A generic agent-oriented architecture

Sara Maalal¹, Dr. Malika Addou²

¹Hassan II University, Higher National School of Electricity and Mechanics ENSEM, Architecture System Team – LISER, Eljadida Road, BP 8118 Oasis, Casablanca, Morocco

²Hassan II University, Hassania School of Public Works EHTP, Architecture System Team – LISER, Eljadida Road, BP 8108 Oasis, Casablanca, Morocco

ABSTRACT

Following the appearance of several agent architectures proposed by designers during the processing of their SMA specific to their areas of research, we have decided to introduce a generic agent architecture composed of layers of agents and other optional components such as databases, web services, or other files. This generic architecture designed to better adapt existing and proposed architectures for multi-agent systems studied in other researches. The architecture proposed in this paper is based on a number of researches highlighting architectural data for specific multi-agent systems [1], [2], [3] etc. Indeed, it offers different agents types (presentation agent, business agent, data access agent...). The proposed architecture, considered as generic, is suitable for a large number of multi-agent systems. It can be configured according to the studied multi-agent system’s structure and to agents as well as the type of collaboration to develop. It has an open system in which we can remove or add new features that will be provided obviously by the agents.

Keywords: Multi-Agent Systems (MAS), Model, Meta-model, Architecture.

1. INTRODUCTION

Multi-agent systems’ structure and contents are continuously evolving. Information appear and disappear and new concepts and components are created and changing very quickly. Multi-agent systems’ architecture must be able to follow this development and must be extensible, adaptive and flexible to use information derived from MAS. As its name suggests, our proposed Agent architecture takes advantage of the Agent-oriented software engineering to ensure flexibility and reusability. The starting point of this architecture is a prototype already made for a collaborative information system [1]. This system has already adopted the agent approach and used a device to facilitate and optimize information and resources exchange and sharing among collaborative partners network in a supply chain. However, this prototype has concerned only one research area which is the design and development of collaborative information systems applied in logistics.

Our architecture uses computer applications’ architectural structure already known by its four layers (Presentation, Business, Data Access and Data Storage) by deploying it on a larger and more effective agent’s software organization, with more specialized agents in order to process simultaneously multiple research areas.

2. ARCHITECTURE DESCRIPTION

For our system design, we propose a four-level model, each level includes one or more agents (see Figure 1) that communicate and collaborate with each other to perform the tasks assigned to them. Depending on these agents’ behavior nature, that can be distributed, centralized or hybrid, we found three cases of architectures. Subsequently we present the model of each case of architecture with its four related layers « Presentation, Business, Data Access and Data Storage »:

2.1 Case of simple architecture

This architecture illustrates the distributed system behavior among several agents namely GUI, Business and Data and Services Access agents. In this case, the decision is not taken by a single specialized agent, but by different agents, each agent having a local activity and limited vision of the whole system.
The model for this architecture is presented as follows:

Figure 1 Generic Agent Architecture for a simple application (Distributed behavior)

- **Presentation Layer (shown in blue):** This layer contains the interfaces that allow users to query the system. Its role is to capture user’s requests in order to meet his needs as well as possible. It includes one or more independent reactive "GUI" agents where each one interacts with the user through a specific graphical interface to help him achieving a specific task. Each query captured by a reactive “GUI” agent is transferred later to the Business layer.

- **Business Layer (shown in red):** This layer mediates between users and the system. It is composed of one or more independent cognitive “Business” agents where each one is responsible to perform special processing of the request received from the Presentation layer. It will then call the Data and Services Access layer to load the resources necessary to its processing implementation. The utility of these cognitive “Business” agents is the fact that each of them has the ability to store all queries that he has processed and the processing he has already made. The storage of these requests and their processing is necessary to avoid any possible redundancies at his level.

- **Data and Services Access Layer (shown in green):** This layer provides simple interfaces for data access and handling. It is composed of one or more independent cognitive "Data and Services Access" agents where each one receives the action plan to be implemented by the "Business" reactive agent corresponding to the Business layer and loads different storage resources to perform this action plan. It also establishes and implements methods to create, read, update and delete data entities stored in the Data Storage layer. Similarly for the cognitive “Business” agent, "Data and services access” agent has the ability to store all action plans received from the cognitive "Business” agent of the Business layer and all methods access it has established and the resources that he appealed to avoid, at his level, any redundant returning of a resource loading or action plans execution request.

