An Awareness-Based Learning Model to Deal with Service Collaboration in Cloud Computing

Mauricio Paletta[1] and Pilar Herrero[2]

[1] Centro de Investigación en Informática y Tecnología de la Computación (CITEC)
Universidad Nacional Experimental de Guayana (UNEG)
Av. Atlántico, Ciudad Guayana, 8050, Venezuela
mpaletta@uneg.edu.ve
[2] Facultad de Informática, Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo S/N; 28660; Madrid, Spain
pherrero@fi.upm.es

Abstract. Cloud computing addresses the use of scalable and often virtualized resources. It is based on service-level agreements that provide external users with services under request. Cloud computing is still evolving. New specific collaboration models among service providers are needed for enabling effective service collaboration, allowing the process of serving consumers to be more efficient. This paper presents AMBAR-C, an adaptation of AMBAR (Awareness-based learning Model for distriButive collAborative enviRonment) designed to allow nodes in a distributed environment to accomplish an effective collaboration among service providers in a "cloud" by means of a multi-agent architecture in which agents are aware of its surroundings throughout a parametrical and flexible use of this information. As occurs in AMBAR, AMBAR-C makes use of heuristic strategies to improve effectiveness and efficiency in collaborations of these particular environments.

Keywords: Collaboration, multi-agent system, awareness, learning, cloud computing.

1 Introduction

Cloud computing [1] is emerging as a new distributed system that works toward providing reliable, customized and QoS guaranteed dynamic computing environments for end-users. The success of achieving this goal in proper time (efficiency) and/or to obtain higher quality results (effectiveness) in these dynamic and distributed environments depends on implementing an appropriate collaboration model between service providers in the *cloud*. Moreover, this collaboration mechanism should include learning abilities necessary for the use of the previous experience acquired (from situations that occurred in the past) in order to improve newly required collaborations. Learning-based heuristic techniques seem to be a good alternative to achieve this goal.

N.T. Nguyen and R. Kowalczyk (Eds.): Transactions on CCI I, LNCS 6220, pp. 85–100, 2010.

On the other hand, according to CSCW (Computer Supported Cooperative Work) awareness is a useful concept employed to achieve cooperation and collaboration in distributed environments because it increases communication opportunities [2]. A collaborative process is leaded by five processes [3, 4]: 1) co-presence, that gives the feeling that the user is in a shared environment with some other user at the same time; 2) awareness, a process where users recognize each other´s activities on the premise of co-presence; 3) communication; 4) collaboration, that together with communication permits users to collaborate with each other to accomplish the tasks and common goals; and 5) coordination which is needed to solve the conflicts towards effective collaborations.

In the same order of ideas, in CSCL (Computer Supported Collaborative Learning), awareness plays an important role as it promotes collaboration opportunities in a natural and efficient way [5] and improves the effectiveness of collaborative learning. Related to this, Gutwin et al identified the following types of awareness [6]: social, task, concept, workspace, and knowledge.

Moreover, SMI (Spatial Model of Interaction) [7] is one of the awareness models proposed as a way to obtain knowledge from the immediately closer world in collaborative virtual environments. It is based primarily on the use of a variety of mechanisms that were defined for this model and in addressing the interaction of nodes in a virtual environment. These are the concepts of medium, aura, focus, nimbus and awareness. The concept of awareness in this context, more explicitly awareness of interaction, is defined by quantifying the degree, nature and quality of the interaction between the elements of the environment.

By using an specific interpretation of the SMI-based awareness concept, AMBAR (Awareness-based learning Model for distriButive collAborative enviRonment) [8] was defined as a MAS-based learning collaboration model for distributed environments endowed with heuristic-based strategies. This model aims to approach the information of awareness in collaborations occurring in the environment for achieving the most appropriate future awareness situations. In this regard, this paper presents AMBAR-C, an adaptation of AMBAR aiming to be applied in *Cloud* environments.

The remainder of this paper is organized as follows. Some details of the AMBAR model are showed in section 2. Section 3 describes the adjustments that were made to AMBAR aiming to define AMBAR-C proposed in this paper. Details of the implementation and the validation of the model are showed in Section 4. Finally, the last section includes the conclusions and outgoing future research related to this work.

2 AMBAR: An Awareness-Based Learning Model for Collaborative Distributive Systems

This section presents some details of the AMBAR model. As can be seen in Fig. 1 AMBAR is structured by the following elements (more details are explained below in this section):

1) The awareness representation and the collaboration process associated with this representation.

2) An architecture used for designing the IA-Awareness intelligent agents (SOFIA).
3) A negotiation mechanism designed to deal with saturated conditions.
4) A mutual exclusion strategy to synchronize the use of critical sections.
5) A load-balancing strategy (CAwaSA).
6) Heuristic-based learning strategies (CAwANN).
7) A communication protocol that allows agents to exchange messages and therefore interact with each other.

