



Foolproof Technique in Case-Based
Reasoning Agent for Price
Negotiation

By

Mr. Narong Tantayanon

Submitted in Partial Fulfillment of the
Requirement for the Degree of

Master of Science

in Computer Science
Assumption University

October, 2003

Foolproof Technique in Case-Based Reasoning Agent for Price Negotiation

By

Mr. Narong Tantayanon



**Submitted in Partial Fulfillment of the
Requirement for the Degree of
Master of Science in
Computer Science
Assumption University**

October , 2003

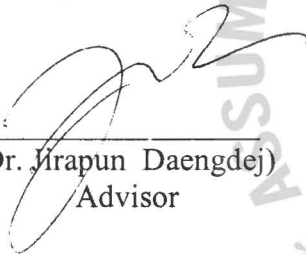
The Faculty of Science and Technology

Master Thesis Approval


Thesis Title	Foolproof Technique in Case-Based Reasoning Agent for Price Negotiation
By	Mr. Narong Tantayanon
Thesis Advisor	Dr. Jirapun Daengdej
Academic Year	1/2003

The Department of Computer Science, Faculty of Science and Technology of Assumption University has approved this final report of the **twelve** credits course. **SC7000 Master Thesis**, submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

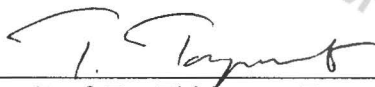
Approval Committee:




(Dr. Jirapun Daengdej)
Advisor



(Asst. Prof. Dr. Pratit Santiprabhob)
Committee Member

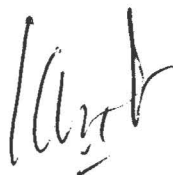


(Asst. Prof. Dr. Thitipong Tanprasert)
Committee Member

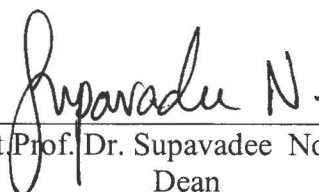


(Asst. Prof. Dr. Surapong Auwatanamongkol)
Commission of Higher Education
University Affairs

Faculty Approval:



(Asst. Prof. Dr. Tang Van To)
Program Director



(Asst. Prof. Dr. Supavadee Nontakao)
Dean

October / 2003

ACKNOWLEDGEMENTS

Thank God for everything as my heart is stirred by a noble theme as I recite my verse for the king; my tongue is the pen of a skillful writer. In your majesty ride forth victoriously in behalf of truth, humanity and righteousness.

I appreciate Dr. Jirapun Daengdej for his encouragement and advice throughout the course of my graduate work, and his guidance in the preparation and writing of this thesis.

Most of all, I am blessed with the support of my parents who place an enormous amount of importance on education. Their love and encouragement, as well as vision and far-sightedness allow me to make my personal and professional choices in my life. Without the support from my parents, I would not have had the opportunity to make this academic endeavor, and eventually to realize my educational goal. I dedicate this work to them.

ABSTRACT

The Case-Based Reasoning (CBR) has been widely used in the Artificial Intelligent (AI) application in order to mimic the human's intelligence by storing and retrieving the collected experiences and adapting them for the solution. This paper focuses on the CBR agent in the negotiation-based environment, in which an agent is possibly lying in order to achieve its goal. Researcher simulates the liar environment by created bargaining domain name "ALTAR". The domain consists of two main players, the buyer and the seller. The lying agent declared as the seller and the trial agent acknowledged as the buyer.

Research studied the behavior of the conventional CBR agent VS lying agent. The result showed the high learning ability of the agent but on the other hand the agent does not have any protection from lying agent. Therefore, this is the drawback of CBR agent that researcher would like to improve.

The idea the researcher forms to solve the problem is to create the self-protection mechanism for Case-based Reasoning agent that can protect itself from noise or liar quote but still could preserve their learning ability, which is the fundamental function of Artificial Intelligence. The researcher names this self-protection CBR agent as "Foolproof CBR agent."

Keywords: CBR, Case-Based Reasoning, Foolproof, CBR Agent, Lying agent.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	i
ABSTRACT	ii
LIST OF FIGURES	iii
LIST OF TABLE	iv
CHAPTER	
CHAPTER 1 Introduction	1
1.1 Case-based Reasoning	2
1.1.1 CBR Cycles	3
1.1.2 One of CBR Drawbacks	7
1.2 What is Agent?	8
1.3 CBR in Agent	11
1.4 Why Agent Lie?	12
1.5 Objective of the thesis?	14
CHAPTER 2 Literature Review	16
2.1 CBR Agent	16
2.1.1 Cooperative Case-based Reasoning	16
2.1.2 Auction-based Retrieval	21
2.1.3 Knowledge and Experience Reuse through Communication among Competent (Peer) Agents	28
2.2 Negotiation Domain	30
2.2.1 Multi-Issue Negotiation under Time Constraints.	30
2.2.2 A framework for argumentation-based negotiation	33

2.2.3 Enabling Peer-to-Peer SDP (Service Discovery Protocol) in an Agent Environment	34
CHAPTER 3 Problem Domain	38
3.1 Registrar	39
3.2 Negotiation Model	42
3.3 Conventional CBR agent in ALTAR	45
3.4 GUI of the ALTAR	46
3.5 Agent Management	48
CHAPTER 4 Proposed Technique	49
4.1 The Conventional CBR Agent in ALTAR	49
4.2 Case Base Preparation	51
4.3 Refinement in Retain Process.	54
4.4 Maintenance Temporary cases to a Regulation case	55
4.5 Adaptation Technique in Revise step in CBR	55
4.6 Extend the Domain Model/ Negotiate Model.	56
4.7 Meta data modification	58
CHAPTER 5 Evaluation	60
CHAPTER 6 Conclusion & Future Research	62
BIBLIOGRAPHY	64

LIST OF FIGURES

Figure 1-1	Case-Based Reasoning Cycles	4
Figure 1-2	CBR Agent.	11
Figure 2-1	A case description in CHROMA [28]	18
Figure 2-2	The case-based reasoning method in CHROMA. [28]	18
Figure 2-3	Task decomposition of case-based reasoning method for diabetes therapy. The retrieve task will be augmented with communication capabilities for searching cases solved by other agents.	30
Figure 2-4	Negotiate Protocol.	34
Figure 2-5	Cluster-based Architecture.	37
Figure 3-1	The procedure of ALTAR	38
Figure 3-2	Negotiate state diagrams, refer from [31]	42
Figure 3-3	CBR components & domain model	45
Figure 3-4	Experiment simulation users interface	47
Figure 3-5	the negotiation between agents	47
Figure 4-1	Negotiation between Conventional CBR agent in ALTAR	50
Figure 4-2	The Case Base model	52
Figure 4-3	Extends the domain/negotiate model.	57
Figure 5-1	Negotiation between foolproof CBR agent and Lying agent	60

LIST OF TABLE

Table 6-1	The stock exchange based negotiate sample	63
-----------	---	----



1. Introduction

“What has been will be again, what has been done will

be done again; there is nothing new under the sun.

Is there anything of which one can say, “Look? This is something new?”

It was here already, long ago; it was here before our time.”

[Holy Bible, Ecclesiastes 1:9-10]

Humans innovate through their experience. Einstein came up with his greatest theories while sailing through the waves. Edison, a man with over 1,000 patents to his credit, would go down to the dock and fish for ideas in the nature. Robert Lutz of Chrysler Corporation was enjoying driving sports car when he conceptualized new hot V-10 powered Dodge Viper sport car. Leonardo da Vinci was inspired to fly by watching the flying creatures.

Likewise, we live and solve day-to-day problems based on our experience. This concept creates one of the Artificial Intelligence (AI) Techniques called **Case-based Reasoning** (CBR). CBR system uses existing experiences to solve a new problem. The CBR details are introduced in Section 1.1.

On the other hand, some complex and large scale problem could not be solved by a single AI. A problem can be redefined into parts, where each part is solved by one of AI technologies called Agent. Each agent has their own structure, objective, and responsible to some parts of problem. In general, several agents will cooperatively solve that unsolvable problem.

Agent technology brings us to a new level of convenience. Agent has been embedded in a technology like robots with self-learning mechanism proposed in AI community. This agent has self-intellectual ability to learn and judge in certain scope that had objective to fulfill the user desire(s). This agent technology and their application are introduced in Section 1.2.

This thesis focuses on the agent created using CBR algorithm, or what is so called **CBR agent**, in the negotiation-based environment. In Section 1.3, the detail of CBR agent mechanism is explained.

However, one of the CBR and CBR agent drawbacks is when it interacts with the deceitful information or noise, it can be contaminated easily. This is the starting point of the thesis. More details on the objective of this thesis can be found in Section 1.4.

"Make it a habit to keep on the lookout for novel and interesting ideas that others have used successfully. Your idea needs to be original only in its adaptation to the problem you are working on."

[Thomas A. Edison]

1.1 Case-Based Reasoning

Case-Based Reasoning (CBR) has been referred to as one of the most practical machine learning algorithms. CBR study is driven by two motivations [1]. First, CBR is regarded as *cognitive science*, which desires to model human behavior. Second, CBR is looked upon as one of the *Artificial Intelligence models* which attempt to

mimic the mankind's intelligence by restoring or reusing the existing experiences and adapting them to be the new solution.

Cognitive science uses CBR in the study of human reasoning that demonstrates reasoning from cases in a wide range of task contexts. For example, studies support the importance of the reminding of prior examples in learning a computer text editor, learning programming, mathematical problem solving. The study processes need developing and testing theories of how humans store, retrieve, and apply prior cases.

CBR is used in the *Artificial Intelligence* study in which humans are viewed as robust problem-solvers. We routinely solve problems and our performance improves with experience we gained.

The experience stored by CBR is likely to be in the format of database called “**case base**”. Theoretically, the case base could store any type of knowledge. CBR will solve the new problem by consulting the case base through the retrieval of the best-fit cases and adapting it to be the answer. Finally, the answer will be feedback into the case base to increase the experience for CBR. Now, we will look at the process of CBR process in more details.

1.1.1 CBR Cycles

The cycle of CBR has five processes [1] as shows in Figure 1-1. The supportive steps of CBR are Retrieve, Reuse, Revise, Review, and Retain. The case base and its process are linked with a domain model which describes the specifics of a certain domain and controls how to manage the system in that domain.

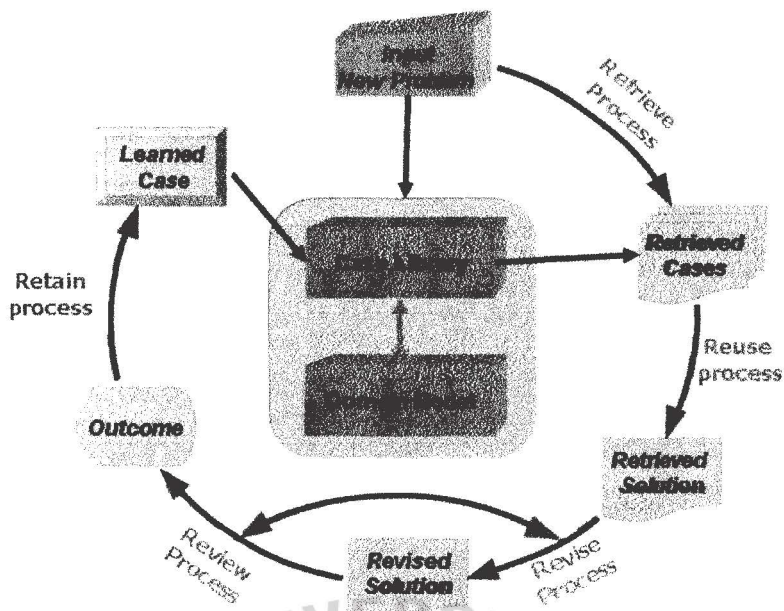


Figure 1-1: Case-Based Reasoning Cycles

1. **Retrieve:** After CBR obtains a new problem, CBR will retrieve the nearest matched case from case base. This process is completed with caution. The more cases there are, the longer time is needed to retrieve all matched cases. This is one of the most important CBR necessary factors to use the case base maintenance (CBM).
2. **Reuse:** Apply one or more solutions from these retrieved cases, perhaps by combining them with each other or with other knowledge sources. This process is matching case or combining some of cases to an answer that is close to the question.
3. **Revise:** Adapt the retrieved solution(s) as needed, in an attempt to solve the new problem. This is not the same as the Reuse step in the sense that it can adapt without any similarity from retrieved cases. For example, the retrieved cases are a

glass worth \$5.00 and a piece of paper worth \$0.50. The result may be equal to a paper glass cost \$3.50.

