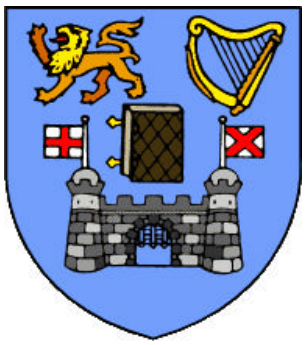
# Case-Based Travel Agent

A dissertation submitted to the
University of Dublin, Trinity College,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science

Frédéric Peuret
DUT, DEST, BSc



Department of Computer Science
Trinity College Dublin

September 1999

## Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

_____
Frédéric Peuret
September, 99

## Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

_____
Frédéric Peuret
September, 99

# Abstract

With the explosion of the Internet, we are getting more and more information and it becomes difficult to digest. Agents exist to help the users. An agent is a component of software (or hardware) that is capable of accomplishing tasks on behalf of its user. To accomplish these tasks, it should be able to adapt itself to the user and should also be able to learn from his decisions. Most of them use a rule-based system and it appears that they are not really adapted to the user. A case-based reasoning system seems to be a better approach.

Rule-based systems need a good definition of the system and are not able to learn very well from the user choices. What is more, it is not able to handle missing information or unexpected values. On the other hand, an agent build with a case-based reasoning system should be able to solve these problems.

In order to show the benefits of a case-based reasoning system for an agent, one of the FIPA (Foundation for Intelligent Physical Agents) scenarios will be used as a context for this dissertation: the Travel Assistance Scenario. This scenario involves different kinds of agents that interact in order to plan a journey for a user. In this dissertation, a case-based system will be implemented using a Case-Retrieval Net (CRN) for retrieval. It will then be used to demonstrate the flexibility and adaptability of this kind of system.

# Acknowledgements

First, thanks to my supervisor, Pádraig Cunningham.

Thanks also to Paul, Graham and David for helping to proof read this script.

Finally, thanks to Tom, my family and friends for their support.

# Contents

# Figures

# Chapter 1

# Introduction

Nowadays, we are living in a society where travelling becomes more and more important, as much for work as for leisure. There are many ways to book journeys but one of them is increasing exponentially: booking by electronic means. A wide variety of travel related services are becoming available through this medium. In fact, there is now so many of them that it has become more and more difficult to plan a journey that suits the particular interests of the users.

The users would like to relay the task of sorting out all the propositions and get some personalised offers. To provide this kind of help, an intelligent agent seems to be the best solution. This is what FIPA (Foundation for Intelligent Physical Agents) propose in its Travel Assistance Scenario.

This chapter introduces the work of the FIPA organisation in the area of intelligent agents. An overview of its solution for the travel problem will be presented. Then, a description of the scope of this dissertation will be made and, finally, a roadmap of the dissertation will be presented.

## *1.1 FIPA*

The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association registered in Geneva, Switzerland. FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment. This goal is pursued by making available in a timely manner, internationally agreed specifications that maximise inter-operability across agent-based applications, services and equipment. This is realised through the open international collaboration of member organisations, which are companies and universities active in the agent field. [4]

## *1.2 Travel assistance scenario*

One of the areas in which FIPA works is in the area of travel assistance. In 1997, FIPA published a document called "*FIPA 97 Draft Specification Part 4: Personal Travel Assistance*". This document provides a specification of basic agent technologies that can be integrated by agent systems developers to make complex systems with a high degree of inter-operability.

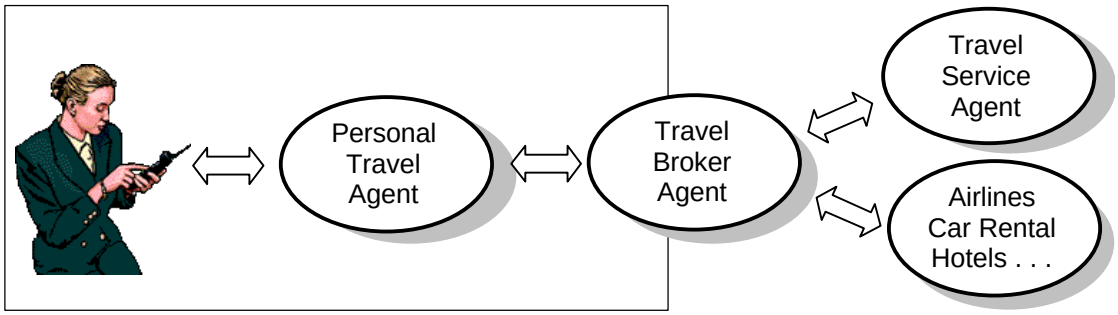
In this travel scenario, agents are used to help the users to plan their travel. It involves three types of agents. Each of them has a specific role:

- The Personal Travel Agents (PTA) is the agent that acts on behalf of the user. It is responsible for booking everything that is needed for the user's travel. It is personal to the user and adapted to his behaviour. It communicates with one or many TBA.
- The Travel Broker Agents (TBA) is like a travel agency. It receives the queries from the PTA and then, it is left to gather the different services provided by the TSA in order to make a complete travel plan.
- The Travel Service Agents (TSA) is an agent attached to a specific domain. It is responsible in delivering up-to-date information about its domain to the TBA.

Here is an overall scenario involving these agents:

A user living in Dublin wants to go to Paris the next day. He gives some information to his agents and delegates to it the task to plan the travel. The user can make a full request or a partial one. The agent is able to complete it using the past history of the user. Then, the agent contacts one or many TBA in order to get some propositions. The TBA(s) contacts the TSAs to build one or several complete plans. Then, it sends it (them) back to the PTA that is able to sort them out. If there is a proposition that suits the user preferences, it is presented to the user in order to get his approval.

The following diagram represents the interaction between the user and the different agents.




*Interaction between the user and the agents*

In the FIPA scenario, the user can get two kinds of help from his PTA. He can use it to plan travel, which is called pre-trip planning, and he can also use it during the trip to change the booking if needed or ask for new activities. This is the on-trip execution. This dissertation will only deal with the pre-trip planning.

The FIPA scenario will be used as a reference for this dissertation.

## *1.3 Dissertation goal*

The goal of this dissertation is to demonstrate the benefits of case base reasoning for an agent like the PTA agent in the FIPA scenario. In order to do that, a full design of the PTA will be completed. Then, the case-based reasoning of the PTA will be implemented and used to make some tests. These tests should demonstrate the flexibility and the ability to learn of an agent using a case-based reasoning.

The aim of this dissertation is not to provide a fully functional agent, but to provide an agent that should be able to perform two tasks: complete a user's request and sort out the offers made by the TBA agent.

## *1.4 Roadmap of the dissertation*

In this chapter, an overview of this document will be done.

**Chapter 2: Background research**

This chapter presents the subjects covered in this dissertation. It is an overview of agents, rule-base systems and case-based reasoning.

**Chapter 3: Case-based versus rule-based reasoning**

This chapter gives the pros and cons of ruled-based reasoning and case-based reasoning. The best solution for the PTA agent will be chosen.

**Chapter 4: Design**

A detailed design of the PTA agent will be done in this chapter. The design will cover a possible web implementation of this agent and the details of the PTA itself.

**Chapter 5: Implementation**

This chapter will describe the work done during the dissertation. It will explain what has been developed and how. It will also explain some decisions made during the implementation.

**Chapter 6: Conclusion**

An overview of what has been accomplished during this dissertation and the remaining work will be carried out in this chapter.

**References**

**Appendices**

# Chapter 2

# Background research

## *2.1 Agents*

The idea of software agents exists since the early days of Artificial Intelligence (AI). In the 70's, Carl Hewitt [5] proposed the concept of a self-contained object, which was interactive and executed concurrently. This object was able to respond to messages sent by similar objects. It was called an 'actor'. But it is only recently that the agent paradigm has become popular. This popularity is mostly due to the characteristics of agents like modularity, speed, reliability, flexibility and the possibility to use them in a wide range of problems.

### *2.1.1* What is an intelligent agent?

Since the beginning of AI, the "intelligence" has been defined by many ways and with different approaches: philosophy, biology and study of social systems. The last approach is probably the one that applies the most to the idea of agents.

The social system provides a metaphor for intelligence in that it exhibits global behaviour that enables it to solve problems that would confound any of its individual members. For example, a member of the New-York society is not able to predict accurately the number of loaves of bread to be consumed in the city on a given day. However, the entire system of New-York bakeries is able to keep the city stocked with bread with a minimal waste. We could do the same analogy with the stock market where an individual has a limited knowledge of a few companies but the stock market is able to set the relative values of hundreds of companies [7].

These examples show that intelligence is reflected by the collective behaviour of large number of very simple interacting, semi-autonomous individuals, or agents. From these examples, we can deduce some rules about agents:

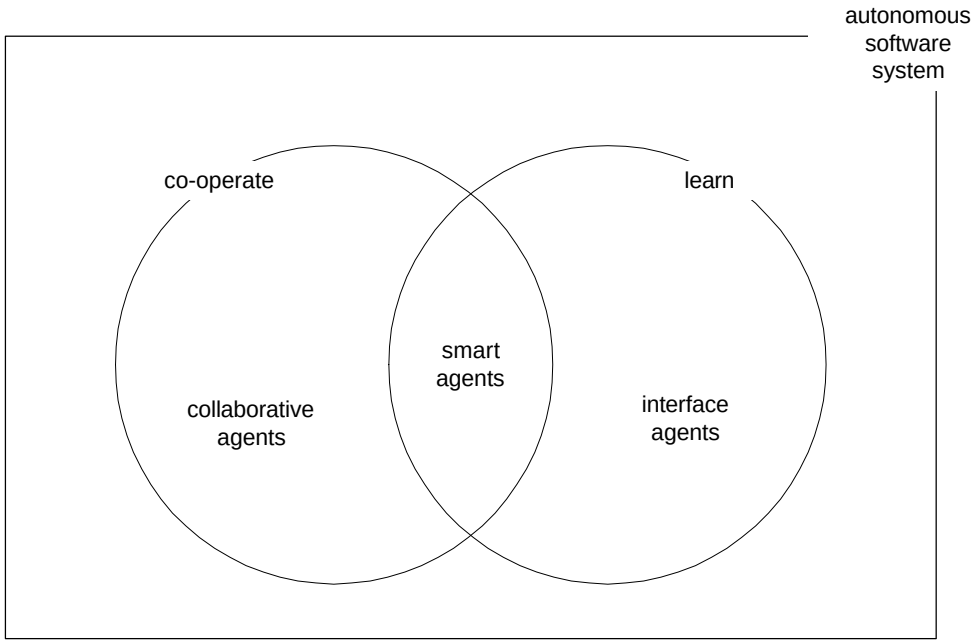
- Agents are autonomous. Each agent has some responsibilities and has no or little knowledge of what the other agents do.
- Agents are "situated". An agent is sensitive to its surrounding environment but has most of the time no knowledge of the full domain of all agents.
- Agents are interactional. Usually, an agent can't solve a problem by itself. It has to communicate and co-operate with other agents. It is the combination of the work of each agent involved in the process that leads to a solution.
- In a society, agents can be homogeneous or heterogeneous. Agents can be all of the same type (like ants in a colony) or they can have their own speciality (like crewmembers in a plane).
- The phenomenon of "intelligence" is emergent. Each agent has some skills and responsibilities, but the overall co-operative result of the society of agents is greater than the sum of its individual contributors. Intelligence seems to be not a property of an individual agent.

