Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications xxx (2006) xxx–xxx
Expert Systems with Applications
www.elsevier.com/locate/eswa

Hybrid multi-agent architecture as a real-time problem-solving model

C. Carrascosa [a,*], J. Bajo [c], V. Julian [a], J.M. Corchado [b], V. Botti [a]

[a] Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera, s/n, 46022 Valencia, Spain
[b] Departamento Informática y Automática, Universidad de Salamanca, Salamanca, Spain
[c] Universidad Pontificia de Salamanca, Salamanca, Spain

Abstract

This paper presents a multi-agent architecture that facilitates the development of real-time multi-agent systems based on the SIMBA approach. The approach allows the integration of unbounded deliberative processes with critical real-time tasks. CBP-BDI deliberative agents collaborate with ARTIS agents in order to solve real-time problems efficiently. The proposal has been successfully tested and evaluated in a case study based on the use of mobile robots for mail delivery.

Keywords: Multi-agent systems; Real-time systems; Case-based reasoning

1. Introduction

The current application of multi-agent systems in real-time environments is an area of increasing interest. In general, the multi-agent system represents an appropriate approach for solving inherently distributed problems, whereby clearly different and independent processes can be distinguished. Examples of problems with these characteristics are mobile robot teams, in which several mobile robots develop a common task, or the problems of control and management of intelligent buildings. In these systems a set of sensors and effectors are distributed throughout the environment, and the agents must be coordinated to meet an acceptable level of safety and efficient use of resources. Moreover, some temporal restrictions must be taken into account. It is important to emphasize that these problems can also be typical examples of real-time systems, which might make multi-agent systems applicable in environments of this kind.

There are few studies related to real-time agent development and real-time multi-agent systems (Goldman, Musliner, & Krebsbach, 2001; Graham, 2001). The SIMBA real-time multi-agent platform is one of these (Carrascosa, Rebollo, Soler, Julian, & Botti, 2003; Soler, Julian, Rebollo, Carrascosa, & Botti, 2002). The main goal in SIMBA is to provide an execution environment where it is possible to merge hard real-time characteristics with intelligent components. As such, the SIMBA approach can be placed in the area of Real-Time Artificial Intelligence Systems (RTAIS) and is a useful tool for solving complex problems which require intelligence and real-time response times. SIMBA allows flexible, adaptive, and intelligent real-time behaviours showing that the multi-agent system paradigm is especially appropriate for developing systems in real-time environments. SIMBA incorporates real-time ARTIS agents. This paper shows how such agents collaborate with CBP-BDI deliberative agents (Bajo & Corchado, 2005; Corchado & Laza, 2003; Glez-Bedia & Corchado, 2002) in the framework proposed by SIMBA, in an efficient way, to solve real-time problems.

One of the main problems that needs to be overcome is the efficient integration of high-level, multi-agent planning processes within this kind of architecture. These complex deliberative processes, which allow the agent to adapt and learn, are unbounded and it is difficult to integrate them in hard real-time systems. Typically, in the multi-

* Corresponding author. Tel.: +34 96 387 7352; fax: +34 96 387 73 59.
E-mail address: carrasco@dsic.upv.es (C. Carrascosa).

0957-4174/$ - see front matter © 2006 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2006.08.031

agent area these processes are carried out by so-called deliberative agents, which decide what to do and how to do it according to their mental attitudes. In a deliberative agent, it is relatively simple to identify decision processes and how to perform them. However, its main drawback lies in finding a mechanism that permits its efficient and temporal bounded execution. Therefore, it would be interesting to integrate complex deliberative processes for decision-making in hard real-time systems in a simple and efficient way.

BDI (Believe, Desire, Intention) deliberative agents are systems with representations that are directed towards the action model (Bratman, 1987). Such agents may incorporate a case-based reasoning (CBR) motor (Aamodt & Plaza, 1994), which constitutes the base of a planning system that is based on previous plans, Case-Based Planning (CBP) (Hammond, 1989; Carbonell, 1983). This type of model meets the conditions needed to introduce a representation and a reasoning based on the action (Pollack, 1992). A CBR-BDI agent (Corchado & Laza, 2003) uses case-based reasoning as a reasoning mechanism, which allows it to learn from initial knowledge, to interact autonomously with the environment and with users and the other agents within the system, and to have a large capacity for adaptation to the needs of its surroundings. We shall refer to the CBR-BDI agents specialised in generating plans, as CBP-BDI agents, where a plan is defined as a sequence of document collection and delivery points.

A multi-agent system that includes deliberative and pure reactive processes has been implemented using the SIMBA platform. In order to validate the hypothesis, we propose the coordination of multi-agent systems. The problem to be used will be developed within a restricted test environment (known number of robots, familiar environment, etc.). In the case study proposed for the evaluation of the hypothesis, the SIMBA architecture will be integrated with both ARTIS agents (Botti, Carrascosa, Julian, & Soler, 1999), (which are capable of guiding mobile robots in real time), and CBP-BDI deliberative agents (which generate and distribute plans in the execution time of the ARTIS agents). Therefore, the deliberative agents are responsible for planning the routes that should be followed by the mobile robots, and the ARTIS agents put these plans into action until insurmountable obstacles are encountered, in which case an alternative plan is requested from the deliberative agent. Here we propose the automation of the management of internal mail in a department that is physically distributed on a single floor of a building. The department is divided into sections. In each section there is one mail robot that is responsible for attending to requests made by a user in the department. These requests can be made using a PDA or a desktop computer. In the same way, the robots are responsible for collecting and delivering external mail received by the department or for mail to be sent out externally. As mentioned above, each robot is governed by an ARTIS agent that is capable of managing the behaviour of each robot. A deliberative CBP-BDI agent is responsible for generating the optimum plans for the collection and delivery of mail, as well as assigning plans to each ARTIS agent when it has the possibility of working under real-time restrictions that are not considered critical.

As part of the work proposed, it was necessary to define the model for communicating among the system's agents, taking into account that the problem is developed with a real-time domain. In other words, responses need to be given in real-time. The interaction between agents does not interfere with the behaviour of the real-time agents and can be adopted temporarily. For the purposes of the study, the case is presented with the aid of AUML and Gaia designs in order to facilitate comprehension and the interrelationship between the agents that make up the multi-agent system.

The article is structured as follows: Section 2 presents the SIMBA multiagent architecture for developing real-time distributed systems; Section 3 presents the CBP-BDI agents, placing special emphasis on their capacity for planning; Section 4 presents the case to be studied; and, lastly, the evaluation is presented and the results obtained are analysed.

2. SIMBA: a multi-agent architecture for real-time problems

SIMBA (Multi-agent system based on ARTIS) (Carrascosa, Rebollo, Soler, et al., 2003; Soler et al., 2002) is an agent platform that allows the development of real-time multi agent systems (RTMAS). The architecture of this platform is shown in Fig. 1. The SIMBA platform consists of a set of ARTIS agents and a special agent (Manager Platform Agent – MPA) which controls the services specified in the standard FIPA (http://www.fipa.org). These services are: agent management services (also called white pages service – AMS); directory management services (also called yellow pages – DF). This agent also controls interoperability with other FIPA platforms across an agent communication channel (ACC). With this platform, the ARTIS agents are transformed into social real-time agents that can communicate with other agents by means of an agent communication language (ACL). It is important to emphasize that hard, real-time communication has not been introduced, and it is not guaranteed to receive the packets on time or without errors.

The main characteristics of the SIMBA platform are:

- Distributed platform, each agent in the platform is executed in a different host.
- FIPA ACL is used as a communication language.
- The size of messages is limited in order to fit into a network packet.
- UDP/IP network protocol is used in the communication between agents within the platform.

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031

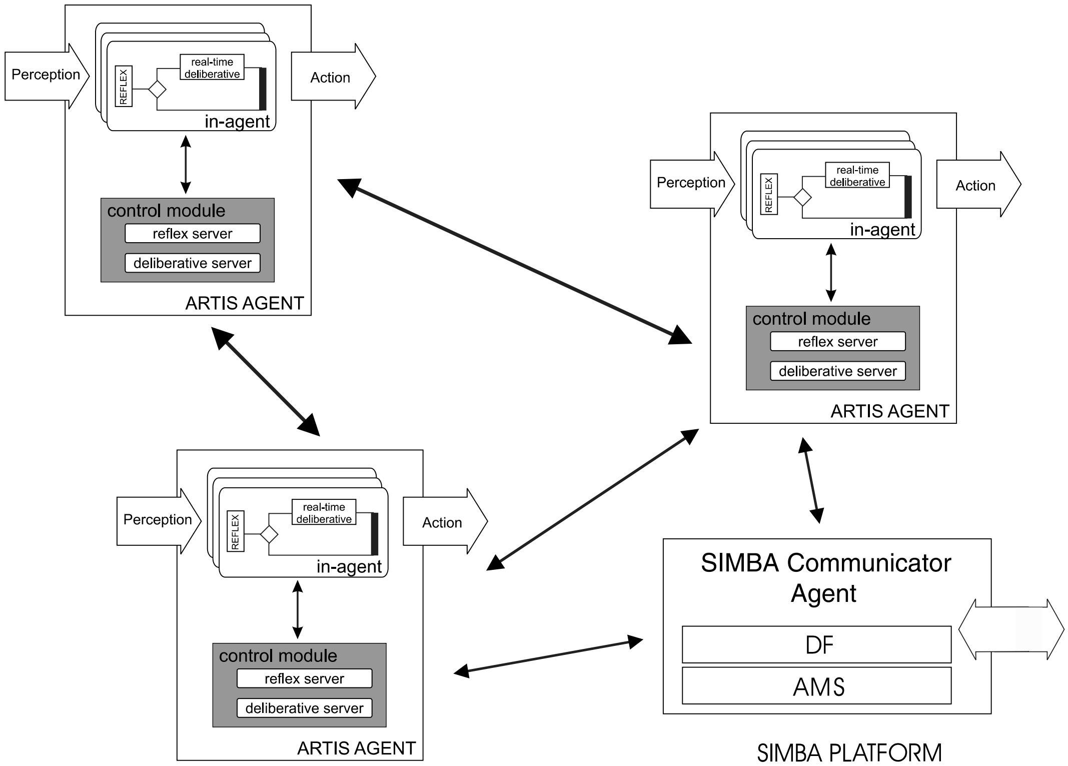

Fig. 1. The SIMBA platform architecture.

