following characteristics. First, an agent is autonomous in its decision taking: an agent can say "no" or "go" (start some activity). Secondly, an agent is unaware of the global function of the system; this global function emerges (of the agent level towards the multi-agent level). Thirdly, an agent can on one hand try to avoid NCS and on the other hand detect NCS and acts to return in a cooperative state. And finally, a cooperative agent is not altruistic in the meaning that an altruistic agent always seeks to help the other

agents. It is benevolent i.e. it seeks to achieve its goal while being cooperative. More formally, the behaviour of an *AMAS* agent can be described with an algorithm based on the one found in Sect. 1.4.4.

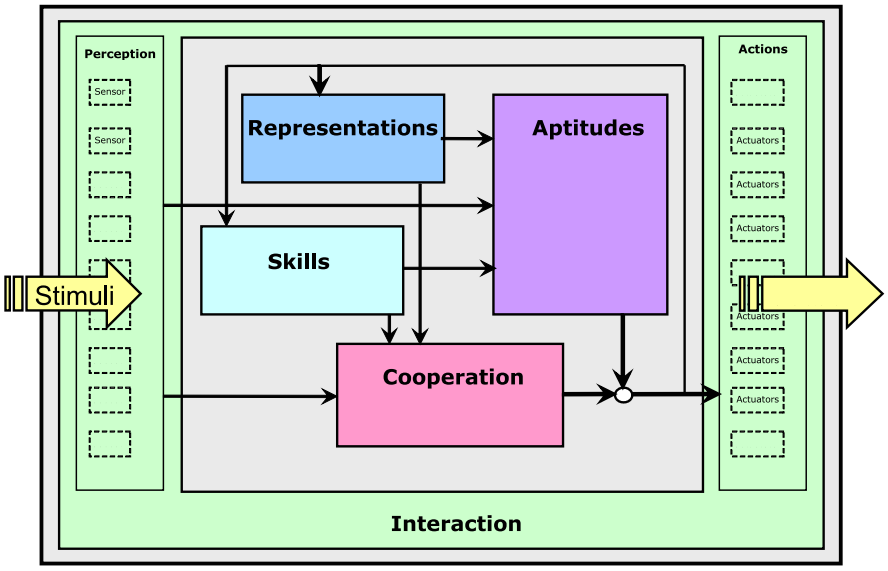

Fig. 1.5 The different modules of a cooperative agent

Cooperative agents are equipped with several modules representing a partition of their "physical", "cognitive" or "social" capabilities (cf. Fig. 1.5). Each module represents a specific resource for the agent during its "perceive-decide-act" life cycle. The first four modules are quite classical in an agent model [35]. The novelty comes from the Cooperation Module which contains local rules to solve NCS.

Interaction Modules. Agents' interactions are managed by two modules. The Perception Module represents the inputs the agent receives from its environment. Inputs may have different levels of complexity and types: integer, boolean for simple agents or even symbolic messages in a mail box for high level agents. The Action Module represents the output and the way the agent can act on its physical environment, its social environment or itself (considering learning actions for example). Similarly to the perceptions, actions may have different granularities: simple effectors activation for a robot or semantically complex message sending for social agents.

Skill Module. Even if cooperative agents mainly try to avoid NCS, they have several tasks to complete. The ways to achieve their goals are expressed in the Skill Module. Skills are knowledge about given knowledge fields and allow agents to realise their partial function – as a part of a MAS that produces a global function. No technical constraints are required to design and develop this module. For example, skills can be represented as a classical or fuzzy knowledge base of facts and rules

on particular domains. It also can be decomposed into a lower level MAS to enable learning, as in the ABROSE online brokerage application [13], where skills were decomposed into a semantic network.

Representation Module. As for the Skill Module, the Representation Module can be implemented as a classical or fuzzy knowledge base, but its scope is the environment (physical or social) and itself. Beliefs an agent possesses on another agent, as well as all information the agent possesses on its environment are considered as representations. Like skills, representation can be decomposed into a MAS when learning capabilities on representation are needed.

Aptitude Module. Aptitudes represent the capabilities to reason on perceptions, skills and representation – for example, to interpret messages. These aptitudes can be implemented as inference engines if skills and representations are coded as knowledge bases. Considering a given state of skills, representations and perceptions, the Aptitude Module chooses an action to do. Cases when there is zero or several proposed actions must be taken into account too (cf. Cooperation Module).

Cooperation Module. The cooperative attitudes of agents are implemented in the Cooperation Module. As the Aptitude Module, this module must provide an action for a given state of skills, representations and perceptions, *if the agent is in a NCS*. Therefore, cooperative agents must possess rules to detect NCS. Several types of NCS have been identified. For each NCS detection rule, the Cooperation Module associates one or several actions to process to avoid or to solve the current NCS.

Internal Functioning of an *AMAS* Agent

Considering the described modules, the nominal behaviour of a cooperative agent is defined as follows. During the perception phase of the agents' life cycle, the Perception Modules updates the values of the sensors. These data directly imply changes in the Skill and Representation Modules. Once the knowledge updated, the decision phase must result on an action choice. During this phase, the Aptitude Module computes from knowledge and proposes action(s) or not. In the same manner, the Cooperation Module detects if the agent is in a NCS or not. In the former case, the Cooperation Module proposes an action that subsumes the proposed action by the Aptitude Module. In the latter case, the only action[6] proposed by the Aptitude Module is chosen. Once an action chosen, during the action phase, the agent acts by activating its effectors or changing its knowledge.