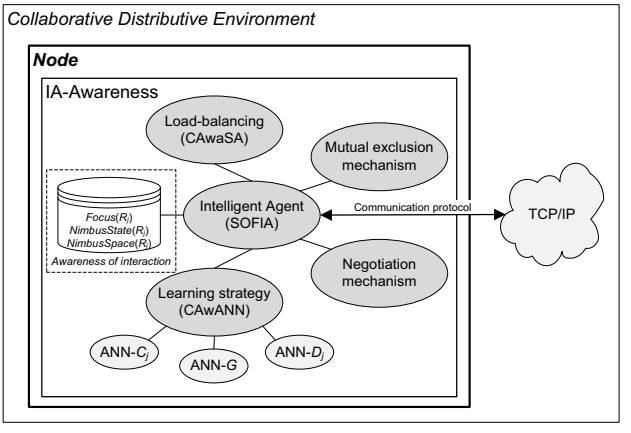


Fig. 1. The AMBAR structure

2.1 Awareness Representation and Collaboration Process

The collaborative process used in AMBAR and whose first results can be found in [9, 10, 11] uses the key awareness concepts originally proposed in [12, 13]. It has a distributed environment E containing a set of n nodes N_i ($1 \le i \le n$) and r different types of resources R_j ($1 \le j \le r$) that nodes can indifferently give to each other. These resources can be shared as a collaborative mechanism among different nodes. It has the following concepts:

1) $N_i.Focus(R_j)$: It is a set of resources which is interpreted as the subset of the distributed environment on which the agent in N_i has focused his attention aiming to interact or collaborate, according to the resource R_j.
2) $N_i.NimbusState(R_j)$: This indicates the current grade of collaboration that N_i can give over R_j. It could have three possible values: *Null*, *Medium* or *Maximum*. If the current grade of collaboration N_i that is given about R_j is not high, and this node could collaborate more over this resource, then $N_i.NimbusState(R_j)$ will get the *Maximum* value. If the current grade of collaboration N_i that is given over R_j is high but N_i could improve the collaboration over this service, then $N_i.NimbusState(R_j)$ would be *Medium*. Finally, $N_i.NimbusState(R_j)$ will be *Null* if N_i cannot offer R_j or if it cannot collaborate with this service any more.
3) $N_i.NimbusSpace(R_j)$: This represents the subset of the distributed environment where N_i aims to establish the collaboration over R_j.

4) $R_j.AwareInt(N_a, N_b)$: This concept quantifies the degree of collaboration over R_j between a pair of nodes N_a and N_b. It is manipulated via *Focus* and *Nimbus* (*State* and *Space*). Following the awareness classification introduced by Greenhalgh [14], values of this concept could be *Full*, *Peripheral* or *Null*. It is calculated according to (1).

$$R_j.AwareInt(N_a,N_b) = \begin{cases} Full, & N_b \in N_a.Focus(R_j) \wedge \\ & N_a \in N_b.NimbusSpace(R_j) \\ Peripheral, & (N_b \in N_a.Focus(R_j) \wedge \\ & N_a \notin N_b.NimbusSpace(R_j)) \vee \\ & (N_b \notin N_a.Focus(R_j) \wedge \\ & N_a \in N_b.NimbusSpace(R_j)) \\ Null, & \text{other case} \end{cases} \quad (1)$$

5) $N_i.TaskResolution(R_1,\ldots,R_p)$: N_i requires collaboration with all R_j ($1 \le j \le p$) to solve a specific task T.

6) $N_i.CollaborativeScore(R_j)$: Determines the score to collaborate over R_j in N_i. It is represented with a value between [0, 1]. The closer the value is to 0 the hardest it will be for N_i to collaborate with the needed R_j. The higher the value is (closer to 1) the completer will the willingness to collaborate be.

Any node N_a in the distributive environment is endowed with an IA-Awareness agent, that has the corresponding information about E, i.e.: $N_a.Focus(R_j)$, $N_a.NimbusState(R_j)$ and $N_a.NimbusSpace(R_j)$ for each R_j. The collaborative process in the system is as follows:

1) N_b must solve a task T by means of a collaborative task-solving process making use of the resources $R_1,\ldots,R_p$, and it generates a $N_b.TaskResolution(R_1,\ldots,R_p)$.

2) N_b looks for the current conditions to calculate the values associated to the key concepts of the model (*Focus*/*Nimbus* related to the other nodes), given by $N_i.Focus(R_j)$, $N_i.NimbusState(R_j)$ and $N_i.NimbusSpace(R_j)$ ($1 \le i \le n$; $1 \le j \le r$). This information is used to decide the most suitable node with which to collaborate related with any resource R_j (by using the load-balancing strategy CAwaSA). Nodes in this particular environment respond to requests for information made by N_b. This is done through the exchange of messages between agents (by using the communication protocol). As a final result of this information exchange the model will calculate the current awareness levels given by $R_j.AwareInt(N_i, N_b)$ as well as the collaboration score given by $N_b.CollaborativeScore(R_j)$.

3) For each resource R_j ($1 \le j \le p$) included in $N_b.TaskResolution(R_1,\ldots,R_p)$, N_b selects the node N_a whose $N_a.CollaborativeScore(R_j)$ is the most suitable to start the collaboration process (greatest score). Then, N_a will be the node in which N_b should collaborate on resource R_j.

4) Once N_a receives a request for cooperation, it updates its *Nimbus* (given by $N_a.NimbusState(R_j)$ and $N_a.NimbusSpace(R_j)$). In like manner, once N_a has finished collaborating with N_b it must update its *Nimbus*.

Each node in the system has an IA-Awareness agent designed to take into account the following features:

1) While each node may have different agents / processes, the IA-Awareness is the one that handles and manages the collaboration process; moreover, it learns to collaborate. In this sense, any need for cooperation from some source that is currently running on the node, communicates through the IA-Awareness service *TaskResolution*(R_1,…,R_p). In response to this service, IA-Awareness returns a list of p nodes, one for each resource R_j, better suited to collaborate with the current node in relation with the corresponding R_j.

