UniS

University of Surrey

School of Management Studies
for the
Service Sector

Intelligent Agents for Electronic Commerce in Tourism

by

Faria Yuen-yi Ng

A thesis submitted in fulfilment of the requirements
for the Award of *Doctor of Philosophy*

Abstract

The current state of electronic commerce in tourism shows that it has become an increasingly complicated task for travellers to locate and integrate disparate information as a result of the rapid growth in the number of online travel sites. Therefore, new means of automating the searching and decision-making tasks are needed. A review of current literature shows that software agents are deemed to be highly suitable for delivering solutions to these problems. However, agents have failed to penetrate the electronic marketplace so far. An analysis of the reason for this failure has led the author to conclude that a new type of architecture is required, allowing a simple and useful first wave product to accelerate the penetration of agents. For this purpose, a proof-of-concept multi-agent prototype - Personal Travel Assistant (PTA) was developed. Firstly, user requirements were compared against what existing network and agent technologies could deliver. Then, a number of obstacles were identified that were used as guidelines to derive the prototype architecture. To overcome the main obstacles in the design, PTA used existing HTTP servers to tackle the interoperability problem and keep development costs low. A multi-agent collaborative learning strategy was designed to speed up knowledge acquisition by transferring and adapting rules encoded in the Java language.

The construction of PTA goes to prove that an open multi-agent system could be deployed in a short time by standardising a small but adaptable set of communication protocols instead of going through a complex and lengthy standardisation process. Also, PTA's structure enables fully distributed computing thus minimising the necessary changes in existing hardware and software infrastructure. The major contribution of PTA to this research area is that its architecture is unique. It is hoped that it will lay the first step on the roadmap that would lead the evolution of agents into the next stage of development.

Table of Contents

CHAPTER THREE

List of Tables

List of Tables

List of Figures

List of Abbreviations

ABTA	Association of British Travel Agents
ACF	Automatic Collaborative Filtering
ACL	Agent Communication Language
AI	Artificial Intelligence
AMTP	Agent Message Transfer Protocol
API	Application Program Interface
ARPA	Advanced Research Projects Agency
ASTA	American Society of Travel Agents
ATP	Agent Transfer Protocol
BDI	Belief-Desire-Intention
CBR	Case Based Reasoning
CGI	Common Gateway Interface
CD-ROM	Compact Disc-Read-Only Memory
CERN	European Centre of Particle Physics
CNP	Contract Net Protocol
DAI	Distributed Artificial Intelligence

DARPA	Defence Advanced Research Projects Agency
DSL	Digital Subscriber Line
ES	Expert Systems
GDS	Global Distribution System
GUI	Graphical User Interface
FIPA	Foundation for Intelligent Physical Agents
HCI	Human-Computer Interface
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
IATA	International Air Transport Association
IC	Integrated Circuit
IIOP	Internet Inter-ORB Protocol
ISDN	Integrated Services Digital Network
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
KSE	Knowledge Sharing Effort
MAS	Multi-Agent System

ORB	Object Request Broker
OS	Object Serialisation
PA	Personal Assistant
PC	Personal Computer
PDA	Personal Digital Assistant
PTA	Personal Travel Assistant
RMI	Remote Method Invocation
SET	Secure Electronic Transaction
SL	Semantics Language
SQL	Standard Query Language
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TIA	Travel Industry Association of America
UK	United Kingdom
URL	Uniform Resource Locator
US	United States
WTO	World Tourist Organisation
WWW	World Wide Web
XML	Extensible Markup Language

Acknowledgement

I would like to thank my supervisor, Mrs. Silvia Sussmann, for her guidance throughout the production of this thesis. She raised my passion for Artificial Intelligence when I chose Expert Systems for my Master's dissertation, and she has kept this fire burning ever since. I am grateful to her encouragement and moral support, especially at the intermediate stage of this thesis when I decided to shift the focus of my topic, and to her making the point that I did not have to put into it everything I had done so far.

I am indebted to the School for granting me a scholarship to embark on this academic pursuit. Special thanks are recorded to all the supportive staff for asking me each time we met - without a hint of doubt - when I would finally finish my PhD.

During my time at Surrey, I have had the great opportunity to meet many peers who have turned into great friends and companions. I would like to thank them for being such an extraordinary research community and for being open to my ideas.

My family has always been a place for me to recover and recharge my batteries. My parents have contributed so much to what I am today and have given me the strength and determination to bring this work to a good end.

Finally, thanks to my husband, Terence, for his love and being a constant source of inspiration.

Chapter One

Introduction

Electronic commerce in tourism is a growing area fuelled by the popularity of the Internet and the World Wide Web (WWW) (Elias, 1999; Marcussen, 1999; WTO, 1999). Driven by the prospective advantages of low cost distribution to a large body of potential customers, a lot of travel suppliers started to sell online. Travellers, on the other hand, like to have access to travel product information at their fingertips.

Over the past two years, a new kind of software application has appeared based on a synthesis of ideas from Artificial Intelligence, Human-Computer Interaction (HCI) and electronic transactions - intelligent agents that help mediate electronic commerce activities. Agents are programs to which one can delegate a task. They differ from traditional software in that they are personalised, autonomous, reactive and adaptive. These qualities make agents useful for a wide variety of information and process management tasks. It is thus not surprising that agents emerge to offer an efficient way to mediate in the information-rich and process-rich environment of the electronic travel market.

The trio combination - travel, electronic commerce, and intelligent agents - appears to present a prime opportunity to exhibit the benefits of agent technology. However, agents have so far failed to transform the platform of electronic commerce due to a number of inter-related technical and psychological barriers, and hence their slow adoption (Guttman et al., 1998; Nwana, 1998). It is therefore the primary objective of this research to offer a viable solution to realise the potential of agents in electronic commerce in tourism.

1.1 Current Status of Electronic Commerce in Tourism

The Internet and the WWW have created an opportunity to conduct business electronically because they serve as a global meeting place for consumers and suppliers. To many travellers, it is a valuable source of all sorts of travel information such as flights, hotels, car hire, restaurants, attractions, news, weather, etc. Never before has a single medium been available through which such massive amounts of information - and in such a broad range - can be gathered. Moreover, this information can be obtained conveniently and at very low cost, which too is something that seems unprecedented (WTO, 1999). The same story applies to travel suppliers[1] who want to offer information or services through the Internet. The barriers as well as the investments needed to do this, are very low (at least when compared to other media).

A few years ago, travellers would take the initiative to visit travel sites to search for suitable products. They would find these sites by casually browsing the WWW, because they were recommended by friends and relatives, or by some advertisements. However, as the number of travel Web sites began to grow at exponential rates (Jupiter Communications, 1998b), and subsequently the amount of travel information available began to strongly increase as well, many travellers found it overwhelmingly difficult to locate the right travel product as it became too time-consuming and too laborious. Travel suppliers were also struggling to reach the target segment and make their sites stand out from many others. There is a need from both travellers and travel suppliers for delegating/outsourcing their electronic commerce activities. Though current technologies such as search engines and push technologies, which will be referred to later on, are valuable services at this moment, they have their disadvantages which are getting more and more apparent with time.

[1] The term 'supplier', throughout the rest of this thesis, means companies that sell travel products and services. However, in chapter six, it is restricted to mean 'producers' only.

From the early days of the Internet up until now, search engines such as Excite, Altavista and Yahoo were very popular means of finding information (CommerceNet/Nielsen Media Research, 1999). Search engines use small programs, called crawlers or spiders to automatically classify and index the content of Web sites. The meta-information[2] collected by these programs is then put into a large database on which keyword search can be placed to localise relevant information. While this method makes it possible to index thousands of Web pages a day, there is a price that has to be paid for using it - the loss of detail and the lack of a comprehensive summary of a page's content. This leads to search engines returning huge result lists as the answer to a query, lists which also contain a lot of 'noise', such as irrelevant, duplicate or outdated links. When thousands of links ('hits') match a query, the user of a search engine often has no choice but to sift through the retrieved entries one by one.

Moreover, the short, necessarily vague queries that most Internet search services encourage with their cramped entry forms exacerbate this problem.

> *'One way to help users describe what they want more precisely is to let them use logical operators such as AND, OR and NOT to specify which words must (or must not) be present in retrieved pages. But many users find such Boolean notation intimidating, confusing or simply unhelpful. And even experts' queries are only as good as the terms they choose.' (Hearst, 1997)*

To travel suppliers, one of their biggest problems is how they can get their sites known to the target segment. With the growing number of travel Web sites, it is crucial to stand out from the rest and target the right travellers. Yet, making their sites

[2] Meta-information include data such as the author of the document, the date of creation, the URL (Uniform Resource Locator), the document type, and some keywords that best describe the Web page's content.

known to potential travellers by submitting them (an 'advertisement') to search engines is a method that is getting less and less effective.

Push channels in the form of Push Technology are information channels on specific topics. Users can indicate which topics are of interest to them and receive the information about these topics as it becomes available on the Internet. They are one of the more recent outlets for travel suppliers to reach the target segment. They also promise to offer the travellers a strong alternative to search engines. Travel information will only be sent to travellers that are genuinely interested in it (e.g., best holiday deals in Hawaii via e-mail notification). Push Technology is built into many travel Web sites in the form of subscriptions to member clubs, newsletters, e-mail notification, etc. Travel suppliers use it to keep their customers in touch with what has changed on, or what has been added to their sites, such as Microsoft Expedia's Faretracker[3] and American Airlines' 'Net SAAver'[4]. Yet, it has not been able to become the dominant way of getting and offering travel information online.

1.2 How Intelligent Agents Fit in

Numerous 'agent' applications in electronic commerce have been launched onto the market. However, the functionality they offer should better be described as 'agent-like' instead, which is a pity since the idea of agents really is a concept with much potential,

> *'Agents, it seems, have popped up overnight in all sorts of applications, leaving some of the most savvy users fuzzy about what they actually are. This confusion, as you might expect, has resulted in some vendors' using the word "agent" to describe programs that don't even come close to the true definition of an intelligent agent. ... This fraudulent activity has given agents a bad*

[3] Faretracker alerts travellers, who specify a specific route, by e-mail on changes of airfares.

[4] Net SAAver e-mails travellers, who have registered at American Airlines' Web site, a list of about 20 discounted fares on domestic routes every week.

> *name, for many potential users now scoff at the concept and view the technology merely as marketing hype. But if you look harder, even beyond the agent implementation available on the PC today, you'll discover that true agent technology is exciting, feasible, and desperately struggling to find its way into the mainstream as an enabling technology.' (Plain, 1997)*

Stand-alone agent applications are being released in abundance, e.g., Jango (1996) and Firefly (1999)[5]. Most of these are 'searchbots' and 'shopbots' that are geared to searching and filtering product information on the World Wide Web based on a group of keywords or concepts. However, current electronic commerce agent systems are mostly ad hoc applications with limited circulation because they are designed with custom methodology and infrastructure to solve limited well-defined problems (Guttman et al., 1998). To be fair, what they have contributed so far is making people familiar with agents and with the tasks they are capable of doing, which is an important first step towards increased usage and acceptance of agents by a broad range of users in the near future.

Many (FIPA, 1997; Guttman et al., 1998) agree that agent applications in electronic commerce will become a necessity to be able to cope with the enormous amounts of commercial offerings available through the Internet. The question does no longer seem to be if there will be a considerable usage of agents, but rather *when* and *how* this will happen.

Travel - one of the most complex real-world transactions - represents a highly suitable domain to show the advantages of agents in electronic commerce (FIPA, 1997; WTO, 1999). As a multi-vendor industry and a multi-component product, a decision to purchase a tourism product is a highly complex decision-making process which involves multi-attribute constraints. For example, booking a flight ticket cannot merely depend on, say lowest fare, because fare is only one of the many attributes. A

[5] Jango is a shopping agent of Excite, Inc. and Firefly is a movie and music recommendation system of Microsoft. These agent systems will be discussed in detail in chapter five.

transaction cannot be based on or completed only for flight arrangements, because hotel, car, and many personal arrangements must also be satisfied. Planning a multi-faceted trip, therefore, requires a high degree of effort, knowledge and intelligence from the travellers.

As search engines and push technology failed to deliver what the travellers wanted, intelligent agents should move onto the scene to carry things to the next stage of evolution. The author identified the importance of personal agents in electronic commerce in tourism in some of her early work, and research has been focused on Personal Travel Assistants (PTA) (Ng, 1995b; Ng & Sussmann, 1996). The Foundation for Intelligent Physical Agents (FIPA)[6] (1997) then recognised travel's high relevancy and used PTA as one of its four application domains for field tests of agent technologies. Later the author extended the research on PTA to multi-agent systems (Ng & Sussmann, 1998). This shift was necessitated by the need to solve the inherently distributed problem-solving in tourism. At present, the information necessary to comprehensively plan a trip using different services tends to be distributed across the Internet on the Web sites of individual travel companies. In order for a personal agent to perform commercial negotiations and transactions, it is essential to communicate with representatives of various suppliers, ideally in the form of agents, to integrate effectively these services to provide a complete travel package. By having individual agents representing all players - travellers, suppliers, brokers, etc. - in a transaction, it is easier to arrive at a situation where the interests of all parties will be well looked after.

It is expected that three major types of agent will emerge in the future online travel market:

- **User agents** (for travellers),
- **Supplier agents** (e.g., airlines, hotels, car hire companies, etc), and

[6] The Foundation for Intelligent Physical Agents (FIPA) is a non-profit standardisation organisation registered in Geneva, Switzerland in 1996.

- **Middle agents** (e.g., travel agencies, global distribution systems, electronic brokers[7], etc.)

As a user agent, it can 'go shopping' on behalf of the travellers. It personalises the HCI and at the same time formulates effective retrieval strategies. It searches for travel information on the Internet to meet the needs of the traveller by selecting products with reference to the traveller's preferences and returning with recommendations of purchases that meet those specifications. More importantly, each individual user agent can work collaboratively with supplier and middle agents in a multi-agent system to perform efficient electronic business negotiations and transactions. Such a system demonstrates the usefulness of agent technologies for both the hard requirements of travel such as flight, hotel, and car arrangements as well as the soft added-value services according to personal profiles, e.g., interests in sports, theatre, or other attractions and events.

These agents will enable travellers to focus more on what they want to do (e.g., which information they need, which task they would like to get done), and much less on how they should accomplish this (e.g., where to look for information). This focus shift is necessary because it saves time and makes life a lot easier. Agents will play serious roles in enhancing the whole range of electronic commerce activities by making them more personal so as to fit the personal needs and preferences of each individual. They will make the information flows in the distribution chain work more smoothly and in a user-friendly manner.

[7] An electronic broker is the outcome of the idea of information brokering (Etzioni & Weld, 1995). Passive brokers simply act as matchmakers by matching an information query from a user agent to appropriate supplier agent(s) in its inventory. Sophisticated brokers send out the query to the appropriate sources and collect the results of each individual source. Before sending these results to the consumer, they will enhance the results, e.g., ranking them, sorting out double entries, etc. In case they fail to obtain an answer to the query, they can delegate this task to a third party (e.g., specialised agents).

1.3 The Way Forward - An Agent Web?

The ideal scenario of an electronic travel market is a fully-fledged multi-agent system - an Agent Web. To achieve this stage of sophistication, however, some fundamental issues will need to be settled.

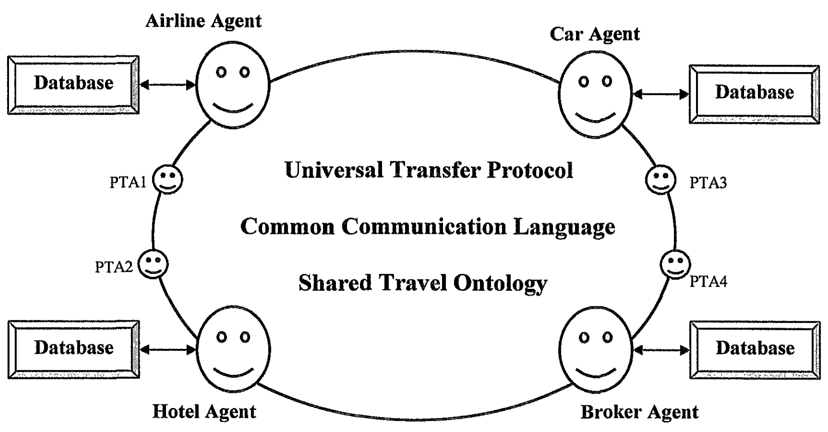


Figure 1.1 The Scenario of an Agent Web

1.3.1 Obstacles

Firstly, heterogeneous systems from various suppliers need to be interoperable. Research effort has arrived at the standardisation of agent communication languages (ACLs) such as KQML (Knowledge Query and Manipulation Language) (DARPA, 1993) and FIPA ACL (FIPA, 1999). In the context of electronic commerce, where agents have to travel, and work remotely on 'foreign' servers, using ACL is the minimum. However, there is no real-world application implemented by these ACLs so far.

Further to this, there is still a great deal of work to be done on ontologies, not just from the travel industry, but other related domains such as banking and geography. An ontology is needed to serve as a medium of common understanding among collaborating agents, i.e., the same term or vocabularies should mean the same concepts or ideas for the same context. Technically, there are already languages to build ontologies, e.g., Ontolingua (Farquhar et al., 1996). However, there is not a well-known ontology built on travelling. Worse still, coming into one is not at all easy. A travel ontology does not exist by itself. Separation and cross-references to other ontologies such as Aviation, Banking, Geography, Entertainment, Tourism, etc. is needed. FIPA (1997) attempted to define a limited travel ontology like origin, destination, budget, preferences, etc. Open Travel Alliance[8] (McNulty, 1999) is a recent effort to arrive at a common dictionary for the travel industry. CommerceNet[9] (1998) and member organisations are working towards common ontologies for electronic commerce. However, it is still an open question how terms should be universally defined and who should manage their evolution.

Secondly, the enabling infrastructure for an Agent Web will call for a drastic change to the current set-up of the Internet community. An Agent Message Transfer Protocol (AMTP), just as the Hypertext Transfer Protocol (HTTP) used for the WWW, is essential for basic agent interactions such as message exchanges. An additional, but optional, Agent Transfer Protocol (ATP) needs to be established and well in place if full agent mobility such as agent hopping and migration is to be achieved. To reach this level, it would be necessary for all existing HTTP servers to be upgraded to support agent servers, and a change in such a global scale is difficult to imagine.

[8] Companies currently participating in the Open Travel Alliance are: Alaska Airlines, American Airlines, Continental Airlines, Delta Air Lines, Midwest Express, Northwest Airlines, Trans World Airlines, United Airlines, Vanguard Airlines, Bass Hotels and Resorts, Hilton Hotels, Hyatt Corporation, Marriott International, Sterling Hotels, Swisshotel, Alamo Rent A Car, Avis Rent A Car, Budget Rent -A-Car, Dollar Rent A Car Systems, Inc, The Hertz Corporation, National Car Rental, and Thrifty Car Rental.

[9] CommerceNet is a non-profit organisation founded in 1994 to foster the growth of Internet commerce. It formed an alliance with Ontology.org in 1999 to resolve the interoperability issue.

1.3.2 Prediction

It is likely that such a revolutionary approach would encounter a lot of resistance from small and large suppliers alike, as it is both human and organisational nature to withdraw and protect oneself during times of transformational change (Ndumu et al., 1998; Nwana & Ndumu, 1999). Apart from the financial investment in training, expertise, new software, etc., there are additional psychological barriers to adapt to new methods of operations.

What is predicted here is that there will be an evolutionary transformation which makes people feel at ease, and at the same time create a migration path to change the old for new. The most probable evolution will be that agents, initially, leverage simpler technologies available without high overhead costs to attract users. After the early adoption stage is reached, agents will gradually evolve into more complicated applications. If this technology is to become widespread in tourism, next-generation software will have to be easy to use and able to blend easily with the current infrastructure to keep development costs down.

While looking back at the developments of agent technologies in the last two years, there was much evidence to support this argument. No matter how attractive and capable the technology proves itself, users and suppliers need some strong reasons to motivate them to adopt this new technology. Commercial products like IBM's Aglets[10] (1996) have been around for more than two years, but the usage rate is surprisingly low[11]. The problem does not lie in the capability of the product itself, but more due to the reasons discussed above. For example, automating straightforward tasks such as searching for the cheapest flight tickets will not justify a big commitment to change the whole infrastructure.

[10] Aglets is a mobile agent platform implemented by IBM. See section 'Mobile Agent Technologies' in chapter three for details.

[11] So far, there is only one real-world application - Tabican - a Japanese travel site at http://www.tabican.ne.jp.

The historical development pattern of the Internet also showed that 'free-trial' products are most willingly picked up by the users. To prompt the users to try out a new technology or software, some cheap or free first-wave products must be readily available for their use, just as the case of Java. If the software is easy to use, users do not need to spend too much time and effort to learn. These early users will build up the demand for agent technology. When the body of agent users reaches a critical mass, and when users favour sites with agents more than those without, suppliers/brokers will automatically be driven to adopt the technology in order to gain competitive edge. Since the software comes at a reasonable price, initial investment is low and the time needed to build up this critical mass is minimal. This will induce a growth cycle where demand of users and suppliers are reinforcing each other at each iterative step.

1.4 Structure of the Thesis

To meet the primary objective of offering a viable solution to research the potential of agents in electronic commerce in tourism, the thesis is set out in four main stages.

Aims and objectives

Chapter two outlines the objectives and methodology of the research.

Literature review

The next two chapters provide a review of the literature.

Chapter three elaborates the theoretical concept of intelligent agents and suggests some future directions in the development of future-generation agent systems.

Chapter four surveys the growth of the WWW and demonstrates its commercial potential for the electronic travel market.

Prototype

Chapter five explains the roles of intelligent agents in the future electronic travel market, discusses the advantages of a collaborative learning multi-agent system over other alternative approaches, identifies the problematic issues and sets the design philosophies for the prototype.

Chapter six presents a multi-agent prototype – Personal Travel Assistant (PTA) System - in which agents interact and learn. The usefulness and limitations of the prototype are evaluated. This represents the contribution of the thesis to knowledge and it is through this that the primary objective is addressed.

Chapter seven looks at how PTA would shape the online travel market in the short and long term and how different players in the travel industry could adapt.

Conclusion

Chapter eight concludes with suggestions on possible future work.

1.5 References

Aglets, (1996), http://www.trl.ibm.co.jp/aglets.

CommerceNet, (1998), *Success of XML for E-Commerce Accelerated by Advanced Knowledge Representation Techniques,* Palo Alto, CA, October 9, 1998, http://www.commercenet.com.

CommerceNet/Nielsen Media Research, (1999), *Top Web Site Properties and Advertisers for February 1999,* March 22, 1999, http://www.commercenet.com.

DARPA, (1993), *Specification of KQML Agent Communication Language,* Technical Report, ARPA Knowledge Sharing Initiative, External Interfaces Working Group.

Elias, E., (1999), *Internet Commerce Transforming the Travel Industry,* SRI Consulting, CommerceNet Research Report #99-34.

Etzioni, O. and Weld, D. S., (1995), Intelligent Agents on the Internet: Fact, Fiction, and Forecast, *IEEE Expert,* August, pp. 44-49.

Farquhar, A., Fikes, R. and Rice, J., (1996), *The Ontolingua Server: A Tool for Collaborative Ontology Construction,* Technical Report KSL-96-26, Stanford Knowledge Systems Laboratory, Stanford, CA.

Gruber, T. R., (1993), A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition,* Volume 2, pp. 199-220.

Guttman, R. H., Moukas, A. G. and Maes, P., (1998), Agent-Mediated Electronic Commerce: A Survey, *Knowledge Engineering Review,* 13(3), June 1998.

FIPA, (1997), *Specification, Part 4.*

FIPA, (1999), *Specification, Part 2.*

Firefly, (1999), http://www.firefly.com.

Hearst, M. A., (1997), Interfaces for Searching the Web, *Scientific American*, March, 1997, http://www.sciam.com.

Jango, (1996), http://www.jango.com.

Jupiter Communications, (1998b), *Online Travel – Five Years Outlook,* 1998, http://www.jup.com.