- **Data Storage Layer (framed in blue):** This layer contains the elements (files, databases and services repository) necessary to perform the action plan defined by the Business layer.

Note that each of the "Business" and "Data and Services Access" cognitive agents of the Business and Data Access and Services layers can be rational, adaptive or intentional according to their roles in the multi agents system being studied.

2.2 Case of complex architecture

This architecture illustrates centralized system behavior among some agents namely GUI, Business and Data and Services Access agents. Here, the system is under the control and the decision of some agents having access to different parts of the system.
The model for this architecture is presented as follows:

![Architecture Diagram]

**Figure 2** Generic Agent Architecture for a complex application (Centralized behavior)

- **Presentation Layer (shown in blue):** This layer contains the interfaces that allow users to query the system. Its role is to capture user’s requests in order to meet his needs as well as possible. It includes one or more reactive "GUI" agents where each one interacts with the user through a specific graphical interface to help him achieving a specific task. Each query captured by a reactive “GUI” agent is later transferred to the Business layer. This interaction results in a transformation of all user’s queries, collected via the various GUIs of this layer, to an objective understood by the system that will facilitate the Business layer functioning. Indeed, these queries are grouped by "GUI" agent as a global query transferred to the Business layer.

  Note that this layer may contain a main GUI managed by the "GUI" agent that appeals to many GUls when each is used to capture a user request. Or, it may have several independent GUls when each is managed by "GUI" agents. Communication and transfer of messages between the different reactive "GUI" agents are illustrated by double arrows at Figure 2. The "GUI" agent is also responsible for displaying to the user the processing result of this global request received from the “Business” agent of Business layer. This requests and responses’ transfer between the two Presentation and Business layers is illustrated by a double arrow at Figure 2.

- **Business Layer (shown in red):** This layer mediates between users and the system. It is composed of a cognitive agent the "Business" agent who decomposes queries into sub-queries and collects the results found in the form of a comprehensive response and returns them to the Presentation layer. This agent is also responsible for developing action plans composed of a series of activities, in order to achieve the target he has set for the agents it manages which are generally reactive. Indeed, each action plan is executed by a reactive agent starting after receiving the processing series of the sub request from the "Business" agent. It will then call the "Data and Services Access" layer to load the resources necessary for its action plan implementation.

  The utility of the cognitive "Business” agent is that it has the ability to choose the suitable reactive "Business” agent for each processing and the ability to memorize all the requests it has processed and divided into sub-queries as well as action plans it has established for its reactive agents with whom he communicates. These processing’ storage is necessary to avoid any possible redundancies at its level.

  Communication and transfer of messages between the cognitive "Business” agent and its associated reactive agents are illustrated by a double arrow at Figure 2.

- **Data and Services Access Layer (shown in green):** This layer provides simple interfaces data access and handling. It is composed of a cognitive "Data and Services Access” agent that receives the action plan to be implemented by the various "Business” reactive agents of the Business layer and appeals to different storage resources to achieve these action plans. It also provides methods to create, read, update and delete data entities stored at the Data Storage layer.

  "Data and services access” agent is related to one or more reactive agents that load the requested resources and execute the pre-established access methods. Similarly for the cognitive "Business” agent, "Data and services access” agent has the ability to store all action plans received from the reactive "Business” agents of the Business layer and all methods access it has established and the resources that he appealed to avoid, at his level, any redundant returning of a resource loading or action plans execution request.
- **Data Storage Layer (framed in blue):** This layer contains the elements (files, databases and services repository) necessary to perform the action plan defined by the Business layer. Like simple architecture, each of the "Business" and "Data and Services Access" cognitive agents of the Business and Data Access and Services layers for this architecture can be rational, adaptive or intentional according to their roles in the system multi agents being studied.