Based on these observations, we can give a generic definition of an agent: it is an element (either computer, or human) of a society that can perceive aspects of its environment and can affect this environment either directly or through co-operation with other agents. It is capable of carrying out goals. Most precisely, in the computing context, it is defined as a component of software (or hardware) that is capable of accomplishing tasks on behalf of its user.

### *2.1.2* Classification of Agents (Typology)

Software agents can be classified in many ways but there are five major criteria to classify them:

- Agents can be classified by their mobility. It is their ability to move in a network.
- Agents can be classified as deliberative or reactive. Deliberative agents have their own reasoning model. They engage contact with other agents to achieve their goal. Reactive agents just react to some stimuli originated by the environment in which they are embedded. It has no representation of this environment.
- Agents can be classified by the attributes they should exhibit (autonomy, learning, co-operation). Autonomy is the agent's ability to act without any external help. Learning is its ability to memorise new facts and reuse them. Finally, co-operation is the way agents help each other to achieve their goal. These three characteristics give us three types of agents: collaborative agents, interface agents and smart agents. The figure x shows these three types.




*Part view of an agent typology*

- Agents can be classified by their role.
- Agents can be classified as hybrid or not. The hybrid agents gather two or more agent's philosophy at once.

### *2.1.3* Overview of the different kind of agents

#### *2.1.3.1 Collaborative Agents*

The main characteristic of collaborative agents is their ability to co-operate with other agents. They communicate and sometimes negotiate in order to reach their goal. Collaborative agents are static most of the time. They are useful when there is a resource limitation or when a centralised system is not wanted.

#### *2.1.3.2 Interface Agents*

Interface agents are mainly focused on learning in order to perform some tasks on behalf of its user. Most of the time, they provide proactive assistance to the user. They monitor the actions he takes, learn how he uses the interface and then propose some better ways of accomplishing the task. They can also communicate with other agents, but only for advice when they do not know how to react. Communication is not their main purpose. They can learn from the user's feedback and by explicit instruction from him. It is this kind of agent that is used in Microsoft Office for example.

#### 2.1.3.3 Mobile Agents

Mobile agents have the ability to move in a wide area network like the World Wide Web for example. They go to the foreign host and then act on behalf of the user. Then, when they have accomplished their task, they come back to the user with the results of the task. This reduces considerably the network traffic. They have to communicate with the agents present on the foreign host in order to know which operation they can perform, but for all that, they are not considered as co-operative agents.

#### 2.1.3.4 Information/Internet Agents

The information agents' goal is to gather information for many sources and then, sort it out for the user. These agents are a response to the explosion of the Internet. Most of these agents are static and embedded within a web browser and use a host of Internet management tools such as spider and search engines in order to gather some information.

#### 2.1.3.5 Reactive Software Agents

The particularity of reactive agents is that they do not have a representation of their environment. They respond in a stimulus-response manner. These agents are basic and have *a priori* no intelligence. It is from the interaction of several of these agents that the intelligence emerges. Systems that use reactive agents tend to be more robust and fault tolerant due to the basic nature of these agents and the low level of communication between them. Flexibility and adaptability are also two benefits of reactive agents. But, on the other hand, the lack of an explicit goal is a severe limitation. It is also difficult to predict the behaviour of the entire system if the environment changes.

#### *2.1.3.6 Hybrid Agents*

A hybrid agent is an agent that combines the characteristics of several kinds of agents like the ones described before. As each type of agent described before has its weaknesses and strengths, the purpose of hybrid agents is to try to minimise the weaknesses and reinforce the strengths. The drawback of hybrid agents is that they tend to be specific to a domain and most of the time, specific to a task. Therefore, it has the same drawback than reactive agents: they cannot adapt themselves if the environment changes.

### *2.1.4* Conclusion

Considering the agents described before, we can determine the type of agents needed in the FIPA scenario. The PTA should be autonomous, able to learn and co-operative. It appears to be a hybrid agent between an interface agent and a co-operative agent.

- It should be autonomous in that sense that it makes some decisions without the help of the user.
- It should be able to learn from the user's decisions and should be able to make some propositions adapted to him. These are the main characteristics of an interface agent.
- Then it also needs to communicate with other agents (the TBA) in order to accomplish its task, which is to propose a travel plan to the user. This is the main characteristic of a co-operative agent.

After this brief overview of agents, we now know the nature of the PTA agent in the FIPA scenario. It is a hybrid agent. Now, we have to determine how to implement it. There are two possibilities. A rule-based reasoning or a case-based reasoning could be used. Before trying to determine which solution is the best for the PTA, an overview of rule-based reasoning and case-based reasoning will be carried out.

## *2.2 Reasoning within the PTA*

### *2.2.1* Rule-base reasoning

In a rule-base system, the knowledge base is a set of rules. These rules are represented as ‘if <condition> then <conclusion>’ statements. The ‘if’ portion represents the condition and the ‘then’ represents the action that will be taken if the condition is satisfied. This is one of the oldest representations of knowledge in expert systems. It is also very close to the way of thinking of humans. To build such a system, a good knowledge of the domain is needed. But, if the domain of expertise is well defined, it is relatively easy to translate the knowledge into rules. This chapter will present what is a rule-base system and how it works.

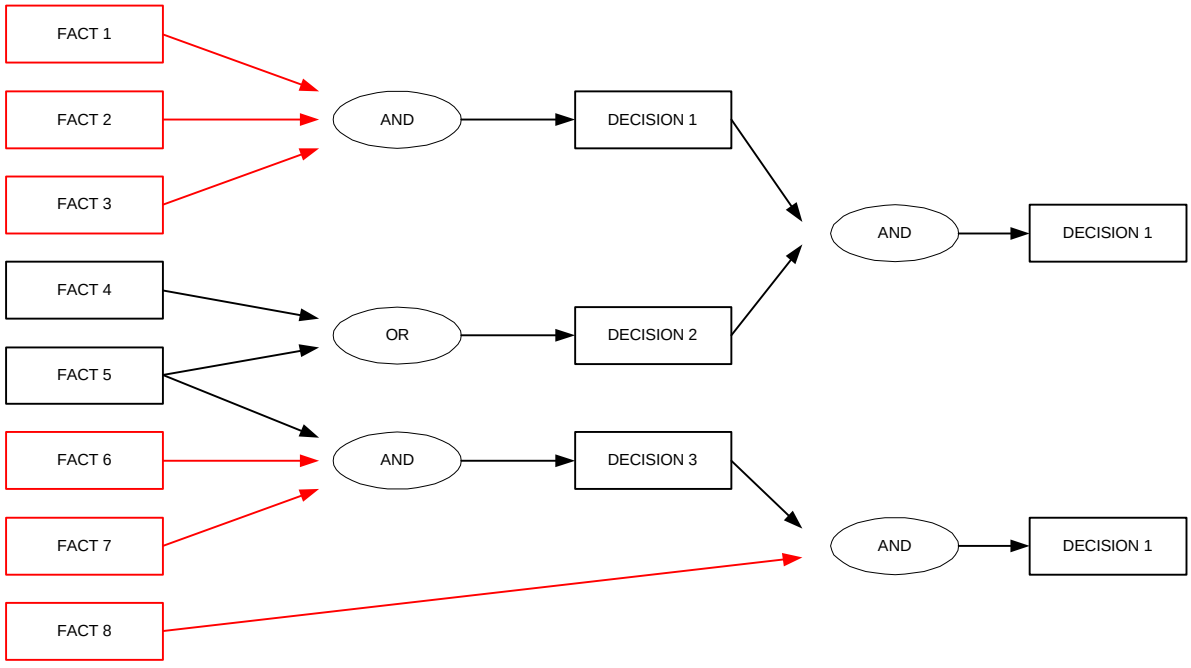
#### *2.2.1.1 How does it work?*

Using a rule-based system consists in searching the good rules to fire among all the rules of the system. There are two kinds of searches in a rule-based system. The search can be data driven or goal driven. In the first case, we have some facts and want to get a conclusion, in the other case, we already have a conclusion but we want to verify it.

#### 2.2.1.1.1 Data driven reasoning

This kind of search is also called ‘forward chaining’. It is used when there are a number of facts that are known, but the actual implications are not. For example, a computer may not work properly but we know that the screen is displaying something, the power light is on and the hard drive seems to be spinning. If all these facts are injected in a forward chaining system, it should lead to a series of suggested diagnostic tests that leads ultimately to a conclusion.

Forward chaining begins by matching known facts with the rules from the rule base. Once a rule has been found, the conclusion of that rule is added to the set of known facts. This new fact may be matched to another rule. The system will stop when a predefined goal is reached or when there are no other rules that match.

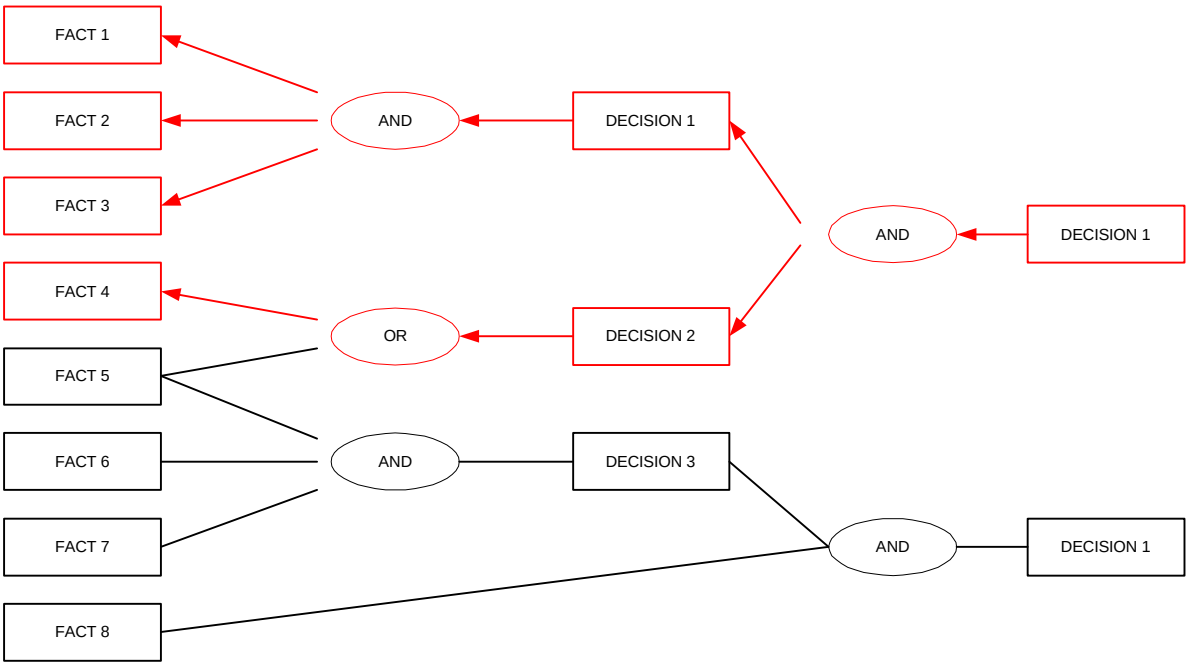



*Forward chaining*

### *2.2.1.1.2 Goal driven reasoning*

This type of search is also called 'backward chaining'. It is used when we have a result and we want to verify it. A doctor could use this kind of research if he has an idea about his patient's illness. He submits his idea to the system which provides him a list of symptoms that characterises the illness. Then the doctor can check if the patient has these symptoms.