1.4.4 The Cooperative Algorithm

The algorithm in Fig. 1.6 may be viewed as a formal representation of the cooperative attitude of the agents described previously: according to the *AMAS* theory,

[6] There is only one action possible, otherwise an NCS is detected.

```
/* Definitions and notations */
    P: Set of possible percept sets (perceptions, representations, or skills) at a given time for the
    agent
    A: Set of possible action sets for the agent
    (p,a): The couples (p:P,a:A) are the rules of possible behaviours for the agent, i.e. the actions
    to be executed for a given percept set
    NCSR: Non Cooperative Situation Rules, set of behaviours rules (p:P,a:A) corresponding to
    the detection of a non cooperative situation and the associated corrective actions
    SR: Skill Rules, set of behaviour rules (p:P,a:A) corresponding to the possible actions de-
    pending only on percept sets (without having to refer to the beliefs of the agent).
    BR: Belief Rules, set of behaviour rules (p:P,a:A) corresponding to the possible cooperative
    actions for given percept sets and by referring to the beliefs (to evaluate the cooperation).
    (p1,a1) ≻ (p2,a2): "≻" expresses a priority relationship of the behaviour (p1,a1) over
    (p2,a2).

Procedure action(p:P) {     /* p is the current percept set of the agent */
   if (inCooperativeSituation(p))
      executeUtilityAction(p)
   else {    /* The agent is in a non cooperative situation */
      executeNcsCounterAction(p);
      executeUtilityAction(p);
   }
}

Function inCooperativeSituation(p:P) Return Boolean {
   for each (p',a') ∈ NCSR
      if(p' ⊆ p)
        return false
   return true;
}

Function executeUtilityAction(p:P) {
   for each (p',a') ∈ SR∪BR | p' ⊆ p
      if (∄ (p",a") ∈ SR ∪ BR | p" ⊆ p and (p",a") ≻ (p',a'))
         do a'
}

Function executeNcsCounterAction(p:P) {
   for each (p',a') ∈ NCSR | p' ⊆ p
      if (∄(p", a") ∈ NCSR | p" ⊆ p and (p",a") ≻ (p',a'))
         do a'
}
```

Fig. 1.6 Procedure and functions for a cooperative agent

agents have to be able to detect when they are in a NCS and in which way they can act to come back in a cooperative situation.

In the algorithm, there are two main states to which the decision process may be confronted: either the agent is in a cooperative situation (depending on its perceptions) (the function called *inCooperativeSituation* returns the value "true"), or it is in a NCS (the function called *inCooperativeSituation* returns the value "false"). In the first state, the agent simply chooses the action with the highest priority among those which can be executed (function called *executeUtilityAction*). Theses actions are said to be utility actions in the algorithm, meaning that they are useful for the goal of the agent, for another agent or for the system. If the agent can prevent NCS, these actions also avoid to generate new NCS. In the other state, the agent is in an NCS and, in addition to some possible utility action it will choose the corrective action with the highest priority depending on its perceptions (function called *executeNcsCounteredAction*).

Following such an algorithm, agents always try to stay in a cooperative situation and so the whole system converges to a cooperative state within and with its environment. This leads – according to the theorem of functional adequacy – to an adequate system.

Thus, this algorithm describes the typical decision process of a generic *AMAS* agent. But the NCS and the actions which could be applied to solve them are not generic: designers have to write their own- and so specific- NCS set and related actions for each kind of agent they wish the system to contain. This work must be performed during the design of the agents: the designer must exhaustively find all the NCS which could occur for each kind of agent and, for each one, find the relevant actions which could solve the lack of cooperation. Methods (like ADELFE) can help for this (see corresponding chapter in this book).

1.5 Applications

One does only truly understand a theory after taking two additional steps: study a comprehensive example of the application of the theory and applying it him or herself. For the later, we strongly encourage the reader to tackle the exercises of this chapter as they are intended to provide material to acquire practical know-hows. The first step will be taken here as this section present no less than two different application illustrating and detailing the use of cooperation conforming to the AMAS theory. The first is a dynamic and open service providing MAS where all the providers and customers (the agents) need to be put in relation with one another. This relationship needs to be constantly updated to ensure the most relevant social network (by being cooperative one with another). The second is a multi-robot resource transportation problem where the robots (the agents) have to share the limited routes to efficiently transport the resources (by choosing cooperatively how to move). Each description focuses on how cooperation can be applied, what Non

Cooperative Situation are for the agents and how it enables them to self-organise towards the adequate emergent function.

1.5.1 A Service Providing MAS

1.5.1.1 Instantiation of the AMAS Approach to a Case Study

The first chosen case study to illustrate the AMAS approach consists in designing a system which enables end-users and service providers to get in touch when they share common points of interest. The application is made for the electronic commerce field, an open, dynamic and distributed context [13].

The main requirement of such an electronic information system is to enable (i) end-users to find relevant information for a given request and (ii) service providers to have their information proposed to relevant end-users. In concrete terms, the system has to provide:

- Personalised assistance and notification for the end-users,
- Propagation of requests between the actors of the system,
- Propagation of new information only to potentially interested end-users,
- Acquisition of information about end-users' real interests, in a general manner, and about providers' information offers.

In such a system, every end-user and service provider has an individual goal: to answer the request he/she has to solve. Each end-user and service provider does not know the global function realised by the system. The system is open and strongly dynamic because a great number of appearances or disappearances of end-users and/or service providers may occur. Moreover, an *a priori* known algorithmic solution does not exist. In this context, classical approaches to tackle such a problem cannot be applied and the use of AMAS is then clearly relevant.

1.5.1.2 Environment Definition and Characterisation

The environment of the system consists of real end-users and service providers who have subscribed to the system. They exert pressure on the system (by submitting requests in order to find relevant service providers or to seek potential customers) and the system has to adapt itself to these constraints. The reorganisation of interaction links between agents representing end-users and service providers is a way for the system to adapt to its environment which can be described as inaccessible, continuous, non deterministic and highly dynamic.

We can say that such a system is functionally adequate when a satisfied end-user wants to use services of the system again, and when each service is fully used, namely in the most profitable way for the supplier.

1.5.1.3 Agent Design

An end-user seeks a relevant service provider according to his/her centres of interest, while a provider tries to find potentially interested end-users according to his/her proposed services. These two actions are totally symmetric and we will only focus on the search for a service.

In that system, entities are autonomous, have a local goal to pursue (to find a relevant service provider or to make targeted advertisement), have a partial view of their environment (other active entities) or may interact with others to target the search more effectively. They are then potentially cooperative. Furthermore, since the system is open, new entities may appear or disappear and they may not be able to communicate as they should (e.g., an entity does not understand requests from a new one). Therefore, such an entity is prone to cooperation failures and can be viewed as a cooperative agent. Each end-user (or service provider) is then represented within the system by an agent called *Representative Agent* (see Fig. 1.7).