2.1. *ARTIS agent: a hard, real-time, intelligent agent*

This point provides a short description of the ARTIS Agent (AA) architecture for hard real-time environments (a more detailed description can be found in Botti et al., 1999; Carrascosa, Rebollo, Julian, & Botti, 2003). The AA architecture could be labeled as a vertical-layered, hybrid architecture with added extensions to work in a hard, real-time environment.

One of the main features of the AA architecture is its hard, real-time behaviour. It guarantees the execution of the entire system specification by means of an off-line analysis of the specification. This analysis is based on well-known predictability analysis techniques in the real-time community and is defined in Garcia-Fornes, Terrasa, Botti, and Crespo (1997). The off-line analysis only ensures the schedulability of real-time tasks. However, it does not force task sequence execution. The AA decides the next task to be executed at runtime, allowing it to adapt itself to changes in the environment and to take advantage of the tasks that use less time than their *wcet*. The AA reasoning process can be divided into two stages. The first one is a mandatory time-bound phase. It obtains an initial result of satisfactory quality. After that, if there is time available (also called slack time in the RTS literature), the AA can use this time for the second reasoning stage. This is an optional stage and does not guarantee a response. It usually produces a higher quality result through intelligent, utility-based, problem-solving methods. This split reasoning process is described in detail in Botti et al. (1999).

The architecture of an AA can be viewed from two different perspectives: the user model (high-level model) (Botti et al., 1999) and the system model (low-level model) (Terrasa, Garcia-Fornes, & Botti, 2002). The user model offers the developer's view of the architecture, while the system model is the execution framework used to construct the final version of the agent.

From the user model point of view, the AA architecture is an extension of the blackboard model, which is adapted to work in hard, real-time environments. It is made up of the following elements:

- A set of sensors and effectors to be able to interact with the environment. Due to the environment's features, the perception and action processes are time-bound.
- A set of beliefs comprising a world model (containing all the domain knowledge relevant to the agent) and the internal state – the mental states of the agent. This set is stored on a frame-based blackboard (Barber, Botti, Onaindia, & Crespo, 1994).
- A set of behaviours that models the answer of the AA to different situations. It could be said that a state (internal along with a representation of the environment) defines a situation (represented by the current beliefs and goals) which activates a behaviour or allows it to go on being active. This behaviour determines the agent's current set of goals and restrictions, along with the knowledge needed to control the situation. Each one of these behaviours is formed by a set of in-agents. The main reason for splitting the whole problem-solving method into

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031

in-agents is to provide an abstraction level that organizes the problem solving knowledge in a modular and gradual way. Each in-agent periodically performs a specific task. An in-agent is also an agent according to Russell's agent definition (Russell & Norvig, 2003). Each in-agent has to solve a particular subproblem, but all the in-agents of a specific AA cooperate to control the entire problem. An in-agent may use information provided by other in-agents. In-agents can be classified into critical and non-critical. Critical in-agents solve essential problems of the AA, so that their execution is assured at least for calculating a low quality answer. Non-critical in-agents solve non-essential problems of the AA to improve its performance quality. A critical in-agent is characterised by a period and a deadline. The available time for the in-agent to obtain a valid response is limited. It must guarantee a basic response to the current environmental situation. From a functional point of view, an in-agent consists of two layers: the reflex layer and the real-time, deliberative layer. The reflex layer assures a minimal quality response (an off-line schedulability analysis of the AA that takes into account all the in-agents in the AA guarantees that this reflex layer will be fully executed). On the other hand, the real-time deliberative layer tries to improve this response (this level will be executed in slack time). The reflex layer of all the in-agents make up the AA mandatory phase, and the real-time deliberative layers form the optional phase. A non-critical in-agent only has the real-time deliberative layer.
- A control module that is responsible for the real-time execution of the in-agents that belong to the AA. The temporal requirements of the two in-agent layers (reflex and deliberative) are different. Thus, the control module must employ different execution criteria for each one.

The ARTIS agents, presented in this section will work in collaboration with a CBP-BDI agent, which generates plans in execution time that help the ARTIS agents to deliver physical mail in an efficient way and to deal with unpredictable problems.

3. CBP-BDI agents

Deliberative agents can be constructed using different conceptual paradigms (Bratman, 1987; Rao & Georgeff, 1995; Wooldridge & Jennings, 1995). One of the most widely used and best known of these is one that defines the agents in terms of their Beliefs, Desires, and Intentions (BDI) (Rao & Georgeff, 1995). This definition of an agent facilitates the construction of dynamic systems that are capable of reasoning and generating imaginative solutions. In order to do this, the agents must incorporate mechanisms that allow them to generate plans. In this case, it is assumed that the agents respond in a rational way and in real time, so they must incorporate mechanisms that allow them to reason and generate results within a limited, preestablished time frame. (Bajo & Corchado, 2005; Corchado & Laza, 2003; Glez-Bedia & Corchado, 2002) propose the use of case-based reasoning systems as a planning mechanism for deliberative agents. These agents are capable of generating new plans from information on past experiences stored in the form of cases. In this article, we go one step further and present the concept of a CPB-BDI agent. The CPB-BDI agent acts as an "intelligent" system that plans its mode of action by reusing information from the most suitable past plans for solving a current problem and adapting it to the current situation (thereby creating the planning space). This section shows how variational techniques can be used and how the minimum Jacobi field helps the agent to obtain the most re-plannable alternative route if a plan is interrupted. The planning is carried out following the framework established by case-based reasoning systems (Aamodt & Plaza, 1994). As such, the resolution of a new problem (in this case, the identification of a new plan) is based on (i) the retrieval of solutions (in this case, plans) used in the past or similar to the case problem; (ii) the adaptation of these solutions to the current problem; (iii) the revision of the solution proposed (optional stage in many CBR systems); and finally, (iv) the inclusion of new experiences in the case base (or plans in this instance). The information is stored in the plan base.

In this study, the CBP-BDI agent has the objective of identifying the most suitable route for a mail agent to follow in order to facilitate the delivery and collection of information. The ARTIS mail agents request routes from the CBP-BDI agent at the beginning of the mail delivery and collection process and whenever an unexpected event interrupts the initial plan. During the planning of the routes, the CBP-BDI agents evaluate the current situation and the packages to be delivered, taking into account that the work should be carried out in as short a time as possible. Both the ARTIS mail agents and the CBP-BDI agents are integrated in the SIMBA multi-agent architecture.

Now we shall introduce the planning CBP-BDI model, taking into consideration that the testing environment is restricted. Let $E = \{e_0, \ldots, e_n\}$ be the set of the possible collection points and mail delivery. $e_j, j \in \{0, \ldots, n\}$ represents the point of collection of the external mail provided by the postman.

In each action a, the agent goes from the delivery point to the mail collection point or vice versa

$$a_j : \underset{e_i}{E} \;\underset{\rightarrow}{\rightarrow}\; \underset{a_j(e_i)=e_j}{E}$$

Agent plan is the name we give to a sequence of actions that (from a current state e_0) defines the path of states through which the agent passes in order to reach the other mail delivery or collection point. Below we model the dynamic relationship between the behaviour of the agent and the changes in the environment.

We represent the behaviour of *agent A* by its *function action* $a_A(t)$ $\forall t$, which is defined as a correspondence between one moment in time t and the action selected by the agent,

$$Agent\ A = \{a_A(t)\}_{t \in T \subseteq N}$$

From the definition of the *action function* $a_A(t)$, we can define a new relationship that includes the idea of an agent's *action plan*,

$$p_A : \underset{(t, a_A(t))}{TxA} \underset{\rightarrow}{\rightarrow} \underset{p_A(t)}{A}$$

in the following way:

$$p_A(t_n) = \sum_{i=1}^{n} a_{iA}(t_i - t_{i-1})$$

Given the dynamic character that we want our agent to have, we propose the continuous extension of the previous expression as a definition of the *agent plan*, in other words

$$p_A(t_n) = \int_{t_0}^{t_n} a_A(t)\,\mathrm{d}t$$

The variation of the agent plan $p_A(t)$ will be invoked essentially by:

1. The *changes that occur in the environment that force the initial plan to be modified*.
2. The knowledge from the success and failure of the plans that were used in the past, which were favoured or penalized via *learning*.

- O indicates the objectives of the agent and O' are the results achieved by the plan.
- R are the total resources and R' are the resources consumed by the agent.

Efficiency of the plan: the relationship between the objectives attained and the resources consumed

$$E_{ff} = \frac{\#(O' \cap O)}{\#R'}$$

(# means cardinal of a set).

The objective is to introduce an architecture for a planning agent that behaves – and selects its actions – by considering the possibility that the changes in the environment block the plans in progress. We call this agent MRP (the most re-planning-able agent) because it continually searches for the plan that can most easily be re-planned in the event of interruption.

- Given an initial point e_0, we use the term *planning problem* to describe the search for a way of reaching a final point $e_i \equiv e^* \in E$ that meets a series of requirements.

Let X be a discrete variable that can take values of a numerable set that we represent as

$$X = \{x_i\}_{i \in N}$$

Then, we can define the *associated accumulated variable*, which we denote as $Ac(X)$, for a new variable that is constructed by assigning each of the possible values xi taken by variable X that is the total of previous results.

