Increasing the Autonomy of Deliberative Agents with a Case-based Reasoning System

J. M. Corchado
Departamento de Informática y Automática
University of Salamanca
Plaza de la Merced s/n,
37008, Salamanca, Spain
Email: corchado@gugu.usal.es
Http://gsii.usal.es/~corchado/

R. Laza, L. Borrajo, J. C. Yañes and M. Valiño
Departamento de Informática
University of Vigo Campus As Lagoas, s/n,
32004, Ourense, Spain

Abstract

This paper shows how deliberative agents can be built by means of a case-based reasoning system. The concept of deliberative agent is introduced and the case-based reasoning model is presented. Once the advantages and disadvantages of such agents have been discussed, it will be shown how to solve some of their inconveniences, especially those related to their implementation and adaptation. The World Wide Web has emerged as one of the most popular vehicle for disseminating and sharing information through computer networks; a distributed agent-based solution for e-business, in which such agents have been used, is also presented and evaluated in this paper.

1 Introduction

In most computing systems, all the executed actions are previously planned and encoded by a programmer. However, in our present-day world, where technological evolution is fast and constant, it is necessary to build up systems with the capacity for adaptation and provided with mechanisms that allow them to decide what to do according to their objectives. Such systems are known as agents [Wo99]. This paper shows how to build deliberative agents, using a case-based reasoning (CBR) system. The proposed method facilitates the automation of their construction and provides them with the capacity of learning and therefore of autonomy.

Agents must be able to reply to events, which take place in their environment, to take the initiative according to their goals, to interact with other agents (even human), and to use past experiences to achieve present goals. There are different types of agents and they can be classified in different ways [WJ94]. One way is the so-called deliberative agents with a BDI architecture, which have mental attitudes of Beliefs, Desires and Intentions. In addition, they have the capacity to decide what to do and how to get it according to their attitudes [Wo99] [Je92] [RG91].

As mentioned above, deliberative agents, with a BDI architecture, are composed of beliefs, desires and intentions. The beliefs represent their information state, what the agents know about themselves and their environment. The desires are their motivation state, what the agents are trying to achieve, and the intentions represent the agents' deliberative states. Intentions are sequences of actions; they can be identified as plans. These mental attitudes determine the agent's behaviour and are critical to attain proper performance when the information about the problem is scarce [Br87] [KG91]. A BDI architecture has the advantage that it is intuitive, and it is relatively simple to identify the process of decision-making and how to perform it. Furthermore, the notions of belief, desires and intentions are easy to understand. On the other hand, its main drawback lies in finding a mechanism that permits its efficient implementation. The formalisation and implementation of BDI agents constitutes the research of many scientists [CL90] [Je92] [Ki94] [RG91] [GL86] [Sh93]. Some of them criticise the necessity of studying multi-modal logic for the formalisation and construction of such agents, because they have not been completely axiomatised and they are not computationally efficient. Rao and Georgeff [RG95] state that the problem lies in the big distance between the powerful logic for BDI systems and practical systems. Another problem is that this type of agent is not able to learn, a necessary requirement for them since they have to be constantly adding, modifying or eliminating beliefs, desires and intentions. It would be convenient to have a reasoning mechanism which would enable the agent to learn and adapt in real time i.e. as the computer program is executing rather than have to recompile such an agent whenever the environment changes.

This paper shows how a BDI agent implemented using a case-based reasoning system can substantially solve the two problems previously mentioned. Implementing agents in the form of CBR systems also facilitates their learning and adaptation. Among the different disciplines of cognitive science, cognitive psychology has widely shown the importance of learning from experience [CS90]. If the proper correspondence between the three mental attitudes of BDI agents and the information manipulated by a case-based reasoning system is established, an agent with beliefs, desires, intentions and a learning capacity will be obtained. Although the relationship between agents

and CBR systems has been investigated by other researchers [FG94] [Na96] [MPA99] [BW98] [WL98] [Ol99], we propose an approach, whose main characteristic is its direct mapping between the agent conceptualisation and its implementation, in the form of a CBR system.

This paper reviews first the concept of case-based reasoning system. Section 3 presents the proposed model, in which a CBR system is used to operate the mental attitudes of a deliberative agent. This section also shows the relationship between the BDI agents and the CBR systems. The e-business application constructed with the help of the agent conceptualisation introduced in this paper is then presented. Finally the agent and the e-business system are evaluated and some conclusions are exposed.

2 Case-based Reasoning Systems

Case-based reasoning (CBR) is used to solve new problems by adapting solutions that were used to solve previous similar problems [RS89]. To reach this objective, CBRs are based on the knowledge stored in their memory, in the form of cases or problems. Figure 1 shows the reasoning cycle of a typical CBR system that includes four steps that are cyclically carried out in a sequenced way: retrieve, reuse, revise, and retain [AP94] [WM94]. During this process the memory can change and new cases may appear. Due to that reason a sub-phase within the retain phase is inclued; in this sub-phase the Expert's knowledge is revised.

