# Evaluating Interaction of MAS Providing Context-Aware Services

Nayat Sanchez-Pi, David Griol, Javier Carbo, and Jose M. Molina

Carlos III University of Madrid,
Avda de la Universidad Carlos III, 22. Colmenarejo. Madrid
{nayat.sanchez,david.griol,javier.carbo,
josemanuel.molina}@uc3m.es

**Abstract.** Much effort has been spent on suggesting and implementing new ar-chitectures of MAS to specific domains. Often each new architecture is not even compared to any existing architectures in order to evaluate their potential ben-efits. The evaluation of Multi-Agent Systems (MAS) is a complex problem and it does not have a single form. The present work follows the research line of considering the agent interaction as the main evaluation criteria, the most impor-tant characteristic of any complex software as autonomous agents according to [9]. So, in this paper, we have suggested an assignment of evaluation values to Agents interaction in an specific MAS architecture for providing context services by means of conversational agents. This evaluation is mainly based on the rele-vance of the messages content brought by an interaction. For dependant nature of the relevance of the messages, the valuation has to be adhoc, but our paper provides an example of how interesting is this alternative in order to evaluate any MAS architecture theoretically.

**Keywords:** multi-agent systems, evaluation.

## 1 Introduction

The evaluation of MAS makes possible two main objectives. Firstly, to understand their behavior and secondly, to compare the operation of several systems. In the literature there are works addressing this evaluation based on the architectural style [1]; software engineering related criteria and characteristics of MAS [7]; [4], or the complexity of interactions [6]. The evaluation of the distributed nature of MAS and the complexity of the interaction inside them is a very difficult task. The consideration of the interaction as the most important characteristic of MAS [9], allows studying and comparing this kind of systems at the level of their interactions. In this paper, we suggest an assignment of evaluation values to Agents interaction in an specific MAS architecture for providing context services. This evaluation is mainly based on the relevance of the messages content brought by an interaction. For dependant nature of the relevance of the messages, the valuation has to be adhoc, but our paper provides an example of how interesting is this alternative in order to evaluate any MAS architecture theoretically. The rest of the paper is structured as follows. Section 2 briefly describes our agent-based architecture to provide context-aware services. Section 3 details the evaluation method based on the weight of the information specified for each message. Section 4 presents the application of the proposed evaluation methodology to a specific scenario. Finally, we draw some conclusions and suggest some future directions of research.

## 2 Our Agent System to Provide Context-Aware Services

The proposed agent-based architecture manages context information to provide personalized services by means of users interactions with conversational agents. As it can be observed in Figure 1, it consists of five different types of agents that cooperate to provide an adapted service. *User agents* are configured into mobile devices or PDAs. *Provider Agents* are implemented by means of *Conversational Agents* that provide the specific services. A *Facilitator Agent* links the different positions to the providers and services defined in the system. A *Positioning Agent* communicates with the ARUBA positioning system [8] to extract and transmit positioning information to other agents in the system. Finally, a *Log Analyzer Agent* generates user profiles that are used by Conversational Agents to adapt their behaviour taking into account the preferences detected in the users' previous dialogues.

