Guidelines to apply CBR in Real-Time Multi-Agent Systems

Martí Navarro and Stella Heras and Vicente Julián

***Abstract*—In real-time Multi-Agent Systems, Real-Time Agents merge intelligent deliberative techniques with real-time reactive actions in a distributed environment. CBR has been successfully applied in Multi-Agent Systems as deliberative mechanism for agents. However, in the case of Real-Time Multi-Agent Systems the temporal restrictions of their Real-Time Agents make their deliberation process to be temporally bounded. Therefore, this paper presents a guide to temporally bound the CBR to adapt it to be used as deliberative mechanism for Real-Time Agents.**

***Index Terms*—Real-Time Multi-Agent Systems, Case-Based Reasoning.**

I. Introduction

The need for developing software solutions applied to complex, non-dynamic and frequently, non-completely specified environments, has contributed to the confluence of two important research areas, the Artificial Intelligence (AI) and the Real-Time System (RTS). Inside the Artificial Intelligence framework, Multi-Agent Systems paradigm (MAS) represents an appropriate approach for solving inherently distributed problems. The work presented in this paper is planned over the existent relationship between MAS and RTS. This work covers the problem of Real-Time Agents (RTAs), which merge intelligent deliberative techniques with real-time reactive actions in a distributed environment.

A Real-Time Artificial Intelligence System (RTAIS) is a system that must accomplish critical processes under a dynamic environment with temporal restrictions by using AI techniques. Here, anytime algorithms [4] and approximate processing [7] are the most promising algorithms. One line of research in RTAI has been to build applications or architectures that embody real-time concerns in many components[7], such as Guardian [9], Phoenix [10], CIRCA/ SA-CIRCA [8], [17] and ARTIS [6], [2]. An appropriated agent for real-time environments must accomplish its goals, responsibilities and tasks with the added difficulty of temporal restrictions. Thus, an RTA can be defined as an agent with temporal restrictions in, at least, one of its responsibilities. The RTA may have its interactions bounded, and this modification will affect all the communication processes in the MAS where the RTA is located.

The main problem in the architecture of an RTA concerns the deliberation process. This process commonly uses AI techniques as problem-solving methods to compute more intelligent actions. However, the temporal restrictions of RTAs give rise to the necessity of providing techniques that allow their response times to be bounded. These techniques are based on RTAIS techniques [7]. In addition, in an RTA an efficient integration of high-level, deliberative planning processes within reactive processes is necessary. These complex deliberative processes, which allow the agent to reason, to adapt and learn, are unbounded and it is difficult to integrate them in real-time systems. However, their main drawback lies in finding a mechanism that permits their efficient and temporal bounded execution.

In view of the successful applications reported in the literature (see Section III), we propose a model where RTAs use a CBR method as deliberative mechanism to take decisions. However, the execution of this CBR method must be bounded in order to observe the RTA temporal restrictions. Thus, this paper presents a guide to temporally bound the CBR cycle to adapt it to be used as a deliberative mechanism for RTAs. The paper is structure as follow: Section II introduces the concept of Real-Time Agent; Section III reviews related successful applications of the CBR method in MAS; Section IV presents a guide with the principal facts to take into account to temporally bound the CBR cycle and finally, some conclusions are shown in Section V.

II. Real-Time Agent

A Real-Time Agent (RTA) is an agent composed of a series of tasks, some of which have temporal constraints [11]. In these agents, it is also necessary to take into account the temporal correctness, which is expressed by means of a set of temporal restrictions that are imposed by the environment. The RTA must, therefore, ensure the fulfilment of such temporal restrictions. By extension, a Real-Time Multi-Agent System (RTMAS) is a multi-agent system with at least one RTA [11]. Systems of this type require the inclusion of temporal representation in the communication process, management of a unique global time, and the use of real-time communication [23].