### 2.3 Case of hybrid architecture

This architecture provides a system that can be shared between distributed and centralized behaviors among several agents namely GUI, Business and Data and Services Access agents. Indeed, we can find for example a distributed behavior of the Presentation layer’s agents and another centralized behavior of the Business and Data Access and Services layers’ agents. So, we will have the following model corresponding to this architecture:

![Figure 3 Hybrid Agent Architecture with distributed GUI agents’ behavior and centralized Business and Data and Services Access agents’ behavior](image)

Or, we can find a centralized agents’ behavior of Presentation and Data and Services Access layers and another distributed agents’ behavior of the Business layer. So we will have the following model corresponding to this architecture:

![Figure 4 Hybrid Agent Architecture with distributed GUI agents’ and Data and Services Access agents’ behavior and centralized Business agents’ behavior](image)

We can find a third example of hybrid architecture where there is a centralized agents’ behavior of Presentation and Business layers and another distributed agents’ behavior of the Data and Services Access layer. The model for this
architecture is the following:

![Figure 5 Hybrid Agent Architecture with distributed GUI agents’ and Data and Services Access agents’ behavior and centralized Business agents’ behavior](image)

### 3. APPLICATION EXAMPLES

This chapter is dedicated to MAS design phase where we present the architecture of three applications: Chat Application chat, Football Team and Collaborative Information System Management. This will help us to test and validate our proposed architecture and consider it as generic for any type of multi-agent system.

#### 3.1 Application Example 1: Chat Application

Our proposed architecture has been used to design a multi-agent system of a chat application [4] which consists of three reactive agents playing the roles of chatters. These agents are considered reactive since each of them does not respond before the application user declares the recipient’s name. Thus, they have a stimulus /response behavior. Therefore, an agent will be prepared to capture the name and message and send it to another suitable agent. It will also react to clear all sent or received messages from its interface.

The designer can build the architecture of its multi-agents system from our generic agent architecture (see section 2). For this, he must choose one of three proposed architectures cases, rely on the chosen architecture initial structure with its four layers "Presentation, Business, Data Access and Data Storage" and project them on his system.

In this first example, our chat application will have a single graphical interface called and used by three reactive agents where each of them will capture the recipient agent name entered by the user to send a message. Therefore, the only possible processing in this application is the verification of the recipient’s name of the message to send. This verification is performed through the consultation of a table or a file containing the chat application user names. We can consider our chat application as a simple application with a distributed behavior and thus its architecture will correspond to the first simple case proposed at the 2.1 section of our architecture Generic Agent and whose four layers were used and adapted to our example. So we get a valid architecture for our chat application illustrated by the following Figure 6:

![Figure 6 Architecture of a chat application](image)
As shown in the figure above, each of the Presentation layer’s reactive chat agents has a graphical interface that lets capture the recipient's name and the message to send. This data is then sent to the “Business” agent, corresponding to Business layer, who initiates the verification of the recipient's name captured by the user. For this, he uses the “Data Access” agent of the Data Access layer that loads the requested table or data file from the Data Storage layer in order to let “Business” agent perform its verification. Finally, the “GUI” agent displays the message in the recipient agent’s GUI.

3.2 Application Example 2: Football Team
Our proposed architecture has been used to design a multi-agent system representing a football team (see Figure 7), whose simulation was studied by El Bejjet [5].

Our football team includes nine agents listed below:

- **Six cognitive agents**: Who play the role of players, these players are cognitive by looking to have enough information on their environment (here the land) and to act according to different situations. So each player is able to see the ball and the other players, to know his adversaries and to react while taking in consideration all these information. Also, as in a football team, here the coordination and the communication are indispensable for a good progress of all operations done by these agents.
- **Two reactive agents**: These agents will be the goal keepers, the interest that they are reactive agents relies on the fact that a goal keeper doesn’t react before the ball is close to the goal. Therefore an agent that plays a referee’s role will react (to get ready to catch the ball) in the case where a ball will enter in a given area.
- **One reactive agent**: Who plays the role of the referee who supervises the behaviors of the two teams other agents in order to not overpass the land’s sides. He is a reactive agent because he only reacts on the event of an agent exiting out of the land.