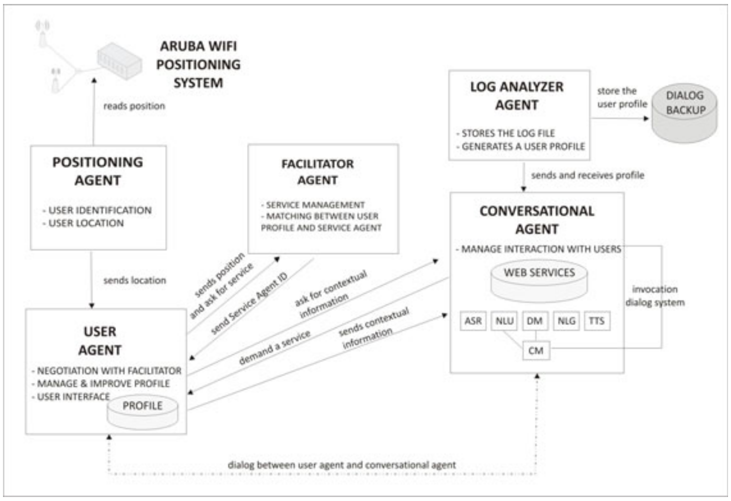
ARUBA WIFI POSITIONING SYSTEM
reads position
POSITIONING AGENT
- USER IDENTIFICATION
- USER LOCATION
sends location
USER AGENT
- NEGOTIATION WITH FACILITATOR
- MANAGE & IMPROVE PROFILE
- USER INTERFACE
PROFILE
FACILITATOR AGENT
- SERVICE MANAGEMENT
- MATCHING BETWEEN USER PROFILE AND SERVICE AGENT
sends position and ask for service
send Service Agent ID
ask for contextual information
demand a service
sends contextual information
LOG ANALYZER AGENT
- STORES THE LOG FILE
- GENERATES A USER PROFILE
store the user profile
DIALOG BACKUP
sends and receives profile
CONVERSATIONAL AGENT
- MANAGE INTERACTION WITH USERS
WEB SERVICES
ASR
NLU
DM
NLG
TTS
CM
invocation dialog system
dialog between user agent and conversational agent

**Fig. 1.** Schema of the proposed multi-agent architecture

A conversational agent is a software that accepts natural language as input and generates natural language as output, engaging in a conversation with the user. To successfully manage the interaction with the users, conversational agents usually carry out five main tasks: automatic speech recognition (ASR), natural language understanding (NLU), dialogue management (DM), natural language generation (NLG) and text-to-speech synthesis (TTS). These tasks are usually implemented in different modules. In our architecture, we incorporate a Context Manager in the architecture of the designed conversational agents, This module deals with loading the context information provided by the User and Positioning Agents, and communicates it to the different modules of the Conversational Agent during the interaction.

To manage context information we have defined a data structure called *user profile*. Context information in our user profile can be classified into three different groups. *General user information* stores user's name and machine identifier, gender, preferred language, pathologies or speech disorders, age, *Users Skill level* is estimated by taking into account variables like the number of previous sessions, dialogues and dialogue turns, their durations, time that was necessary to access a specific web service, the date of the last interaction with the system, etc. Using these measures a low, medium, normal, high or expert level is assigned. *Usage statistics and preferences* are automatically evaluated taking into account the set of services most required by the user during the previous dialogues, date and hour of the previous interactions and preferred output modality.

The interaction with the different agents follows a process which consists of the following phases:

1. The ARUBA positioning system is used to extract information about the positions of the different agents in the system. This way, it is possible to know the positions of the different User Agents and thus extract information about the Conversational Agents that are available in the current location.
2. The Positioning Agent reads the information about position (coordinates $x$ and $y$) and place (*Building* and *Floor*) provided by the ARUBA Positioning Agent by reading it from a file, or by processing manually introduced data.
3. The Positioning Agent communicates the position and place information to the User Agent.
4. Once a User Agent is aware of its own location, it communicates this information to the Facilitator Agent in order to find out the different services available in that location.
5. The Facilitator Agent informs the User Agent about the services available in this position .
6. The User Agent decides the services in which it is interested.
7. Once the User Agent has selected a specific service, it communicates its decision to the Facilitator Agent and queries it about the service providers that are available.
8. The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location.
9. The User Agent asks the Conversational Agent for the required service.
10. Given that the different services are provided by context-aware Conversational Agents, they ask the User Agent about the context information that would be useful for the dialogue. The User Agent is never forced to transmit its personal information and preferences. This is only a suggestion to customize the service provided by means of the Conversational Agent.
11. The User Agent provides the context information that has been required.
12. The conversational agent manages the dialogue providing an adapted service by means of the context information that it has received.
13. Once the interaction with the Conversational Agent has finished, the Conversational Agent reads the contents of the log file for the dialogue and send this information to the Log Analyzer Agent.

14. The Log Analyzer Agent stores this log file and generates a user profile to personalize future services. This profile is sent to the Conversational Agent.