If X is discrete, the ith value of the variable $Ac(X)$ is defined as

$$Ac(x_i) = \sum_{j=1}^{i} x_j \quad \forall x_i \in X$$

If the variable X is continuous and its values are in the interval $[a, b]$, it is represented by the function $x(t)$; we define the variable $Ac(X)$ at a point $x_i \in [a, b]$

$$Ac(x_i) = \int_a^{x_i} x(t)\,\mathrm{d}t \quad \forall x_i \in [a, b]$$

Given a problem E and a plan $p(t)$, we can construct functions Ob and Rc, that are accumulated from the objectives and costs of the plan. For all time points t_i, we can associate two variables

$$Ob(t_i) = \int_a^{t_i} O(t)\,\mathrm{d}t, \quad Rc(t_i) = \int_a^{t_i} R(t)\,\mathrm{d}t$$

This allows us to construct *a planning space* (or space that represents the environment for planning problems) as a vectorial hyperdimensional space where each axis represents the *accumulative variable* associated with each objective and resource.

The planning space defined in this way conforms to the following properties:

1. *Property 1*: The representation of the plans within the planning space are always monotonously growing functions. Given that $Ob(t)$ and $Rc(t)$ are functions defined as positive (see definition), function $p(t)$ expressed at these coordinates is constant or growing.
2. *Property 2*: In the planning space, the straight lines represent plans of constant efficiency. If the representation of the plans are straight lines, the slope of the function is constant and coincides with the definition of the efficiency of the plan.

$$\frac{\mathrm{d}}{\mathrm{d}t} p(t) = cte \iff \lim_{\Delta \to 0} \frac{\Delta O(t)}{\Delta R(t)} = cte$$

In an n-dimensional space, the extension of the straight concept line is called a geodesic curve. In this sense, we can introduce the notion of *geodesic plans* that are defined as those that maintain efficiency at a constant throughout their development.

The concept of a geodesic plan can be better understood as a "*plan of minimum risk*". If the environment is changeable, any other relationship with efficiency that is not constant will imply that the agent makes plans for the future (it considers that, in the future, certain efficiency relationships will be met and as such it makes sense to assume greater or lesser efficiency ratios). In an environment that changes unpredictably, any plan that is distal to the geodesic plan means that a certain risk is accepted.

Given a problem, the agent must search for the plan that determines a solution with a series of restrictions $F(O; R) = 0$. In order to deal with these restrictions, we are going to make a change in the coordinates: instead of seeking plans of constant efficiency that are adjusted to $F(O; R) = 0$, we construct the hyperplan that collects all such information, and we calculate the straight line within it (which is in general no-euclidean).

In the plan base, we search for those plans that are initially compatible with the problem faced by the agent, with the requirements imposed on the solution according to the desires, and in the current state (Aamodt & Plaza, 1994). If we represent all the possible plans $\{p_1, \ldots, p_n\}$ within the *planning space*, we can obtain a *subset of states* that the agent has already attained in the past in order to resolve similar problems.

- With the mesh of points obtained within the planning space (which is generally irregular) and using interpolation techniques, we can obtain a working hyperplan $h(x)$ (that encapsulates the information on the set of restrictions from restored experiences), from which we can calculate geodesic plans.
- From the values given $\{f(x_i)\}_{i=1,\ldots,n}$, where $X = \{x_i\}_{i=1,\ldots,n}$ are variables in the planning space, the theory of functions of radial base as combinations of B-Splines proposes an expression of $h(x)$ in the following way (Reuter, Tobor, Schlick, & Dedieu, 2003):

$$h(x) = m(x) + \sum \lambda_i \phi(\|x - x_i\|_2) \quad x, x_i \in \Re^d, \\ \lambda_i \in \Re \ \forall i \tag{1}$$

The coefficients λ_i of the function $h(x)$ are determined by requiring h to satisfy the interpolation conditions

$$h(x_j) = f(x_j) \quad j = 1, \ldots, n$$

where functions $\phi(x)$ are a complete base of orthogonal functions. Duchon (1977) has demonstrated that the selection of cubic functions are the most suitable in interpolation problems for obtaining the Smoothest function (Hegland, Roberts, & Altas, 1997)

$$\phi(x) = (\|x\|_2)^3$$

The system of equations (Eq. (1)) can be resolved either directly or by the conjugated gradient method. The cost of the solution will be at most $O(k^3)$ (Beatson & Light, 1997). The software used to make these calculations is known as JSpline+ (Spline library for Java), which uses a development based on radial functions (Duchon B-splines) (Duchon, 1977) so that the information on the restriction space $h(x)$ can be reduced to tackle the coefficient vector λ_i. The coefficients vector λ_i encapsulates all the information needed to manage the restriction associated with a problem.

The variation calculation (Schutz, 1993) consists in a set of mathematical techniques that allows us to know the geodesic paths between one point in a non-euclidean space and a set of points represented by a function that we call the *function of final states* and which we denote as f_sf.

In general, the simplest variation problem is given when f_sf is only one point in the space, $f_sf = e^*$, and the geodesic g that links with e^* is obtained (Fig. 2).

In a problem where the set of end points is $n > 1$, *variation techniques with mobile frontiers* are used. They offer a set of geodesics between the starting point and each one of the points of the final set. If $f_sf = \{e_1, \ldots, e_m\}$, we obtain a geodesic set $\{g_1, \ldots, g_m\}$.

Below, we apply variation calculation techniques for the planning problem that has been set.

Given a problem that requires a plan that allows it to pass from to $e^* \in f_sf$ conforming to restriction $F(O; R) = 0$, we can construct the hyperplan of restrictions $h(x)$, with which we can apply variation calculation. Suppose for simplicity's sake that we have a planning space of dimension 3 with coordinates $\{O, R_1, R_2\}$.

Between the point e_0 and the objective points f_sf and over the interpolation surface $h(x)$, the Euler Theorem (Glez-Bedia & Corchado, 2002; Jost & Li-Jost, 1998) guarantees that we will obtain the expression of the geodesic plans by resolving the following system of equations:

$$\begin{cases} \dfrac{\partial L}{\partial R_1} - \dfrac{\mathrm{d}}{\mathrm{d}O}\dfrac{\partial L}{\partial R_1'} = 0 \\ \dfrac{\partial L}{\partial R_2} - \dfrac{\mathrm{d}}{\mathrm{d}O}\dfrac{\partial L}{\partial R_2'} = 0 \end{cases} \tag{2}$$

where R_i is the function acculmulatedR, O is the function of accumulatedO, and L is the distance function on the hyperplan $h(x)$,

$$L = \int_h \mathrm{d}l$$

In order to obtain all the geodesic plans that, on the surface $h(x)$ and beginning at e_0, allow us to reach any of the points $e^* \in f_sf$, we must impose that the initial point is $e_0 = (O_0, R_0)$ as a condition of the surrounding.

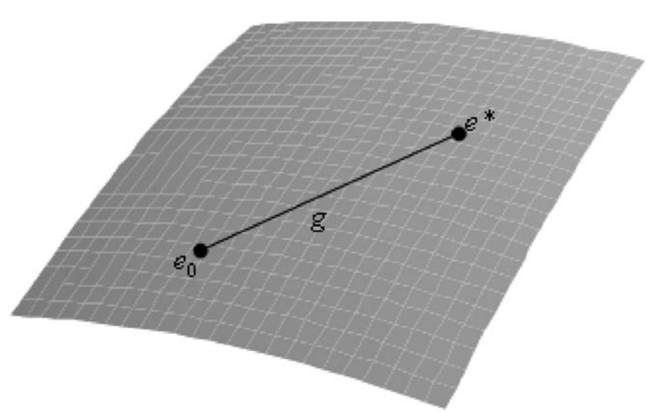

Fig. 2. Geodesic g linking the initial and final points.

Using variation techniques, we obtain expressions for all the geodesic plans that, beginning at e_0 allow us to attain the desired point.

Once plans that will create efficient solutions between the current state and the set of solution states have been obtained, we will be able to calculate the plan around it (along its trajectory) by a denser distribution of geodesic plans (in other words, a greater number of geodesic plans in its environment). The tool that allows us to determine this is called the *minimum Jacobi field associated with the solution set* (Lee, 1997).

Let $g_0 : [0,1] \to S$ be a geodesic over a surface S. Let $h : [0,1]x[-\varepsilon,\varepsilon] \to S$ be a variation of g_0 so that for each $t \in (-\varepsilon,\varepsilon)$, the set $\{h_t(s)\}_{t\in(-\varepsilon,\varepsilon)}$:

- $h_t(s)\ \forall t \in (-\varepsilon,\varepsilon)$ are geodesic in S,
- they begin at $g_0(0)$, in other words, they conform to $h_t(0) = g_0(0)\ \forall t \in (-\varepsilon,\varepsilon)$.

In these conditions, taking the variations to a differential limit, we obtain

$$\lim_{t\to 0}\{h_t(s) = g_0(s+t)\} = \lim_{t\to 0}\{h(s,t)\} = \left.\frac{\partial g_0}{\partial t}\right|_{(s,0)} = \frac{\mathrm{d}g_0}{\mathrm{d}s} \equiv J_{g_0}(s)$$

We use the term $J_{g_0}(s)$ to refer to the Jacobi Field of the geodesic g_0 for the set $\{g_n(x)\}_{n\in N}$. In the same way that the definition has been constructed, we give a measurement for the distribution of the other geodesics of $\{g_n(x)\}_{n\in N}$ around g_0 throughout the trajectory.

Given a set of geodesics, some of them are always g^* which, in their environment, have a greater distribution than other geodesics in a neighbouring environment. This is equivalent to saying that it presents a variation in the distribution of geodesics that is lower than the others and, therefore, the Jacobi Field associated with $\{g_n(x)\}_{n\in N}$ reaches its lowest value at J_{g^*}.

Let us return to the MRP agent problem that, following the recuperation and variation calculation phase, contains a set of geodesic plans $\{p_1,\ldots,p_n\}$. If we select the p^* that has a minimum Jacobi Field value, we can guarantee that, in the event of interruption, it will have around it a greater number of geodesic plans to be able to continue. To select this plan would mean selecting the solution that can most easily revert to another if it is interrupted.