2) There are services (abilities) that report on current levels of *Focus*(R_j), *NimbusState*(R_j) and *NimbusSpace*(R_j) for a specific resource R_j.

3) Once all the necessary information is achieved, the search for the most suitable nodes to collaborate related with any R_j is done by using the service *FindSuitableNodes*(R_1,…,R_p).

4) When conditions on the environment are not appropriated enough to establish a collaboration process (N_i.*NimbusState*(R_j) = *Null* for most of the N_i, R_j), the nature of the node N_b initiating a collaborative process to answer a N_b.*TaskResolution*(R_1,…,R_p) can lead to having no options, so that N_b can start a negotiation process that allows for N_b to identify new candidates to collaborate with. The detection of this saturated conditions is accomplished by using the service *IsOverloaded*(N, R).

5) The initiation and completion of the collaboration associated with the resource R is achieved through the implementation of services *StartCollaboration*(R) and *EndCollaboration*(R).

2.2 Agent Architecture

SOFIA (SOA-based Framework for Intelligent Agents) [10, 15] is the architecture used to design the IA-Awareness agents (“IA” stands for “Intelligent Agent”). It focuses on the design of a common framework for intelligent agents with the following characteristics: 1) it merges interdisciplinary theories, methods and approaches, 2) it is extensible and open as to be completed with new requirements and necessities, and 3) it highlights the agent´s learning processes within the environment. SOFIA’s general architecture contains four main components (see Fig. 2):

1) The Embodied Agent (IA-EA) or the “body”: It is a FIPA-based structure [16] because it has a Service Directory element which provides a location where specific and correspondent services descriptions can be registered. The IA-EA encloses the set of services related to the abilities of sensing stimuli from the environment and interacting with it.

2) The Rational Agent (IA-RA) or the “brain”: This component represents the agent’s intelligent part and therefore, it encloses the set of services used by the agent to implement the processes associated with these abilities. It is also a FIPA-based structure.

3) The Integrative/Facilitator Agent (IA-FA) or the “facilitator”: It plays the role of simplifying the inclusion of new services into the system as well as the execution of each of them when they are needed. The basic function of the IA-FA is to coordinate the integration between the IA-SV and the rest of the IA components. This integration is needed when a new service is integrated with the IA and therefore it is registered into the corresponding Service Directory, even when an existing service is being executed.

4) The IA Services or "abilities" (IA-SV): It is a collection of individual and independent software components integrated to the system (the IA) which implements any specific ability either to the IA-EA or to the IA-RA.

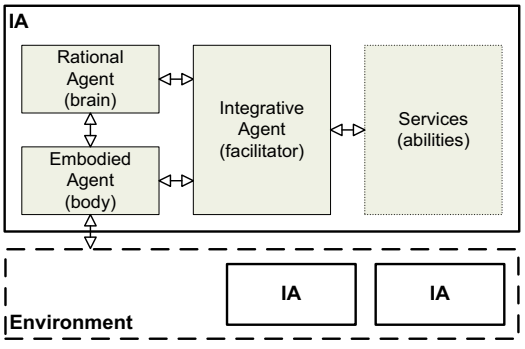


Fig. 2. The SOFIA general architecture

2.3 The Negotiation Mechanism

The negotiation mechanism included in AMBAR whose first results can be found in [17, 18] consists of three elements: 1) a heuristic algorithm used for deciding the most suitable node to initiate negotiation based on current conditions; 2) a heuristic method developed to accept/decline a need for collaboration during a negotiation; 3) a protocol for exchanging messages between agents.

For deciding the most suitable node to negotiate with, the idea is to define an unsupervised-based learning strategy aiming to correlate current information of the nodes in the distributive environment based on clusters. It is worth taking into account that "most suitable node" means a candidate that accepts the requirement established to collaborate with it, so that the negotiation is successful.

The basic idea is to find / identify a node *N* that is a potential candidate to negotiate with, taking into account the possibility to make changes in its *Nimbus* in relation with the resource *R*. *N* can then collaborate with this node in relation with *R*. "Potential" in this case means that *N* accepts to negotiate with the requesting node i.e. the negotiation is successful.

To achieve this goal a Neural-Gas (NGAS) [19] based strategy is used. The decision is to identify the closest node to the hyper-plane defined by the space given by the current environment conditions. In other words, it is necessary to determine the winning unit by testing the NGAS with the environment. The goal of this learning process is 1) to cluster the input data into a set of partitions such as the intra-cluster variance which remains small compared with the inter-cluster variance, and 2) to estimate the probability density function. This clustering scheme seems possible as we expect a strong correlation among the awareness information involved.

This learning strategy is a part of the mechanism that has been identified as CAwANN (Collaborative distributive environment by means of an Awareness & Artificial Neural Network model) [20]. In AMBAR the NGAS-based neural networks are identified as ANN-*G* (see Fig. 1).

Regarding accepting/declining collaboration, as with the decision of the most suitable node to negotiate with, this is also an Artificial Neural Network (ANN) based strategy. In this case there are s supervised ANN called ANN-D_j, one for each resource R_j in the system. All ANNs are defined in the same way and trained by using both Multi-Layer Perceptron (MLP) [21] and Radial Based Function Network (RBFN) [22] strategies.