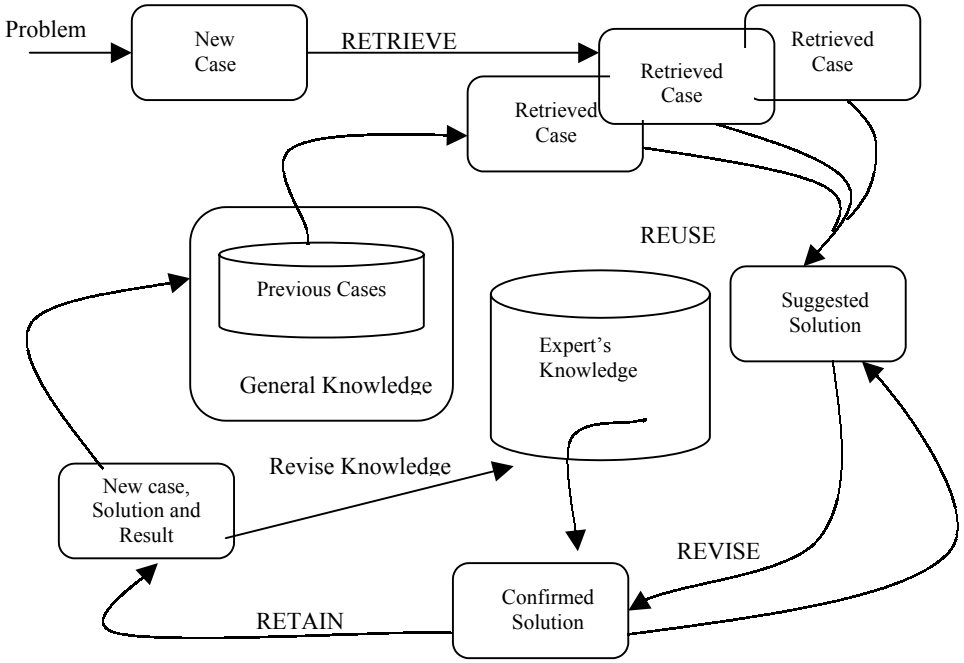

Figure 1: CBR Cycle of Life.

Each of the reasoning steps of a CBR system can be automated, which implies that the whole reasoning process could be automated to a certain extent [CL01]. This assumption means in our opinion, that agents implemented using CBR systems could be able to reason autonomously and therefore to adapt themselves to environmental changes.

The automation capabilities of CBR systems have led us to establish a relationship among cases, the CBR life cycle, and the mental attitudes of BDI agents. Based on this idea, a model that facilitates the implementation of the BDI agents using the reasoning cycle of a CBR system is presented.

3. Implementing Deliberative Agents using CBR Systems

This section identifies the relationships established between BDI agents and CBR systems, and shows how an agent can reason with the help of a case-based reasoning system. Our proposal defines a direct mapping from the concept of an agent to the reasoning model, paying special attention to two elements: (i) how the mapping should allow a direct and straightforward implementation of the agent and (ii) how the agent is able to learn and evolve

with the environmental changes. In the presented model, the CBR system is completely integrated in the agents' architecture, referring to the above-mentioned works, in which the agents see the CBR system just as a reasoning tool. Our proposal is also concerned with the agent's implementation and presents a "formalism" easy to implement, in which the reasoning process is based on the concept of intention. In this model, intentions are cases, which have to be retrieved, reused, revised and retained. This is the main difference between this work and the work presented in [MPA99] [Ol99], in which cases have a more complex structure.

To achieve both goals, the structure of the CBR system has been designed around the concept of a case. The problem, the solution and the result obtained when the proposed solution is applied make a case. Figure 2 shows these components: the problem defines the situation of the environment at a given moment, the solution is the set of states undergone by the environment as a consequence of the actions that have been carried out inside it, and the result shows the situation of the environment once the problem has been solved.

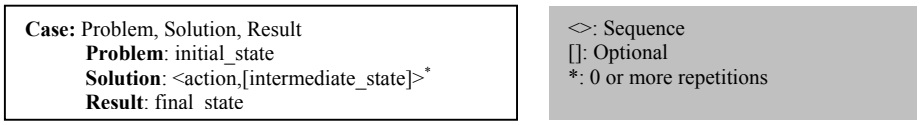

Figure 2. Definition of a case in a case-based reasoning system.

Figure 3 defines what are the beliefs, desires and intentions for a BDI agent. Each state is considered a belief; the objective to reach may also be a belief. The intentions are plans of action that the agent is forced to carry out in order to achieve its objectives [BIP88], so an intention is an ordered set of actions; each change from state to state is made after carrying out an action (the agent remembers the action carried out in the past when it was in a specified state, and the subsequent result). A desire will be any of the final states reached in the past (if the agent has to deal with a situation, which is similar to a past one, it will try to achieve a similar result to the previously obtained result).

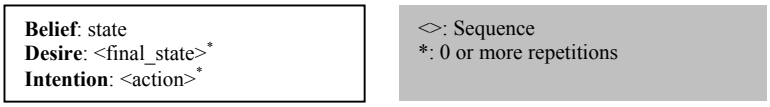

Figure 3. Definition of the mental attitudes of a BDI agent.

The relationship between CBR systems and BDI agents can be established implementing cases as beliefs, intentions and desires which caused the resolution of the problem. The obvious relationship between BDI agents and CBR systems can be identified by comparing Figures 2 and 3.

Using this relationship we can implement agents (conceptual level) using CBR systems (implementation level). Then we are mapping agents into CBR systems. The advantage of this approach is that a problem can be easily conceptualised in terms of agents and then implemented in the form of a CBR system. So once the beliefs, desires and intentions of an agent are identified, they can be mapped into a CBR system.

3.1 Notation

In order to show how to implement such agents, a formal notation is introduced to describe the CBR systems and their reasoning processes. The following section defines the life cycle of a CBR system in terms of the mental attitudes of the BDI agents. Situated in a world or in an environment, an agent senses and affects its environment. The CBR-BDI agent is denoted by a 5-tuple {E, GAL, CM, PAL, EK}. This notation can also be used to define the beliefs, desires and intention of the agents, due to their correspondence with the elements that make up a CBR system. The components of a CBR system in this context are the following:

E: Environment, it is the set of attributes that identify the operational environment in which the agent is embedded. Each environmental variable has associated a name, a range of values and a Boolean value that indicates if such a variable is part of the index field. All cases should be indexed, therefore it is necessary to identify the variables that form part of the index. E is described by a set of values for a fixed set of non-null environment variables, denoted as $E = <e_1, e_2, ...,e_n>$ where $e_i \in E$, $i \in \{1,2,...,n\}$. Therefore E will describe the current environment. Each variable e_i is a 3_tuple $\{name\_var_i, <value_i>, index\}$ where $name\_var_i$ is the name of the variable, $value_i$ is a set of values for each variable and *index* is a boolean field, *1* if the variable is in the index and *0* otherwise.