It is well-known that a typical real-time system is made up of a set of tasks characterized by a deadline, a period, a worst-case execution time and an assigned priority. These restrictions in the system functionality affect the features of an agent that needs to be modelled as a real-time system. The main problem is that if its tasks are not temporally bounded properly, it is not possible to guarantee the fulfilment of the tasks before a deadline is expired and to schedule a plan with these tasks.

The reasoning process of the RTA must be temporally bounded to allow it to perform the tasks for deciding the

Martí Navarro is with Polytechnic University of Valencia. E-mail: mavarro@dsic.upv.es
Stella Heras is with University of Polytechnic University of Valencia. E-mail: sheras@dsic.upv.es
Vicente Julián is with University of Polytechnic University of Valencia. E-mail: vinglada@dsic.upv.es

strategy to reach its objectives. In this way, the RTA will be able to determine whether it has enough time to deliberate and to take into account the temporal cost of its cognitive task when it plans the execution of new tasks. Next sections review CBR applications to MAS and propose a temporally bounded CBR method as deliberative mechanism for the cognitive task of the agent.

III. CBR AS DELIBERATIVE MECHANISM FOR AGENTS

In the AI research, the combination of several AI techniques to cope with specific functionalities in hybrid systems has a long history of successful applications. A CBR system provides agent-based systems with the ability to reason and learn autonomously from the experience of agents. These systems propose solutions for solving a current problem by reusing or adapting other solutions that were applied in similar previous problems. With this aim, the system has a case-base that stores its knowledge about past problems together with the solution applied in each case. The most common architecture of a CBR system consists of four phases [1]: the first one is the *Retrieval* phase, where the most similar cases are retrieved from the case-base; then, in the *Reuse* phase, those cases are reused to try to solve the new problem at hand; after that, in the *Revise* phase the solution achieved is revised and adapted to fit the current problem and; finally, in the *Retain* phase, the new case is stored in the case-base and hence, the system learns from new experiences. The integration of CBR systems and MAS has been studied following many approaches. Therefore, the literature of this area reports research on systems that integrate a CBR engine as a part of the system itself [12], other MAS that provide some or each of their agents with CBR capabilities or even, the development of BDI agents following a CBR methodology [3]. This section is focused on the review of the second approach, CBR applied to MAS, since it fits the scope of our paper.

Since the 90's, the synergies between MAS and CBR are many, although the approaches differ. One early approach was the development of multi-agent CBR systems, which are MAS with cooperative agents characterized by the distribution of their case-bases and/or certain phases of the CBR cycle between them [13], [16], [20]. The main effort in this research area is focused on the policies that agents follow to manage the CBR cycle.

The application of CBR to manage argumentation in MAS is a different and more recent approach that has produced important contributions both in the areas of AI and argumentation theory. In this field, important works are the case-based negotiation model for reflective agents proposed in [22], the new case-based selection model ProCLAIM [24], which extended the architecture of the decision support MAS for the organ donation CARREL+ and the Argumentation Based Multi-Agent Learning (AMAL) framework [19], which features a set of agents that try to solve a classification problem by aggregating their expert knowledge.

Furthermore, an area where the integration between CBR and agent techniques has produced a huge amount of successful applications is the robot navigation domain. An important contribution was a case-based model for managing the ROBOCATS system [15], playing in the Robocup league. Also, the RUPART system [5], which features a hybrid planner for a mobile robot that delivers mail in real-time. Another application of CBR to manage autonomous navigation tasks was a system for the automatic selection and modification of assemblage parameters proposed in [14]. Finally, an important research that applies CBR to MAS with mobile robots modelled team playing behaviour in the robot soccer domain by using CBR [21].

The cited above are outstanding examples of systems that join research efforts and results of both CBR and MAS, but they are only a sample reported in the literature of this prolific area. However, few of these systems cope with the problem of applying CBR as deliberative engine for agents in MAS with real-time constraints. In this case, the case-based reasoning cycle must observe temporal restrictions. The next section tackles this challenge and provides solutions to deal with it.