2.4 The Mutual Exclusion Mechanism

The nature of a node initiating a collaborative process to answer a *TaskResolution*(R_1,…,R_p), provokes a change in the conditions of the collaboration levels of the environmental nodes involved in the process. Since this information is required by the process of taking action, the levels of collaboration between the nodes turn to a critical section, so that a mutual exclusion mechanism is required. The strategy used in AMBAR is a variation of the Naimi-Tréhel's token-based algorithm [23]. In the AMBAR token-based approach [24], the token travels with a queue Q which has the nodes that require the exclusive use of the critical section and haven't been able to satisfy that need.

2.5 The Load-Balancing Strategy

It has a set of p resources R_j ($1 \leq j \leq p$). For each resource it must identify the most suitable node in the environment with which to collaborate according to the corresponding resource. "Most suitable" means that it should considers the following assumptions:

1) The node N_b that seeks collaboration should be on the *Focus* of the node N_a to be able to identify it, i.e. $N_b \in N_a.Focus(R_j)$.
2) The score of collaboration given by $N_a.CollaborativeScore(R)$ must indicate the full readiness to collaborate over R (value equal or close to 1).
3) The selection must be done so that there would be a load-balancing mechanism distributed equally among all possible nodes with which to collaborate. This should take into account the current environment conditions given by $N_i.NimbusState(R_j)$ and $N_i.NimbusSpace(R_j)$ $\forall i,j$ $1 \leq i \leq n$, $1 \leq j \leq r$.
4) The answer must be given in a reasonable time after the request is generated.

Strategy used to solve this problem is based in the Simulated Annealing (SA) [25, 26] technique which is a generalization of a Monte Carlo method that searchs for a minimum in a more general system forming the basis of an optimization technique to solve combinatorial and other problems. This strategy is called CAwaSA (Collaborative Distributive Environment by means of Awareness and SA) and its first results can be found in [27].

2.6 Heuristic-Based Learning Strategies

In addition to the unsupervised-based method for selecting a potential candidate to negotiate on saturated conditions and the supervised-based method to learn the decision of whether or not a node must change the information that describes its current

conditions related with collaboration, AMBAR also incorporates a supervised-based method for learning collaborations based on levels of awareness. This process is about learning the association between the current status of the environment and the levels of collaboration obtained from that specific situation given by the $N_i.CollaborativeScore(R_j)$ ($\forall i$, $1 \leq i \leq n$ y $\forall j$, $1 \leq j \leq r$). The ANNs used in this solution are called ANN-C and, as happens with the ANN-D_j, the ANN-C are defined by using both MLP and RBFN strategies.

To differentiate one resource from another, given the fact that each service can have a different treatment in the levels of collaboration, the IA-Awareness has a different ANN-C_j network for each resource R_j. This is an important aspect of the strategy because:

1) Each resource can be trained separately from the rest.
2) The training process is less complex and, therefore, it is expected to obtain a higher quality in the response given by each ANN-C_j.
3) The model is expansible because new ANNs can be added when a new resource has to be incorporated into the environment.
4) Each node has a particular set of ANN-C_j, i.e. the IA-Awareness of each node trains and uses this ANNs according to the particular treatment a node wants to give to each resource R_j, making the collaboration model more flexible.

2.7 The Communication Protocol

Messages for AMBAR-based agent interaction are defined according to the FIPA performative and used for: 1) querying the current conditions of each node in the environment given by its *Focus*/*Nimbus*; 2) performing the mutual exclusion mechanism; 3) performing the negotiation mechanism; and 4) informing the initiation and completion of the collaboration associated with a particular resource. There are in total ten different messages.

The seven elements previously identified and briefly explained are those in which AMBAR is structured and IA-Awareness agents are defined, which are the basic components of the simulation tool proposed in this paper. The next section describes the proposed changes that were made to AMBAR to adapt this model to *cloud* environments in order to obtain the proposed AMBAR-C.

3 AMBAR-C: An Awareness-Based Learning Model for Service Collaboration in Cloud Computing

This section presents the differences between AMBAR and AMBAR-C explained here in order to offer a learning model for service collaboration in *cloud* environments. In this regard, the most important change is related to the use of services rather than resources. Therefore, in AMBAR-C there are s different types of services S_j ($1 \leq j \leq s$) that nodes can indifferently give to each other so that, concepts of $N_i.Focus(S_j)$, $N_i.NimbusState(S_j)$, $N_i.NimbusSpace(S_j)$ and $S_j.AwareInt(N_a, N_b)$ are redefined according to the given service S_j.

Any node N_a in the *cloud* is endowed with an intelligent agent who has the corresponding information about E, i.e.: $N_a.Focus(S_j)$, $N_a.NimbusState(S_j)$ and $N_a.NimbusSpace(S_j)$ for each S_j. The MAS-based collaborative process in the *cloud* follows these steps:

1) N_b solves a task T by means of a collaborative task-solving process by making use of the services $S_1,\ldots,S_p$.

2) N_b looks for the *cloud* current conditions given by $N_i.Focus(S_j)$, $N_i.NimbusState(S_j)$ and $N_i.NimbusSpace(S_j)$ $\forall i$, $1 \leq i \leq n$ and $\forall j$, $1 \leq j \leq s$. This information is used to decide the most suitable node with which to collaborate related with any service S_j. Service $N_i.GetCurrentState(S_j)$ is used to inquiring over node N_i for this information.

3) For each service S_j ($1 \leq j \leq p$) and by using the heuristic-based learning strategies that the intelligent agents in AMBAR-C have (see Section 2), N_b selects the node N_a which is the most suitable to start the collaborative process. Then, N_a will be the node in which N_b should focus on asking collaboration related on service S_j.