GAL: Is the General Actions Library and stores all the actions that can be carried out in the environment. Normally such actions should be initially defined by an expert. Each action has associated a name, a number of input parameters (which may be the same as some of the parameters that form part of E) and a number of output parameters that form part of E. One action affects a predefined number of environmental variables, which are associated with such this action. Then GAL = $<ga_1, ga_2, ..., ga_p>$ where $ga_i \in$ GAL, $i \in \{1,2,...,p\}$, ga_i is described as a 3_tuple $\{Name_i, InPar_i, OuPar_i\}$, where $Name_i$ is the name of the action, $InPar_i$ is a set of input parameters $InPar_i \subseteq$ E and $OuPar_i$ is a set of output parameters $OuPar_i \subseteq$ E.

CM: Case Memory, CM, stores sets of previous cases CM = $<c_1, c_2, ...,c_k>$, where each case c_i is formulated as a 3-tuple $\{B, D, I\}$ representing past beliefs, past desires and past intentions.

- *B*: Agent's beliefs are the exact values for each variable of the environment, for any instance of time. $B = <b_1,b_2,...,b_n>$ where each belief is a state, which is a 2_tuple $b_i = \{E, valueE\}$. $E = <e_1, e_2, ...,e_n>$ where e_i are the variables which describe the environment and $valueE = <v_1,v_2,...,v_n>$ where $v_i \in value_i$, $i \in \{1,2,...,n\}$ is the exact value for each variable at a certain instant. Then each belief represents the value that each environmental variable takes at a given time.
- *D*: Desires, *D,* correspond to set of objectives (appropriate final states), which determine the resolution of the problem. D = $<d_1, d_2, ..., d_h>$, where $d_i \in B$, $i \in \{1,2,...,h\}$, $D \subset B$. If the agent is embedded in a deterministic environment it will be possible to identify in advance which are the desired values for the attributes that define such an environment. Nevertheless, if such information is not known, or the environment is not deterministic, D is composed of the final states of the recovered cases during the retrieval stage. Therefore D is initially empty and new states are added to it as the reasoning process progresses.
- *I*: An intention, *I*, is an ordered sequence of actions that should be carried out to solve a given problem. *I* represents a trigger for a corresponding plan from the producer actions library PAL. They are previous actions, and have the values of the input parameter and output parameter.

PAL: Producer Actions Library is a collection of intentions, PAL = $<I_1, I_2, ..., I_g>$ where $I_i \in$ PAL, $i \in \{1,2,...,g\}$, I_i is described as an ordered sequence of actions, $I_i = <a_1, a_2, ..., a_j>$ where each change from belief to belief (state to state) is produced after carrying out an action a_i, a_i is a 3_tuple $\{Name_i, ValueInputPar_i, ValueOutputPar_i\}$ where $Name_i \in ga_i$; ga_i is a general action from general actions library GAL. $ValueInputPar_i$ and $ValueOutputPar_i$ are sets of exact values for input parameter and output parameter of ga_i. *ValueInputPar* determines the exact values of the input parameters before the action is carried out, $Name_i$, is a past experience and *ValueOutputPar* indicates the exact values of the output parameters after the action has been carried out. $ValueInputPar_i = <v_1,v_2,...,v_k>$ where $v_i \in value_i$, $i \in \{1,2,...,k\}$ and $ValueOutputPar_i = <v_1,v_2,...,v_l>$ where $v_i \in value_i$, $i \in \{1,2,...,l\}$.

EK: Expert knowledge EK is composed of a set of default rules associated with the case adaptation and case retention process. The procurement of these rules is automated or Expert-defined. EK = $<r_1,r_2,...,r_n>$ where the r_i may be partitioned into the sets (Previous, Consistent); Previous = < $(name\_var_1,v_1)$, $(name\_var_2,v_2)$,..., $(name\_var_r,v_r)$ > where $v_i \in value_i$, $i \in \{1,2,...,r\}$ is the exact value for some variables. And Consistent = $<(name\_var_{r+1},v_{r+1}),..., (name\_var_n,v_n)>$ where $v_i \in value_i$, $i \in \{r+1, r+2,...,n\}$ is the exact value for others variables.

Once all the parameters that identify the problem are defined, the agents will be able to start working and reasoning following the CBR reasoning cycle. Once the agent has a new problem, B, a similarity metric is used to retrieve, from the case memory, **CM**, the beliefs, $B_{1..n}$, intentions, $I_{1..n}$, (which implies that we must recover the actions already carried out) and desires $D_{1..n}$ that may be used to achieve its aim. The similarity metric may be either explicitly defined as e.g. the scalar product or the cosine of the angle between the vectors or may be implicit in the use of Artificial Neural Networks (ANNs) or Kernel methods(see below). **EK** is then used during the adaptation stage to obtain the plan, I_{end}, which may be used to achieve the desire, **D**. To achieve such desire it is necessary to carry out actions stored in the **PAL** or new actions described in the **GAL**. Then the revision may be carried out by human experts, using simulation techniques [CAR01], etc. If the *revision* process concludes that the proposed solution is not acceptable, the plan is sent back to the reuse stage, as indicated in Figure 1, where a new solution will be proposed. The new case, c_{end}, is finally indexed and stored in the case-base, **CM,** during the retain stage. This process implies that knowledge held in memory changes. So the rules which represent the expert's knowledge may also change.

As was stated in the introduction of the paper, the aim of this investigation is to develop a methodology for constructing deliberative agents capable of learning and adapting to new situations. To set up an agent using this architecture we need to identify an initial set of beliefs, desires and intentions and include them in the case-base of the agent in the form of cases. Then a number of metrics for the retrieval, reuse, revise and retain steps have to be

defined. Besides, rules that describe the Expert's knowledge must be established, if available. Once the agent has been initialised it starts the reasoning process and the four steps of the CBR system are run sequentially and continuously until its goal is achieved.