For our problem, the minimum Jacobi field is synonymous with the capacity for replanning. This suggests the following definition: given a problem with certain restrictions $F(O;R) = 0$, we can call the geodesic plan p^* with minimum associated Jacobi field associated with the set $\{g_n(x)\}_{n\in N}$ as the *most re-plan-able solution.*

The behaviour model G for the MRP agent is defined. For each problem that it represents, the agent selects the most replannable solution, which is defined as the geodesic plan with minimum Jacobi field that expresses

$$G(e_0,p_1,\ldots,p_n) = p^* \iff \exists n \in N / J_{g_n} \equiv J_{g*} = \mathrm{Min}_{n\in N} J_{g_n}$$

With this result, we can characterise the agent's mode of behaviour. If the plan p^* is not interrupted, the agent will reach a desired state $e_j \equiv e^* \in f_s f,\ j \in \{1,\ldots,m\}$. A weighting $w_f(p)$ is stored in the learning phase. With the updating of weighting $w_f(p^*)$, the planning cycle of the CBP (Cased-Based Planning) engine is completed. Next, we see what happens if p^* is interrupted.

Let us suppose that the agent has initiated a plan p^*, but at a moment $t > t_0$, the plan is interrupted due to a change in the environment.

The geodesic planning (the section of plans with a constant slope in the planning space) meets the conditions of the Bellman Principle of Optimality (Bellman, 1957). In other words, each one of the plan's parts is partially geodesic between the selected points.

This guarantees that if g_0 is geodesic for interrupted e_0 in t_1, because e_0 changes to e_1, and g_1 is geodesic to e_1 that is begun in the state where g_0 has been interrupted, it follows that:

$$g = g_0 + g_1 \text{ is geodesic to } e = e_0(t_1 - t_0) + e_1(t_2 - t_1)$$

In other words, we can construct our global plan in "pieces". Every time the environment changes and interrupts the execution plan, a new geodesic plan is selected and the *overall plan will be geodesic*.

The dynamic process follows the CBP cycle recurrently: every time a plan is interrupted, it generates the surroundings of the plans from the case base and adjusts them to the new problem. It then calculates the geodesic plans and selects the one which meets the minimum conditions of the associated Jacobi field. The dynamic planning model of the agent $G(t)$ is characterised in this way (Fig. 3). The following properties of $G(t)$ are particularly relevant in the dynamic context:

1. *Property 1*: All the Jacobi fields are variations of geodesics.
   It can be demonstrated (Milnor, 1973) that there exists an isomorphism between all the Jacobi fields that are constructed between the end points.
2. *Property 2*: All the geodesic variations are Jacobi fields (Milnor, 1973).

These two results allow us to introduce the concept of a *global Jacobi field*. We use the term *Global Jacobi field or Dynamic Jacobi field* $J(t)$ to describe a Jacobi field made up of a set of partial or successive Jacobi fields. The above properties allow us to ensure that the change from one partial Jacobi field and the next preserves the conditions of a Jacobi field because it produces a change between geodesics.

It can observed that a minimum global Jacobi field $J(t)$ also meets Bellman's conditions of optimality (Bellman,

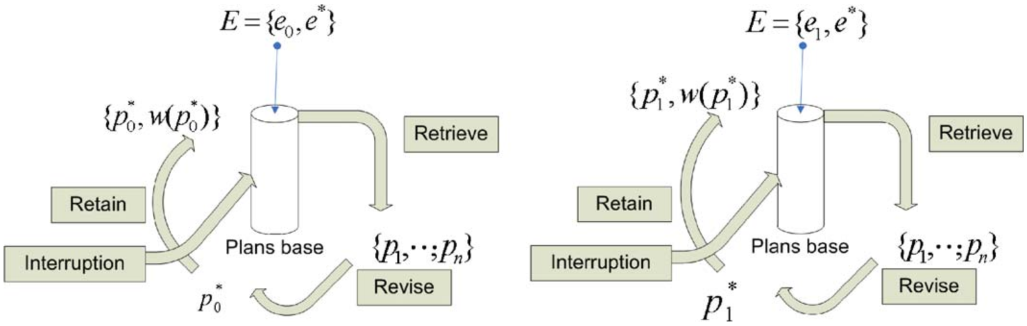

Fig. 3. Model for behaviour $G(t)$.

1957). In other words, a minimum global Jacobi field must select minimum Jacobi fields "in pieces"

$$J_{\min}(t) = \{J_{\min}(t_1 - t_0), J_{\min}(t_2 - t_1), \ldots, J_{\min}(t_n - t_{n-1})\}$$

If successive Jacobi fields generate one Jacobi field, and minimum Jacobi fields generate one minimum Jacobi field, the MRP agent that follows a strategy of replanning $G(t)$ as indicated in order to survive a dynamic environment, it generates a *global plan* $p^*(t)$ that, compared with all possible global plans $\{p_n(t)\}_{n\in N}$, presents a minimum value in its Jacobi field $J_{g^*}(t) \equiv J_{p^*}(t)$.

This section has formally defined an agent, that when placed in a dynamic environment seeks plans that lend it greater capacity for replanning.

4. Case study: postman robots

The problem to solve consists of the automated management of the internal and external mail (regular, non-electronic mail) in a department. In order to do this, the system must allow requests for the shipment of a letter or package from one office on one floor to another office on the same floor, as well as the reception of external mail to be taken to a collection point for later distribution.

Once this service has been requested, a mobile robot (or postbot) must gather the shipment and direct it to the destination. It is important to note that each mail or package distribution must be finalized before a maximum time, that is specified in the original request. In order to be able to carry out all of this, the resources employed include a series of mobile robots – Mobile Pioneer 2 – and a radio network for communication around the plant.

Given these resources, the problem will be solved through a real-time multi-agent system in which heterogeneous agents collaborate by means of a SIMBA platform. This platform gives real-time support to the system since the physical agents that manage the mobile robots must satisfy critical temporal restrictions, and are designed according to the ARTIS hard real-time agent architecture. In addition, all the planning processes for the delivery and collection of mail around the plant are managed by a deliberative planning agent. This agent will give the most suitable distribution routes to each available robot. This planning agent has been developed following the CBP-BDI model.

As mentioned in the introduction, the department is divided into sections. In each section, there is a mail agent that attends to mail requests. If an agent of one section receives a task and is busy carrying out a previously assigned task, it can request help from agents in adjacent sections (for example if there in not enough battery power to carry out another delivery).

If there is an agent from an adjacent section that is not carrying out a task at the time, it will take on the new task. If there are more than one agent available, the task will be assigned to the agent with the longest battery life. If all the agents within the section are busy, the agent that is capable of the most suitable re-planning will carry out the task. Once the agent has completed this extra task (from a different section), it returns to its own section if there are more tasks to be carried out, or if it has run out of battery power. Otherwise, it will continue to help the agent that requested it.

4.1. Analysis and design of the system

The option chosen for defining a suitable analysis and design methodology for our problem is to use a combination of Gaia (Wooldridge, Jennings, & Kinny, 2000) and AUML (Bauer, 2001; Bauer & Huget, 2003; Odell & Huget, 2003; Odell, Levy, & Nodine, 2004). This combination takes advantage of the benefits of both systems. An analysis of the problem can be made using the criteria of organisation and a preliminary design of GAIA. The Gaia design has been adapted so that AUML techniques can be applied (Bauer, 2001; Bauer & Huget, 2003; Odell & Huget, 2003; Odell et al., 2004). Fig. 4 shows the steps to be taken in this approach. First Gaia is used to obtain the analysis and high-level design, and then AUML is used to obtain a detailed design at a low level.

The first step of the process is to carry out a high-level analysis and design using Gaia. The roles of the system are identified: a planner role, whose principal responsibility is to plan the routes that the robots should take to deliver the mail as efficiently as possible; a distribution role, whose principal responsibility is to carry out the plans indicated by the planner; a user role which makes the requests for the sending of internal mail and receives delivery confirma-

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031

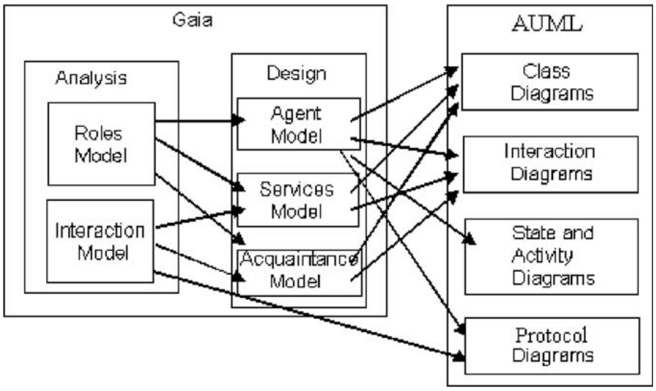

Fig. 4. Gaia-AUML analysis and design process.

tions from each service; and a postman role, which reports the arrival of new external mail and the collection of mail that is sent out externally. A role model is then created using these roles. Fig. 5 shows the Gaia role model for the planner role. It illustrates the following: how a role description is given; how its protocols and activities are described; the permissions that the role has concerning the system; and the responsibilities or functionalities that the role carries out for the system.

In addition, the different interactions that are produced between the roles are also identified. These interactions are:

- Plan Execution: The Planner role sends a plan to the PostBot role to be carried out. The plan is communicated step by step, and each achieved objective is reported.
- Robot state: The Planner role asks the PostBot role about its state (location, state of battery, etc.) before proceeding with a plan.
- Execution incident: The PostBot role requests a solution for an incident from the Planner role.
- Updated Plan Execution: The Planner role provides the PostBot role with an updated plan.
- Low battery: The PostBot role detects a low battery state and communicates it to the Planner role.
- New mail arrival: The Postman role reports the arrival of new external mail to the department.
- Internal mail request: The User role wishes to send internal mail.

