Applying a Case-Based Reasoning System Development Tool in the Design of BDI Agents

Ken Yen-Ru Cheng, Chiung-Hon Leon Lee[*], Alan Liu
Department of Electrical Engineering and Center for Telecommunication Research
National Chung Cheng University
[*]Department of Computer Science and Information Engineering,
Nanhua Universiti
Taiwan
d95415007@ccu.edu.tw, leonlee@dragon.ccut.edu.tw, aliu@ccu.edu.tw

Abstract

Most deliberative agents are designed based on the BDI architecture. Although the BDI architecture is uncomplicated and its decision process is easy to understand and execute the BDI architecture is difficult to implement and has no functionality of learning. Extending the architecture of BDI with CBR to develop the deliberative agent could make the development of agent easier compared to implementing a pure BDI architecture with the functionality of learning and adaptation. In this paper, we develop a domain-independent system development tool, JCBRDT. By using JCBRDT, the system developers could use CBR to develop a BDI agent, create a CBR system easily, and save the time on designing and maintaining the CBR system.

Keywords: agent, BDI, CBR, development tool.

1 Introduction

An agent is one kind of software programs with functionality of replying the events from the environment, doing some active behaviors according its goals, interacting with other agents or human beings, and achieving the current goal according the per-experiment. A deliberative agent means that agent whose behaviors are based on a reasoning system [8]. For deliberative agents, many system architectures are brought up. Many of these architectures are based on the Belief-Desire-Intention (BDI) architecture. Under the BDI architecture, an agent’s behaviors are composed of three psychological properties: beliefs, desires and intentions [2]. The term, "Beliefs", means the information that the agents presume about itself and environment, "Desires" the states that the agents want to reach, and "Intentions" the steps describing how to arrive at the desired states from the current states. An intention could be represented by a series of actions and these actions could be regarded as a plan.

Although the BDI architecture is straightforward and its decision process is very easy to be realized and accomplished, it is difficult to find an efficient way to implement it as a system. Rao and Georgeff [2] point out two problems about BDI architecture: (1) the powerful logicality of BDI architecture may not achieve full implementation; (2) this kind of agent has some difficulties to have the functionality of learning.

To solve these problems, some researchers use Case-Based Reasoning (CBR) to export BDI architecture to develop the deliberative agents [9, 11]. A CBR approach helps an agent to learn and adapt instantaneously and work well without recompile the program code when the environment changes. By connecting the three properties of BDI agents with the information reasoned with CBR, the system developer could build agents with the capabilities of believes, desires, intention, and learning.

In this paper, we describe our Java Case-Based Reasoning Development Tool, JCBRDT. By using this development tool, the system developer can expand the development of agent with CBR capability and develop a CBR system easily. JCBRDT is a domain-independent system development tool. The CBR systems which are designed by the system developer himself are usually used in one specific domain. JCBRDT, however, could be used to develop different domains by changing the content of XML files to switch the applied domain and save the system developers' time to design and maintain these CBR systems. The following section starts from reviewing related work, and the section three states our approach. Section 4 describes the implementation of our system, and the last section presents a brief conclusion.

2 Literature Review

JCBRDT is mainly expanded from jCOLIBRI [6] and provides a new solution to the weakness of jCOLIBRI. In this section, we introduce the concept of the design of jCOLIBRI and then describe the semantic similarity used in JCBRDT.

2.1 Concepts of jCOLIBRI

jCOLIBRI proposes a domain-independent architecture to help the system developer to design a CBR system and add the CBR ontology into the design process of CBR system. CBR Ontology is an ontology constructed by the terminology, reasoning process, and description and acquisition of cases.

jCOLIBRI creates the following four XML files after the design process:

- **Tasks and Methods:** the solutions of each task;
- **Case Base:** the type and data contain source of case base;
- **Case:** the structure of case and
- **Problem Solving Methods:** the solution of problem.

CBR system plans the case structure and methods according the information above.

2.2 Semantic Similarity

The issue of similarity is important in CBR process. We focus on the concept of semantic similarity, which is a subject of how to let computers or machines to understand the language of human kind. The development of WordNet [5] helps the study of semantics. The term classification that developed by WordNet researchers could save the time to develop a word ontology. Most systems use the term classification of WordNet to calculate the semantic similarity.

