The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents

Davy Capera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Pierre Glize

The AMAS theory for complex problem solving based on self-organizing cooperative agents

Davy Capera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Pierre Glize
Institut de Recherche en Informatique de Toulouse
118, route de Narbonne, 31062 Toulouse Cedex
{george, picard, gleizes, glize}@irit.fr

Abstract

In this paper, we present an approach for the design of complex adaptive systems, based on adaptive multi-agent systems and emergence. We expound the AMAS theory (Adaptive Multi-Agent Systems) and its technical working. This theory gives local agent design criteria so as to enable the emergence of an organization within the system and thus, of the global function of the system. We also present the theorem of functional adequacy witch ensures that a cooperative self organizing system performs a suitable work. Applications of this theory in the multi-agent system framework led us to define the architecture and a general algorithm for cooperative agents.

The originality of our approach lies in the very generic manner our re-organization rules work and that they are completely independent from the function the system has to compute.

1. Introduction

When confronted to complex phenomena in natural and social systems, the main problem of classical approaches is of course the building of an appropriate model. For some years now, we successfully used adaptive multi-agent systems for the solving of dynamic and complex problems. In this paper we want to describe our way to build systems when the classic computer techniques are not suitable, so this paper is about the use of self-organization as a design basis. An organisation between the agents enables them to produce a collective function, that is why, by way of the emergence of organisations, we think that the important point is the global function that emerges [5]. The originality of the AMAS theory (Adaptive Multi-Agent Systems), described in this article, is that it gives the basis for the design of the agents by giving criteria that are local to the agents and that guide their behaviour so as to make their organisation emerge.

The first part of this paper explains the motivations of our approach which consists of using emergence and self-organization as a mean to palliate the weaknesses of current techniques. The second part explains the AMAS theory which is the foundation of our way to handle adaptive MAS, focussing on the functional adequacy issue. The third part is the technical part associated with the theory and which guides the design process of self-organising agents according to the AMAS theory. We then present the cooperative algorithm which describes the basic behaviour of a cooperative agent.

1.1. Goals and motivations

The way we usually conceive computational systems leads to the need, for the designer, to have some important initial knowledge: first, the exact purpose of the system, and second, every interaction to which the system may be confronted in the future. This point of view where we leave no free margin for the system's operation was certainly guided by two converging considerations since the beginning of computer science:

- to guarantee in the most formal possible way that the system effectively computes the "right" function,
- to optimise memory capacities, computing speed and the very limited proprioceptive capacities of the first computers.

The other side of the coin of this "total control" is an ever-growing design task for the developer with the constant increase of computer power and their inter-connections. That is the reason for the existence of so many software development techniques, which nevertheless only manage to slow down – without stopping – the increase of human energy needed for this development.

1.2. The evolution of systems

The evolution of computer science forces us to consider that it is more and more difficult – if not impossible – not only to control accurately the activity of software with increasing complexity but also to describe completely how their work [13]. These two aspects are exemplified in the notions of autonomic computing and ubiquitous computing:

1. *"Even if we could somehow come up with enough skilled people, the complexity is growing beyond human ability to manage it. As computing evolves, the overlapping connections, dependencies, and interacting applications call for administrative decision-making and responses faster than any human can deliver. Pinpointing root causes of failures becomes more difficult, while finding ways of increasing system efficiency generates problems with more variables than any human can hope to solve. Without new approaches, things will only get worse."* [6].
2. "*The Ubiquitous Computing era will have lots of computers sharing each of us. Some of these computers will be the hundreds we may access in the course of a few minutes of Internet browsing. Others will be imbedded in walls, chairs, clothing, light switches, cars - in everything. Ubiquitous Computing is fundamentally characterized by the connection of things in the world with computation. This will take place at a many scales, including the microscopic"* [11].

Indeed, the notion of distribution due to the Internet or the Web becomes more and present in many applications. So you cannot think of having a global control over this type of applications. Moreover, the power of the computers enables us to compute more and more complex applications. This complexity comes from the fact that the system is composed of many interacting entities that can be autonomous, heterogeneous and evolutionary [8]. Finally, the many connecting possibilities between the different hardware imply that we have to take into account open and heterogeneous environments and systems, moreover more and more dynamic. One of the possibilities to counter these difficulties is to give more autonomy to the software so as to enable it to adapt itself as good as it can to unexpected events.