Fig. 6 illustrates an example of an interaction using the GAIA methodology, in which a robot needs to recharge its battery while executing a plan.

Once the Gaia analysis has been completed, a Gaia high-level design is carried out to obtain the agents, services, and known models. The agent model that appears in Fig. 7 shows the agents that participate in the system, the roles that each agent plays, and the multiplicities in execution time. For example, the Planner agent plays the PLANNER role and there will only be a single Planner agent in execution time.

Fig. 8 shows the acquaintance model for our SMA. It shows the relationships or communication routes that exists between the different agents in the systems. In this example, it shows how the Planner agent has the PostBot, User, and Postman agents as its known agents.

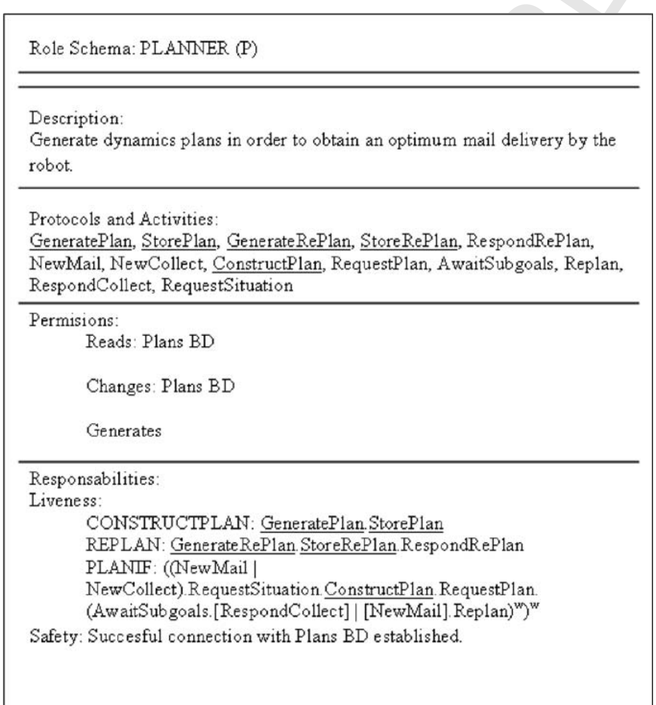

Fig. 5. Roles model for the PLANNER role.

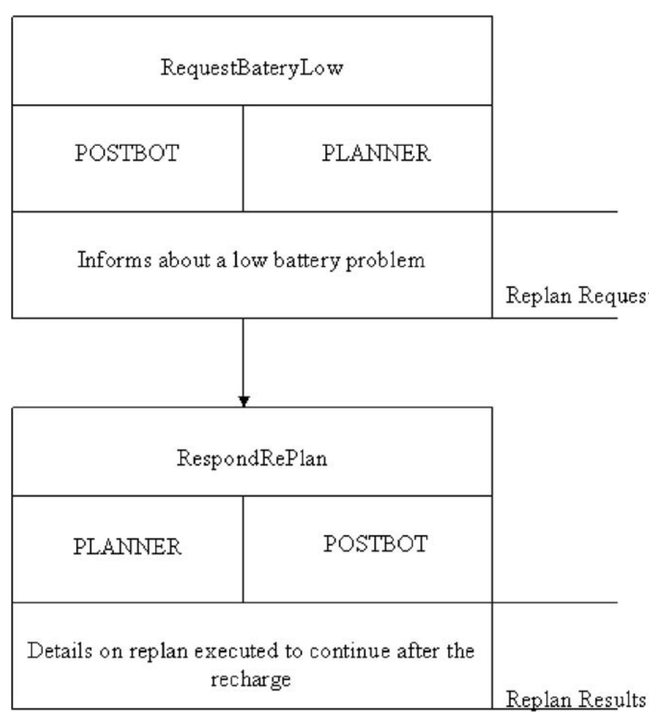

Fig. 6. Replan execution BateryLow interaction model.

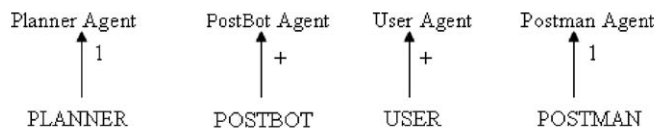

Fig. 7. Model of Gaia agents for the PostBots.

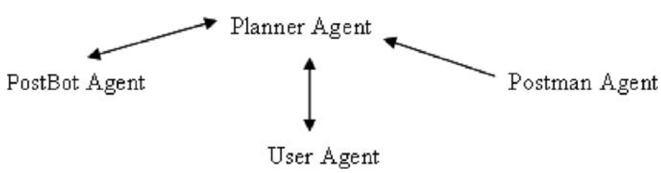

Fig. 8. Gaia acquaintance model for the mail robot problem.

Once the Gaia high-level design has been carried out, a detailed design of a low level AUML is carried out. As mentioned above, this derives from the results obtained after applying the Gaia methodology to obtain the agents, interactions and protocol models for the AUML activities and states. A diagram of classes showing the capacities and services of each agent is made for each agent. The roles of the agent are obtained from the roles that were identified in the Gaia agent model, but with a more detailed description. The AUML role definition is more specific than Gaia and introduces the concept of capacity to carry out each role. An AUML role is obtained from each Liveness responsibility from the Gaia role model.

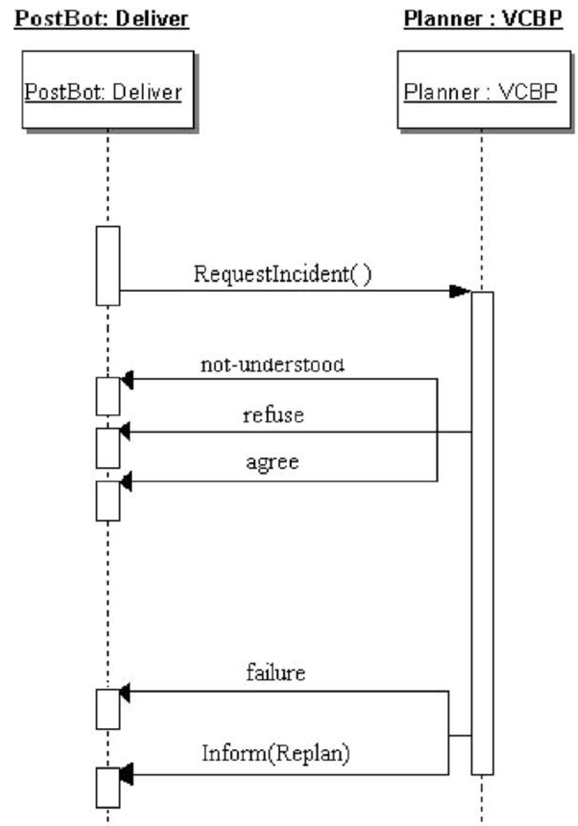

Fig. 10. Replan execution protocol.

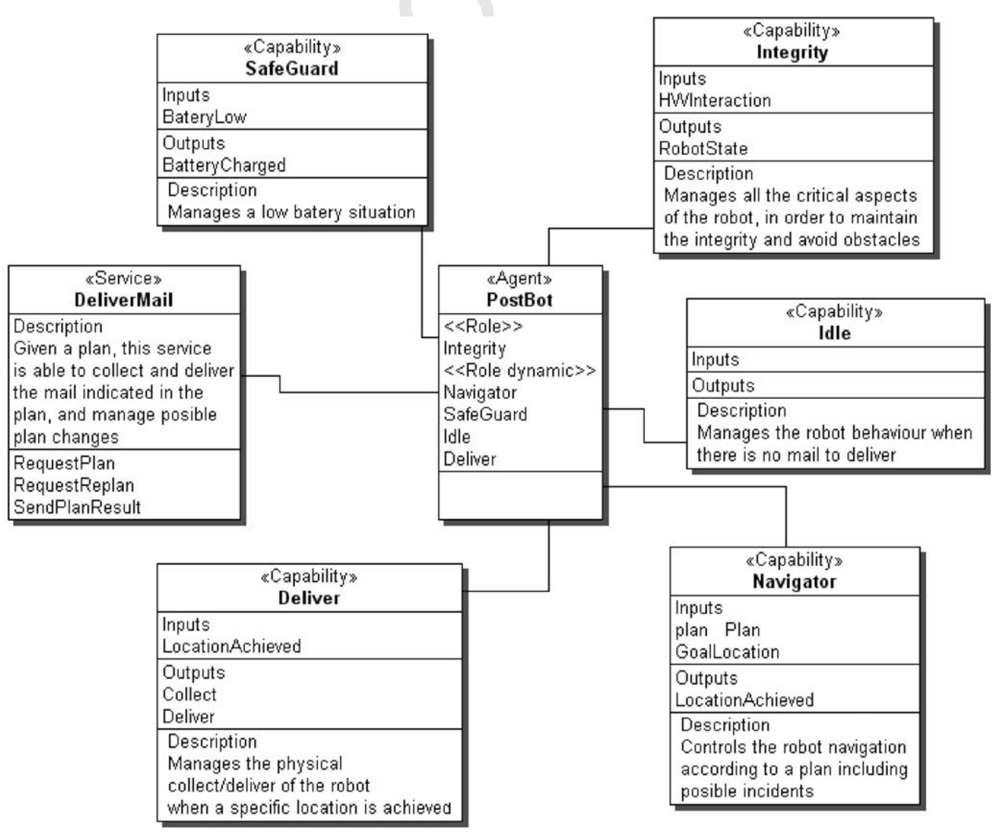

Fig. 9. AUML class diagram for the PostBot agent.