WordNet is a word system constructed with the concept of psychology and it is also an English dictionary based on semantic concepts. There are four categorizations in its terms: noun, verb, adjective, and adverb and the relations of these terms are separated into four parts: synonym/antonym, hypernym/hyponym, holonym and meronym.

The algorithms using WordNet to calculate the similarity usually use Information Content (IC) to be the index of the specific degree of the concept. Resnik [14] uses IC to calculate similarity and he uses the common parent concept to measure the similarity between two concepts. This common parent concept is called Most Specific Common Abstraction (MSCA). The similarity degree is zero when MSCA is zero. Jiang and Conrath [7] extend Resnik's method and consider the edge distance between two concepts simultaneously. Seco et al. [13] use the hyponyms in WordNet to evaluate the value of IC.

3 Research Approaches

CBR systems have aided many researchers in various domains [10]. However, there are only few researches discussing about the CBR system generator. That urges us to create a domain independent CBR system development tool. This development tool could be used in varied domains and save system developers' time to develop and maintain CBR system.

3.1 Method to Shorten the Similarity Calculating Time

Because of the capability of learning, a CBR system stores more and more cases as time passes. More cases means that more time needed to search the most similar case in a case base. To shorten the similarity calculating time, we use the following calculating formula to measure the similarity of two cases.

$$No\_Weight\_Agerage: \qquad sim_{ASV} = \frac{\sum_{k=1}^{N} sim_k}{N} \tag{1}$$

$$Weighted\_Agerage: \qquad sim_{ASV} = \frac{\sum_{k=1}^{N} w_k \times sim_k}{\sum_{k=1}^{N} w_k} \tag{2}$$

Where sim_{ASV} (Average Similarity Value) is the whole similarity of two cases, sim_k is the similarity of the kth characteristic of two cases with the value between 0 and 1, N is the amount of the characteristics of a case, and w_k is the weight of kth characteristic.

The purpose of case retrieval is to find a case that has the highest similarity with the description of user's problem. Using this property, we could shorten the similarity calculation time. When finding the sum of characteristic similarities of retrieved case could not be higher than the whole similarity of the found case that presently has highest similarity, the next calculation should be omitted. That means the retrieving process needs less time if it finds a high similarity earlier.

Using an example to illustrate our method, assume that sim_{ASVP} (Previous Average Similarity Value) is the highest similarity in the case retrieving process; sim_{ASVC} (Current Average Similarity Value) is the similarity of the case that is considered currently; sim_{Ck} is the similarity of the kth characteristic of the case that is considered currently. Suppose that sim_{ASVP} is 0.95 and N is 5. First, multiply sim_{ASVP} by N, that is 5, and the result is 4.75. After that subtract 1 from this result and the new result is 0.75 called the first result. Second, if the first result is larger than sim_{C1}, sim_{ASVP} would be greater than sim_{ASVC} according to the counting result (a) in List 1. We could terminate the calculating sequence for the current case.

On the contrary, if the first result is not larger than sim_{C1}, add the first result, 0.75, with 1 and the new result is 1.75 called the second result. If the second result is larger than the addition of sim_{C1} and sim_{C2}, that means sim_{ASVP}

would be greater than sim_{ASVC} according the counting result (b) in List 1, then we could terminate calculating sequence for current case. Continue this calculating process until finishing the calculation of sim_{C5}. From this example, we know that some calculating steps could be omitted if a case which has higher similarity is found earlier.

3.2 Measurement of Similarity

In JCBRDT, we use object, distance, string and semantic similarity comparing functions as the comparison functions for case similarity. Different data types have different similarity comparison function. The data types are Integer, Double, Boolean, String, and Word. We show the relationship between data type and similarity comparison functions in Figure 1. The left part represents the data type which can be provided to the system designers and the right part represents the similarity comparison functions. The data type Integer is either Interval or Equal; Double is either Interval or Equal; Boolean could only use Equal; String is either MaxString or Equal; Word is Equal or WNSimilarity. The purpose for the relationship between data type and similarity comparison function is to avoid the error in calculating process of similarity comparison function.