The system attempts to verify the goal(s) by finding rules whose conclusions can satisfy the goals(s), and then it has the task of verifying each rule's premise. Any premise that cannot be immediately verified (i.e. the information required is not present in the system at the time of the inference), are treated as new goals (or sub-goals).




*Backward chaining*

### *2.2.2* Case-based reasoning

The interest in Case-Based Reasoning (CBR) has increased rapidly the last few years. It has a new approach in comparison with traditional AI approaches. It does not rely on the knowledge of a domain to solve problems. It uses past experience and adapts it to the new problem in order to solve it. It has also the ability to learn from each experience. In this chapter, an overview of what is case-based reasoning will be presented.

#### *2.2.2.1 Introduction*

Case-based reasoning uses a database of problems (with their solutions) to resolve new problems. The database can be built by human experts through the knowledge engineering process or it can be collected from the past cases encountered before by the program that uses the case-based reasoning. The example of a lawyer is relevant to explain what is a case-based reasoning. When a lawyer has a new case, he makes some research to find a past case that is close to the current one. He looks for a precedent. Then, he uses it in court to defend his client. If the lawyer can demonstrate that the case is the same as the old one, he will get the same judgement. If the case is not exactly the same, the lawyer will adapt the old one to the new one and, consequently, the past solution will also be adapted. A past problem and its solution are used to solve the current problem.

Case-based systems also have the ability to learn from their experiences. When a problem is solved, the case-based reasoner can add the problem description and the solution description in its database. It is then immediately available and considered as a new piece of knowledge.

#### *2.2.2.2 What is a case?*

A case represents a situation that happened in the past. It is in general represented as a pair “problem, solution”. The problem represents the description of the past situation and the solution is the description of the decision taken at this time. In a case, there is no way to know how the decision was taken. We just get the solution as it is. This could appear to be simplistic, but in fact, it is very useful. Because there is no generalisation, a case keeps all the particularity of a situation. If it was an unusual situation, and therefore an unusual solution was taken, we will be able to make this special decision again if we face the same problem.

#### *2.2.2.3 What is a case base?*

A case-base represents the knowledge of a system. It is a set of cases. The structure of a case base can be different depending of how it will be used. (For this agent, a Case Retrieval Net will be used. See chapter 4.3.3)

#### *2.2.2.4 What is the difference between a case-based reasoning and a database?*

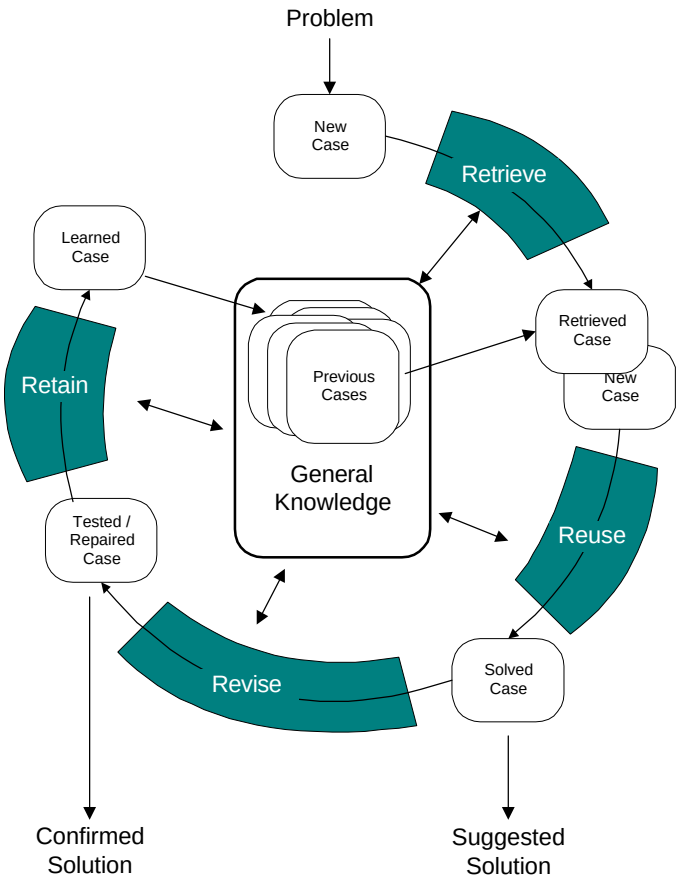
In a database, exact matches are required. We can use a past case only if it is exactly the same as the current one. The notion of retrieval in a database and in a case-based reasoning is nearly the same. The only difference is that a query to a database retrieves some data using an exact matching while a query to a case-based reasoning retrieves some data using an inexact matching.

### 2.2.2.5 Learning in a case-based reasoning

The learning process in a case-based reasoning is simple. It just consists of adding or deleting some cases in the case base. Each new case in the case base is immediately accessible and can be used to solve a new problem. That is why we can say that the case base has 'learnt'.

### 2.2.2.6 General overview of case-based reasoning

The main phases of the case-based reasoning activities are described in the CBR-cycle of Aamodt and Plaza (1994):




*CBR-cycle*

There are four phases in case-based reasoning:

- **Retrieving**: During this phase, the case-based reasoner searches the database to find a case close to the current situation.
- **Reusing:** This consists in using the retrieved case and, if needed, adapting it to the new situation. At the end of this phase, the case-based reasoner should be able to make a proposition.
- **Revising**: because of an inexact matching, the solution proposed at the end of the second phase may not be adequate. The revise phase can be thought of as a correction of the first proposition.
- **Retaining**: It is the 'learning' phase. It is here that the completed case is added to the database.

Both rule-base systems and case-based systems have been defined. Now, a comparison of these two possibilities will be done. From this, it will be determined which solution is the best for the PTA.

# Chapter 3

# Cased-based versus rule-based reasoning

This chapter will outline the advantages and disadvantages of rule-based and case-based reasoning. Then, the most relevant system for a PTA agent will be discussed.

## *3.1 Advantages of rule-based reasoning*

- The ability to use directly experimental knowledge acquired from human experts. It is very easy to transform some knowledge into a set of rules.
- The modularity of rules eases construction and maintenance. In rule-based reasoning, the link between the rules is weak. It is easy to change, add or delete a rule.
- Good performances in a limited domain. If the domain is limited and well defined, it is easy to cover all the possibilities and foresee all the possible situations. So, that is why a rule-based system can be very accurate and fast in a limited domain.
- Good explanation facilities. As it is explained in the chapter 2.2.1, it is possible in a rule-based system to obtain the information that has led to a conclusion.
- Rules chaining are easy to trace and debug.

## *3.2 Disadvantages of rule-based reasoning*

- Often, rules from human expert are heuristic and lack a deeper knowledge of the domain. If the knowledge of the domain is not extremely precise there is always a chance that an unexpected case occurs and break the rule-based system.

- Heuristic rules cannot handle missing information or unexpected data values. If a value is missing or a value is not expected, the rule-based system is not able to fire its rules. So, it can't give any result.
- Rules are not able to adapt when there is a new problem.
- The knowledge is very task dependent. Often, a set of rule is adapted to a particular problem and it is not possible to adapt it to another problem.

## *3.3 Advantages of case-based reasoning*

- Extensive analysis of domain knowledge is not required. In case-based reasoning, the most important thing is to know how to compare two cases. This task in not directly dependant of the domain.
- It allows shortcuts in reasoning. If an appropriate case is found, a solution can be found very fast (faster than it would be to generate a solution from scratch)
- It enables systems to avoid past errors and exploit past success. In case-based reasoning, the system keeps a record of each situation that occurred and uses it for each new problem. That is why it 'learns'.

## *3.4 Disadvantages of case-based reasoning*

- There are no explanatory facilities. In case-based reasoning, there is no way to explain why a solution is taken because it is just based on its similarity with other cases. A solution is not a logical decision as in a rule-based system.
- A large case can suffer problems of efficiency due to the storing and computing of the past cases. In case-based reasoning, a lot of cases need to be retrieved in order to compare them. That can be a major drawback if there is a large number of cases in the database.

- It is difficult to get good criteria for indexing and matching cases. The vocabulary for the retrieving and similarity matching algorithm must be carefully hand crafted. This can offset many of the advantages case-based reasoning offers for knowledge acquisition.

## *3.5 Conclusion*

*"Human experts are not systems of rules, they are libraries of experiences."*

Riesbeck and Schank (1989)

Ruled-based systems are often brittle because they can't learn from experience and they collapse dramatically when faced with a problem beyond their plateau of expertise [2]. On the other hand, case-based reasoning can deal with such situations by acquiring new cases. The PTA agent needs to adapt itself to the user's behaviour but, because we don't know his behaviour in advance, it is not possible to create a set of rules adapted for each user. A case-based reasoning system does not need to know as much about the domain as does a rule-based reasoning one.

Rule-base reasoning has some explanatory facilities that case-based reasoning does not have. Nevertheless, in the PTA context, it is not important because it does not ask the PTA to justify its choices. The main characteristic it should have is its adaptation to the user. And this can only be provided by a case-based reasoning.

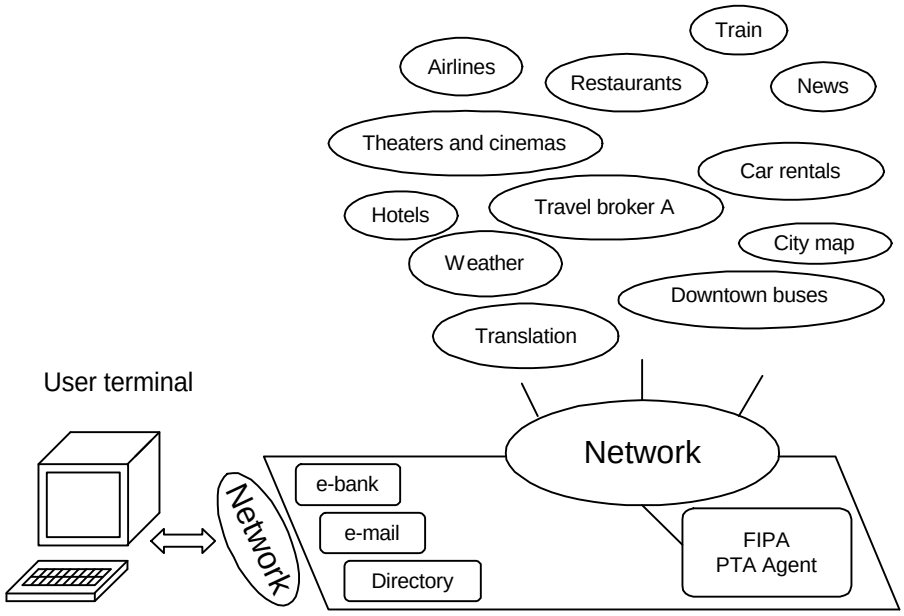
It seems that the advantages of rule-based reasoning are not very useful for an agent like a PTA. What is more, its drawbacks are too important to be overcome in this situation. Finally, the best solution for the PTA agent is a case-based reasoning.