IV. TEMPORALLY-BOUNDED CBR

CBR systems are highly dependent on their application domain. Therefore, designing a general CBR model that might be suitable for any type of real-time domain (hard or soft) is, to date, unattainable. In real-time environments, the CBR phases must be temporally bounded to ensure that solutions are produced on time. In this section, we present some guidelines with the minimum requirements to be taken into account to implement a CBR method in real-time environments.

The design decision about the data structure of the case-base and the different algorithms that implement each phase are important factors to determine the execution time of the CBR cycle. The number of cases in the case-base is another parameter that affects the temporal cost of the retrieval and retain phases. Thus, a maximum number of cases must be defined by the designer. Note that, usually, the temporal cost of the algorithms that implement these phases depend on this number.

For instance, let us assume that the designer chooses a hash table as data structure for the case-base. This table is a data structure that associates keys to concrete values. Search is the main operation that it supports in an efficient way: it allows the access to elements (e.g. phone and address) by using a hash function to transform a generated key (e.g. owner name or account) to a hash number that is used to locate the desired value. The average time to make searches in hash tables is constant and defined as $O(n)$ in the worst case. Therefore, if the cases are stored as entries in a hash table, the maximum time to look for a case depends on the number of cases in the table (i.e. $O(number\_cases)$). Similarly, if the case-base is structured as an auto-balanced binary tree the search time in the case-base in the worst case would be $O(log(n))$.

In this research, we propose a modification of the classic CBR cycle (Figure 1) to adapt it to be applied in real-time domains. Figure 2 shows a graphical representation of our approach. First, we group the four reasoning phases that implement the cognitive task of the real-time agent in two stages: the learning stage, which consists of the revise and

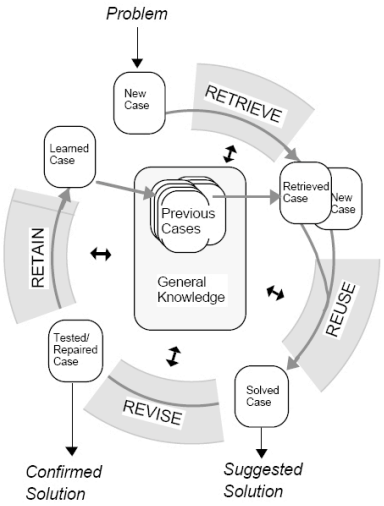

Fig. 1. Classic CBR cycle.

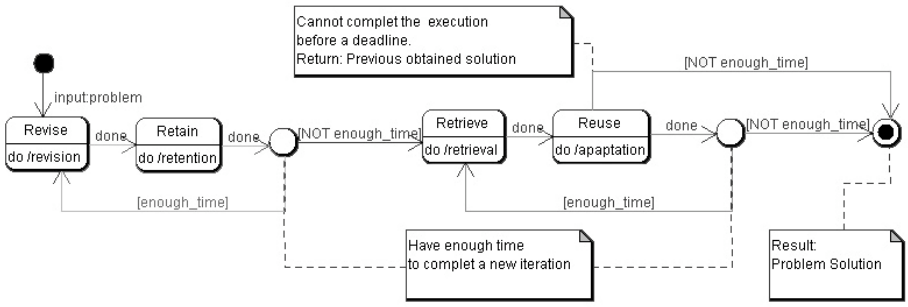

Fig. 2. Temporally Bounded CBR cycle.

retain phases and the deliberative stage, which includes the retrieve and reuse phases. Both phases will have scheduled their own execution times. In this way, the designer can choose between assigning more time to the deliberative stage (and hence, to design more 'intelligent' agents) or else, keeping more time for the learning stage (and thus, to design agents that are more sensible to updates).