4) Service $N_a.S_j(data)$ is used for N_a to collaborate with N_b based on the needs of N_b to collaborate; *data* is the additional information N_a requires to execute S_j. As a consequence of receiving this service, N_a should update its *Nimbus* (given by $N_a.NimbusState(S_j)$ and $N_a.NimbusSpace(S_j)$). In like manner, once N_a has finished collaborating with N_b it must update its *Nimbus*.

5) In case of saturated conditions related with a service S_j and therefore no node N_a is selected as a candidate to collaborate with N_b, the AMBAR-C negotiation mechanism could be used as a variant of the AMBAR mechanism (Section 2.3). In AMBAR-C after deciding the most suitable node N_c to negotiate with, N_b executes the service $N_c.Negotiate(S_j)$ whose answer determines whether or not N_c accepts to collaborate with N_b related to S_j.

3.1 The Mutual Exclusion Mechanism

As occurs in AMBAR (Section 2.4), the levels of collaboration between the nodes in AMBAR-C turn also into a critical section, so that a mutual exclusion mechanism is also required. In AMBAR-C there is a different token for each service involved in a collaborative process. This difference has a positive impact on the efficiency and effectiveness of the model since it is now possible to start a number of collaborative processes in parallel, one for each service involved in the task and not on a unique collaborative process for all resources, as in the case of AMBAR.

The mutual exclusion mechanism of AMBAR-C makes use of the following services:

1) $N_a.AskToken(N_b, S_j)$ which is used for a node N_b to ask if another node N_a has the token associated with the service S_j. If N_a has the token it puts N_b in the corresponding queue Q; otherwise N_a does nothing.

2) $N_b.SetToken(S_j, Q)$ which is used for a node N_a, that is supposed to have the token associated to S_j, to send the token to N_b that is supposed to be the first node in Q.

3.2 Agent Communication

Instead of using message exchange for agent communication which is the case on AMBAR, AMBAR-C makes use of service exchange. As was seen previously in this section it can be done by using the list of services indicated in Table 1.

Table 1. Services for agent communication

Service	Description
GetCurrentState(S_j)	Ask current state related to S_j.
Negotiate(S_j)	Negotiate on saturated conditions related to S_j.
AskToken(N_b, S_j)	Require the token N_b be able to collaborate related to S_j.
SetToken(S_j, Q)	Assign the token related to S_j with its corresponding Q.

Next Section presents some details related with the implementation of AMBAR-C as well as a case of study to validate its applicability.

4 Implementation and Validation

As was mentioned in Section 2, IA-Awareness is the SOFIA-based agent used for implementing AMBAR. This IA-Awareness has been implemented in JADE [28] because it is FIPA-compliant as well as an open-source (based on Java). IA-Awarennes-C which is a specialization of IA-Awareness has been implemented to design AMBAR-C. IA-Awarennes-C uses the same nucleus of IA-Awarennes although there is a difference with the component "Embodied Agent" or IA-EA (see Fig. 2) which in the case of IA-Awarennes-C uses Web Service[1] technology for agent communication.

On the other hand, aiming to measure the effectiveness (θ) and efficiency (ξ) of the collaboration process, expressions (2) and (3) were defined respectively. Note that both measures (θ, ξ) are positive values in [0, 1] where 1 is the maximum value for effectiveness and efficiency. Variables in expressions are the following:

- *PSN*: It is the percentage of successful negotiations made in saturated conditions, based on the number of negotiations that receive a positive response from a node requesting to change its current saturated conditions in relation to the total attempts made.
- *MDN*: It is the mean duration in seconds of the negotiation process under saturated conditions. The process starts at the moment the node requires the cooperation until it receives an answer, whether affirmative or negative. For both answers the possible retries to be made are taken into account.
- *ATC*: It is the average time of collaboration in seconds.
- *PSC*: It is the percentage of successful collaborations based on the number of services in which there was positive response from a node to collaborate with, in relation to the total quantity of services in which collaboration was required.

[1] `http://www.w3.org/2002/ws/`

- *TOT*: It is the number of times in which any timeout expires.
- *AMT*: It is the average number of queries sent in the request for the token.
- *ATT*: It is the average waiting time in seconds for the token.
- *ATB*: It is the mean duration in seconds of CAwaSA method in solving the load-balancing problem.
- *ATL*: It is the mean duration in seconds of the learning process.

Moreover, to differentiate the context in which effectiveness or efficiency is calculated, individual expressions for the entire model (MOD), the mutual exclusion mechanism (MEM), the load-balancing problem (LBP), the negotiation mechanism (NEG), and the learning process (LPR) are given. Factors α, β, χ and δ, all of them in [0, 1] which sum is exactly equal to 1, determine the importance of each component of the model (MEM, LBP, NEG and LPR) for measuring θ and ξ.