In our architecture we need an objective description of the concepts and relationships of the domain of knowledge of the messages exchange by agents. This explicit and formal specification of a shared conceptualization is what we usually called ontology [5]. An ontology allows that a content of a message can be interpreted unambiguously and independently from the context.

Eight concepts have been defined for the ontology of the system. The definition is: *Location* (*XCoordinate* int, *YCoordinate* int), *Place* (*Building* int, *Floor* int), *Service* (*Name* String), *Product* (*Name* String, *Characteristics* List of Features), *Feature* (*Name* String, *Value* String), *Context* (*Name* String, *Characteristics* List of Features), *Profile* (*Name* String, *Characteristics* List of Features), *DialogLog* (*Log* List of Strings).

Our ontology also includes six predicates with the following arguments: *HasLocation* (*Place*, *Position*, and *AgentID*), *HasServices* (*Place*, *Position*, and *List of Services*), *isProvider* (Place, Position, AgentID, Service), *HasContext* (*What*, *Who*), *HasDialog* (*DialogLog* and *AgentID*), *HasProfile* (*Profile* and *AgentID*), and *Provide* (*Product* and *AgentID*).

The free software JADE (Java Agent Development Framework)[1] has been used for the implementation of our architecture. It was the most convenient option as it simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines and the configuration can be controlled via a remote GUI.

## 3 Evaluation Proposal Based on Agents Interaction

The consideration of interaction as the main of evaluation has been addressed by other researches. These studies have verified that this kind of evaluation originates different types of problems [6]. Firstly, the effect of an interaction unit (a single message) in an agent system could be equivalent to the definition of $n$ units (messages) in another system. This way, the weight assigned to the same interaction is *1* in the first system and it $n$ in the second. Secondly, the interaction units that are received and cannot be used by an agent could be a bias in the measurement of interaction in MAS. Our proposal is based on [6]. The first task is to classify the possible received messages into specific sets sharing the same type. Then, a weight is associated to a message according to its type. If two messages with the same type produce very different effects on the agent, then this assignment does not provide a correct solution. The effects of considering interactions as the main feature of the evaluation, consists of initially processing a message and then decide a responsive action. The initial processing is carried out in two phases, which consist of the memorization that deals with the change at the internal state caused by the received message, and the decision that concern the choice of the action that will be handled. According to the evaluation model, two kinds of functions are considered:

[1] http://jade.tilab.com/

– A function *Interaction* associates a weight to each message according to its type. This function can be computed adopting the primitives proposed by [3] to the type of interaction. This work describes four possibilities kinds of messages: present, request, answer, and inform. These four types have to be distinguished due to the different basic behaviors that they model from the sender or the receiver points of view. Therefore, if $M^A_{sent}$ : the set of messages sent by agent A and if $M^A_{received}$ : the set of messages which may be received by agent A, the function Interaction associates for a message sent by the agent A, a message received by the agent B:

$$Interaction = M^A_{sent} \rightarrow M^B_{received} \tag{1}$$

– This solution partially resolves the problem. However, it is only valid if two messages of the same type have equivalent effects on the agent. This way, we introduce the function $\Phi$ [6] to associate a different weight to a message according to the change on the internal state and the actions triggered after its reception. This function evaluates the different effect of a message in agent systems. For better understanding $\Phi$ is divided into two terms: the first one evaluates Decision, $DD^A$ and the second one evaluates Memorization, $MM^A$. The term $MM^A$ associates a value to the variation of the internal state (caused by memorization step). To quantify these two terms, some measurable characteristics of the internal state must be defined. The specification of these characteristics is related to each specific application domain. The variation on one of these characteristics implies the function $MM^A$ is considered as the sum of these weights:

$$MM^A = M^A_{received} \times S^A \rightarrow S^A \tag{2}$$

With regard to $DD^A$, this term associates a value to the triggered actions (i.e., results of decision step). To quantify this term, different types of actions must be defined and associate a weight to each of them. Then, the value of the function $DD^A$ is calculated as the sum of the weights assigned to the triggered actions. Let $S^A$ be the set of possible internal states for the agent A and let $A^A$ be the set of actions may be done by agent A, then:

$$DD^A = M^A_{received} \times S^A \rightarrow A^A \tag{3}$$

Finally, the function $\Phi$ is defined as the sum of these functions $DD^A$ and $MM^A$ and the evaluation of the interactions in the MAS is based on the combination of the two functions Interaction + $\Phi$.

