S-DLCAM: A Self-Design and Learning Cooperative Agent Model for Adaptive Multi-Agent Systems
Wafa Mefteh, Frédéric Migeon, Marie-Pierre Gleizes, Faiez Gargouri

To cite this version:

HAL Id: hal-01148192
https://hal.archives-ouvertes.fr/hal-01148192
Submitted on 4 May 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: http://oatao.univ-toulouse.fr/
Eprints ID: 12478

To link to this article: DOI: 10.1109/WETICE.2013.58
URL: http://dx.doi.org/10.1109/WETICE.2013.58


Any correspondance concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr
S-DLCAM: A Self-Design and Learning Cooperative Agent Model for Adaptive Multi-Agent Systems

Wafa MEFTEH**, Frédéric MIGEON**, Marie-Pierre GLEIZES*, and Faiez GARGOURI**
*University of Paul Sabatier, IRIT, 118 Route de Narbonne, 31062 Toulouse, Cedex 9, France
(mefteh, migeon, gleizes)@irit.fr
**University of Sfax, MIRACL, Pôle Technologique de Sfax BP 242, 3021, Sakiet Ezizit Sfax. TUNISIE.
(faiez.gargouri@fsegs.rnu.tn), (wafa.mefteh07@gmail.com)

Abstract—Given the incomplete knowledge that an Adaptive Multi Agent System (AMAS) has on its dynamic environment, the detection and the correction of problems encountered called Non Cooperative Situations for the construction of the good behaviour of the AMAS agent can challenge even the most experienced designer. Our goal is to help the AMAS designer in his task by providing an agent behaviour able to self-design. In this paper, we propose a self-design and learning cooperative agent model.

Keywords—Adaptive Multi-Agent Systems; Cooperative Agent; Self-Design and Learning Cooperative Agent Model.

I. INTRODUCTION

For Adaptive Multi-Agent System (AMAS) [1], the development of adaptation implies the need to focus on the agent level. This is to give the agent the means to decide autonomously to change its relationships with other agents in order to move toward a cooperative organization. Thus, depending on the interactions that the AMAS has with its environment, the organization between its agents emerges. Building such self-organized systems is not a trivial task. In this paper, we propose a new cooperative agent model based on Self-Design and learning mechanisms developed from the agent model associated with the AMAS theory [1], [2], [3]. We take in account the following important works: [4] (in which Capera et al. present a model based upon a sort of extended automata product, dedicated to multi-agent systems) and [5] (in which Russel and Norvig present how an agent can find a sequence of actions that achieves its goals, when no single action will do). Indeed, we consider that the Self-Design and Learning Cooperative Agent (S-DLCA) life cycle goes through two levels: the preliminary level (PL) (nominal and cooperative behaviour) given by the designer and the heigh level (HL) which is responsible of the detection and correction of the Non Cooperative Situations (NCS) that the agent may encounter during its life. This model was developed under SeSAm (http://www.simsesam.de/) and it can be used by any AMAS designer in order to help him in the detection and correction of the NCS using the new ADELFE methodology extensions [6].

II. REDEFINITION AND LOCATION OF THE NON COOPERATIVE SITUATIONS IN THE AGENT LIFE CYCLE

We consider that the agent life-cycle goes throw three phases: Perception & Interpretation (P & I), Reasoning & Decision (R & D) and Communication & Action (C & A). We identify new types of Non Cooperative Situations that an AMAS agent may encounter and we locate them with the old ones in the agent life cycle (Table I).

<table>
<thead>
<tr>
<th>P &amp; I</th>
<th>R &amp; D</th>
<th>C &amp; A</th>
</tr>
</thead>
<tbody>
<tr>
<td>• INCOMPLETE: the agent is unable to exploit a given interpretation in its reasoning or decisions;</td>
<td>• INCOMPETENCE: the agent is unable to extract the information content of a signal;</td>
<td>• INABILITY: this NCS can be encountered when the agent’s reasoning leads to a conclusion that does not agree with its knowledge;</td>
</tr>
<tr>
<td>• AMBIGUITY, UNCERTAINTY: the agent is not sure about the interpretation that it assigns to a perception;</td>
<td>• AMBIGUITY, EQUIVOcation: the agent gives two (or more) interpretations to the same perception.</td>
<td></td>
</tr>
<tr>
<td>• INCOMPLETE: the agent is unable to exploit a given interpretation in its reasoning or decisions;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• INCOMPETENCE: the agent is unable to exploit a given interpretation in its reasoning or decisions;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• AMBIGUITY, EQUIVOcation: the agent gives two (or more) interpretations to the same perception.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• AMBIGUITY, EQUIVOcation: the agent gives two (or more) interpretations to the same perception.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• AMBIGUITY, EQUIVOcation: the agent gives two (or more) interpretations to the same perception.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• AMBIGUITY, EQUIVOcation: the agent gives two (or more) interpretations to the same perception.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table I

REDEFINITION AND LOCATION OF THE NCS IN THE AGENT LIFE-CYCLE

III. THE STATIC STRUCTURE OF THE PROPOSED AGENT MODEL

Figure 1 presents the static view of our agent model and its different modules are defined in table II.
IV. THE DYNAMIC BEHAVIOUR OF THE PROPOSED AGENT MODEL

The agent perceives its environment and stores its perceptions in the list of perceptions ($L_p$). It gives interpretations about these perceptions and stores them in the list of interpretations ($L_i$). It connects each perception "p" with the predicates "pr" it knows and likely to be perceived. $D_r(p, i)$ represents the degree of certainty of the interpretation given to this perception. Algorithm 1 presents how the agent can detect an NCS related to the first phase ($NCS_p$). Then, the agent reasons on the $L_i$ using its Knowledge Base ($K_b$). It gives conclusions which are stored in the list of conclusions ($L_c$) and it decides what activities to perform and save them in the list of decisions ($L_d$) and finally schedules them and stores them in the list of scheduled activities ($L_s$).