Making machines more autonomous is a way to simplify the task of the designer. Formal theories can help us to represent and reason about time, space and dynamics of an evolving world. However, we can also consider that such specifications are useless or even impossible in some situations:

- The system's environment is dynamic, making it ineffective to enumerate exhaustively all the situations the system may encounter.
- The system is open and therefore dynamic because it is constituted of a shifting number of components.
- The task the system has to achieve is so complex that we cannot guarantee a perfect design.
- The way by which the system may achieve the task it has been assigned is difficult or even impossible to apprehend globally by the designer.

The ability for artificial systems to confront really unexpected situations (like the ones resulting from a lack of spatial or temporal knowledge) implies a research mainline that leads to a different system design method than the traditional global top-down approach based on the modelling of the world [7].

2. The AMAS Theory

The first paragraph explains the central role of the system's organisation for its global function. The second one brings the theoretical justification of the interdependence that can exist between a cooperative local behaviour and the functional adequacy of the collective global function.

2.1. Adapt the system by its parts

By specifying *a priori* a model for a system that will have to deal with unexpected events, you constrain (maybe inopportunely) the space of possibilities. Since Bertalanffy [1], many authors 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.*".

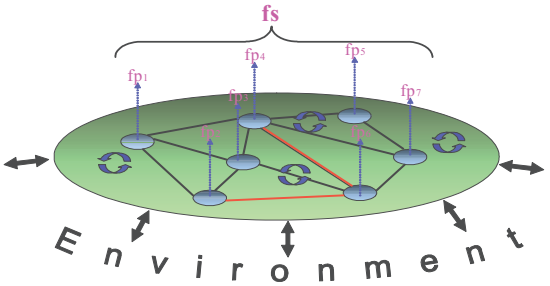

Figure 1. Interacting agents constitute the system.

Following this view, we consider that each part Pi of a multi-agent system S achieves a partial function fp_i of the global function fs (figure 1). fs is the result of the combination of the partial functions fp_i, noted by the operator "**o**". The combination being determined by the current organisation of the parts, we can deduce $fs = fp_1$ **o**

fp_2 **o** … **o** fp_n. As generally fp_1 **o** $fp_2 \neq fp_2$ **o** fp_1, by transforming the multi-agent organisation, you change the combination of the partial functions and therefore you modify the global function fs. Therefore, this is a way to adapt the system to the environment. A pertinent technique to build this kind of systems is to use adaptive multi-agent systems. As usually meant Wooldridge by "multi-agent systems" [12], we will be referring to systems constituted by several autonomous agents, plunged in a common environment and trying to solve a common task.

2.1. The theorem of functional adequacy

We define the functional adequacy of a system by the fact that a system adequately carries out the task for what it was conceived.

Theorem. *For any functionally adequate system, there is at least a co-operative internal medium system that fulfills an equivalent function in the same environment.*

The demonstration of this theorem in [3] results from the application of the following axiom and the four lemmas. For each of them we have added a short textual explanation.

Axiom. *A functionally adequate system has no antinomic activity on its environment*.

The veracity of this assertion could be proved if we are an external observer of all the systems and their environments in order to avoid any perturbation. This cannot exist in our physical world[1].

Lemma 1. *A cooperative system is functionally adequate*.

A cooperative system has only beneficial activities for its environment. Without any antinomic activity for these systems, the previous lemma can be used.

Lemma 2. *For any functionally adequate system S there exists at least a cooperative system S* which is also functionally adequate in the same environment*.

The demonstration uses a thinking experiment in order to construct the system S* from the initial system S. It has four steps: specifying an algorithm to construct a cooperative system, showing the termination of the algorithm, proving that the new system realizes a function equivalent from the system S, concluding that this is a right functionally adequate system S*.

Lemma 3. *Any system having an internal cooperative medium is functionally adequate*.

An internal medium of a system contains all its parts and physical supports needed to their exchanges. A system with cooperative internal medium has only cooperative exchanges with its environment because these exchanges are a subset of its parts interactions.

Lemma 4. *For any cooperative system, there exists at least a cooperative internal medium system that is also functionally adequate in the same environment*.

The method is identical to the lemma 2. The reasoning process involves all the system parts. The cardinality of the parts is assumed finite for any real system.