## 4 Evaluation Our the Proposed MAS Architecture

In this section we present the application of the evaluation method described in the previous section o our context-aware agent architecture. We compute the described evaluation functions and assign different weights to each message in the agents interaction. In section 4.1 we define the Interaction function and in section 4.2, the $\Phi$ function is described.

## 4.1 Weights vs Type of Message: Function *Interaction*

The Interaction function according to the four message types described in the previous section (present, request, answer, and inform). Following, we detail the different phases during the interaction of the different agents and the different messages that are generated for the provision of the service. Three types of messages have to be distinguished because of the different basic behaviors that they model from the sender or the receiver points of view:

- A request includes a change of state of the sender, waiting for the answer.
- An inform includes no change of state for both the sender and the receiver. It might generate other informs, and possibly answers.
- A present includes a possible change in the state of the sender and/or of the receiver. Typically, a present will enable entering a society and introduce itself to other agents

Services are offered by means of a connection with the conversational agent.

1. The ARUBA positioning system notices a change in the position of a given User agent: *no message involved*
2. The Positioning Agent reads the information about position provided by the ARUBA Positioning Agent in the corresponding file: *no message involved*
3. The Positioning Agent communicates the information about position (coordinates x and y) and place (Building and Floor) to the User Agent: *present message*.
4. Once a User Agent knows its location, it asks the Facilitator Agent about the different services that are available in that location : request message.
5. The Facilitator Agent informs the User Agent about the available services: *inform message*
6. The User Agent then decides the services in which it is interested: *no message involved*
7. Once the User Agent has selected a specific service, it communicates its decision to the Facilitator Agent and ask it about the service providers that are available: *request message*
8. The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location: *inform message*
9. The User Agent asks the Conversational Agent for the required service: *request message*
10. The Conversational Agent asks the User Agent about the context information that would be used to adapt the provided service: request message
11. The User Agent provides the context information that has been required: *inform message*
12. The conversational agent manages the dialogue providing an adapted service by means of the context information that has been received: *no message involved*
13. Once the interaction with the Conversational Agent has finished, this agent sends a log file to the Log Analyzer Agent: *inform message*

14. The Log Analyzer Agent stores this log file and generates a new user profile to personalize future services. This profile is sent to the Conversational Agent: *inform message*

Therefore we associate the next values to the message types:

- request: 2 (a change of state, a reaction produced)
- inform: 1 (no change of state)
- present: 1.5 (1 or 2 change of state)

In order to assign weights to each type of messages, we follow this criteria: if there is any interest over a negotiation from the Users side, then the maximum weight is assigned to the Agree-proposal or Reject-proposal messages; if it is the system who tries to recommend a service based on its behavior, the maximum weight is assigned to Inform-ref or Propose messages.

## 4.2 Weights vs Treatment of a Message: Function $\Phi$

As stated in Section 3, the $\Phi$ function computes the variation of the internal state of the agent caused by a memorization step and also a decision step. Memorization is evaluated by means of the $MM^A$. The defined ontology allows to measure that the internal state has changed due to the number of concepts involved or the number of attributes involved. We could have also considered the different relevance of the set of attributes and concepts by assigning different weights to each of them. For instance:

- Phase 10: Changes in concepts and their attributes may have a low value. In this phase, the User Agent provides the context information. Since the User Agent is never forced to transmit its personal information and preferences, the relevance of this package is low. The weights that are assigned to the context information depend on the utility of concepts and attributes to achieve the provision of the required service during the dialogue interaction.
- Phase 3 and 12: The weights assigned to the changes produced by the messages in the phase 3 depend on the number of services that whether available (positive) or not (negative) in the specific location (position and place). In this phase, the Positioning Agent communicates the position and place information to the User Agent. The weights are assigned similarly to the changes produced by each message in the phase 12. In this phase, the Facilitator Agent informs the User Agent about the services that are available in the current position.
- Phase 13: The utility of the messages corresponding to this phase, in which the Conversational Agent sends the log file of the dialogue to the Log Analyzer Agent, depends on the coincidence of this log with regard to the previous dialogues (partly time-decrescent function).
- Phase 14: Finally, a high weight is assigned to the message of this phase, in which the Log Analyzer Agent sends a user profile to the Conversational Agent. Since this profile is generated using several dialogue logs to personalize the provided service, a high relevance is assigned to this message.