Our football team is a complex application with a centralized behavior in the three layers “Presentation, Business and Data Access” and therefore, its architecture corresponds to the second complex case proposed at the section 2.2 of our Generic Agent Architecture whose four layers were used and adapted to our example. So we get valid architecture for our multi-agent system simulating a football team as we illustrate in Figure 8 below:

As shown in the figure, our application’s Presentation layer has a main GUI managed by the main “GUI” reactive agent. This main GUI is used for inserting stadium’s data and football team’s members with their number and type. It then uses nine graphical interfaces offered by nine reactive “GUI” agents. Each GUI is useful to inform the player,
goalkeeper or referee data namely their roles, names, sizes, weights, perceptions...
Then, this data is sent to the main “Business” cognitive agent of the Business layer. This agent is responsible for establishing the link between the nine agents (six cognitive representing the game players and three reactive representing the two goalkeepers and the referee) and transferring data captured at GUIs level of the Presentation layer. In order to help achieving the objectives of “Manager” agents that it manages, the main “Manager” agent uses the “Data Access” agent of Data Access layer that loads the requested data table or data file from the Data Storage layer, so that the agent can perform its role. It also loads the web service ensuring the simulation of each “Business” agent which transfers the football team general simulation result to the Presentation layer’s “GUI” agent.

3.3 Application Example 3: Collaborative Information System Management
In this third example, we have used our generic architecture for the design of a collaborative platform (see Figure 9) based on a supply chain that ensures cooperation, information backup and sharing between supply chain’s actors in real time. The main objective of this platform is to facilitate and optimize information and resources exchange and sharing among the collaborative partners network. This collaborative platform has been studied and designed by Dachry [6]:

We detail below our system’s agents components:

- **One reactive GUI agent:** This agent can be seen as a wizard that allows users to interact with the system. It allows acquiring all user requests, transforming these requests in understood goals by the system, communicating them to the “Manager” agent and displaying the results to users. It also checks users’ identification (authentication) and sends their profiles to the “Manager” agent [7].

  This agent is considered as a reactive agent due to its stimulus action mode after receiving users’ requests and “Manager” agent’s responses.

- **One cognitive Manager agent:** This agent is considered as the system mediator, the agent process the target set by the “GUI” agent and is able to reformulate this goal into subqueries that are transferred to the MAS planning (set of “Composer” agents). When the subqueries’ execution is completed, the “Manager” agent collects the results found in the form of a global response [7].

  This agent is considered as a cognitive agent due to its ability to choose the appropriate “Composer” agent and to its capacity to remember all requests that he processed and all action plans that he established for his “Composer” agent he communicates with.

  Since this agent is cognitive, it will therefore have a knowledge base represented by the agent’s possible representations section. In our example, we find representations relating to “Composer” agents it manages i.e. their identifiers, names, and availabilities. The second representations that this agent may have are those related to the action plans it has established for its “Composer” agents i.e. action plans identifiers, ordered articles types and finally processed requests identifiers.

- **Five reactive Composer agents:** where each is attached to a specific type of supply chain partners [8]. We offer a producer composer agent, a customer composer agent, a supplier composer agent, a agent composer agent and subcontractor composer agent.

  For each sub request sent by the “Manager” agent, one of the composer agent will perform a series of processing and proposes a solution plan for the fixed goal [7]. It communicates also with internal resources (databases / services base) and external resources (databases / remote services) to execute the established action plan.

![Figure 9 Concept of collaborative information system](image-url)
This agent is considered as a reactive agent due to its stimulus action mode after receiving “Manager” agent’ requests and this by calling the appropriate service and by collecting “Service” agent’s responses.

- **One cognitive Service agent:** This agent loads services necessary to perform the action plan produced by one of the “Composer” agent.