Marcussen, C. H., (1999), *Internet Distribution of European Travel and Tourism Services: The Market, Transportation, Accommodation and Package Tours,* The Research Centre of Bornholm, Denmark, ISBN 87-90881-28-1.

McNulty, M. A., (1999), Suppliers Seek XML Standard, *Business Travel News,* May 3, 1999.

Nwana, H. S., (1998), Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints, *Proceedings of Agents '98,* Minneapolis, May 1998, pp. 189-196.

Ng, F. Y. Y., (1995b), Intelligent Agents as Personal Travel Assistants, in Nuryanti, W. (ed.) *Tourism and Culture: Global Civilisation in Change, 1995 Indonesian-Swiss Forum on Culture and International Tourism,* Yogyakarta, Indonesia, 23-26 August, 1995, pp. 360-371.

Ng, F. Y. Y. and Sussmann, S., (1996), A Personal Travel Assistant for Holiday Selection - A Learning Interface Agent Approach, in Klein, S. et al. (eds.) *Information and Communications Technologies in Tourism, ENTER 96 International Conference,* Innsbruck, Austria, 17-19 January, 1996, Springer-Verlag, Wien, New York, pp. 1-10.

Ng, F. Y. Y. and Sussmann, S., (1998), Intelligent Agents in Electronic Travel Markets, *Proceedings of New Zealand Tourism and Hospitality Research Conference, Third Biennial Conference, 'Advances in Research',* Akaroa, New Zealand, 1-4 December, 1998, PART 2, Day Four, Group 2.

Open Travel Alliance, (1999), http://www.opentravel.com.

Plain, S. W., (1997), KQML at Your Service, *Computer Shopper,* March, 1997.

World Tourism Organisation, (1999), *Marketing Tourism Destinations Online: Strategies for the Information Age,* ISBN: 92-844-0328-6.

Chapter Two

Research Objective and Methodology

2.1 Research Objective

The primary objective of the research is to build proof-of-concept agent prototypes for use in a new electronic commerce architecture for tourism. The new architecture will bridge the gap between ultimate user requirements and the limitations of current technologies. The agent prototypes are to provide a model on how next generation agents should be applied in tourism and what forms these agents should take in order to increase penetration and gain wide adoption. The ultimate goal is to create a first step in the direction of a successful evolution path of agents.

2.2 Methodology

The methodology employed in the research has been dictated by the research objective to build an agent prototype. Based on the literature, this has involved a process establishing requirements, possibilities and delineating obstacles from which it has been possible to develop the prototype. At each stage, the development of the prototype is informed by and brought back to the requirements, possibilities and obstacles.

The designs of the agent prototypes were systematically derived as shown in figure 2.1. The detailed steps are elaborated in the following sections.

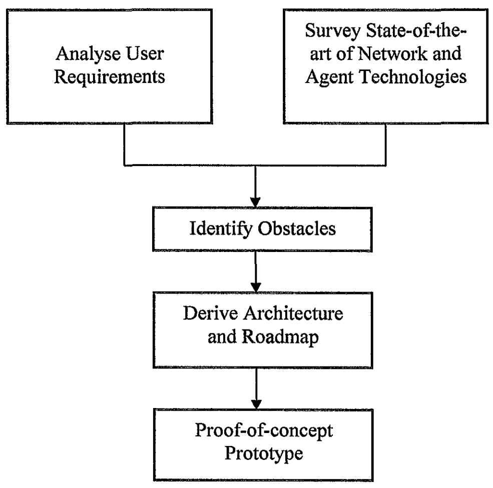

Analyse User Requirements
Survey State-of-the-art of Network and Agent Technologies
Identify Obstacles
Derive Architecture and Roadmap
Proof-of-concept Prototype

Figure 2.1 Research Methodology

2.2.1 User Analysis

User requirements in electronic commerce were derived from several sources. Consumer behaviour has been thoroughly analysed in the traditional literature (e.g., Moutinho, 1987; Middleton, 1988; Mansfeld, 1992). Well-established consumer models, being human nature, are readily applicable to the new context of electronic commerce in tourism. More recent literature (e.g., Guttman et al., 1998; Moukas et al., 1998; Nwana et al., 1998) addresses directly the user requirements in the age of the World Wide Web.

Feedback from users was collected from tourism market surveys (e.g., Jupiter Communications, 1998b; PhoCusWright, 1998b; Travel Industry Association of America, 1999). The statistical data provide valuable insight into user requirements. For example, a major reason given by users for not buying online indicates that users need to feel secure before giving away credit card numbers and other personal data (PhoCusWright, 1998b).

Numerous online travel sites were visited to gain in-depth understanding of user requirements through practical experience. The strengths and weaknesses of the visited sites help to derive and validate requirements.

2.2.2 Technology Survey

State-of-the-art technologies were surveyed to see how far these technologies can satisfy user demands. For example, one of the main user requirements is the accurate location of suppliers and products. The tools developed for these purposes are search engines and push technologies. Their performances were discussed in the literature and evaluations were also performed via practical experience. Search engines (e.g., Yahoo, Excite, Alta Vista) were tested to see if the search results are sufficient and precise enough to be useful to the customers. Push technologies in many travel web sites (e.g., Travelocity, Expedia) were also tested for their usefulness.

An in-depth survey on agent technology, which is generally regarded as the next stage in electronic commerce, was conducted by consulting a vast body of traditional and online literature. Existing web sites with agent technologies (e.g., Jango, 1996; Firefly, 1999) were tested to see if the online services offered are satisfying the user requirements mentioned earlier.

While the general goals and characteristics of agents are understood, agents can take many forms using many different underlying technologies. Current trends in hardware and software technologies, especially those used in the Internet, were surveyed to identify the likely candidates with which agents can be built.

2.2.3 Obstacle Identification

A comparison was then made between the findings from the user analysis and the technology surveys to identify areas of mismatch. Many technologies have been proposed to address some of these areas. For example, an increasing number of sophisticated visual and audio tools have been launched to help build intelligent interfaces. Powerful encryption algorithms and network security models have been introduced to increase the confidence of users. This research concentrates on the remaining obstacles that are not yet addressed.

Obstacles were also reviewed by tracing the history of development of the Internet and electronic commerce, which was part of the results from the technology survey. For example, users would not pay a premium for agent software and they would not easily commit to a particular technology unless the future popularity of the technology is evident. In addition, the co-operative development of an interoperable standard has not always been a success. These observations explained the low adoption rate of existing agent technologies.

2.2.4 Architecture and Roadmap

While there is consensus on the ultimate goal of agent technology, issues like how to achieve this goal and what the final form should take remain unresolved. A new architecture was developed to overcome the main obstacles identified above. It was identified that the success of the new agent architecture would have to depend on two criteria. The new architecture must be deployable in the immediate future, and it must be capable of evolving into the ultimate world of agents. To this end, the new architecture was developed together with the milestones of evolution in mind.

2.2.5 Prototype Design

Agent prototypes were designed to overcome the list of obstacles identified. To satisfy the architectural constraints, these agents must be capable of performing commercial transactions on the Internet using current hardware and software technologies. Once deployed, these agents must demonstrate high adaptability such that new knowledge, standards and protocols can be picked up easily with built-in mechanisms.

2.3 References

Expedia, http://www.expedia.com.

Firefly, (1999), http://www.firefly.com.

Guttman, R. H., Moukas, A. G. and Maes, P., (1998), Agent-Mediated Electronic Commerce: A Survey, *Knowledge Engineering Review,* 13(3), June 1998.

Jango, (1996), http://www.jango.com.

Jupiter Communications, (1998b), *Online Travel – Five Years Outlook,* 1998, http://www.jup.com.

Mansfeld, Y., (1992), From Motivation to Actual Travel, *Annals of Tourism Research,* 19(3), pp. 399-419.

Middleton, V. T. C., (1988), *Marketing in Travel and Tourism,* Butterworth-Heinemann, Oxford.

Moukas, A., Guttman, R. and Maes, P. (1998), Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective, *Software Agents Group, MIT Media Laboratory,* Cambridge, MA, http://ecommerce.media.mit.edu.

Moutinho, L., (1987), Consumer Behaviour in Tourism, *European Journal of Marketing,* 21(10), pp. 1-44.

Nwana, H. S., (1998), Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints, *Proceedings of Agents '98,* Minneapolis, May 1998, pp. 189-196.

PhoCusWright, (1998b), *Travel E-commerce Survey,* November, 1998, http://www.phocuswright.com.

Travel Industry Association of America, (1999), *1998 Technology and Travel Survey,* January 1999, http://www.tia.org.

Travelocity, http://www.travelocity.com.

Chapter Three

What are Intelligent Agents?

The idea of 'agent' has existed for decades. Early researchers like Minsky (1963) and Negroponte (1970) have studied problems that demonstrate some type of agent behaviour. Recent studies germinate a rich set of emerging views and draw the attention and interests of a broad spectrum of research areas such as software engineering, robotics, knowledge representation, natural language processing, machine learning, etc. Regardless of its Artificial Intelligence (AI) affiliation, agents are perceived to be able to help all types of end-users in all kinds of decision-making in everyday life situations - ranging from comparatively small systems such as personalised e-mail filters to large, complex, mission critical systems like air-traffic control. It is the naturalness and ease with which such a variety of applications can be characterised in terms of agents that leads researchers and developers to be so excited about the potential of this technology.

Given this degree of interest and level of activity, in what is a comparatively new and multi-disciplinary subject, it is not surprising that the field of agent-based computing can appear somewhat chaotic and incoherent. This chapter attempts to impose some order on the notion of 'intelligent agents'. It then proceeds to discuss the important capabilities that are expected in agent systems. An analysis of future challenges for agent researchers and developers concludes the chapter.

3.1 The History of Agent Theory

The scientific study of agent behaviour and design predates AI, going back to the early days of Cybernetics in the 1940's (Wiener, 1948). As an interdisciplinary subject, Cybernetics generated theoretical attempts to define the behaviour and structure of abstract machines that had properties corresponding to biological, cognitive systems. From the early 1960's, the dominant intellectual force in agent theory was to be found in AI, and allied fields such as the philosophy of mind (Minsky, 1963) and natural language semantics. The development of software agents in the 1970s' included the philosophical works on concepts of artificial agents of Dennet's 'intentional systems' (Dennet, 1978), and of McCarthy's mentalistic conceptualisation of machines (McCarthy & Hayes, 1969).

The idea of employing agents to delegate computer-based tasks goes back to research by Negroponte (1970) and Kay (1984). They had in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice when it was stuck. Though agent research had been going on for more than fifteen years by now, agents really became popular within the artificial intelligence and computing communities around 1994. During this year, several key agent-related publications appeared, e.g., the first of now several special issues of the Communications of the ACM on agents appeared in 1994. Indeed, during late 1994 and throughout 1995 and 1996, there was an explosion of agent-related articles in the popular computing press. The field has clearly matured since the publication of certain key papers and books including Wooldridge and Jennings (1995), Nwana (1996), Bradshaw (1997) amongst many others. It is no surprise that this explosion coincided with that of the World Wide Web.

Several annual and biennial conferences are now held in the area including the International Conference on Multi-Agent Systems (ICMAS), the International Conference on the Practical Application of Intelligent Agents and Multi-Agent

Technology (PAMM), and the International Conference on Autonomous Agents (AA). These conferences and numerous other agent-related national and international workshops, many 'agent' special issues of journals, agent books, and agent standardisation initiatives such as FIPA all bear testimony to a quickly maturing field.

3.2 Defining 'Intelligent Agents'

Until now, researchers have not yet agreed on a consensus definition for the word 'agent'. There are at least two reasons why it is so difficult to define precisely what agents are. Firstly, agent researchers do not own this term in the same way as AI researchers own the term, for example, fuzzy logic. Rather, the word 'agent' is used widely in everyday life as in travel agents, estate agents, etc. Secondly, even within the software discipline, agent is really an umbrella term for a heterogeneous body of research and development.

Different research areas reuse the term to mean very different things. In the world of computer science, various names are associated with these intelligent agents, such as knowbots/softbots (Etzioni & Weld, 1994; Kautz et al., 1994), personal assistants (Mitchell, et al., 1994), over-the-shoulder coaches (Selker, 1994), etc. To be fair, there are some good reasons for having such synonyms. Firstly, agents come in many different types. For example, agents inhabiting software environments (consisting of computers and networks) are often called softbots. Agents that help users to perform specific tasks are sometimes called expert assistants. Synthetic agents operate in a simulated environment such as computer-animated environments. Secondly, agents can play many different roles in their environments. For example, softbots may work as personal assistants running on computers of individual users. Expert assistants may work in medical monitoring, industrial control, business process management, or computer integrated manufacturing. Synthetic agents, which emphasise character qualities like believability and personality, rather than deep intelligence or expertise, may play roles in interactive systems for entertainment, art, games, education or consumer software.

Given the multiplicity of roles agents can play, this is quite impossible and even very impractical to come to a rock-solid formal definition of the concept 'intelligent agent' without reference to context. As Russell and Norvig (1995) remark, 'The notion of an agent is meant to be a tool for analysing systems, not an absolute characterisation that divides the world into agents and non-agents'. The only concepts that yield sharp edge categories are mathematical concepts, and they succeed only because they are content-free. Agents 'live' in the real world, and real world concepts yield fuzzy categories (Franklin & Graesser, 1996).

Therefore, instead of a *formal* definition, a list of characteristics of agents will be given below. When put together, these characteristics give an overall impression of what an agent is.

3.3 Agent Characteristics

Indeed, there are various characteristics of agent systems that differentiate them from ordinary programs. The first group of characteristics is 'basic' and is core to an agent. The fact that an agent should possess these characteristics is something that most researchers have agreed upon at this moment. An agent may possess additional optional qualities other than these fundamental characteristics.

3.3.1 'Basic' Agent Qualities

Wooldridge and Jennings (1995) specify four main attributes that determine agenthood:

- ***Autonomy***: An agent takes initiative and exercises control over its own actions. It can pursue its own agenda independently, thus reducing human workload by generally only interacting with its user when it is time to deliver results. This requires aspects of periodic action, spontaneous execution, and initiative, in that the agent must be able to take pre-emptive or independent actions that will benefit the user in terms of work volume and speed.

- ***Social ability***: An agent has high-level human-like communication skills. It can engage in complex communication with other agents, including people, to obtain information or enlist help in accomplishing its goals.
- ***Proactivity***: Unlike standard programs directly invoked by the user, agents do not simply act in response to user-initiated commands. They are able to exhibit goal-directed behaviour by taking the initiative.
- ***Reactivity:*** Agents perceive their environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it.

For some researchers[1] (Mitchell et al., 1994; Nwana, 1996) (and for the purpose of this thesis), an agent's ability to learn and improve its performance overtime is crucial to demonstrate its intelligent behaviour.

- ***Learning Ability***: An agent is designed to assist users to handle some tasks more efficiently. As users have different tasks, and even those who share the same task may do it in different ways, an agent must be able to learn the task at hand and adjust itself to the habits, working methods and preferences of its user.

3.3.2 'Optional' Agent Qualities

For some researchers, the term agent has other more specific characteristics than those sketched out in the previous section. This is largely due to the different research projects that these researchers are involved in and the examples that they have in mind. For example, researchers (Foner, 1997) who work on interface agents to assist

[1] The author also considers 'learning' a fundamental quality of an 'intelligent' agent. This is also the core concept of the prototype.

users accomplishing some tasks may emphasise on user-agent collaboration, trustworthiness, personalisation, etc. Other researchers (Bates, 1994; Shoham, 1993) perceive an agent to be a computer system that is either conceptualised or implemented using concepts that are more usually applied to humans. For example, it is quite common in AI to characterise an agent using mentalistic notions, such as belief, desire, intention, etc. Mobile agent researchers characterise an agent as those that can move around an electronic network (White, 1995), transport itself from one machine to another and across different system architectures and platforms.

At this moment, no consensus has yet been reached about the relative importance (weight) of each of these characteristics in the agent as a whole. Naturally, some agents will have additional characteristics, and for certain types of applications, some attributes will be more important than others. However, it is the presence of all the attributes in a single software entity that provides the power of the agent paradigm and distinguishes agent systems from related software paradigms such as object-oriented systems, distributed systems, and expert systems (Jennings et al., 1998).

3.3.3 'Agency' and 'Intelligence'

The degree of autonomy and authority vested in the agent is called its *agency*. It can be measured at least qualitatively by the nature of the interaction between the agent and other entities in the system in which it operates. At a minimum, an agent must run asynchronously. The degree of agency is enhanced if an agent represents a user in some way. This is one of the key values of agents. A more advanced agent can interact with other entities such as data, applications, or services. Further advanced agents collaborate and negotiate with other agents.

What exactly makes an agent 'intelligent' is something that is hard to define. It has been the subject of many discussions in the field of AI, and a clear answer has yet to be found.

Yet, Gilbert et al. (1995) gave a workable definition of what makes an agent intelligent,

> *'Intelligence is the degree of reasoning and learned behaviour: the agent's ability to accept the user's statement of goals and carry out the task delegated to it.*
>
> *At a minimum, there can be some statement of preferences, perhaps in the form of rules, with an inference engine or some other reasoning mechanism to act on these preferences.*
>
> *Higher levels of intelligence include a user model or some other form of understanding and reasoning about what a user wants done, and planning the means to achieve this goal.*
>
> *Further out on the intelligence scale are systems that learn and adapt to their environment, both in terms of the user's objectives, and in terms of the resources available to the agent. Such a system might, like a human assistant, discover new relationships, connections, or concepts independently from the human user, and exploit these in anticipating and satisfying user needs.'*

The minimum level is somewhat *pseudo-intelligent*, like knowledge-based or expert systems, which make decisions based on facts and rules that reside in a knowledge base handcrafted by humans. These systems are knowledgeable, but far from intelligent in the strict sense. They do not have the flexibility to adapt to dynamic situations because rules in these systems are pre-determined and fixed. The middle level may include systems that are *barely intelligent.* Their ability to translate information into knowledge and build up their own knowledge base makes them stand out from the last group. However, if their knowledge is 'frozen' once it is acquired, then this is hardly true intelligence at all. *Real intelligent* systems should possess the ability to learn and acquire new knowledge, adapt and evolve their rules to suit a dynamic environment.

3.4 Examples of Agent Systems

The table below shows the application areas (Jennings et al., 1998) where agent technology is currently used.

Domain	Application
Industrial	Manufacturing
	Process control
	Telecommunications
	Air-traffic control
	Traffic and transportation systems
Commercial	Information Management
	Electronic commerce
	Business process management
Entertainment	Games
	Interactive theatre and cinema
Medical	Patient monitoring
	Health care

Table 3.1 Intelligent Agents Application Areas

Comparing these systems is not an easy task as their possibilities and degree of elaboration vary strongly. Since information management and electronic commerce on the Internet are two of the most intensely studied areas in recent years, four examples have been selected for further illustration.

3.4.1 Information Management

These systems provide services that help users find the information they request more efficiently. This can be done, for example, by a single agent like a newsgroup agent, or a society of agents such as stock market agents.

3.4.1.1 Newsgroup Filtering

There are now research projects on filtering agents on the Internet such as e-mail and newsgroup filtering (Maes, 1994). Filtering agents are mostly pseudo-intelligent assistants that help users sort out useful information from large sources of data. For example, Sheth and Maes (1993) describe a news filtering agent, called NewT, whose role is that of helping the user filter and select articles from a continuous stream of Usenet Netnews. The idea is to let the user create one or many 'news agents' (e.g., one agent for sports news, one for financial news, etc.) and train them by examples (i.e., by presenting to them positive and negative examples of what should or should not be retrieved).

Systems of this sort suffer from a number of disadvantages. Firstly, the systems almost always base their understanding of the content of the news articles or other information sources on features or keywords. These are known to be unreliable in capturing the true meaning of natural language texts. Secondly, even if the features selected can effectively represent the pieces of information to be filtered, there is no guarantee that the users' interests are successfully deduced based upon these features.

3.4.1.2 Portfolio Management

Due to the limited function of pure filtering agents, a number of projects integrate information retrieval and filtering. Retrieval agents are more useful especially when implemented in the context of a multi-agent system. For example, the WARREN system (Sycara et al., 1996) consists of agents that co-operatively self-organise to monitor and track stock quotes, financial news, financial analysts reports, and company earnings reports in order to appraise the portfolio owner of the evolving financial picture.

Using this approach, each separate information source will be modelled by an agent and the retrieval agent will broadcast a request for information. Upon receipt of such a request, each information agent will reply with a message containing all the information to which they have access with relevance to the initial request. The retrieval agent, having received all these separate pieces of information, is responsible for aggregating the information sensibly and presenting the information in a unified and easy to access fashion. This approach has the added benefit of allowing users to deal with distributed sources of information effectively, for example, the WWW.

3.4.2 Electronic Commerce

Electronic commerce is one of the most intensely studied application areas due to the popularity of the Internet. Buyers need to find sellers of products and services, they need to find product information (including technical specifications, viable configurations, etc.) that solves their problem, and they need to obtain expert advice on their purchase. Sellers, on the other hand, need to find buyers and target the right customers. It is speculated that intelligent agents will be used widely to facilitate the solving of these problems.

3.4.2.1 MAGMA

The MAGMA system (Tsvetovatyy & Gini, 1996) involves Personal Assistants (PAs) representing buyers and sellers in a virtual marketplace and performing negotiation. The essential idea is that users tell their PAs what they would like to be done, e.g., sell an item for the best possible price, and trust them to figure out how to accomplish the task, thus freeing their time and energy for more interesting pursuits. In addition, users hope that agents might be able to sell goods better (e.g., at a higher price) than they would be able to, by taking advantage of their edge in processing speed.

MAGMA agents access a global blackboard, called an *'offer board'*, that contains offers from other buyers and sellers. The negotiation scheme involves sellers specifying to their agents, the maximum price the item should sell at and posts this data to the offer board. The buyers indicate to their agents the category of item they are interested in and the number of best offers to retrieve. Each buyer agent displays these offers and the buyer will select which offer to follow up. The buyer agent will then send an accept offer to the relevant seller agent.

3.4.2.2 Kasbah

Kasbah (Chavez & Maes, 1996) is a classified ads service on the WWW that incorporates intelligent agents. Kasbah is meant to represent a 'marketplace' (a Web site) where Kasbah agents, acting on behalf of their owners, can filter through the ads and find those that their users might be interested in. The agents then proceed to negotiate to buy and sell items.

The negotiation scheme for buyers and sellers involves *'price-raise' and 'decay function'* respectively. For sellers, the PAs will offer their items at the highest desired price, and then decrease the price over time according to the decay function (which is user-specified as being linear, quadratic or cubic). So, when the desired date to sell the item arrives, the asking price should be about the lowest acceptable price. The converse is true with buyers and their 'price-raise' functions.

3.5 Agent Architectures

There is no one-and-only methodology to construct agents. One widely-used tradition in the area of agent architectures is that of *practical reasoning* agents (Bratman, 1988). Practical reasoning agents have architectures which are modelled on or inspired by a theory of practical reasoning in humans. By practical reasoning, it means the kind of pragmatic reasoning that humans use to decide what to do. Practical reasoning has long been an area of study by philosophers, who are interested in developing theories that can account for human behaviour. Typically, theories of practical reasoning make use of a *folk psychology*, whereby behaviour is understood by the attribution of *attitudes* such as beliefs, desires, intentions, and so on. Human behaviour can be thought of as arising through the interaction of such attitudes. Practical reasoning architectures are modelled on theories of such interactions. The best-known and most influential type of practical reasoning architecture is the so-called *belief-desire-intention (BDI)* model (Georgeff & Lansky, 1987).