$$\begin{aligned}
&\theta(\mathrm{MOD}) = \alpha\theta(\mathrm{MEM}) + \beta\theta(\mathrm{LPB}) + \chi\theta(\mathrm{NEG}) + \delta\theta(\mathrm{LPR}) \\
&\theta(\mathrm{MEM}) = 3/AMT \\
&\theta(\mathrm{LBP}) = PSC/100 \\
&\theta(\mathrm{NEG}) = PSN/100 \\
&\theta(\mathrm{LPR}) = (PSC + PSN)/200
\end{aligned} \tag{2}$$

$$\begin{aligned}
&\xi(\mathrm{MOD}) = \alpha\xi(\mathrm{MEM}) + \beta\xi(\mathrm{LPB}) + \chi\xi(\mathrm{NEG}) + \delta\xi(\mathrm{LPR}) - 0.05 * TOT \\
&\xi(\mathrm{MEM}) = \begin{cases} ATT/ATC, & \mathrm{A}TT <= ATC \\ ATC/ATT, & \mathrm{A}TT > ATC \end{cases} \\
&\xi(\mathrm{LBP}) = 1 - ATB/ATC \\
&\xi(\mathrm{NEG}) = 1 - MDN/ATC \\
&\xi(\mathrm{LPR}) = 1 - ATL/ATC
\end{aligned} \tag{3}$$

4.1 A Case Study

In order to validate the use of AMBAR-C a case of study related with the design of packages of combined trips is presented. It proposes to attend a reservation that is made implicit in a trip and also to attend all the possible items involved, such as transport, room and board, and tourism. This case is related to a distributed system that possesses nodes that collaborate with each other to offer a combined package that satisfies the necessities of the traveler.

The application is developed searching to design a package of combined trips made for diverse purposes (tourism, business, pleasure and others). Businesses that work in and with the travelling area are benefited from this application. Generally, a package of combined travels involves a series of items such as: room and board, hotel, transportation, vehicle rental, tickets to shows or tourist attractions among others. Occasionally, clients that chose this type of businesses to request a travelling package are offered an incomplete package because the business finds it impossible to include some of the items in the package. As a consequence the client must work on some of the remaining arrangements separately from the business.

It is necessary then an application that allows businesses that work within the travelling area to collaborate with each other to complete the requested travelling packages. If this happened then a company that could not complete a package because it cannot offer a certain item can then complete the package by asking the missing item to some other companies that can offer it. Companies benefit from this because their profits rise by participating as a whole in combined packages that are requested in any of them and their clients receive a complete package without having to work on some of the missing items individually.

The nodes N_i of the collaborative distributive environment E in this case are formed by those businesses that posses one or more services related to the elements that can be a part of a combined travelling package and that wish to participate in this collaboration system. The quantity of nodes allowed to be in the system at the beginning is unlimited. The services S_j are related in this case to the services or elements that can be a part of a combined travelling package and the related operations such as: making reservations, cancelling reservations, changing a reservation or purchase and purchase and payment of an item. For example, for an item related with transportation the following services are at hand: making reservations for a certain kind of transportation, cancelling a reservation, changing a previously made reservation or purchase, purchasing tickets for a certain kind of transportation, and so on.

The apparition of a new element that could be a part of a combined travelling package implicates the incorporation of four new services in the environment. These services are associated to one of the operations previously described. The quantity of elements to be considered in this application to be a part of a combine travelling package is unlimited. Specific parameters such as the company in charge of the final service (for example: airlines, vehicle rental companies and hotels), the evaluation given from users, the number of stars of a hotel, the price and so forth, can be used to define new resources in such a way that the collaboration in this specific parameter might become focused.

Based on this representation of the environment, the interpretation that is given to expression $N_1.Focus(S_1) = \{N_2, N_3, N_4\}$, for example, is that the business represented by N_1 can, whenever there is a necessity of including in a combined travelling package the element represented by S_1 (for example make a reservation for any kind of transportation), search collaboration from the businesses N_2, N_3 and N_4 that are supposed to be habilitated to make this kind of operation. An equivalent expression to, for example, $N_1.NimbusState(S_2) = Null$ indicated that the business N_1 offers the service represented by S_2, and that N_1 is or is not collaborating in relation with the service S_2 at present. Finally, an expression equal to $N_4.NimbusSpace(S_4) = \{N_2, N_3\}$ indicates that at present N_4 is, for example, making transactions to acquire tickets for a particular transportation (S_4) due to different collaboration requests made by N_2 and N_3.

4.2 Experimental Results

Experiments were conducted with AMBAR-C in a TCP/IP-based local network which assumes that each node can directly communicate with any other node. Different scenarios were simulated aiming to rate the capability for managing the growth of the nodes in the different environment conditions. The scenarios were defined by

changing the quantity of nodes/PCs n (agents) as well as the number of services s according to $n \in \{4, 8\}$ and $s \in \{2, 6, 10\}$. Therefore 6 different scenarios were simulated: 1) $n = 4$, $s = 2$; 2) $n = 4$, $s = 6$; 3) $n = 4$, $s = 10$; 4) $n = 8$, $s = 2$; 5) $n = 8$, $s = 6$; and 6) $n = 8$, $s = 10$. Moreover:

1) The initial condition of the *cloud* for each scenario ($N_i.Focus(S_j)$, $N_i.NimbusState(S_j)$ and $N_i.NimbusSpace(S_j)$; $1 \leq i \leq n$; $1 \leq j \leq s$) was randomly defined by considering the following: one node belongs to the *Focus* of another node with a probability of 0.75 and to the *NimbusSpace* with a probability of 0.85.

2) All N_b nodes execute an automatic process that randomly selects a need to collaborate in relation with a particular service.

3) The simulation time was 120 minutes for each scenario.

4) A timeout of one minute to a node for waiting the information of the current state of each node and a timeout of two minutes given for negotiation of resources under saturated conditions were configured.