3.2.1 Object Similarity Function

An object similarity function compares two values as objects. The comparison method is to compare these two objects if they are equal or not and the result is 0 or 1. The mathematical formula is as follows:

3.2.2 Distance Similarity Function

Distance similarity function calculates the distance

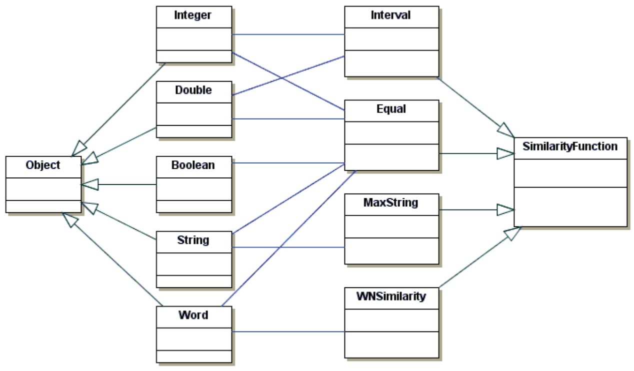


Figure 1 Concept Diagram of Relationship between Data Types and Similarity Functions

$$sim(Object_1, Object_2) = \begin{cases} 1 & if\ Object_1 = Object_2 \\ 0 & if\ Object_1 \neq Object_2 \end{cases} \quad (3)$$

between two values and changes this distance to a similarity value. The maximum interval between these two values should also be known to calculate the value of similarity. The mathematical formula is

$$sim(Double_1, Double_2) = 1 - \frac{|Double_1 - Double_2|}{Max_{Interval}} \quad (4)$$

$Double_1$ represents the data type of the first value with a type, "Double" and $Double_2$ represents the data type of the second value and is also "Double". The difference between $Double_1$ and $Double_2$ is calculated as the absolute value of subtracting $Double_1$ from $Double_2$. $Max_{Interval}$ is the maximum interval between $Double_1$ and $Double_2$.

3.2.3 String Similarity Function

MaxString is a method based on longest common substring algorithm. The value of similarity is to find the maximum common string length in two string and then divide by the maximum string length of two string.

3.2.4 Semantic Similarity Function

Information Content (IC) is a technique in information theory to measure semantic similarity between two words. The following formula calculates an IC value in WordNet [13]:

$$ic_{wn} = 1 - \frac{\log(hypo(c)+1)}{\log(\max_{wn})} \quad (5)$$

Where $hypo(c)$ is the number of hyponym of c; $\max_{wn}$ is a constant and represents the maximum amount of concept categorization. The semantic similarity formula is

$$\begin{aligned}
& IF\ sim_{ASVP} = 0.95 \quad THEN\ 0.95 \times 5 = 4.75 \\
& \Rightarrow 4.75 - 5 + 1 = 0.75 \\
& IF\ 0.75 > sim_{C1} \\
& \Rightarrow 1.75 = 0.75 + 1 > sim_{C1} + 1 \geq sim_{C1} + sim_{C2} \cdots (a) \\
& \Rightarrow 2.75 = 1.75 + 1 > sim_{C1} + sim_{C2} + 1 \\
& \qquad \geq sim_{C1} + sim_{C2} + sim_{C3} \cdots (b) \\
& \Rightarrow 3.75 = 2.75 + 1 > sim_{C1} + sim_{C2} + sim_{C3} + 1 \\
& \qquad \geq sim_{C1} + sim_{C2} + sim_{C3} + sim_{C4} \\
& \Rightarrow 4.75 = 3.75 + 1 > sim_{C1} + sim_{C2} + sim_{C3} + sim_{C4} + 1 \\
& \qquad \geq sim_{C1} + sim_{C2} + sim_{C3} + sim_{C4} + sim_{C5} \\
& \qquad = sim_{ASVC} \times 5 \\
& \Rightarrow sim_{ASVP} > sim_{ASVC}
\end{aligned}$$

List 1 Similarity Calculating Process

$$sim_{res'}(c_1, c_2) = 1 - \frac{\log(hypo(MSCA(c_1, c_2)) + 1)}{\log(\max_{wn})} \quad (6)$$

$$sim_{jcn}(c_1, c_2) = 1 - \left(\frac{ic_{wn}(c_1) + ic_{wn}(c_2) - 2sim_{res'}(c_1, c_2)}{2} \right) \quad (7)$$

Where $MSCA(c_1, c_2)$ is Most Specific Common Abstraction of c_1 and c_2.