4. **Review:** Evaluate the outcome(s) when applying the constructed solution to the current problem. If the outcome is not acceptable, then the solution will require further revision.
5. **Retain:** Consider adding the new triplet of problem, revised solution, and outcome to the case base as a new case. This is the way CBR increases their experience.

CBR has been applied to many applications in various fields. The practical aspect is based on the idea that the tasks of work we normally do day after day is mostly repeat. For example, CBR is applied in the law enforcement context. CBR might get a new problem as *“How many years behind bars should it be for a thief who steals \$2,000 jewelry and kills 2 jailers?”*

In Retrieve step: CBR will retrieve two cases {Stealing \$2,000 jewelry is equals to 5 years in jail.} {Murdering an innocent person is equals to 30 years in jail.}.

In Reuse step: The retrieved cases do not exactly match the problem so it will be passed to the Revise step.

In Revise step: The system may emulate the year or use some method to adapt it to be {Stealing \$2,000 jewelry AND murdering two innocent persons is equal to 65 years in jail.}

In Review step: The outcome gives evidence to the user. The next step will be waiting for a response from the user. If the outcome is accepted by the system's user

then CBR will move on to the Retain step. Otherwise, the CBR will return to the Revise step.

In Retain step: CBR will save case {Stealing \$2,000 jewelry AND killing 2 innocent persons is equal to 65 years in jail.} to the case base. This new case base will be reused to solve the problem with this same question again. Moreover, in the next time, analysis will be replied faster as CBR find the matched answer only in reuse step.

Another example is to assume that you are the customer service officer in a printer company, since the company continues business for a while. You will face the question like *can not print, can not load paper, blur printing, printer not response* and etc which are the basic troubles that will annoy you time after time. You have to repeat the same answer again and again if you do not provide any system to handle those tasks.

For this domain of problem, the solution is using the support system. The support system gathers the answers that are positively true as the base knowledge to respond to those questions. Practically, there are two ways to handle it.

1. Passive way: to provide the basic knowledge as the trouble shooting with answer guide for users by hard copy, soft copy, the Internet, or etc.
2. Active way: use the support system that allows users to be cooperative, which is the concept of CBR. For example on the “Microsoft Support” website, after user asks a question to the system it will match and retrieve the answers, it has provided by ranking (the higher rank is assumed to be closer to the question). After user read through each answer, user may freely comment the rate of enlightenment for the question. The reply will

feed back to the system and the rank of answer to that question is higher.

So, the answer will refine the rank to shift closer to the exact question.

1.1.2 One of CBR Drawbacks

As the CBR process showed, the major component of CBR is cases in case base. Those cases represent the experience in CBR which CBR used to solve the problems.

A good example of the significance of cases in CBR was practically performed in Richard Heider project, called Cassiopee. He developed a decision support system for the technical maintenance of Boeing 737 jets engines. The objective of this project was to create a corporate memory of troubleshooting knowledge. 30,000 cases were obtained from a database of engine failure descriptions. Each failure report case contained as:

- i) Failure symptom such as high oil consumption, abnormal noise, thrust deficiency, and etc.
- ii) Faulty equipment showing a list of engine parts that are needed replacing or maintaining
- iii) Free form text narrative describing the failure event.

With careful case-by-case selection, 1,500 cases from 30,000 cases were chosen by a specialist (can not separate job to more person as cases may duplicate or conflict) as being representative of the range of engine failures. These became Cassiopee's case base. This is the hardship of the case base start up. Because CBR

researcher knows the significant of case base that does not expect any wrong case included in case base. This can be explained by example. Imagine that if one wrong case, such as “*engine temperature overheats to 320 degree centigrade solved by speed up engine*” instead of “*slow down engine*”, has been included into the case base then it will be uncountable destruction.

In the next Section, the introduction of agent will show the basic concept of agent.

1.2 Agent

The term “agent” can be simply illustrated visually. We start from depicting two situations in the regular daily life in the near future.

The central air-traffic control system of Thailand suddenly fails due to the bomb from terrorists. Providentially, air-traffic control agent in neighboring countries had detected the failure. The agent then clones itself and attends to the failed system until the system recovers.

You run out of the office, while the email message has arrived. It contains notification about accepting the budget for your oversea business convention. The mail agent predicted that you would like to see it as soon as possible. Agent begins to look into travel arrangements, consulting a number of databases and other networked information sources, interact with ticket selling agents, step into the virtual market space or maneuver along traveling websites. Agent will learn and collect that knowledge until some triggering point then agent will analyze the data and give out the best solution to the user.

These scenarios can be satisfied in the future as the foundation of the key components known as “agents”.

The question “what an agent is?” had been depicted by Carl Hewitt (at the Thirteenth International Workshop on Distributed AI). He recently remarked that the question “*what is an agent?*” is not an easy definition for the agent based computing community in just the same way that the question “what is intelligence?” is to the core AI community. The problem is that while the term is widely used by many people who work in closely related areas, it defies attempts to produce a single universally accepted definition. Each researcher is hoping to explicate his or her use of the word “agent.” Let us orient ourselves by examining and comparing some of these definitions.

The AIMA Agent [2] “*An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.*”

AIMA (Artificial Intelligence: a Modern Approach) definition depends heavily on what the researcher takes as an environment, and on what sensing and acting mean. Thus, if the researcher wants to make contrast between agent and program, the researcher must restrict at least the notions of sensing and acting upon environment.

The Maes Agent [3] “*Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed.*”

Pattie Maes from MIT's Media Lab defines an agent as systems which act autonomously to "realize a set of goals". The word autonomous sense in agent technology means that it should operate without any intervention.

"Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires." [IBM's Intelligent Agent Strategy white paper]

This definition, from IBM's Intelligent Agent Strategy white paper, views an intelligent agent as acting for another, with authority granted by the other. It is autonomously working that employed by some knowledge or intelligent to fulfill their goal.

All of these definitions can show us the rough specification of the agent. The basic functions that specify the agent was discussed in Michael Wooldridge & Nicholas R. Jennings paper [4] as follows:

- Autonomy: agents can operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- Social ability: agents can interact with other agents via some kind of agent-communication language.
- Reactivity: agents can perceive their environment and respond in a timely fashion. Agents run as Artificial Intelligent that it learn to adapt to the new environment. Environment may be the physical world, a user via a

78647 e.4

graphical user interface, market place where gather with agents, the Internet)

- Pro-activeness: agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviors by winning the plan.

1.3 CBR in Agent

The previous Section had described the basic idea of agent. Now we will look into the core intellectual that agent uses to make it wise. Agent created using the AI technique(s), which could be Case-Based Reasoning, Rule-Based Reasoning, and Model-Based Reasoning. However, in this thesis, we only focus on CBR as based reasoning engine in agent called "CBR Agent". Figure 1-2 shows the detail of CBR agent components.

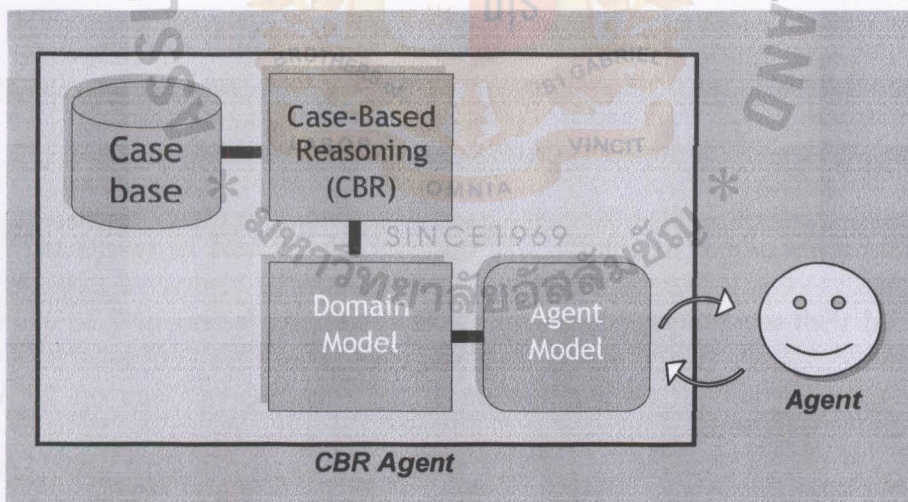


Figure 1-2 CBR Agent.

CBR is cooperatively working with case base as knowledge. All of the CBR processes are generally driven by a domain model which specifies:

1. The retrieve algorithm that match the domain.
2. The adaptation method that well suit to generate the answer to the problem domain.
3. The feedback system that understands the domain and selectively chooses the experience to store back into the case base.

In CBR research, there have been no wide application of CBR into agent study.

The reason is because CBR intellectual are based on case base. The CBR agent had to carry the case base and travel along the network or market space. In general, CBR agent learn and insert their experience in case base, therefore overall agent size will increase which will slow down the performance of the maneuver along networks, thus performance's agent decline. Another reason refers to one of the major efficiency of the CBR which depends on case base size. The bigger size of case base needs more power and time for CBR to search.

1.4 Why Agent Lie?

"The survey of New York Times in "The attitude of people toward the second anniversary of 9-11 crisis" concluded that New York people want their life **back to normal** but no one can explain what is normal, maybe the normal does not exist anymore" [New York Times, 11 September 2003]

The world had changed since 9-11 crisis, the terrorists went into hiding while the terrorist's targets exhibitions and these could not be investigated. Preventive action through every single means may be the only way to response. For example, Researchers at Carnegie Mellon's Robotics Institute have developed armies of

intelligent agents that act as response teams able to spontaneously find, interconnect, and cooperate to overcome the chaos that impedes emergency responses to disasters. Moreover, they think that this is the threats to all the international security which have to change to include well orchestrated, we must add to our lines of defense superior information gathering and fusing capabilities. Visual recognition agents that can instantly perceive threats, relay information, and coordinate responses with teams of agents and humans, can proactively stop attacks before they occur, and respond instantly when they cannot be prevented.

Agent technologies offer the greatest opportunity for sifting through the mountains of often opposing information. This helps to supply the right information to the right decision makers, in time to avert and respond to crises.

Dr. Katia Sycara of the Intelligent Software Agents Lab in the Robotics Institute is conducting agent research for the Defense Advanced Projects Agency, Office of Naval Research, and the Air Force Office of Scientific Research, and is prepared to provide background information on technology challenges to emergency response, and the solutions offered by intelligent software agents.

A technology demonstration, including video, and computer graphic segments illustrating the role of intelligent agents in an emergency bomb response scenario, is available.

The mentioned activities showed agent in offensive manner however the threats may target the agent itself such as the lying agent. Since the hypothesis that all the agents are trustworthy, the cooperation was accepted without suspicion.

Look into the current world driven by economic and benefit, not only terrorist conspired in attacking the center of economics' sign but also intend to destroy the flow of economic activities. The agents in business domain such as bargaining,

auction, goods exchanges, service exchanges, and etc are the target of the lying agent. Lying agent will send out lying paradox, which may not only directly attack on price but also destroy the opposite integrity, or build up chaos.

Then the assumption of lying agent is the agent that spread out noise, which intend to conspire with the other agent. The noise is declared as the deceitful information that is sent out by the lying agent.

1.5 Objective of the thesis?

The objective of this thesis is to improve the conventional CBR agent in dealing with noise in price negotiation domain. Noise declared as the lying quoted from lying agent in the domain. When conventional CBR agent interacts with lying agent the deceitful information from lying agent will be remembered by the CBR agent (because one of specification of agent is to perceive their environment). Then, the liar cases would contaminate the experience stored in the case base, and finally the conventional CBR agent will fool to buy the overpriced product.