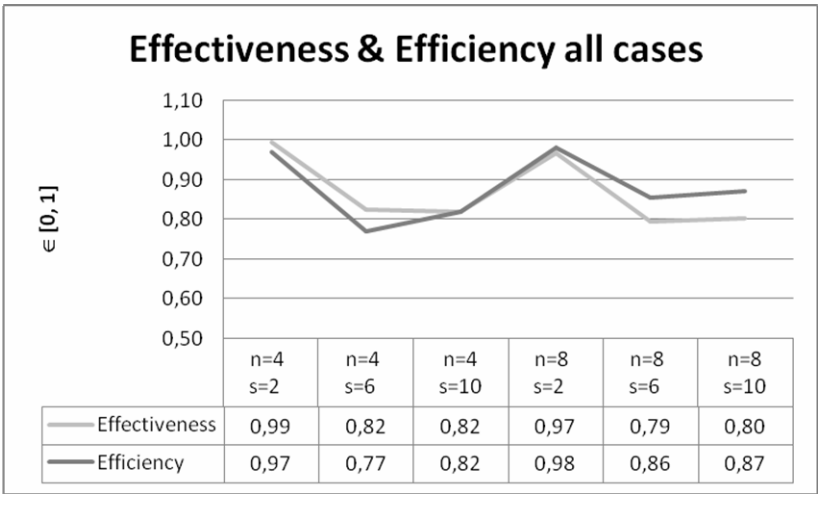
5) By setting $\alpha = 0.1$, $\beta = 0.25$, $\chi = 0.3$ and $\delta = 0.35$ we give more importance to the learning process aiming to analyze differences in efficiency and effectiveness of the model before and after the model has learned to collaborate.

Table 2. Measures obtained from experimentation

Measure	n=4 s=2	n=4 s=6	n=4 s=10	n=8 s=2	n=8 s=6	n=8 s=10
PSN	100.00	68.13	64.29	93.75	78.13	100.00
MDN	0.00	2.14	1.23	0.13	2.16	0.44
ATC	3.40	3.46	3.34	7.47	14.87	19.28
PSC	98.96	94.24	97.70	99.89	78.08	57.90
TOT	0.00	0.00	0.00	0.00	0.00	0.00
AMT	3.05	3.01	3.02	3.04	3.28	3.64
ATT	2.47	5.86	9.58	6.73	167.38	445.15
ATB	0.00	0.02	0.05	0.06	0.68	2.03
ATL	0.01	0.02	0.02	0.01	0.03	0.02
θ(MEM)	0.98	1.00	0.99	0.99	0.92	0.83
θ(LBP)	0.99	0.94	0.98	1.00	0.78	0.58
θ(NEG)	1.00	0.68	0.64	0.94	0.78	1.00
θ(LPR)	0.99	0.81	0.81	0.97	0.78	0.79
θ(MOD)	0.99	0.82	0.82	0.97	0.79	0.80
ξ(MEM)	0.72	0.60	0.35	0.89	0.13	0.05
ξ(LBP)	1.00	0.99	0.98	0.99	0.95	0.89
ξ(NEG)	1.00	0.38	0.64	0.98	0.85	0.98
ξ(LPR)	1.00	0.99	0.99	1.00	1.00	1.00
ξ(MOD)	0.97	0.77	0.82	0.98	0.86	0.87

Table 2 shows the measures obtained after the simulations. Fig. 3 shows the effectiveness and efficiency of the model related with these measures. According to these results it is possible to make the following observations and/or conclusions:

1) The average effectiveness is 0.86 and the average efficiency is 0.88.
2) Both effectiveness and efficiency have a similar trend of behaviour.
3) Nor the variation in the number of nodes or the variations in the number of services resources have a particular tendency to improve or worsen the effectiveness and efficiency.
4) The effectiveness and efficiency tends to improve with the time and therefore the learning process becomes better. Moreover, due to the fact that it is a learning-based mechanism from past situations, it is assumed that as there is much more to learn, the metrics associated with it must be improved.



	n=4 s=2	n=4 s=6	n=4 s=10	n=8 s=2	n=8 s=6	n=8 s=10
Effectiveness	0,99	0,82	0,82	0,97	0,79	0,80
Efficiency	0,97	0,77	0,82	0,98	0,86	0,87

Fig. 3. Effectiveness and efficiency obtained from experimentation

5 Conclusion and Future Work

This paper presents AMBAR-C a new collaboration model used for a multi-agent based system in collaborative *cloud* computing environments. It has been designed as an adaptation of AMBAR (Awareness-based learning Model for distriButive collAborative enviRonment). The method proposed in this paper is endowed with heuristic techniques to: 1) solve the load-balancing problem, 2) decide for the most suitable node to collaborate with, and 3) to decide whether or not to collaborate.

AMBAR-C has been designed for *cloud* environments that includes nodes that are aware of the existence of others notes that can establish a collaborative process. Results show that AMBAR-C is 86% effective and 88% efficient according to the proposed functions of effectiveness and efficiency. Therefore, this model ensures collaboration in these environments in short time.

Although this method has not been tested in real case scenarios, it has been designed to be suitable for real *cloud* computing environments. In fact the experimentation and validation were carried out to demonstrate that this method could be extended to real case scenarios with no problems. Moreover, there are no limitations for the use of this collaboration model. This model in most likely to work with no problems in environments that are not necessarily *cloud*-based where collaboration/cooperation is needed.

We are currently working on testing this method in real environments as well as using the model in actual end-use applications.