The theorem is easily obtained by operations of surjection and inclusion of the systems sets defined in the lemmas (functionally adequate, cooperative, cooperative internal medium). This result allows focusing only on very particular systems (those with cooperative internal mediums) to obtain functionally adequate systems in a given environment. They have several properties [4]:

- A cooperative system in the environment is functionally adequate, which prevents it to know the global function that it has to realize in order to adapt itself.
- This theory does not impose any goal on the system. According to its perceptions of the environment, it will act so as to keep its behavior cooperative according to its skills, representations of itself, other agents and environment.
- The feedback concept is not constraining in this theory because the system must only evaluate if the changes taking place in the medium are cooperative from its point of view without knowing if these changes are dependent on its own past actions.

Nevertheless, the theories are not in the technology, even if they are embodied in it. We will see that the agents and multi-agents are a technological framework that is essential for the application of the AMAS theory.

3. The AMAS Technology

In this chapter, we show how this theory can be applied to adaptive multi-agents systems. We explain the basic behaviour of an agent facing locally uncooperative situation and the consequence at the organization level.

[1] This axiom plays a similar role as the Church's thesis for the class of effective computable functions.

3.1. The architecture of an AMAS agent

The objective is to design systems that do the best they can when they encounter difficulties. These difficulties can be seen like exceptions in traditional programs. From an agent point of view, we call them Non Co-operative Situations (NCS). 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). More precisely three kinds of non co-operative situations can be detected by the agent:

- when a perceived signal coming from the environment is not understood or is ambiguous,
- when perceived information does not produce any activity of the agent,
- when the conclusions are not useful to others.

A co-operative agent in the AMAS theory has the four following characteristics. First, an agent has activity autonomy: an agent has the ability to decide to say “no” or 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 detect non-cooperative situations and acts to return in a co-operative 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. Generally, five parts are essential for a co-operative agent for a coherent collective behavior to be observed starting from the aggregation of individual behaviors.

1. Skills are knowledge of a particular field which enables the agent to perform the partial function which is assigned to him. No technical constraint is imposed for the development (production system, object-oriented method …).
2. Representation of itself, other agents and environment confers to the agent a belief on what the agent knows of itself, others and of its environment.
3. The social attitude enables the agent to modify its interactions with other agents. This is based on what we call the cooperation: if an agent detects a non-cooperative situation, it acts to return to a cooperative state.
4. An interaction language is necessary for agents to communicate directly or not.
5. Aptitudes are capacities an agent possesses to reason on its representations and its knowledge.

More formally, the behaviour of an AMAS agent can be described as an algorithm based on the one found in the section 4.

3.2. The macro-level organization

For a multi-agent system, implementing this adaptation 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. We start from the principle that, to have a good 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, an agent always receives relevant information for it to compute its function, and it always transmits relevant information to others. The designer provides the agents with local criteria to discern between cooperative and non-cooperative situations. The detection and then elimination of non-cooperative situations between agents constitute the engine of self-organisation.

Thus, depending on the real-time interactions the multi-agent system has with its environment, the organisation between its agents emerges and constitutes an answer to the aforementioned difficulties (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 fp_i, but the combination of all the partial functions produces the global emergent function fs. Depending on the interactions between themselves and with the environment, the agents change their interactions i.e. their links. This is what we call self-organisation.

By principle, the emerging purpose of a system is not recognizable by the system itself, its only criterion must be of strictly local nature (relative to the activity of the parts which make it up).

4. Neocomputation or the future issues for the Amas theory

In this section we present and explain the cooperative algorithm that arises from AMAS-related considerations and analyze its relevance for open computational systems.

4.1. The cooperative algorithm

The following algorithm (figure 2) may be viewed as a formal representation of the cooperative attitude of the

agents exposed previously: according to the AMAS theory, agents have to be able to detect when they are in a non cooperative situation and in which way they can act to come back in a cooperative situation. There are two main possible "states" to which the decision process could be faced: either the set of applicable cooperative actions- related to perceptions- is not empty, or it is. In the first instance, the agent chooses the highest priority action among the selected ones; otherwise the agent is in non cooperative situation and has to select the highest priority action according to the situation perceived. 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 non cooperative situations and the actions which could be applied to solve them are not generic: designers have to write their own- and so specific- non cooperative situations set and related actions for each kind of agent they wish the system to contain. This work must be performed during the design of agents: the designer must find exhaustively all the non cooperative situations which could occur for each kind of agent and, for each one, find the relevant actions which could solve the lack of cooperation.