Algorithm 2 illustrates the detection of the NCS related to the second phase ($NCS_r$). It is an INCOMPETENCE when an interpretation "i" is not used to produce any conclusion or to make any decision. It is an Unproductiveness when the given conclusion already exists in $K_b$. It is an Incoherence when the given conclusion breaks one or more agent’s rules. Finally, it performs activities elaborated in the R & D phase and saves each performed action in the list of performed actions ($L_pa$).

Algorithm 3 illustrates the detection of the NCS related to the third phase ($NCS_s$). A decided activity is not performed because there is a Conflict and/or a Concurrence and/or a Uselessness and/or the agent hasn’t the skills that enable it to perform this activity. It is a Conflict

**Table II**

<table>
<thead>
<tr>
<th>Sensors</th>
<th>This module enables the agent to sense its environment.</th>
</tr>
</thead>
</table>
| Knowledges Representation | This module represents the language of representation used by the designer to represent the agent’s knowledge.
| Recognition | This module enables the agent to recognize/interpret what it perceives. |
| Knowledge Base | This module represents the agent’s knowledge about itself and its environment. |
| Reasoning Capabilities | This module is given by the designer in order for the agent to reason on its interpretations using its knowledge. |
| Goals | This module represents the local goal that the agent must achieve in the form of a future agent state. |
| Skills | This module represents the atomic actions and the sequences of actions that the agent can perform. |
| Cooperative Utility Criteria | This module is responsible for the utility measurement of an agent state. It allows the agent to choose between different future states that can lead to its objective. |
| Next-Best-Action Search Cooperative attitude | This module is responsible for searching for the best next action to decide. The next best action is decided cooperatively. |
| Interaction Language Effectors | This module represents the protocols used by the agent to interact with other agents. |
| Other Agents Criticalities Learning / Adaptation | This module supports the criticalities of the other agents. The agent must behave in the way that permits it to equilibrate the criticalities of the other agents. |
| Agent criticality | This module is responsible for the learning of the agent in order for it to adapt to its environment and achieve the functional adequacy of the system. |

**Figure 1.** The static view of the Self-design and Learning Cooperative Agent.
Algorithm 3 Detection of $NCS_a$

```plaintext
Inability(a) ← FALSE
L_{npa} ← L_{sa} \{ L_{pa}\}
for all $a_1 \in L_{npa}$ do
    Inability(a1) ← TRUE
    Uselessness(a1) ← TRUE
    L_{rest} ← L_{sa} \{ a1\}
for all $a_{rest} \in L_{rest}$ do
    if ($a_{1.ad} \subseteq a_{rest-pr}$) then
        Uselessness(a1) ← FALSE
    end if
end for
for all $Ag \in L_{pag}$ do
    Conflict(a1, Ag) ← FALSE
    Concurrency(a1, Ag) ← FALSE
    for all $a_2 \in L_{Ag}$ do
        if ($a_1.ad = a_2.ad \land a_1.de = a_2.de$) then
            Concurrency(a1, a2) ← TRUE
        else if ($a_1.de \subseteq a_2.pr$) then
            Conflict(a1, a2) ← TRUE
        else if ($a_1.ad \subseteq a_2.pr$) then
            Uselessness(a1) ← FALSE
        end if
    end if
end for
```

if the deletions made by the execution of an action $(a_{1.de})$ contain preconditions of a perceived agent’s action $(a_{2.pc})$. The perceived agents are stored in the list of perceived agents $(L_{pag})$. The actions to be made by a perceived agent “Ag” are saved in the list of actions to be performed by a perceived agent $(L_{Ag})$. It is a Concurrency if all of the agent’s action additions $(a_{1.ad})$ (and respectively all deletions $(a_{1.de})$ are among the additions $(a_{2.ad})$ (respectively deletions $(a_{2.de})$) of a perceived agent’s actions. It is a Uselessness when the additions made by the agent’s action $(a_{1.ad})$ are not part of the preconditions of its other actions $(L_{a1}.pc)$ nor the preconditions of perceived agent’s actions $(L_{a2}.pc)$.

To deal with the encountered NCS, the agent realizes a set of specific actions to go out from each type of NCS. For the Conflict, Concurrency and Uselessness, the agent can anticipate them if possible since the R & D phase. For each encountered NCS, the agent tries to follow the following actions:

- $\alpha$: Relate the current situation to other previous situations based on its experience in order to find a way to correct the current situation.
- $\beta$: if $\alpha$ it tries to ask the perceived agents if they can help it.
- $\delta$: if $\beta$ it asks the designer to improve his work by giving more examples (for an $NCS_p$), enhancing the reasoning capabilities and the knowledge base (for an $NCS_s$) or enhancing the skills (for an $NCS_s$).

We propose that the agent operates in two modes: the “Experimentation Mode” and the “Deployment Mode”. During the “Experimentation Mode”, when detecting a NCS, the agent can ask the designer if it can find a solution for it. When a correction of a NCS occurrence is proposed to the agent, it learns from this in order to avoid it the next time. After many executions, the agent should be able, during the Deployment Mode, to correct by itself the encountered NCS based on its learning from the many corrections made during the “Experimentation Mode”.

V. CONCLUSION

In this paper, we provide a new cooperative agent model for Adaptive Multi Agent Systems. This model is based on learning mechanisms to give the agent the ability to self-design. Our objective is to help the designer and facilitate his task by automating as much as possible the task of detection and correction of the Non Cooperative Situations.

REFERENCES


