Bringing Multi-Agent Systems in human organizations: application to an Information Multi-Agent System

Emmanuel ADAM and René MANDIAU

LAMIH UMR CNRS 8530, University of Valenciennes
Le Mont Houy - FRANCE – 59313 Valenciennes Cedex 9
{emmanuel.adam, rene.mandiau}@univ-valenciennes.fr

Abstract. Agents are more and more used to search information on Internet. However, they are principally used as single agents and not as a part of a multi-agent system. Indeed, only few projects use agents that communicate or collaborate between them. This lack of communication induces often users to be isolated in front of their computer. We think that it is necessary that the user of an information search system (like an actor of a technological watch cell or a searcher in a laboratory for example) be aware of what his/her colleagues search (or at least, has knowledge of a part of their searches). This should avoid redundancies of information and work, and should (re) generate a team feeling among the actors. That is why we propose, in this article, an information multi-agent system, which is composed itself of multi-agent systems located on the computer of each actor of a technological watch department, and which collaborate between them and with the actors. This multi-agent architecture has been chosen in agreement with actors of a real department, following the analysis and modeling of their activities. This method to integrate multi-agent system into human organization is also discussed in this article.

1 Introduction

The boom in Internet technology and companies networks has contributed to completely changing a good number of habits, which have been well established in companies for several decades. Enterprises are now set off on a race for information: being the first to find the good information becomes an essential objective to competitive enterprises. So, it is important to own a fast tool of information search and distribution. Admittedly, tools have already been suggested such as: search engines, meta-engines, tools for automatic search (which search at determined regular intervals), and, more recently, information agents, capable of searching information, of sorting and of filtering it.

The problem of these solutions is that they do not take into account the human factors, such as the notion of the group or even the man-machine co-operation. We have previously developed a method (AMOMCASYS, meaning the Adaptable Modeling Method for Complex Administrative Systems) to design and to set-up multi-agent systems within human organization, more precisely in the cooperative processes of these organizations [1]. So we have reused this method to develop multi-agent systems for helping cooperative information management within a team of
technological watch. The main advantage of our method, and of our system, is that it takes into account the cooperation between actors of workflow processes. Indeed, we have noticed that most of human organizations follow a holonic model (each part of the organization is stable, autonomous and cooperative and is composed of sub holonic organizations whose it is responsible) and we have built our method by integrating these notions.

This paper describes firstly the AMOMCASYS method that we have built to model human processes and to design multi-agent systems, which interact with actors of the processes. Then, the method that we propose for design such multi agent systems is shown in two steps: a step for the individual characterization of the agents; and a step for the design of the agents’ cooperative working. Finally, this article presents an application of our method to the design of multi-agent systems that help actors of a technological watch cells to search and exchange information.

An Adaptable Modeling Method for Complex Administrative Systems (AMOMCASYS)

Holonic principles

Before designing a multi-agent system to assist management of information within a human organization, we think that it is necessary to understand its working mechanisms. This should allow us to take into account all of its characteristics and to bring multi-agent systems in this one in a pertinent way.

We have shown in [1] that most of complex administrative systems follow the holonic model that was proposed by Arthur Koestler in 1969 [2].

A series of grouped rules defines holonic systems, which are called Open Hierarchical Systems (OHS). Here we propose to give an interpretation of these rules according to the multi-agent point of view. We can retain the following principles from the rules:

- A holonic system possesses a tree structure. In fact it can be seen as a set of interwoven hierarchies. Each holon is responsible of a layer of other holons.
- A holon is a *stable, autonomous et cooperative* part of the system and can be considered as a whole. Holons unite themselves to form a whole and each one can be broken down into holonic agents, this corresponds to the recursive breakdown of the problem into sub problems [3].
- A holon obeys precise principles, but is able to adopt different strategies according to its need.
The complex activities and behavior are situated at the top of the hierarchy; the interactions with the environment and the “simpler” reactive acts are located at the base of the holarchy.

The communications must follow the hierarchy and, according to the direction, must be filtered or detailed.

The hierarchy is defined by the responsibility that a holon, which compose the system, has on a process or a sub-process and so on the holons that act in it. For example, an organization by projects can be represented with a tree structure where each node (each holon) is responsible of a part of the project.