As the name indicates, BDI agents are characterised by a 'mental state' with three components: beliefs, desires, and intentions. Intuitively, beliefs correspond to information that the agent has about its environment. Desires represent 'options' available to the agent - different possible states of affairs that the agent may choose to commit to. Intentions represent states of affairs that the agent has chosen and has committed resources to. An agent's practical reasoning involves repeatedly updating beliefs from information in the environment, deciding what options are available, 'filtering' these options to determine new intentions, and acting on the basis of these intentions. Figure 3.1 shows the four key data structures of a BDI architecture:

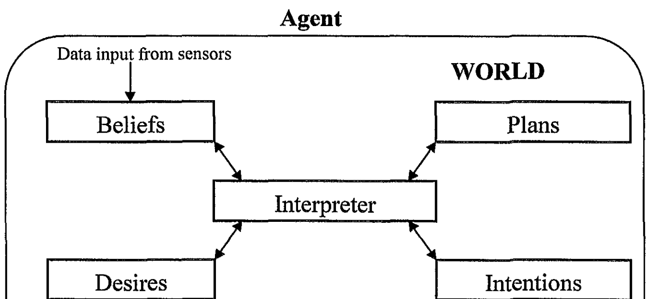


Figure 3.1 The Architecture of a BDI Agent

1. An agent's *beliefs*, correspond to information the agent has about the world, which may be incomplete or incorrect;

2. An agent's *desires* intuitively correspond to the tasks allocated to it (its goals);

3. An agent's *intentions* represent desires that it has committed to achieving. The intuition is that an agent will not, in general, be able to achieve *all* its desires, even if these desires are consistent. Agents must therefore fix upon some subset of available desires and commit resources to achieving them. These chosen desires, which the agent has committed to achieving, are *intentions*. An agent will typically continue to try and achieve an intention until either it believes the intention is satisfied, or it believes the intention is no longer achievable.

4. The final data structure in a BDI agent is a *plan library*, which is a set of plans (or recipes) which specifies courses of actions that may be followed by an agent in order to achieve its intentions.

The *interpreter* in Figure 3.1 is responsible for updating beliefs from observations made of the world, generating new desires (tasks) on the basis of new beliefs, and selecting from the set of currently active desires some subset to act as intentions. Finally, the interpreter must select an action to perform on the basis of the agent's current intentions and procedural knowledge.

A number of BDI agent systems have been implemented, the best-known of which is probably the *Procedural Reasoning System (PRS)* (Rao & Georgeff, 1995). This BDI agent framework is currently undergoing parallel evaluation trials at Sydney airport, receiving live data from the radar. Essentially, each aircraft agent estimates its Estimated Time of Arrival (intentions/ETA), based on its beliefs (e.g., current wind velocity) and desires (desired ETA) and using accessible world semantics.

3.6 A Critique on Agent Capabilities

Rather than giving a straightforward account of types of agent, or a broad classification into single and multi-agent systems, this section intends to illustrate the different capabilities of agents with a critical twist. A typology of agents will inevitably end up with a lot of confusion. There is no such delineation possible because agents usually have crossover characteristics and functions. The approach of 'single vs multi' is workable but far from ideal, as it may cause a lot of unnecessary repetition.

An agent system may contain one or more agents. The single agent perspective is based on the idea of a 'personal assistant'. The agent acts as an expert assistant to a user to carry out some tasks. These agents are usually called personal/interface agents, and they usually possess the first two capabilities discussed below – interface personalisation (in human-agent interaction) and learning. In a multi-agent system where there is more than one agent, each participating agent can be viewed as a single agent. However, due to the fact that these agents <u>must</u> participate in sophisticated inter-agent interactions, it is essential for them to have <u>additional</u> capabilities for communication, co-ordination, and negotiation. Mobility, which is neutral to single or multi-agent systems, is also included as the last capability. However, the usefulness of 'mobile agents' remains debatable, and is highly dependent on the application context.

3.6.1 Interface Personalisation

A compelling reason for users to use agents is that they will make things simpler and life easier. The interface between the user and agents is a very important factor for success. One controversial aspect of personal agents is the whole issue of *anthropomorphism*, e.g., giving agents a human-like face such as an animated cartoon icon. At one extreme, some projects create highly anthropomorphic agents which attempt to convey the whole range of human emotions (Bates,.1994). However, an agent does not necessarily have to be anthropomorphic. Having human appearance does not make a piece of software an agent if it lacks the crucial elements which define 'intelligence'.

As far as human-agent interaction is concerned, the 'intelligence' of an agent will be mainly determined by *personalisation*, i.e., to take individual preferences and characteristics of users into account and adapt its behaviour to these factors. The ability to personalise has been regarded as the predominant characteristic of personal assistants or interface agents, which appeared together within the research area of intelligent user interfaces.

The driving force of personalisation research has traditionally been the goal to improve usability, supportiveness, and effectiveness of interactive software systems. The objective is to work towards indirect manipulation (Kay, 1990). Instead of issuing direct commands to an interface, a user can delegate high level tasks to an agent without much supervision. In addition to this, recent interests in one-to-one marketing as a central means of electronic commerce have increased the demand for personalisation capabilities dramatically.

A very important issue to consider is how to determine the 'just necessary level' of user interaction. This problem requires techniques to define user preferences and understand how preferences interact with task complexity. There seems to be an issue here – that of the interplay between the nature of the task and the modelling or learning required. If the task is quite mundane, is it really worth anyone's while having a personal agent for it that requires modelling the user? On the contrary, if the task is quite complex, can a personal agent provide any really useful assistance without a deep cognitive model of the user and the task? This is an issue of a fine balance.

However, acceptability of Personal Assistants - and all other assistants - will be based on their ability to tune in to users' preferences and produce reliable results. Most obviously, information about the user upon which personalised behaviour can be based must be available and kept within an information store, which is often called a *user model*. There are two major approaches to construct user models.

The first approach is often called *user profiles* where contents are explicitly entered. User information is collected explicitly, for example, through direct questioning, which is then stored and maintained, e.g., in a database (or a knowledge base, if sophisticated representation and inference mechanisms are involved). However, there are many scenarios where information about the user is not explicitly available but must be acquired from observations of user behaviour. In these cases, complex *learning* mechanisms may be needed, which result in association-like information about a user. For Personal Assistants, making inferences based on user profiles may be sufficient to perform basic routine functions. However, the addition of end-user modelling through learning will be of increasing importance, and it seems there is a need for more research in this area.

3.6.2 Learning

As discussed earlier, an agent must possess the ability to build a user model in order to be really useful. User models may contain information such as user interests and preferences, user plans and short term intentions, user ability, etc. *Machine Learning Techniques* might be useful in training agents to build user models.

3.6.2.1 Symbolic Classifiers

Symbolic classification attempts to classify a set of examples into one of a finite number of abstract classes.

No. of Legs	Backbone	Hair	Wings	Class
4	No	No	Yes	Insect
2	Yes	Yes	No	Mammal
4	Yes	No	No	Reptile

Table 3.2 Symbolic Classifier - Sample Dataset

For example, in the dataset shown in Table 3.2, there are three examples, each in a separate class. Each element of which consists of a number of possibly relevant attributes and the class that the example should be classified into.

Given this training set of this type, the algorithm produces a decision tree by selecting at each level of the tree an attribute on which to split the remaining set based on a measure of the information content of that attribute.

3.6.2.2 Sub-Symbolic Classifiers

The sub-symbolic approach is based upon Neural Network systems. These systems take patterns as their input and classify them into one of a finite number of categories. These systems can be further split into two sub-categories based upon the architecture of the network and the method used to train them.

The first category shares many of the properties of the symbolic classifier systems such as that described above. These are termed *'Supervised Neural Networks'* and are typically trained using a set of input patterns for which the desired output category is known in advance. Some measure of the error between the actual and desired output is calculated which the training algorithm attempts to minimise as the training set is repeatedly presented.

The second class of Neural Network system requires no trainer to tell them the correct classification for the training examples and hence are termed *'Unsupervised Neural Networks'*. These networks fall into the general class of competitive learning techniques. Typically consisting of two layers, the first layer accepts input patterns, normalises this input and feeds it forward to a second layer. A node in the second layer is selected based upon the maximal activation and is put forward as the current hypothesis. Following this, a matching phase occurs, where the activated node representing the hypothesis is matched with input I, the quality of the match is assessed and if a certain threshold is not passed, Layer II is reset and input I activates a new node and the process is repeated. Thus the system can be said to develop its own classifications.

3.6.2.3 Reinforcement Learning

Another class of unsupervised learning algorithms is the reinforcement class of learning algorithms. One example of this type of approach to learning is Watkin's Q Learning algorithm (Watkins, 1989). Q Learning works by calculating an estimate for state-action pairs *Q(s,a)*, which are defined to be the expected discounted sum of taking action a in state s and pursuing an optimal policy from there on. Once these values are learnt, the correct course of action can be determined at any state by taking the action with the highest *Q(s,a)* value. These *Q* values are estimated on the basis of experience according to the *Q* Learning algorithm. This algorithm is guaranteed to converge to the correct *Q* values if the environment is static and depends on the current state and action taken in it. In addition to the limits mentioned above, this mechanism also tends to be prohibitively slow for anything but the simplest real world problems.

3.6.2.4 Learning by Observation

As noted above, in theory, a program can learn how to perform any task simply by applying reinforcement learning given sufficient time. However if there is an available 'expert' who already knows how to complete the task or even some of the sub-tasks involved, it is possible to make use of this experience and speed up the learning process by imitating the expert and thereby gaining knowledge of how to tackle the task. This learning technique is based upon the type of learning undertaken by many animates. It should be noted that this approach does not assume an active trainer, the program is expected to learn about the task simply by watching and copying experts as they display their usual behaviour (Mazur, 1994).

3.6.2.5 Memory-Based Learning

In this type of learning, an agent keeps a memory of the activities of the user, finds recurrent patterns and stores them as situation-action pairs (Stanfill & Waltz, 1986) which are simply raw data about what happened. Situations are described as a set of attributes. When a new situation occurs, the agent will try to predict the action of the user, based on the situation-action pairs stored in its memory. The agent compares the new situation with the memorised situation-action pairs and tries to find a set of close matches. The most similar of these memorised situations contributes to the decision of which action to take or suggest in the current situation.

The distance between a new situation and a memorised situation is computed as a weighted sum of the distances between the values of each. The distance between attribute-values is based on a metric computed by observing how often in the example-base the two values in that attribute correspond to the same action. The agent also maintains a set of weightings for each attribute. It helps to determine which attributes of the situation are most relevant to predicting the action. The weight given to a particular attribute depends upon the value for that attribute in the new situation, and is computed by observing how well that value has historically correlated with the action taken.

3.6.2.6 Programming by Examples

If a user does not want to wait for the agent to pick up a certain pattern, it is possible for the user to instruct the agent explicitly by creating a hypothetical example and show the agent what should be done. This technique involves adding the example to the agent's memory. The agent records the actions, tracks relationships among objects and changes the existing memory to incorporate the example that it is shown. However, the agent's competence is necessarily restricted to situations similar to those it has encountered in the past.

3.6.2.7 Multi-agent Collaboration

Collaborative agents exchange information and augment their knowledge (Werner, 1996). When faced with an unfamiliar situation, an agent consults its peers who may have the necessary experience to help it. Experienced agents can help a new agent to come up to speed quickly as well as help agents in unfamiliar situations. This type of collaboration allows agents of different users to co-operate to best aid their individual users. Agents thus have access to a much larger body of knowledge than that possessed by any individual agent. Agents learn to trust the suggestions of some of their peers more than others for various classes of situation. Each agent also learns which of its peers is a reliable 'expert' for different types of situation. Multi-agent collaboration enables an inexperienced agent to make accurate predictions with high confidence as soon as it is activated as well as fill in gaps in even an experienced agent's knowledge. This steepens the 'learning curve' and improves the handling of entirely novel situations (Lashkari et al., 1994).

3.6.2.8 Evolutionary Computation

Existing AI techniques such as evolutionary computations can be used to evolve a population of personalised agents. An evolutionary computation (Angeline, 1995) selects a subset of the population to act as parents for a new population. Selection of parents is based on the relative worth of the candidates in the population as judged by a fitness function. It then applies a set of operators, often called mutations, to copies of the selected parents that alter their content. The resulting children comprise the subsequent population. Evolutionary computations can be viewed as search in a space of genotypes for the ones that are the fittest (or the best adapted) to survive in the environment. Cycles of genetic variation followed by selection of the fittest produce a relatively fitter species with every generation (Sheth & Maes, 1993). Evolutionary methods have advantages in solving problems in which the learning agent cannot accurately sense the state of its environment.

3.6.2.9 Case Based Reasoning (CBR)

Another area of work with possible relevance to the learning of user preferences is *Case Based Reasoning (CBR).* CBR systems use similarity-based inference to infer solutions of new problems on the basis of a precedent case that is, to some extent, similar to the current problem.

For example, in a multi-user system, the period of interaction with an individual user will probably be fairly short. This is a potential problem as learning techniques tend to demand a fairly sustained period of interaction. One possible way of tackling this problem is to identify *classes of user* as fundamental units instead of considering individual users. This would require an initial classification of users into one of a number of stereotypical user groups which could then be specialised to suit the individual user based upon whatever information could be gleaned during his/her limited interaction with the system. More interestingly, each user group could be treated as a kind of virtual user with individual user's interactions forming parts of a much longer session of interaction. This way, one could achieve the kind of length of interaction necessary to adapt the group stereotypes. This kind of scheme appears to be analogous to the kind of scheme typically employed by CBR systems. In these systems, there is a set of generalised cases which are specialised to fit the individual problems. If these specified cases differ significantly from the others in the case library, they too can be generalised and added to the library.

Comments: It seems unlikely that a single learning technique will suffice for all situations. It is possible that the acquisition of user interests, for example, can be achieved in a more simplistic manner than a more complex learning problem such as inferring a user's goals and/or plans to achieve such goals. Of the available techniques, the traditional classifier techniques are unsuitable in the domain of user modelling which is basically unsupervised in nature. The unsupervised neural networks might have a role in classifying users if some sort of stereotypical system is employed. Other traditional techniques like reinforcement learning and memory-based learning are too slow. They are probably unsuitable for most real time situations apart from the most mundane tasks where the agents can capture a lot of reference examples within a short duration. Newer methods like learning by observation also take time and are only good for cases where there is a lot of repetitive behaviour. Learning by demonstration involves a great deal of work from the user. Evolutionary computation is fairly new and untested. Case-based reasoning has its use when there is a shortage of previous cases. Collaborative learning is a potential method that will lead to more fruitful research because of its naturalness and collective nature (it is also the learning method used in the prototype).

3.6.3 Multi-Agent Interactions

Traditionally, research into systems composed of multiple agents was carried out under the banner of Distributed Artificial Intelligence (DAI). DAI is a sub-field of Artificial Intelligence which has, for more than a decade now been investigating knowledge models, as well as communication and reasoning techniques that computational agents might need to participate in 'societies' composed of computers and people. More generally, DAI is concerned with a society of problem solvers or agents interacting in order to solve a common problem: computers and persons, sensors, aircraft, robots, etc. Such a society is termed a m*ulti-agent system (MAS).*

A MAS can be defined as 'a loosely coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities' (O'Hare & Jennings, 1996). The goal of multi-agent systems (MAS) is clear enough and has been proven in many multi-agent prototypes across the globe: creating a system that interconnects separately developed agents, thus enabling the ensemble to function beyond the capabilities of any singular agent in the set-up. The need for collaboration between agents occurs for a number of reasons; however, most are rooted in the problem of scarcity of resources – computing, information, know-how, etc. Since, individual agents possess different resources and capabilities, a solution to a given problem may be beyond the capabilities of any one agent, requiring that a number of agents pool their resources and collaborate with one another in order to solve the problem.

MASs can be classified into co-operative and self-interested, depending on the degree of co-operation exhibited by the individual agents. The co-operative-to-antagonistic spectrum in MASs has been surveyed by a number of researchers (Genesereth et al, 1986; Rosenschein & Genesereth, 1985).

Early work in DAI was exclusively on interacting agents that had all been designed by a single designer. Therefore, agents could be counted on to act for the greater good of the system since they could all be programmed that way by the designer, who was only concerned with increasing the general system's performance and not the performance of individual agents. Such agents are considered *co-operative.*

In the mid-1980s, a distinct research direction within DAI began to take shape. Researchers began to note that the situation of total co-operation, known as the benevolent agent assumption accepted by most DAI researchers, is not always true in the real world where agents may have conflicting goals. They began to ask questions related to individually motivated agents that had been designed by independent designers. When considering system behaviour, they could not count on agents co-operating just because they would be designed that way. What was important to each independent designer was the benefit they could derive from their individual agents. Such agents are considered *self-interested,* competitive or non-co-operative and may exhibit antagonistic behaviour.

Agents in both types of MAS have sophisticated pattern of interactions - communication, *co-ordination* and *negotiation.* In order to solve common problems coherently, the agents must communicate among themselves, co-ordinate their activities and negotiate once they find themselves in conflict. Conflicts can result from simple limited resource contention to more complex issue-based computations where the agents disagree because of discrepancies between their domains of expertise. Co-ordination is required to determine organisational structure amongst a group of agents and for task and resource allocation, while negotiation is required for the detection and resolution of conflicts.

It is the flexibility and high-level nature of these interactions which distinguishes MASs from other forms of software and which provides the underlying power of the paradigm. While communication is a rich area in its own right and will be discussed further in the next section, co-ordination and negotiation are distinctive elements in MASs.

Many co-ordination and negotiation techniques are in use. The various approaches discussed below have their relative advantages and disadvantages and hence at this time, there is no universally best method. There seem to be two extremes. The theoretical methods produce good results in well-constrained environments, but many of their underpinning assumptions are not well suited to developing real-world systems. On the other hand, the less formal techniques operate well in limited domains, but suffer from a lack of grounding and rigorous evaluation.

3.6.3.1 Co-ordination

Co-ordination has been studied by researchers in diverse disciplines in social sciences, including organisation theory, political science, social psychology, anthropology, law and sociology. For example, organisation theorists have investigated the co-ordination of systems of human beings, from small groups to large formal organisations (Thompson et al, 1991). Even biological systems appear to be co-ordinated though individual cells or 'agents' which operate independently and in a seemingly non-purposeful fashion. Human brains exhibit co-ordinated behaviour from apparently 'random' behaviours of very simple neurones.

Co-ordination is central to a MAS, for without it, any benefits of interaction vanish and the group of agents quickly degenerates into a collection of individuals with a chaotic behaviour. Essentially, co-ordination is a process in which agents engage in order to ensure a community of individual agents acting in a coherent and harmonious manner. If all the agents in the system have complete knowledge of the goals, actions, and interactions of their fellow community members and have infinite processing power, it is possible to know exactly what one agent is doing at present and what it is intending to do in the future. In such instances, it is possible to avoid conflicting and redundant efforts and the system can be perfectly co-ordinated. However such complete knowledge is simply not feasible in any community of reasonable complexity.

There are several reasons why multiple agents need to be co-ordinated:

- *Prevent chaos*: No agent possesses a global view of the entire agency to which it belongs. Consequently, agents only have local views, goals and knowledge that may interfere with rather than support other agents' actions. Co-ordination is vital to prevent chaos during conflicts.
- *Meet global constraints*: Agents performing network management may have to respond to certain failures within seconds and others within hours. Co-ordinating agents' behaviours is therefore essential to meet such global constraints.
- *Diversified capabilities and expertise*: Agents in a MAS possess different capabilities and expertise and therefore need to be co-ordinated to maximise output and efficiency.
- *Interdependent activities*: Agent's actions are frequently interdependent and hence an agent may need to wait on another agent to complete its task before executing its own.

3.6.3.1.1 Organisational Structuring - Delegation

The easiest way of ensuring coherent behaviour and resolving conflicts seems to consist of providing the group with an agent that has a wider perspective of the system, thereby exploiting an organisational or hierarchical structure. This is the simplest co-ordination technique and yields a classic *master/slave* or client/server architecture for task and resource allocation among slave agents by some master agent. The master controller could gather information from the agents of the group, create plans, and assign tasks to individual agents in order to ensure global coherence.

However, such an approach is impractical in realistic applications because it is very difficult to create such a central controller informed of all agents' intentions and beliefs. Durfee et al. (1989) point out that such centralised control as in the master/slave technique is contrary to the basic assumptions of DAI.

3.6.3.1.2 Contract Net Protocol - Contracting

The most renowned co-ordination technique for task and resource allocation among agents and determining organisational structure is the Contract-Net Protocol (CNP) (Smith & Davis, 1981).

In this approach, a decentralised market structure is assumed and agents can take on two roles, a manager or a contractor. The basic premise of this form of co-ordination is that, if an agent cannot solve an assigned problem using local resources/expertise, it will decompose the problem into sub-problems and try to find other willing agents with the necessary resources/expertise to solve these sub-problems. The problem of assigning the sub-problems is solved by a contracting mechanism consisting of contract announcement by the manager agent, submission of bids by contracting agents in response to the announcement, and the evaluation of the submitted bids by the contractor, which leads to awarding a sub-problem contract to the contractor(s) with the most appropriate bid(s).

The simplicity of the approach makes it one of the most widely used co-ordination mechanisms with many variants in the literature. Some interesting variants to the contract-net protocol include various auction protocols such as the English, Dutch and double auctions (for negotiation). Some of its advantages include *dynamic task allocation* via self-bidding which leads to better arguments, agents can be introduced and removed dynamically, it provides natural load-balancing as busy agents need not bid (Huhns & Singh, 1998). Its limitations involve the fact that it does not detect or resolve conflicts (as discussed above), the agents in the contract net are considered benevolent and non-antagonistic (which in real-world scenarios is not realistic). Moreover, it is rather communication-intensive, the costs of which may outweigh some of its advantages in real-world applications.

3.6.3.1.3 Multi-Agent Planning

More traditional AI research has led to viewing the problem of co-ordinating multiple nodes as a planning problem. Multi-agent planning emphasises certain avoidance of inconsistent and conflicting situations, which is critical in applications such as air-traffic control. By forming a multi-agent plan, the nodes determine all of their actions and interactions beforehand, leaving nothing to chance. There are two types of multi-agent planning, namely, centralised and distributed.

In *centralised* multi-agent planning, the separate agents form their individual plans and then send these plans to a central co-ordinator, who analyses them and finds potential plan conflicts (Georgeff, 1983). The idea behind this approach is that the central co-ordinator can: (1) identify critical regions of plans around which agents should synchronise and (2) insert plan steps for sending and waiting for synchronisation messages to ensure proper synchronisation. The individual partial plans can then be merged into a multi-agent plan with conflicting interactions eliminated. A disadvantage of this technique is that having the agents first form their plans as if they were acting alone and then co-ordinating these plans can miss opportunities for co-operation that would have been possible had the agents built their individual plans concurrently with reasoning about what other nodes are doing. It also shares many of the disadvantages inherent in the master/slave approach to co-ordination.

The *distributed* technique for multi-agent planning foregoes the use of a central co-ordinator, and instead allows agents to model each other's plans (Georgeff, 1984). Agents communicate in order to build and update their individual plans and their models of others' until all conflicts are removed.

Both approaches to multi-agent planning require that nodes share and process substantial amount of information and hence multi-agent planning generally involves more computation and communication than other approaches.

3.6.3.1.4 Social Laws

As mentioned earlier, if all agents had complete knowledge of each other's goals, actions and interactions, it would be possible to avoid conflicting and redundant efforts. However, the effort of achieving this state would be prohibitively high and is only feasible in routine situations. For example, in an urban traffic context, a routine situation is where the car in front signals that he is turning right, or when the traffic lights go green, so one can co-ordinate one's actions, based on knowing what others are doing according to social law (traffic laws). In real-world domains, there are also familiar and unfamiliar situations. A familiar situation is not routine. A familiar situation might occur when the traffic lights are broken and a policeman is directing the traffic. Again, based on social law, his directions will co-ordinate the traffic well. An unfamiliar situation might be where the traffic lights are down and there is no policeman directing the traffic.

Co-ordination decreases as the situation becomes less familiar, as more analysis and reasoning is required, which is laborious and could result in conflicts between agents. Therefore, agents should be given social laws, so that their processing relies on skills, which result in fast, effortless decisions and is propitious for co-ordinated activities between agents. Agent performance is governed by stored patterns of predefined procedures that map directly from observation (i.e., perception) to an action. Unfamiliar situations are adapted to familiar situations, using case-based reasoning. The agent model consists of three levels, skill (routine), rules (familiar), and knowledge (unfamiliar). The low levels, routine familiar situations are strengthened by enriching each agent with social regularities, such as, co-ordination rules (drive on left, speak in turn, etc.), co-operative rules (recycling cans or consuming energy: actions by one agent makes sense only if other agents also perform action) and social collections (e.g., roles, groups, organisations, etc.).