This agent is considered as a cognitive agent due to its capacity to remember all action plans that he received from a “Composer” agent and its ability to decide of the appropriate service to perform this action plan [7]. It will therefore have a knowledge base represented by the agent’s possible representations section. In our example, we find representations regarding the services it loaded for the corresponding “Composer” agent so that it can perform its action plan. These representations define these services identifiers, names, and availabilities. The second representations that this agent may have are those related to the action plans received by “Composer” agents i.e. action plans identifiers, ordered articles types and processed requests identifiers and in order to quickly detect action plans and queries for which he has already loaded services for processing.

The projection of our generic agent architecture’s four layers ”Presentation, Business, Data and Services Access and Data Storage” (see section 2) on our multi-agent system simulating a collaborative system between actors in the supply chain gives us the valid architecture for our application illustrated in Figure 10 below:

![Figure 10 Architecture of a collaborative information system](image)

As noted in the previous figure, our collaborative system is characterized by a centralized behavior of its agents in the three ”Presentation, Business and Data Access” layers and therefore its architecture corresponds to the second complex case proposed at Section 2.2 of our generic agent architecture.

Indeed, the Figure 10 indicates that our application’s Presentation layer has a main GUI managed by the reactive “GUI” agent. This main GUI uses two graphics interfaces provided by two reactive “GUI” agents. The first GUI is used to verify the authentication of users wanting to make orders and the second GUI is used to specify the order details and choose the delivery address.

The main “GUI” agent also handles displaying to the user the result of the processing performed by the Business layer’s “Manager” agent. Indeed, it transmits the data captured by the “Order GUI” agent of the Presentation layer in a request form to the cognitive “Manager” agent of the Business layer. It processes the request by dividing it into sub-queries and sends them after to the five reactive “Composer” agents (producer, customer, supplier, subcontractor, and carrier) depending on their tasks’ nature.

The “Manager” agent is also responsible to develop action plans for “Composer” agents it manages. Indeed, each action plan is implemented by a reactive “Composer” agent that starts after receiving the sub-query’s processing series from the “Manager” agent. It will then call the ”Data and Services Access” layer to load the resources required for its action plan implementation.

Indeed, each “Composer” agent will be the order processing’s responsible by calling the appropriate services. For this, it calls the main “Data Access” agent of the Data Access layer that manages several reactive “Database (BD)” agents or
cognitive “Services” agents. These agents are respectively responsible to access to the requested table or web service from the Data Storage layer so that the “Composer” agent can execute its action plan. Finally, he transfers his processing result to the “Manager” agent of the Business layer that brings all the five “Composer” agents processing results in the form of a comprehensive response and returns them to the Presentation layer’s main “GUI” agent.

4. CONCLUSION

In this paper, we presented our architecture following the agent-based paradigm. This architecture has as main mission the projection of the various models provided on the different layers of agents belonging to a multi-agent system and the generalization of all architectures implemented for MAS that had previously been studied. Indeed, since there is no optimal agents’ architecture which precisely depends on the considered application domain and system requirements, we have tried to give a general architecture that can be simple, complex or hybrid depending on the agent behavior nature (distributed, centralized or both) at each Presentation, Business and Data and Services Access layer. Our proposed architecture tries to reconcile and factorize cognitive and reactive architectures simultaneously. It has the advantage of being easy to adjust by the designer in order to develop an efficient architecture specific to his MAS according to its agent’s behavior.

References


AUTHORS

Sara Maalal was born in Rabat the Morocco’s capital in 1985. She received his professional master in Computer Engineering and Internet (3I), Security Networks and Systems Option, in 2008 from the Faculty of science of HASSAN II University, Casablanca, Morocco. In 2010 she joined the system architecture team of the National and High School of Electricity and Mechanic, Casablanca, Morocco. As a software engineer, her actual main research interests concern Designing and modeling Multi-Agent Systems.

Malika Addou received her Ph.D. in Artificial Intelligence from University of Liege, Liege, Belgium, in 1992. She got her engineer degree in Computer Systems from the Mohammadia School of Engineers, Rabat, Morocco in 1982. She is Professor of Computer Science at the Hassania School of Public Works, Casablanca, since 1982. Her research focuses on Software Engineering (methods and technologies for design and development), on Information Systems (Distributed Systems) and on Artificial Intelligence (especially Multi-Agent Systems technologies).