As the holonic architecture is well adapted on human organization where actors exchange between them some information, we have proposed to reuse this architecture to design multi-agent system that have to manage and exchange data.

Here we can find at least two of the characteristics of the agents in a MAS sense: autonomy and cooperation [4]. The third characteristic, the capacity to adapt itself to an environment is suggested by stability. A holon can therefore be seen as an agent whose stability is an essential point.

Our aim is the design of a multi-agent organization providing assistance to the actors in cooperative processes. This organization must be fixed (which does not imply rigidity) in order to be able to meet the user demands as quickly as possible. This is why we have used the social rules defined in the holonic concept in order to simplify and to accelerate the design of a multi-agent society (in the [5] sense). This holonic concept is especially useable and useful in structured and cooperative fields [3].

However, before setting-up a multi agent system, and more generally software, it is necessary to model the human organization in which it has to work. That is why we have proposed a method adapted to human organization where actors have different levels of responsibility and where actors have to cooperate around documents.

Use of AMOMCASYS

AMOMCASYS (meaning the Adaptable Modeling Method for Complex Administrative Systems) was designed for the modeling of the cooperative functioning of procedures in holonic human organization by integrating several software engineering methods after their comparisons [6].

The integration allows, in a relatively short period of time, to build a method suited to its needs, in our case: to have a clear method, allowing explicit description of

1 However, if the holons are stable, they do not have to be rigid. Indeed, the stability of the whole system is more important than the stability of each of its parts. So, it is sometimes necessary that some holons be temporarily destabilized so that the whole system can take more long-term protection strategies.
cooperation (communication, coordination, and collaboration) and the degrees of responsibility of the actors.

AMOMCASYS is made up of four models, a data model, a data flow model, a processing model and a dynamic model:

- Concerning the data model, the one proposed by UML [7] allows to represent the information’ structure (documents, ... and their relations (inheritance, ...). Regarding the specification of MAS, this model allows to define agents’ structure.
- The dataflow model, which represents the information flows between actors, is based on SADT dataflow model (IDEFO) [8] that we have adapted to represent the responsibilities’ levels of the actors and of the agents.
- If the activity model represents all possible flows between actors, it does not represent the conditions for which the information follows a particular path. That is why it is sometimes necessary to use a processing model. This model has to also represent cooperation between actors and their hierarchy and/or responsibility relationships. For this, we use the data processing model of OSSAD method [9]. As the previous model, this model can be used for MAS design, the actors having different responsibility levels are replaced by agents having different responsibility levels too. These two models allow us to check if the specified organization follows holonic communication rules. However, if these models can be reused in the low-level design step, they are not sufficient to lead to the high-level one. Indeed, there is an important lack relating to the dynamic of the studied human organization.
- So, the AMOMCASYS method is composed of a dynamic model that uses parameterized Petri nets [10]. This model implicates of defining three levels of rules of the process working: global rules, local rules and personal rules. This model is not yet used in the MAS design but only in the human organizations modeling.

This method, supported by a CASE tool (Visual Basic layer based on the commercial software VISIO), enabled us not only to reveal the key points of the procedures where the multi-agent systems should be brought, but also to improve them in an organizational way.

Three steps are necessary to set-up a MAS with AMOMCASYS: firstly, the processes where the multi-agent has to be set-up are modeled by using the data model and the dataflow model; secondly, the agents are introduced in the processes with the dataflow model and in cooperation with the process actors (fig. 1); and finally, the data exchanges and the working mechanism of the multi-agent system are modeled with the processing model.

2 For example, the time for dealing with one procedure involving about 15 actors was halved, by improving cooperation and increasing the responsibilities of the actors.
Fig. 1. Example of integration of software agents into a cooperative processes

Figure 1 presents the integration of software agents in the information retrieval process of a technological watch team. Each agent is linked to an actor. Agents search information, filter them, compare them and transmit them to actors, which check them to record them into a database. The integration of agents in the process has been done in cooperation with the actors by using the dataflow model and corresponds to the second step of our method.

That is why our MAS are designed in two steps: a step to design the roles that the agents play; and a step to design the cooperative working of the agents and their interactions with human actors of the process in which they are integrated.