3.6.3.2 Negotiation

Agents in a MAS use negotiation for conflict resolution and hence co-ordination. Therefore, most co-ordination schemes involve some sort of negotiation, and hence the distinction between negotiation and the other co-ordination approaches discussed earlier may be quite fuzzy at times.

Though negotiation is highly important for the modelling of multi-agent systems, there is no clear and common definition of what negotiation is. A basic definition is given by Bussmann and Muller (1992),

> *'...negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter.'*

Agreement might be about price, military arrangements, about a meeting place or time, about joint action, or about a joint objective. The search process may involve the exchange of information, the relaxation of initial goals, mutual concessions, lies or threats. Therefore, the basic idea behind negotiation is reaching a consensus.

3.6.3.2.1 Competitive Negotiation

This is commonly used in situations where 'agents of disparate interests attempt to make a group choice over well-defined alternatives' (Rosenschein & Zlotkin, 1994). Therefore, competitive negotiation involves independent machines with independent goals that interact with each other. They are not a priori co-operative, willing to share information or back down for the greater good, namely they are competitive. Each *Competitive Agent*'s goal is to maximise its own interests, while attempting to reach agreement with other agents. Examples are buying/selling agents where negotiation to resolve conflicts is at a competitive level, namely each is trying to obtain the lowest/highest price possible for its own good and not for the good of the market community as a whole. These agents are working on behalf of an individual user and not as part of a unified community.

3.6.3.2.1.1 Game Theory

There is now a growing body of work on negotiation that is based on the traditional game theory (Luce & Raiffa, 1957). The key researchers in this area are Rosenschein and Zlotkin (1994), who use tools of game theory to achieve co-ordination amongst a set of rational and autonomous agents without an explicit co-ordination mechanism built into these agents a priori. In other words, they do not assume the 'benevolent agent assumption'.

The key concepts in this game theory approach to negotiation are:

- *Utility Functions*: Utility can be defined as the difference between the worth of achieving a goal and the price paid in achieving it.
- *Space of Deals*: A deal is an action an agent can take which has an attached utility.
- *Strategies and Negotiation protocols*: The negotiation protocol defines the rules which govern the negotiation, including how and when it ends (e.g., by agreement or without a deal).

The actual negotiation proceeds as follows. Utility values for each outcome of some interaction for each agent are built into a pay-off matrix, which is common knowledge to both (typically) parties involved in the negotiation. The negotiation process involves an interactive process of offers and counter-offers in which each agent chooses a deal which maximises its expected utility value. There is an implicit assumption that each agent in the negotiation is an expected utility maximiser. At each step in the negotiation, an agent evaluates the other's offer in terms of its own negotiation strategy.

Despite its mathematical elegance, game theoretical models suffer from restrictive assumptions that limit their applicability to realistic problems. As Nwana (1996) criticises, agents are presumed to be fully rational and acting as utility maximisers using pre-defined strategies. All agents have knowledge of the pay-off matrix and therefore full knowledge of the other agent's preferences. This is certainly unlike the real world where negotiations are conducted under uncertainty, involve multiple criteria rather than a single utility dimension, agents are non-benevolent, have partial or incomplete knowledge of their own domains, and keep their own utilities private.

3.6.3.2.1.2 Auctions

An area of growing interest in the Electronic Commerce domain is the concept of the Virtual Enterprise. Currently, the time interval within which a product can be marketed successfully is dramatically decreasing; therefore, enterprises are faced with hard terms of competition. Decreasing innovation cycles, changing market situations as well as growing specialisation in individual market segments demand new ways of economic thinking, increasingly forcing enterprises into co-operations or virtual enterprises, sometimes even with direct competitors. These co-operations enable enterprises to share skills, costs, access to one another's markets and resources and, at the same time, decrease the risk of investments.

An example of a MAS, which attempts to enable individual resources via competitive negotiation to form virtual enterprises, is AVE - Agents in Virtual Enterprises (Fischer et al., 1996). Negotiation occurs via an *auctioning mechanism* in AVE between individual competitors for a specific task. In this auction, the product/service manager has the role of auctioneer, the subjects of the auction are the individual partial processes forming the overall service, and the bidders are enterprises that are interested in contributing these partial processes to the virtual enterprise. There are several different auction mechanisms proposed (Murnighan, 1991), each theoretically proven to be of equivalent efficiency. It employs *Vickery's Mechanism* (Vickery, 1961), where the highest bid wins; however, the winning bidder has to pay the second-highest bid price. This is the sealed-bid analogue to an English auction: a bidder must only beat the next highest bidder to win; therefore it may be willing to pay more than its last bid. In contrast to most negotiation protocols considered so far in MAS research, the selection of the highest bid for a partial process is a multi-attribute problem.

3.6.3.2.1.3 Bidding

Skarmeas and Clark (1996) use a MAS to control a switch-based network in order to provide support to customer requests for services of different requirements in network resources. The negotiation techniques employed in the telecommunications domain generally involve simple *bidding mechanisms* based on the contract net protocol. Each agent is decomposed into simpler processes, one such process being responsible for co-ordination and negotiation. Using negotiation, the agents handle (1) customer requests such as the routing of a call from Los Angeles to London and (2) network node failure by deciding who will be the new manager of resources controlled by a recently failed agent.

Communication and co-ordination occur using a central message-board, which has been implemented using the language platform, April (Agent Process Interaction Language). Hence, co-ordination adopts the master/slave approach. Advertisements for services (e.g., routing) are posted to the message-board, which then forwards (or broadcasts) them onto the relevant parties. Bidders respond directly to the sender (not through the message board), who then selects the best offer.

3.6.3.2.1.4 Human-Inspired Approaches

Sycara (1989) adopts a case-based approach to negotiation, based on the belief that human negotiators draw from the *past negotiation experiences* to guide present and future ones. The rationale is that negotiation is an iterative activity which involves exchange of proposals and counter-proposals in order for the parties to reach agreements.

The PERSUADER system consists of three agents: a company, its union, and the mediator whose task is to help the other two agents reach an acceptable compromise. PERSUADER's input is the set of conflicting goals of the company and union, and the dispute context. Its final output is either a single plan in the form of an agreed upon compromise, or an indication of failure if the parties to the dispute did not reach agreement in a particular number of negotiation cycles. The negotiation involves multiple issues, such as wages, pensions, seniority, subcontracting, etc. Each agent's multi-dimensional utility model is private knowledge. Agents can modify each other's beliefs, behaviours and intentions via persuasion. It models the iterative exchange of proposals and counter-proposals to reach consensus in the real world.

3.6.3.2.2 Co-operative Negotiation

This is used in situations where agents have 'a global goal or single task envisioned for the system' (Smith & Davis, 1981), e.g., distributed systems that have been centrally designed to pursue a single global goal. These agents are sometimes called 'collaborative' (Chu-Carroll & Carberry, 1995). *Collaborative Agents* share their knowledge and beliefs to try to maximise the benefit of the community as a whole. For example, agents in air traffic control (representing the pilots and computerised air traffic control systems) engage in a collaborative negotiation process to resolve conflicts via collaborative discourses, instead of competitive bidding (Chu-Carroll & Carberry, 1995). It involves an agent detecting conflicts regarding proposed actions and beliefs from other agents and initiates collaborative negotiation to resolve such conflicts. This negotiation involves the agent modifying the proposal with appropriate justification for this modification, based on its own beliefs. The agents then update their knowledge about other agents' preferences and the negotiation process resumes with a new proposal. Conflicts between agents are handled in a concurrent conflict resolution cycle.

3.6.4 Inter-Agent Communication[2]

Communication enables the agents in a multi-agent system to exchange information on the basis of which they co-ordinate their actions and co-operate with each other. There are at least two prerequisites to allow agents to inter-operate, namely a common Agent Communication Language (ACL) and an ontology.

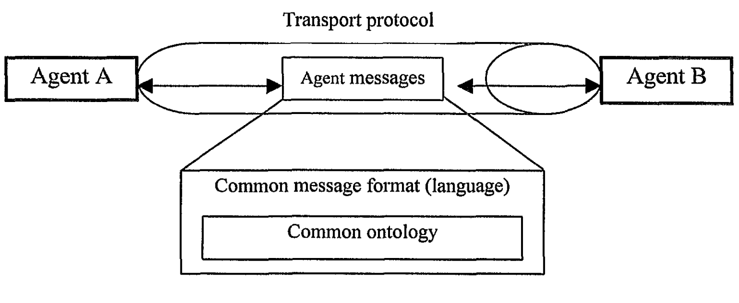


Figure 3.2 Basic Groundwork for an Effective Agent Communication

3.6.4.1 Agent Communication Languages (ACLs)

The inherently heterogeneous and distributed nature of a multi-agent system makes communication among agents a difficult process. It is important to design an

2 This issue will be discussed further in the 'Future Challenges' section of this chapter.

expressive common language for communication with an agent-independent semantics, so that agents can communicate with their peers by exchanging messages and interact together through explicit linguistic actions. As communicating agents will have different knowledge bases, the language system must allow for these differences, in order for communication and co-operation to succeed, despite these disparities. Thus each agent should have a linguistic layer supporting an agent independent semantics which provides a message-based interface that is independent of the agent's internal data structures and algorithms.

A number of ACLs have been designed. An ACL provides agents with a means of exchanging information and knowledge. ACLs derive their inspiration from speech act theory[3] (Searle, 1969), i.e., messages are actions or communicative acts, as they are intended to perform some action by virtue of being sent. It means that utterances by agents are not simply propositions that are true or false, but attempts on the part of the agent to convey some belief or knowledge or an intention. Speech act theory uses the concept of performatives[4] to allow an agent to convey its *beliefs, desires and intentions*. The performatives are the speech-act component of the language and determine what one can 'do' or 'perform' with the content of the message. For example, performatives 'assert', 'affirm', 'state', convey a belief, performatives 'ask', 'order', 'enjoin', 'pray', or 'command' convey a wish or a desire, and performatives 'vow', 'pledge', or 'promise' convey an intention.

The two main standards proposed to date are Knowledge Query and Manipulation Language (KQML) (DARPA, 1993) and FIPA ACL (FIPA, 1999).

[3] The theory was originally developed by linguists in an attempt to understand how humans use language in every day situations, to achieve everyday tasks, such as requests, orders, promises, etc.

[4] A speech act can be put in a stylised form that begins 'I hereby request ...' or 'I hereby declare ...'. In this form, the verb is called the performative, since saying it makes it so. Verbs that cannot be put into this form are not speech acts, e.g., 'I hereby solve this problem' does not actually solve the problem.

3.6.4.1.1 KQML (Knowledge Query and Manipulation Language)

KQML is an evolving standard ACL, being developed as part of the DARPA knowledge-sharing effort (KSE). The central concept is that knowledge sharing requires communication, which in turn, requires a common language. KQML is a high-level, message-oriented communication language and protocol[5] for information exchange independent of content and syntax and applicable ontology. Thus, KQML is independent of the transport mechanism (e.g., TCP/IP, IIOP[6]), content language (e.g., KIF (Knowledge Interchange Format), Prolog), and ontology assumed by the content. Conceptually, three layers in a KQML message can be identified - the content, message and communication layers. The content layer specifies the actual content of the message; the set of performatives provided by the language constitute the message layer (e.g., tell, reply, ask-if, advertise, etc.); the protocol for delivering the message that subsumes the content defines the communication layer.

3.6.4.1.2 FIPA Agent Communication Language

The Foundation of Intelligent Physical Agents (FIPA) Specification (1997) defines an interaction protocol as an explicitly shared multi-agent plan containing communicative acts (like speech acts). The specification adopts SL (Semantics Language) for the formal definition of language semantics. It says the SL language may be used for actual representation of message content. SL propositions are expressed in a logic of mental attitudes and actions, formalised in first order modal language with identity. The mental modal of an agent is based on the representation of three primitive attitudes: belief, uncertainty and choice. A fundamental property of

[5] A dialogue protocol is a common pattern of conversations used to perform some generally useful tasks. The protocol is often used to facilitate a simplification of the computational machinery needed to support a given dialogue task between two agents.

[6] Transmission Control Protocol/Internet Protocol (TCP/IP) and Internet Inter-Object Request Broker Protocol (IIOP) are sets of networking protocols (low-level) that drive the Internet, regulating how data is transferred between computers.

SL's proposed logic is that the modelled agents are perfectly in agreement with their mental attitudes.

FIPA ACL is almost identical to KQML with respect to their basic concepts and the principles they observe. They are syntactically identical. Both languages assume a basic non-conformance to a reserved content language. The two differ primarily in the details of their semantic frameworks. This is a problem because, without a precise semantics, agent designers cannot be certain that the interpretation they are giving to a 'performative' is in fact the same as the one another designer intends to have. Another difference between the two is in their treatment of pragmatic issues such as registering, updating registration information, and finding other agents that can be of assistance in processing requests. Though semantics have dominated the debate, pragmatic concerns such as offering naming and registration services along with basic brokering facilities should be among the immediate concerns.

Despite these problems, KQML has played an important role in defining what an ACL is and what the issues are when it comes to integrating communication into agent systems. Though the lack of a sanctioned specification has probably impeded the adoption of KQML for many big projects, it has allowed much experimentation with dialects and variations on the theme. It is hoped that FIPA will supply the needed sanctioning body for the next iteration of a standardised language.

3.6.4.2 Ontology

An ontology is an explicit specification of the structure of a certain domain, e.g., electronic commerce. It provides a vocabulary for representing and communicating knowledge about some topics and a set of relationships and properties that holds for the entities denoted by that vocabulary. Agents that communicate in a common language will still be unable to understand one another if they use different vocabularies for representing shared domain concepts. A common ontology is necessary to make sure that the same concept, object, or entity has a uniform meaning across applications even if different applications use different names for it, i.e., the semantic content is preserved among applications. This is another big problem that should require attention, as discussed in chapter one.

3.6.5 Mobility

Mobility refers to an agent's ability of migrating from a host to another in a network (White, 1995). The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport itself to another host and resume execution. Strong mobility refers to the ability of a system that allows the migration of both the code AND the execution state[7] (mobile objects) of a mobile agent, e.g., General Magic's Telescript is a strong mobility platform. Weak mobility refers to the ability of a system that allows mobile code (e.g., Java applets) movement between different hosts in a system.

Mobile agents need to be able to run in many (perhaps heterogeneous) hosts, and thus tend to be programmed in languages that can be interpreted, e.g., scripts. They usually require an environment in which to run on each host. A mobile agent environment is a software system distributed over a network of heterogeneous computers. Its primary task is to provide an environment in which mobile agents can execute.

3.6.5.1 Mobile Agent Technologies

There are many commercial mobile agent systems already on the market. Most of them are built on top of the Java system, such as Aglets (1996), Concordia (1998), Odyssey (1997) and Voyager (1995). However some projects have attempted to build mobile agent systems from the ground up, e.g., Telescript (White, 1995).

[7] Agent code is the set of instructions used by an agent. Agent state is the execution state, or attributes of an agent.

3.6.5.1.1 Java (Sun Microsystems)

Java[8] is a programming language that has features like portability, threads, and networking capability which make it really useful in a mobile context. Its main features are (Gosling & McGilton, 1995):

- Object-oriented;
- Platform independent and portable;
- A comprehensive, but not perfect, security system;
- Multi-threaded execution with synchronisation between threads;
- Sophisticated and comprehensive networking capabilities;
- Support for distributed systems through the Remote Method Invocation (RMI) and Object Serialisation (OS) facilities[9].

Taken individually, the characteristics discussed above can be found in a variety of software development environments. What makes it stand out is the manner in which Java and its run-time system have combined them to produce a flexible and powerful programming system which supports distributed computing.

Java, on the other hand, contains no mobile agents. It is simply an enabling technology for mobile agents.

3.6.5.1.2 Telescript (General Magic)

Telescript is an object-oriented mobile agent language. It claims to be 'a platform that enables the creation of active, distributed network applications'. Its main

[8] Java is chosen as the language for the prototype of this thesis, and will be discussed in detail in chapter six.

[9] See 'Requirement Analysis' section of chapter six for details.

achievements to date are its involvement in the Sony MagicTalk[10] product, and its research into electronic market places. The main concepts of Telescript are: agents, places and the 'go' and 'meet' instructions. Agents travel from place to place using the 'go' instruction. They can interact with each other and any services located at the places through the use of the 'meet' instruction.

Telescript is a whole new platform-independent system consisting of a language, and interpreter called the Telescript engine. The places in Telescript are equivalent to the concept of a mobile agent environment where static services are located at a host. The computation is the Telescript language itself and the agent is simply a Telescript object. A Telescript agent maintains its execution state during travel. Security features in Telescript are implemented with the authority, region and identity concepts. Access to resources is granted through the use of permits.

Though Telescript is technically a very sophisticated mobile agent system, it is also costly. A Telescript engine requires significant computer resources to run (e.g., 96 megabytes of RAM memory). Financially, it is a very expensive piece of software that inhibits its acceptance as a general mobile agent system among nearly all mobile agent developers.

3.6.5.1.3 Mobile Agent Commercial Systems

Product	Language	Company
Aglets	Java	IBM Japan
Concordia	Java	Mitsubishi Electric, USA
Odyssey	Java	General Magic
Voyager	Java	Object Space

Table 3.3 Examples of Mobile Agent Systems

[10] MagicTalk (http://www.genmagic.com) is an intelligent personal communications system. It uses Telescript to allow different forms of communication (phone, fax, email, etc.) to interact with the user irrespective of his geographical location.

3.6.5.1.3.1 Aglets (IBM, Japan)

Aglets (Chang & Lange, 1996) are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it brings along its program code as well as its data. Conceptually, the aglet is a mobile agent because it supports the ideas of autonomous execution and dynamic routing for its itinerary. An aglet can also leverage all other facilities of the Java system.

Aglets (Chang & Lange, 1996) has the following features:

- A globally unique naming scheme for agents,
- A travel itinerary for specifying complex travel patterns with multiple destinations and automatic failure handling,
- A white board mechanism allowing multiple agents to collaborate and share information asynchronously,
- An agent message-passing scheme that supports loosely coupled asynchronous as well as synchronous peer-to-peer communication between agents,
- A network agent class loader that allows an agent's Java byte code and state information to travel across the network, and
- An execution context that provides agents with a uniform environment independent of the actual computer system on which they are executing.

3.6.5.1.3.2 Concordia (Mitsubishi Electric, USA)

Mitsubishi Electric Information Technology Center of America has developed a Java-based framework for development and management of network-efficient mobile agent applications for accessing information anytime, anywhere, and on any device. Concordia offers a flexible scheme for dynamic invocation of arbitrary method entry points within a common agent application. It provides support for agent persistence and recovery and guarantees the transmission of agents across a network.

Concordia has also been designed to provide for fairly complete security coverage from the outset. Within Concordia, an agent's travel plans are specified by its Itinerary. The Itinerary is a completely separate data structure from the agent itself. Concordia provides two forms of asynchronous distributed events: selected events and group-oriented events. The event selection paradigm enables agents to define the types of event they wish to receive. In contrast, group-oriented events are distributed to a collection of agents (known as an event group) without any selection.

3.6.5.1.3.3 Odyssey (General Magic)

Odyssey is General Magic's initial implementation of mobile agents in Java. It borrows from many of General Magic's concepts in the Telescript tool set. Odyssey is an agent system implemented as a set of Java class libraries that provides support for developing distributed, mobile applications. Odyssey technology implements the concepts of places and agents. It models a network of computers, however large, as a collection of places. A place offers a service to the mobile agents that enter it. A communicating application is modelled as a collection of agents. Each agent occupies a particular place. However, an agent can move from one place to another, thus occupying different places at different times. Agents are independent in that their procedures are performed concurrently. Odyssey provides Java classes for mobile agents and stationary places. Odyssey agents are Java threads. They rely on the same security services as all other Java applications.

3.6.5.1.3.4 Voyager (ObjectSpace)

Voyager is a Java-based agent-enhanced Object Request Broker (ORB) developed by ObjectSpace, Inc. It allows Java programmers to create sophisticated network applications quickly and easily using both traditional and agent-enhanced distributed programming techniques. It provides for creation of both autonomous mobile agents and objects. Voyager agents roam a network and continue to execute as they move. Voyager can remotely construct and communicate with any Java class, even third party libraries, without source. It allows seamless support for object mobility. Once created, any serialisable object can be moved to a new location, even while the object is receiving messages. Messages sent to the old location are automatically forwarded to the new location.

Disregarding the claims and performances of these systems, existing mobile agent implementations show two major trends: (1) *Java* is rapidly becoming the language of choice, and (2) Each implementation introduces its own variety of supporting agents and services for tasks such as naming, authentication, monitoring, and brokering (Labrou, 1999). Some agreement on the assumptions of these services is needed so that they can be provided as a standard suite of tools.

3.6.5.2 Security Considerations

Mobile agent security can be split into two broad areas (Chess et al., 1995b). The first involves the protection of host nodes from destructive mobile agents while the second involves the protection of mobile agents from destructive hosts.

3.6.5.2.1 Protection of Hosts from Malicious Agents

A mobile agent system is an open system (Chess et al., 1995a). Therefore, just like in any open system, the host nodes are subject to a variety of attacks, both old and new. Attacks on host security fall into four main categories:

- *Leakage*: acquisition of data by an unauthorised party.
- *Tampering*: alteration of data by an unauthorised party.
- *Resource stealing*: use of facilities by an unauthorised party.
- *Vandalism*: malicious interference with a host's data or facilities with no clear profit to the perpetrator.

The traditional methods of attack include eavesdropping, masquerading, message tampering, message replay and viruses. A mobile agent can employ any of these methods of attack, which in turn, can be guarded against using standard techniques such as cryptography, authentication, digital signatures and trust hierarchies.

However a mobile agent is unique in that its code is executed by a host. Thus an executing mobile agent has automatic access to some of the host's resources. With this level of access, mobile agents can mount attacks by altering other local agents, propagating viruses, worms and Trojan horses, impersonating other users and mounting denial of service attacks. The standard approach to this problem is to reject all unknown code from entry into a host. However, this is not a viable solution in a mobile agent environment.

Telescript offers one approach to the problem. The Telescript approach provides three mechanisms which can be applied at various degrees of granularity.

- *Process (Agent) Safety and Security*: This allows safe interaction among agents and between agents and the host. It achieves this by using Authenticated Identities, Protected References, quota on Permits and Engine-mediated Protocols for agent rendezvous with other agents and host entry.
- *System Safety*: This controls access to system supplied resources. It is achieved by forcing access to these resources through what are effectively proxy objects. The

following proxies are supported: External File (file access), External Handle (network access) and the Control Manager (management functions).

- *Network Security*: This provides authentication facilities, communication privacy and system level authorisation. It is achieved through the use of Region Policies, Secure Channels and Export Restrictions.

The key to this approach is the fact that Telescript is based on an interpreted language that facilitates detailed control over the capabilities of the unknown process/agents running on top of it. Java (Gosling & McGilton, 1995) also provides similar security features.

3.6.5.2.2 Protection of Agents from Malicious Hosts

This area deals with the issue of protecting mobile agents from hosts that want to scan the agent for information, alter the agent's state or code, or kill the agent. The critical problem is that in order for the agent to run, it must expose its data and code to the host environment which supplies the means for that agent to run. Thus the mobile agent is unprotected from the host.

Current consensus is that it is computationally impossible to protect a mobile agent from a malicious host (Rasmusson & Janson, 1996). Instead of tackling the problem from a computational (hard) point of view, current research is looking at sociological (soft) means of enforcing good host behaviour.

3.6.5.3 Communication Protocol Requirements

Generally speaking, the following communication protocol functions are required for agent mobility:

- Naming conventions for all entities in the mobile agent system (agents, hosts, services etc.).

- Access to information regarding a remote mobile agent environment.
- The ability to move a mobile agent into a 'suspended' life-cycle state ready for transportation to a remote host.
- The ability to transport a mobile agent, which is in the 'suspended' state, to a remote mobile agent environment.
- The ability to receive a suspended mobile agent from a remote host and reconstitute it in a new environment.