# Chapter 4

# Design

## *4.1 The PTA and its environment*

In this scenario, FIPA gives a description of the PTA agent within its environment. It should be accessible from anywhere in a network. It must also be able to communicate with the other agents on this network. The communication language proposed by FIPA is the FIPA Agent Communication Language (ACL).




*PTA and its environment*

The next chapter proposes a possible web implementation of the PTA agent.

## *4.2 Design of a web implementation.*

The agent needs to be accessible from anywhere. Therefore, the obvious solution is to access it via a web page. This solution has the advantage that it is very easy to use and very accessible to the user. However, this solution is not trivial for implementation.

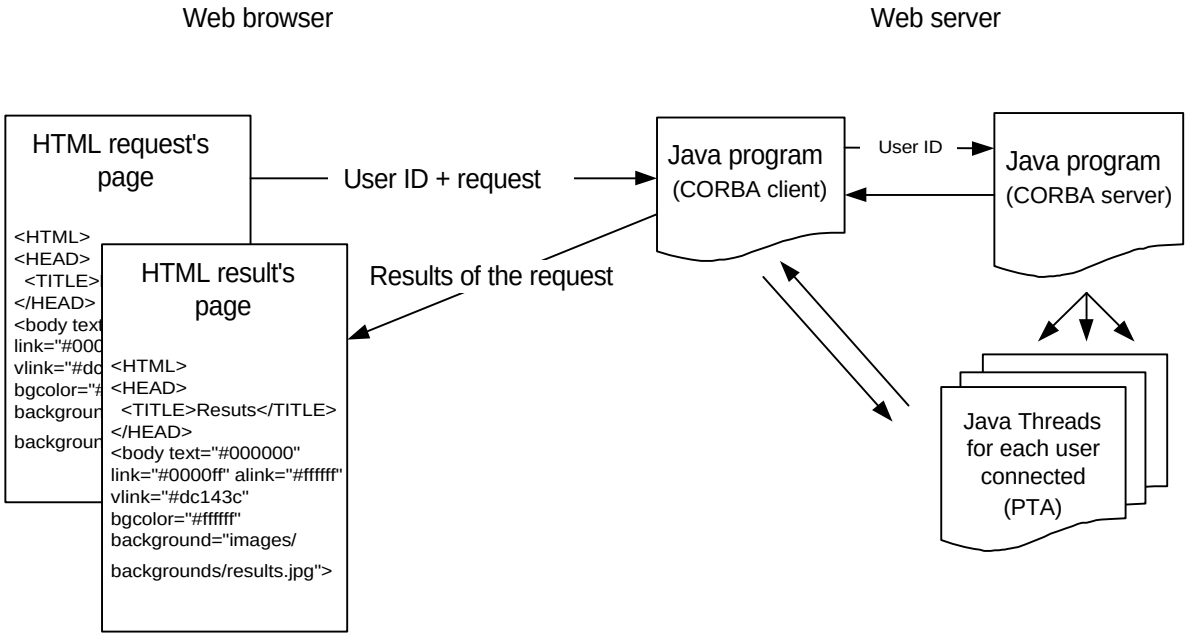
### *4.2.1* Problem

HTTP is a stateless protocol. That means that a server does not keep information about a user. When a user sends a request to a server, he basically asks for a file which most of the time is an html file. This request can be made directly (the user asks for a specific file) or via a Common Gate Interface (cgi) script. A cgi script can call a piece of software and gives him some parameters it has received. We have here a beginning of solution but it is not enough because, when the program is finished, it is closed and it loses all the information about the user. In our case, the PTA needs to build a case retrieval net (see chapter 4.3.3) when the user connects to it. It is a 'heavy weight' operation and if the user has a large amount of past cases in his database, it may take several seconds to build his case retrieval net. Such a delay for each request is not acceptable. Therefore, the server should be able to keep alive a PTA for each connected user.

### *4.2.2* Solution

One possible implementation would be to use the Common Object Request Broker Architecture (CORBA) technology on the server side. CORBA defines a standard architecture for Object Request Brokers (ORBs). An ORB allows the creation of software objects whose member functions can be invoked by client programs located anywhere in the network. These clients can use the objects of the CORBA server as if they were local.

The following diagram shows a possible implementation:




*Web implementation*

A web server can indirectly keep some information about a user by sending him a cookie[1]. The server initiates this cookie and then, each time the client makes a request, he sends his cookie with it automatically. Therefore, by using the cookies, the server can identify the request of one user from another.

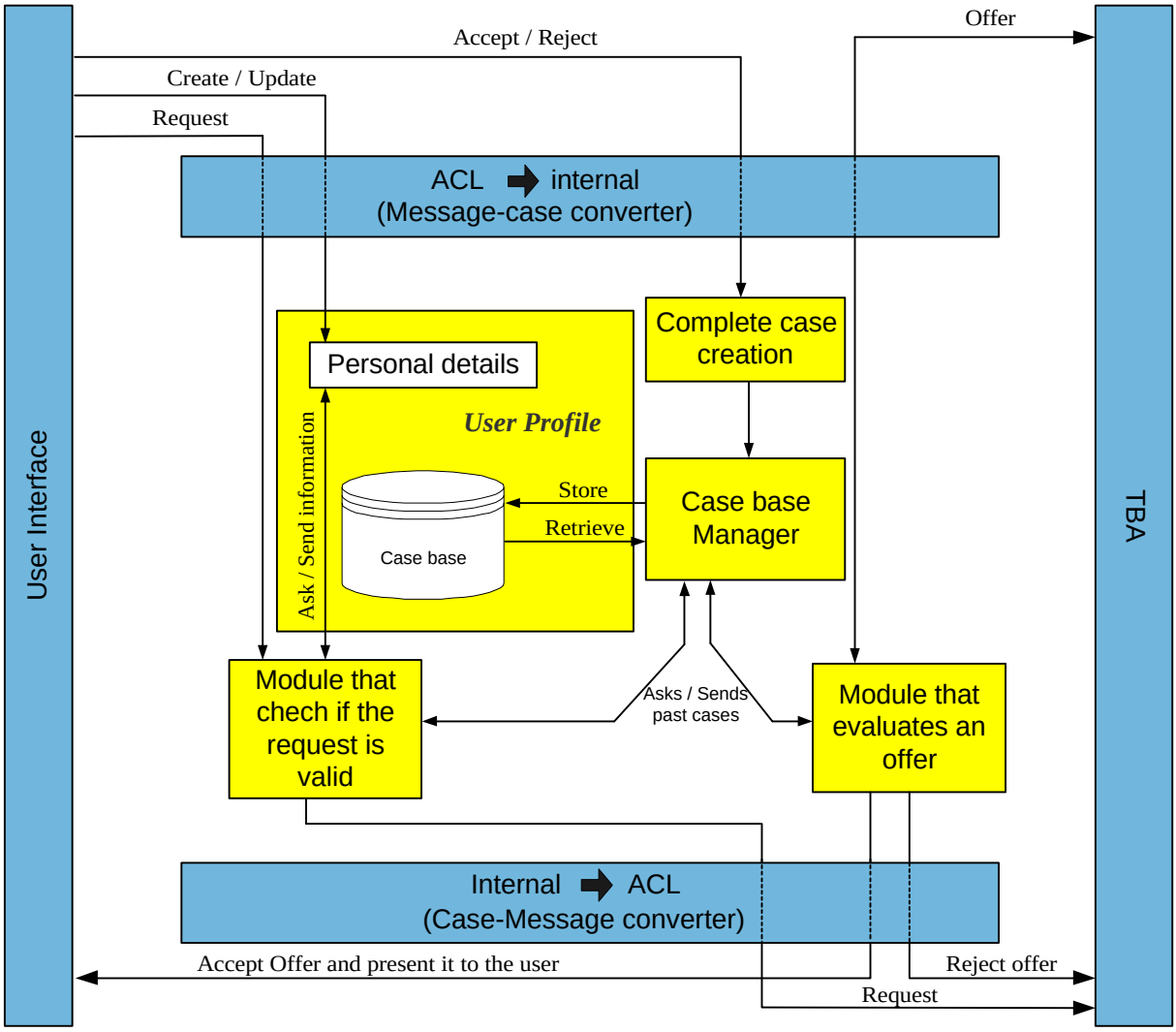
On the server side, there is a piece of software that is always running. It is the CORBA server. This program is in charge of creating a thread for each new user. Basically, a thread would be a PTA. When the CORBA server receives a user name, it checks if a thread is already running for him. If not, a new thread is created.

Then the client has to connect his PTA. With CORBA, it is relatively easy. When the web server receives a request, it runs a piece of software, which is a CORBA client. This piece of software is able to deal with the user's requests. It just has to access his information to do so. As we have seen before, this program receives the user ID. Therefore, in order to get the information needed, it connects itself to the CORBA server. Giving the user ID, it is then able to access to his PTA directly. Then, the program treats the request and sends back the results to the user as an html page.

[1] A cookie is just a text file stored in the client side. In it, there is some information the server wants to keep.

## 4.3 Design of the PTA

The PTA has two major tasks to perform: fulfil an incomplete user's request and choose an offer among those presented by the TBA. The design of the PTA focuses on these two tasks.




*PTA's design*[2]

In the next section, a brief overview of the PTA's components will be covered.

[2] Only the yellow components of this diagram will be implemented in this dissertation

### *4.3.1* Overview of the PTA's components

In order to understand the overall functioning of the PTA, a short description of each module presented in the previous diagram may be useful.

- **Message-case converter**

This module is a simple parser. It takes the information from an ACL message and put it in the appropriate variables in the PTA program.

- **Case-message converter**

This module takes the information from the variables of the PTA program and makes an ACL message from it.

- **User interface**

The interface is independent of the PTA. It gets the information from the user and sends it to the PTA as an ACL message. This interface could be an html page that use forms to get the details of the user's request.

- **Module that check if the request is valid**

This module checks if there is some information missing from the user's request. If there is, it tries to get it from the personal details and/or from the past cases and build a complete request.

- **Module that evaluate an offer**

This module evaluates the TBA's offers. It accepts or refuses them. To accomplish this task, it compares the TBA's offer with the past user's cases.

- **User profile**

The user profile is composed of two types of information: The personal user's information and his preferences.

- **Case base manager**

This module is in charge of retrieving cases given a set of information. The retrieving process will be described later in the dissertation. This module can also add or delete some cases from its database.

- **Complete case creation**

This module is used at the end, when the user has accepted or rejected a proposed case. It creates a final case in which there is all the information and the decision taken by the user. This case is then passed to the case manager in order to store it in the database.

The case-based reasoning system is the core of the PTA. It is used to complete a request and to choose an offer from those presented by the TBA.

### *4.3.2* Case-based reasoning

In the PTA, the case-based system is the heart of the agent. Most of the actions that it performs use the case-based system. It is used to complete a case, to choose a case and event to store the user's preferences.

#### *4.3.2.1 User Profiling*

The user profile is composed of two types of information. There is the personal information that doesn't change very often (name, address, telephone, etc) and the preferences about travel that change depending of user behaviour (user profile).