A representative agent (*RA*) aims at finding relevant *RAs* according to the request that its associated end-user or service provider submitted for solution. In accordance with the agent model given in Sect. 1.4.3, an *RA* consists of the following components:

- The skills of an *RA* are those of the entity it represents.
- Representations that an agent possesses about itself or about other *RAs* may evolve at runtime and they have then to be adjusted. Because of this dynamics, we can use an AMAS to implement them (more information about this point can be found in [13]). When an *RA* receives a request, it has to query its representations on itself to know if it is relevant to solve this request. If it is not, it has then to query its representations on other known agents to identify if it knows an agent able to solve the received request.
- The aptitudes of an *RA* enable it to modify its representations and to interpret a received request. For example, when an end-user makes a request, his/her *RA* has to update its representations to learn the new centres of interest of its end-user.
- Messages exchanged between *RAs* concern the requests to be solved. Physical exchanges of these requests can be made using the mailbox concept, a buffer enabling asynchronous communication.

1.5.1.4 Non Cooperative Situations Determination at the RA Level

NCS have to be instantiated to the current problem and actions to be done when an agent is faced with an NCS have to be defined. An *RA* is situated at the right place in the organisation of the system if the 3 meta-rules given in Sect. 1.4 are checked. An *RA* may encounter four non cooperative situations during its perception/decision/action cycle:

1. **Total incomprehension**

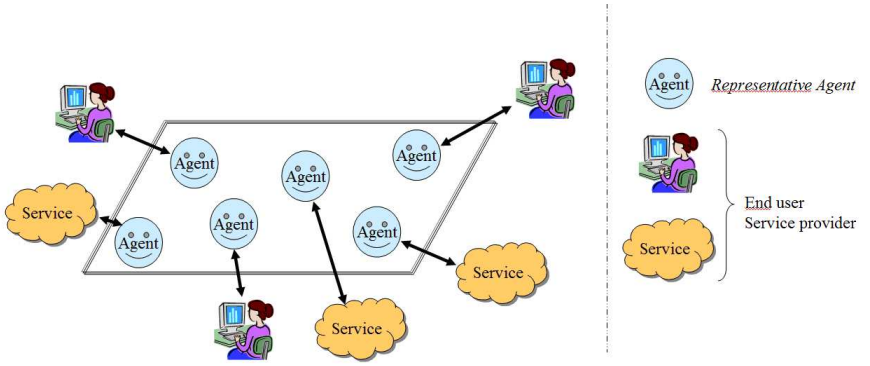

Fig. 1.7 A service providing multi-agent system architecture

Description: An agent faces total incomprehension when it cannot extract any informative content from the received message: this may be due to an error in transmission or if the transmitter gets a wrong belief about it. This NCS is detected during the interpretation phase when the agent compares the received request with its own representation (words matching) and cannot extract any informative content from the message; it has not the necessary competence.
Actions: Because the agent is cooperative, the misunderstood message is not ignored; the agent will transmit the message to an agent that seems to be relevant according to its representations on others.

2. **Partial incompetence**
   Description: An agent is faced with partial incompetence when can extract an informative content from only one part of the received message. This NCS is detected during the interpretation phase when the agent compares the received request with its own representation (words matching) and can extract an informative content from only a part of the message.
   Actions: The receiving agent sends back the partial answer associated with the understood part of the message. It sends the other part of the request to a more relevant agent.
3. **Ambiguity**
   Description: An ambiguity occurs when the an agent can extract several informative content from the received message it is faced to an ambiguity. This NCS is detected during the interpretation phase when the agent compares the received request with its own representation (words matching) and can extract several informative contents from the message.
   Actions: An agent is supposed to intentionally and spontaneously send understandable data to others. Therefore, the receiver of an ambiguous message sends back all its interpretations of the received request. The initial sender is then able to choose the most pertinent one and update its representation about the receiver's skills.
4. **Concurrence**

Description: A situation of concurrence occurs when two agents have similar skills for a given task. This NCS is detected during the interpretation phase, when the agent compares the received request with its own representation (words matching). If it can extract an informative content from only a part of the request, the agent compares this request with the representation it has about other agents to find rival agents. An agent *A* competes with an agent *B*, from *B*'s point of view, if *A* can extract informative content from the same part of the request as *B*.
Actions: Redundancy is beneficial when an agent has not been able to reach its aim or to accept a task it has been asked to undertake. In these cases, it refers the problem to its rival(s).

NCS being now instantiated to the current problem, the cooperative behaviour of an *RA* is the following:

- when it detects an NCS, it acts in the world to come back to a cooperative state.
- when it does not detect an NCS, it follows its own goal.

The simple continued cooperative activity of the agents ensures that the global organisation (which agents are in relation with each other) is always the most relevant to satisfy every involved entity. By studying the situations in which cooperation can occur and identifying corresponding behaviours, the application has been enhanced. The analysis of cooperation provides the needed self-organising mechanisms and can be seen as a guide for obtaining the desired emergent functionality.