Some optional services that aid navigation include:

- Directory and referential services can aid mobile agents in the discovery of relevant services that the agent might visit.
- Network topology information services could supply information about the state of the network. This information can help a mobile agent to make planning decisions based on the quality of different parts of the underlying network(s).

3.6.5.4 The Current Debate

There is certainly a strong interest and growing community of mobile agent research. There are also frequent concerns (Nwana & Ndumu, 1999) that mobile agents may promise too much superficially, but in actual fact might deliver little.

Mobile agent advocates usually claim that the technology has the following main advantages:

- *Reduce bandwidth*: Mobile agents consume fewer network resources since they move the computation to the data.

- *Asynchronous autonomous interaction*: Tasks can be encoded into mobile agents and then dispatched. The mobile agent can operate asynchronously and independent of the sending program.

- *Minimise latencies*: Mobile agents can be dispatched from a central system to control real-time entities at a local level, e.g., nuclear system control.

- *Robustness and fault tolerance*: The ability of mobile agents to react dynamically to adverse situations makes it easier to build fault tolerant behaviour.

What the opposite camp suggests is, first of all, these advantages remain controversial and are highly application-dependent such as in critical situations where latencies are intolerable e.g., nuclear system control. Besides, it is quite obvious that mobile agents are still a solution with no clear problem.

What is certain is that really useful mobile agent applications may only emerge after their static counterparts have been fully developed. There may be certain conditions under which mobile agents are useful, and there are others when they are not. Empirical research is required to point out where and when mobile agents are truly valuable because 'true' mobile agents would tend to be large, and the costs of sending large agents across the network may well be greater that its benefits. Future is the best jury, i.e., mobile agents need to prove their usefulness in real applications, and more importantly, prove that their usefulness exceeds those security risks outlined earlier. Otherwise, mobile agents just bring an additional set of problems on top of those that are associated with static agents already. This is because designers have to worry about issues like remote code execution for 'true' mobile agents. They also have to worry about issues of weak or strong migration; the latter refers to a situation where the entire agent state (i.e., data and execution state) is transferred along with code to the next location. Strong migration can be very time consuming and expensive for large agents with multiple threads – which they would have to be if they are of any use. Weak migration, on the other hand, leads agent designers to design more complicated agents because of the security concerns.

Other obvious research challenges include the following:

- Research has to be carried out towards mechanisms for agent security; much of this is already on going.

- Mobile agents clearly require transactional management support structures if they will be of any use. What sort of transaction models should be employed? For example, when an electronic cash carrying agent migrates to a supplier's server, both the server's state and the agent's state change after completion of the payment transaction. In case the transaction failed, the customer agent must be able to recover the states just before the transaction such that it can return to its owner. The server's state also needs to be 'rolled-back'.

3.7 Future Challenges

The previous section presents a review on agent capabilities. Currently, most existing systems are built by designers and developers who are themselves active in the agent research community, and hence, most of the implementations are ad hoc in nature, built with custom methodology for use in a limited domain. While this may suffice for niche applications, agent technology has the potential to be far more ubiquitous. It is obvious that some fundamental issues will need attention before agent systems can be introduced to the public en masse.

3.7.1 Human Factors

The issue of *trust* is very important in any agent system, especially when money is involved. By definition, delegation implies relinquishing control of a task to an entity with different memories, experiences and possibly agendas. Thus, there is a certain risk that the agent will do something wrong. A crucial issue in developing trust in agent systems is the ability of an agent to exhibit somewhat *predictable behaviour*. It is essential that people feel in control of their lives and surroundings. They must be comfortable with the actions performed for them by autonomous agents, in part through a feeling of understanding, and in part through confidence in the systems. The first step is probably to *build reliable systems* that do not act unreasonably (within the limits of current theory and technology), while insisting that humans remain in the loop. This introduces some inefficiency into the overall system, but people are likely to demand it for quite a while. However, if the design is right in the first place, it is just a matter of time before people get used to trusting their agents.

Furthermore, people expect their *privacy* to be guaranteed by intelligent agents. Luckily agents do not necessarily imply a loss of privacy. Technically, security mechanisms are constantly being improved with secure communications technology such as public encryption and authentication services. Regarding the issue of whether there is a guarantee that users' agents will not spread confidential details, either unintentionally or intentionally, to those parties that they interact with, this is again a design and hence trust issue. However, none of the negative aspects of agents such as loss of control and privacy are inevitable. All can be eliminated or minimised, but only if these aspects are considered in the design on the outset.

3.7.2 Economic Factors

The ultimate test of agents' success will be mass usage. User adoption of agents will not be driven by the agent's capabilities, but by the *availability of agent applications* that the user finds useful and convenient. At the same time, the motivation of suppliers to adopt agents is that they prove to be *profitable* investments and whether or not they meet a certain market or *user demand* (and how well this demand has been met). Areas with clear market potential may include (Nwana & Ndumu, 1999):

- electronic commerce – providing value-added and personalised support and recommendation to online retail, supply chain integration and personalised customer interaction;
- corporate Intranets or managed Extranet solutions (e.g., the AmericaOnline, BT's LineOne or Excite's online communities) - providing value-added and personalised searching and collation support;
- personal assistants such as General Magic's Portico system wherein personal agents allegedly provide a service which filters emails and communications, routes telephone calls or answers them, books appointments, etc.
- resource allocation and management systems, for example, for telecommunications network management, production planning, air-traffic control, etc. and
- agent-based middleware – to mediate between applications and network-layer applications.

The emphasis on market sector problems suggests that future research work will be forced to tackle real-world problems. The road to this success, as mentioned in chapter one, is to have early simple (and cheap), but potentially useful versions of agents available for users to experiment with and try out the technology. A lot of attention should be paid to the demands of users in the 'real world'.

3.7.3 Technical Factors

Having said that 'real-world' applications are needed to drive the demand of agents, there are some technical issues that need to be settled before this can really be done well.

3.7.3.1 Interoperability

Current research has come up with a variety of agents that help users with different types of everyday tasks such as e-mail filtering. There are few problems in terms of how these agents collaborate, how they communicate, what ontology they use, as long as they are created with the same methodology and hence homogeneous. However, users want to move towards a situation in which these agents can be heterogeneous and manufactured by different vendors. In addition, users want these heterogeneous agents to be able to collaborate with one another. Consider a customer who buys an agent from one company to filter his/her e-mail, and a personal news-filtering agent from another company. Ideally the two agents should be able to exchange information and collaborate. For example, topics that the e-mail agent has found the customer gives high priority to are probably things that he/she wants to receive news stories about as well. Similarly, if the customer gets lots of e-mail from a particular person at a particular company, he/she probably wants to receive news stories that mention that same company, and so on. This will require more generic languages and a shared ontology that agents can use to exchange information.

- *Inter-Agent Communication:* this problem has been discussed briefly in chapter one in the context of tourism. Communication is at the heart of co-ordination and negotiation in multi-agent systems. Despite all the standardising efforts, FIPA ACL and KQML are still ambiguous and vague. KQML has had mis-identified or missing performatives which FIPA ACL is trying to address. However, this culminates in a situation where one still does not know when to use ACL's myriad of performatives. The average agent designer would not be an expert in modal logics before they are expected to use FIPA ACL. Though having a common ACL may offer an answer, this is not the one-and-only approach. The ultimate goal is to have a standardised high-level communication protocol.

- *Ontology:* As mentioned earlier in chapter one, this problem is at the core of the agent interoperability issue. Current research fails to appreciate the magnitude of the ontology problem. The ontology issue has always been considered secondary to other issues such as co-operation, negotiation, formalisation and logics for beliefs, desires and intentions, etc. Perhaps, this is a case of sheer escapism. FIPA (1998) is increasingly becoming aware of the ontology problem, but concerted research towards ontology problem is absolutely crucial in the long run.

3.7.3.2 Legacy Software Integration

Requirements for open, heterogeneous component-based systems include backward-compatibility to 'legacy' systems. Agents must be able to interact with legacy software. Genesereth and Ketchpel (1994) suggest three possible solutions to the problem.

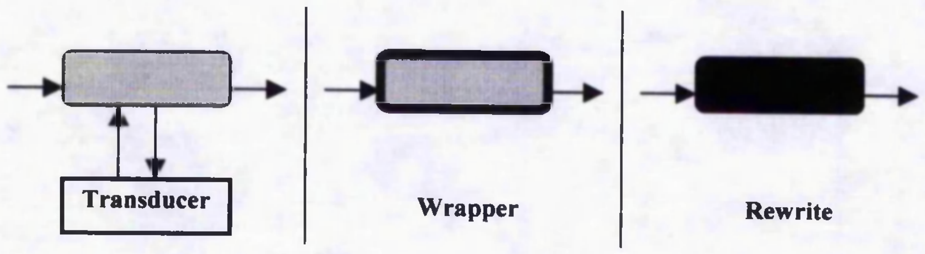


Figure 3.3 Three Approaches to Integrate Legacy Software

One approach is to implement a transducer that mediates between an existing program and other agents. The transducer accepts messages from other agents, translates them into the program's native communication protocol, and passes those messages to the program. It accepts the program's responses, translates into ACL, and sends the resulting messages on to other agents. A second approach is to implement a 'wrapper', i.e., inject code into a program to allow it to communicate in ACL. The wrapper can directly examine the data structures of the program and can modify those data structures. The third and most drastic approach is to rewrite the original program.

The 'wrapper' technique where the legacy program is augmented with code that enables it to communicate seems the most feasible. The idea is to incorporate the legacy systems into an agent system by 'wrapping'' them with an 'agent layer' that provides an agent-level application program interface[11] (API). By this, the functionality of the legacy systems can be extended by enabling them to work with other newly developed software components.

3.7.3.3 Evaluation Techniques

This problem is still outstanding. In the short term, criteria for judging agent solutions should be defined, e.g., whether they offer 'value-added' solutions to other conventional applications and approaches, e.g., expert systems, distributed problem-solving approaches, blackboard approaches. The degree of 'value-added' could range from 'excellent' (i.e., no solution is probable without an agent approach), through 'minimal' (where an agent solution is marginally of value) to 'very poor' (where conventional approaches offer better solutions). One may argue that rigorous evaluation is essential. Methods and tests need to be developed to verify and validate agent systems, so it can be ensured that they meet their functional specifications. However, success or failure is not always related to technical superiority (or to the lack of it), so it is hard to measure prospects, chances and relevance of different techniques and technologies. In the long term, the most objective criterion for evaluating agents will be the market. Users will not be buying into technologies unless there are clear benefits which accrue from them, unless they fill a need. The same will be true of agents if they are to be successful in the long term. They will not be taken up otherwise.

[11] An API is an agreed-upon input/output format for a particular application program (APP). Humans, or application programs, may rely on that format and in particular use it to 'call' the APP and to interpret the output returned by the APP.

3.7.4 Legal and Social Concerns

- *Legal Responsibility:* when a user relinquishes some of his/her responsibility to one or more software agents, he/she should be (explicitly) aware of the authority that is being transferred to it/them. How rules can be formulated and how laws can be used to regulate (unwanted) agent behaviour and to deal with various (future) legal issues (e.g., who is responsible for an agent's actions)?

- *Social Responsibility:* How to make sure that agents conform to acceptable social behaviour such as tidiness (an agent should leave the world as it found it), thrift (an agent should limit its consumption of scarce resources) and vigilance (an agent should not allow client actions with unanticipated results)?

3.8 Concluding Remarks

It is important to point out that despite the hype, agent systems are not fundamentally different, neither are they something 'new'. Indeed, a lot of the solutions to personal and multi-agent systems are already available. What researchers need to do is a creative synthesising of some of these already invented wheels. Where necessary, 'new wheels' may be invented to link up the old ones. This is the design philosophy of the PTA system described in this thesis. In carrying out this research project, a wide and diverse literature, including agent communication languages, distributed object technologies, the co-ordination, co-operation, negotiation literature, some views from rational agency, planning, scheduling, methodological issues, research on ontology, HCI design, etc. were consulted. Existing theories and techniques were drawn and employed wherever suitable. During this process, a new wheel is constructed – a collective learning multi-agent system running on existing HTTP servers. The whole then becomes greater than the sum of its different parts, and hence novel. However, this does not mean it is fundamentally different.

'Technical innovation - the devising of new tools - is surely a desirable activity. But unless there is a balance between our fascination with tools and our concern for the ends they may help us achieve, the tool becomes tyrannical. What stares us in the face today is the startling fact that, not only has the balance been upset, but one of its terms has virtually disappeared. Technological innovation now proceeds for its own sake, driven by its own logic, without reference to human need. We are a society obsessed with new tools, but incapable of asking in a serious way, "What are we developing these tools for?"' (Talbott, 1995)

After all, our pressing need is not for more techniques, but *solutions* to real-world *problems*.

3.9 References

Aglets, (1996), http://www.trl.ibm.co.jp/aglets.

Angeline, P. J., (1995), Evolution Revolution: An Introduction to the Special Track on Genetic and Evolutionary Programming, *IEEE Expert,* June, pp. 6-10.

Bates, J., (1994), The Role of Emotion in Believable Agents, *Communications of the ACM,* 37(7), July, pp. 122-125.

Bradshaw, J. M., Dutfield, S., Benoit, P. and Woolley, J. D., (1997), KAoS: Toward an Industrial Strength Open Agent Architecture, in Bradshaw, J. M. (ed.) *Software Agents,* MIT Press, Cambridge, Mass., pp. 375-418.

Bratman, M. E. Lsrael, D. J. and Pollack, M. E., (1988), Plans and Resource-bounded Practical Reasoning, *Computational Intelligence,* Volume 4, pp. 349-355.

Bussmann, S. and Muller, J., (1992), A Negotiation Framework for Co-operating Agents, in Dean, S. M., (ed.), *Proceedings of CKBS-SIG,* Dake Centre, University of Keele, pp. 1-17.

Chang, D. and Lange, D., (1996), Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW, *Proceedings of the OOPSLA '96 Workshop.*

Chavez, A. and Maes, P., (1996), Kasbah: An Agent Marketplace for Buying and Selling Goods, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technologies (PAAM-96),* London, U. K., pp. 75-90.

Chess, D., Grosof, B., Harrison, C., Levine, D., Parris, C. and Tsudik, G., (1995b), *Itinerant Agents for Mobile Computing,* Technical Report, October 1995, IBM, T.J. Watson Research Center, NY.

Chess, D., Harrison, C. and Kershenbaum, A., (1995a), *Mobile Agents: Are they a good idea?,* Technical Report, March 1995, IBM, T.J. Watson Research Center, NY.

Chu-Carroll, J. and Carberry, S., (1995), Conflict Detection and Resolution in Collaborative Planning, in *Intelligent Agents II, Lecture Notes in Artificial Intelligent 1037,* Heidelberg: Springer Verlag.

Concordia, (1997), http://www.concordia.mea.com

DARPA (1993), *Specification of KQML Agent Communication Language,* Technical Report, ARPA Knowledge Sharing Initiative, External Interfaces Working Group.

Dennet, D, (1978), *Brainstorms,* MIT Press, Bradford, 1978.

Durfee, E., Lesser, V. and Corkill, D., (1989), Trends in Co-operative Distributed Problem Solving, *IEEE Knowledge & Data Engineering,* 1(1), pp. 63-83.

Etzioni, O. and Weld, D. S., (1995), Intelligent Agents on the Internet: Fact, Fiction, and Forecast, *IEEE Expert,* August, pp. 44-49.

Etzioni, O. and Weld, D. S., (1994), A Software-Based Interface to the Internet, *Communications of the ACM,* 37(7), July, pp. 72-76.

FIPA (1998), *Specification, Part 12.*

FIPA (1999), *Specification, Part 2.*

Firefly, (1999), http://www.firefly.com.

Fischer, K., Muller, J., Hemig, I. and Sheer, A-W, (1996), Intelligent Agents in Virtual Enterprises, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96),* London, UK, 1996, pp. 75-90.

Foner, L. N., (1997), Entertaining Agents: A Sociological Case Study, *Proceedings of the First International Conference on Autonomous Agents (Agents 97),* Marina de Rey, CA, 1997, pp. 122-129.

Franklin, S. and Graesser, A, (1996), Is It an Agent, or Just a Program? In Muller, J. P., Wooldridge, M. and Jennings, J. R. (eds.) *Intelligent Agents III* (LNAI Volume 1193), Springer-Verlag, Berlin, Germany, 1997, pp. 21-36.

Genesereth, M. R. and Files, R. E., (1992), *Knowledge Interchange Format,* Version 3.0, Reference Manual, Computer Science Department, Stanford University.

Genesereth, M., Ginsberg, M. and Rosenchein, J., (1986), Co-operation without Communication, *Proceedings of Annual Association of Artificial Intelligence,* Philadelphia, pp. 51-57.

Genesereth, M. and Ketchpel, S. P., (1994), Software Agents, *Communications of the ACM,* 37(7), pp. 48-53.

Georgeff, M., (1983), Communication and Interaction in Multi-Agent Planning, *Proceedings of the Third National Conference on Artificial Intelligence,* Washington, D.C., Morgan-Kaufmann, San Mateo, California, pp. 125-129.

Georgeff, M., (1984), A Theory of Action for Multi-Agent Planning, *Proceedings of the Fourth National Conference on Artificial Intelligence,* Austin, Texas, San Mateo, California, pp. 121-125.

Georgeff, M. P. and Lansky, A. L., (1987), Reactive Reasoning and Planning, *Proceedings of the Sixth National Conference on Artificial Intelligence, (AAAI '87),* Seattle, WA, pp. 677-682.

Gosling, J. and McGilton, H., (1995), *The Java Language Environment: A White Paper*. Sun Microsystems, 1995.

Gilbert, D, Aparicio, M, Atkinson, B, Brady, S., Ciccarino, J., Grosof, B., O'Connor, P., Osisek, D., Pritko, S., Spagna, R. and Wilson, L., (1995), *The Role of Intelligent Agents in the Information Infrastructure,* IBM White Paper, US.

Hensley, P., Metral, M., Shardanand, U, Converse, D. and Myers, M., (1997), *Proposal for an Open Profiling Standard,* Technical Note, W3C, June, 1997.

Huhns, M and Singh, M. P., (1998), *Readings in Agents,* Morgan Kaufmann Publishers, San Mateo, CA, 1998.

Jennings, N. R., Sycara, K. P. and Wooldridge, M., (1998), A Roadmap of Agent Research and Development, *Journal of Autonomous Agents and Multi-agent Systems,* 1(1), July, 1998, pp. 7-38.

Kautz, H. A., Selman, B. and Coen, M., (1994), Bottom-Up Design of Software Agents, *Communications of the ACM,* 37(7), July, pp. 143-146.

Kay, A., (1984) Computer Software, *Scientific American,* 251(3), pp. 53-59.

Kay, A., (1990), User Interface: A Personal View, in Laurel, B., (eds.) *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 191-208.

Lashkari, Y., Metral, M. and Maes, P., (1994) Collaborative Interface Agents, *Proceedings of the National Conference on Artificial Intelligence,* MIT Press, Cambridge, Mass., pp. 444-449.

Lenat, D. B., (1995), CYC: A Large-scale Investment in Knowledge Infrastructure, *Communications of the ACM,* 38(11), pp. 33-38.

Luce, R. and Raiffa, H., (1957), *Games and Decisions*, John Wiley & Sons, NY.

Maes, P., (1994), Agents that Reduce Work and Information Overload, *Communications of the ACM*, 37(7), July, pp. 31-40.

Mazur, J., (1994), *Learning and Behaviour*, Prentice Hall.

McCarthy, J. and Hayes, P., (1969), Some Philosophical Problems from the Standpoint of Artificial Intelligence, in Melzer, B. and Michie, D. (eds.), *Machine Intelligence 4,* Edinburgh University Press, 1969, pp. 463-502.

Minsky, M., (1963), Steps Towards Artificial Intelligence, in Feigenbaum, E. and Feldman, J. (eds.) *Computers and Thought*, McGraw Hill.

Mtichell, T., Caruana, R, Freitag, D, McDermott, J. and Zabowski, D., (1994), Experience with a Learning Personal Assistant, *Communications of the ACM,* 37(7), July, pp. 80-91.

Murnighan, J., (1991), *The Dynamics of Bargaining Games,* Englewood Cliffs, NJ, Prentice Hall.

Negroponte, N., (1970) *The Architecture Machine: Towards a More Human Environment,* Cambridge, Mass., MIT Press.

Nwana, H. S. and Ndumu, D. T., (1999), A Perspective on Software Agents Research, *Knowledge Engineering Review,* 1999, (To appear).

Nwana, H., (1996), Software agents: An Overview, *Knowledge and Engineering Review,* 11(3), November 1996.

O'Hare, G. and Jennings, N., (1996), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, NY.

Rao, A. and Georgeff, M., (1995), BDI Agents: From Theory to Practice, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS '95),* San Francisco, USA, pp. 312-319.

Rasmusson, A. and Janson, S., (1996), Personal security assistance for secure Internet commerce, *Proceedings of the New Security Paradigms '96,* ACM Press, September 1996.

Rosenschein, J. and Genesereth, M., (1985), Deals among Rational Agents, *Proceedings of 9th Artificial Intelligence Conference: Intelligent Agents,* Los Angeles, pp. 91-99.

Rosenschein, J. S. and Zlotkin, G., (1994), *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers,* MIT Press, Boston, MA.

Russell, S. J. and Norvig, P., (1995), *Artificial Intelligence: A Modern Approach,* Englewood Cliffs, NJ: Prentice Hall.

Searle, J., (1969), *Speech Acts*, Cambridge, MA, Cambridge University Press.

Selker, T., (1994), Coach: A Teaching Agent that Learns, *Communications of the ACM,* 37(7), July, pp. 92-99.

Sheth, B. and Maes, P., (1993) Evolving Agents for Personalized Information Filtering, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 345-351.

Shoham, Y., (1993), Agent-oriented Programming, *Artificial Intelligence*, 60(1), pp. 51-92.

Skarmeas, N. and Clark, K., (1996), Process Oriented Programming for Agent-Based Network Management, *Proceedings of the Intelligent Agents for Telecoms Applications, ECAI '96 Workshop.*

Smith, R. and Davis, R., (1981), Frameworks for Co-operation in Distributed Problem Solving, *IEEE Transactions on Systems, MAN and Cybernetics,* SMC-11, No. 1, January, 1981.

Stanfill, C. and Waltz, D., (1986) Towards Memory-Based Reasoning, *Communications of the ACM,* 29(12), December, pp. 1213-1228.

Sycara, K. P., Decker, K., Pannu, A., Williamson, M. and Zeng, D., (1996), Distributed Intelligent Agents, *IEEE Expert,* 11(6).

Sycara, K., (1989), Multi-Agent Compromise via Negotiation, in Gasser, L. and Huhns, M. (eds.), *Distributed Artificial Intelligence 2*, Morgan Kaufmann.

Talbott, S. L., (1995), *The Future Does Not Compute - Transcending the Machines in Our Midst,* Sebastopol, CA, O'Reilly & Associates, 1995.

Thompson, G., Frances, J., Levacic, R. and Mitchell, J., (1991), *Market, Hierarchies and Networks - the Co-ordination of Social Life*, Sage Publications.

Tsvetovatyy, B. and Gini, M., (1996), Toward a Virtual Marketplace, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96),* London, UK, 1996, pp. 75-90.

Voyager, http://www.objectspace.com.

Watkins, C., (1989), *Learning from Delayed Rewards*, PhD Thesis, University of Cambridge, England.

Werner, (1996), Co-operating Agents: A Unified Theory of Communication and Social Structure, in Gasser and Huhns, (eds.), *Distributed Artificial Intelligence,* Volume 2, Morgan Kaufmann.

White, J. E., (1995), *Telescript Technology: The Foundation for the Electronic Marketplace,* White paper, General Magic Inc.

Wiener, N., (1948), *Cybernetics,* Wiley & Sons, NY.

Wooldridge, M. and Jennings, N. R., (1995), Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review,* 10(2), 1995, pp. 115-152.