From the experimental data in [15], we find out that the similarity of nouns and verbs are better than the similarity of adjectives and adverbs because that the amount of nouns and verbs are larger than the amount of adjectives and adverbs, and each noun or verb always has a hypernym and hyponym. Nouns and verbs are better than adjectives and adverbs to be the characteristic of a class.

3.3 Case Adaptation Methods

JCBRDT uses Jess [4,12] to modify the cases. There are three parts in Jess Rule of JCBRDT:

1. Knowledge Base: the methods about loading the needed domain knowledge;
2. Initial Value: the methods about retrieving the similar cases and the description part in question.
3. Rule: the method about adapting cases.

From these three parts, system developer could use the adapting knowledge built by Protégé [1].

3.4 XML in JCBRDT

JCBRDT defines three XML Schemas: Case Structure, Case Base and Connector. These XML Schemas are Java objects mapped by XML binding techniques. JCBRDT use these XML Schemas to produce a domain dependent XML files which contain the needed information about a CBR system, such as the title and data type of characteristics, and the similarity function.

3.4.1 Case Structure

XML Schema in Case Structure [3] defines the structure of a case as shown in Figure 2. There are two parts in case structure: Description Part and Solution Part. The attribute, CaseSim, in the Description Part represents thewhole similarity of a case. Because different user requirements need different similarity functions, system developers can extend the class and design the similarity function. The attribute, DPAttribute, represents a characteristic of each case. The attribute, AttributeSim, has the similarity function of each characteristic. The attribute, Name, is the title of characteristics, Type the data type of characteristics, and Weight the weightiness of characteristics. The Solution Part is the same with the Description Part without LocalSim and Weight because it does not need to calculate similarity.

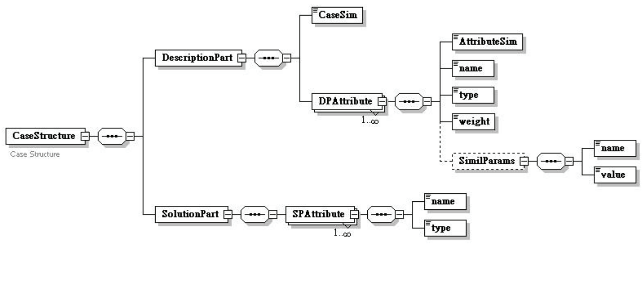


Figure 2 Case Structure

3.4.2 Case Base

The purpose of Case Base XML Schema is to create a mapped case base object as shown in Figure 3. JCBRDT maps the content of Text or XML files into the related Java object. The CBR system can only access the data in memory and does not open/close the Text or XML files at every moment. It will increase the executing speed of CBR system.

3.4.3 Connector

JCBRDT supports four types of case base: MySQL, Text, XML and ConnectorClass, as shown in Figure 4. The ConnectorClass case base represents the case base created by Java objects directly. The MySQL case base storing cases in hard drive has large capacity but slow accessing speed. Text, XML and ConnectoClass case bases storing cases in memory have fast accessing speed but small capacity. The cases in Text case base are easier to be modified, added and deleted than the ones in XML case base. However, the cases in Text case do not have error checking and correction since they lack a case content structure. The cases in XML case base could be verified after the modification, addition or deletion. The ConnectorClass method is more independent and needs fewer resources because it does not need an external system or file to store cases. However, it needs system developers to create every case into case base with Java programming. It will increase the developing load of system developers and maintain loading of case base, and it could not save the new cases in the executing process of system.