1.5.2 Multi-Robot Resource Transportation

1.5.2.1 Resource Transportation Problem

The resource transportation problem is a classical task in Collective Robotics [31], and it was proposed as a relevant benchmark for robotic systems by [4]. Robots must transport resources (boxes) as fast a possible from a zone A to a zone B, separated by a constrained environment. In Picard's work [26, 29] presented here, these zones are linked by two corridors too narrow for robots to cross one another side by side (cf. Fig. 1.8). This environment leads to a spatial interference problem, e.g. robots must share common resources: the corridors. Once engaged in a corridor, what must a robot do when facing another robot moving in the opposite sense? Spatial interference has been tackled by [32] in the case of robots circulating in corridors and having to cross narrow passages (doors). Their solution is to solve conflicts by aggressive competition (with explicit hierarchy), similarly to eco-resolution by [8]. [21] propose to solve such problems thanks to attraction-repulsion mechanisms based on altruistic behaviours triggering – a reverse vision of the eco-resolution. In the application described here, Picard expounds a viewpoint halfway between the two firsts, in which robots are neither altruistic nor individualist and cannot directly communicate any information or intention. Moreover, no planifier system will an-

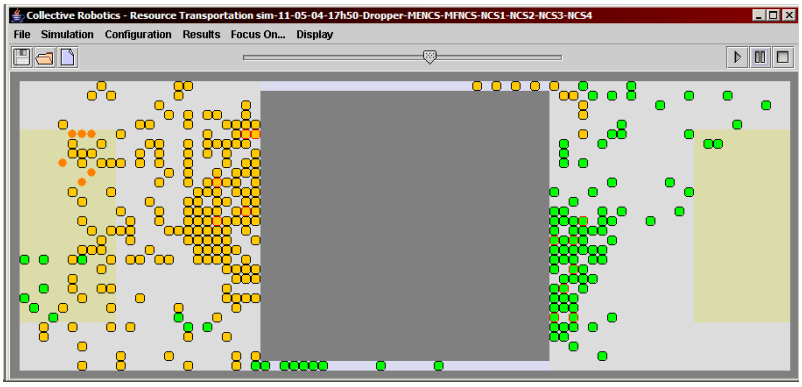

Fig. 1.8 The environment of the resource transportation problem is composed of: a claim room (at left), a laying room (at right) and two narrow corridors (at top and bottom). Robots pick boxes against the left wall of the claim room (claim zone) and drop them against the right wall of the laying room (laying zone).

ticipate trajectories because the use of planification in multi-robot domain remains inefficient, considering the high dynamics of a robot's environment.

1.5.2.2 Cooperative Model Instantiation

This section shows the instantiation – i.e. fulfilling each module – of the cooperative agent model in order to design robots able to realise the transportation task. This work appears in the ADELFE process (see the "Methodologies" chapter in this book) in the *Design Work Definition*, and more precisely in the *Design Agents Activity* [28]. ADELFE process is an extension to the *Rational Unified Process* (RUP) and consists in four work definitions, which are specifically adapted to agent-oriented software engineering: preliminary requirements, final requirements, analysis and design. Requirements defines the environmental context of the system. Analysis identifies the agents within other object classes.

Modules Fulfilling

The *Perceptions Module* represents inputs for agents. Concerning robots, they can know positions of the two zones (claim and laying). Indeed, this example only focuses on adaptation to a circulation problem instead of a foraging one. For example, we consider that the task of robots is to transfer boxes between rooms and that robots do not get involved in identifying box locations. Here is a possible list of perceptions for transporter robots: position of the claim zone, position of the laying zone, a perception cone in which objects are differentiable (robot, box or wall), proximity sensors (forward, backward, left and right), a compass and the absolute spatial po-

sition. The environment is modelled as a grid whose cells represent atomic parts on which a robot, a box or a wall can be situated. The Perceptions Module also defines limit values of perceptions (e.g. 5 cells).

The *Actions Module* represents outputs of agents on their environment. Possible actions for transporter robots are: *rest*, *pick*, *drop*, *forward*, *backward*, *left* and *right*. Robots cannot drop boxes anywhere in the environment but only in the laying zone. They cannot communicate directly or drop land marks on the environment. In the case of social agents that are able to communicate, communication acts are specified in this module.

The *Skills Module* contains knowledge about the task the agent must perform. Skills enable robots to achieve their transportation goals. Therefore, a robot is able to calculate which objective it must achieve in terms of its current state: if it carries a box then it must go to the laying zone, otherwise it must reach the claim zone. Depending on its current goal, the Skills Module provides an appropriate action to process to achieve it. Robot's goals are: *reach claim zone* and *reach laying zone*. Moreover, robots have intrinsic physical characteristics such as their speed, the number of transportable boxes or the preference to move forward rather than backward – as ants have. Such preferences are called *reflex values*.

The *Representations Module* contains knowledge about the environment (physical or social). Representation a robot has on its environment is very limited. From its perceptions, it cannot identify a robot from another, but can know if it is carrying a box or not. It also can memorise its past absolute position, direction, goal and action.

The *Aptitudes Module* enables an agent to choose an action in terms of its perceptions, skills and representations. Concerning transporter robots, a design choice must be taken at this stage. In terms of the current goal, the Skills Module provides preferences on each action the robot may do. The Aptitudes Module chooses among these actions what will be the next action to reach the goal. Many decision functions can be considered; e.g. an arbitrary policy (the action having the highest preference is chosen) or a Monte Carlo method-based policy that is chosen for our example. Therefore, the Aptitudes Modules can be summed up in a Monte Carlo decision function on the preference vector (the list of action preferences for an agent) provided by the Skills Module. In the same manner, the *Cooperation Module* provides preference vectors in order to solve NCS described in Sect. 1.5.2.3.

Action Choosing

At each time t, a robot has to choose between different actions that are proposed by the two decision modules (skills and cooperation). At time t, each action act_j of the robot r_i is evaluated. For each action, this value is calculated in terms of perceptions, representations and reflexes in the case of a nominal behaviour:

$$V_{r_i}^{nomi}(t, act_j) = wp_{r_i}(t, act_j) + wm_{r_i}(t, act_j) + wr_{r_i}(act_j)$$

with:

- $V_{r_i}^{nomi}(t, act_j)$ represents the value for the action act_j at time t for the robot r_i,
- $wp_{r_i}(t, act_j)$ represents the calculated value in terms of perceptions,
- $wm_{r_i}(t, act_j)$ represents the calculated value in terms of memory,
- $wr_{r_i}(t, act_j)$ represents the calculated value in terms of reflexes.

As for aptitudes, an action preference vector is generated by the Cooperation Module: $V_{r_i}^{coop}(act_j, t)$. Once these values calculated by the two modules for each action of a robot, the vector on which the Monte Carlo drawing will process is a combination of the two vectors in which the cooperation vector subsumes the nominal vector:

$$V_{r_i}(t) = V_{r_i}^{nomi}(t) \prec V_{r_i}^{coop}(t)$$

1.5.2.3 Cooperative Behaviours Study

In the previous section, the different modules of a robot and its components have been detailed, except the Cooperation Module. This section aims at discussing cooperation rules to establish in order to enable the multi-robot system to be in functional adequacy with its environment.