Chapter Four

Electronic Travel Market Analysis

The Internet and the WWW are radically changing the future of traditional commerce. Indeed, electronic commerce is already considered to be a multi-billion dollar industry, and many analysts (Guilfoyle et al.; 1997, Guttman et al., 1998; Nwana, 1998) predict that agent-mediated electronic commerce would revolutionise Internet commerce even more. The Internet did prove to impact the travel industry significantly - perhaps more so than any other sector. With an increasing variety of travel services available online, the possibility of arranging an entire trip itinerary from a traveller's personal computer becomes more like a reality.

This chapter starts with a survey of the Web to assess its future potential in electronic commerce in tourism. It then points out some of the *problems* that may cast shadows on its long-term prospects. By this, it is possible to single out the core research issues that need to be addressed in order to realise a fully integrated agent-mediated electronic travel market.

4.1 History of The Internet and the World Wide Web

The Internet originated from the development of a project in 1969 by the Advanced Research Projects Agency of the US Department of Defence (ARPA) with a far different intention - the network called ARPANET (Gilster, 1994). It was a network connecting university, military and defence contractors, with an aim to aid researchers in the process of sharing information. As early as 1973, the ARPA began the Internetting Project. The goal was to determine how to link networks and overcome the different methods each network uses to move its information. In 1983, the original ARPANET was split into MILNET (US government military network) - to be used for military communications - and NSFNET (research and academic network) - for continuing research into networking. In 1991, NSFNET and the commercial nets were finally connected to form what is today called the Internet. In 1993, the US Defence Communications Agency mandated TCP/IP for all ARPANET hosts. In doing so, it established a standard by which the Internet could grow. From this point onwards, it would be possible to add more gateways, connecting more networks, while the original core networks remained intact.

From its origin as a US government research project, the Internet grew to serve the research and academic communities. It has become a major component of network infrastructure, linking millions of machines and users around the world. More recently, there has been tremendous expansion of the network into the *commercial* user domain.

One of the most exciting applications on the Internet is the emergence of the World Wide Web. It was invented at the European Centre of Particle Physics (CERN) in 1992. The WWW is a wide-arena *hypermedia* distribution system based on the Hypertext Transport Protocol (HTTP) to support multiple servers. Hyper Text Mark-up Language (HTML) is used to create Web pages which consist of structural hypertext links and can incorporate text, graphics, moving video and sound. They are set up at sites all over the world to provide a wealth of useful information.

4.2 Growth of Web Usage

It is very difficult to produce online market forecasts. As past data showed, they tend to be under-forecasted (Forrester Research, 1998, Jupiter Communications, 1999a). First, past data to be used as observations are not sufficient, so quantitative methods are not entirely reliable. Forecasts of online revenues are even more difficult because Web sites started to make their presence in 1994, but most of them did not generate noticeable revenues until 1996. Unlike traditional sales or retail where experienced managers' opinions can be used as reliable guidelines, there are no experts on online travel. Qualitative methods like panel consensus, therefore, are not possible. Yet, it is worthwhile to: (1) provide some baseline numbers to get insights of the potential of the Web, and (2) demonstrate the difficulties inherent in estimating a complex and moving target. Hence, it must be noted that rapid Web growth and the impact of unforeseen technological innovations will limit long term extrapolation to best estimates. Most reports also show a strong share of US users and Web sites in the sample. This is because Internet business took off earlier in the United States and there has been a larger body of users (see Table 4.1). The probability of a US user or Web site being included in a sample is thus much higher.

Region	1997(%)	1998(%)	1999(%)
North America	65	57	55
Europe	19.7	21.7	23.8
Asia	14.7	17	17.2
South America	2	3	2.7
Africa	0.6	0.8	0.9
Middle East	0.5	0.5	0.5

Source: NUA, Inc. (1997-1999)

Table 4.1 Internet Users by Geographical Location

The purpose of the following survey is to give an idea on the commercial potential of the Web in the travel industry. The survey is based on the results of a number of reports and supporting data from: ActivMedia, Inc. (a US market research firm), American Society of Travel Agents (ASTA), Association of British Travel Agents (ABTA), Datamonitor, (an international market analysis firm), Forrester Research, Inc. (a US market research firm), NUA, Inc. (an international Internet research firm), Phocuswright (a US research firm on travel and tourism), Jupiter Communications (a US market research firm), Travel Industry Association of America (a national, non-profit organization, based in Washington, D.C.), and Find/SVP (a US market research firm). Efforts were made to reconcile the available data, and in cases of controversies, analyses were made to justify the credibility and reliability of the data.

4.2.1 Business Access

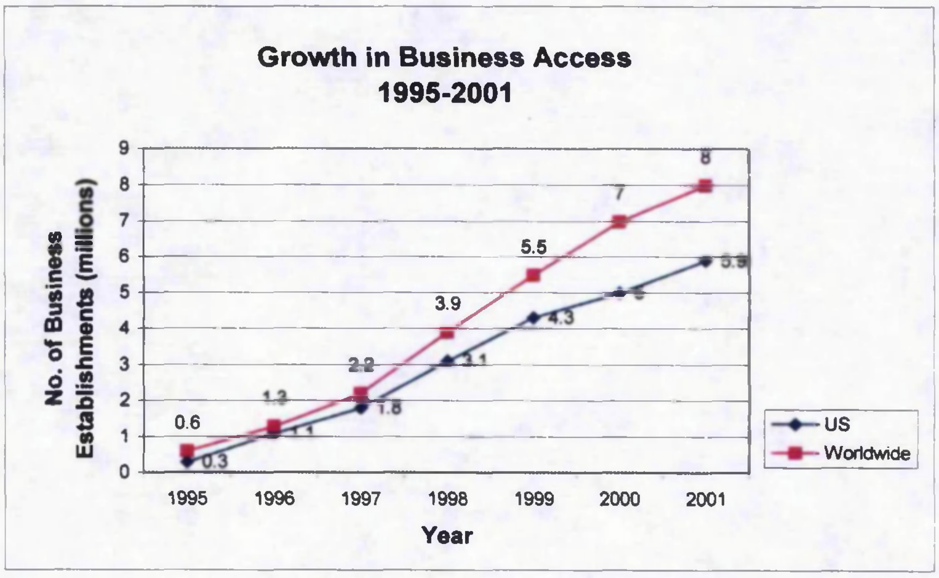
Findings

Companies rushed onto the Internet in waves. Electronics and software companies were first to move onto the Web during late 1994 and early 1995. Later in 1995, publishers and consumer marketers moved onto the Web. Then, manufacturers and small consumer marketers formed the wave moving online during late 1995 and early 1996. In September 1994, there were approximately 600 commercial sites. By May 1995, the number had grown tenfold, to just over 6,000 commercial Web sites. In December 1995, the number had grown to 23,500 commercial sites (ActivMedia Inc., 1995b). As of July 1996, there were almost 95,000 commercial Web sites, including nearly 10,000 in Japanese. In 1995, only 3% of companies with Internet access had Web sites. During the first half of 1996, the number of commercial Web sites listed on Yahoo grew at an astonishing rate of 19% per month. By 1997, 15% of all connected business with Internet access had Web sites[1] (ActivMedia Inc., 1997a).

Predictions

It is predicted that online business access will grow to 5.9 million in the US, or 92% of US business establishments, by 2001, as shown in Figure 4.1.

[1] ActivMedia tracked the number of unduplicated public listings found in its internal Global Directory Unduplication study, which provides the sampling base for the annual 'Real Numbers Behind Net Profits' reports. The sampling method for the 1997 report is illustrated in Appendix A.



Source: ActivMedia, Inc.

Figure 4.1 Growth in Business Access 1995-2001

Worldwide business access on the Web is expected to grow at an even faster rate than in the more mature US market - from 2.2 million at the end of 1997 to 8 million by 2001. Much of this growth will be in the Pacific Rim, followed by Europe, India, South Africa, Israel, China and a few dozen other nations that are beginning to develop Internet infrastructure, but not expected to blossom until the next century, when US and Japanese penetration will be nearly complete.

The only major limiting factor in sight to curtail growth of the Internet and commercial Web sites is supply, i.e., whether the infrastructure can keep up with the rapid growth. Competitive pressures among Web developers and storage providers promise to keep the price of entry low. Cost cutting and competitive pressures among marketers continue to push more toward electronic marketing. Unless something unforeseen halts growth, 70% of all connected businesses will have Web sites by the end of 2001 (ActivMedia, Inc., 1997b).

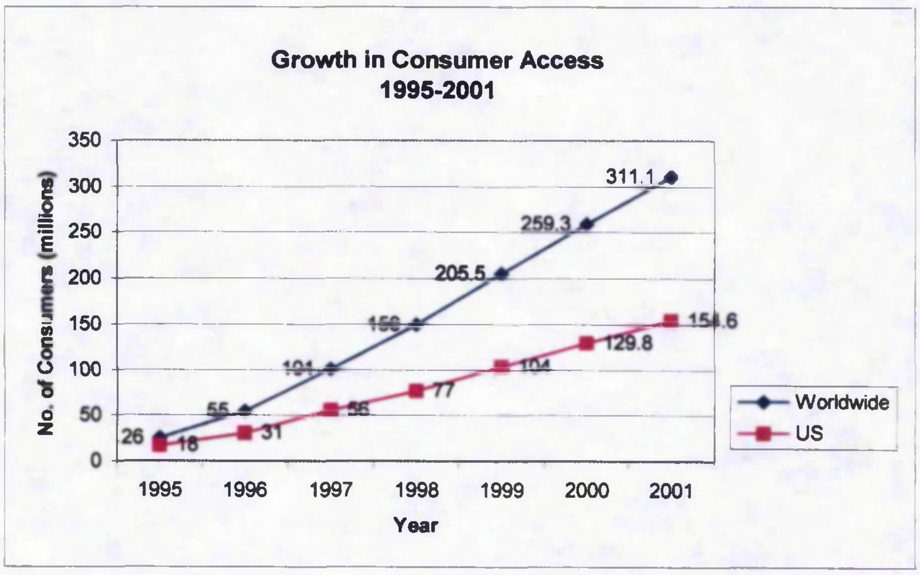
More forces will drive Web growth even after it has penetrated most companies in the developed world. Much has been made of the growth of the 'Intranet' versus the 'Internet'. Between networks that allow open access to the world and those deep behind company firewalls lie restricted access sites, open to registered users - customers, suppliers, subscribers, team members - who may be inside or outside the enterprise. As companies become more familiar with the net, and enabling technologies provide industry-specific tools for commerce on the Web, periods of relative equilibrium punctuated by waves of entrants will broaden the market base, increasing both uses and types of users on the Internet and at restricted access Web sites and intranets (ActivMedia, 1997b).

4.2.2 Consumer Access

Findings

Consumer access[2] to the Web is growing more slowly than business access, but 50-60% growth per year is not to be ignored. Web access in 1994 was fewer than a million people worldwide. By the end of 1995, 18 million individuals in the United States, and about 26 million worldwide, were using the Web, as shown in Figure 4.2.

[2] NUA's definition for an Internet user is an individual, adult or children, who uses a computer from home, work, or school to connect with commercial online services, or other computers over the Internet.



Source: NUA, Inc.[3]

Figure 4.2 Growth in Consumer Access 1995-2001

Fuelled by the incorporation of Web access into MS Windows and consumer promotions from other telecommunication companies, the number of consumers on the Web grew to 56 million in the United States and 101 million worldwide in 1997 (NUA, 1998).

Predictions

Projections for future consumer use of the Web and Internet vary widely across different research companies. This is partly due to the different criteria and terminology they use for customer Internet 'use'[4].

[3] For details on data tables, see Appendix B.

[4] For example, the definitions used by three different surveys in 1995 are as follows:

O'Reilly, with the lowest estimate, uses a restrictive definition of Internet use, defining an Internet user as an individual 18 or over with direct access to the Internet, and access to email and other Internet applications. The O'Reilly count *excludes* individuals whose only access to the Internet is through an online service such as America Online.

Despite the disparate results, all reports showed a solid growth trend. It is believed that technological advances in network infrastructure, increasing bandwidth, improved usability of Internet browsers, and the growth of consumer-oriented content will continue to boost consumer Web usage in the home. Secure Web access with telephone and television systems will be the major reason for the future growth in customer access. Non-PC Internet appliances, such as Net-capable televisions, will significantly expand online usage beyond PC households. TV-based access devices (including game consoles) will emerge as the leading non-PC Internet access appliance, growing quickly from 5.2 million households (or 13% of the consumer online market) in 2000 to 12.7 million households (or 22% of the consumer online market) by the year 2002. Personal computers, however, are likely to remain the premier platform for Internet access in the near future (Jupiter Communications, 1997). New services and existing content providers will cater specifically to the non-PC device market using broadband video delivery for the cable modem and DSL (Digital Subscriber Line) -based televisions, and wireless technologies for portable phones and PDAs (Personal Digital Assistants). By the year 2000, it is predicted that 16% of Internet access will be from non-PC devices, and television will emerge as the premier non-PC access device (Jupiter Communications, 1997).

Many of these technologies should gain a foothold by 2000, continuing to draw more consumers to the Web - and provoking an even greater surge in Web commerce, with online transactions well on their way to becoming the norm. By the year 2001, about 5% of the world's people will have Web access with penetration reaching half the population in the United States and Japan (Jupiter Communications, 1997). As

FIND/SVP, with a somewhat larger estimate than O'Reilly, uses a similar but less restrictive definition than O'Reilly, because its definition of an adult Internet user is an individual 18 years or over who currently uses at least one Internet application, in addition to email, and *includes* individuals who access the Internet through commercial online services.

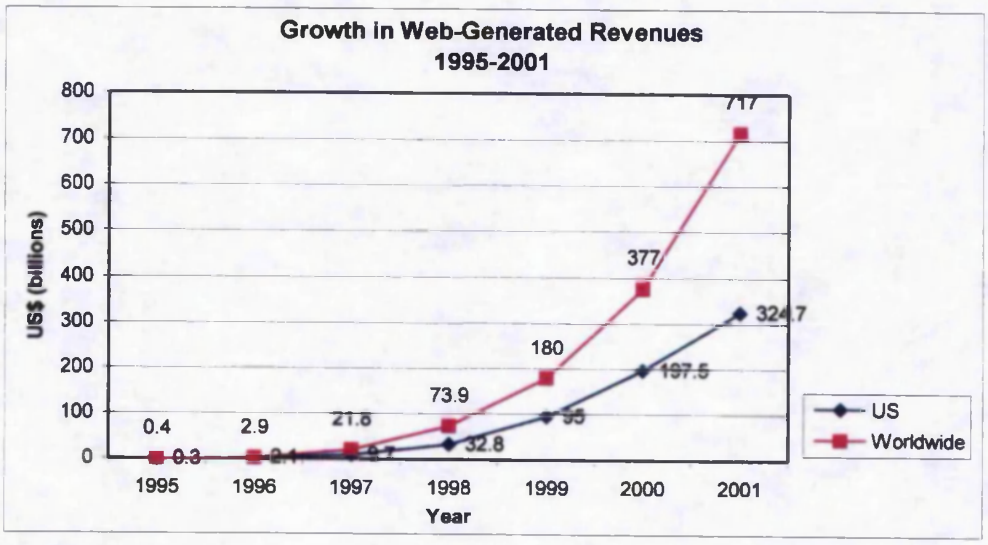
Neilsen/CommerceNet, with the largest estimate, uses the most liberal definition of online use, defining an 'online' user as an individual 18 or over who ever uses a computer from home, work, or school to connect with computer bulletin boards, commercial online services, or other computers over the Internet.

growth slackens in saturated areas, consumer demand should peak in Europe and Australia, and rise in China, India, the Middle East, South America and other later growth areas.

4.2.3 Web Revenues

Findings

In 1995, the Web accounted for US$436 million in sales, up from US$8 million in 1994, as shown in Figure 4.3. From then on, Web sites are well on their way to generating US$2.9 and US$21.8 billion in revenues[5] in 1996 and 1997 respectively (ActivMedia,Inc., 1997a).



Source: ActivMedia, Inc.

Figure 4.3 Growth in Web-Generated Revenues 1995-2001

[5] Revenues reported include 'all revenues generated by having an online presence, whether sales were closed online or not', the broadest measure of the impact of the Web.

Predictions

Even with conservative growth estimates, keeping average site revenues nearly flat, the influx of sites will build Web sales to more than US$700 billion by 2001. In the United States alone, Web sales are projected to account for more than US$300 billion in sales (ActivMedia,Inc., 1997a). A later research by Forrester Research (1998) predicted that online retail sales (closed online) in the United States will be worth US$184 billion by 2004, while the Internet will influence another US$500 billion worth of goods purchased offline.

4.2.4 Online Shoppers

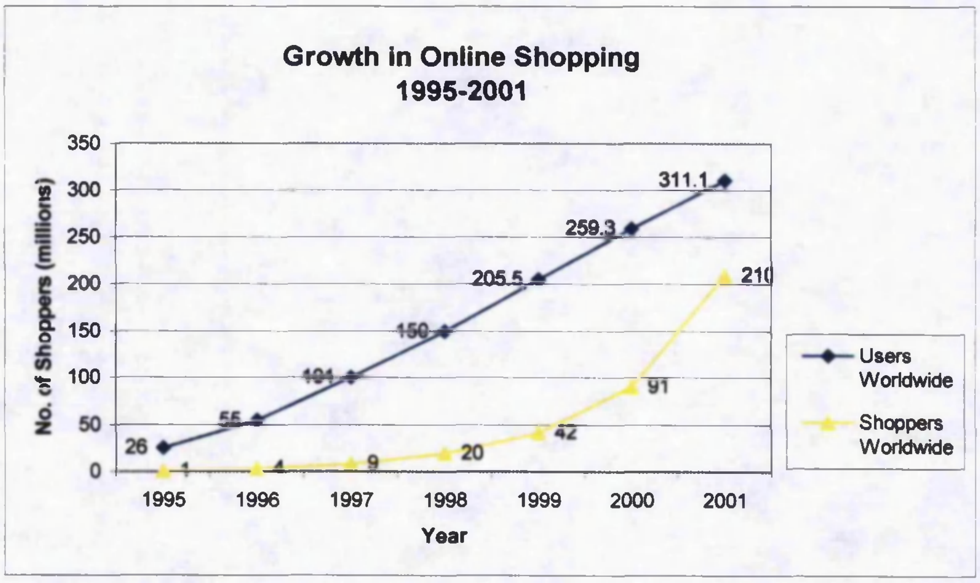
Findings

The number of online shoppers[6] is growing *modestly* from a mere one million in 1995 to nine million in 1997. Cyber-shopping threatens traditional merchant storefronts. Over 85% of online shoppers report a high degree of comparison shopping for features and prices among known products, and nearly 60% express willingness to purchase from any suitable vendor regardless of location (ActivMedia, Inc., 1997a).

However, the average online shopper-to-buyer conversion rate is still low - 2.7% (Intermarket Online, 1999). 62% of online merchants have a conversion rate of 2% or less and 5% reported rates in excess of 6%. Only one in five (20%) Internet users have actually made a purchase online but approximately two-thirds (64%) have used the Internet to research purchases.

The barriers most frequently cited by individuals were product pricing (77%), potential problems with returns (67%), concern about credit card security (65%) and personal privacy issues (58%) (Intermarket Online, 1999).

[6] ActivMedia's 1997 FutureScapes study was based on the results of interviews on 6000 Internet users about their buying behaviour. Online shoppers are defined as Internet users who have used the Web to purchase products or services online.



Source: ActivMedia, Inc.

Figure 4.4 Growth in Online Shopping 1995-2001

Predictions

Once secure transaction mechanisms are in place, it is predicted the online shoppers will rise dramatically to 210 million in 2001. The Web will continue to challenge traditional methods of retailing. According to ActivMedia (1997a), over 80% of the Internet users expect to shop online in place of mail order in the near future and 66% expect to shop more online and less in catalogues within two years. The longer people have been online, the more likely they are to shop there.

4.3 Survey on Electronic Commerce in Tourism

The area of travel has emerged as one of the fastest growing segments in terms of Internet business, sales and marketing activities. The survey in the following section uses a number of findings from ActivMedia (1997b). For details on sampling methods, see Appendix A. The sample used is as follows:

Region	Total
United States	71.7%
Europe	14.4%
Canada	8.1%
Australia	2.3%
Asia & Japan	1.6%
Central & South America	1.3%
Others	0.6%
Total No. of Sites	2811

Source: ActivMedia, Inc.

Table 4.2 Total Sample Breakdown by Regions

Region	Travel
United States	57.6%
Europe	17%
Canada	10.7%
Australia	1.8%
Asia & Japan	4.3%
Central & South America	6.1%
Others	2.5%
Total No. of Sites	164

Source: ActivMedia, Inc.

Table 4.3 Travel Sample Breakdown by Regions

4.3.1 Business Presence

The impact of electronic commerce on the travel industry can be seen by the large number of major companies that have already developed, or have begun developing, online sites. There are already large online self-booking systems providing GDS (Global Distribution System) interfaces which allow travellers to tap into the same computer systems that travel agents use to book flights, rental cars and hotels. Among the largest include Sabre's Travelocity, Microsoft Expedia, Preview Travel, Internet Travel Network (now renamed Getthere.com), Trip.com and BizTravel.com.

Independent travel agents' Web sites, such as Lowestfare.com and Cheaptickets.com, allow online purchase of air tickets and holidays. Hotel sites and centralised hotel booking systems like Pegasus' TravelWeb offer online reservations for hotel rooms. Auction sites such as Travelfacts.com and Priceline.com are also getting more popular.

As far as suppliers are concerned, there is little doubt about the rising trend. Hence, it is more important to understand the business status, composition and intentions of the suppliers. In general, there are a few major characteristics:

1. Travel sites have longer than usual Web experience (Table 4.4) and are getting more *profitable* (Table 4.5 and 4.6). A wide range of travel sites is now generating online revenues. Evidence of success and profitability of existing online suppliers will drive more suppliers online in the near future. According to ActivMedia (1995a), 71% of businesses with Web sites in the travel industry had some sales by December 1995, which is far higher than any other sector. Travel also had the highest mean sales per company (real estate was not taken into account). In a later survey, (ActivMedia, 1997b), out of the 164 travel sites studied, 50.8% (up from 33.9% in 1996) indicated that their sites are profitable, and 26.5% expected that their sites will be profitable in one to two year's time.

Months of Experience	**1997(%)**	**1996(%)**
<6	29.8	42.8
7-12	28.9	30.4
>12	**40.3**	23.2

Source: ActivMedia, Inc.

Table 4.4 Length of Web Site Experience

Status	**1997(%)**	**1996(%)**
Profitable from sales now	**50.8**	33.9
Profitable from sales in 12-24 months	26.5	35.7
Profitable from sales in 3-5 years	4.5	3.6
Reduce expenses	2.4	3.6
Generates useful PR	12.1	17.9
Disappointing in most respects	3.6	1.8

Source: ActivMedia, Inc.

Table 4.5 Travel Web Sites Profitability

Growth Rate	**1997(%)**
1-9%	**22.1**
10-24%	**22.1**
25-49%	5.8
50-99%	2.9
100-199%	2.9
>200%	1.9
0%	21.2
Negative	2.9

Source: ActivMedia, Inc.

Table 4.6 Revenue Growth from Month to Month

2. The new entrants on the market are characterised by *small establishments* (41.3%) with only 1-2 employees. The majority of these smaller suppliers are struggling to break the dominance of 'full-service mega-sites' (an emergent category just three years ago). These mega-sites are now responsible for most of the revenue being produced. As there is little incentive for large suppliers to increase market transparency and share their revenues with newcomers, the major problem of smaller suppliers is for their sites to be known by travellers.

No. of Employees	1997(%)	1996(%)
1-2	**41.3**	35.7
3-10	33.8	32.1
11-100	18.1	23.2
100+	6.2	5.4

Source: ActivMedia, Inc.

Table 4.7 Size of New Travel Web Sites

3. Though sales are not a dominant feature for most sites, there are obvious intentions from suppliers to use the Web to generate revenues directly. Online sites are increasingly *oriented towards actual 'transactions'* than they were in the past.