Thus the objectives of this thesis are:

1. Study the behavior of conventional CBR agent in comparison to Lying agent in price negotiation domain.
2. Propose foolproof technique that allows self protection for the conventional CBR agent from liar.

This thesis proposed a self defensive technique called “Foolproof” which is added into CBR agent. The words “Self-defensive” mean that this technique does not solve the problem in the domain, but just protect itself only. More details will be illustrated in Chapter 2.



2. Literature Review

This Section is the literature reviews to show the prior works. The reviews can be classified into 2 related fields: the CBR agent and negotiation domain.

2.1 CBR Agent

2.1.1 Cooperative Case-based Reasoning

Enric Plaza et al. [6] showed the necessities of agent abilities in cooperation and learning. An agent needs to learn whenever it lacks some knowledge to perform some task. In the same way, an agent needs to cooperate with other agents because it lacks some knowledge or capability to perform a task. However, the essential can be summarized as follows:

1. Improving individual performance.
2. Improving quality of solution.
3. Improving efficiency (mainly in speed of finding solutions).
4. Improving the scope of solvable problems (competence).

This paper focused the study on Homogenous/Peer/Learning agents. The Homogenous agents are the agents that have the same representation language. Moreover, the communications among the Homogenous agents do not require a translation phase. The Peer agents are capable of solving the task at hand. In other words, they may not be merely specialists at specific subtasks but they are capable to solve the overall task by themselves. The Learning agents are the agents that solve the task based on knowledge acquired by learning from their individual, usually divergent,

experienced in solving problems and cooperating with other agents. This paper construct the framework named as federated peer learning (FPL) framework that founded on Homogenous/Peer/Learning agents.

The problem solving solution of agents will be biased by their individual learning based on their differed experience, since different sets of problems will actually occur in different locations. Consequently, cooperation may profit from these biasing by improving the overall performance of the involved agents.

The above framework is running in the developed domain called CHROMA. CHROMA is the system that uses chromatography techniques to purify proteins from tissues and cultures. CHROMA includes two learning methods (a case-based method and an inductive method) and two problem solving methods (a CBR method and a classification method that uses the induced knowledge).

The interconnection communication in CROMA is using Noos language, which is developed by Enric Plaza and team at the Institute for Integrating Learning and Problem Solving. It is the reflective object-centered representation language designed to support knowledge modeling of problem solving and learning. It is constructed based on the task/method decomposition principle and the analysis of knowledge requirements for methods.

Domain knowledge is represented in Noos by a collection of feature terms describing and their relation for a given domain. Thus case description in CHROMA could be depicted as in Figure 2-1

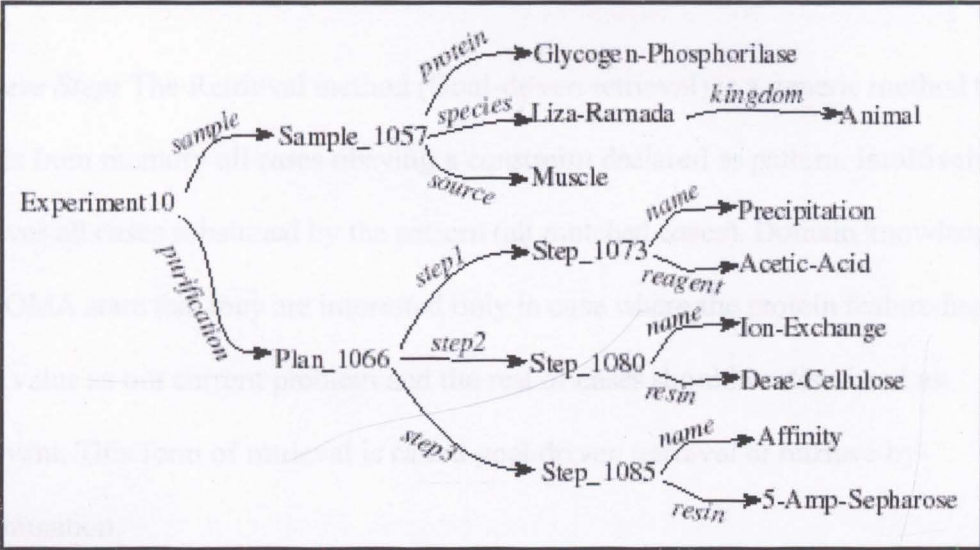


Figure 2-1 A case description in CHROMA [6]

The CBR model for this problem domain (Protein Purification) depicted in Figure 2-2.

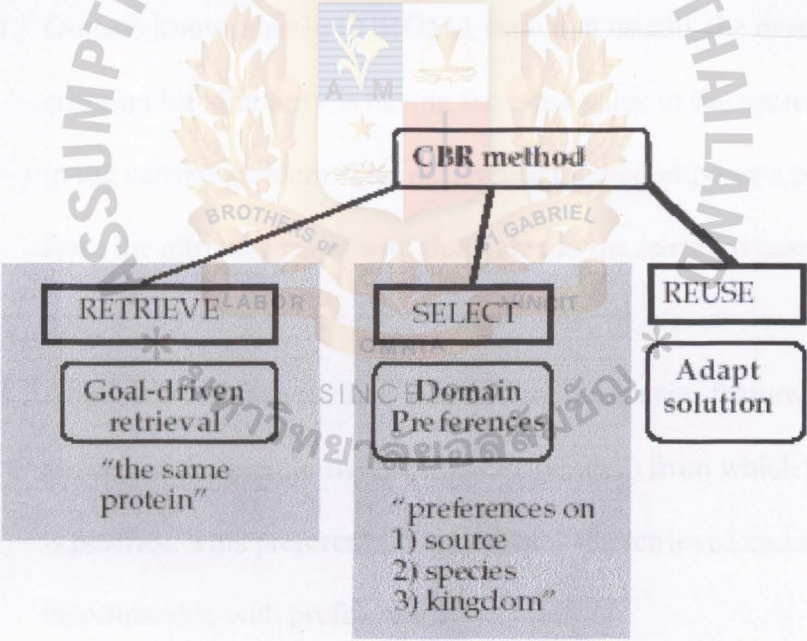


Figure 2-2 the case-based reasoning method in CHROMA. [6]

The configuration of the specific CBR method used in CHROMA is the following.

Retrieve Step: The Retrieval method (Goal-driven retrieval) is a generic method that selects from memory all cases obeying a constraint declared as pattern. Intuitively, it retrieves all cases subsumed by the pattern (all matched cases). Domain knowledge in CHROMA state that they are interested only in case where the protein feature has the same value as out current problem and the rest of cases should be dismissed as irrelevant. This form of retrieval is called goal-driven retrieval or retrieve-by-determination.

Reuse Step: A second component is a preference method that allows imposing a partial order among retrieved cases. In CHROMA there are three basic preferences:

1. Domain knowledge in CHROMA state that usually the most important criterion for similarity is having the same value in the source feature as in the current problem. This preference method imposes a partial order from the retrieved cases with that value to the retrieved cases that do not.
2. Another preference method is regarding the species feature – i.e. the species of the sample tissue or culture (source) from which the protein is purified. This preference discriminates the retrieved cases that are incomparable with preference 1.
3. The final component is also a preference regarding the entity of the source, and it is applied to all retrieved cases that are not preferred among them by the preceding preference methods.

In this Paper, they have developed plural Noos language based on Noos. Plural provides a seamless extension of Noos that supports distributed scope and reference for all the basic constructs in Noos. A Plural Noos agent is a particular Noos application with a known address and with several acquaintances. An agent address is composed of one IP address, a port number, and one identifier. The last identifier is needed since more than one agent can coexist within the same Plural Noos process. The acquaintances of an agent are those agents whose address is known by the agent. Each Plural agent can have different acquaintances. If an agent A_i belongs to the acquaintances of an agent A_j , then A_j belongs to the acquaintances of A_i .

A Plural Noos agent can be involved in solving only one problem at a time. Each problem solving process has a different identifier. When a Plural agent is solving a problem only accepts requests related to the same process identifier. In this way, possible deadlocks are avoided. Other deadlocks caused by circularities inside the same problem solving process are detected by the Plural Noos implementation. When an agent A_i requires a service from another agent A_j , and this one is already busy solving another different problem, A_i receives a busy message. Then A_i decides to wait some time to request the service to A_j again or ask it to another member of its acquaintances.

Since learning is lazy in CBR systems, cooperation involves expanding the set of precedents to be used in similarity bases reasoning from the individual case base of a CBR agent to the case base of a collectivity of CBR agents. They argue that there are two general ways to do so: Distributed Case-based Reasoning (DistCBR) and Collective Case-based Reasoning (ColCBR). Intuitively, both DistCBR and ColCBR are based on solving a problem by reusing with the knowledge learned by other CBR

agents. The difference between both modes is regarding which similarity-based reasoning method is used.

DistCBR is based on an agent transmitting the problem and the task to be achieved to another agent, and the CBR method used is that of the receiving agent. In this sense, the CBR process is distributed since every agent works using its own method of solving problem.

ColCBR is based on an agent transmitting also the method that is to be used to solve that problem to another agent and that method will use the knowledge learnt by the receiving agent. In other terms, the originator is using the memory of the other agents as an extension of its own, as a collective memory, try means of being able to impose to other agents the use of the CBR method of the originator.

From the standpoint of implementing those cooperation modes, they said that DistCBR is supported by the foreign evaluation capability and ColCBR is supported by mobile methods (also called “Remote programming”) capability of Plural Noos.

2.1.2 Auction-based Retrieval

Francisco J. Martin & Enric Plaza [5] proposed a framework of CBR agent in auction domain. The agents are running in multi-agent environment that can cooperate with each other which he named “CoopCBR”. This work focus on distribute partial case base, using CoopCBR protocol as the retrieval roadmap. His assumption is that the agent is not applicable to find a good precedent on its own case base then it will summon the “auction” (mediator) to find an appropriate case owned by other agent.

Auctions are an attractive domain of interest for AI researchers. The transaction of online auctions in the internet, such as Auctionline, Onsale, InterAUCTION, eBAY and many others, is increasing steadily. This has established auctioning as a main-stream form of electronic commerce. Thus, agent-mediated auctions appear as a convenient mechanism for automated trading when multi-party negotiations are involved.

The on-line auctions may successfully reduce storage, delivery or clearing house costs in many markets. On the other hand, auctions are not only employed in web-based trading but also as one of the most prevalent coordination mechanisms for agent-mediated resource allocation problems (energy management, climate control, flow problems, etc).

Auction-Based Retrieval

In this paper, when a CBR agent A is not capable of finding a “good” precedent on its own case base, it will summon an auction in order to find an appropriate case owned by other agent(s). Thus, agent A will select the participant agents (that will play the role of bidders) and will conduct the auction process (playing the role of auctioneer). The good to be auctioned is the (hypothetical) benefit obtained for finding a “good” precedent. A CBR agent can use the same auction to sell more than one good. In this case, each good is auctioned in a different round. For the purpose of identifying and characterizing the ontological elements involved in an auction, an auction descriptor defines as follows:

Auction Descriptor: define an Auction Descriptor $A^{a,n}$ for the n th auction of auctioneer agent A as the 3-tuple (B_n^a, G_n^a, D_n^a) :

- $B^{a,n}$ is a non-empty, finite set of bidders' identifiers representing the set of participants in auction n th of agent A.
- $G^{a,n}$ is non-empty, finite set of goods' identifiers representing the set of goods to be auctioned in auction n th of agent A. Auction n th of agent A will have a round for each good in G_n^a .
- $D^{a,n}$ stands for instance of the particular bidding protocol dynamics descriptor to be deployed in auction $A^{a,n}$. A bidding protocol dynamics descriptor defines the different parameters followed by a CBR agent in an auction guided by a particular bidding protocol.

Note that for each good $g_i^{a,n}$ in $G^{a,n}$, agent A will generate a collection $R_i^{a,n}$ of offers then:

Offer: offer define as a pair $(c_{\pi i}, p)$ where $c_{\pi i}$ is a pattern for retrieval and p is a real number that represents the price that agent A is disposed to pay for finding a precedent case subsumed by retrieval pattern $c_{\pi i}$.