4 Case study: The E-business Engineering Sales Support System

The construction industry is an information intensive economic sector. This activity, as many others, require the use of a great amount of data, ranging from product data to technical publications, from buildings regulations to best practice guides. This section describes an information system that has been developed for a construction company, D&B Constructions, in which a CBR-BDI agent has been used. From now on, we will refer to the BDI agents implemented as mentioned in the previous section as CBR-BDI agents. This distributed agent based system helps the company to make as much profit as possible from the information published on the Internet and the information that the company holds, and to reuse it as much as possible especially to estimate budgets.

The e-business engineering sales support system incorporates several specialised agents that search for and organise information and data, and several assistant sales support agents. The first prototype of this multiagent system has been implemented in Java using servlets (running in an Apache, Tomcat, jserv, Linux environment). A commercial system is under construction presently, after an initial successful testing period. The specialised agents are Java applications that run on the company Intranet and the assistant agents run in a portable computer connected to the Internet via a mobile phone.

The D&B Constructions deals with medium to small construction problems and it specialises in installing heating and air conditioning systems in a wide area of the Northwest of Spain. They have a sales force that is growing continuously, which implies that continuously new salesmen are taken on board without much experience in many cases. Until now the salesmen had to visit the clients on demand, had to take notes of their problems and then they had to contact an engineer or an experienced salesman, which had to estimate the work price and personnel and material required to carry on the work. The system here outlined was developed to aid the sales force, and in particular the inexperienced personnel, in the estimation of costs, thus reducing the process bureaucracy.

4.1 The System Architecture

In the expansion policy of B&D Constructions one of the main points is its incorporation of new technologies. Several steps will be taken in this direction for developing a web based information system that allows the company to publish information about their activities and that facilitates the communication between the administration, the sales force, the providers and the clients.

Figure 4 presents the architecture of the multiagent system. The planning agent is a CBR-BDI agent, implemented with the architecture described in Section 3. The planning agent is the only CBR-BDI agent used in this architecture, it estimates the construction cost, and the personnel and material required to carry out a construction project. It also generates reports about clients (or potential clients) using the information stored in the company databases and the one obtained by the internet search agent from the web. The planning agent generates working plans using their incorporated CBR system. The internet agent incorporates a web search engine that looks continuously for potential clients, information about them, new providers and products. This agent starts looking from a predetermined web address and searches for new ones using natural language processing strategies, optimized for the Web, based on the combination of the grammatical characterization of words and statistical decision techniques [Co01]. This agent is monitored and guided by a marketing expert. We are also studying the possibility of implementing this agent using the CBR-BDI model here presented or as proposed in [Ol99]. Assistant agents (they can be as many agents as salesmen) are interface agents that facilitate the communication between the salesmen and the planning agent, they also hold summary information about the clients visited by its salesman owner and by the rest of the salesmen.

Before a salesman visits a client, he/she interrogates his/her assistant agent providing a description of the client (Name, Address and Activity). The assistant agent compares this data with previous queries and if a match is found, using relaxed K-nearest neighbor algorithms, the data it holds about the client is presented to the salesman [WG99]. This information is related to previous building work carried out for the client, his financial status, comments about him, noted by the firm's personnel during previous relations with this client, location information and other potentially useful data. This information is valuable especially when an inexperienced salesman starts a negotiation process. If the assistant agent cannot help the salesman or if the salesman demands more information, his assistant agent contacts the planning agent, which searches for information about the client in its case-base.

This agent also interrogates the internet search agent asking for information about clients. The internet search agent obtains information from the web, analyses and indexes it using natural language processing algorithm optimized for Internet, as mentioned above. Information about potential clients, new materials and providers is sent to the administration agent, which can be interrogated by any of the Construction Company managers, engineers or sales supervisors. They can, then, use this pruned information to target new business. The administration agent is an interface agent that facilitates the interaction between the users (Company managers) and the rest of the elements of the system: agents, databases and even salesmen.

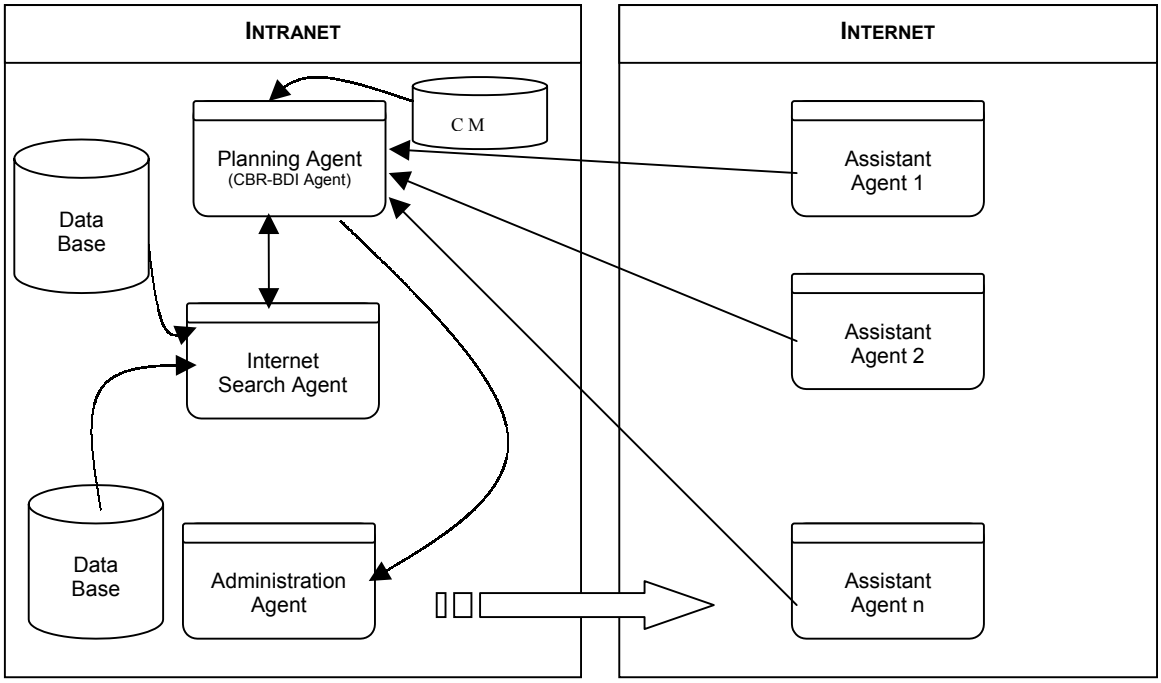

Figure 4: Agent-oriented e-business architecture.