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031

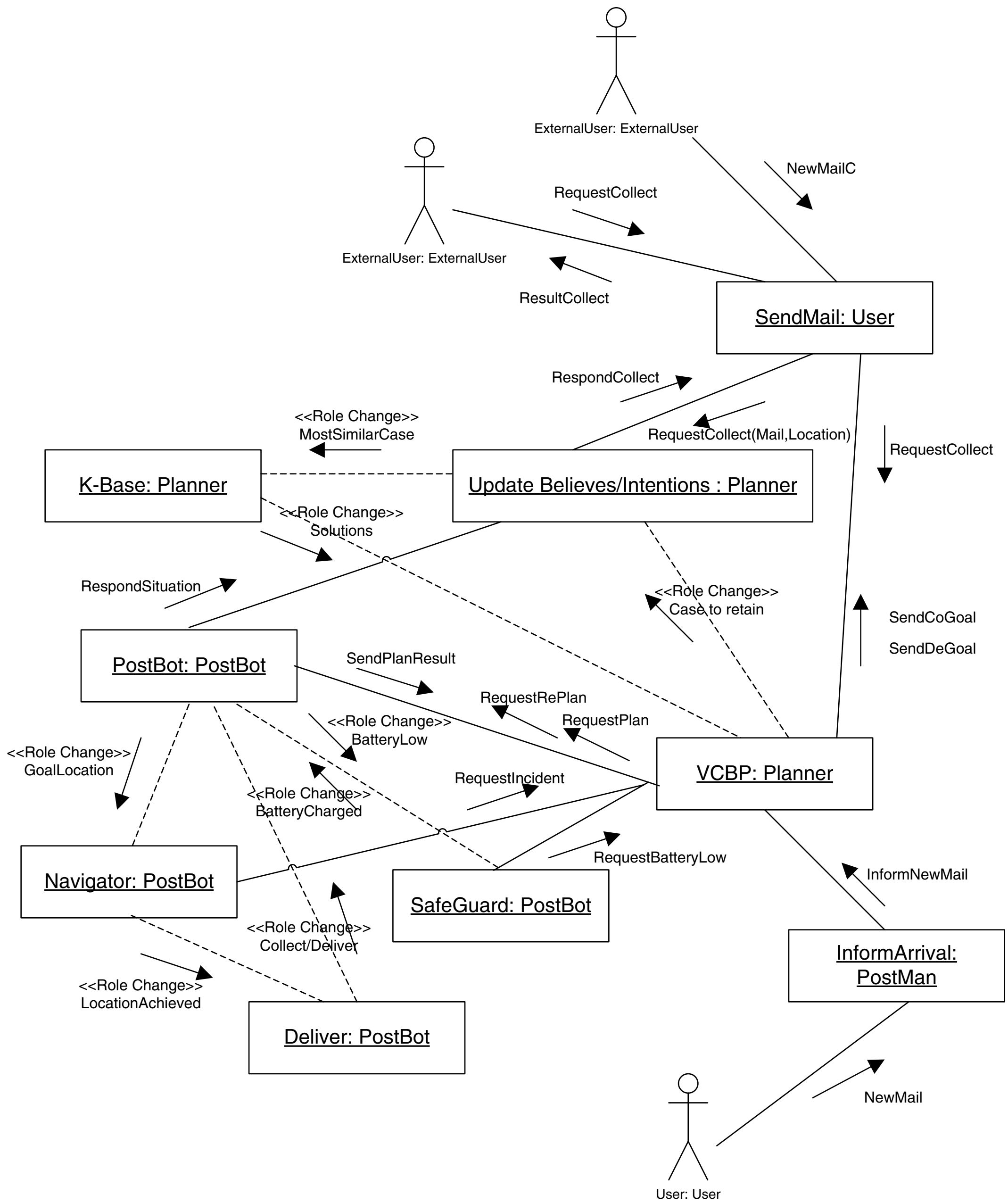

Fig. 11. Collaboration diagram. Postman agent reports the arrival of external mail.

Fig. 9 illustrates the class diagram for the PostBot agent. The architecture of the real-time bound agent ARTIS is used for the agent design. Therefore, the PostBot role is played by an ARTIS type agent. The agent offers a service of mail distribution and implements five capabilities. A robot agent is basically characterised by a critical objective that maintains the robot's integrity and that it will always be active. This aspect is covered by the Integrity capability. In this case the robot agent makes use of the resource Robot Hardware in order to access the state and act upon the activators of the robot. In addition, the robot has a Leisure capability when it does not have any delivery to make and only offers the service of postman. The Navigation

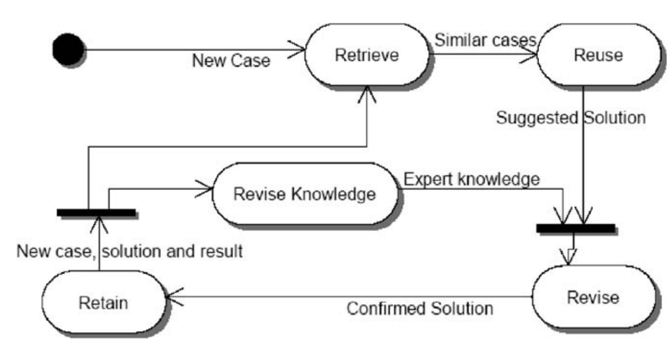

Fig. 12. Activity diagram for the planning activity.

capability is required by the robot when it has to carry out a complex maneuver in order to deliver or collect mail. The Distributor capability is activated when the robot is already at the destination point where it needs to physically collect or deliver a piece of mail. Lastly, the Emergency capability allows the robot to detect that the battery level is reaching minimum. In this event, the only objective of the agent is to return to the base in order to recharge the battery as soon as possible.

In real-time systems, it is very important for the communication and interaction to satisfy the possible time constraints. A series of communication protocols has been established among agents. For example, Fig. 10 illustrates the Replan Execution protocol that indicates the steps necessary for the communication between a PostBot agent and a Planner Agent when the PostBot agent detects an incident that occurs during the execution of a plan (an obstacle, etc.). In this case, the PostBot Agent makes a request to the Planner agent to replan and deliver a new plan. Fig. 10 also shows the roles used to make the interaction possible. The Planner agent responds by indicating whether it accepts, rejects, or fails to understand the request. Once the replanning process has been completed, the Planner agent communicates the result to the PostBot agent. This result could be a new plan that is delivered using an inform message, or it could be an error, which is indicated by a failure message.

The interactions that produce the system can be represented using AUML diagrams. In this case, collaboration diagrams are used even though sequence diagrams could be used without any problem. The interactions can be obtained from the interaction model. Fig. 11 shows a collaboration diagram that represents the interaction produced when a Postman agent reports the arrival of new external mail.

This figure shows the interactions that are produced in the multi-agent system when a user makes a request for it to carry out a delivery of internal mail. The user sends the request to the User agent which transfers the new task to the Planner agent. The Planner agent receives the order as a new case when it plays the Update Beliefs/Intentions role. In order to resolve the problem of the new case, the agent carries out the retrieval stage. First, the Planner agent asks the PostBot agent that is the most appropriate for the task. Then, it searches the case base memory for similar cases. Once the retrieval is completed, the Planner agent passes on to the reuse stage. To do this, the Planner agent changes its active role as a K-Base role and searches optimum solutions for the proposed case using the most similar retrieved case. Then a new change of role takes place, and the Planner agent takes on the VCBP role in which it applies variational calculus to find the most suitable solution for the problem from among the optimum solutions. In this role, the Planner agent is ordered to provide the plan to the distributor agent, to replan if necessary, and to carry out a review of the solution obtained.

Once the modifications has been carried out, the Planner agent goes to the learning stage. In order to do this the Planner agent changes role again to the Update Beliefs/Intentions role. This role stores the results obtained in the case base memory and learns from them in order to convert these results into knowledge.

When the Planner agent (in its role as VCBP) delivers the plan for mail delivery to PostBot, it awaits reports from the PostBot agent on the effectiveness of the plan. In this situation, a series of problems or events may occur that make it necessary to modify the delivery plan, or to replan. At the beginning, when the PostBot agent receives the plan, it will attempt to achieve its first objective by taking on the Navigation role. The PostBot agent assumes that there are no problems in its movements and that its capacity to meet all the critical requirements to maintain its integrity, such as avoiding obstacles, is always active. When the robot reaches the delivery/collection point, its physical delivery role will be activated (Deliver). Once the mail is delivered or collected, the Postbot agent will return and determine

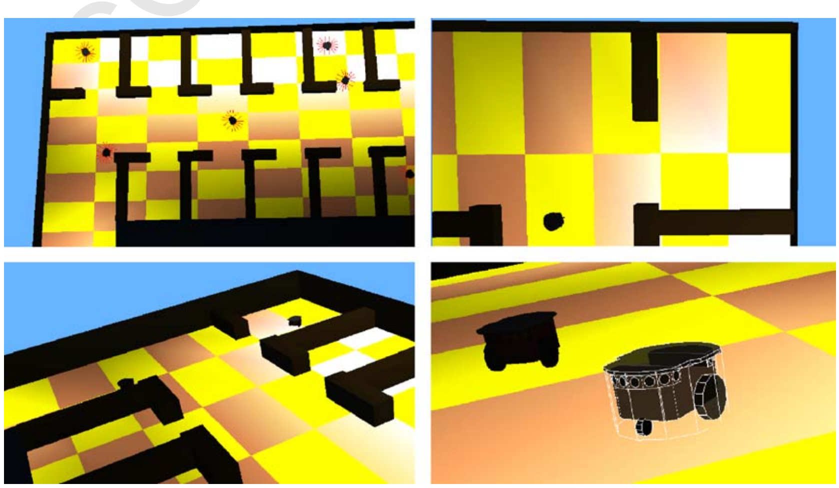

Fig. 13. Different views of the simulated system.

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031

its next objective, reporting on the success or failure of its action. Throughout this process, certain problems may occur that should be dealt with by the system:

- The PostBot agent reports that its battery is running low and that it must be recharged. This occurs when the SafeGuard role is activated. This role changes th critical objective to send the robot to the recharge point. This change requires starting a replanning process.
- The PostBot agent (in its Navigation role) reports that the robot has encountered some kind of unexpected obstacle that prevents it from making the delivery/collection.
- A Postman reports a new delivery of internal mail.
- A new user makes a new request for the delivery of internal mail.