Cooperative Unblocking

Beyond the limited number of only two robots acting to transport boxes in a same environment, the nominal behaviour cannot be sufficient. Indeed, a robot owns skills to achieve its tasks, but not to work with other robots. In this very constrained environment, spatial interference zones appear. If two robots, a first one carrying a box and moving to the laying zone and a second one moving to the claim zone to pick a box, meet in a corridor, the circulation is blocked – because they cannot drop boxes outside the laying zone. Then, it is necessary to provide cooperative behaviours to robots. Two main NCS (non cooperative situations) can be reactively solved:

A robot is blocked. A robot r_1 cannot move forward because it is in front of a wall or another robot r_2 moving in the opposite direction[7]. In this case, if it is possible, r_1 must move to its sides (left or right). This corresponds to increasing values of the cooperative action vector related to side movements: $V_{r_1}^{coop}(t, right)$ and $V_{r_1}^{coop}(t, left)$. If r_1 cannot laterally move, two other solutions are opened. If r_2 has an antagonist goal, the robot which is the most distant from its goal will move backward (increasing $V_{r_i}^{coop}(t, backward)$) to free the way for the robot which is the closest to its goal (increasing $V_{r_i}^{coop}(t, forward)$ even if it may wait). If r_2 has the same goal than r_1, except if r_1 is followed by an antagonist robot

[7] If r_2 moves in another direction than the opposite direction of r_1, it is not considered as blocking because it will not block the traffic anymore.

Condition	***Action***
$ret \wedge freeR$	$\nearrow V_{r_i}^{coop}(t, right)$
$ret \wedge freeL$	$\nearrow V_{r_i}^{coop}(t, left)$
$ret \wedge \neg(freeL \vee freeR) \wedge ant \wedge toGoal \wedge cGoal$	$\nearrow V_{r_i}^{coop}(t, backward)$
$ret \wedge \neg(freeL \vee freeR) \wedge ant \wedge toGoal \wedge \neg cGoal$	$\nearrow V_{r_i}^{coop}(t, forward)$
$ret \wedge \neg(freeL \vee freeR) \wedge ant \wedge \neg toGoal$	$\nearrow V_{r_i}^{coop}(t, backward)$
$ret \wedge \neg(freeL \vee freeR) \wedge \neg ant$	$\nearrow V_{r_i}^{coop}(t, forward)$

With:
- *ret*: r_i is returning;
- *freeR*: right cell is free;
- *freeL*: left cell is free;
- *ant*: in front of an antinomic robot;
- *toGoal*: r_i is moving to goal;
- *cGoal*: r_i is closer to its goal than its opposite one;
- $\nearrow$: increasing.

Table 1.2 Example of specification of the "*a robot is returning*" uselessness NCS.

or if r_1 moves away from its goal (visibly it moves to a risky[8] region), r_1 moves backward; else r_1 moves forward and r_2 moves backward.
A robot is returning. A robot r_1 is returning[9] as a consequence of a traffic blockage. If it is possible, r_1 moves to its sides (an is no more returning). Else, r_1 moves forward until it cannot continue or if encounters another robot r_2 which is returning and is closer to its goal than r_1. Table 1.2 sums up the behaviour in this situation. If there is a line of robots, the first returning robot is seen by the second one that will return too. Therefore, the third one will return too and so on until there are no more obstacles.

These rules correspond to resource *conflict* (corridors) or *uselessness* when a robot must move backward and away from its goal. In the case of robots, situations will not be specified as incomprehension because robots are unable to communicate directly. These rules, which are simple to express, ensure that robots cannot block each other in corridors. But, this cooperation attitude only solves problem temporarily, creating returning movement and then implies time loss to transport boxes.

Cooperative Anticipation

By taking into account the previous remark, it seems possible to specify cooperation rules to anticipate blockage situations in order to make the collective more efficient. We call this *optimisation* cooperation rules. Previous rules enable robots to extract from blockage. A robot is in such a situation because it was crossing a zone frequented by antinomic robots [10]. So as to prevent this situation, robots must be able to avoid such risky zones: zones from which antinomic robots come. In accordance, an anticipation rule can be specified:

[8] It is risky in the sense it may occur a lot of non cooperative situations such as conflicts.

[9] A robot is considered as returning until it has no choice of side movements.

[10] Robots with an antinomic behaviour to the considered robot, for instance going in the opposite direction in a corridor

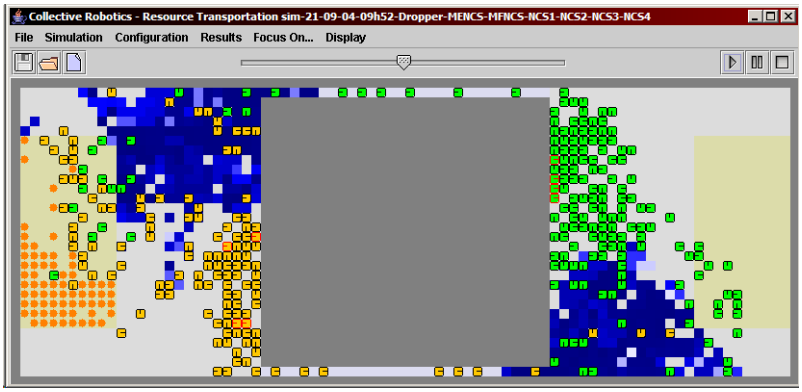

Fig. 1.9 The robots self-organise and a corridor dedication emerges. We can also see the positioning of all the virtual markers (dark squares) for all the robots and the two goals.

A robot sees an antinomic robot. If a robot r_1 perceives a robot r_2 having an antinomic goal, if r_1 can move to its sides it does it else it moves forward.