Design of a MAS into a human organization

Although the definition of our MAS structure has been facilitated by the use of holonic principles, the modeling of the system organization and the characterization of the functionality of the agents remain problematic. Indeed, the research published on this subject is mainly theoretical. Only a few recent research projects allow organization modeling with an application goal, but they are mainly devoted to the field of collective robotics [11] [12].

Here, we propose modeling and specification in two stages: the first stage concerns the individual functioning of each type of holonic agent; the second concerns the functioning of the group, describing communications between agents and between actors and agents.

5/18  DRAFT
Individual design of the holonic agents

In order to describe the general characteristics of various types of agent, we use a grid adapted from Ferber [13]. This grid gives a description in three dimensions, instead of the five dimensions initially suggested (cf. table 1). The physical dimension, which is too dependent upon the system, does not appear in the description of general characteristics. The relational dimension is attached to the social dimension. Concerning the organization functions, the conative and organizational functions, dealing with planning, have been grouped together; the conative function being more oriented towards needs, desires and urges which our holonic agents do not have, at least for the time being.

This grid enables us to define the functions for each holonic agent relating to: knowledge (the representational function also describes the non-procedural knowledge); action planning (the organizational function); interactions; the maintenance (the preservation function) and the actions specific to the role of the agent (the productive function). These functions are described in relation to the agent’s environment, the other agents and the agent itself.

Table 1. Design grid adapted from Ferber’s analysis grid [13]

<table>
<thead>
<tr>
<th>Dimensions \ Function</th>
<th>Social</th>
<th>Environmental</th>
<th>Personal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representational</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planning of the group (of the other roles)</td>
<td>Representation of the world</td>
<td>Representation of itself, of its capacities</td>
<td></td>
</tr>
<tr>
<td>Organizational</td>
<td>Planning of social actions, communications</td>
<td>Planning of actions in the environment</td>
<td>Planning control, meta-planning</td>
</tr>
<tr>
<td>Interactive</td>
<td>Description agent-society interaction, performative</td>
<td>Perception and action mechanisms in relation to the environment</td>
<td>Auto-communication, Auto-action</td>
</tr>
<tr>
<td>Productive</td>
<td>Management, coordination and negotiation tasks</td>
<td>Analysis, modification and creation tasks</td>
<td>Auto-modification, learning</td>
</tr>
<tr>
<td>Preservation</td>
<td>Preservation of the society, the relations, the network of contacts</td>
<td>Preservation of resources, defense and maintenance of territory</td>
<td>Self-preservation, repair, maintenance</td>
</tr>
</tbody>
</table>

This grid is applied to design the different types (different roles) of agents. However, it is interesting to note that a MAS can be considered as one single holonic agent. So it is possible to use this grid to define, at a higher abstraction level, the different functions of the multi-agent system.

But, even though this grid enables us to have a clear view of the agents’ actions according to the environment and according to the other agents, it does not allow a definition of the functioning of the whole organization. Indeed, it does not allow the design of the cooperative functioning of the whole multi-agent organization. So, it is necessary to use a method, such as AMOMCASYS, which allows us to do it.
Cooperative working of the holonic agents

Regarding the design of the MAS to be integrated to the human process, AMOMCASYS data model allows us to represent the principal Holon class (which describes the general agent structure) as well as the classes associated to the knowledge on the environment (representation of the process, the actor, the workstation, the responsible, the subordinates). Each holonic agent has five main functions: to plan its action according to the process and its current state (corresponding to the organizational function); to receive and send to other holonic agents (corresponding to the interaction function); to act (corresponding to the productive function, to the specialty of the agent) and to manage the links between the responsible and the subordinates (corresponding to the preservation function). Of course each holonic agent has an implicit function: ’initialize’ (enabling it to acquire knowledge upon the MAS).

The four main functions (the organizational, interactive, productive and conservative functions) imply co-operations between holonic agents and sometimes between the agents and the actors (the users). The processing model of the AMOMCASYS method can model these co-operations, as we will see in the following case study.

Bringing an Information Multi-Agent System in a technological watch department

The case study that is presented in this article has been led in a technological watch department of a large company. In this application, we have designed a MAS in order to assist actors of a technological watch department [14] in their tasks. This specification has been done following the analysis and the modeling of the department processes. In these processes, actors (called watchmen) have to collect information on Internet, to manage them and to distribute them to their clients. So we had to design a MAS for the information retrieval.