JCBRDT stores the case base connecting method into the Connector XML Schema file. From this file, the system

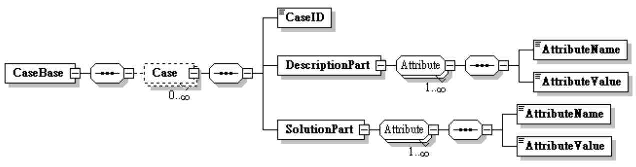


Figure 3 Case Base

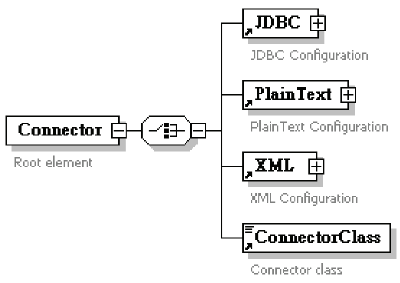


Figure 4 Connector

developer will know the connecting method and modify this if needed.

4 Implementation

The system architecture is described in this section and explained by an example. The comparison with other development tool is also shown in this section.

4.1 System Architecture

There are two main sub-systems in JCBRDT: CBR Frame Generation (CBRFG) and CBR Core (CBRCore). The functionality of CBRFG is letting system developer use a graphical interface to setup related data about CBR system: case structure, case base storing method and rule adaptation method. The graphical interface is based on the functions provided by CBRCore. The purpose of CBRFG is creating a required system of system developer by using simple operations. The CBR application created by CBRFG is a shell and the CBRCore is its reasoning kernel which contains components needed in CBR life cycle.

4.2 Example: Pizza Recommendation System

We use a pizza recommendation system to explain the utilization of JCBRDT. A system developer uses JCBRDT to create a pizza recommendation system which recommends what kind of pizza is suitable for the consumer who enters the personal information about his interests. We use an ontology about pizza modified from the pizza case study in Protégé [1].

The JCBRDT process to create a pizza recommendation system is shown in Figure 5. First, a system developer checks if there is any pizza recommendation system in JCBRDT. If the answer is "no", the system developer uses CBRFG to design a pizza recommendation system. By executing the functions, "Set Case Structure", "Set Case Base Source" and "Set Case Adapt Rule", CBRFG creates three objects, "Case Structure Information", "Case Base Source" and "Case Adapt Rule". CBRFG completes the method "Generate CBR Source Code" and creates a pizza recommendation system. Second, a consumer interacts with this system and inputs a problem description about pizza. This problem description fires an event in CBRCore. CBRCore retrieves the most similar case to the consumer's problem description in the case base and adapts it to a suitable solution. Finally, the consumer receives this solution from the pizza recommendation system.

The execution result of the pizza recommendation system is shown in Figure 6. A consumer input his personal information: his name is Henry_Yang; his sex is male; his age is 18; his occupation is student; he is not an vegetarianism; his degree of spiciness is mild and his negative topping is TomatoTopping. The retrieved most similar case is that the name is Herry_Teng; the sex is male; the age is 30; the occupation is sailor; the man who is not a vegetarianism; the degree of spiciness is mild and the negative topping is TomatoTopping. The recommended pizza is American.

In Figure 7 we see that the toppings in the original case are TomatoTopping, PeperoniSausageTopping and MozzarellaTopping and the toppings in the adapted solution are PeperoniSausageTopping and MozzarellaTopping. Because TomatoTopping is the consumer's negative topping, it was deleted from the original case by the pizza recommendation system.

4.3 Comparison with Other Tool

The CBR system created by jCOLIBRI only retrieves the most similar case, and it is a null adaptation; JCBRDT uses Jess for case adaptation and it is a rule adaptation. JCBRDT uses Protégé to create adaptation knowledge bases and let the solution part of cases match system developers' requests. Users or other systems can directly use the solution from CBR systems built by JCBRDT.