As mentioned above, the multiagent [We96] system has been built using a Java-based library. This library is an extension of the one used to implement the STEB (Simulated Tactical Environmental Bubble) system [CL01]. The STEB system was designed to forecast the temperature of ocean waters ahead of ongoing vessels. It is a multiagent system composed of several software agents that co-operate between them and that are operational in different locations: war/oceanographic vessels and in an Oceanographic Laboratory (Plymouth Marine Laboratory). The agents installed on the vessels use hybrid Case Based Reasoning-Artificial Neural Network (CBR-ANN) system [CL01] to forecast and communicate with the rest of the agents using Knowledge Query and Manipulaton Language (KQML) performatives.

Similarly to the STEB system, when constructing the e-business multiagent system here presented, a decentralised architecture was selected, in which agents interact between one another when they require information or need to share data. The agents communicate with each other using a message passing protocol. Such messages are KQML performatives. The agents of this system collaborate between each other sharing information and working together to achieve a given goal. They use a simple collaboration mechanism. For example, if Salesman A is associated with the Assistant Agent A, and visits a Client X, the Assistant Agent A has to contact (send the problem of the Client X, via a performative) the Planning Agent. Then the Planning Agent generates the solution plan, as will be shown in Section 4.2, and sends it back to the Assistant Agent.

In a system of these characteristics, data security has to be taken into consideration. A Role-based Access Control with elements that allow the certification of operations has been implemented to guarantee the data security and the information protection [CAR01]. This security system protects the databases and the information stored in the system from external “agents” or non accredited personnel.

4.2 A CBR-BDI Planning Agent

Since our intention is to develop a dynamic distributed e-business solution it may be adequate to use agents with adaptation and learning capabilities. The working mode of the planning agent will be explained to show how the agents presented in section 3 can reason, acquire new knowledge and help the distributed information system to evolve. The other agents of the system do not have a CBR-BDI architecture because they are responsible for carrying out tasks that do not require reasoning. They are interface agents (assistant and administration ones) or have mechanical tasks to perform such as the internet search engine. Nevertheless, as has been mentioned above, we are studying the possibility of implementing the Internet search agent in the form of a CBR-BDI agent, to achieve more autonomy and efficiency. There are two tasks carried out by the planning agent: estimation of the construction cost, the personnel and material required and generation of reports about clients. We will focus on the first task. The second task is carried out following an automatic sequence of queries to the agent case-base and to the system databases. To facilitate the understanding of the problem we have simplified the problem reducing the number of variables used to describe a building or a house (the implemented system uses 45 attributes) and the number of states that define an intention.

In this context, the planing agent may be defined as follows. The variables that form part of the working environment, **E,** are (where e_i **is a 3_tuple {**$name\_var_i$**,** $<value_i>$**,** $index$ **}**):

Building Type = *{BT,<h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12>,1}*

Size = *{S, <50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 205, 215 ,225, 235, 250, 265, 275, 300, 400, 500, 600, 700, 800, 900, 1000>, 1}*

Insulating Material = *{IM,<I1, I2, I3, I4, I5, I6, I7, I8, I9, I10>, 1}*

Construction Year = *{CY,< <1900, 1900, 1905, 1910, 1915, 1920, 1925, 1930, 1935, 1940, 1945, 1950, 1955, 1960, 1965, 1970, 1975, 1980, 1985, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, >2005>,0}*

Heating Pipe = *{HP,< no, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, G1, G2, G3, G4, G5, G6, P1, P2, P3, P4, P5, P6, E1, E2, E3, E4, E5, E6>,1}*

Heating Radiators = *{HR,<0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, >35>,0}*

Heating System = *{HS, <B1 6-600, B2 6-600, B3 6-600, B4 6-600, B1 7-600, B2 7-600, B3 7-600, B4 7-600, B1 7-700, B2 7-700, B3 7-700, B4 7-700>, 0}*

Others variables are:

Material-price = *{Material-price,<100, 125,150,155,160, 165,170,175,180,185,190,195,200,210,215,220,225, 230,235,240,245,250,255,260,265,270,275, 280,285,290,295,300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 425, 450, 475, 500>,0}*

workers required / ratio hours/men = *{workers required/ ratio hours/men, <1t/1d, 1t/2d, 1t/3d, 2t/1d, 2t/2d, 2t/3d, 2t/4d, 3t/1d, 3t/2d, 3t/3d, 3t/4d, 3t/5d, 4t/1d, 4t/2d, 4t/3d, 4t/4d, 4t/5d >,0}*

pipes = *{pp, <25m pp+elements, 30m pp+elements, 35m pp+elements, 40m pp+elements, 45m pp+elements, 50m pp+elements, 55m pp+elements, 60m pp+elements, 65m pp+elements, 70m pp+elements, 75m pp+elements, 80m pp+elements, 85 m pp+elements, 90m pp+elements, 95m pp+elements, 100m pp+elements >,0}*

installation kit = *{installation kit, < installation kit10, installation kit11, installation kit12, installation kit13, installation kit14, installation kit15, installation kit20, installation kit25, installation kit26, installation kit27, installation kit28, installation kit29, installation kit30>,0}*