Nevertheless, this reactive anticipation presents a major problem: once a robot has avoided the risky zone, no mechanism ensures that it will not go in it again, led by its goal. In order to tackle this difficulty, robots can be equipped with a memory of the risky zones (in the Representations Module). Each time t a robot r_i experiments an anticipation situation facing a robot r_j, it adds to its memory a tuple (or virtual marker) $\langle posX(r_j,t), posY(r_j,t), goal(r_i,t), w\rangle$ in which $posX(r_i,t)$ and $posY(r_i,t)$ represent the coordinates of r_j at the moment t. $goal(r_i,t)$ represents the goal r_i was achieving at time t. w represents a repulsion value. The higher the value is, the more the robot will try to avoid the zone described by the marker when it is achieving another goal than $goal(r_i,t)$. Therefore, the robot inspects all its personal markers[11] whose distance is inferior to the perception limit (to fulfil the locality principle). A marker with a weight w and situated in the direction dir at a distance d induces that $V_{r_i}^{coop}(t, dir_{opp})$ will be increased of w (dir_{opp} is the opposite direction to dir).

As the memory is limited, tuples that are added must disappear during simulation run-time. For example, the weight w can decrease of a given value δ_w (called *forgetting factor*) at each step. Once $w = 0$, the tuple is removed from the memory. This method corresponds to the use of *virtual* and *personal* pheromones. Finally, as ants do, robots can reinforce their markers: a robot moving to a position corresponding to one of its marker with another goal, re-initialises the marker. In fact, if the robot is at this position, it might be a risky zone when it tries to achieve another goal.

1.5.2.4 Cooperation and the Emergence of Corridor Dedication

Fig. 1.9 shows the corridor-usage after a while: their cooperative behaviour guides the robots to use one corridor when trying to reach the boxes and the other to bring them back. The anticipative behaviour is mainly responsible for this observance of the emergence of a sense of traffic. Robots collectively dedicate corridors to particular goals. In fact, markers of all the agents are positioned only at one corridor entry for one direction as shown in Fig. 1.9. This view is only for monitoring purpose: robots do not perceive all the markers, only their own. We can assign the emergent property to this phenomenon because robots do not handle any notion of corridor – unlike some previous works [27]. Thus, just with local data, robots established a coherent traffic behaviour that leads to an optimisation of the number of transported boxes.

In this application, readers can see the relevance of cooperation as a local criterion for agents to self-organise to be more adapted to a task. Considering the ignorance of the global task and the environment, the self-organising collective reaches an emergent coherent behaviour, which is then more robust to environmental risks. This example tackles a simple problem in a simple static environment in which the collective achieves its global task. Other simulations in difficult and dynamic environments confirm the relevance of cooperative self-organising collectives.

1.6 Conclusion

To understand how cooperation can be used to build complex artificial systems, a theory of self-organising MAS, called *AMAS* has been presented in this chapter. This approach considers groups of cooperative agents which modify their interaction when non cooperative situations occur, to reach a functional adequacy with the environment. This approach has been illustrated by two different example applications: a service providing MAS and a multi-robot resource transportation problem. In both, the global or macro-level functionality of the system emerges from the cooperative interactions between micro-level entities, the agents.

It is mainly because uniqueness of a generic mechanism of learning or self-organising is rarely admitted in the scientific community, that research in this field explores many different directions. This opinion is clearly stated by Minsky [25]: "*I doubt that in any one simple mechanism, e.g., hill-climbing, will we find the means to build an efficient and general problem-solving machine. Probably, an intelligent machine will require a variety of different mechanisms. These will be arranged in hierarchies, and even in more complex, perhaps recursive, structures. And perhaps what amounts to straightforward hill-climbing on one level may sometimes appear (on a lower level) as the sudden jumps of "insight"*". But some others, like Inhelder [23], consider that a general theory of leaning might be feasible: "*in the contrary*

[11] Robots cannot share their memory as they cannot communicate.

to Chomsky's assertion, who states that no general theory of cognitive learning is possible, we strongly believe that knowledge learning is to a very general process, be it logico-mathematical knowledge, physics, or natural language learning".

From the point of view exposed here, a general theory can not be tributary of constricting presuppositions like the knowledge of the global function to achieve, of specific attributes of the environment, or of an explicit environment-system feedback. This explains that the notions of emergence and self-organisation are deeply embedded in this work. Cooperative self-organisation does not need any presupposition about the finality of the system. Component agents only follow their own individual objective (to be the most cooperative possible) rather than try to adapt by individual learning to external perturbations. Even if such a system does not possess any programmed finality at the agent level (except being cooperative), self-organisation leads to the required collective result by emergence.

The *AMAS* theory is a guide to design adaptive groups of agents in a simplified manner since designing parts of the system is of a constructive nature. Instead of starting from the global collective function and decomposing it in more elementary functions, we start by designing agents (their elementary functions) as well as the local criteria and behaviours that will guide their collective reorganisation. This is the detection and treatment of non cooperative situations. This theory strongly relies on emergence: the agents learn in a collective and non-predefined way. They modify their organisation in relation to disturbing interactions with the environment and thus, their global function. The two illustrating examples show the usability and relevance of this approach.

Quite a few other applications showed the adequacy of this emergent approach for managing adaptation and complexity in artificial systems, but there is still a lot of theoretical work to be done to explore its properties. For instance, we never seemed to observe local attractors. If this method of search space exploring seems unconcerned by a complex search space, it may be because the agents just ignore this search space and explore another one, the one constituted of the cooperative organisations of the system. But this will have to be proved. Another main objective is the diffusion of an *AMAS* design method among developers in the academic world as well as the industrial one. For this, readers can refer to the "Methodologies" chapter where the *Adelfe* method is presented: a toolkit to develop software with emergent functionality [28, 2].

Classic learning and adaptation techniques, which mostly have need of the knowledge of a cost function associated with the global function, can not pretend to produce emergent phenomena. This is why they fit the scope of the limitations enunciated by the "no free lunch theorem" of Wolpert and MacReady [34]. This theorem stipulates that all search space exploration algorithms (deterministic as well as stochastic), which make use of a search function to optimise a cost, globally have equivalent performances. Indeed, each one of them is only efficient given the bias introduced by the knowledge of the cost function. On a sufficiently vast corpus, they have he same performance: very efficient algorithms for a specific class of problems are the worst in another, and those with less efficiency are nevertheless average for all classes.