Structure of the information multi-agent system

An information multi-agents system (IMAS) is generally composed of information agents that search, on the basis of requests that are sent to them (directly or indirectly through of a database) information on databases (local or distributed) or on Internet sites.

Information agents' activities are often coordinated through coordinator agents. These agents own knowledge on information agents (such as their addresses, their
search domains for example) to which they send requests (in a targeted way if they own knowledge on their competences or by broadcast techniques). Coordinator agents have to gather collected information, in order to check it, compare it or filter it.

Most of information multi-agents systems are directly in touch with the user, upstream (to receive new requests) and/or downstream (to display search results). In order to have a reactive interface and distributed to the users, some IMAS propose the use of interface agents acting as interfaces between the users and the system.

In our IMAS, called CIASCOTEWA (meaning CO-operative Information Agents' System for the COoperative TEchnological WAtc), each agent proposed in the second step of our method (figure 1) is in fact a IMAS composed of agents.

So, we associate an information agents system (a CIASTEWA, for CO-operative Information Agents' System for the TEchnological WAtc) with each actor of the watch team. This allows the global system to have larger flexibility and allows each watchman to have larger autonomy. Indeed, it is easier to integrate a new watchman by adding a sub-system to the global system than by reconfiguring a centralized system.

A CIASTEWA is a sub holonic multi agent system, which has to search information, to collect it, to sort it and to communicate relevant information to others CIASTEWA. The architecture of a CIASTEWA is shown in figure 2.

Each of this sub system is made up of:

- a local database that contains the user requests, their results and information on the user,
- an interface agent that assists the users to express their requests and allows them the interaction with the results provided by information agents, or with the other users of the group,
- a coordinator agent that has the task to coordinate actions of the other agents,
- an information responsible agent, which distributes the request that are recorded in the local database to the request agents according to a search strategy,
- request agents that distribute the request whose they are responsible to search engine agents
- search engine agents that have to find information on the Internet.

So, each CIASCOTEWA helps the user, to who it is dedicated, to search relevant information and to communicate it with other actors. In order to maintain or create the feelings of community or group among the actors, which is often forgotten with the use of new technologies (the individuals are isolated with their workstation), we have proposed to develop self-organizing capacities in order to generate communities of CIASTEWA, which have to answer at the same kinds of requests. This reorganization is indicated to users in order to encourage them to cooperate, if they want to do it, with other users having the same centers of interests.
In fact, works on the generation or identification of communities in IMAS are appeared recently, like in [15] where the agentification of Web server needs the creation of agents’ communities. And, we think that “to find the right person” who could know where are located the answers is the best way to find the good information. Some works are carried out in this direction by [16] and [17]. For example, in a large laboratory, it is frequent that searchers have momentarily the same center of interest without knowing it. In our system, they are informed of this and so encouraged to exchange their information.

The cooperation between the several agents of a CIASTEWA is organized around a user database that contains data about the user, the user requests and the corresponding results.

A CIASTEWA is also linked to a group database (fig. 2). This one collects requests of the group that are qualified as public by the users, and temporally results of these requests during the comparison step (fig. 1). This knowledge about the team allows coherence in the global MAS and allows a cooperative watch by the watchmen.

As example of cooperation accorded by the system, we have the fact that each watchman knows what the others have collected; this avoids redundancy in the information management and storing. Of course, the redundancies of information stored could be suppressed by the use of a centralized system (like a proxy server or a centralized agent) but such a centralized system is quite less flexible than a distributed system concerning the add or suppression of an actor from the retrieving information system. Another example: when a user adds a new request, this one should be
compared to the others, if it is a subset of an existing one, the results are extracted by its results and the watchman is informed by its MAS that he has analog request to another actor (we suppose that this should make cooperation easier between watchmen having the same centers of interest).

But, the cooperative functioning of the different agents of the CIASCOTEGA can only be specified after the definition of the different roles composing a CIASTEW.

**Individual design of the CIASTEW agents**

Five roles have to be designed in a CIASCOTEGA: the interface agent role \( \phi_1 \), the coordinator agent role \( \phi_2 \), the information responsible agent role \( \phi_3 \), the request agent role \( \phi_4 \), and the search engine agent role \( \phi_5 \).