GAL (ga_i **is a 3_tuple {**$Name_i$**,** $InPar_i$**,** $OuPar_i$**}):**

Action ga1: {Installation of heating pipes,(S, Material-price, workers required/ ratio hours/men, pp, installation kit),HP}
Action ga2: {Installation of radiators (S, Material-price, workers required/ ratio hours/men, installation kit), HR}
Action ga3: {Installation of diesel heating system (BT, CY, Material-price, workers required/ ratio hours/men, installation kit), HS}

PAL (a_i **is a 3_tuple {**$Name_i$**,** $ValueInputPar_i$**,** $ValueOutputPar_i$**}):**

Action A1: {Installation of heating pipes, (225, 200 €, 3t/2d, 85 m pp+elements, installation kit27), D2}
Action A2: {Installation of radiators,(225, 266 €, 2t/1d, installation kit13),16}
Action A3: {Installation of diesel heating system,(h2, 1973, 350 €, 2t/0.5d, installation kit12),B2 6-600}

Where for example *Action A1* indicates that the cost of the installation of the heating pipes, of type *D2*, was 200 €, 3 technicians were required during 2 days, 85 metres of pipes were used together with the elements needed to install them and the installation tool kit used was number 27.

B (b_i **is a 2_tuple {**E**,** $valueE$**}.** $E = <e_1, e_2, ..., e_n>$ **,** $valueE = <v_1, v_2, ..., v_n>$ **):**

Belief B1: {BT=h2; S=225; HP=no; IM= I7; CY=1973; HR=0 }
Belief B2: {BT=h2; S=225; HP=D2; IM= I7; CY=1973; HR=0}
Belief B3: {BT=h2; S=225; HP=D2; IM= I7; CY=1973; HR=16}
Belief B4: {BT=h2; S=225; HP=D2; IM= I7; CY=1973; HR=16; HS= B2 6-600}

Where *B1* indicates that the building is of type *h2*, which means that is a country house with 2 floors, the size, 225, is given in square metres. Also indicates that the house does not have any heating system, that the insulating material is of type *I7*, which is a very precarious insulation system and finally it indicates that the house was build in 1973.

D ($d_i \in B$):

Initially empty. Desires will be added to it. Such new desires are the final states of the most similar retrieved cases to the problem to solve.

I (a_i is a 3_tuple {$Name_i$, $ValueInputPar_i$, $ValueOutputPar_i$}):

I1 = <A1, A2, A3>

A1: {Installation of heating pipes, (200 €, 3t/2d, 85 m pp+elements, installation kit27), D2}

A2: {Installation of radiators,(266 €, 2t/1d, installation kit13),16}

A3: {Installation of diesel heating system,(350 €, 2t/0.5d, installation kit12),B2 6-600}

Where *I1* is a plan, composed of three actions.

EK (r_i is a 2_tupla (Previous, Consistent) :

r1: {((S,225),(BT,h2)), D2}

r2: {((S,100),(BT,h3)), D1}

r3: {((CY,1980),(BT,h2)), E1}

r4: {((CY,1973),(BT,h1)), B3 7-700}

Where for example, r4 indicates that a building of type h1, constructed in 1973 requires a heating system of the type B3 7-700. Knowledge is expresed in the form of rules, that for example relate the type of construction with a time period and a particular type of heating system.

The desire of the agent with respect to a particular Client is to determine a plan, which is the one determined by the Client, satisfying his time, price and quality restrictions. For example, if a Client, whose property is defined by the B_{new} *(BT=h2; S=175; HP=no; IM= I2; CY=1980)* requires an heating system, the goal of the planning agent will be to determine a plan that transform B_n in B_{FINAL}*=(BT=h2; S=175; HP=E1; IM= I2; CY=1980)*.

Then, once the intentions and beliefs of the agents are identified, the agent will be able to generate plans to achieve its desires, providing that the metrics of each of the stages of the CBR system, that defines the agent reasoning mode, had been defined. The goals of the agent change with each new Client. A brief description of the metrics used by this agent during the reasoning is now presented.

Retrieval:

The retrieval stage must be carried out using a method that guarantees the retrieval of a reasonably small number of cases that are related to the current problem case. We have experimented with a number of different methods for evaluating the similarity of the current problem with those in the Case Base: K-means, Sparse Kernel Principal, Componet Analysis, the Self Organising Map and the Scale Invariant Map. Best results have been obtained with the Sparse Kernel Principal Component Analysis.

Kernel methods can be used in case-based reasoning systems when cases can be represented in the form of numerical feature vectors. Kernel methods were initially developed in the context of supervised learning and were used in Support Vector Machines (SVMs). (An extensive bibliography can be found at http://www.kernel-machines.org). Unsupervised Kernel methods, though, have tended to lose the very valuable property of sparseness which the SVMs have: in SVMs, the crucial cases which determine a function are identified and all the others are not used in the construction of the classifier. If we were to use e.g. Kernel Principal Component Analysis (KPCA) directly we would retrieve many cases to some extent. We [FC01] have thus developed a Sparse KPCA method which identifies prototypical cases in the retrieval stage.

Briefly, we map the data into a high dimensional feature space, typically using a Gaussian kernel

$$K_{ij} = \exp\left(- \left(x_i - x_j\right)^2 \Big/ \left(2\sigma^2\right) \right)$$

We then perform Sparse KPCA [SMS99,FC01] by doing an eigenvector-eigenvalue decomposition on this matrix which can be shown to be a simple method of finding the Principal Components in the feature space. By limiting ourselves to the hypercube (rather than hypersphere) containing the data, we can show that the Principal Components occur at the vertices of the hypercube which makes the task of finding the best case very simple.

The first principal component is calculated by

$$x_k = \arg\max_{x_i \in \chi} \sum_{i=1}^{M} \left| K_{ki} \right|^2$$

which again requires us only to evaluate the kernel matrix.