On the contrary, any effective theory of emergence does not fit in the scope of this theorem since, by principle, the function to reach is unknown and so, there can not exist an associated cost function. The *AMAS* theory could well be efficient for any class of problems. The glimpse of this possibility is one of the fascinating reason to explore emergence in artificial systems and thus the ability to manage complex adaptive systems.

1.7 Problems and Exercices

1.1. Carrying heavy boxes
There are two sort of boxes (light and heavy) lying around in a depot and they need to be carried away. A human can only lift a light box and a heavy box is two times the weight of a light one.

(a) How can the problem be solved by cooperation (ok, this one is really easy). Do the humans need to be able to communicate to cooperate ?

(b) We now have robots to carry the boxes but with the same limitations as the humans. On top of that, because of expenses, only basic perception means are installed on the robots (they only perceive the boxes, not the other robots) and no communication device is available. Is it still possible to solve the problem ? What simple tweak to the behaviour of the robots would ensure that even the heavy boxes will be carried away given enough time ?

1.2. Foraging Ants and Cooperation

When ants forage for food, they leave a chemical substance on the ground when returning to their nest carrying food. This substance is called pheromones and is a mark which other ants can detect. Pheromones can accumulate on a given spot or path and evaporate over the course of time. Readers can find plenty of on-line resources explaining how pheromones work in different species and comprehensive descriptions of ant behaviours. Readers can also take a look at the "Methodologies" chapter which uses an artificial ant foraging application as a case study. But a basic understanding is quite enough to tackle this exercise.

(a) Explain how this use of pheromones can be qualified as "cooperative".

(b) Could the behaviour of natural ants be enhanced to be even more cooperative? Imagine designing an artificial ant for a simulation or building an ant-like robot using pheromone-like marks and basic perceptions. Describe a few enhancements to their basic behaviours (using pheromones or simple perceptions) which would make them more cooperative and thus produce better results. At least six enhancements

can be found.

1.3. Cooperation in an Open System

Benevolent or selfless agents spontaneously cooperate whatever the cost to themselves. This is not the case for most agents in open systems.

a) What can two agents do to ensure that the cost of cooperation is evenly distributed.

b) What is one of the most efficient ways to have plenty of cooperation going on in an open system and still ensure that rogue or malevolent agents do not profit of the cooperative attitude of others.

1.4. Classifying Cooperation Means

List as many means as you can which enable cooperation.

1.5. Colour Cubes Game

Consider the following game: in a house with several rooms connected by doors there are cubes of different colours dispersed among the rooms and several robots. We want all the cubes of the same colour in the same room. A robot can carry up to 4 cubes and has three available actions: pick up cube, drop a cube, go to another room. There are no communication means.

Your client wants to build a simulation which would solve the problem. For this he asks you to design a multi-agent system in which each agent controls a virtual robot which can act once per simulation step.

Describe the algorithm of the agents so that a solution is efficiently reached and does not depend on the number of robots, rooms, cubes or colours.

Key Points

- Cooperation is inherent to social interactions, structures and organisations.
- Cooperation is present in numerous natural systems, ranging from social insects to human societies.
- Cooperation can take place in different forms and at different degrees.
- Cooperation is a powerful mechanism to implement self-organisation in artificial systems.

Glossary

Cooperation The act or attitude of interacting in whatever the means with another for mutual benefit. It usually involves sharing of information or competences.

Adaptation Process leading to a modification of a system for it to better respond to its environment.

Adequacy between a system and its environment State reached when a system flawlessly interact with its environment. Every action/reaction is relevant.

Self-organisation Process where a system changes its internal organization to adapt to changes in its goals and the environment without explicit external control.

Emergence Pattern or function at the global level of a system appears solely from local interactions among the lower-level components of the system.

Acronyms

MAS	Multi-Agent System
AMAS	Adaptive Multi-Agent Systems
NCS	Non Cooperative Situation
EPE	Emergent Programming Environment

Solutions

Problems of Chapter 1

Answers to questions concerning the notion of cooperation.

(Note : as most of the problems of this chapter are thinking exercises with fuzzy boundaries, we will only provide help and guidelines here, instead of complete solutions.)

1.1 Carrying heavy boxes
Help : in the case where robots can not communicate, what would happen if two robots where to try and lift a heavy box at the same time by chance ? Maybe if there are enough robots, a robot only has to wait a little when trying to lift a heavy box for if another robot happens to try to lift the same box, they will actually be "cooperating" !

1.2 Foraging Ants and Cooperation
Help : you can start by asking yourself what situation could be qualified as non-cooperative. For instance, if you have two ants searching for food, one going north to south, the other south to north, and if they meet each other, it would be useless for each ant to pursue its path, since the other has already explored this direction.

1.3 Cooperation in an Open System
Help : negotiation protocols can ensure a certain regulation in these systems. Reader can also take inspiration from the Game Theory field or Peer-to-Peer algorithms. In particular, reader can start by looking at the "Tit for Tat" strategy).

1.4 Classifying Cooperation Means
Help : for instance, sending a message asking for help is a suitable way to enable cooperation as long as there exists other entities in the system willing to answer

such a call. This case would be qualified as "explicit cooperation using direct communication between willing entities".

1.5 Colour Cubes Game

Help : list all the non-cooperative situations a robot can face when entering a room. Structure your algorithm as a subsumption architecture constituted of 3-parts modules as follows:

- *situation type and priority*
- *detection conditions*
- *resolving action*

References

These are the references of *chapter 9 - Cooperation* to be included in the global bibliography section.