Following, the operation of our Time Bounded CBR cycle (TB-CBR) is explained. Firstly, the main difference that can be observed between the classical CBR cycle and the TB-CBR cycle is the starting phase. Our real-time application domain and the restricted size of the case-base gives rise to the need of keeping the case-base as updated as possible. Commonly, recent changes in the case-base will affect the potential solution that the CBR cycle is able to provide for a current problem. Therefore, the TB-CBR cycle starts by the learning stage, checking if there are previous cases waiting for being revised and possible stored in the case-base. In our model, the solutions provided at the end of the deliberative stage will be stored in a solution list while a feedback about their utility is received. When each new CBR cycle begins, this list is accessed and while there is enough time, the learning stage of those cases whose solution feedback has been recently received is executed. In case the list is empty, this process is omitted.

After that, the deliberative stage is executed. Thus, the retrieval algorithm is used to search the case-base and retrieve a case that is similar to the current case (i.e. the one that characterises the problem to solve). Each time a similar case is found, it is sent to the reuse phase where it is transformed to a suitable solution for the current problem by using a reuse algorithm. Therefore, at the end of each iteration of the deliberative stage, the TB-CBR method is able to provide a solution for the problem at hand, although this solution can be improved in next iterations as long as the deliberative stage has enough time to perform them.

Hence, the temporal cost of executing the cognitive task is greater than or equal to the sum of the execution times of the learning and deliberative stages (as shown in equation 1):

$$\begin{aligned} t_{cognitiveTask} &\geq t_{learning} + t_{deliberative} \\ t_{learning} &\geq (t_{revise} + t_{retain}) * n \\ t_{deliberative} &\geq (t_{retrieve} + t_{reuse}) * m \end{aligned} \quad (1)$$

where $t_{learning}$ and $t_{deliberative}$ are the total execution time of the learning and deliberative stages; t_x is the execution time of the phase x and *n* and *m* are the number of iterations of the learning and deliberative stages respectively. The requirements

needed to temporally bound each phase of the TB-CBR cycle are explained below. In order to bound the temporal cost of the algorithms that implement these phases and to ensure an adequate temporal control of them, the execution time of these algorithms is approximated by its *worst-case execution time* (WCET). The WCET sets a maximum threshold for the temporal execution of each algorithm and thus, this prevents the temporal constraints of the system to be broken.

A. Revise Phase

In order to keep the case-base as up to date as possible, the revise phase is performed first. During this phase, the accuracy of the final solutions obtained in previous executions of the TB-CBR cycle is checked. The revision algorithm (i.e. $f_{revision}$) only checks one solution per iteration, fixing the potential problems that it had in case of erroneous results. The outcome of this phase is used to update the case-base. Thus, the maximum temporal cost of this phase is bounded by the WCET of the revision algorithm:

$$t_{revise} = WCET(f_{revision}(solution)) \tag{2}$$

Note that, in order to guarantee a known maximum execution time, this checking must be performed automatically by the computer without human interference. This WCET does not depend on the number of stored solutions or the number of cases in the case-base and again, must be determined by the designer of the algorithm.

B. Retain phase

In this phase it is decided whether a checked solution must be added as a new case in the case-base. Here, keeping the maximum size of the case-base is crucial, since the temporal cost of most retention algorithms (i.e. $f_{retention}$) depends on this size. If there is a case in the case-base that is similar enough to the current case, this case (its problem description and solution) is updated if necessary. On the contrary, if there is not a case that represents the problem solved, a new case is created and added to the case-base. In order to keep the maximum size of the case-base, this could entail removing an old case from it. This decision should be taken by the retention algorithm. Nevertheless, the maximum temporal cost that the retain phase needs to execute one iteration is the retention algorithm WCET.

$$t_{retain} = WCET(f_{retention}(solution, CaseBase)) \tag{3}$$

C. Retrieve Phase

In the retrieve phase, the retrieval algorithm (i.e. $f_{retrieval}$) is executed to find a case that is similar to the current problem (i.e. $currentCase$) in the case-base. Since WCET depends on the structure of the case-base and its number of cases, the designer must calculate this WCET and use this time to estimate the necessary time to execute an iteration of the retrieval algorithm.