Because of the static nature of the first set of information, it can be kept in a database or a file. The user should be able to access this information and change it.

The common solution for a user profile is to keep it in a file or a database as a set of values. These pieces of information can be changed depending upon the user's decisions, but at the end, we have a generalisation of the user's behaviour. Only the most common user's decisions are kept in memory. This solution is not good enough for the travel domain. Even if the user behaviour is constant, he can have different preferences depending upon the destinations or the companies he flies with. We need to get the preferences depending upon the context. It seems that using a case-based reasoning system is the best solution.

In the PTA agent, there is no distinct user profile. The past cases are used to complete a user's request. It is assumed than the user's behaviour is constant. That means that for two similar flights, it is likely that the preferences will be the same. Therefore, there is no adaptation of the past cases. They are used as they are retrieved. This solution has the advantage that the completion is adapted to the situation (see scenarios for more details).

#### 4.3.2.2 Comparison of cases

Most of the time, in case-based reasoning, a case is compared with a set of other cases, not only with one. It helps to avoid using corrupted cases or cases that are not especially very close. In the context of the PTA, this technique is not used because it leads to a generalisation of the user behaviour. A user can have different preferences for different flights and we do not want to mix them. The mix of two 'good' flights could give an association of values that does not suit to the user. So, only one case is used to complete a request or make a choice among those offered by the TBA.

#### 4.3.2.3 Evaluation of an offer

When the PTA receives an offer from the TBA, it first compares it with the user request. At this stage, there are three possibilities: the offer corresponds to the request, the offer is close to the request and the offer does not correspond to the offer.

If the offer is very close to the request made by the user, there is no need for further research. The case is accepted by the PTA and presented to the user. This is the simplest case.

If the offer is close to the request but some major criteria are not the same, the PTA needs to determine if it is acceptable or not. In order to do that, it searches through the past cases to see if there is a similar case and if it was accepted. If a case is found, the PTA can take a decision. If not, the case will be presented to the user because there is no way to take a decision.

Finally, if the offer has not the same origin and/or destination as the request, it is obviously not a valid offer. Nevertheless, this is certainly a special offer from the TBA and it may interest the user. Therefore, the PTA researches a case and takes a decision in line with the retrieved case.

When an offer is accepted, it is presented to the user. Then, the user can accept or reject it. The PTA adds the user's decision to the case and finally adds it to the database.

We can notice that the PTA is often searching the database for a past case. In case-based reasoning, the search for a past case can be a problem if there are a lot of cases. To minimise this problem, a case retrieval net has to be used.

#### *4.3.2.4 Learning process*

The learning process of the PTA agent could be classified as a lazy approach. It is lazy in that sense that it stores the cases in a database and uses them at run time. There is no analysis of the cases until the PTA has to answer to a user's request. The PTA has to get a few cases before being really efficient. This is not a real drawback because that just means that for the first requests, the PTA will ask the user and will not take any decision by itself. Then, it will be able to take some decisions based on the user's choices. This lazy approach is a characteristic of case-based reasoning. This approach has also the advantage that the order in which the cases arrive in the database has no importance.

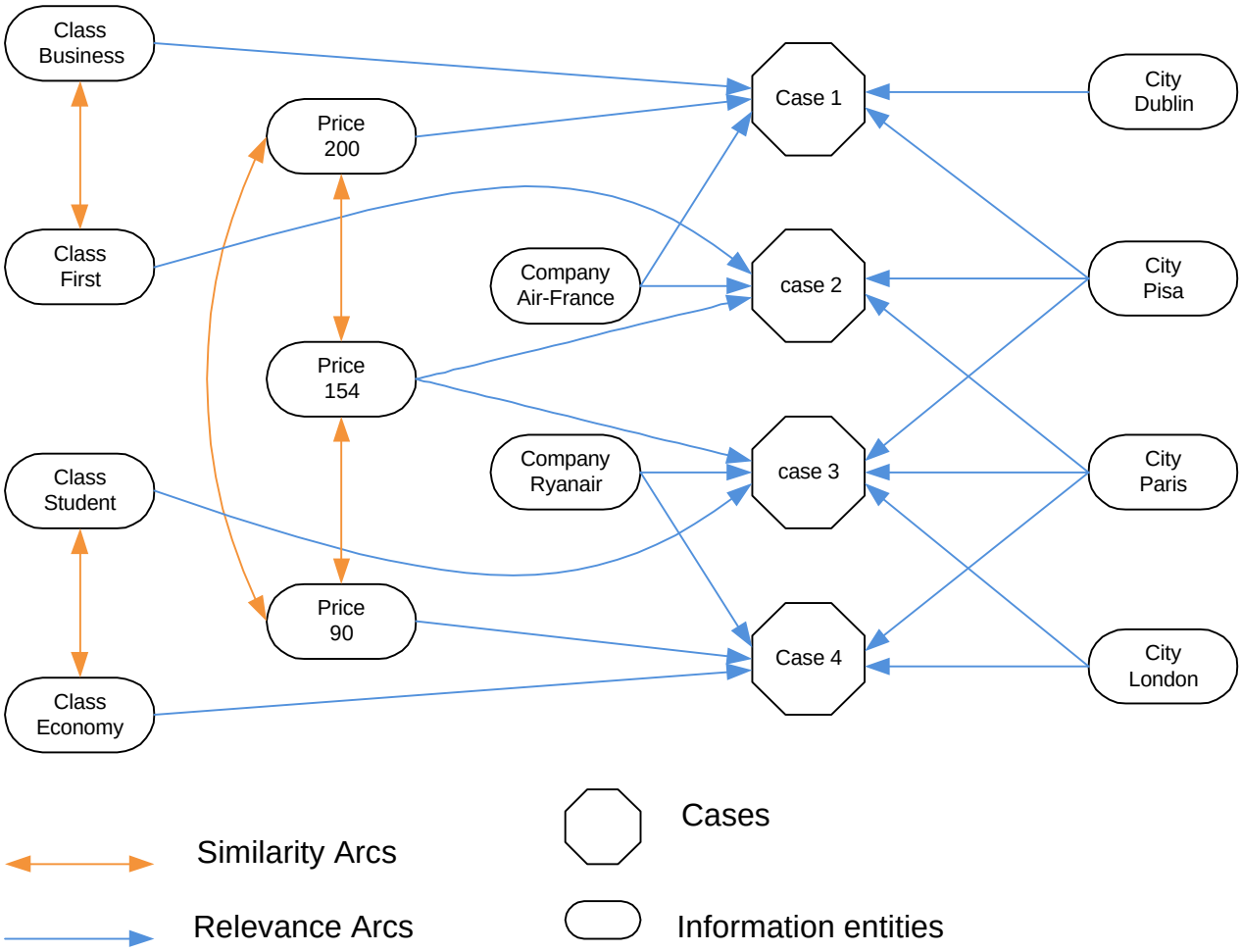
### *4.3.3* Case retrieval net

Retrieval is a very important part in case-based reasoning. It has a great impact on the efficiency of a system. There are many ways to retrieve information but case retrieval net seems to be the best one for the PTA case. A Case Retrieval Net (CRN) is a memory model that is able to deal with ambiguous terms, can support the concept of information completion and can handle case bases of reasonable size efficiently [6].

A case retrieval net is composed of two components: the Information Entities (IE) and the cases. An information entity represents the most basic knowledge item in the domain. It is represented as an attribute-value pair. So, a case is described as a unique case descriptor and a set of information entities. A query is also represented as a set of information entities. The case retrieval net is composed of as many information entities as the number of different pieces of information present in the domain. Information entities may be connected by similarity arcs if they are close. These arcs have a value that represents the level of similarity between two information entities. Cases are designated by relevance arcs from the information entities. In the case retrieval net, the structure of the cases is broken. For a given case, there can be two or three cities of origin for example, but there is no way to know from which leg of the trip they are.

Given this structure, the activation of the case retrieval net is performed by:

1. Activating the information entities given in the query
2. Propagating activation according to similarity through the net of information entities
3. And collecting the achieved activation in the associated nodes. [6]




Part of a case retrieval net

# Chapter 5

# Implementation

In this chapter, the decisions made during the implementation will be presented. The implementation of the case-based reasoning will be detailed as well as the case retrieval net.

## *5.1 Decisions taken during the implementation*

The first objective of this dissertation was to build the PTA in it's all. But, due to the lack of pieces of software that would allow the building of a case retrieval net, this objective has been changed.

The implementation of a fully functional PTA would have been too long so, some decisions were taken during the implementation. The implementation now focuses only on the engine of the PTA. The communication with the user via a web interface and with the TBA were not implemented.

This focusing on the engine is partially due to a lack of pieces of software that allows the building a case retrieval net. A piece of software called CBR Works was tested, but it was determined that it was not suitable for the PTA agent. It is not well defined how this software manages the cases, but it seems that it is not using retrieval net. What's more, it has its own interface and it would have been difficult to integrate it to the web design described before. That is why the case retrieval net has been implemented from scratch.

During the implementation, the case retrieval net has been adapted. This adaptation concerns the similarity arcs. Building such arcs appeared to be tricky and time consuming on the assumption that most of them would not be used. So, they were replaced by a module that is able to return an information entity similar to the one it receives as a parameter.

## *5.2 Storage of the cases*

To store the cases, two solutions were possible: a file or a database. The PTA should be on a server in order to be accessible. So a file solution would have been too greedy in memory. Each PTA would have to read its own file, maybe all of it at the same time. What is more, a file is a static solution. Once the structure is defined, it is very difficult to adapt it if there are some changes in the cases. Therefore, the database solution was obvious. It is able to handle a lot of queries at the same time and it is easier to maintain. So, the PTA uses an Oracle database to store the case of the user. The purpose of the Oracle database is only for storage. It will not be used to check the integrity of keys or other security options of Oracle (this could be done after as an optimisation).

Three tables are needed to describe a case. Because it can be made of one or many segments, the segments are kept in separate table. That is why a third table is needed in order to make a link between the cases and the segments.

There is the description of these tables:

**Table "Cases"**

This table has the general information about the case i.e. the price or if it was accepted or not.

| Field name | Type | Example of values |
|---|---|---|
| | | |
| ID | Boolean | 1 |
| Segments | Integer | 1 |
| Price | Integer | 70 |
| Currency | String | IRPound |
| Accepted | String | Yes |

**Table "Hops"**

This table has the detailed information about a segment of a trip.

| Field name | Type | Example of values |
|---|---|---|
| | | |
| ID | Integer | 1 |
| Origin_Country | String | France |
| Dest_Country | String | Ireland |
| Origin_City | String | Paris |
| Dest_City | String | Dublin |
| Origin_Airport | String | Orly |
| Dest_Airport | String | Dublin |
| Departure_date | String | 01/06/99 |
| Return_Date | String | 05/06/99 |
| Departure_Time | String | 08.45 |
| Return_Time | String | 21.30 |
| Class | String | Business |
| Seat | String | Window |
| Meal | String | Yes |
| Company | String | Air-France |

**Table "Cases_Hops"**

This table is a link between the cases and the segments. It is composed of the primary keys of the two other tables.

| Field name | Type | Example of values |
|---|---|---|
| | | |
| ID_Case | Integer | 1 |
| ID_Hop | Integer | 1 |

## *5.3 Comparison of cases*

The comparison of two cases is a very important part of the PTA agent. It is the result of a comparison that will determine if a past case is close enough to the current one or not. In this implementation, the PTA has a module in charge of this comparison.