The term $DD^A$ indicates the variation of the internal state due to a decision step. This function associates a value to each triggered action. Different types of actions must be defined to quantify this term, each having a specific weight. Then, the value of $DD^A$ is calculated as the sum of the weights of the triggered actions. The set of actions involved in our agent architecture can be classified as external or internal. External actions involve communicative responses for the given message. The weight of this reactive action is equivalent to the weight of the content included in the received message. Internal actions involve the processing and decision making described in the following phases:

- Phase 4: Once a User Agent knows its location, it asks the Facilitator Agent about the different services available in that: request message. Simple query to the internal database of the Facilitator Agent. No intelligence involved: minimal weight.
- Phase 6: The User Agent decides the services in which it is interested. Intelligent and relevant decision with a real economic cost: maximal weight.
- Phase 8: The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location: Simple query to the internal database of Facilitator agent. No intelligence involved: minimal weight.
- Phase 11: The User Agent provides the context information that has been required. Intelligent and relevant decision with a privacy cost: maximal weight
- Phase 13: Once the interaction with the Conversational Agent has finished, the Conversational Agent sends the log file generated after the dialogue to the Log Analyzer Agent. Simple query to the internal database of Conversational agent. No intelligence involved: minimal weight.
- Phase 14: The Log Analyzer Agent stores the log file and updates the user profile to personalize future services. Intelligent and relevant decision: medium weight.

Then we need to compute $DD^A$ function that associate the variation of internal state caused by decision step. This function associates a value to the triggered actions. To quantify, certain type of actions must be defined. A type of actions having a weight. Then, the value of the function $DD^A$ is considered as the sum of the weights of triggered actions. The set of actions involved in our agent system can be classified as external and internal. Where the external actions means communicative responses to the given message, where the weight of this reactive action is equivalent to the weight of the content included in the responsive message. On the other hand, internal actions involve the processing and decision making of the next phases:

### 4.3 Practical Application: The Case of an Airport Domain

We have used a previously defined domain of an Airport [8] as a testbed of our proposal. To introduce an experimentation case we use two different architectures. We compare them using different types of messages for a common domain: Airport presented in [8]. In previous work we design an architecture that has three types of agents: Central Agent, Provider Agents and User Agents [2]. From now on we call it MAS-CENTRAL-AGENT. Later we adapt this architecture to cope with new functionalities including the speech based interface and that is the new architecture presented in Figure 1, from now on we call it: MAS-CONVERSATIONAL-AGENT. New agents functionalities in this new architecture state as follow: *Positioning agent* main tasks rely on the user identification and user location into the environment. *Facilitator agent* is the responsible of the services management and the discovering of services agent identification. *Conversational Agents* provide the specific services. Finally, a *Log Analyzer Agent* generates user profiles that are used by Conversational Agents to adapt their behaviour taking into account the preferences detected in the users' previous dialogues.

If we compute the amount of messages exchanged of each type for each architecture, we could then draw some conclusions.

We take experimentation using "Service Recommendation" in an Airport Domain [8] for MAS-CENTRAL-AGENT and MAS-CONVERSATIONAL-AGENT. The comparison involves the following 5 type of messages: *Agree-proposal; Inform-ref; Propose; Query-if; Reject-proposal; Request.* For instance, once the passenger John Mayer is inside the Airport, the system recommends him with a SPA service based on the reputation. An example of message in FIPA is:

```
(inform-ref
:sender (agent-identifier :name SPA)
:receiver (set (agent-identifier :name john mayer))
:content
((action (agent-identifier :name SPA)
(try (spa $product $reputation))
:protocol fipa-request
:language FIPA-SL
:ontology airport-ontology
:reply-with try-spa)
```