So the solution to finding the first Principal Component using this method is exceedingly simple. We may think that subsequent "principal vectors" can be found by removing this vector from further consideration and ensuring that the subsequent solutions are all orthogonal to the previously found solutions. Consider first the "naive" solution which is simply to remove the winner of the first competition from consideration and then repeat the experiment with the remainder of the data points. However these data points may not reveal further interesting structure: typically indeed the same structure in input space (e.g. a cluster) may be found more than once.

An alternative is to enforce orthogonality using a Gram Schmidt orthogonalisation in feature space. The method is defined by

$$K_{i+1}(x_j, x_k) = K_i(x_j, x_k) - \frac{K_i(x_j, x_i) K_i(x_k, x_i)}{K_i(x_i, x_i)}$$

One difficulty with this method is that we can be moving out of the space determined by the norm. The solution is renormalise this point. An alternative which has proved to be very successful in practise [FC01] is to simply remove from future consideration the first Principal Component point and all other points which have high dot product (as measured by the K matrix) with it. However in the current context, these similar points may be valuable cases to solve the problem. Thus we project the problem case onto the feature space using

$$K(x, x_j) = \exp\left(-(x - x_j)^2 / \sigma\right)$$

Then we identify the most similar case to the problem case using Sparse KPCA. Now, all the cases of its cluster are retrieved and reused in the following reasoning stage. This method reduces the computational work because once a cluster is identified, the agents need only work with the cases of such a cluster.

Reuse:

An initial solution can be obtained by using the sequence of actions carried out in the past, or modifying the sequence of actions, adapting them to the new problem. At this point, two possibilities may occur. First, that the environment was, in the past, in a state that is almost identical to the new problem, and the same solution required now was obtained. In that case, the same sequence of actions that in the past was successful, can be carried out. In the second case, the retrieved cases are similar to the present state but with some differences. So a sequence of actions, that is a mixture of those that were recovered in the previous phase, is constructed. To do so, an acyclic directed graph is created, whose first vertex is the new problem/state, and the last vertices are the final states. The construction of the graph is carried out starting from the new state and applying to it a function which determines its similarity with all the recovered states; those that are more similar will determine what action will be carried out starting from the new state. This process is repeated with all the cases until a final state is reached.

Once the graph has been constructed, the algorithm of Dijkstra is used [SWW99] to determine the shortest way (the way that goes through fewest vertices) taking the new state as the origin. Such a path will define the actions that must be carried out from the new state, and so, they will make up the new intention. In this case, the Expert Knowledge, EK, relies on this algorithm.

Revise:

The revision process is carried out by an experienced engineer.

Retain:

After the work has been carried out, the plans are stored in the form of cases. Once a new case is created, it is stored in a temporal case-base. A senior salesman accesses this case-base via the administration agent and decides which of these cases/instances should be stored by the CBR-BDI planning agent. The ability of the kernel methods to select prototypical cases and to identify those cases that are already represented by these prototypes can be used to successfully prune the case base without losing valuable information [FC01]. Other techniques to automate this stage are under investigation.

5 Results and Conclusions

The acceptance of this agent-based distributed system, by the Company staff, has been excellent. For evaluation purposes, during the testing period, the case-base of the planning agent has been fed with 320 cases related to the installation of heating systems. These cases were selected to cover a wide spectrum of possible installations that the company could carry out (D&B Constructors hold a data base with over 6200 installations from January 1997). The system was interrogated on 35 occasions and its output was qualified as adequate by experienced salesmen.

In 32 occasions the estimation differed by less that 4% of the one given by an expert, and in the other 3 occasions differed by less than 11%. These deviations were caused by two different reasons: in the first case the client required a combination of installations and equipment, which did not appear in any of the 320 cases stored in the agent case-base. The other two errors were caused by human mistakes, a misinterpretation between the salesman and the client. Therefore from the point of view of the planning agent, these last two errors should not be taken into account. In the other case, the error could be minimised during the review phase. With respect to this point, strategies are under investigation to identify problems and/or "generated plans" with potential risks.

It is expected that the accuracy of the business solution will increase as more cases are introduced in the planning agent memory. Company experts have estimated that the use of this agent-based system could reduce the installation sales cost up to 35% of the actual cost, and the time of the sale up to 50%.

The CBR-BDI architecture presented in this paper solves one of the problems of the BDI architectures, which is the lacking of learning capacity. The reasoning cycle of the CBR systems helps the agents to solve problems, facilitate its adaptation to changes in the environment and to identify new possible solutions. New cases are continuously introduced and older ones are eliminated.

Morá [Mo98] have described the gap that exists between the formalisation and the implementation of BDI agents. What we propose in this article is to define the beliefs and intentions clearly (they don't need to be symbolic or completely logic), and to use them in the life cycle of the CBR system, to obtain a direct implementation of a BDI agent. This paper has shown how single agents can be developed with this technology, and how such agents can be successfully used to construct an efficient agent based system for e-business. The work presented in this paper is the basis of a wider project that aims to construct a tool for developing dynamic agent-based e-business solutions in a flexible and efficient manner.

Acknowledgement

This work has been partially supported by the CICYT projects TEL99-0335-C04-03 and SEC2000-0249, the PGIDT00 project MAR30104PR and the project SA039/02 of the JCyL.

Bibliography

[AP94] Aamodt A. and Plaza E. (1994) Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches, AICOM. Vol. 7. No 1, March.

[BIP88] Bratman M.E., Israel D., and Pollack M.E. (1988) Plans and resource-bounded practical reasoning. Computational Intelligence, 4. pages 349-355.

[Br87] Bratman M.E. (1987) Intentions, Plans and Practical Reason. Harvard University Press, Cambridge, M.A.

[BW98] Bergmann R and Wilke W. (1998) Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. European Conference an Artificial Intelligence (ECAI'98), John Wiley and Sons.