The table 3 presents the behavior of the coordinator agent. The roles of the other types of agents are described by the same way.

**Table 2. Definition of a coordinator agent**

<table>
<thead>
<tr>
<th>Dimensions/Function</th>
<th>Social</th>
<th>Environmental</th>
<th>Personal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representational</td>
<td>Knows the ( \phi_2 ), ( \phi_3 ), and all the others ( \phi_4 )</td>
<td>Knows the requests and results of the user, the requests and the results of the group.</td>
<td>Knows its name, its IP address.</td>
</tr>
<tr>
<td>Organizational</td>
<td>Watches over the coordination of actions of the group of agents of the CIASTEW.</td>
<td>Manages the databases of the user and the group.</td>
<td>Checks that the request does not exist</td>
</tr>
<tr>
<td>Interactive</td>
<td>Is the responsible of ( \phi_2 ) and ( \phi_3 )</td>
<td>Interacts with the others ( \phi_4 )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>Productive</td>
<td>Fills the group database with results and requests provided by the others ( \phi_4 )</td>
<td>Sends requests and results to the ( \phi_2 )</td>
<td>Modifications of knowledge, of its role.</td>
</tr>
<tr>
<td>Preservation</td>
<td>Checks if its contacts ( \phi_2, \phi_3 ) are active...</td>
<td>Checks its contacts ( \phi_4 ) are active, check the databases.</td>
<td>Deferred to its contacts.</td>
</tr>
</tbody>
</table>

**Cooperative functioning of a CIASTEW**

After having described the individual roles of agents that compose a CIASTEW, we have to define the cooperative interactions between them. An extract of the processing model representing a cooperative activity within a CIASTEW is shown on figure 4 of annex 1.
This figure presents the recording of a new request in a CIASTEWA. The user adds a request to its CIASTEWA through the interface agent. This one asks to the coordinator agent if the request exists in the group database, if it is a subset of another one or if it includes requests of other actors. In all these cases, a message is displayed to the user. The request is then recorded in the user database and a message is sent to the information responsible agent. This message asks it to execute the requests not yet carry out. For this, the agent creates a request agent for each request. Each request agent creates a search engine agent for each search engine specified in the request. Each of these request agents connects to Internet in order to find results and send them to its responsible. When the request agent has received the results of each of its subordinated, it filters them (it deletes the doubles) and sends the results to its responsible. When the information responsible agent has received response from each of the request agents, it compares results with the group results, given by the coordinator agents, and writes, in the results characteristics, which actor has also received the result.

Thanks to the AMOMCASYS method, we have defined other interactions between users and agents of the CIASCOTEWA such as the mailing, the annotation, the deletion of a result, the modifying and the deletion of a request.

3.2.4 Application of the CIASCOTEWA
A prototype has been built from these specifications and is currently used within our laboratory.

Firstly, we have to define the structure of a CIASTEWA by creating the agents and defining theirs links (theirs acquaintances). Then, as each CIASTEWA is composed of five kinds of agent, we have to implements five behaviors. Each behavior contains attributes relative to the knowledge of the agent to which it is associated and functions that are can be called by the acquaintances of the agent.

Thanks to the MAGIQUE platform [18], which is a java library that is dedicated to the design of hierarchical multi-agent system, we have setup a multi-agent prototype. Indeed, MAGIQUE is dedicated to Hierarchic Multi-Agent System where agents own competencies that they can learn or loose at run-time. By default, all the MAGIQUE agents contain a communication skill, which allow them to communicate with the other agents of their hierarchies.

To define a CIASTEWA with MAGIQUE, we created five empty agents to which we added skills (sets of behaviors): SupSkill for the coordinator agent; AgentFrameSkill for the interface agent; LaunchSearchSkill for the information responsible agent; SearchSkill for the request agents; OneEngineSearchSkill for the search engine agents. When an information responsible agent creates request agents, it creates empty agents and tells them that it is their boss. Then it asks to them to learn their competency (SearchSkill) and to perform their tasks. We have the same working when a request agent creates search engine agents. These search engine agents could create other sub agents if necessary (to interrogate different addresses of the same
search engine for example) without having impacts on the working of the global system. Indeed, if the search engine agents return the required data, it is not important that it does it alone or with the help of sub agents. This principle is applicable for all the agents as it is specified by the holonic model.