In order to assign weights to each type of messages, we follow this criteria: if there is any interest over a negotiation from the Users side, then the maximum weight is assigned to the Agree-proposal or Reject-proposal messages; if it is the system who tries to recommend a service based on its behavior, the maximum weight is assigned to Inform-ref or Propose messages. In the case of MAS-CONVERSATIONAL-AGENT, the Inform-ref message belongs to Phase 12 of our MAS interaction and would have a maximal weight because it is the systems intention to recommend a service based on the behaviour of the passenger. In this case is that the passenger is tired. Then, following the evaluation method described above and used in [6], MAS-CONVERSATIONAL-AGENT in contrast to MAS-CENTRAL-AGENT. Among the MAS-CENTRAL-AGENT properties the reactivity, communication, robustness and scalability highlight apart from the others. On the other hand, the MAS1 is more robust when evaluating fairness and load balancing. In this sense, the difference between the two systems, when performing the comparison on the number of messages sent is minimal. Therefore, MAS-CONVERSATIONAL-AGENT is valid.

## 5 Conclusions

In this paper, we have detailed a generic interaction-based proposal for the evaluation of MAS. The evaluation of the interactions in Agent Systems makes possible to compare the various alternative agent architectures and protocols between them. The idea of considering agent interactions as evaluation benchmark is not new since it was proposed by [6] in a general form. In this paper, we have applied it and specified it in order to evaluate our Multi-Agent System (MAS) that provides Context-Aware Services. We have assigned evaluation values according to the general idea of [6]. Additionally we gave weights to the exchanged messages by the agents in our MAS architecture. Finally, the example of the Airport domain let us experience the complexity of the evaluation problem and state clear criterion for weight assignments. As future work we would aim to include a further evaluation of our system using different execution instances in several domains with this technique. Although the original evaluation method (from [6]) is itself general, it needs adhoc adaptation to be applied in whatever agent system, so presenting a particular adaptation of this general method is interesting due to it facilitates the general understanding of the evaluacion process of MAS, and it contributes to the general neccestiy of a theoretical comparison of different agent architecture approaches to any distributed problem.

## Acknowledgments

Funded by projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB, CAM MADRINET S-0505/TIC/0255, and DPS2008-07029-C02-02.

## References

1. Davidsson, P., Johansson, S., Svahnberg, M.: Characterization and evaluation of multi-agent system architectural styles. In: Software Engineering for Multi-Agent Systems IV, pp. 179–188 (2006)
2. V., Sanchez-Pi, N., Carbo, J., Molina, J.M.: Reputation in user profiling for a contextaware-multiagent system. In: EUMAS (2006)
3. Gaspar, G.: Communication and belief changes in a society of agents: Towards a formal model of autonomous agent, d.a.i. 2. In: Demazeau, Y., M.J.P.: (ed.) Descentralized A. I. 2, pp. 245–255. Elsevier Science, Amsterdam (1991)
4. Giunchiglia, F., Mylopoulos, J., Perini, A.: The tropos software development methodology: Processes, models and diagrams. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 63–74. ACM Press, New York (2002)
5. Gruber, T.: The role of common ontology in achieving sharable, reusable knowledge bases. In: 2nd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, USA, pp. 601–602 (April 1991)
6. Joumaa, H., Demazeau, Y., Vincent, J.: Evaluation of multi-agent systems: The case of interaction. In: 3rd International Conference on Information and Communication Technologies: From Theory to Applications, pp. 1–6. IEEE, Los Alamitos (2008)
7. Mylopoulos, J., Kolp, M., Giorgini, P.: Agent-oriented software development. Methods and Applications of Artificial Intelligence, pp. 748–748 (2002)
8. Sánchez-Pi, N., Fuentes, V., Carbó, J., Molina, J.M.: Knowledge-based system to define context in ommercial applications. In: Proc. of the 8th ACIS Conference SNPD 2007, Tsingtao, China, pp. 694–699 (2007)
9. Wooldridge, M.J., Ciancarini, P.: Agent-oriented software engineering: The state of the art. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 1–28. Springer, Heidelberg (2001)