References

1. Weiss, A.: Computing in the Clouds. NetWorker 11(4), 16–25 (2007)
2. Matsushita, Y., Okada, K. (eds.): Collaboration and Communication. Distributed collaborative media series 3. Kyoritsu Press (1995)
3. Kuwana, E., Horikawa, K.: Coordination Process Model - based Design for Synchronous Group Task Support System. Technical Reports of Information Processing Society of Japan, No. 13, pp. 1–6. Groupware (1995)
4. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys 26(1), 87–119 (1994)
5. Ogata, H., Yano, Y.: Knowledge Awareness: Bridging Learners in a Collaborative Learning Environment. International Journal of Educational Telecommunications 4(2), 219–236 (1998)
6. Gutwin, C., Stark, G., Greenberg, S.: Support for Workspace Awareness in Educational Groupware. In: Proc. Computer Supported Collaborative Learning (CSCL '95), pp. 147–156 (1995)
7. Benford, S.D., Fahlén, L.E.: A Spatial Model of Interaction in Large Virtual Environments. In: Proc. 3rd European Conference on Computer Supported Cooperative Work, pp. 109–124. Kluwer Academic Publishers, Dordrecht (1993)
8. Paletta, M., Herrero, P.: Collaboration in Distributed Systems by means of an Awareness-based Learning Model. Accepted to be published in Recent Patents on Computer Science (CSENG). Bentham Science Publishers (to be published, 2010)
9. Paletta, M., Herrero, P.: Learning cooperation in collaborative grid environments to improve cover load balancing delivery. In: Proc of IEEE/WIC/ACM Joint Conferences on Web Intelligence and Intelligent Agent Technology, vol. E3496, pp. 399–402. IEEE Computer Society, Los Alamitos (2008)
10. Paletta, M., Herrero, P.: Towards fraud detection support using grid technology. Multiagent and Grid Systems - An International Journal 5, 311–324 (2009)
11. Paletta, M., Herrero, P.: Foreseeing cooperation behaviors in collaborative grid environments. In: Proc. of 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09), vol. 50, pp. 120–129. Springer, Heidelberg (2009)
12. Herrero, P., Bosque, J.L., Pérez, M.S.: An agents-based cooperative awareness model to cover load balancing delivery in grid environments. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 64–74. Springer, Heidelberg (2007)
13. Herrero, P., Bosque, J.L., Pérez, M.S.: Managing dynamic virtual organizations to get effective cooperation in collaborative grid environments. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part II. LNCS, vol. 4804, pp. 1435–1452. Springer, Heidelberg (2007)

14. Greenhalgh, C.: Large Scale Collaborative Virtual Environments. Ph.D. Thesis, University of Nottingham, UK (1997)
15. Paletta, M., Herrero, P.: Awareness-based learning model to improve cooperation in collaborative distributed environments. In: Håkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2009. LNCS, vol. 5559, pp. 793–802. Springer, Heidelberg (2009)
16. Foundation for Intelligent Physical Agents: FIPA Abstract Architecture Specification. SC00001, Geneva, Switzerland (2002), http://www.fipa.org/specs/fipa00001/index.html
17. Paletta, M., Herrero, P.: A MAS-based Negotiation Mechanism to deal with Service Collaboration in Cloud Computing. In: Proc. International Conference on Intelligent Networking and Collaborative Systems (INCoS 2009), pp. 147–153. IEEE Computer Society, Los Alamitos (2009)
18. Paletta, M., Herrero, P.: A MAS-based Negotiation Mechanism to deal with saturated conditions in Distributed Environments. In: Proc. Second International Conference on Agents and Artificial Intelligence (ICAART 2010) (to be published, 2010)
19. Martinetz, T.M., Schulten, K.J.: A neural gas network learns topologies. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.) Artificial Neural Networks, pp. 397–402 (1991)
20. Paletta, M., Herrero, P.: An awareness-based artificial neural network for cooperative distributed environments. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M. (eds.) IWANN 2009. LNCS, vol. 5517, pp. 114–121. Springer, Heidelberg (2009)
21. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall, Englewood Cliffs (1998)
22. Jin, R., Chen, W., Simpson, T.W.: Comparative studies of metamodelling techniques under multiple modeling criteria. Struct. Multidiscip. Optim. 23, 1–13 (2001)
23. Naimi, M., Trehel, M., Arnold, A.: A log (N) distributed mutual exclusion algorithm based on path reversal. J. Parallel Distributed Computing 34(1), 1–13 (1996)
24. Paletta, M., Herrero, P.: A Token-Based Mutual Exclusion Approach to Improve Collaboration in Distributed Environments. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS (LNAI), vol. 5796, pp. 118–127. Springer, Heidelberg (2009)
25. Kirkpatrick, S.: Optimization by simulated annealing: Quantitative Studies. J. Statistical Phy. 34(5-6), 975–986 (1984)
26. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machines. J. Chem. Phy. 21(6), 1087–1091 (1953)
27. Paletta, M., Herrero, P.: An awareness-based simulated annealing method to cover dynamic load-balancing in collaborative distributed environments. In: Baeza-Yates, R., et al. (eds.) Proc. 2009 IEEE/WIC/ACM International Conference on Intelligence Agent Technology (IAT 2009), pp. 371–374. IEEE Computer Society, Los Alamitos (2009)
28. Bellifemine, F., Poggi, A., Rimassa, G.: JADE-A FIPA-compliant agent framework. Telecom Italia internal technical report. In: Proc. International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAM'99), pp. 97–108 (1999)