Figure 3 presents a screen copy of a CIASTEWA.

![Screen copy of a CIASTEWA](image)

**Fig. 3.** Screen copy of a CIASTEWA.

In our prototype, each CIASTEWA (fig. 3) allows to an actor to: consult results of a request; modify, delete, add a request; send a result to other actors of its neighborhood; add notes to a result; delete a result. The user has the following information about a result: its address; its page name; its summary; its size; its date; its owner; the names of actors who own it (for that, the CIASTEWA communicate with the others CIASTEWA of the group); the requests of the user that are linked to the result; and the search engines that have given the result.

A Magique agent can communicate with other one only if they are members of the same hierarchy. In order to make possible the communication between CIASTEWA, which are linked each to an actor of the technological watch cells, and so to create a CIASCOTEEWA, it has been necessary to create a "super agent" that is the responsible of all the coordinator agents. This agent has the BossSkill skill that contains no
function (the interactive function and the knowledge all the agents of the CIASCOTEWA system are implicit in a Magique agent).

At the present time, the subgroups of CIASTEWA are created by the users through a configuration file that defines, among other things, coordinator agent's acquaintances of a CIASTEWA.

Indeed, a set of xml files is necessary to initialize a CIASTEWA. Each of them corresponds to a knowledge: agent-config.xml corresponds to the personal knowledge; actors.xml, wastebasket.xml, search-engines.xml correspond to the environment knowledge; requests.xml and results.xml correspond to the user database (the agent-config.xml file is given in annex).

We have added some capacities to the BossSkill skill that should allow the system to be able to self-organize itself according to the interest centers of the user, by using Kohonen self-organizing maps and interest distances, but we do not have yet examined this functionality results. This reorganization of the system aims at decreasing number of communications between CIASTEWAs and aims at increasing the feeling of team among the user by informing them that they have interest centers in common (we think that this should increase cooperation between them).

4 Conclusion

In order to specify a co-operative information agents’ system into a human organization, we have defined a method composed of three steps: a step of analysis and modeling of the human organization; a step of modeling the insertion of agent systems into the human organization; and a step of design of the multi-agent system.

Our work uses the AMOMCASYS method that we have defined to analyze and model complex administrative system, and is underlies by holonic model. Indeed, this model allowed us to understand human organizations and we have showed (Adam, 2000) that this model allows the design of multi-agent system particularly adapted to the studied human organizations.

Actually, the first prototype of the CIASCOTEWA system that we have proposed has been setup in short term (few months), and has been particularly well accepted by actors, thanks to the participative design that we propose.

At this time, the CIASCOTEWA is used in our laboratory in order to develop capacities of self-organization. A centralized CIASCOTEWA, accessible by jsp pages is currently set-up in a knowledge management department of a large company.

We aim at develop a more automatic design of the agent, by using our last work, like the deployment of a holonic multi-agents system among a network from a xml configuration file and by use of the MAGIQUE platform.
References

ANNEX 1: AMOMCASYS’ processing model of the creation process of a new request
ANNEX 2: XML configuration file of a CIASTEWA

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE agent-config SYSTEM "agent-config.dtd">
<agent-config>

<local-host>
    <user name="Adam" firstname="Emmanuel"/>
    <host name="pc-b127-2"/>
    <mail-host name="pulsar.univ-valenciennes.fr"/>
</local-host>

<super-boss ip-boss="192.168.6.27" port-boss="4444"/>

<knowledges>

<data>
    <actors path="data/actors.xml"/>
    <requests path="data/requests.xml"/>
    <results path="data/results.xml"/>
    <wastebasket path="data/wastebasket.xml"/>
    <searchengines path="data/searchengines.xml"/>
</data>

<display>
    <actors-ss chemin="style/actors.xsl"/>
    <wastebasket-ss chemin="style/wastebasket.xsl"/>
    <searchengines-ss chemin="style/searchengines.xsl"/>
    <requests-ss chemin="style/requests.xsl"/>
    <results-ss chemin="style/results.xsl"/>
</display>

</knowledges>

</agent-config>