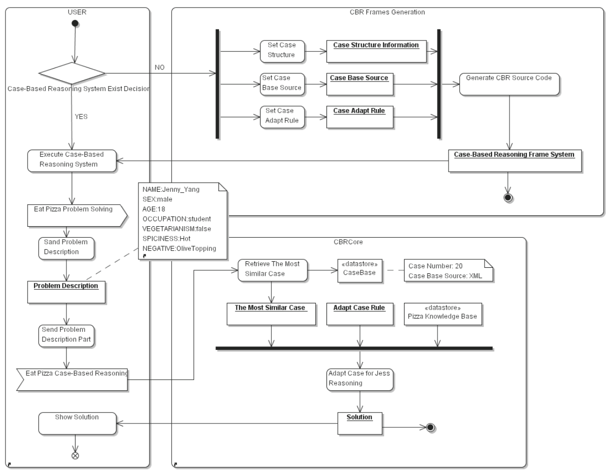
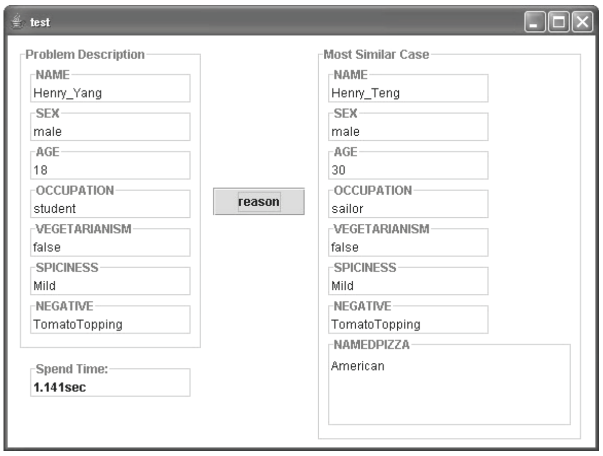


Figure 5 Activity Diagram of Creating Pizza Recommendation System



Problem Description

NAME	Henry_Yang
SEX	male
AGE	18
OCCUPATION	student
VEGETARIANISM	false
SPICINESS	Mild
NEGATIVE	TomatoTopping

Spend Time: 1.141sec

Most Similar Case

NAME	Henry_Teng
SEX	male
AGE	30
OCCUPATION	sailor
VEGETARIANISM	false
SPICINESS	Mild
NEGATIVE	TomatoTopping
NAMEDPIZZA	American

Figure 6 The Result of Pizza Recommendation System

Table 1 Systems Comparison Chart

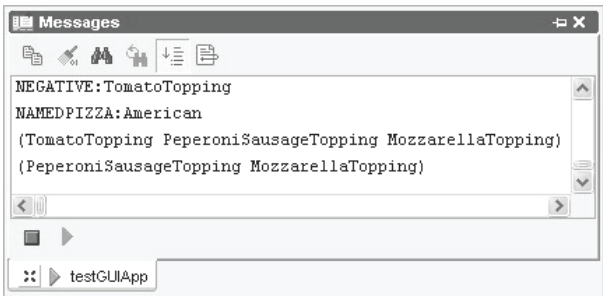
Items \ Systems	JCBRDT	jCOLIBRI
Using XML to represent CBR system information	Yes	Yes
Adapting cases automatically	Yes	No
Using Ontology to assist case adaption	Yes	No
Sources of case base	XML, MySQL, Text	MySQL, Text
Supporting semantic similarity functions	Yes	No

For the source of case base, JCBRDT provides a XML case base. The reason of using XML is to enhance the drawback of Text. A Text case base is easy to be used and has a high speed in adding cases, but it is difficult to modify the content of cases.

Semantic similarity functions in JCBRDT let system developers add semantic characteristics in cases. For example, system developers can semantic similarity function to compare the semantic similarity between "student" and "author". The systems comparison chart is shown in Table 1.

5 Conclusions

Our research shows that a domain independent CBR system development tool is feasible. There is a common view in CBR executing process but there are different views in CBR techniques for different application domains. Let us consider some directions for the future of JCBRDT. First, a CBR system could be used in many domains, but the similarity functions provided by the current JCBRDT could not satisfy all of system developers' needs. It is our goal to apply JCBRDT to more application domains to gain experiences in designing more similarity functions. Second, for case structure, JCBRDT constructs a case with a description part, a solution part and characteristics, but some system developers need a more complex case structure. Providing a method to let system developers to define case structure themselves will be a good function to add. Third, JCBRDT provides only one method in case retrieval and adaptation. There are other methods, such as using index and hierarchy to shorten the case retrieval time. CBRDT will provide more methods about case retrieval and adaptation for system developers to choose the most suitable method by their own requests.



```
NEGATIVE:TomatoTopping
NAMEDPIZZA:American
(TomatoTopping PeperoniSausageTopping MozzarellaTopping)
(PeperoniSausageTopping MozzarellaTopping)
```

Figure 7 The Adapted Solution in Pizza Recommendation System

Acknowledgement

This research was partially supported by the Ministry of Economic Affairs under the grant No. 96-EC-17-A-02-S1-029 and the National Science Council under the grants No. 97-2752-E-008-001-PAE and 97-2221-E-194-048-MY3.