$$t_{retrieve} = WCET(f_{retrieval}(currentCase, CaseBase)) \tag{4}$$

Each execution of the retrieval algorithm will provide a unique case similar to the current problem (if it exists in the case-base). This result is used as input for the reuse phase. However, in next iterations of the deliberative stage more similar cases can be retrieved with the intention to provide a more accurate solution for the problem.

D. Reuse Phase

In this phase, the cases obtained from the retrieve phase are adapted to use them as a potential solution for the current problem. These cases are stored in a list of selected cases (i.e. $cases_{list}$). Each time the reuse phase is launched, the adaptation algorithm (i.e. $f_{adaptation}$) searches this list and produces a solution by adapting a single case or a set of cases to fit the context of the current problem to solve. Therefore, the execution time of this algorithm depends on the number of cases that the algorithm is working with.

$$t_{reuse} = \begin{cases} WCET(f_{adaptation}(firstCase)) \\ f_{adaptation}(listOfCases) \end{cases} \tag{5}$$

To guarantee that the RTA assigns enough time to perform the cognitive task and provides at least one solution, the designer must know the WCET to execute one iteration of the adaptation algorithm (i.e. $f_{adaptation}(firstCase)$). In order to control the execution time of the adaptation algorithm in subsequent iterations (i.e. $f_{adaptation}(listOfCases)$), the RTA must be able to stop the execution of the algorithm in case that it realises that the assigned time to complete the deliberative stage will be overcome. Then, the RTA provides the best solution among the solutions completed in previous iterations. This solution is stored in a list of solutions for being verified in the learning stage.

V. Conclusion

The main contribution of this article is to set some guidelines to develop a CBR method for real-time systems. In this type of systems, timing requirements must be previously known in order to guarantee the correctness of the system. However, since the execution time of each CBR method depends on the specific algorithms that have been used to implement the CBR cycle, a general estimation cannot be made. Therefore, this work provides the designer of the system with a new approach for the CBR cycle, called TB-CBR, which eases the process of bounding each phase of the CBR method. This method implements the CBR cycle in two stages: the deliberative stage, whose execution is mandatory; and the learning stage, whose execution can be optional depending on the temporal requirements.

This TB-CBR proposal has been implemented and tested in an example that consists in a system that manages the mail in a department plant by using mobile robots [18]. In this example, robots deliberate to know whether they have enough time to deliver mail. This deliberation is implemented by using a TB-CBR method.

ACKNOWLEDGMENT

This work was partially supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022 and by the Spanish government and GVA funds under TIN2006- 14630-C0301 and PROMETEO/2008/051 projects.

REFERENCES

[1] A. Aamodt and E. Plaza, *Case-based reasoning; Foundational issues, methodological variations, and system approaches*. AI Communications, vol. 7, no. 1, pp. 39-59, 1994.

[2] C. Carrascosa and A. Terrasa and A. García-Fornes and A. Espinosa and V. Botti, *A Meta-Reasoning Model for Hard Real-Time Agents*. CAEPIA, vol. 4177, pp. 42-51, 2005.

[3] J. M. Corchado and A. Pellicer, *Development of CBR-BDI Agents*. International Journal of Computer Science and Applications, vol. 2, no. 1, pp. 25-32, 2005.

[4] T. Dean and M. Boddy, *An analysis of time-dependent planning*. Proc. of the 7th National Conference on Artificial Intelligence, pp. 49-54, 1988.

[5] S. E. Fox and P. Anderson-Sprecher, *Robot Navigation: Using Integrated Retrieval of Behaviors and Routes*. Proc. of FLAIRS Conference, pp. 346-351, 2006.