In these situations, the Planning agent reacts by searching for a new plan that is the optimum and that will resolve

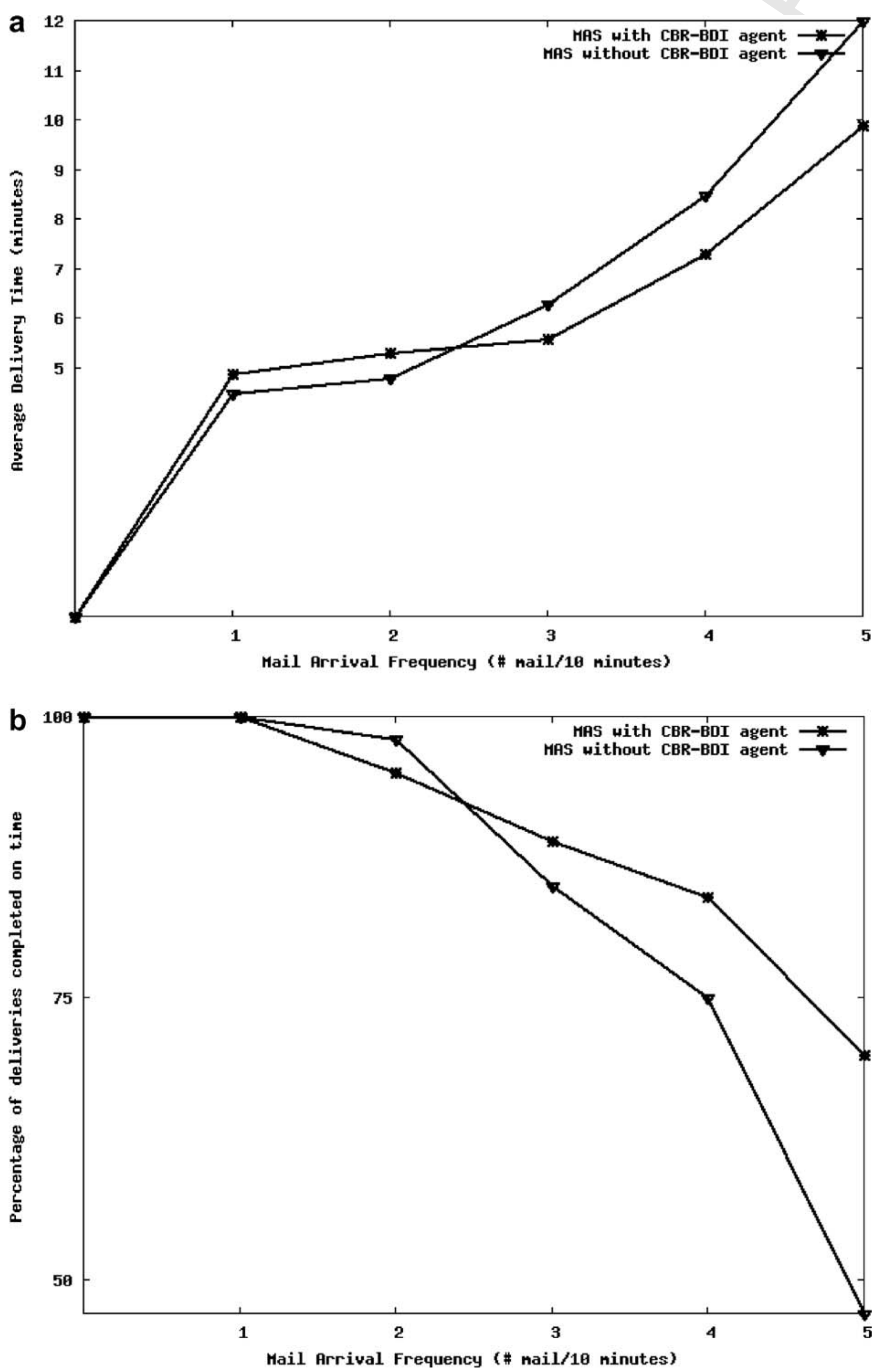

Fig. 14. (a) Average delivery time in the whole system and (b) percentage of deliveries completed on time when increasing the mail arrival frequency.

the problem in a suitable way. The Planner agent replans, changing the goals of the PostBot agent. It is necessary to point out that continuous replanning actions may be carried out, so that messages such as RequestIncident/RequestBatteryLow and RequestRePlan may be considered as a kind of loop.

Lastly, the activity diagrams are obtained in order to understand the behaviour of the agents. The activity diagram in Fig. 12 corresponds to the planning activity and illustrates how the Planner agent sets up a CBP cycle in order to obtain a plan.

5. Experimental results and conclusions

An experimental simulation prototype was implemented using the SIMBA platform on a mobile robot simulation

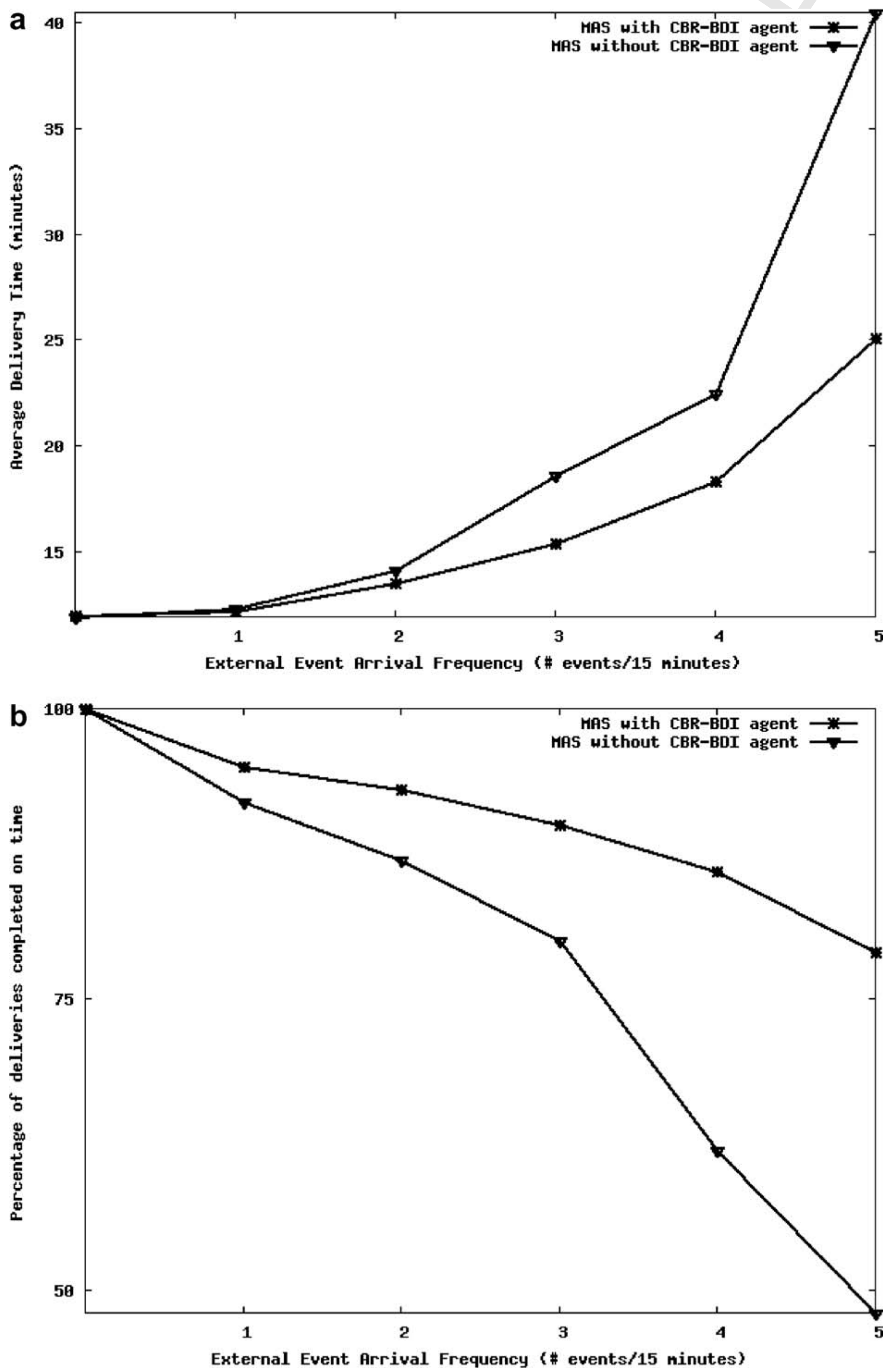

Fig. 15. (a) Average delivery time in the whole system and (b) percentage of deliveries completed on time when increasing the external event arrival frequency.

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031

software. Each PostBot was implemented as an ARTIS agent. The Planner agent has been developed as a CBP-BDI agent using JadeX. Finally, the rest of the agents were developed using the Jade development toolkit. The simulation software was *webots* which is developed by the Cyberbotics enterprise (http://www.cyberbotics.com/). Several simulation experiments were conducted to evaluate different aspects. In the simulation, the multi-agent architecture involved one Planner agent, five PostBot agents, and one PostMan, and one User agent.

A view of the simulated environment is shown in Fig. 13. The simulation presents a department formed by a set of offices connected through a single corridor. The UserAgent reports on new mail through a Jade-Leap agent using a PDA. Mail reception is carried out by the PostMan agent, which informs the PlannerAgent. After new mail or a package is received, the Planner agent decides to assign the new delivery to one of the five PostBot agents. Each mail or package distribution must be finalized before a maximum time.