Purposes	1997 (%)
Publish info	77.7
Marketing	75
Direct online sale of products/services	**67**
Pre-sale support & purchasing info	58
Provide free links to third parties	38.4
Provide paid ads	31.3
Private customer/vendor relations	21.4
Collect information/online research	20.5
Post-sale customer service	14.3

Source: ActivMedia, Inc.

Table 4.8 Purposes of Travel Web Sites

4.3.2 Traveller Access

According to TIA (1999) [7], 6.7 million American adults or 9% of Internet users used the Internet or an online service to make travel reservations in 1998. Travel reservations include the actual booking/paying for an airline ticket, hotel room, rental

[7] The report is based on a telephone survey of 1,200 US adults conducted in September 1998.

car, package tour, etc. This is up from 5.4 million in 1997, which represents a 24% increase in online travel reservation volume in one year.

The Travel E-commerce Survey[8] (PhoCusWright, 1998b) showed that the online traveller population consisted of 35 million Americans or 18% of adult Americans. Of those 35 million online travellers, 80% looked, 58% checked prices, and 18% booked; of that 18%, 83% bought airline tickets, 40% reserved a hotel room, 32% rented a car, and 3% bought a vacation or tour. Of the 58% who checked prices, 70% did go on to buy that ticket later, such as from airline (28%) or travel agent (39%). The survey found that more online travellers use the Internet (68%) than use travel magazines or guidebooks to plan their trips.

Some major characteristics are identified:

1. The growth prospect is solid. Increasing competition among online travel sites will increase awareness and draw more consumers to the market. Business travellers - who travel regularly and at short notice - will be a key driver of growth. The expected take-up of interactive TV (iTV) will give a major boost to online traveller access because it is expected that package tour operators will take advantage of using this mass-market medium to communicate the value of a holiday to consumers through television programming.

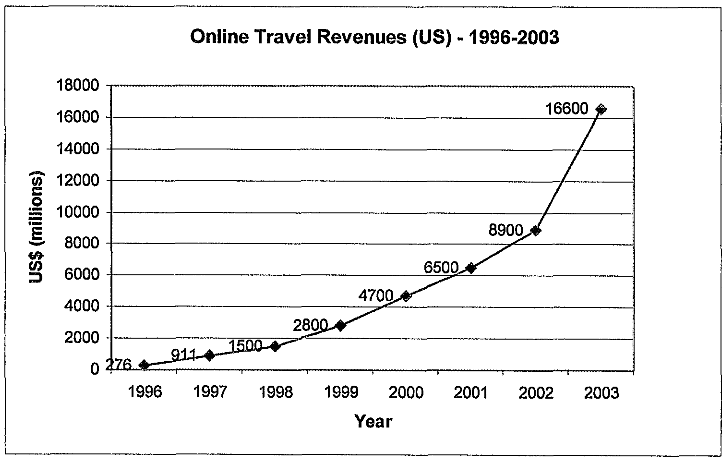
2. There is a large body of lookers and an increasing number of online bookers, though book-to-look ratio remains low. The majority of those who use the Internet to look at travel options do not purchase them online. Travel agents' ability to keep the differential between online and offline prices is still a key barrier to motivate online lookers to book in the short term (see section 4.3.5.5).

[8] The survey was conducted on 10,000 completed calls, 2,500 online travellers and 500 Random In-depth Interviews. An online traveller is an adult American who has travelled by air in the past year and visited a Web site in the past month.

3. Online purchasing behaviour is driven by experience and frequency of Web usage. Length of experience on the Web and frequency of Web usage are the best predictors of online buying. Currently, the ratio of online to traditional bookings is 1:5, but it is expected to rise to 4:5 in the United States and 1:10 in Europe in 2002 (Jupiter Communications, 1998b).

4.3.3 Travel Revenues

According to TIA (1998) and Jupiter Communications, travel is one of the high spending segments for the online consumer market. This is driven by a combination of a high price point and relative frequency of purchases. It is recorded that US$276 million was generated by online travel in 1996 in the United States, including travel products like air tickets, hotel, car rental, cruises, vacation packages as well as advertising earned by travel-oriented sites. Online spending for travel services increased to US$911 million by the end of 1997. Travel purchases are predicted to grow dramatically in the next few years, making this product segment likely to remain in the top list for online consumer spending through the year 2003. The report projected an increase of 87% in online travel sales in 1999, to US$2.8 billion or 4.2% of the market, and then by another 68% in the year 2000, to US$4.7 billion or 6.8% of the market. From then on, online travel sales will continue to grow to reach the US$8.9 billion mark in 2002 and US$16.6 billion in 2003 (Jupiter Communications, 1999a), shown in Figure 4.5.



Source: Travel and Interactive Technology Report of the TIA/Jupiter Communications

Figure 4.5 Online Travel Revenues 1996-2003

Datamonitor (1999) also acknowledged that the travel industry will be one of the fastest growing sectors for online sales in Europe over the next few years. In 1997, online sales of travel products in Europe were worth just US$7.7 million. By 2002, they will be worth US$1.7 billion. The two major European markets are United Kingdom and Germany.

The following major market characteristics were extracted:

1. The field of major players in the online travel market is consolidating quickly, as rapid growth in traffic to the large full-service mega-sites combined with high operating costs created by that traffic make the top tier a place for only the 'deep-pocketed'. Online travel revenues are currently *dominated by six large travel Web sites*, including Travelocity, Expedia, Preview Travel, Internet Travel Network, generating 75% of the Internet revenue in the travel industry. Out of the total US$276 million Web revenues in 1996, US$217 million, or 79%, went to mega-

sites, and US$56.6 million was earned by supplier sites (Jupiter Communications, 1998b). This situation is likely to change slightly as new sites come online and existing sites redouble their marketing efforts, but the bulk of revenues will continue to be dominated by a handful of sites in the next couple of years.

2. Online travel revenues were *dominated by air travel sales* in 1996, and will continue to be dominated by them in the near future, as consumers are more comfortable purchasing air tickets online because the product segment has long competed on price and is considered more as a commodity by consumers. Air travel currently generates over 80% of online travel revenue. However, though air travel sales will increase significantly in dollar terms over the next four years, it will drop by 20% in percentage terms as the number of consumers making online car hire and hotel reservations online increases (Jupiter Communications, 1999a). It is predicted that car and hotel reservations will rise from 11.3% in 1996 to 20.4% in the year 2000.

3. Though commodity items like air tickets, road and rail transport, are currently attracting the most interest, in the long term, *customised package holidays and accommodation offer high potential*. These premium products typically attract a higher margin, making them an attractive proposition for online distribution - as tour operators can make large savings from cutting costs in distribution, and agents can gain higher commissions than by selling commodity items. However, these premium travel products are most often highly information intensive. In addition to basic details such as time, destination and price, customers require additional information such as destination information and details of accommodation facilities. More advanced technologies are required to assist the consumer to make a complex sequence of buying decisions.

4.3.4 Favourable Factors

The surveys show a number of phenomena: (1) There is a thriving Internet community in the travel industry (both customers and suppliers); (2) There is an increasing tendency towards online distribution in travel; and (3) Travellers are becoming more experienced and want more customised products. Indeed, there exist some favourable factors that will enhance the growth trend further.

4.3.4.1 Qualities of Travel Product

The fundamental qualities of the travel product listed in the table below make it a good candidate to succeed in the online market.

Qualities	Products
An appeal to the demographic	PCs, software, consumer electronics, **travel**
A considered/research-intensive purchase	Cars, **vacations**, homes
A known/understood item	Books, CDs, videos, **air tickets**

Source: Online Shopping Report, Jupiter Communications 1998

Table 4.9 Major Qualities of Successful Online Product Categories

Tourism, as an industry, is fragmented and heterogeneous. Travel suppliers and intermediaries represent a worldwide network, and hence production and distribution is based on co-operative and distributed processes. The Internet has totally transformed the playing field of travel because it enables access by the whole world to the whole world. There is no longer a 'domestic' market. Travellers are no longer restricted by the geographical locations of suppliers and they can virtually access any suppliers from anywhere of the world. Unlike traditional media, where advertising placement almost exclusively targets viewers/readers of individual countries, the national boundaries on the Internet are much less defined. Data from the online advertising report of Jupiter Communications show that significant numbers of visitors from European and Asia/Pacific Rim are coming to US sites, despite relatively low usage/penetration in many countries of these regions. In fact, several content sites surveys have reported non-US traffic accounts for 30 to 50% of the overall (Jupiter Communications, 1998a).

Secondly, the travel product, e.g., a vacation package, is both complex, consisting of a set of very different products, and perishable. Hence, the availability of information is especially important because: (1) consumers are located far away from the product or service of purchase; (2) travel involves the use of discretionary money and free time, and is a high-risk purchase, and (3) the intangible nature of travel products suggests that secondary or tertiary sources must be used, as a consumer is not able to actually observe the potential purchase. Motivations for information search are, therefore, risk-related (Crompton & Ankomah, 1993). The rich and diversified content of the Internet fills the need for information search (to reduce risk) as well as product and price comparisons at a reasonable cost.

Lastly, commodity products, such as air tickets, are presently the most successful online category because they are well-understood items. Airlines were the first to establish an online presence, but more importantly, air tickets are highly suitable to online selling. In the commodity travel sector, price differentiation is all-important, and by using the low cost Internet distribution channel, airlines can deliver lower-priced flights.

4.3.4.2 'Self-Serviced' Tourists

The market for travel services is becoming highly partitioned into single segments that can be assembled into highly individual products. Bundling separate services into a personal product is made possible through the so-called 'atomisation' of the travel market (Tschanz & Zimmermann, 1996). Coupled with this trend is the continuous shift from mass to *individual* tourism. Tourists are becoming more experienced and sophisticated. They want customised products, global advice, service quality, market transparency, and have a certain degree of self-service mentality. This new breed of tourists are usually less price-sensitive, but more information-oriented, and they demand highly *personalised* travel arrangements. They welcome the convenience of accessing online systems to locate a wide range of sources of information to prepare for a trip. Therefore, the so-called 'do-it-yourself' package is getting ever more popular.

On the other hand, consumer expectations have risen faster than the traditional travel agent has been able to raise standards. Consumer satisfaction with traditional travel agents has fallen substantially over the last five years (Equinus, 1998). This is caused by a serious shortage of expertise and the lack of training, resulting in travel advice with inferior quality. There are clear signs that more travellers will be driven to seek information online as traditional travel agents can no longer deliver added-value services.

4.3.4.3 Minimising Distribution Costs

What is the reason for the push by travel suppliers to reach out to customers directly? The rising cost of distribution is probably the main reason. Commissions paid to travel agents are the airlines' third biggest expense, after labour and jet fuel. According to the International Air Transport Association (IATA) (1995), commissions accounted for US$1.55 billion, or close to 10% of domestic carriers' total revenues in the first quarter of 1995. IATA estimated that the cost of distribution was approximately 25% of the airline industry's global operating costs. In Europe, it was estimated that distribution costs were 22% of British Airways' revenues, of which 8.5% were agents' commissions.

The two most important benefits that the Internet offers to the travel industry are the fact that all the information is potentially available worldwide to a huge target audience, and secondly that as a distribution channel, it is very cheap. Web sites can be set up and administered for a cost that pales into insignificance compared to the overheads involved in establishing, say a call centre, manned by reservation staff.

Each reservation made online works out far cheaper than the same reservation made through a call centre. When considering all associated overheads, the transaction costs of bookings made online can be up to ten times lower than those made through call centres (Datamonitor, 1999). For example, after three years' online experience, United Airlines has noted the 'significant reductions in channel costs' made possible by electronic commerce. While the cost of booking a typical US$300 ticket through a travel agent or an airline employee represents 17% and 12% respectively, an electronic booking amounts to just 6%. In 1997, United Airlines saved US$2.6 million on customers looking for flight availability information, US$461,000 on arrival and departure information, and US$686,000 on general information by shifting customers off the telephone (Rosen, 1999).

Apparently, airlines are trying new ways to go direct to the customers to save distribution costs. As money is always a powerful drive in business, travel suppliers across the whole spectrum of the industry will follow to stay competitive in the industry. Such a trend of direct marketing and distribution is gathering momentum.

4.3.5 Key Obstacles

Though the strong underlying forces are pushing the electronic travel market to move forward, major obstacles like high user involvement, lack of intelligence support, narrow bandwidth, security concerns and travel agents' ability to keep price differential are pulling it back. The table below shows some of the major reasons for not buying travel online (PhoCusWright, 1998b).

Reasons	%
Security	64
Travel Agents have better offers	48
Afraid of missing best price	36
It is work that someone else should do	35
Do not have enough knowledge	34
Afraid of making mistakes	28
Time-consuming	26

Source: PhoCusWright, 1998

Table 4.10 Reasons for Not Buying Travel Online

4.3.5.1 High User Involvement

The sheer difficulty of finding the right information has put many customers off. As indicated by the stated reasons in Table 4.10, travellers regard this as work that someone else should do (35%), and regard the task as too time consuming (26%). At present the information necessary to plan a comprehensive trip tends to be distributed across the Internet on the Web sites of individual companies. A few years ago, it was not too difficult to find flight schedules, rail timetables, and hotel availability information on the Web. However as the number of travel Web sites available online continues to grow, so does the problem of finding and integrating the available information to satisfy a traveller's personal requirements and constraints. Gathering and integrating information has become too laborious and time-consuming for an average human.

Currently, search engines are the most popular methods that people use to look for product information on the Internet. However, the disadvantages of using search engines are getting more apparent with time. Keyword searches are never precise enough to match the specific needs of tourists and no one wants to spend hours sifting through hundreds of records and still end up without a single good match.

Even assuming the results of search engines are satisfactory, accessing online information still requires *time*, *patience* and *perseverance*. Increasing response time and high network complexity result in serious disadvantages for the online bookers. Individual and business travellers have core competencies that demand more and more focus. They cannot afford to allocate time and energy away from their more important daily activities to navigate the massive networks and search for relevant information. Though a booking transaction can now be automated to a great extent, drifting through site after site in search of the needed information is arduous and boring, and there is nothing inherently creative in the process. Moreover, access to information via a centralised index can be both cumbersome and slow. For example, there is no simple way for a traveller to access the fare information in an efficient way. When accessing online flight data, most online systems cannot quote the fare until after the departure and return dates, departure and return times and both airports have been selected by the user. It takes an average of ten minutes of hard hunting to get out data about all the fares between two points.

4.3.5.2 Lack of Intelligence Support

Clearly, a lot of customers are getting online more frequently, but actual buying is not yet commonplace and is restricted mostly to air tickets which are seen more as commodity items. Why? Lack of intelligence support is one of the major reasons that hold back customers. As seen from Table 4.10, non-buyers are concerned about the fact that they do not know enough (34%), may miss the best price (36%), and are afraid of making mistakes (28%). Selecting a tourism product involves complex trade-offs and risks due to the intangible nature and the wide range of available alternatives. Even for a straightforward purchase like an air ticket, the wide variety of services on offer will mean that making the best choice will involve balancing factors such as price, availability, brand, quality, locality, etc. before committing to any particular vendor or service provider. It is well proven in cognitive psychology that there is a finite limit to the capacity of the human brain to process information relating to a large number of alternatives (Crompton et al., 1993). Therefore, when it comes to complex purchases like integrating several travel products into a package, travellers are easily confused and overwhelmed. In addition to basic details such as time, destination and price, extra information such as destination information and details of accommodation facilities are needed. Furthermore, the traveller must make a complex sequence of buying decisions based on numerous inter-related multi-attribute constraints. This is where and why users need intelligence support most dearly.

4.3.5.3 Narrow Bandwidth

The growth in Web users has outpaced the increase in bandwidth. It results in busy traffic and low speed of access to information. To make matters worse, many online travel systems are heavily loaded with graphics, such as Travelocity, Expedia, etc. The original intention of enhancing their site appeal may end up pushing consumers away. As it takes a long time to download every single page embedded with large graphics and images, it is simply impossible for the busy or impatient customers to get past the registration screen. The bottleneck here is: as long as Web sites are designed for human use, it is natural for them to include graphics to look attractive to the human eye. Until next-generation 'super-capacity' broadband carriers are introduced, limited bandwidth will continue to make downloading Web content slow and frustrating.

4.3.5.4 Security

As with any business done electronically, security is a major concern for both individual and travel companies, that worry about computer hackers getting into their inventory of airline seats, hotel rooms and rental cars. Companies are also concerned about liability if a hacker gets travellers' credit card numbers or worse, their itineraries, which could be used to plan burglaries.

To be fair, the issue of security has been exaggerated out of all proportion in the last few years. It is the public's 'perception' of insecurity that hampers, or holds back, feelings of security on the net, a feeling that the public do not feel when giving their credit card numbers to someone over the phone whom they do not even know. Changing that public perception may be difficult until standards are published and well in place.

However, electronic commerce demands higher levels of security than simple information retrieval because money is involved. It is delightful to see that many of these problems are already dealt with at lower network levels, such as through SSL (Secure Socket Layer) and the SET (Secure Electronic Transaction) standards. Many suppliers are building firewalls to protect their information through authentication, encryption and secured servers. Numerous electronic institutions, e.g., VeriSign and Thawte[9] are offering services like digital signatures and certificates to increase customer confidence. These digital certificates enable secure online communications and transactions, privacy, and authentication to enterprises, Web sites, and consumers.

4.3.5.5 Travel Agents' 'Price Differential'

The results of the Travel E-commerce Survey (PhoCusWright, 1998b) show that the online 'booking-to-looking' ratio is still low. Out of the 35 million online travellers, 80% looked, 58% checked prices, and only 18% actually booked. Of the 58% who checked prices, 70% did go on to buy that ticket later, such as from airline (28%) or travel agent (39%).

These figures are partly a result of the discrepancies between online and traditional travel agent prices[10]. As long as the travel agents are still enjoying the advantages of keeping the price differential and buying power, it will be cheaper for the customers to book with them. Though, in the near term, price differences will eventually be eroded away as online suppliers reform their pricing strategies to encourage more online

[9] VeriSign: http://www.verisign.com; Thawte: http://www.thawte.com.

[10] Though 64% of respondents (online lookers but offline buyers) stated security as the major reason for not buying online, this fear is expected to disappear with increase in Web experience.

bookings, it may still be cheaper for the traveller to look and check prices online and book with an offline agent at the moment.

4.4 The Next Stage?

'The Web appears to provide what PC users have always wanted, i.e., the capability to point, click, and get what they want no matter where it is. Whereas earlier manifestations of the information revolution bypassed many people who were uncomfortable with computing technology, it appears that the Web is now attracting a large cross section of people, making the universality of information infrastructure a more realistic prospect.' (National Research Council, 1994)

However, far from the Web's promise to take over as a powerful tool for electronic commerce in tourism, the current structure has led to the disorientation of both travellers and suppliers. Therefore, something is missing here to remove the obstacles and take advantage of the favourable market climate.

Many researchers (Linden et al., 1997) started to advocate the idea of a personal travel assistant that arranges trip itineraries on behalf of its users by integrating distributed travel services. Travellers will be able to *delegate* to agents the responsibility of accessing many service providers in just the same way they now access search engines, select and purchase travel products and services. The promise is to provide *personalised* journey information and assistance to potential travellers by gathering information from the WWW according to the traveller's own personal preferences.

It has been claimed that agent technology may be the only way that customers will be able to cope with the complexity and sheer volume of future commercial offerings on the Internet. Now the Internet is here, agents are available, so why does such an undoubtedly useful personalised travel service not yet exist?

4.5 References

ActivMedia Incorporation, (1995a), *FutureScapes study,* Peterborough, NH, US, October, 1995, http://www.activmedia.com.

ActivMedia Incorporation, (1995b), *Executive Report,* Peterborough, NH, US, December, 1995, http://www.activmedia.com.

ActivMedia Incorporation, (1997a), *FutureScapes study,* Peterborough, NH, US, October, 1997, http://www.activmedia.com.

ActivMedia Incorporation, (1997b), *The Real Numbers behind 'Net Profits',* Peterborough, NH, US, 1997, http://www.activmedia.com.

BizTravel.com, http://www.biztravel.com.

Cheaptickets, http://cheaptickets.com.

Crompton, J. L. and Ankomah, P. K., (1993), Choice Set Propositions in Destination Decisions, *Annals of Tourism Research,* 20, pp. 461-475.

Datamonitor, (1999), *Interactive Services Strategic Planning Programme,* 24 May, 1999, http://www.datamonitor.com.

Equinus, (1998), *Consumer Web Survey,* http://www.equinus.com.

Expedia, http://www.expedia.com.

FIND/SVP, (1996), *The American Internet User Survey: New Survey Highlights,* http://www.findsvp.com.

Forrester Research, Inc., (1998), *Resizing Online Business Trade,* Forrester Research, November, 1998, http://www.forrester.com.

Guilfoyle, C., Jeffcoate, J. and Stark, H. (1997), *Agents on the Web: Catalyst for E-commerce,* Ovum, Inc., April, 1997, http://www.ovum.com.

Guttman, R. H., Moukas, A. G. and Maes, P., (1998), Agent-Mediated Electronic Commerce: A Survey, *Knowledge Engineering Review,* 13(3), June 1998.

Intermarket Online, (1999), *The Internet Commerce Briefing,* August, 1999, http://www.intermarketgroup.com.

International Air of Transport Association, (1995), *Press Release,* November, 1995, http://www.iata.org.

Internet Travel Network, http://www.itn.net.

Jupiter Communications, (1997), *Consumer Internet Report,* July, 1997, http://www.jup.com.

Jupiter Communications, (1998a), *Online Shopping Report,* 1998, http://www.jup.com.

Jupiter Communications, (1998b), *Online Travel – Five Years Outlook,* 1998, http://www.jup.com.

Jupiter Communications, (1999a), *Travel Suppliers Missing Online Market Potential,* May 17, 1999, http://www.jup.com.

LowestFare, http://www.lowestfare.com.

National Research Council, (1994), *Realising the Information Future - The Internet and Beyond,* Washington D. C., US, http://www.nap.edu/nap/online/rtif.

NUA, Inc., (1997-1999), *Internet Surveys,* http://www.nua.ie.

Nwana, H. S., (1998), Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints, *Proceedings of Agents '98,* Minneapolis, May 1998, pp. 189-196.

PhoCusWright, (1998b), *Travel E-commerce Survey,* November, 1998, http://www.phocuswright.com.

Preview Travel, http://www.previewtravel.com.

Priceline.com, http://www.priceline.com.

Rosen, C, (1999), *Online Sales Heat Market,* http://www.btnonline.com.

Travel Industry Association of America, (1998), *Travel and Interactive Technology Report,* February, 1998, http://www.tia.org.

Travel Industry Association of America, (1999), *1998 Technology and Travel Survey,* January 1999, http://www.tia.org.

Travelfacts, http://www.travelfacts.com.

Travelocity, http://www.travelocity.com.

TravelWeb, http://www.travelweb.com.

Trip.com, http://www.trip.com.

Tschanz, N. and Zimmermann, H., (1996), The Electronic Mall Bodensee as Platform for the Development of Travel Services, in Klein, S. et al. (eds.) *Information and Communications Technologies in Tourism, Proceedings of ENTER 96 International Conference in Innsbruck, Austria, 17-19 January, 1996,* Springer-Verlag Wien, New York, pp. 200-210.

Chapter Five

Agents' Potential in Electronic Commerce in Tourism

The aim of this chapter is to identify the goals and challenges for agents in the context of electronic commerce in tourism. It starts with a discussion on the advantages of agents in this domain. It then proceeds to investigate the roles that agents may play and the opportunities that they create with reference to the different stages of the travel decision-making model. Then, the suitability of a multi-agent approach in the travel industry as well as the key barriers involved in the Personal Travel Assistant (PTA) system are investigated. The slow uptake of agent technologies clearly shows that current fledging attempts to deliver agent-based commerce have avoided these problematic issues in favour of short-lived successes from restricted ad hoc systems that work well in limited domains.

To unleash the full potential of agents, it is necessary to focus the design of the prototype to suit the user market. In analysing the user requirements, the design philosophy of PTA is set. The proposed multi-agent collective learning system appears to be a very logical and realistic approach to facilitate the realisation of agents' promise.