A collection $R_i^{a,n}$ of offers will be partially ordered according to agent A's preferences about offer patterns $Q = (c_{\pi 1} \dots c_{\pi i})$ and totally ordered with respect to offer prices p . Moreover, each good has associated a starting offer p_α the first pair (P, p) to be offered and a reserve offer P_ω the offer at which the good will not be sold and then withdrawn (the last offer generated).

Notice that p_α represents the most specific pattern and therefore the highest offer whereas p_ω , on the contrary, represents the most general pattern and therefore the lowest offer.

In an auction $A^{a,n} (B_n^a, G_n^a, D_n^a)$ each CBR agent $b_i^{a,n} \in B_n^a$ acting as a bidder receives offers containing the corresponding retrieval pattern used in its privately owned case base CB_i to search for good precedent cases.

An agent $b_i^{a,n}$ that has a precedent case C_i^r in CB_i offer r sends a bid to the auctioneer offering case C_i^r . Notice that it will only send a request for bid and not the actual case yet. Then, if the auctioneer accepts that bid, it will request case C_i^r . Once the auctioneer checks the properties of case C_i^r it will declare bidder $b_i^{a,n}$ the winner. Notice that whereas the auctioneer agent conducts most of the auction process: it selects which are the bidders, selects the offers to be sent and the order in which they are sent; and also takes charge of deciding whether the bidder offer is “good enough” for winning the round. The decision of when to submit a bid is up to the bidder agent, which can decide to submit a bid only when it considers appropriate.

In an auction $A^{a,n} (B_n^a, G_n^a, D_n^a)$ agent A keeps track of the credit that measures the credibility given to each bidder agent $b_i^{a,n} \in B_n^a$.

Credit: define the credit of an agent i as the mapping $C_i^{a,n} : B^{a,n} \longrightarrow IR$ that assigns to each buyer in $B^{a,n}$ his available credit during auction n^{th} of agent A .

The auctioneer will either increase or decrease a bidder's credit according to the bids it submits (i.e. the goodness of the cases sent). When an agent loses its credit

the auctioneer will not take into account its subsequent bids, will expel it from the current auction and probably will not summon it for further auctions.

Information Revelation: The auctioneer also decides the policy followed for information revelation (bidder identity, bids, collisions, expulsions, winner's bid, winner identity, etc.). In this paper, they distinguish two kinds of auctions: public and private according to the range of information made public by auctioneer agent. In public auctions, all participants receive information concerning all other participants while in private auctions a bidder will only receive information concerning itself. Notice that whereas private auctions can diminish communications overhead, information revealed during public auctions can be acquired and used by involved agents. That is, agents can later address directly subsequent problems and lessen communication costs.

Auction Scheduling: Once a CBR identifies the potential for calling for an auction. It will announce to all chosen participants indicating the starting time and the auction descriptor (according to the character of the auction: public or private). Afterwards, interested bidders have to send a request for admission. Similarly, agent society can agree more than one auction, it is up to agents to establish starting times and take part in more than one auction at a time. When auctioning a good, one could choose among a wide range of bidding protocols (DBP, UBP, etc.). DBP stands for downward-bidding protocol or "Dutch" and UBP stands for upward-bidding protocol or "English". Each of these protocols can be characterized by a set of parameters that they refer to as bidding protocol dynamics descriptors, so that different instantiations

of such descriptors lead to different behaviors of their corresponding bidding protocols. In this paper, they concentrate on the downward bidding protocol (DBP).

Dutch Auction-based Retrieval (DBP)

1. The auctioneer chooses a good out of a set of goods.
2. With a chosen good g , the auctioneer opens a bidding round by quoting offers (namely retrieval patterns for the current problem and the corresponding resale price) downward from the good's starting offer p_α containing one of the most restrictive patterns.
3. The auctioneer waits for bids, but only a pre-established time slot Δ_{offers} . For each offer called by the auctioneer, several situations might arise during the open round:
 - **No bids** – No buyer submits a bid at the current price. If the reserve offers p_0 containing one of the least restrictive patterns has not been reached yet, the auctioneer quotes a new price. If the reserve offer is reached, the auctioneer declares the good withdrawn and closes the round. That is, the auctioneer has no more retrieval patterns and therefore the round is closed without any precedent case being retrieved.
 - **One bid** – Only one buyer submits a bid at the current price. The good is sold to this buyer whenever his credit can support his bid. Namely, the auctioneer asks the agent b_i to send its precedent case C_r^i such that $C_r^i \subseteq p$. If later this case turns out to be no good, $C_r^i \not\subseteq p$, i.e. there is an unsupported bid, the unsuccessful bidder is expelled out

from the auction, and the round is restarted by the auctioneer at a higher offer.

- **Multiple bids** – Several buyers submit their bids at the current offer. In this case, a collision comes about, the good is not sold to any buyer, and the auctioneer restarts the round at a higher offer. Nevertheless, the auctioneer tracks whether a given number of successive collisions (Σ_{coll}) are reached, in order to avoid an infinite collision loop. This loop is broken by randomly selecting one buyer out of the set of colliding bidders.

4. The first three steps repeat until the auctioneer has no more goods left.

Five parameters that control the dynamics of the bidding process are implicit in this protocol definition.

Dutch Dynamics Descriptor: Dutch Dynamics Descriptor define D_{DBP} as the 5-tuple $(\Delta_{\text{price}}, \Delta_{\text{offers}}, \Delta_{\text{rounds}}, \Sigma_{\text{coll}}, \Pi_{\text{sanction}}, \Pi_{\text{rebid}})$ such that

- $\Delta_{\text{price}} \in \mathbb{N}$ (price step). Decrement of price between two consecutive quotations uttered by the auctioneer.
- $\Delta_{\text{offers}} \in \mathbb{N}$ (time between offers). Delay between consecutive price quotations.
- $\Delta_{\text{rounds}} \in \mathbb{N}$ (time between rounds). Delay between consecutive rounds belonging to the same auction.
- $\Sigma_{\text{coll}} \in \mathbb{N}$ (maximum number of successive collisions). This parameter prevents the algorithm from entering an infinite loop as explained above.

- $\Pi_{\text{rebid}} \in IR$ (price increment). This value determines how the new offer is calculated by the auctioneer from the current offer when either a collision, or an unsupported bid occur.

Note that the identified parameters impose significant constraints on the trading environment. For instance, Δ_{offers} and Δ_{rounds} affect the agents' time-bound, and as a result the degree of situation viable for bidding strategies. Notice that buyers are expelled at the first unsupported bid that they submit. A factor could be used to expel a buyer only when a number of unsupported bids are reached.

2.1.3 Knowledge and Experience Reuse through Communication among Competent (Peer) Agents

Francisco J. Martin, Enric Plaza, and Joseph Luis Arcos [7] address the extension of the knowledge model approach. This work is also based on multi-agent with cooperative system. Additionally, they proposed the competence for each agent, the competence that could indicate the level of problem solving ability. Within this peer society agent, it could cooperate on solving one problem based on the competence. The agent may solve one problem by itself if its competence on that problem is high or it may send the request to other agents which have more competence for this specific problem.

A competent agent is an agent that:

- i. Is reflectively aware of its own competence, for a range of tasks, in solving problems.

- ii. Is aware of the competence of the other agents in a cooperative Multi Agent System. In this paper, they propose design of a competence agent that endowing it of competence models of itself. The competence models of acquaintance agent has to exist for any specific task (upon which cooperation will take effect) and only sense inside a given cooperation mode.

Competent Agent: Competent Agent is a tuple $A_i(M,A,C,S)$ with $M=M_1,...,M_n$ Cooperation modes, $A=A_1,...,A_m$ acquaintance agents, competence models $C=C_1,...,C_k$, and reflective competence models (or competence self-models) S .

Competence Model: Competence Model $C \in c$ of agent A_i is a tuple $C=(A,T,P,M)$ where $A \in a$, T is a task of A , P is a problem specification, and $M \in n$ is a cooperation mode. More over, a competence self model $S \in s$ is a tuple $S=(T,P)$.

Clearly, the competence model depends on the agent A to which a request is done and also on the task T requested to that agent. Moreover, the competence depends also on the type of request asked to A concurring T : this aspect is modeled by the notion of cooperation mode. Two cooperation modes for CBR are proposed in Figure 2-3, DistCBR and ColCBR: in DistCBR experience in the form of cases of A is shared by A_i but the problem solving knowledge used to solve T belongs to A ; conversely, in ColCBR experience in the form of cases of A is shared by A_i and the problem solving knowledge used to solve T belongs to A_i .

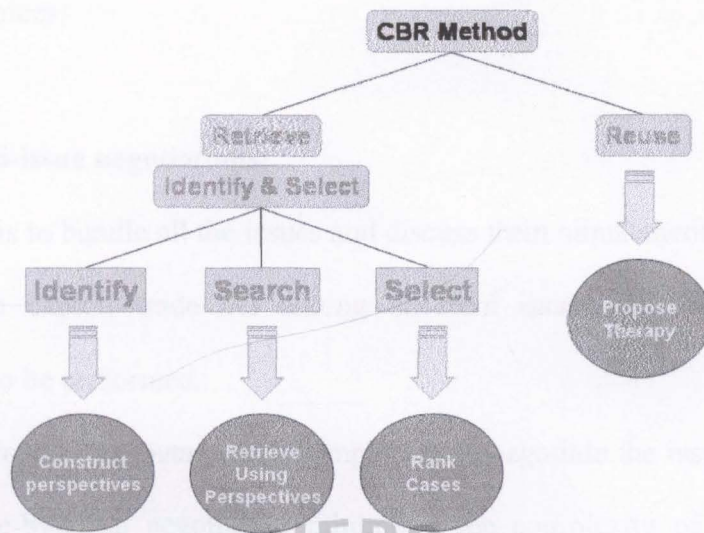


Figure 2-3 Task decomposition of case-based reasoning method for diabetes therapy. The retrieve task will be augmented with communication capabilities for searching cases solved by other agents.

2.2 Negotiation Domain

This section surveys the agent that is used in Negotiation domain.

2.2.1 Multi-Issue Negotiation under Time Constraints.

Shaheen S. Fatima et al. [8] presented a model for multi-issue negotiation under time constraints in an incomplete information setting. It is important that the agents should not only bargain over the price of a product, but also take into account aspects like the delivery time, quality, payment methods, and other product specific properties. In such multi-issue negotiations, the agents should be able to negotiate outcomes that are mutually beneficial for both parties. However the complexity of the bargaining problem increases rapidly as the number of issues increases. Given this increase in

complexity, there is a need to develop software agents that can operate effectively in such circumstances.

Bilateral multi-issue negotiations:

One approach is to bundle all the issues and discuss them simultaneously. This allows the players to exploit trade-offs among different issues, but requires complex computations to be performed.

The other approach (computationally simpler) is to negotiate the issues sequentially. Although issue-by-issue negotiation minimizes the complexity of the negotiation procedure, an important question that arises is the order in which the issues are bargained. This ordering is called the negotiation agenda. Moreover, one of the factors that determine the outcome of negotiation is this agenda. To this end, there are two ways of incorporating agendas in the negotiation model.

One is to fix the agenda exogenously as part of the negotiation procedure.

The other way (more flexible), is to allow the bargainers to decide which issue they will negotiate next during the process of negotiation. This is called an endogenous agenda. Against this background, this paper presents a multi-issue negotiation model with an endogenous agenda.

To provide a setting for our negotiation model, they consider the case in which negotiation needs to be completed by a specified time (which may be different for the different parties). Apart from the agents' respective deadlines, the time at which agreement is reached can affect the agents in different ways. An agent can gain utility with time, and have the incentive to reach a late agreement (within the bounds of its deadline). In such a case it is said to be a strong (patient) player. The other possibility is that it can lose utility with time and have the incentive to reach an early agreement.

It is then said to be a weak (impatient) player. As we will show, this disposition and the actual deadline itself strongly influence the negotiation outcome. Other parameters that affect the outcome include the agents' strategies, their utilities and their reservation limits. However, in most practical cases agents do not have complete information on all of these parameters. Thus in this work they focus on bilateral negotiation between agents with time constraints and incomplete information.