As it has been said before, a case is composed of several information entities. Therefore, the comparison of two cases consists of comparing all the information entities. The similarity of two cases is the sum of the similarities of each information entity of these cases. In order to compare two cases, a weight for each information entity present in the cases is used. These weights are defined by the user and are stored in a table of the database. That means that if the user gave a large importance to the company, two cases with the same company will be considered as close even if there are a lot of other differences.

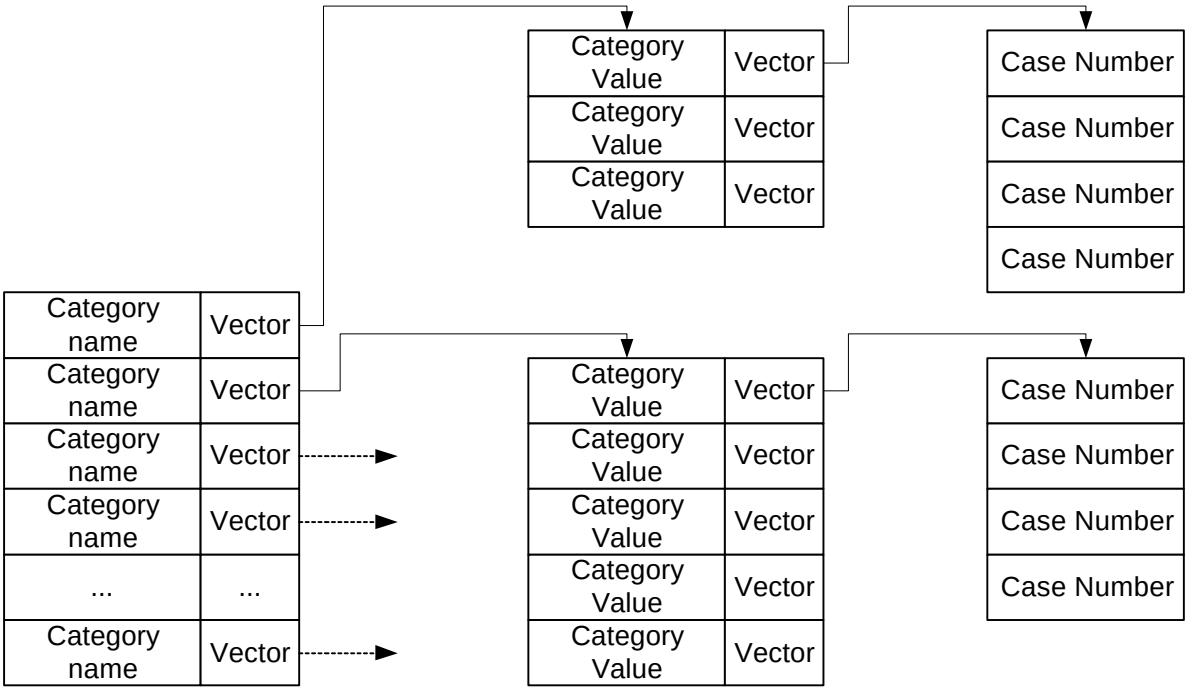
In the PTA, the comparison is composed of two steps. The first one is a direct comparison of the two cases. The similarities are not used at this stage. Then, if the result of the comparison is not satisfactory, a new comparison is made using the similarities. For example the criteria ‘origin_city’ will be changed into ‘destination_city’.

## *5.4 Case retrieval net*

The implementation of the case retrieval net was made using the Java language. Given the nature of the case retrieval net (set of pointers) its implementation was tricky. In Java, there is not the concept of pointers like in C or C++. So, to simulate a net of pointers Vectors[3] has been used.

The structure of the case retrieval net is composed of three vectors.

The first one contains objects that have a category name and another Vector. The category name is a String. The second Vector contains objects that have a category value and a Vector. The category value is also a String. Finally, the last vector contains Strings that represents some case numbers.




Structure of the case retrieval net

[3] The Vector class implements a resizable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.

We can notice that in this representation, there are no similarity arcs. This is because a module that returns some information entities similar to the one it has received has replaced them. For example, if the module receives an entity ‘class:Student’, it will return ‘class:Economy’. Or it will return ‘Destination_City:Dublin’ if it has received ‘Origin_City:Dublin’.

This method has several advantages over the similarity arcs. It is assumed that the user is more or less constant in his choices. Therefore, most of the time the retrieved case will be close to the user request and the similarity arcs will not be necessary. Assuming that, the first advantage of a module against the similarity arcs is a subsequent gain of time. We don’t have to manage the similarity arcs for each new case. Another advantage is that a module is easier to manage than arcs. If there is a new kind of similarity, we do not have to change all the cases, only the module.

For the moment, these similarities are hard coded. But, in future work, it could be improved. The module could take the similarities in a table of the database or in a file. The user could also customize it by defining some new similarities.

## 5.5 Tests

The tests consisted in playing some scenarios and in seeing if the results were in accordance with what was expected. A database with a few flights was set up for these tests.

**1st Scenario: Completion of a request for a regular flight**

*Description:*
This test consists of asking for the program for a regular flight. The request is a simple Dublin-Paris. In the database, there are 4 cases that deal with theses destinations.

*Test: An incomplete request for a Dublin-Paris trip is made*
Only a few criteria were given:

Hops: 1
Price: 90
Country of origin: Ireland
Country of destination: France
City of origin: Dublin
Country of destination: Paris
Departure date: 15/11/1999
Return date: 29/11/1999

*Expectation:*
The program should find a case similar to the query and use it to complete the user's request.

*Results:*
The program detected that the request was not complete and searched for a close case using the case retrieval net. A case that matches exactly with the request was found and used to complete the request. A full request was build using this retrieved case.

**2nd Scenario: Completion of a request for a flight with a new destination**

*Description:*

This test consists of asking the program for a flight with a new destination. The request is Dublin-Berlin. In the database, there are no cases that deal with this destination.

*Test 1: An incomplete request for a Dublin-Berlin trip is made but there is no case close enough among the past cases.*

The following request was given to the program:

Hops: 1
Price: 90
Country of origin: Ireland
Country of destination: Germany
City of origin: Dublin
Country of destination: Berlin
Departure date: 15/11/1999
Return date: 29/11/1999

*Expectation:*

The program should not find a case similar to the query. Therefore, it should ask the user for more details.

*Results:*

The program retrieved some cases, but they were all under 50% of similarity, so, it asked the user to give more information.

*Test 2: An incomplete request for a Dublin-Berlin trip is made and a case is found among the past cases.*

The following request was given to the program:

Hops: 1
Price: 90
Country of origin: Ireland
Country of destination: France
City of origin: Dublin
City of destination: Paris
Departure date: 15/11/1999
Return date: 29/11/1999
Class: Student
Seat: Window

*Expectation:*

The program should find a case similar enough to the query and it should use it to complete the user's request.

*Results:*

A case close to 70% was found in the database. It was a Dublin-Pisa trip. The program used it to complete the request. It got all the missing information except one: The airport of destination (because it is a new destination). Therefore, it put the value 'unknown' for the criteria 'destination_airport'.

### 3rd Scenario: Choosing an offer

*Description:*

This test consists in checking the validity of the program's choices. Three possible situation will be tested: The offer is very close to the request, the offer is not very close to the request and the offer has a different origin and/or destination than the request.

*Test 1: The offer is very close to the request*

The offer proposed to the program is close to the request at 90%.

*Expectation:*

If the offer is very close to the request, the program should accept it immediately, without looking for a similar case in the database.

*Results:*

The program accepted the request without looking at the database.

*Test 2: The offer is not very close to the request and there is a similar case in the database*

The offer proposed to the program was close to the request at 60%.

*Expectation:*

If the offer is not very close to the request, the program should search for a similar case in the database and use it to make a decision.

*Results:*

The program searched for a case similar to the offer in the database. A case with a similarity of 85% was found. This past case was refused. The program rejected the offer and did not presented it to the user.

*Test 3: The offer is not very close to the request and there is no similar case in the database*
The offer proposed to the program was close to the request at 60%.

*Expectation:*
If the offer is not very close to the request, the program should search for a similar case in the database and use it to make a decision. If no close case is found, it should propose the offer to the user.

*Results:*
The program searched for a case similar to the offer in the database but found nothing. It presented the offer to the user.

*Test 4: The offer has a different origin and/or destination and there is a similar case in the database*
The request was a Dublin-London but the offer was a Dublin-Paris.

*Expectation:*
If the offer is different to the request, it does not mean that is should be rejected immediately. It just means that it may be a special offer from the travel broker agent. The program should search for a similar case in the database in order to make a decision about this offer. If a close case is found and if it was accepted, it should be presented to the user as a special offer.

*Results:*
The program detected the difference of destination and searched for a case in the database. A case similar at 65% was found. It was a case that had been accepted before. The program presented the case to the user as a special offer.

*Test 5: The offer has a different origin and/or destination and there is no similar case in the database*

The request was a Dublin-London but the offer was a London-Paris.

*Expectation:*

The program should search for a similar case in the database in order to make a decision about this offer. If no case is found, the program should not be able to make a decision. Therefore, it should present it to the user.

*Results:*

The program detected the difference of origin and destination and searched for a case in the database. No case was found. The program presented the case to the user as a special offer.

### 4th Scenario: error handling

*Description:*

This test consists in checking if the program can handle errors. Two kinds of errors will be tested: missing information and unexpected values.

*Test 1: missing information*

The program receives an offer where some pieces of information are missing (the company and the seat). The offer is strictly the same as the user's request except these two information entities.

*Expectation:*

The program should not even notice the lack of information. The two cases should be compared as normal. The information entities with the missing values should just be considered as different.

*Results:*

The program compared the user's request and the offer. The result of the comparison was 80%. The missing values were just considered as some different values from the request.

*Test 2: unexpected value*

Some values from the user's request were input false on purpose. Two values were changed: 'yes' was input for the category 'window' and 'aisle' for the category 'meal'.

*Expectation:*

As for the missing values, the program should not notice the false values. The two cases should be compared as normal. The information entities with the false values should just be considered as different.

*Results:*

The program compared the user's request and the offer. The result of the comparison was 75%. The false values were just considered as some different values from the request.

## *5.6 Conclusions based on tests*

The previous tests show the benefits of case-based reasoning in the context of the PTA agent. An agent using a case-based reasoning is effectively very flexible. It is able to handle missing information and unexpected values. It is also capable of taking decisions adapted to the user by using past cases of this user.

Another interesting characteristic of the PTA agent is that it 'extensible'. The implementation is not completely finished at this time so it has not been tested, but considering how the case retrieval net is build, it is easy to imagine that a new information entity can easily be added to the cases. If a new case is added with new information entities, the comparison with the past cases will be fine. For the agent, the case that has not the new information entity is just an incomplete case, and we have seen before that it can handle this situation very easily (see scenario 3).