4.2. Amas and open computational systems

As seen in the algorithm of an Amas agent, its behavior depends only on its skills, the current state of the world, and the beliefs about other agents; there is no need of information about the global goal of the system. Two classes of systems can be designed:

- If we suppose an Amas system composed of a finite set of agent classes (fully specified at the initial design phase), because the new agents skills remain constant, the agent entities can easily appear and disappear dynamically without any agent modification. The system adapts to unexpected change in the environment by modifying only its internal organization.
- In the other cases, agents have some learning capabilities. Thus the agents in contact must be able to change their beliefs about each other in order to act cooperatively in the future. By viewing each piece of belief as an autonomous entity, the ability to learn on the beliefs level can also be seen as an Amas self-organization process (this discussion is outside of the scope of the paper).

An Amas system adapts its behavior in a totally decentralized way, and is able to provide coherent global behavior because its entities have a "cooperative will". In our opinion, decentralized adaptation and coherent global behaviour assurance are the basic requirement for open computational systems defined as "complex and dynamic structures of autonomous entities". We put in this field all the applications we call *neocomputation* applications, namely: autonomic computing, pervasive computing, ubiquitous computing, emergent computation, ambient

```
c ⊆ P                                   /* c is a subset of percepts */
a ⊆ A                                   /* a is an action */
SR ⊆ SkillRules                         /* SR is the subset of applicable behaviors from the skill rules */
BR ⊆ BeliefRules                        /* BR is the subset of applicable behaviors from the belief rules on other agents */
NCSR ⊆ NonCooperativeSituationRules     /* NCSR is the set of applicable behaviors in the current non cooperative situation */
R = SR ∪ BR ∪ NCSR ⊆ Behavior Rules     /* R is the total set of applicable behaviors */
(c,a) / c ⊆ P and a ⊆ A                 /* (c,a) is a behavior */
(c1,a1)   (c2,a2)                       /* express a priority relation of the behavior (c1,a1) on (c2,a2) */

/* The action function allows the agent to decide what action is relevant at the present time according to the perceptions p */
Function action (p:P) : A
var fire :f(SR)                         /* fire is the set of possible behaviors obtained from the f function */
var fireCooperativeAction : f'(BR)      /* fireCooperativeAction is the set of possible cooperative behaviors obtained from the f'
                                        function */
var fireNCS : f"(NCSR) )                /* fireNCS is the set of behaviors deleting the current non cooperative situations obtained
                                        from the f" function */
var a :A

begin
  fire := {(c,a) / (c,a) ∈ SR, p∈c}
  fireCooperativeAction := {(c',a) / (c,a) ∈ fire and (c',a) ∈ BR and c'=c ∪ c" et p ∈ c ∪ c" }
  if fireCooperativeAction <> Ø then    /* If fireCooperativeAction is not empty, then the agent is in cooperative situation */
   for each (c,a) ∈ fireCooperativeAction do
        if (∄(c', a') ∈ fireCooperativeAction / (c', a') ≺ (c,a)) then return a
        endif
   endfor
   else                                 /* Non cooperative situations are detected */
                                        /* If fire = Ø and fireCooperativeAction = Ø, then the agent cannot use its percepts it
                              is a non cooperative */
                                        /* If fire <> Ø and fireCooperativeAction = Ø, then the agent cannot do 'a' because it
                              beliefs 'a' non cooperative for other agents */
   fireNCS := {(c,a) / (c,a) ∈ NCSR and p∈ c}
                                        /* By consequence, the agent must fire a behavior belonging to NCSR in order to suppress
                              the non cooperative state */
    for each (c,a) ∈ fireNCS do
        if (∄(c', a') ∈ fireNCS / (c', a') ≺ (c,a)) then return a
        endif
    endfor
  endif
return null
End function action
```

Figure 2. The algorithm formalizing the cooperative behaviour of an agent.

intelligence, amorphous computing … This set of neocomputation problems have in common the inability to define the global function to achieve, and by consequence to specify at the design phase, a derived evaluation function for the learning process. Nevertheless, in the case of classical computation, Amas theory is irrelevant because the goal assigned to a system can control efficiently the behavior of the system in a centralized way.

5. Conclusion and Perspectives

In the widely unexplored field of the manipulation of organization, we chose the path of emergence. Therefore, in this paper, we presented, on the theoretical level, the concept of emergence and the AMAS theory referring to it. We also presented the cooperative algorithm which describes the typical behaviour of a cooperative agent built in accordance with the AMAS theory. Throughout the paper, we explained why we think that this original approach for the manipulation of dynamic multi-agent organisations is especially relevant for many applications.