Single-issue model for negotiation –The Negotiation Protocol

This is basically an alternating offers protocol. Let B denote the buyer, S the seller and let $[P_{\min}^a, P_{\max}^a]$ denote the range of values for price that are acceptable to agent a , where $a \in \{b, s\}$. A value for price that is acceptable to both B and S , i.e., the zone of agreement, is the interval $[P_{\min}^b, P_{\max}^b]$. $(P_{\max}^b - P_{\min}^b)$ is the price-surplus. T^a denotes agent a 's deadline. Let $P_{b \rightarrow s}^t$ denote the price offered by agent b at time t . Negotiation starts when the first offer is made by an agent. When an agent, say S , receives an offer from agent B at time t , i.e., $P_{b \rightarrow s}^t$, it rates the offer using its utility function U^s . If the value of U^s for $P_{b \rightarrow s}^t$ at time t is greater than the value of the counter-offer agent S is ready to send at time t' , i.e., $P_{s \rightarrow b}^{t'}$ with $t' > t$ then agent S accepts. Otherwise a counter-offer is made. Thus the action A that agent S takes at time t is defined as:

$$A^s(t', P_{b \rightarrow s}^t) = \begin{cases} \text{Quit} & \text{if } t > T^s \\ \text{Accept} & \text{if } U^s(P_{b \rightarrow s}^t) \geq U^s(P_{s \rightarrow b}^{t'}) \\ P_{a \rightarrow b}^{t'} & \text{otherwise} \end{cases}$$

Within this context, they determined optimal strategies for agents but did not address the issue of the existence of equilibrium. Here we adopt this framework and prove that mutual strategic behavior of agents, where both use their respective optimal strategies, results in equilibrium. We then extend this framework for multi-issue negotiation between a buyer and a seller for the price of more than one good/service. Specifically, each agent has a deadline before which agreement must end on all the issues. However, the order in which issues are bargained over and agreements are reached is determined by the equilibrium strategies. These strategies optimize the time at which an issue is settled and are therefore appropriate for the sequential implementation scheme.

In this model the order in which issues are bargained over and agreements are reached is determined as part of the bargaining equilibrium. They show that the sequential implementation of the equilibrium agreement gives a better outcome than a simultaneous implementation when agents have time preferences. The proposed price defined in range $[P_{\max}, P_{\min}]$. The results show the chart between Price and time of the buyer and seller. The price may lift up for the first period and decline in the slope lower than the prior. Finally, the trends of both agents intersect at the point (deal point) they call Pareto-optimal.

2.2.2 A framework for argumentation-based negotiation

Carles Sierra et al. [9] describe a general framework for negotiation in which agents exchange proposals backed by arguments which summarize the reasons why the

proposals should be accepted. In such domains negotiation is essential to persuade others of the value of co-operation. The argumentation is persuasive because the exchanges are able to alter the mental state of the agents involved.

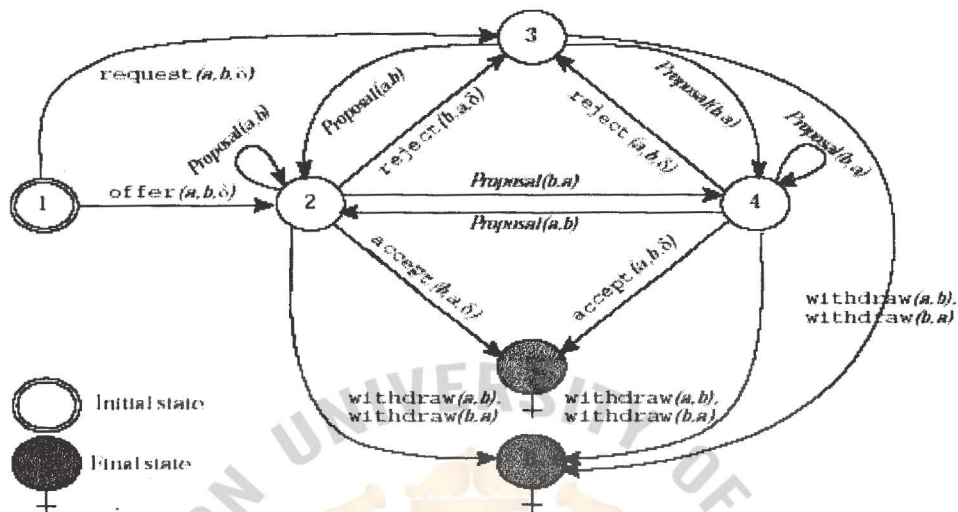


Figure 2-4 Negotiate Protocol.

Figure 2-4 shows the negotiate protocol. The $\text{accept}(x,y,\delta)$ and $\text{reject}(x,y,\delta)$ illustrations δ refer to last proposal. $\text{Proposal}(x,y)$ stands for any illocution constructed with any of the following particles: offer, threaten, reward, appeal between agents x and y .

2.2.3 Enabling Peer-to-Peer SDP (Service Discovery Protocol) in an Agent Environment

Naveen Srivasan and Timothy Finin [10] give an opinion that negotiation, cooperation and collaboration between multi-agents systems is the key success in the agent world. They view those agents that are capable to do the actions are offering the services. The main objective of this thesis is improving the discovering of those

services. This paper studies the Gnutella network (peer to peer network) which collaborates with Directories Facilitators (DF). This network do not support scalable with efficiency. Consequently, they proposed the scalable architecture (cluster-based architecture), which forms decentralized clusters of DF. These clusters are self governing, fault-tolerant and support global service discovery.

Cluster-Based Architecture

The existing DF should be modified to support infrastructure for global search. The several problems and the solutions in DF model will be showed:

Problem A: The environment is very dynamic. So, DF should try to return accurate search result. In the centralized model, this can be achieved by updating the central server, but this process will result in excessive traffic near the central node, in the distributed model similar to the Gnutella network, the DFs do not require forwarding update messages. Also, for search query, nodes perform a real time search and hence do not return any stale results. Thus a distributed model fits well in a dynamic environment so the DFs should also be modified to form a distributed network with other DFs in other agent platforms.

Problem B: The service registration and deregistration process is under the control of agents and not the DF. This can not depend on agents to deregister their services when they leave the agent platform. This will leave stale information in the DF. This problem can be solved by adding lease time with the service registration process. Lease time is the specific time period for which the service registration in DF is valid

after which DF deregisters the service. The agent has to re-register if it is still providing that service. Through this process has an overhead for agents in the platform, the DF will not have stale service registration.

Problem C: Malicious agent can register fake service in the DF. This problem can be handled by using a reputation mechanism. The DF should be responsible for hosting the reputation mechanism, so that, it can return the reputation of each agent in the search result. An agent, after receiving search result, can start interacting with the agents in the search results based on their reputations. After the interaction, the agents can send feedbacks to the DF about the agents that provided the service. The DF uses these feedbacks to modify the reputation of the agent that provided the service.

Problem D: DAML-S, an initiative by semantic web community, provides a core set of markup language constructs for describing the properties and capabilities of web services. It also provides facility to automate Web Service discovery, execution, interoperation, composition and execution monitoring. DAML-S ontology will enhance the searching capability of DFs. DFs need not perform complex operation like Matchmaker but perform some basic inference with DAML-S ontology. For example, if there is a composite service registered with the DF, then it should also register the services that form composite service individually, if they already registered.

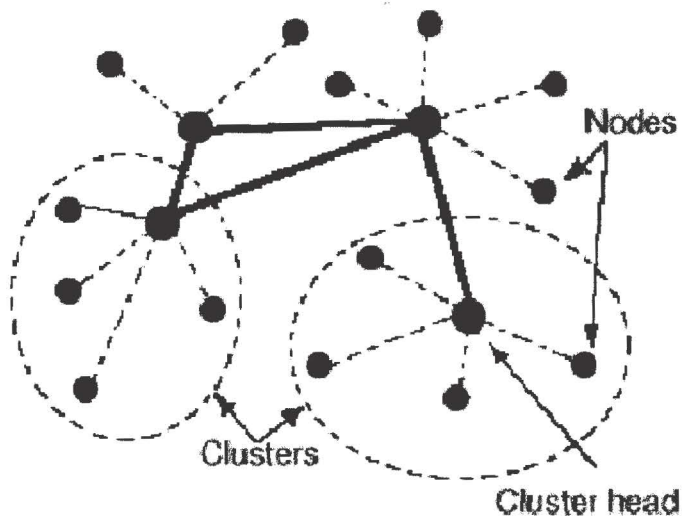


Figure 2-5 Cluster-based Architecture

Then this paper proposed a cluster based architecture that is **distributed**, self-governing and as faster query response time. Figure 2-5 shows the architecture. This paper assumes that DFs of each agent platform knows few other DFs, and form a weakly connected network. The board idea of cluster selects one node (cluster head) to act as a proxy for the entire cluster. This proxy knows few other proxies in the network and forms a distributed network like Gnutella. While a centralized model, like Napster is formal between the cluster nodes and its proxy. Through proxies from a Gnutella type network, the number of nodes in the network, and hence the responds on behalf of the cluster the query response time is in the clusters have to frequently poll the cluster head and elect a new cluster head if they find that cluster head left the network, otherwise the cluster will be isolated from the entire network.

3. Problem Domain

To fulfill the objective of this thesis, the test bed is created originating on the problem domain that simulate a market space for various agents trading. This problem domain is named to be “ALTAR”. This works from design overall system and components of the problem domain then program the component in Market place and agents that interact within the space. The programming language was using the visual basic dot net which stands on the top of the dot net framework (this framework could use various languages and run upon the same assembly through the virtual machine – same paradigm as JAVA). For the data support system, this thesis has used DBMS name Microsoft SQL Server 2000. This is used for storing the core data of case base and also the data need to automate the market place to run smoothly.

The creation process had to be built on the argument as shown in Figure 3.1.

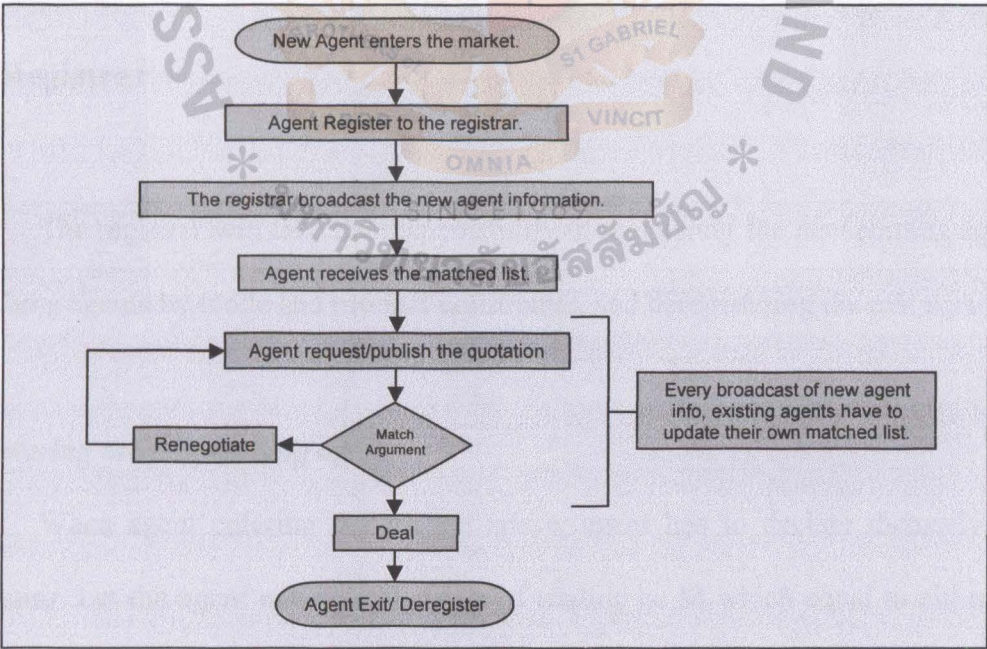


Figure 3-1 the procedure of ALTAR

The procedures of the market space start from new agent entering the ALTAR. The registrar will take care of the registration process of agents then broadcast the detail of that new agent (agent ID, Name, Trading Mode, Product – More detail in Section 3.2). Registrar will generate a list of agents (matched list) followed by the objective of the new agent, which matches the agents by product constrain. This list will be provided to agent as the started list of possible agent that it can interact to sell/buy the intended product. After this, the matched list had to be updated by the agent when registrar announces the new information. Then agent will start the trading activities (which be shown in detail in Section 3.3), agent then publish the quotation if the quote matches the argument then the deal will be accepted but if it does not match, the renegotiation will be revised until done.