References

[1] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang and C. Wroe, *OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns*, 14th International Conference on Knowledge Engineering and Knowledge Management, Whittlebury Hall, UK, 2004, pp. 63-81.

[2] A. S. Rao and M. P. Georgeff, *BDI Agents: From Theory to Practice*, First International Conference on Multi-Agent Systems, San Franciso, USA, 1995, pp. 312-319.

[3] A. Sanchez, J. A. Recio, B. Diaz-Agudo and P. Gonzalez-Calero, *Case Structures in jCOLIBRI*, Twenty-fith SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, 2005.

[4] E. F. Hill, *Jess in Action: Rule-Based Systems in Java*, Manning Publications, 2003.

[5] G. A. Miller, *WordNet : A Lexical Database for English*, Communications of the ACM, Vol. 38, 1995, pp. 39-41.

[6] J. A. R. Garcia, A. Sanchez, B. Diaz-Agudo and P. A. Gonzalez-Calero, *jCOLIBRI 1.0 in a nutshell. A software tool for designing CBR systems*, Proccedings of the 10th UK Workshop on Case Based Reasoning, 2005, pp. 20-28.

[7] J. J. Jiang and D. W. Conrath, *Semantic similarity based on corpus statistics and lexical taxonomy*, Proceedings of International Conference Research on Computational Linguistics, Taiwan, 1997, pp. 19-33.

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

[9] J. M. Corchado and R. Laza, *Constructing Deliberative Agents with Case-based Reasoning Technology*, International Journal of Intelligent Systems, Vol. 18, No. 12, 2003, pp. 1227-1241.

[10] J. M. Corchado, J. Pavon, E. Corchado and L. F. Castillo, *Development of CBR-BDI Agents: A Tourist Guide Application*, 7th European Conference on Case-based Reasoning, 2004, pp. 547-559.

[11] M. Glez-Bedia and J. M. Corchado, *A Planning Strategy Based on Variational Calculus for Deliberative Agents*," Computing and Information Systems Journal, Vol. 10, No. 1, 2002, pp. 2-14.

[12] M. Menken, *Jess Tutorial*, Vrije Universiteit, Amsterdam, The Netherlands, 2002, pp. 1-57.

[13] N. Seco, T. Veale and J. Hayes, *An Intrinsic Information Content Metric for Semantic Similarity in WordNet*, Proceedings of the 16th European Conference on Artificial Intelligence, 2004, pp. 1089-1090.

[14] P. Resnik, *Using Information Content to Evaluate Semantic Similarity in a Taxonomy*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 448-453.

[15] P. Zhang, *The Research and Implementation of Semantic Based Web Services Discovery*, Knowledge Engineering Group, Tsinghua University, 2005, pp. 36-45.

Biographies

Ken Yen-Ru Cheng received the BS degree in Electrical Engineering from the Tatung Institute of Technology in 1994 and the MS degree in Electrical Engineering from the National Chung Cheng University in Taiwan in 1996. He is a Ph.D. student in the Electrical Engineering Department at National Chung Cheng University. His research interests are artificial intelligence, software engineering, intelligent agents, requirements engineering, and embedded software engineering.

Chiung-Hon Leon Lee received the Ph.D. degree in Electronic Engineering from the National Chung Cheng University in Taiwan in 2006. He is an assistant professor at Department of Computer Science and Information Engineering, Nanhua University, Taiwan. His research interests are in agent-based software engineering, Web services, knowledge representation, and fuzzy time series.

Alan Liu received the Ph.D. degree in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1994. He is a professor at Department of Electrical Engineering, National Chung Cheng University in Taiwan. His research interests in artificial intelligence and software engineering include knowledge acquisition, requirements analysis, intelligent agents, and applications in embedded systems and robotic systems. He is also a member of IEEE, ACM, and TAAI.