1. Axelrod, R.: The Evolution of Cooperation. Basic Books (1984)
2. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Engineering Self-Adaptive Multi-Agent Systemsă: the ADELFE Methodology, chap. 7, pp. 172–202. Idea Group Publishing (2005)
3. von Bertalanffy, L.: General System Theory. George Braziller, New York (1968)
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press (1999)
5. Camps, V., Gleizes, M.P., Glize, P.: A self-organization process based on cooperation theory for adaptive artificial systems. In: 1^{st} International Conference on Philosophy and Computer Science "Processes of evolution in real and Virtual Systems", Kraków, Poland (1998)
6. Capera, D., Gleizes, M.P., Glize, P.: Mechanism type synthesis based on self-assembling agents. Journal on Applied Artificial Intelligence **18**(8-9) (2004)
7. Dorigo, M., Di Caro, G.: The Ant Colony Optimization Meta-Heuristic. McGraw-Hill (1999)
8. Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley Professional (1999)
9. Georgé, J.P., Edmonds, B., Glize, P.: Making Self-organizing adaptive multi-agent systems work, chap. 16, pp. 321–340. Kluwer Publishing (2004)
10. Georgé, J.P., Gleizes, M.P.: Experiments in Emergent Programming using Self-organizing Multi-Agent Systems. In: Multi-Agent Systems and Applications IV, 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005, Budapest, Hungary, September 15-17, 2005, *LNCS*, vol. 3690, pp. 450–459. Springer (2005)
11. Georgé, J.P., Gleizes, M.P., Glize, P., Régis, C.: Real-time simulation for flood forecast: an adaptive multi-agent system staff. In: D. Kazakov, D. Kudenko, E. Alonso (eds.) Proceedings of the AISB'03 symposium on Adaptive Agents and Multi-Agent Systems(AAMAS'03), pp. 109–114. SSAISB, University of Wales, Aberystwyth (2003)
12. Gleizes, M.P., Camps, V., Glize, P.: A theory of emergent computation based on cooperative self-oganization for adaptive artificial systems. In: Fourth European Congress of Systems Science. Valencia, Spain (1999)
13. Gleizes, M.P., Glize, P., Link-Pezet, J.: An adaptive multi-agent tool for electronic commerce. In: The workshop on Knowledge Media Networking IEEE Ninth International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 2000). Gaithersburg, Maryland (2000)
14. Goldberg, D.: Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley (1989)

15. Grassé, P.P.: La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insectes Sociaux **6**(1), 41–80 (1959). DOI 10.1007/BF02223791. URL http://dx.doi.org/10.1007/BF02223791
16. Haken, H.: Synergetics: an introduction. Springer (1978)
17. Heylighen, F.: Evolution, Selfishness and Cooperation; Selfish Memes and the Evolution of Cooperation. Journal of Ideas **2**(4), 70–84 (1992)
18. Holland, J.H.: Adaptation in Natural and Artificial Systems. The MIT Press (1992)
19. Huberman, B.: The performance of cooperative processes. MIT Press / North-Holland (1991)
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). DOI 10.1126/science.220.4598.671. URL http://dx.doi.org/10.1126/science.220.4598.671
21. Lucidarme, P., Simonin, O., Liégeois, A.: Implementation and Evaluation of a Satisfaction/Altruism Based Architecture for Multi-Robot Systems. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA, pp. 1007–1012. IEEE (2002)
22. Maturana, H.R., Varela, F.J.: Autopoiesis and Cognition : The Realization of the Living (Boston Studies in the Philosophy of Science). Springer (1991)
23. McCarthy Gallagher, J., Reid, K.: The Learning Theory of Piaget and Inhelder. Authors Choice Press (2002)
24. McLaughlin, B.P.: Emergence and supervenience. Intellectica: Emergence and explanation **2**(25), 25–43 (1997)
25. Minsky, M.: Steps toward artificial intelligence. In: E.A. E. A. Feigenbaum, J. Feldman (eds.) Computers and Thought, pp. 406–450. cGrawHill Book Company (1963)
26. Picard, G.: Agent Model Instantiation to Collective Robotics in ADELFE . In: M.P. Gleizes, A. Omicini, F. Zambonelli (eds.) Fifth Internatinal Workshop on Engineering Societies in the Agents World (ESAW'04) , Toulouse, France, 20/10/04-22/10/04, pp. 209–221. Springer Verlag, LNCA 3451 (2004). URL http://www.irit.fr/ESAW04
27. Picard, G., Gleizes, M.P.: An Agent Architecture to Design Self-Organizing Collectives: Principles and Application. In: AISB'02 Symposium on Adaptive Multi-Agent Systems (AAMASII), vol. LNAI 2636, pp. 141–158. Springer-Verlag (2002)
28. Picard, G., Gleizes, M.P.: The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems. In: F. Bergenti, M.P. Gleizes, F. Zambonelli (eds.) Methodologies and Software Engineering for Agent Systems. Kluwer (2004)
29. Picard, G., Gleizes, M.P.: Cooperative Self-Organization to Design Robust and Adaptive Collectives. In: 2nd International Conference on Informatics in Control, Automation and Robotics (ICINCO'05), 14-17 September 2005, Barcelona, Spain, Volume I, pp. 236–241. INSTICC Press (2005)
30. de Rosnay, J.: The Macroscope: a New Worl Scientific System. Harper & Row (1979)
31. Vaughan, R., Støy, K., Sukhatme, G., Matarić, M.: Blazing a trail: Insect-inspired resource transportation by a robotic team. In: Proceedings of 5th International Symposium on Distributed Robotic Systems (2000)
32. Vaughan, R., Støy, K., Sukhatme, G., Matarić, M.: Go ahead make my day: Robot conflict resolution by aggressive competition. In: Proceedings of the 6th International Conference on Simulation of Adaptive Behaviour (2000)
33. Weiss, G.: Multiagent systems. A modern approach to distributed artificial intelligence. The MIT Press (1999)
34. Wolpert, D.H., MacReady, W.G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation **1**, 67–82 (1997)
35. Wooldridge, M.: An introduction to multi-agent systems. John Wiley & Sons (2002)
36. Wu, B.: Manufacturing Systems Design & Analysis: Context & Techniques. Springer (1994)
37. Zwirn, H.: Les limites de la connaissance. Éditions Odile Jacob, coll. Sciences (2000)