The AMAS theory allows to build systems in which the global function is really emergent considering the agents activity which realize very simple treatments, and considering their self-organization process conducted by adjustment rules (uselessness, conflict, concurrency) which do not depend of the expected global function. The AMAS theory is part of the field of the artificial emergent systems study which is one of the main challenges for the next years (cf. part 1.2).

If we had to describe in short the AMAS theory, we would say that it concerns itself with making the system build itself and not with building the system ourselves. We know the desired global behaviour and since we gave the system the ability to adapt, it will find a solution.

It is a non-linear problem because of the current behaviour of an agent depending of the previous behaviours of its neighbours, which depend themselves of the previous activity of the first agent, and so on...

Keith Sawyer says [9]: “In many computational models of emergence, the end state of the system is of primary interest. In collaborative emergence, the most interesting aspects of the simulation will be the incremental processes of emergence and downward causation”. The AMAS theory compliant systems are example of collaborative emergence, in which the goal is not to obtain a given end state but a never ending adaptation process because the systems are plunged in a dynamic environment. We think that it is an important class of MAS simulating hard problems and having specific characteristics (dynamic environment, non-termination, huge space search).

In order to test a theory (by falsification or validation) a great amount of experiments must be done in various fields. For this reason, we have implemented the AMAS theory on many applications like ants foraging, e-commerce, flood forecast, equation solving or traffic routing in telephonic network. In these cases, AMAS technology gives at least similar performances than other methods when they exist (www.irit.fr/SMAC). Among others, this approach allows us to make abstraction of constraints like theoretical models, complex protocols, planning tools, and nevertheless be appropriate for the design of open and heterogeneous systems, needing complex interactions, coordination and learning.

Our goal in the near future is to spread this way of designing MAS. For this, we are actually working on a design methodology called ADELFE (laboratory for the design of emergent functionality software) for the engineering of open and evolutionary systems [2].

6. References

[1] Bertalanffy von, L. (1968) “General System Theory“ New York, George Braziller.

[2] Bernon C., Gleizes M.-P., Picard G., Glize P. (2002) “The ADELFE Methodology for an Intranet System Design“ Agent Oriented Information Systems Workshop, Toronto.

[3] Camps V., Gleizes M.P., Glize P. (1998) “Une théorie des phénomènes globaux fondée sur des interactions locales“ Proceedings of 6th francophone conference on multi-agent systems, Editions Hermès.

[4] Gleizes, M.P., Camps V., Glize P. (1999) “A theory of emergent computation based on cooperative self-organization for adaptive artificial systems“ Fourth European Congress on Systemic, see also http://www.irit.fr/SMAC.

[5] Goldstein J. (1999) “Emergence as a Construct: History and issues“ Emergence Volume 1, Issue 1.

[6] Horn P.M. “Autonomic computing - IBM's Perspective on the State of Information Technology” http://www.ibm.com/research/autonomic (2001).

[7] Liu, J., (2001), "Autonomous Agents and Multi-Agent Systems – Explorations in Learning, Self-Organization and Adaptive Computing", World Scientific, ISBN 981-02-4282-4.

[8] Parunak H. van Dyke, Vanderbok R.S. (1997) “Managing emergent behavior in distributed control systems“ Proceedings of ISA-Tech'97

[9] Sawyer, R.K. (2000) “Simulating Emergence and Downward Causation in Small Groups”, Lecture Notes in Artificial Intelligence, Vol. 1979,Springer Verlag Berlin Heidelberg,(2000) 49-67

[10] Weiß, G. (1996) “Distributed artificial intelligence meets machine learnin“. Learning in multi-agent environment. ECAI’96 Workshop LDAIS, ICMAS’96 Workshop LIOME. Springer

[11] Weiser M., Brown J. S. (1996) “Designing Calm Technology“, PowerGrid Journal, Vol. 1. N°1, http://powergrid.electriciti.com/1.01

[12] Wooldridge M. (2002) “An introduction to multi-agent systems“ John Wiley & Sons

[13] Zambonelli F. and Van Dyke Parunak H. (2002) – Signs of a Revolution in Computer Science and Software Engineering – *In Proceedings of Engineering Societies in Agent World (ESAW’02)*, Madrid.