[6] A. García-Fornes, *ARTIS: Un modelo y una arquitectura para sistemas de tiempo real inteligentes*. Ph.D. Dissertation, DSIC, U. Politecnica Valencia, 1996.

[7] A. Garvey and V. Lesser, *A survey of research in deliberative Real-Time Artificial Intelligence*. The Journal of Real-Time Systems, vol. 6, pp. 317-347, 1994.

[8] R. P. Goldman and D. J. Musliner and K. D Krebsbach, *Managing Online Self-Adaptation in Real-Time Environments*. Proc. of 2nd Int. Workshop on Self Adaptive Software, 2001.

[9] B. Hayes-Roth and R. Washington and D. Ash and A. Collinot and A. Vina and A. Seiver, *Guardian: A prototype intensive-care monitoring agent*. Artificial Intelligence in Medicine, vol. 4, pp. 165-185, 1992.

[10] A. E. Howe and D. M. Hart and P. R. Cohen, *Addressing real-time constraints in the design of autonomous agents*. The Journal of Real-Time Systems, vol. 2, pp. 81-97, 1990.

[11] V. J. Julián and V. Botti, *Developing real-time multi-agent systems*. ICAE, vol. 11, no. 2, pp. 150-165, 2004.

[12] N. Karacapilidis and D. Papadias, *Computer supported argumentation and collaborative decision-making: the HERMES system*. Information Systems, vol. 26, no. 4, pp. 259-77, 2001.

[13] D. B. Leake and R. Sooriamurthi, *When Two Case Bases Are Better Than One: Exploiting Multiple Case Bases*. Proceedings of ICCBR, 2001.

[14] M. Likhachev and M. Kaess and R. C. Arkin, *Learning Behavioral Parameterization Using Spatio-Temporal Case-Based Reasoning*. Proc. of ICRA, vol. 2, pp. 1282-1289, 2002.

[15] C. Marling and M. Tomko and M. Gillen and D. Alexander and D. Chelberg, *Case-based reasoning for planning and world modeling in the robocup small size league*. Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments, IJCAI, 2003.

[16] L. Mc Ginty and B. Smyth, *Collaborative Case-Based Reasoning: Applications in Personalised Route Planning*. Proceedings of ICCBR, pp. 362-376, 2001.

[17] D. J. Musliner and J. A Hendler and A. K. Agrawala and E. H. Durfee and J. K. Strosnider and C. J. Paul, *The Challenge of Real-Time in AI*. IEEE Computer, January, pp. 58-66, 1995.

[18] M. Navarro and S. Heras and V. Julián, *Ensuring Time in Real-Time Commitments*. In Proc. of IBERAMIA, pp. 183-192, 2008.

[19] S. Ontañon and E. Plaza, *Learning and Joint Deliberation through Argumentation in Multi-Agent Systems*. Proc. of AAMAS, 2007.

[20] E. Plaza and J. L. Arcos and F. Martn, *Cooperative Case-Based Reasoning*. LNAI, vol. 1221, pp.180-201, 1997.

[21] R. Ros and R. López de Mántaras and J. L. Arcos and M. Veloso, *Team Playing Behavior in Robot Soccer: A Case-Based Approach*. Proc. of ICCBR, vol. 4626, pp. 46-60, 2007.

[22] L. K. Soh and C. Tsatsoulis, *A Real-Time Negotiation Model and a Multi-Agent Sensor Network Implementation*. AAMAS, vol. 11, no. 3, pp. 215-271, 2005.

[23] J. Soler and V. Julián and A. García-Fornes and V. Botti, *Real-Time Extensions in Multi-agent Communication*. LNAI 3040, pp. 468-477, 2003.

[24] P. Tolchinsky and S. Modgil and U. Cortés and M. Sánchez-Marré, *CBR and Argument Schemes for Collaborative Decision Making*. Proc. COMMA, vol. 144, pp. 71-82, 2006.