# Chapter 6

# Conclusion

This chapter gives a summary of the work that has been done during this dissertation. It also describes the remaining work to get a complete Personal Travel Assistant. Finally, the future work needed to implement the complete FIPA scenario will be discussed.

## *6.1 Work accomplished*

The main achievement of this dissertation is a complete design of a PTA agent in the context of the FIPA scenario. The exact nature of the PTA agent has been defined after a review of the different possible types. During this dissertation, a good understanding of agents was attained.

The engine of the PTA agent was implemented using case-based reasoning technology. The program is able to complete a user's request and choose an offer in line with the user's preferences. To accomplish these tasks, it uses a case-based reasoning system. To store the cases, an Oracle database has been used and, to retrieve the cases, a case retrieval net has also been implemented. During this implementation, knowledge of Java programming was improved. More precisely how to drive an Oracle database from a Java program. A good understanding of the retrieving problem in case-based reasoning has also been reached.

A set of tests has been run using the implemented program. The results have demonstrated that a case-based reasoning is very efficient for an agent like the PTA agent. It appeared that is more adaptable than a rule-based system and it is able to learn from the choices of the user.

## 6.2 Remaining work

The implementation is not completely finished and some improvements could be done. Here is a list of the remaining work:

- The program has to store the new cases in the database.
- The weight of the information entities has to be managed dynamically by the program.
- The similarity module could take its information from a file or a table in the database.
- The database could be managed properly. E.g. it could check the integrity of the values.

## 6.3 Future work

To transform this program into a fully working agent, some work still needs to be done. First, it has to be able to communicate with other agents. The modules that transform an ACL message into internal variables and the one that do the opposite should be implemented. Then, the agent needs to be accessible from anywhere, so its interface has to be built in order to make it accessible via a web page. The server using CORBA should also be implemented. And, finally, the other agents should be implemented too.

An improvement could be done on the comparison of cases. For the moment, the program uses only one past case to complete a request or choose an offer. But, when there are many similar cases available in the same category, their integration and aggregation may be done in order to get a better solution.

# References


1. Aamodt A., Plaza E.: '*Case-Based Reasoning: Foundational Issues, methodological Variations, and System Approaches. AI Communications*', IOS Press, Vol. 7: 1, pp. 39-59, (1994)
2. Brown M.G.: 'A memory model for case retrieval by activation passing', Ph.D. dissertation, Department of Computer Science, University of Manchester, technical report UMCS-94-2-1, (1993)
3. Caroline C. Hayes: '*Agents in a Nutshell A Very Brief Introduction*', IEEE Transactions on Knowledge and Data Engineering, vol. 11, No. 1, (January/February 1999)
4. FIPA - Foundation for Intelligent Physical Agents, '*FIPA 97 Draft Specification part 4, Personal Travel Assistance*', (1997)
5. Hewitf C: '*Viewing Control Structures as Patterns of Passing Messages*', Artificial intelligence, 8, No3, pp 323-364 (1977)
6. Lenz M., Auriol E. and Manago M.: '*Diagnosis and Decision Support*'. In Lenz, M., Bartsch-Sporl, Burkhard H. D. and Wess S. (eds.), Case-Based Reasoning Technology. From Foundations to Applications, Springer (Lecture Notes in Computer Science; Vol. 1400: Lecture Notes in Artificial Intelligence), (1998)
7. Luger G. F and Stubblefield W. A: '*Artificial intelligence: Structures and strategies for complex problem solving*', Third edition, (1997)
8. Nwana H. S. and Ndumu D.T.: '*An introduction to Agent Technology*', Software Agent and Soft Computing, (1996)
9. Waszkiewicz P., Cunningham P and Byrne C: '*Case-based User Profiling in a Personal Travel Assistant*', (1999)

# Appendices

## Scenarios

The purpose of these scenarios is to show the possibilities of the PTA agent. We will assume that there are a large number of cases past cases stored in the database. The structure of the messages or the structure of the cases may be different in the implementation.

# Scenario 1
# The user makes a request for a regular flight.

In this scenario, the following ability of the PTA will be shown:

*Ability to find the cases quickly using the Case Retrieval Net*

In AI, search is very important. In a case based reasoning, search is maybe more important than in other AI field because the number of past cases can be huge. The Case Retrieval Net is an efficient way to search in the cases. It could be compared to a neural network where the activation of certain nodes (IE) gives an immediate solution (a case).

*Ability to manage the preferences easily with a case based*

The past cases are used to complete a user's request. There is no storage of the user's preferences somewhere else than in the past cases.

**Request from the user**

The user wants to go to Paris from Dublin. It is a trip he does very often for his business. Using the interface, he gives the following information:

From: Dublin
To: Paris
Departure: 01/07/99
Return: 03/07/99

Because the user makes this trip very often, he does not want to always give all the information. He just gives the minimum.

**The PTA fills the request**

When the PTA receives this request, it is an ACL message like this:

```
(request
 :sender PersonnalTravelAgentInterface
 :receiver PersonalTravelAgent
 :content (:tripsummary
           (:origin (:countrycode IR :city Dublin)
            :destination (:countrycode FR :city Paris)
            :time ( :departure (:date 990701)
            :Return (:date 990703))))
 :ontology fipa-PTA
 :language xxx
)
```

The first thing the PTA performs is to parse this message in order to be able to work on it. When this is done, all the information is now in some variables of the PTA:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := FR
DestCity := Paris
DepartureDate := 990701
ReturnDate := 990703
```

But some other variable are not defined:

```
Company := null
Class := null
Seat := null
Meal := null
DepartureTimeAfter := null
DepartureTimeBefore := null
ReturnTimeAfter :=null
ReturnTimeBefore :=null
Hops := null
...
```

This missing information is stored in the past cases. The PTA retrieves the closest past case and then completes the missing information. In order to do this, the agent uses the case retrieval net. Each relevant piece of information is used to "fire" the information entities of the net and finds the relevant cases.
(For a detailed description of the case retrieval net, see chapter 4.3.3)

The closest case found is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := FR
DestCity := Paris
DepartureDate := 01/10/1997
ReturnDate := 15/10/1997
Company := AirFrance
Class := Student
Seat := Window
Meal := Yes
DepartureTime := 12.35
ReturnTime := 14.25
Hops := 1
Price := 90
Currency := IRPound
...
Accepted Yes
```

Using this case, the PTA can now complete the request. The PTA converts the exact hours into a range of hours in order to get more offers. Now, the request is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := FR
DestCity := Paris
DepartureDate := 990701
ReturnDate := 990703
Company := AirFrance
Class := Student
```

```
PreferedSeat := Window
Meal := Yes
DepartureTimeAfter := 11.35
DepartureTimeBefore := 13.35
ReturnTimeAfter := 13.25
ReturnTimeBefore := 15.25
Hops := 1
Price := 90
Currency := IRPound
...
```

Now, there is enough information to make a request to the TBA.

**The PTA sends an ACL message to the TBA**

The PTA has to build an ACL message from its variables. The ACL message is:

```
(Query-ref
 :sender PersonnalTravelAgent
 :receiver TravelBrokerAgent
 :content (:tripsummary
           (:origin (:countrycode IR :city Dublin)
            :destination (:countrycode FR :city Paris)
            :Hops 1
            :time (:departure
                     (:date 990701
                      :hour (:After 11.35 :Before 13.35))
                   :Return
                     (:date 990703
                      :hour (:After 13.25 :Before
                 15.25)))
            :Company AirFrance
```

```
                :Class Student
                :Seat Window
                :Meal Yes
                :Price 90
                :Currency IRPound))
  :ontology fipa-PTA
  :language xxx
)
```

Then, the PTA sends this message to the TBA.

**The PTA receives an answer from the TBA**

The PTA receives the offer(s) of the TBA. In this scenario, three offers are made:

1st offer:

```
(Query-ref
  :sender TravelBrokerAgent
  :receiver PersonalTravelAgent
  :content (:tripsummary
            (:origin (:countrycode IR :city Dublin)
             :destination (:countrycode FR :city Beauvais)
             :Hops 1
             :time (:departure
                      (:date 990701 :hour 10.45)
                    :Return
                      (:date 990703 :hour 20.45))
             :Company Ryanair
             :Class None
             :Seat None
             :Meal No
             :Price 49
```

```
            :Currency IRPound))
 :ontology fipa-PTA
 :language xxx
)
```

2nd offer:

```
(Query-ref
 :sender PersonnalTravelAgent
 :receiver TravelBrokerAgent
 :content (:tripsummary
           (:origin (:countrycode IR :city Dublin)
            :destination (:countrycode FR :city Paris)
            :Hops 1
            :time (:departure
                     (:date 990701 :hour 08.35)
                   :Return
                     (:date 990703 :hour 21.00))
            :Company AirFrance
            :Class Business
            :Seat aisle
            :Meal Yes
            :Price 160
            :Currency IRPound))
 :ontology fipa-PTA
 :language xxx
)
```

3rd offer:

```
(Query-ref
 :sender PersonnalTravelAgent
 :receiver TravelBrokerAgent
 :content (:tripsummary
           (:origin (:countrycode IR :city Dublin)
            :destination (:countrycode FR :city Paris)
            :Hops 1
            :time (:departure
                     (:date 990701 :hour 08.35)
                   :Return
                     (:date 990703 :hour 21.00))
            :Company AirFrance
            :Class Student
            :Seat Window
            :Meal Yes
            :Price 96
            :Currency IRPound))
 :ontology fipa-PTA
 :language xxx
)
```

**The PTA evaluates the offers**

The PTA has now to evaluate these two offers. Using the case retrieval net, it retrieves the closest past choices and evaluates each offer.

1st offer:

This offers differs from the request on 4 points: The company, the meal, the departure time and the price.

A similar case is retrieved:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := FR
DestCity := Beauvais
DepartureDate := 980704
ReturnDate := 980715
Company := Ryanair
Class := Economy
Seat := Window
Meal := No
DepartureTime := 07.00
ReturnTime := 17.00
Hops := 1
Price := 70
Currency := IRPound
...
Accepted Yes
```

This case was accepted and it is close enough to the current one. So, this case will be presented to the user in order to get his approval.

2nd offer:

This offer is close to the request. Only the seat and the price differ. Nevertheless, because of the importance of the price, a similar case will be searched in the database before presenting the case to the user.