From the Figure 1-2 which showed the structure of CBR Agent, the details of domain model and agent model (protocol) are discussed in Section 3.1, 3.2 and 3.4.

3.1 Registrar

The registrar will take the responsibility in registering the new coming agents, matching agents by mode and product constraints, and deregistering the exit agent.

Registering the new coming agent

When agent entering the market space, agent has to declare themselves to Registrar. Let the agent name be N , mode of trading be M which equal to either buy or sell and trading product be P . The set of information that agents have to declare (I_A) when registering is:

$$I_A = (N, M, P)$$

After agent inform their information then the unique ID generate for the new agent, which intended for agents to communicate without confusion. After that, the registrar will store the agent information to the Agents list. The Agent list (L_A) is the tuple of:

$$L_A = (ID, N, M, P)$$

Furthermore, the registrar will broadcast L_A to the market. Then matching agent step will activate.

Markup: Broadcast new agent information protocol

<Broadcast>

</Infor ID="..." Name ="Agent_Name" Mode="Buyer/Seller">

<Product>Item1</Product>

<Product> Item2</Product>

:

</Broadcast >

Matching agents

The matching process will link the buyer and the seller with the same product argument. The registrar will activate this event when new agent is entering the market space and when agent requested.

Deregistering the exit agent

When an agent informs to exit the market space, registrar uses the informed agent ID to search and delete the data of that agent in the agents list. Then the number of agent in the list is counted, if the number of agent is equal zero then the latest ID is reset.

Markup: broadcast agent exit protocol.

</Broadcast >

</Exit ID="...">

</Broadcast >



3.2 Negotiation Model

The negotiation model describes the scope of the negotiate activity. The model is showed in Figure 3-2. This diagram refers from the work of Carles Sierra [9] in which the time constraints is excluded.

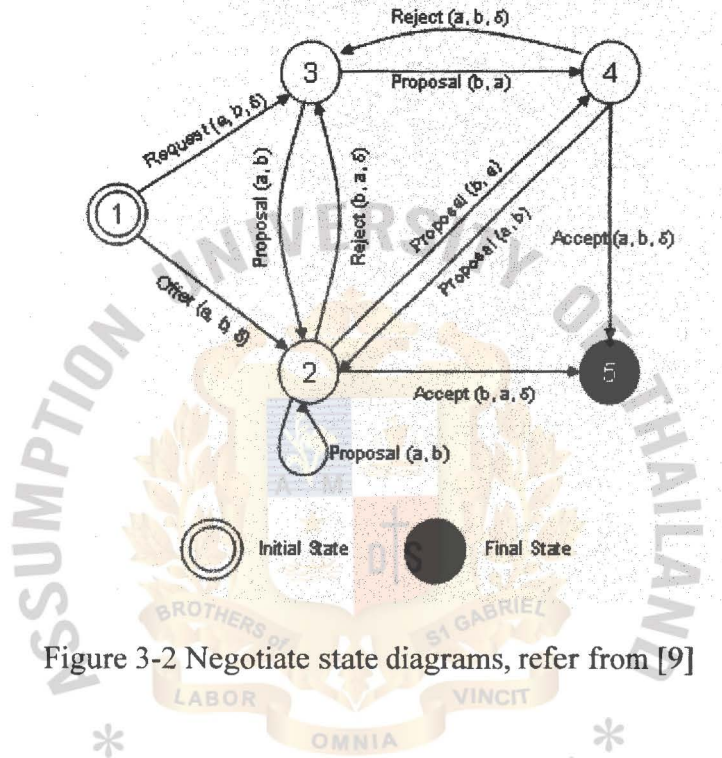


Figure 3-2 Negotiate state diagrams, refer from [9]

The definitions in the diagrams are:

1. Given $a, b \in \text{agents}$.
2. $\text{Deal}(X)$ is the accept argument function of agent X .
3. δ is the dealing argument.
 - a. If $\delta \in \text{Deal}(X)$ then the deal is accepted by agent X .
 - b. If $\delta \notin \text{Deal}(X)$ then the deal is rejected by agent X .

For example, if Deal argument of X agent is "proposed price is less than or equal 200" then if δ is "propose 300" is not in the set of deal argument

then rejected but if δ is “propose 250” is in the set of deal argument then accepted

4. Offer (X, Y, δ) is the offering of proposal from agent X to agent Y.
5. Request (X, Y, δ) is the requesting of proposal from agent X to agent Y.
6. Proposal (X, Y) is the sending of proposal from agent X to agent Y.
7. Accept (X, Y, δ) is the accepted deal from agent X to agent Y which agree in the argument in no.3.
8. Reject (X, Y, δ) is the reject of proposal sent from agent X as it disagree the argument model in no.3.

The negotiate state in Figure 3-2 where the negotiation activities can be described in to three states as:

1. A negotiation always starts with a deal proposal (offer or request).
2. After agent obtains the proposal, if the analyzed solution toward proposal is rejected then agent will stay in renegotiate state.
3. If proposal is accepted, it will direct the agent to final state.

Markup:

Request protocol: Buyer request for the proposal.

```
<Message Target="seller Agent" Source="Buyer Agent">  
  </Request Product item="Product Name" >  
</Message>
```


Offer protocol: Seller responds to proposal.

<Message Target="Buyer Agent" Source="Seller Agent">

</Quotation Product item="Product Name" Price="Selling Price">

</Message>

Negotiate protocol: buyer negotiates the price.

<Message Target="seller Agent" Source="Buyer Agent">

</Negotiate Product item="Product Name" Price="Expected Price">

</Message>

Accept protocol: if buyer accepts the proposal quoted from seller.

<Message Target="seller Agent" Source="Buyer Agent">

</Purchase Product item="Product Name" Price="Deal Price">

</Message>

3.3 Conventional CBR agent in ALTAR

The domain model consults the CBR agent as showed in Figure 4-1. The conventional CBR agent structure had been modified in order to execute in this negotiation domain – ALTAR.

Retrieve step: using the SQL command retrieve cases from case base by using product constraint.

Reuse Step: same as conventional reuse step of CBR that will find the match case by product constraint that quote from another agent as proposal.

Revise Step: the adaptation of the retrieved cases is case base on the price average in adaptation technique.

$$adapted_price = \frac{\sum_{i=1}^n P_i}{n}$$

Where

- P_i is the product price of retrieved cases from case base.
- n is number of retrieved cases.

Review Step: evaluate the adapted price that the argument must follow the objective of the agent based on problem domain.

Retain Step: feed back the solution to case base to increase the knowledge experience of CBR.

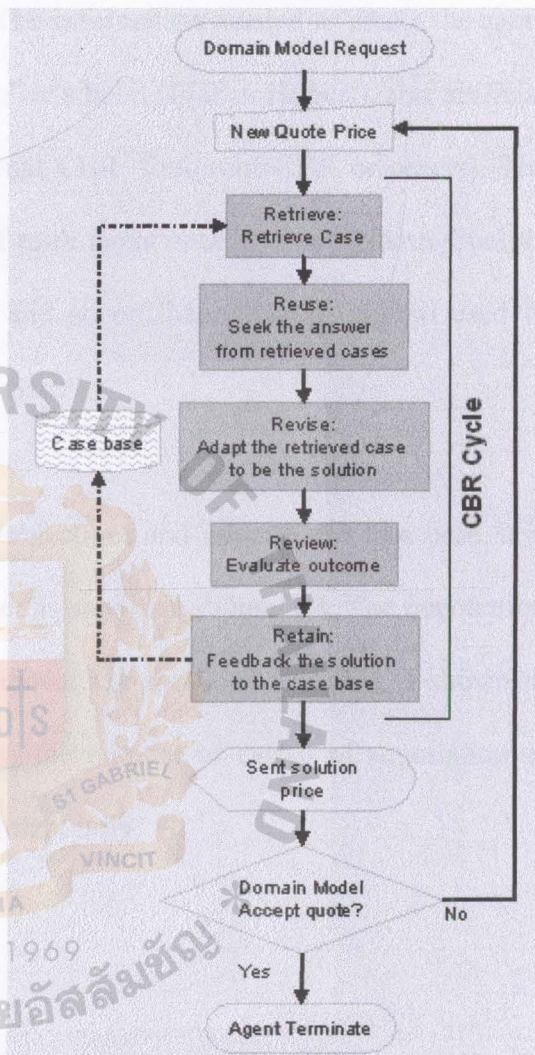


Figure 3-3 CBR components & domain model

3.4 GUI of the ALTAR

An interface of Altar application is shown in Figure 3-4. The top-left panel is used for creating and managing the agent. The information needed to create the agent are name, type of agent (buyer or seller), seller's habit (Liar or Honest), and artificial intelligence techniques applied (Conventional CBR, foolproof CBR or others). The bottom-left panel provides the objective of each agent such as "Sell Nikon Coolpix 2000 for \$350", "Buy Mobile for \$250" and so on. Moreover, it is also used to manage the case base in case of CBR agent type.

After the agent creates and sets the objectives and initiates the case base, it is sent as the buying agent to the virtual market as shown in Figure 3-4. The negotiation episode is executed under the specific argument via specified protocol as shown in Figure 3-5. After the success of bargaining, the number of cycles of negotiation or adaptation will be counted and collected consequently.

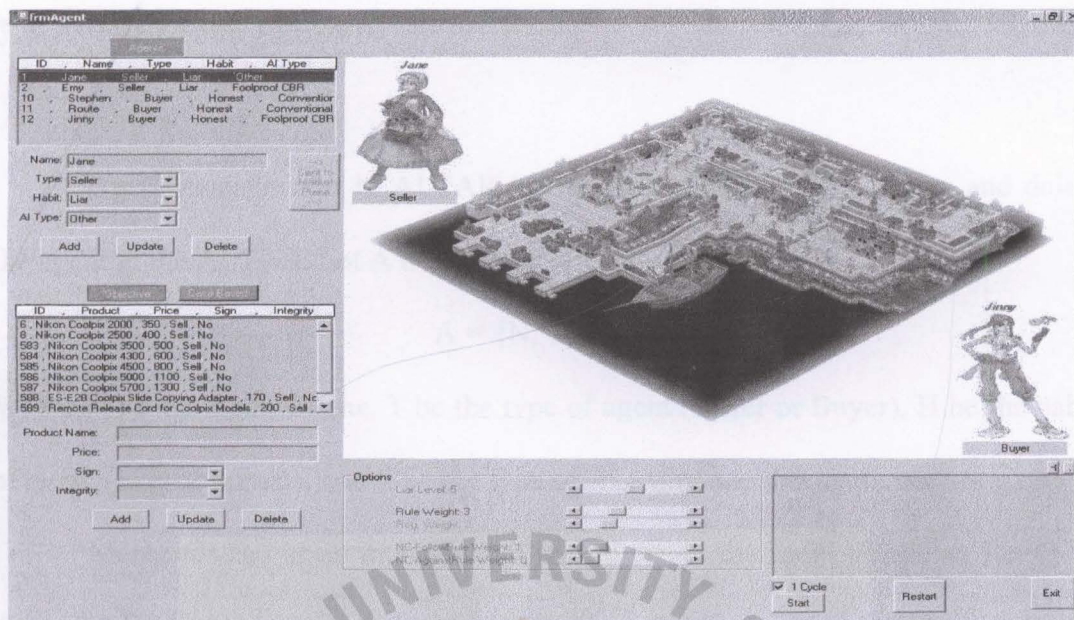


Figure 3-4 Experiment simulation users interface

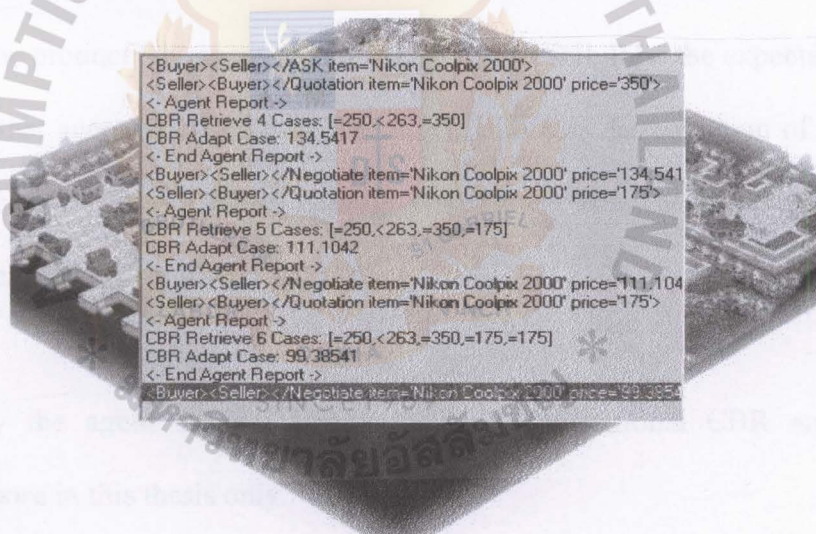


Figure 3-5 the negotiation between agents

3.5 Agent Management

Agent management in ALTAR has the function to create, update and delete the agent in the ALTAR. Let A be the tuple of agent.

$$A = \{N, T, H, R\}$$

Where N be the Agent Name, T be the type of agent (Seller or Buyer), H be the habit of agent (Liar or honest), and R be the type of AI technique that agent use.