[CAR01] Corchado J.M., Aiken J, Rees N. (2001) Artificial Intelligence Models for Oceanographic Forecasting. Plymouth Marine Laboratory. ISBN-0-9519618-4-5.

[CL90] Cohen P.R. and Levesque H.J. (1990) Intention is choice with commitment. Artificial Intelligence, 42(3).

[CL01] Corchado J. M. and Lees B. (2001) A Hybrid Case-based Model for Forecasting. Applied Artificial Intelligence. Vol 15, no. 2, pp105-127.

[Co01] Corchado J. M. (2001) CBR Agents in an e-commerce environment. Proceedings of WOOPS-ECOOP, Budapest, June.

[CS90] Caplan L.J. and Schooler C. (1990) Problem Solving by Reference to Rules or Previous Episodes: The Effects of Organized Training, Analogical Models, and Subsequent Complexity of Experience. Memory & Cognition, 18(2). pages 215-227.

[CF02] Corchado, E. and Fyfe, C. (2002) The Scale Invariant Map and Maximum Likelihood Hebbian Learning, Proceedings of the Sixth International Conference on Knowledge based Intelligent Engineering Systems, KES2002.

[FC01] Fyfe C. and Corchado J. M. (2001) Automating the construction of CBR Systems using Kernel Methods. International Journal of Intelligent Systems. Vol 16, No. 4, April 2001.

[FG94] Feret M. P. and Glasgow J. I. (1994) Explanation-Aided Diagnosis for Complex Devices, Proceedings of the 12th National Conference an Artificial Intelligence, (AAAI-94), Seattle, USA, August 94.

[GL86] Georgeff M.P. and Lansky A.L. (1986) Procedural knowledge. In Proceedings of the IEEE Special Issue on Knowledge Representation, volume 74. pages 1383-1398.

[GR95] Gärdenfors P. and Rott H. (1995) Belief Revision. In Handbook of Logic in Artificial Intelligence and Logic Programming, volume IV, chapter 4.2.

[Je92] Jennings N.R. (1992) On Being Responsible. In Y. Demazeau and E. Werner, editors, Decentralized A.I. 3. North Holland, Amsterdam, The Netherlands.

[KG91] Kinny D. and Georgeff M. (1991) Commitment and effectiveness of situated agents. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91), pages 82-88, Sydney, Australia.

[Ki94] Kinny D., Ljungberg M., Rao A.S., Sonenberg E.A., Tidhar G, and Werner E. (1994) Planned team activity. In Artificial Social Systems. Lecture Notes in Artificial Intelligence (LNAI-830). Amsterdam, Netherlands. Springer Verlag.

[Ko98] Kohonen, T. (1998) Self Organising Maps. Springer.

[Mo98] Móra M. C., Lopes J. G., Viccari R. M. and Coelho H., (1998) BDI Models and Systems: Reducing the Gap. ATAL-98.

[MPA99] Martín F. J., Plaza E., Arcos J.L. (1999). Knowledge and experience reuse through communications among competent (peer) agents. International Journal of Software Engineering and Knowledge Engineering, Vol. 9, No. 3, 319-341.

[NLC96] Napoli A, Lieber J and Curien R. (1996) A formal Analysis of Case-based reasoning in an object-based representation Context. In Faltings B and Smith I, editors. 3rd European Workshop and Case-based Reasoning, Lausane, Lecture Notes in Artificial Intelligence, 1168, pp295-308. Springer-Verlag, Berlin, 1996.

[Ol99] Olivia C., Chang C. F., Enguix C.F. and Ghose A.K. (1999) Case-Based BDI Agents: An Effective Approach for Intelligent Search on the World Wide Web", AAAI Spring Symposium on Intelligent Agents, 22-24 March 1999, Stanford University, USA

[RG91] Rao A. S. and Georgeff M. P. (1991) Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann Publishers, San Mateo, CA.

[RG95] Rao A. S. and Georgeff M. P. (1995) BDI Agents: From Theory to Practice. First International Conference on Multi-Agent Systems (ICMAS-95). San Franciso, USA, June.

[RS89] Riesbeck, C. K. and Schank, D. B. 1989. Inside Case Based Reasoning. Erlbaum.

[Sh93] Shoham Y. (1993) Agent-Oriented programming. Artificial Intelligence, 60(1): pages 51-92.

[SMS99] Smola, A. J., Mangasarian, O. L. And Scholkopf, B. (1999) Sparse Kernel feature analysis. Technical Report 99-04, University of Wisconsin, Madison.

[SWW99] Schulz F., Wagner D. and Weihe K. (1999) Dijkstra´s Algorithm On-line: An Empirical Case Study from Public Railroad Transport. Algorithm Engineering . pages 110-123.

[We96] Weiβ G. (1996). Adaptation and learning in multi-agent systems: some remarks and Bibliography. Proceedings IJCAI'95 workshop. Montreal, Canadá. August, 1995.

[WG99] Watson I and Gadingen D.(1999) A Distributed Case-Based Reasoning Application for Engineering Sales Support. In Proceedings of IJCAI-99. Morgan Kaufmann Publishers Inc. Vol 1, pp 600-605.

[WJ94] Wooldridge M. and Jennings N. R. (1994) Agent Theories, Architectures, and Languages: A Survey. Procs. ECAI-94 Workshop on Agent Theories, Architectures, and Languages.

[WL98] Wendler J. and Lenz M. (1998) CBR for Dynamic Situation Assessment in an Agent-Oriented Setting. Proc. AAAI-98 Workshop on CBR Integrations. Madison (USA) 1998.

[WM94] Watson I. and Marir F. (1994) Case-Based Reasoning: A Review. Cambridge University Press, 1994. The knowledge Engineering Review. Vol. 9. Nº3.

[Wo99] Wooldridge M. (1999) Intelligent Agents. Multiagent Systems. A modern approach to Distributed Artificial Inteligence. Edited by Gerhard Weiss. Pages 27-77.