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := FR
```

```
DestCity := Paris
DepartureDate := 980802
ReturnDate := 980805
Company := AirFrance
Class := Business
Seat := Aisle
Meal := Yes
DepartureTime := 08.45
ReturnTime := 21.05
Hops := 1
Price := 300
Currency := IRPound
...
Accepted No
```

Last time this case occurred, it was rejected (Maybe because of the price). So, this time it will be rejected too. This case will not be presented to the user.

Between the request and this proposition, only the price and the seat differ. Despite these few differences, the case is rejected. That means that these two criteria are more important than the others. The PTA will adjust the weight of these criteria in its retrieval net.

3rd offer:
Basically, this offer is identical to the request. There is a difference of price, but it is small. So, there is no need to search a close case in the database. The offer is directly presented to the user.

**The PTA stores the offers**

When the used give his answer for an offer, the PTA completes the case and stores it in its case base. The only thing the PTA adds here is the answer of the user. It is just a new field with two value: Yes or No.

# Scenario 2
# The user ask for a kind of trip he has done a long time ago

In this scenario, the following ability of the agent will be shown:

Ability to "remember" unusual cases

In a Rule based system, a general behaviour of the user is defined. When a decision has to be done, it is based on the decision the most often taken by the user. However, if there are some special conditions that lead to a special solution, this case will not be remembered and, if it occurs again, the proposed solution will not be adapted. In a rule based system, each different case is kept, even if this case is very unusual. No assumption is done on the importance of a case. Because of this, the previous problem is solved. If a special scenario occurs a second time, the past case is retrieved and the solution is adapted to what was done before.

**Request from the user**

The user is a French student in Dublin. He has some family in Pisa but has not seen them for a long time and would like to take advantage of his holidays to see them again. Last time he went in Pisa was three years ago (At this time, he already has a PTA). The user asks the following to his PTA:

From: Dublin
To: Pisa
Departure date: 01/08/99
Return date: 31/08/99

Because this trip is not a regular one so the user prefer to give more details for his request:

Departure time: After 9am but before 1pm
Return time: after 1pm but before 4pm
Class: Student

**The PTA fills the request**

This part will not be fully detailed. For more details, see the first scenario.

The full request made by the PTA is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := IT
DestCity := Pisa
DepartureDate := 990801
ReturnDate := 990831
Company := Ryanair
Class := Student
PreferedSeat := none
```

```
Meal := No
DepartureTimeAfter := 09.00
DepartureTimeBefore := 13.00
ReturnTimeAfter := 13.00
ReturnTimeBefore := 16.00
Hops := 1
Price := 100
Currency := IRPound
...
```

The case retrieved is the one stored the last time the user went to Pisa (three years before).

**The PTA sends it to the TBA**

This part will not be detailed. For more details, see the first scenario.

**The TBA makes an offer**

This part will not be detailed. For more details, see the first scenario.

**The PTA receives an answer from the TBA**

The PTA receives the offer(s) of the TBA. In this scenario, three offers are made:

1st offer:

```
(Query-ref
 :sender TravelBrokerAgent
 :receiver PersonalTravelAgent
 :content (:tripsummary
   (:origin (:countrycode IR :city Dublin)
    :destination (:countrycode IT :city Pisa)
    :Hops 2
    :SegmentDetails
       (:Segment1
```

```
            (:origin (:countrycode IR :city Dublin)
             :destination (:countrycode FR :city Beauvais)
             :time ( :departure (:date 990801 :hour 1230)
             :Return (:date 990831 :hour 1800))
             :Company Ryanair
             :Class None
             :Seat None
             :Meal No)
         :Segment2
            (:origin (:countrycode FR :city Paris)
             :destination (:countrycode IT :city Pisa)
             :time ( :departure (:date 990801 :hour 1800)
             :Return (:date 990831 :hour 1230))
             :Company AirFrance
             :Class Student
             :Seat Window
             :Meal Yes)))
     :Price 120
     :Currency IRPound))
 :ontology fipa-PTA
 :language xxx
)
```

2nd offer:

```
(Query-ref
 :sender TravelBrokerAgent
 :receiver PersonalTravelAgent
 :content (:tripsummary
   (:origin (:countrycode IR :city Dublin)
    :destination (:countrycode IT :city Pisa)
    :hops 2
```

```
      :SegmentDetails
         (:Segment1
             (:origin (:countrycode IR :city Dublin)
              :destination (:countrycode GB :city Stansted)
              :time ( :departure (:date 990801 :hour 1000)
              :Return (:date 990831 :hour 1400))
              :Company Ryanair
              :Class None
              :Seat None
              :Meal No)
          :Segment2
             (:origin (:countrycode GB :city Stansted)
              :destination (:countrycode IT :city Pisa)
              :time ( :departure (:date 990801 :hour 1300)
              :Return (:date 990831 :hour 0900))
              :Company Ryanair
              :Class None
              :Seat None
              :Meal No)))
      :Price 110
      :Currency IRPound))
  :ontology fipa-PTA
  :language xxx
)
```

3rd Offer:

```
(Query-ref
  :sender PersonnalTravelAgent
  :receiver TravelBrokerAgent
  :content (:tripsummary
             (:origin (:countrycode IR :city Dublin)
```

```
                :destination (:countrycode IT :city Pisa)
                :Via none
                :time ( :departure (:date 990701 :hour 0835)
                        :Return (:date 990703 :hour 2100))
                :Company AirFrance
                :Class Student
                :Seat aisle
                :Meal Yes
                :Price 220
                :Currency IRPound))
  :ontology fipa-PTA
  :language xxx
)
```

**The PTA evaluates the offers**

The PTA has now to evaluate these three offers. Using the case retrieval net, it retrieves the closest past choices and evaluates each offer.

Three cases are retrieved.

1st offer:

The closest case retrieved is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := IT
DestCity := Pisa
ViaCountry := FR
ViaCity := Beauvais
DepartureDate := 960704
ReturnDate := 960715
Node1:
```

```
Company := Ryanair
Class := None
Seat := None
Meal := No
DepartureTime := 07.00
ReturnTime := 21.00
Node2:
Company := AirFrance
Class := Student
Seat := Aisle
Meal := No
DepartureTime := 12.00
ReturnTime := 12.00
Price := 105
Currency := IRPound
...
Accepted Yes
```

This case was accepted and it is basically the same as the one proposed by the TBA. So, this case will be presented to the user in order to get his approval.

2nd offer:

This case is close to the offer, but the destination was not the same:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := DE
DestCity := Frankfurt
ViaCountry := GB
ViaCity := Stansted
DepartureDate := 970708
```

```
ReturnDate := 970725

Node1:
    Company := Ryanair
    Class := None
    Seat := None
    Meal := No
    DepartureTime := 07.00
    ReturnTime := 21.00

Node2:
    Company := Lufthansa
    Class := Student
    Seat := window
    Meal := Yes
    DepartureTime := 10.30
    ReturnTime := 17.00

Price := 98
Currency := IRPound
...
Accepted Yes
```

This case has enough similarities with the offer of the TBA. This case had been accepted, so the offer will be presented to the user.

3rd offer:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := IT
```

```
DestCity := Pisa
DepartureDate := 960704
ReturnDate := 960715
Company := AirFrance
Class := Student
Seat := Aisle
Meal := Yes
DepartureTime := 08.45
ReturnTime := 21.05
Via := none
Price := 300
Currency := IRPound
...
Accepted No
```

This case is the same as the one proposed except that the price is different. It is certainly one of the cases proposed 3 years before when the user asked for the same trip. This offer has been rejected at this time so it will not be presented to the user.

**The PTA stores the offers**

When the used give his answer for an offer, the PTA completes the case and stores it in its case base. The only thing the PTA adds here is the answer of the user. It is just a new field with two value: Yes or No.

In this scenario, it is shown that even a situation that happen a very few times is managed with the same accuracy as a scenario that happen every week.
We also see that a case that has not the same destination but has some similarities can be retrieved and used to take some decisions. This is useful if the request from the user is for a new destination.

## Scenario 3
## The user asks for a trip but use some new preferences

In this scenario, the following ability of the agent will be shown:

Ability to add some new criteria without any modification

If a new criterion is used, this should be transparent to the user. The case base manager just adds this criterion as an IE in the retrieval net. The relevance links can be defined by default at the beginning.

**Request from the user**

The user is now using a PTA for a long time. There are some new improvements on planes and now, the company offer some new services. The user can now ask for a smoking place, he can ask for a special meal or even ask for special seat where he could plug his laptop. In order to follow these evolutions, the interface between the user and the PTA has been adapted.
Here is a request the user can do now:

From: Dublin
To: New York
Departure date: 01/01/2000
Return date: 15/01/2000
Departure time: After 9am but before 1pm
Return time: after 9pm but before 1pm
Class: Business
Networked Seat: Yes
Meal: Yes
Type of Meal: Chinese
Smoking: No

**The PTA fills the request**

This part will not be fully detailed. For more details, see the first scenario.

As we have seen in the previous scenarios, the PTA retrieves the closest case in order to complete it. The problem here is that some details given by the user do not exist on the past cases. In fact, this is not a problem at all. The PTA fire all the Information Entities of the retrieval net it knows. Each criterion the PTA does not know is kept in memory and will be added to the CRN at the end.

The retrieved case will be a case as close as possible without the new information. The retrieved case is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := US
DestCity := New York
DepartureDate := 980504
ReturnDate := 980523
Company := Air America
Class := Business
Seat := Window
Meal := Yes
DepartureTime := 08.45
ReturnTime := 08.30
Via := none
Price := 500
Currency := IRPound
...
Accepted Yes
```

The full request made by the PTA is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := US
DestCity := New York
DepartureDate := 20000101
ReturnDate := 20000115
Company := AirAmerica
```

```
Class := Business
PreferedSeat := window
Meal := Yes
DepartureTimeAfter := 07.45
DepartureTimeBefore := 09.45
ReturnTimeAfter := 07.30
ReturnTimeBefore := 09.30
Via := None
Networked Seat := Yes
Type of Meal := Chinese
Smoking := No
Price := 500
Currency := IRPound
...
```

**The PTA sends it to the TBA**

This part will not be detailed. For more details, see the first scenario.

**The TBA makes an offer**

This part will not be detailed. For more details, see the first scenario.

**The PTA receives an answer from the TBA**

The PTA receives the offer(s) of the TBA. In this scenario, just one offer is made:

1st offer:

```
(Query-ref
 :sender TravelBrokerAgent
 :receiver PersonalTravelAgent
 :content (:tripsummary
           (:origin (:countrycode IR :city Dublin)
           :destination (:countrycode US :city New York)
```

```
            :Hops 1
            :time (:departure (:date 20000101 :hour 08.35)
             :Return (:date 20000115 :hour 09.00))
            :Company Air America
            :Class Business
            :Seat aisle
            :Networked seat Yes
            :Meal Yes
            :Type of Meal Italian
            :Smoking No
            :Price 580
            :Currency IRPound))
  :ontology fipa-PTA
  :language xxx
)
```

**The PTA evaluates the offers**

The PTA retrieves the closest case. It just ignores the new information but keep it in memory.

1st offer:

The closest case retrieved is:

```
OriginCountry := IR
OriginCity := Dublin
DestCountry := US
DestCity := New York
DepartureDate := 980504
ReturnDate := 980523
Company := Air America
Class := Business
```

```
Seat := Window
Meal := Yes
DepartureTime := 08.45
ReturnTime := 08.30
Hops := 1
Price := 500
Currency := IRPound
...
Accepted Yes
```

The case is very close to the offer, so it is proposed to the user.

**The PTA stores the offers**

When the used give his answer for an offer, the PTA completes the case and stores it in its case base. The only thing the PTA adds here is the answer of the user. It is just a new field with two value: Yes or No.

In this scenario, the user has accepted the offer. So, the PTA completes it and stores it in its case base. However, because of the new information, it also has to add some new information entities in the case retrieval net. Each new piece of information the PTA has kept in memory is added and some links are made between them and the new case.
For the next case, the new information entities will be in the case retrieval net and will be used as all the other information entities.

We can see that, if the insertion of new information in the case retrieval net is well defined, there is no problem for the PTA in dealing with it.