Moreover, the agent management also manages the agent objective. Let O be the tuple of agent object.

$$O = \{PR, P, S\}$$

Where PR be the product that agent would like to buy or sell, P be the expected price of the product that agent would like to buy or sell, and S is the condition of trading price which include trading equal to specific price, lower than the set upper bound price, and higher than the set lower bound price.

Note: currently the agent support agents that use conventional CBR and new techniques propose in this thesis only.

4. Proposed Technique

Before proposing the new technique, the more important is the behavior of the conventional CBR agent in the negotiate problem domain. Therefore this study will examine the conventional CBR agent under test bed (ALTAR). The conventional CBR agent will assume to be buyer that match with varies selling agent. This experiment is tracking the times of negotiation that buyer use to negotiate until deal curtain selling price of product. This result is the base line to evaluate the successor experiments.

4.1 The Conventional CBR Agent in ALTAR

The study of conventional CBR agent VS lying agent in ALTAR aims to check the hypothesis that when the conventional CBR agent perform under lying environment, the behavior of CBR agent will be fooled to buy the overprice product without awareness. Fool is an event that another agent propose a price which is considered, from our agent point of view, to be too high or too low.

Figure 4-1 depicts the result of negotiation between conventional CBR agent and lying agent. Y-axis represents number of negotiation. X-axis shows proposed product price. The expected product price in this scenario is 250 \$. The graph shows that when the selling agent proposed the product price which is less than expected product price (250 \$), the negotiation time is 0. This implies that buying agent perform buying transaction at its acceptable price. Contrarily, the buying agent and

the selling agent perform a lot of negotiation when the proposed product price is higher than expected product price. When dealing with lying agent, the experimental result shows the weakness of CBR agent that with the increasing number of negotiation time, the CBR agent will finally agree with proposed product price.

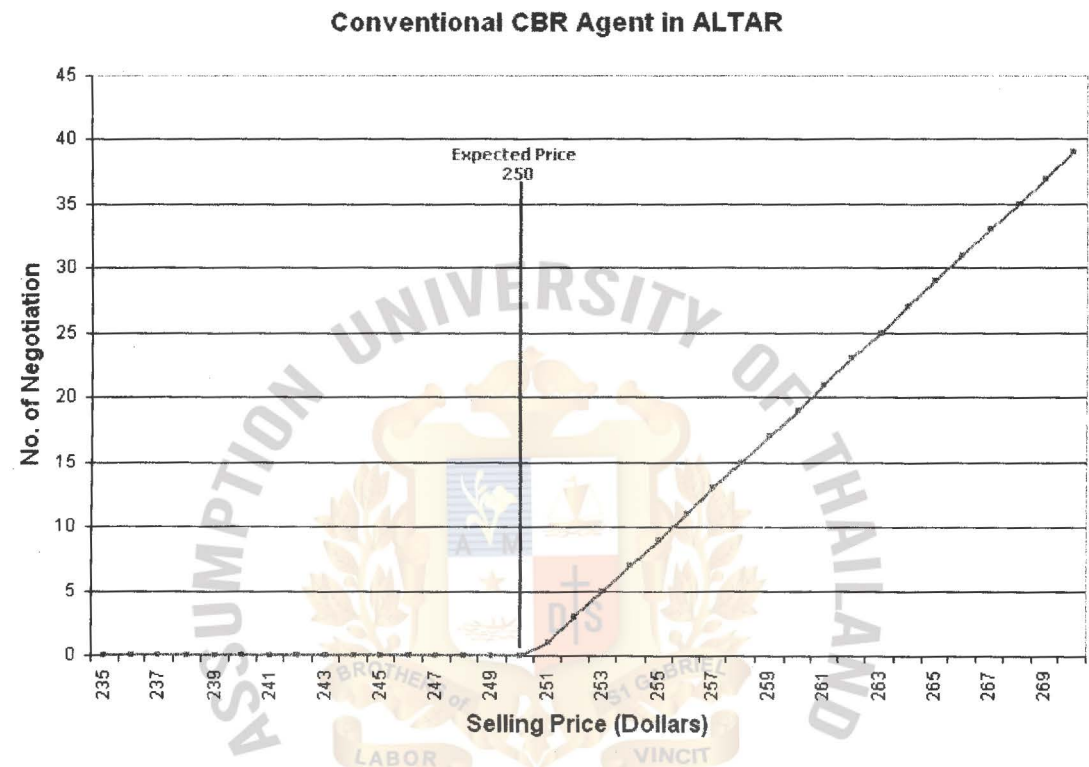


Figure 4-1 Negotiation between Conventional CBR agent in ALTAR

This chart will be the base line for improving the conventional CBR agent to have the self protection mechanism against the deceitful quote.

Then after having the baseline, the technique called “Foolproof” has proposed for improving the conventional CBR agent to have self protection from lying paradox from lying agent (named as foolproof CBR agent).

In the next Subsection, as for improving the case analysis and case base management, the thesis shows the refining existing architecture of foolproof CBR agent

1. Case base preparation.
2. Refinement in Retain Process.
3. Maintenance Temporary cases to a Regulation case.
4. Adaptation technique in revise step of CBR.
5. Extends the Domain Model/ Negotiation Model.
6. Meta data modification.

4.2 Case Base Preparation

The foolproof CBR agent is based on the idea of the justice that every accused person must be judged by the law. The Foolproof technique has separated some cases in case base representing the law (law cases) that are used to judge the coming experience whether it conforms or goes against the law. As depicted in Figure 4-2, the case base is categorized into two groups Meta data/LAW and Experience.

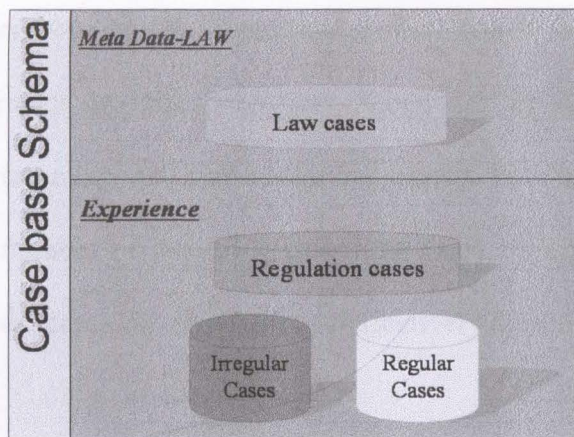


Figure 4-2: The Case Base model

Meta Data group

Law cases define the law, which is law case as prior knowledge for the agent to judge the incoming cases. The merit of this knowledge is to delay the adaptation to the new environment. The law cases are defined by using the following signs; equal ($=$), greater than ($>$) and less than ($<$). In the price negotiation domain, equal sign means precise buying price while greater than and less than signs indicates the upper and lower bound of the price that is acceptable to buy. The Meta data group is predefined by the master of agent. This law cases must not be inconsistency with each other.

Experience group

Temporary cases are incoming cases kept during the negotiation phase with an identified agent. They are classified into two groups; regular case and irregular case. The regular case is new experience that follows the law cases. Oppositely, the irregular case goes against the law cases.

Regulation cases originate by the intention to protect one of the weak point of CBR that base the efficiency on case base size. When one episode of negotiation is finished, the temporary cases will summarize and transform to be one regulation case. The overwhelming number of incoming cases that must be kept in the case base is decreased. Afterwards, cases searching time is effectively decreased.



4.3 Refinement in Retain Process.

After the incoming case is passed through the review process in CBR cycle, it is inserted into the temporary knowledge of the case base according to the following algorithm:

```
IF Price of incoming case > upper bound price of law case then
    Incoming case's Integrity Attribute = Ni //Irregular case//
ELSEIF Price of incoming case < lower bound price of law case then
    Incoming case's Integrity Attribute = Ni //Irregular case//
ELSE
    Incoming case's Integrity Attribute = Nr //Regular case //
END IF
```

For example, if the Law case is {Nikon CoolPix 885 price below \$100} then the incoming case {Nikon CoolPix 885 price equals \$100} is regular cases. On the other hand, the incoming case {Nikon CoolPix 885 price equals \$120} belongs to irregular case.

4.4 Maintenance Temporary cases to a Regulation case

Because superfluous of case base decreases the conventional CBR agent's performance, Case Base Maintenance (CBM) technique is applied to the foolproof CBR agent in order to efficiently manage the case base. The steps of cases' transformation are as follow.

1. CBM module is triggered after the foolproof CBR agent finishes one negotiation episode (one agent move in, trade, and move away).
2. CBM summarizes by averaging the price of the regular cases only that traded agent.

$$\text{regulation price} = \frac{\sum_{i=1}^n P_i}{n}$$

where P_i equal to price of regular cases, n is number of regular cases

3. CBM insert new regulation case combined with product and calculated price.
4. CBM restores the regulation case into learning knowledge level.
5. CBM deletes all the regular cases from the case base.

4.5 Adaptation Technique in Revise step in CBR

In the Revising process, the adaptation of the retrieval cases is based on the weighted average. The adaptation techniques proposed in foolproof CBR is adding a variable named "Degree of trust" to multiply with the average of irregular cases.

$$\text{Adapted price} = \frac{W_{Law} \left(\frac{\sum_{j=0}^{n_1} P_{Law_j}}{n_1} \right) + W_{Reg} \left(\frac{\sum_{j=0}^{n_2} P_{Reg_j}}{n_2} \right) + W_r \left(\frac{\sum_{k=0}^{n_3} P_{regular_k}}{n_3} \right) + W_{ir} \left(\frac{\sum_{m=0}^{n_4} P_{irregular_m}}{n_4} \right)}{W_{Law} + W_{Reg} + W_r + W_{ir}}$$

Where

- W_{ir} is the degree of trust. This variable using for tuning the usage of the average price that retrieved from irregular case. It has value recommend being 0 to 50% of $W_{Law} + W_{Reg} + W_r + W_{ir}$. The zero value turns the foolproof agent to be the agent that neglects the irregular case. The value set to around 50% response the foolproof CBR agent to be same as conventional CBR Agent.
- P_{Law} , P_{Reg} , P_r , and P_{ir} are the prices from Law cases, Regulation cases, Regular case, and Irregular case respectively.
- n_1 , n_2 , n_3 and n_4 is number of cases from various sources

4.6 Extending the Domain Model/ Negotiate Model.

To extend the force of lying agent self protection, the domain model of conventional CBR had been extend to support the preempt agent and ban agent.