The first set of experiments studied the performance of the system according to package or mail arrival frequency. The simulation prototype was tested by increasing this frequency incrementally and by testing two different parameters: average delivery time in the whole system and percentage of deliveries completed on time. Two sets of tests were simulated: one employed the planning agent system CBP-BDI and another worked without it. In the second set of tests, only a simple mail dispatcher was available to assign the mail randomly to each of the robots, while each robot carries out the orders in strict order of their arrival.

Fig. 14 illustrates the results obtained in one of the simulations carried out according to the parameters given. In the case of graphic (a), as the frequency of mail arrivals increased the average time of delivery, logically, also increased. Furthermore, better behaviour was observed when the CBP-BDI architecture was incorporated. The tests were carried out with a frequency of up to five letters every 10 minutes, since at greater frequencies, the system collapsed after a time. In graphic (b), there is evidence that the greater the frequency of arrival, the less the delivery time constraints were met. This behaviour was more pronounced when no CBP-BDI agent was available.

The second set of experiments was to study the system replanning behaviour when external events affect normal system behaviour, i.e., the cancellation of a mail delivery/collection. In order to carry this out, the simulation was tested introducing events that would cause a replanning in the system. The same parameters as in the previous experiment were measured. In this case, each PostBot agent began with a plan that incorporated five delivery/collection orders. We observed how each initial plan was carried out and how it changed when new events that made replanning necessary occurred.

In Fig. 15, the results demonstrate that when the frequency of replanning events increased, the average delivery time and the percentage of deliveries completed on time were affected. In the case of the measurements made from the average delivery time (graphic a), there was a noteworthy improvement in the behaviour of the system that incorporated the CBP-BDI agent due to its capacity for replanning. The results obtained for the percentage of

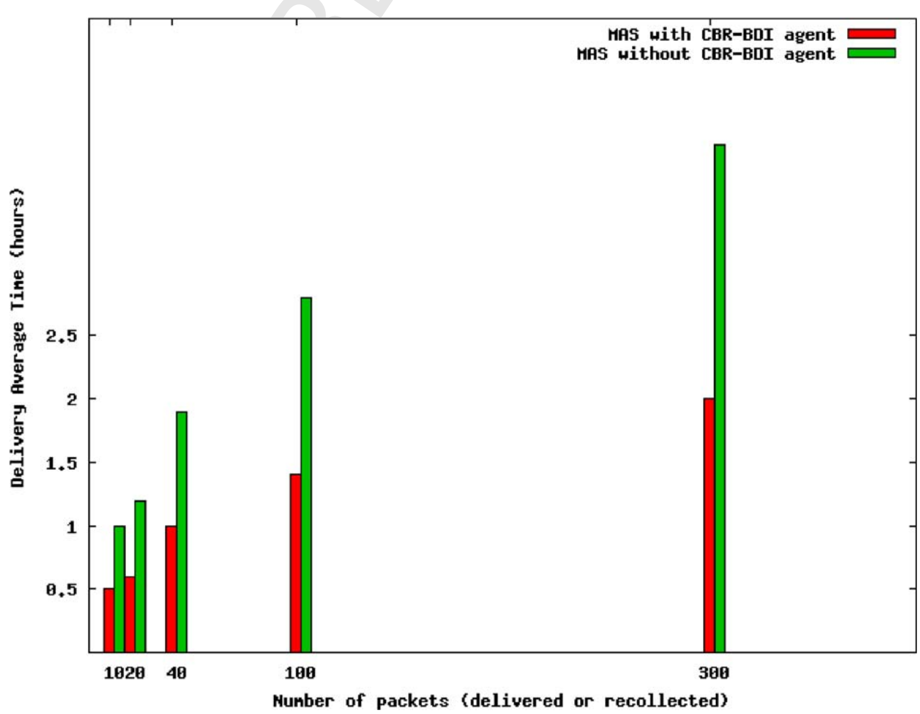

Fig. 16. Average time for delivery of packages with and without the collaboration of a CBP-BDI agent.

deliveries completed on time (graphic b) show that: without CBP-BDI agent, there is a sharp decrease in the percentage of success; with CBP-BDI agent, the success rate is over 80%.

Finally, Fig. 16 shows the time it took the PostBot agents (implemented with the ARTIS architecture) to deliver and collect *N* packages with and without the collaboration of CBP-BDI agents. Without this collaboration, the PostBot agents followed a pre-established route and they resolved any incidents without any replanning.

In summary, the main goal of this approach is to increase the flexibility of real-time system implementations, which has been achieved. This approach gives an extremely high degree of flexibility while at the same time retaining the time constraints needed in systems of this kind.

Finally, this paper has presented a flexible and efficient integration of high-level, multi-agent planning processes with real-time behaviours in a complex and dynamic environment. A multi-agent system that includes deliberative and pure reactive processes has been implemented using the SIMBA platform. This approach has been tested in the automated management simulation of internal and external mail in a department. The results are promising for deployment within a real scenario in the near future.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications, 7*(March), 39–59.

Bajo, J., & Corchado, J. M. (2005). Evaluation and monitoring of the air–sea interaction using a CBR-agents approach. In *Proceedings of the 6th international conference on case-based reasoning, ICCBR 2005, Chicago, IL, August 2005*. LNAI 3620 (pp. 50–62). Springer-Verlag, Berlin, Heidelberg. ISBN: 3-540-28174-6.

Barber, F., Botti, V., Onainda, E., & Crespo, A. (1994). Temporal reasoning in Reakt: an environment for real-time knowledge-based systems. *AI Communications, 7*(3), 175–202.

Bauer, B. (2001). UML Class diagrams revisited in the context of agent-based systems. In *Proceedings AOSE 2001*. Montreal: Springer.

Bauer, B., & Huget, M. P. (2003). FIPA modeling: agent class diagrams.

Beatson, R., & Light, W. (1997). Fast evaluation of radial basis functions. *Computational Mathematics and Applications, 24*(12), 7–20.

Bellman, R. E. (1957). Dynamic programming. Princeton, NJ: Princeton University Press.

Botti, V., Carrascosa, C., Julian, V., & Soler, J. (1999). Modelling agents in hard real-time environments. In *MAAMAW'99 Proceedings. LNAI* (1647, pp. 63–76). Springer-Verlag.

Bratman, M. (1987). Intention, plans and practical reason. Cambridge: Harvard U.P..

Carbonell, J. (1983). Learning by analogy: formulating and generalizing plans from past experience. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.). *Machine learning: An artificial intelligence approach.* Cambridge, MA (pp. 137–162).

Carrascosa, C., Rebollo, M., Soler, J., Julian, V., & Botti, V., (2003). SIMBA architecture for social real-time domains. In *EUMAS 2003: the first European workshop on multi-agent systems*.

Carrascosa, C., Rebollo, M., Julian, V., & Botti, V. (2003). Deliberative server for real-time agents. In *3rd International central and eastern European conference on multi-agent systems (CEEMAS 2003), Vol 2691* (pp. 485–496).

Corchado, J. M., & Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems, 18*(12), 1227–1241, ISSN 0884-8173.

Duchon, J. (1977). Spline minimizing rotation-invariant seminorms in Sobolev spaces. In: W. Schempp, & K. Zeller (Eds.), *Constructing theory of functions of several variables, Vol. 571* (pp. 85–100).

Garcia-Fornes, A., Terrasa, A., Botti, V., & Crespo, A. (1997). Analyzing the schedulability of hard real-time artificial intelligence systems. *EAAI*, 369–377.

Glez-Bedia, M., & Corchado, J. M. (2002). Analytical model for constructing deliberative agents. *Engineering Intelligent Systems, 3*, 173–185.

Goldman, R. P., Musliner, D. J., & Krebsbach, K. D. (2001). Managing online self-adaptation in real-time environments. In *Proceedings of second international workshop on self adaptive software, Balatonfured, Hungary*.

Graham, John R. (2001). Real-time scheduling in distributed multi-agent systems. *Ph.D. thesis*. Department of Computer and Information Science, University of Delaware.

Hammond, K. (1989). Case-base planning: viewing planning as a memory task. New York: Academic Press.

Hegland, M., Roberts, S., & Altas, I. (1997). Finite element thin plate splines for surface fitting. *Computational techniques and applications* (CTAC97). World Scientific.

Jost, J., & Li-Jost, X. (1998). Calculus of variations. UK: Cambridge University Press.

Lee, J. M. (1997). *Riemannian manifolds. An introduction to curvature*. New York: Springer-Verlag, Inc..

Milnor, J. (1973). *Morse theory*. *Annals of mathematical studies*. Princeton University Press.

Odell, J., & Huget, M. P. (2003). FIPA modeling: interaction diagrams.

Odell, J., Levy, R., & Nodine, M. (2004). FIPA modeling TC: agent class superstructure metamodel. *FIPA meeting and interim work*.

Pollack, M. E. (1992). The uses of plans. *Artificial Intelligence, 57*, 43–68.

Rao, A. S., & Georgeff, M. P. (1995). BDI agents: from theory to practice. In *First international conference on multi-agent systems (ICMAS-95). San Francisco, USA, June*.

Reuter, P., Tobor, I., Schlick, C., & Dedieu, S., (2003). Point-based modelling and rendering using Radial basis functions. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Austalasia and South East Asia (Graphite 2003)* (pp. 111–118).

Russell, S., & Norvig, P. (2003). *Artificial intelligence: a modern approach* (second ed.). Prentice-Hall International Editions.

Schutz, B. F. (1993). *Geometrical methods of mathematical physics*. Cambridge U.P.

Soler, J., Julian, V., Rebollo, M., Carrascosa, C., & Botti, V. (2002). Towards a real-time multi-agent system architecture. In *1st International workshop on challenges in open agent systems, Bologna, Italy*.

Terrasa, A., Garcia-Fornes, A., & Botti, V. (2002). Flexible real-time linux. *Real-Time Systems Journal, 2*, 149–170.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review, 10*(2), 115–152.

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems, 3*(3), 285–312.

Please cite this article as: C. Carrascosa et al., Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications (2006), doi:10.1016/j.eswa.2006.08.031