Preempt Agent

To ensure that the foolproof CBR agent gets enough procedure information from the environment, it must not strict the negotiation to any specific agent. The foolproof CBR agent then keeps dealing with other agents, one agent per a negotiation episode at a time, to get the acceptable answer (cheap selling price) from one of them. The extension of domain to fulfill the object showed in Figure 4-3. Every rejected proposal will be evaluated by function ϕ and if $\text{limit_episode}(\phi)$ is true then the seller agent will reject.

$$\text{Limit_episode}(\phi) = \{\text{number of reject in each episode} \geq \text{Episode_Limit_Value}\}$$

and

Recommended: Episode_Limit_Value equal the times that Conventional Agent use to deal with Lying Agent, which proposed product price equal expected upper bound price.

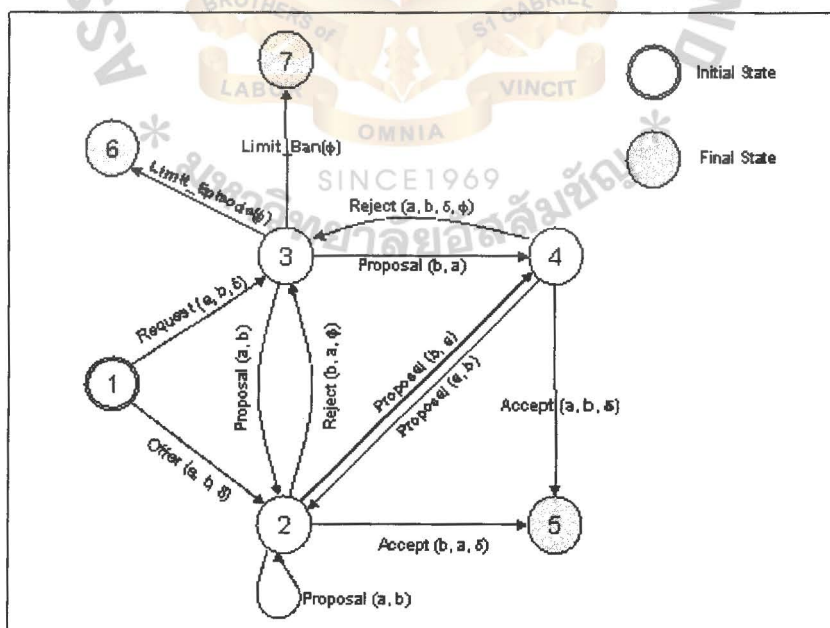


Figure 4-3 extends the domain/negotiate model.

Ban Agent

At the end of each negotiation episode, the foolproof CBR agent must conclude the negotiation whether it is successful or not. If the outcomes turns fail and the function $\text{Limit_Ban}(\phi)$ is true then it marks that dealt agent as banned agent. So that, the foolproof CBR agent needs not to deal with this agent again in the next time.

$\text{limit_ban}(\phi) = \{\text{number of irregular cases in all episode} \geq \text{Banned_Limit_Value}\}$

and

Recommended: Banned_Limit_Value should more than limit_episode but less than twice of times that conventional agent use to deal with lying agent, which proposed product price equal expected upper bound price.

4.7 Meta data modification

In case that the environment is changed, all agents are shifting the price up (like war time). The foolproof CBR agent will later conclude that those dealt agents are liars. When number of liar detection exceeds the checking point (too many lying agents are observed), it learns that the environment is now changed and its knowledge must be updated. Hence, the foolproof CBR agent must revise its law cases in Meta data level by using the following procedures:

1. If the number of banned agent exceeds the checking point. [The check point is recommended to be at least 80 percent of the average agents in domain.]
2. Summarize the prices from all the banned agents' cases.
3. Replace the expected product price with the average price computed from summarized price and expected product price defined in existing law case.
4. Evaluate the upper or lower bound of summarized price.
 - a. If the summarized price is more than upper bound price in the law case then the upper bound is replaced by the average of the upper bound price and summarized price.
 - b. If the summarized price is less than lower bound price in the law case then the lower bound will be replaced with averaging of the lower bound price and summarized price.
5. Delete the relevant cases in regulation and temporary cases.
6. Banned agents are released.

5. Evaluation

In order to evaluate the performance of foolproof CBR, the result of negotiation between foolproof CBR agent and lying agent is shown in Figure 5-1. It shows Y-axis as the number of negotiation times. X-axis shows proposed product price. Foolproof mechanism enables foolproof CBR agent to deal with lying agents. Foolproof CBR agent will firstly define expected product price, acceptable lower bound and acceptable upper bound for a particular product. This experiment preset W_{Law} , W_{Reg} , and W_r be 1 as for study base on conventional CBR agent. The test change the value of degree of trust (W_{ir} , represent as F) varies from 0.1, 0.3 0.7, 0.9 and collect the number of negotiation time.

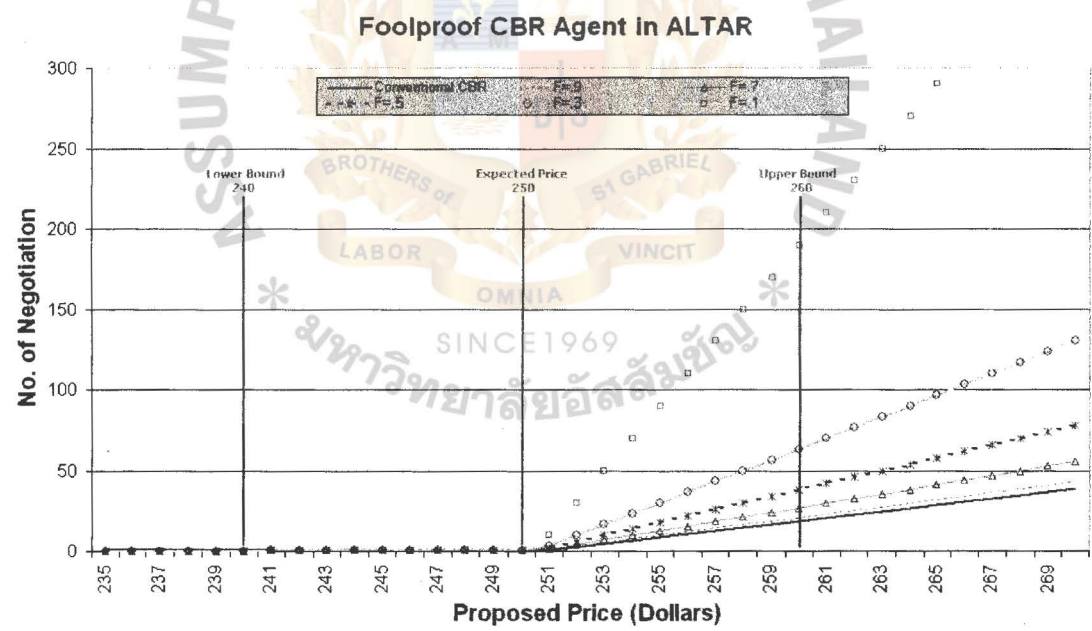


Figure 5-1 – Negotiation between foolproof CBR agent and Lying agent

Comparing with foolproof CBR agent with 0.1 or very low degree of trust, foolproof CBR agent with 0.9 degree, or very high degree, spends extremely more

time in negotiating with the lying agent when the proposed product price is higher its expected product price and the upper bound.

Compared to conventional CBR agent, foolproof CBR agent will not easily agree with the unwanted proposed product price.

In case that many agents perceived to be liar, the situation may be changing of environment such as in war time price of product is higher. So, the foolproof CBR agent should revise its Law cases in meta data level by using the mentioned procedures when the noise level or number of liar in the case base reach the threshold level at 80 % of negotiating agents.



6. Conclusion & Future Research

The paper discusses about self-protected mechanism for Case-Based Reasoning called “Foolproof Case-Based Reasoning”, which enables a CBR agent to deal with lying agents. A foolproof CBR agent enhances a conventional CBR agent by deploying Law cases adaptation in order to update the metadata or Law cases of an agent. The experimental result has shown that Foolproof CBR Agent has better performance in dealing with lying agents, comparing to conventional CBR agent.

Consequently, the following issues must be taken into consideration for future research.

- Complicated multi-agent domain

The foolproof agent should be able to deal with various types of agents.

- Intelligent lying pattern

The lying agent can learn from its listeners' expressions and the lying agent is able to select the proper plan for its attack.

- Apply Foolproof CBR Agent to Stock Exchange Domain

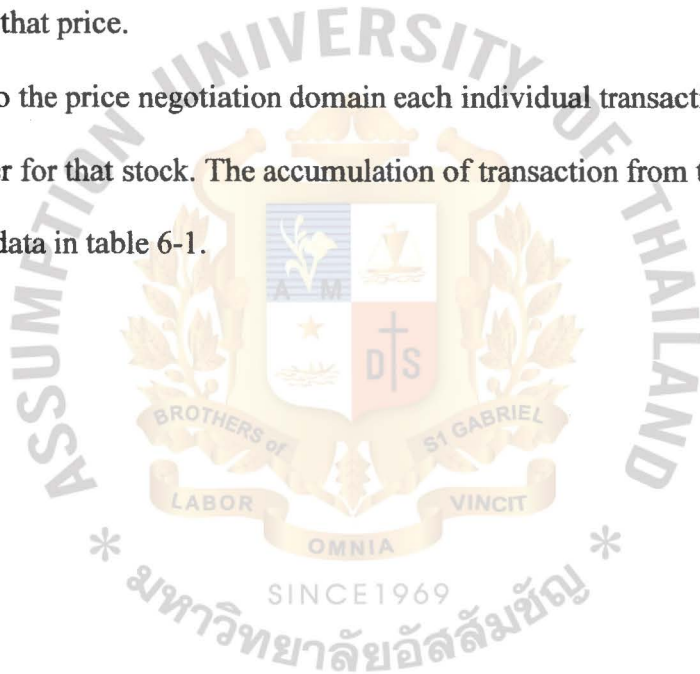
The stock exchange domain is basically the matching between buyer and seller by pricing constraint. The data of domain simple show as table 6-1.

Bid		Offer	
฿ 41.00	120,200	฿ 42.00	1,238,200
฿ 40.00	89,700	฿ 43.00	242,600
฿ 39.00	203,900	฿ 44.00	234,900

Table 6-1 the stock exchange based negotiate sample

The data separate into two main columns-bid and columns-offer. Within the s- column represent the bid/offer price and the amount of stock waiting for bidding and offering for that price.

Compared to the price negotiation domain each individual transaction are an agent that bid/offer for that stock. The accumulation of transaction from trivial agent is the summarize data in table 6-1.



BIBLIOGRAPHY

- [1] Leake, D., ed., 1996, Case-Based Reasoning: Experiences, Lessons, and Future Directions. Menlo Park: AAAI Press/ MIT Press, 1996.
- [2] Russell, Stuart J. and Peter Norvig (1995), Artificial Intelligence: A Modern Approach, Englewood Cliffs, NJ: Prentice Hall
- [3] Maes, Pattie (1995), "Artificial Life Meets Entertainment: Life like Autonomous Agents," Communications of the ACM, 38, 11, 108-114
- [4] Micheal Wooldridge & Nicholas R. Jennings, Intelligent Agents: Theory and Practice, Knowledge Engineering Review 1995
- [5] Francisco J. Martín & Enric Plaza, Auction-based Retrieval, ???
- [6] Enric Plaza, et al., Cooperative Case-Based Reasoning, G Weiss (Ed.), Distributed Artificial Intelligence meets Machine Learning 1997
- [7] Francisco J. Martín, Enric Plaza, and Josep Lluís Arcos, Knowledge and Experience Reuse through Communication among Competent (Peer) Agents, International Journal of Software Engineering and Knowledge Engineering, Vol. 9, No. 3, P.319-341
- [8] Shaheen S. Fatima, Michael Wooldridge, Nicholas R. Jennings. Multi-Issue Negotiation Under Time Constraints, 2002.
- [9] Carles Sierra, Nick R. Jennings, Pablo Noriega, Simon Parsons. A framework for argumentation-based negotiation, 1997.
- [10] Naveen Srivasan and Timothy Finin, Enabling PeertoPeer SDP (Service Discovery Protocol) in an Agent Environment, 2001