5.1 Agents Come into Play

If the Internet is to become a useful, powerful and open environment that facilitates electronic commerce, some sort of 'oil' is needed to make the whole process of commercial activities run smoothly. One thing for sure is that this process will be enhanced and catalysed by various new enabling technologies. Intelligent agents will certainly play an important role, though they will not be the only actors.

5.1.1 'Intelligence' is Key

The developments on and around the Internet are bearing a strong resemblance to the development of computers and their interfaces. In the very beginning, computers were hardly user-friendly, they were command-line-driven and had no form of online help whatsoever. Slowly this changed when the first help functions were added. One of the most important changes has been the introduction of the *Graphical User Interface* (GUI), which enabled a much more abstract view on the operation of a computer. The popularity of computers, particularly that of home computers or PCs, is largely due to the introduction and further developments of the GUI.

The Internet developments have followed this pattern in many ways. At first there were not many people using it, and most of them were highly educated users who were well capable of working on it without much support or nice interfaces. With the introduction of the Internet's own 'graphical user interface' - the WWW in combination with its graphical browsers - this changed drastically. From that moment, even novice users were able to use the various Internet services without having to know how each individual service should be used.

After the introduction of GUIs on computers followed a massive production of all kinds of applications and programs, most of which exploited GUI capabilities as much as possible. The same is bound to happen on the Internet too. The major difference between these applications and the applications that have been written for PCs and the like, is that the former will have to be more *flexible* and *robust.* In other words, they will have to be more ***intelligent*** to be able to function properly in the dynamic and uncertain environment the Internet is known to be.

5.1.2 Agents vs Expert Systems

The dynamic and distributed nature of the Internet requires that software not merely respond to requests for information but intelligently anticipate, adapt, and actively seek ways to support users. If 'intelligence' is the key element for the next-generation software on the Internet, there are two eligible candidates to lead this role – expert systems and agents. The major difference between an agent and an expert system is that the former is characterised by its intelligence and the latter by its knowledge. It is worth mentioning that agents do not equate with 'possessing high level of intelligence'. In fact, a 'young' agent may be less smart than an ES. It is the behaviour to act intelligently that makes agents distinctive, but it does not mean that they have a high level of competence to start with.

Expert systems (ES) use a representation of human expertise in a specialist domain in order to perform functions similar to those normally performed by a human expert in that domain. ES make decisions based on facts and rules that reside in a knowledge base which is a collection of information gathered through a series of interviews with an expert or groups of experts in a given field. The systems capture the knowledge and actions that experts use and make this knowledge available to everyone via a computer. Thus, it is possible for non-experts to gain assistance from a specialist in a particular field without having the specialist around. In actual fact, it is the use of this heuristic *knowledge* that differentiates ES from all other programs.

It is best to illustrate the difference between 'knowledge' and 'intelligence' by looking at the relationships of data, information, knowledge and intelligence.

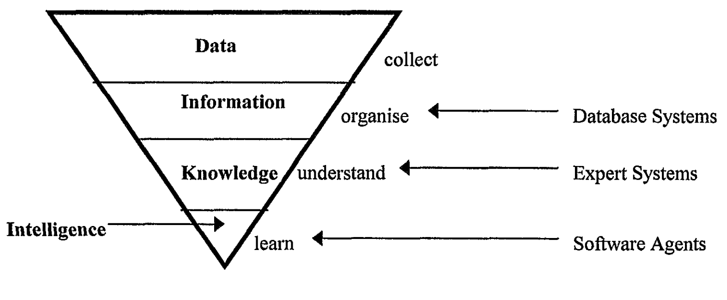


Figure 5.1 Relationships of Data, Information, Knowledge, Intelligence

Data is made up of facts. Information is a number of facts processed in a meaningful way. The difference between knowledge and information is that information is data organised in a meaningful manner while knowledge requires an acquaintance with and understanding of the information. Hence, knowledge includes facts, beliefs, and rules. Intelligence is the ability to adapt old rules and acquire new rules, to augment the knowledge through learning (Gilbert et al., 1995).

Though ES and agents may look similar superficially, they have quite different properties and traits and hence are suited to different application areas. One point to note here is that agents are not necessarily better than ES. There are certain applications that agents will excel, while others may be more suitable to be solved by ES.

Expert Systems	Intelligent Agents
Act on user commands	Act autonomously
Generalisation across users	Personalisation to individuals
Well-defined problems	Fuzzy problems
Fixed knowledge base	Evolving knowledge base through learning

Table 5.1 A Comparison between Expert Systems and Intelligent Agents

Expert systems are passive. They take commands from users and the job is most often a one-shot computation. In contrast, intelligent agents are autonomous and situated in the software environments. User involvement can be reduced to a minimum because delegation to agents is possible without direct user manipulation.

Agents are more suitable to serve in a constantly changing environment. They have the ability to learn individual needs and preferences to build a user model. Very often, different users want assistance with different parts of the task and in a different fashion. Even the same user may need a different kind of help at different times. Every case is unique. A system that can be of any use in a dynamic situation like travel has to be very flexible, both in the problems it covers, and in its approach to those problems. To meet the demands of these users, it is more appropriate to use an intelligent assistant than an omniscient expert.

Expert systems, by nature, are systems that can produce perfect answers to well-formatted questions. They are intended for situations where the problem is well-defined and the difficulty is to find an answer to this problem. For this approach, correctness and efficiency are the important issues. To ensure efficient and correct answers, it might be necessary to restrict the system to solving only those problems that can be solved exactly, while giving no answer in other cases, hence endangering the robustness of the system. Intelligent agents are able to approach less structured tasks - problems that are difficult to specify clearly enough for a computer to solve. Hence, a flexible agent, one that can give reasonable answers and comments within a wide range of problems, may be more valuable than one that produces correct answers, but only to a limited number of questions.

Moreover, an expert system approach requires a huge amount of work from the knowledge engineer. He/she has to endow a system with a lot of domain-specific background knowledge about its application and about the users, and little of this knowledge or the system's control architecture can be reused when building systems for other applications. A further problem is that the major part of the knowledge base is fixed. It is possibly incorrect, incomplete, becomes obsolete overtime, and cannot be easily adapted or customised to individual user differences, or to the changing habits of one user. In other words, expert systems have to be reprogrammed if significant changes have to be made on the knowledge base. In an electronic commerce arena that changes immensely from day to day, hour to hour, in such random fashion, situations not covered by the rules will always arise. Decisions are never easy when experience with similar situations is lacking. Agents can produce useful assistance with significant reduced manual development and maintenance of the knowledge base. They are able to capture useful training data, and generalise from such data to learn a customised knowledge base competitive to hand-coded knowledge.

5.1.3 Advantages of Agents in E-commerce

When looking at the current state of affairs on Internet commerce, the need to devise new methods to *delegate* shopping activities is getting stronger and stronger (Nwana, 1998). *Agents* are meant to fill this need precisely.

Market Trends	Customer Needs	Agent Capability
Rapid growth of content	Reduce user involvement	Delegation
Mass customisation	Individual personalisation	Intelligence
Insufficient bandwidth	Speed	Bandwidth saving

Table 5.2 Agents Meeting Customers' Needs

5.1.3.1 Reduce Workload

The biggest advantage that agents bring is simply their ability to automate previously manual operations. The key here is *delegation* with minimal human intervention. It seems simple in principle, but it makes an enormous impact on commerce, because it removes friction from the commercial environment, making more information available more widely. This is an issue of *scale*, of *access*, and of *information*. Agents can visit more sites, gather more information, recommend Web sites that would have escaped tourists' attention, or simply because tourists would not have had the time to explore. The number of things an agent can realistically do can be orders of magnitude beyond what a human can realistically do, and this can have a major impact, even when the agents themselves are relatively unsophisticated.

5.1.3.2 Learn to Personalise

Sophisticated agents bring another layer of capabilities to the tasks they carry out. The ability to do *automated negotiation*, automated *planning* and to *learn* are areas where sophisticated agents can leverage their strength in electronic commerce applications. Their learning abilities help in cloning users' habits and preferences, enabling them to offer intelligence support with a personal touch that will benefit users in decision-making. Agents make it possible for the users to get good, sound, and creative advice with the disguise of a user-friendly interface. At an advanced level, agents *interact with other agents* and come to viable agreements, plan for information gathering, and learn about the make-up of their user's preferences and the items available at specific sites.

5.1.3.3 Save Bandwidth

As mentioned before in the previous chapter, graphics and images are put onto Web pages to please the human eye. A lot of bandwidth is wasted in transferring pages loaded with huge graphics which is non-functional in nature as far as information is concerned. Agents are machines that read bits and bytes, so they do not react to sight stimulation. Sending agents to work on the Web has the advantage of using *minimal bandwidth* as agents transfer messages which are far smaller in size than Web pages. More importantly, only filtered materials, not graphics, will be retrieved by these agents. An extra advantage is that they reside in the computer or network 24 hours a day, seven days a week, and they never stop working. Efficiency is achieved because they can execute tasks outside peak hours and *spread the traffic load* on the Internet more evenly.

5.1.4 Potential Roles of Agents in Travel E-Commerce

From a travel decision-making perspective, the numerous roles that agents may play in electronic commerce in tourism can be identified. There are several descriptive theories and models that attempt to capture travel buying behaviour (Moutinho, 1987; Middleton, 1988; Mansfeld, 1992). Though these models use different terms to name and dissect the various stages, they all share a similar list of six fundamental stages. By adapting these traditional models into electronic commerce settings, one can identify at least the following stages: (1) Travel Motivation; (2) Product Search; (3) Alternative Evaluation; (4) Negotiation; (5) Purchase; and (6) Post-Trip Evaluation.

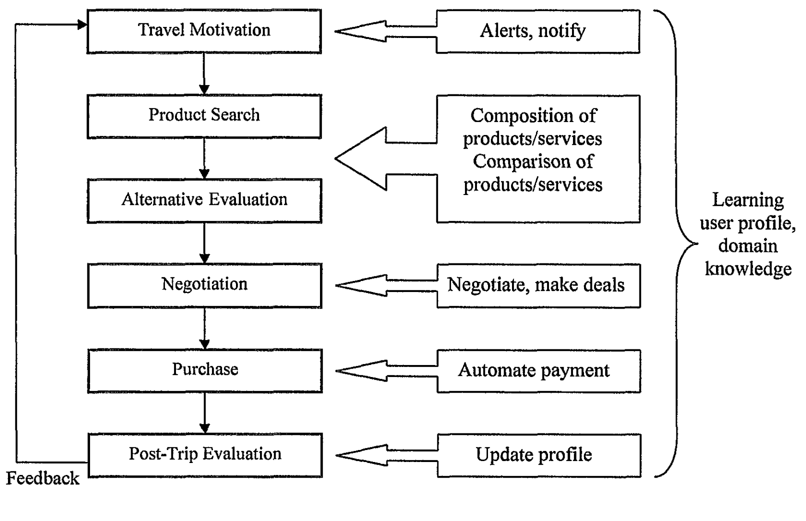


Figure 5.2 Travel Decision-Making Model for Electronic Commerce

It is perceived that agents will play serious roles in these different phases:

Travel Motivation: This is the situation where the traveller becomes aware of some unmet need. Within this stage, the traveller can be stimulated through product information. A PTA's role is to notify. Essentially, the PTA keeps a personal profile of the traveller, identifies the current need, and selectively chooses which travel 'advertisements' the user should read. Supplier agents may keep profiles of potential customers, and tailor their advertisements to the customers.

Product Search & Alternatives Evaluation: These stages are where agents have classically been deployed in electronic commerce. They comprise the core decision-making process to help determine *what* to buy and *who* to buy from. The PTA will use its owner's profile and requirements, such as the destinations to be visited and the duration of the stay, to automatically locate, select and interact with the appropriate Internet information sources and services. These phases encompass the searching and gathering of alternatives based on the needs and preferences (criteria) specified by the travellers. The process involves identifying appropriate suppliers, comparing their products, etc. which typically requires some reasoning and planning.

Travel is a 'super-transaction' consisting of many components. When the PTA plans a journey, it is likely to consist of several legs, from the traveller's home to his/her desired destination. Therefore, a transaction cannot be based or concluded only for flight arrangements, because hotel, car, and many personal arrangements must also be established. For example, travellers usually do not want to just buy a single product. They have preferences over bundles of products, e.g., air tickets, hotel rooms, etc. most of the time. At the same time they have a limited budget. So, agents will help the tourists efficiently strike the trade-offs between different travel products in a bundle, and price. The items of a bundle are usually purchased from different suppliers. This process necessitates reasoning, planning and constraint satisfaction capabilities in order to deal with timing and other dependencies in the travel itinerary while attempting to satisfy the traveller's preferences. In essence, the PTA needs to co-ordinate the services of the different service providers so that as a whole, the service providers behave coherently in their attempt to provide an integrated service. For instance, the PTA should be able to reason that the flight and hotel accommodation should be booked prior to considering local transportation, and ensure that a taxi is booked to coincide with the arrival of train.

Negotiation: The Negotiation stage is comparable to the Choice/Decision in traditional models. Traditional models do not identify this stage explicitly because prices and other aspects of travel products are often fixed in the offline market, leaving no room for negotiation. However, negotiation will be a key component of electronic commerce. Fixed menu offers - currently prevalent in physical commerce – are likely to give way to agent-mediated negotiation. There are several reasons for this. While haggling is not the norm in real commerce due to the time that would be wasted, in electronic commerce, software agents can do the negotiation. Getting a particular price from a supplier agent does not imply that other agents will request the same. This is because the transaction can be kept out of the view of others.

Purchase: Once the user is satisfied with the itinerary, the PTA can be authorised to make the appropriate travel and hotel bookings, which can be paid for electronically over the Internet using the user's credit card.

Post-Trip Evaluation: This stage involves an evaluation of the satisfaction of the overall trip experience which may be used as feedback to guide future purchases. PTA will adapt its knowledge base to include the feedback from the traveller. This is considered as part of the learning process.

5.2 Adoption of a Multi-Agent Approach

Having studied the potential of agents in electronic commerce and the advantages they may bring to both travellers and suppliers, it is still necessary to determine the best application approach to the problem.

5.2.1 Agents in Proprietary Systems

Already, more and more 'single agent' applications are launched onto the electronic commerce market and the term 'software agents' is used widely. There are applications of 'shopping' agents in large search engines such as Excite's Jango (1996) and Firefly (1999). Similarly, the same rationale can be applied to tourism.

For example, full-service mega-sites, such as Travelocity, Expedia, Preview Travel, etc. emerged as new players in the industry a few years ago. In an open market with a large number of available services, the need for intermediary services will inevitably increase. These mega-sites aim to serve the travellers by converging travel products and services as well as destination information at one single point. These new forms of intermediary service offer one-stop-shopping to the travellers, in other words, try to make the search for travel information on the Web more manageable to the travellers. It is by all means possible to insert an agent service in any of these mega-sites because there is no need for standards/interoperability as information is all stored on the same site. However, this also implies that these sites are basically <u>closed</u> markets. Hence, there are several important disadvantages:

- The ability to make decisions is still restricted by the depth and breath of travel product information on these mega-sites. How can one be sure that the scope of coverage of these sites are representative enough?
- Another drawback is that these mega-site agents belong to the suppliers, not the customers. No matter how intelligent they appear, their ability of personalisation is highly limited because they are catering for the general interests of a large group of users instead of establishing a one-to-one relationship with each user.
- One last but equally important disadvantage is the lack of trust on the side of the consumers. Can they possibly trust the suppliers' agents totally when they do not own or 'know' these agents in the first place?

5.2.2 Agents in an Open System

To go one step further, all the travel sites can be merged into one single open online market (Hopken, 1999). This idea has obvious advantages over the mega-sites because searches are no longer localised and can be done with the widest Web coverage. Apart from the arguable idealistic assumptions[1] of this 'top-down' standardisation approach, there are also some fundamental technical hurdles that need to be crossed.

Clearly, a universal standard for information transfer is needed in the long term to achieve interoperability in such a system. Taking advantage of XML (Extensible Markup Language) seems to be a reasonable course of action. XML is a data content meta-language allowing for semantic tagging of data (W3C, 1997). Microsoft and Netscape have each promised support for XML with style sheets in their respective Web browsers to help replace HTML with XML as the language of the Web. The WWW Consortium has recently proposed the first version of the XML specification. However, XML is not a panacea for system interoperability. Even with tagged data, tags need to be semantically consistent across merchant boundaries at least for the full value chain of the travel industry. However, a travel ontology is non-existent at this moment and coming into one requires cross-industry co-operation which is never a trivial issue.

In the following paragraphs, the inter-dependency of user automation and multi-agent systems will be illustrated. This helps explain why a multi-agent approach is more appropriate to solve the problem.

[1] By this assumption, the following obstacles are not taken into account: (1) the high costs, and hence the source of investment involved in building the infrastructure from scratch, e.g., XML servers; (2) the need of unanimous support from travel retailers and suppliers across the whole value chain, e.g., assuming that large and small suppliers have same intentions; (3) the high costs in re-writing legacy systems; (4) the huge inertia that it creates.

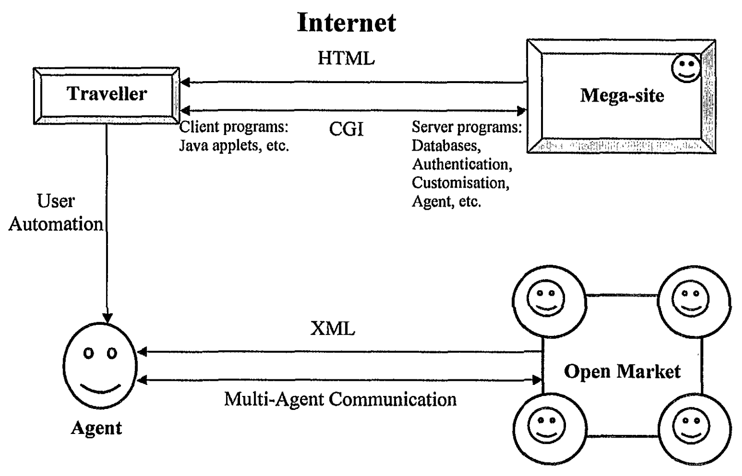

Internet
HTML
Traveller
Mega-site
Client programs: Java applets, etc.
CGI
Server programs: Databases, Authentication, Customisation, Agent, etc.
User Automation
XML
Multi-Agent Communication
Open Market
Agent

Figure 5.3 Inter-dependency of User Automation and Multi-Agent Systems

The traditional form of communication on the Internet between the client and the server is via HTML and CGI (Common Gateway Interface). HTML is used to support static information exchange. For example, when the traveller sends requests for information, the supplier replies by sending the relevant material across the Internet in HTML standard. The HTML standard also specifies how form entries are transmitted back to the server. In order to have two-way interactions, programs must exist on the server to make use of the received information (e.g., form entries) via CGI. Gradually, CGI programs evolved and split into server side programs and client side programs. Server side programs can be used to handle real-time databases, authentication, customisation, agent interfaces, etc, while client side programs, such as Java applets, Java Scripts, etc, perform some of the functions of traditional server programs, and can be used for pure information display such as animation. The additional advantage of this evolution is that the server can send programs to the client which can be run directly on the client's computer.

In the world of electronic commerce, the interactions between the traveller and the supplier are supported by both kinds of communications. However, HTML alone is passive and does not support interactive functions such as authentication and agent interface for customer personalisation. Therefore, in the mega-site scenario, to make the site e-commerce enabled, it needs CGI or server programs to support the aforementioned functionality.

Now, it is assumed that XML standards are in place in the open market scenario. XML, though makes search a lot easier, is still passive. Like HTML, it does not support interactive communications between the client and the server. CGI programs are still needed to provide the interactive functionality. Following from the arguments in previous sections, it is predicted that many travellers will be happy to delegate their search tasks to agents in the future. One may argue that it is enough to have the user agent alone to do the search. While XML will enable easier search on some information, it is unlikely that companies will allow unrestricted access to sensitive data stored in their inventory databases. Clearly, if the ultimate level of user automation is to be achieved, server programs will need to interact intelligently with the agents on the client side. Therefore, a prediction could safely be made – CGI programs would evolve into agents in order to provide the intelligent interactive interface. A multi-agent system, if this deduction is accurate, is hence the only workable solution.

5.2.3 Criteria for Multi-Agents

Multi-agent systems offer a way to relax the constraints of centralised control to provide systems that are decentralised and distributed. Generally speaking, if the utility of using a multi-agent is greater than the utility gained from a single 'agent' application, then such an approach should be used. Multi-agent systems are ideally suited to:

- problems which are inherently distributed;

- provide solutions that require the collation and integration of information from distributed 'self-interested' information sources;
- provide solutions where the expertise is distributed, and
- allow for the interconnecting and interoperation of multiple existing legacy systems, e.g., expert systems, decision-support systems, etc.

For example, a traveller wants to arrange a trip from a town outside London to a city on the West Coast of the USA. This is the sort of activity that people wish to delegate to their personal travel software agents, in the same way as they do to their human secretaries. Today, it is still largely the case that the secretary would consult other human travel agents, who in turn contact yet others to arrange the flight and itinerary. These others include hotel agents, railway agents, rental car agents, etc. However with much of the information now being found online, but being owned by different suppliers who all want to profit from the information and service, why don't users have personal travel agents to negotiate with supplier agents?

By looking at the problem closely, it becomes clear that it has all the characteristics that match with those on the 'suitability' checklist. The problem naturally crosses organisational boundaries because it requires information from various travel suppliers. However, physical distribution is not the only reason for a multi-agent approach. The ownership of information and the self-interest intention of suppliers are important here. As mentioned, information is distributed over different organisations, such as airlines, hotels, etc., so no single player can (or does) have access to all the information. In other words, the problems to be tackled do not have one overall goal, but rather consist of balancing the (possibly conflicting) goals of different players. To put it more precisely, the interoperation of separately developed and self-interested agents provide a service beyond the capability of any agent in the set up, and in the process, all or most gain financially. Economically speaking, all these agents have comparative advantages over each other due to specialisation, and trading their services is good for all.

Talking about self-interest here, there is another compelling advantage of a multi-agent approach for the travel scenario. When one takes a full view of the travel scenario to include the personal agent as well as the supplier agents, the only possible way to protect the self-interest of each party involved is to have his/her OWN representatives. The objective is to avoid dual representation because of the issue of trust. How can users trust that the agent will act solely for their own interests if these agents do not belong to them in the first place?

5.3 Obstacles for the Multi-Agent Approach

The advantages of a multi-agent approach are clear enough. With so many roles that agents can play, there should have been a lot of commercial applications. However, there is no sign that the current market situation is close to the realisation of the ideal scenario shown below. Now, it is time to revisit the promises more closely while simultaneously examining some of the facts.

5.3.1 An Ideal Scenario

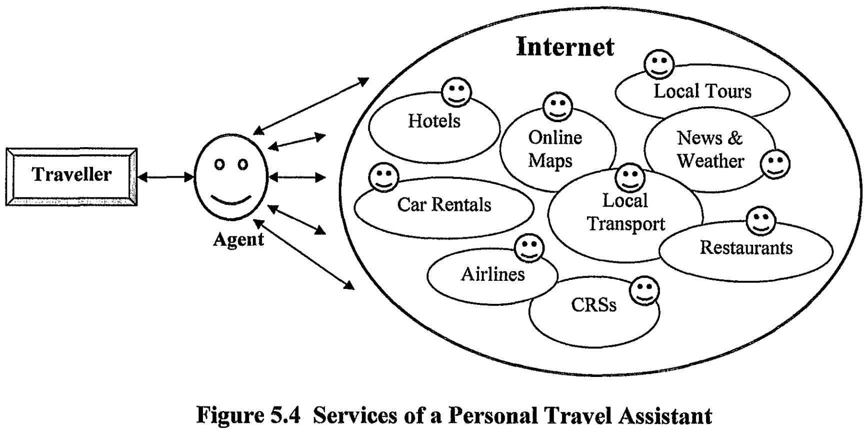


Figure 5.4 Services of a Personal Travel Assistant

Suppose a traveller wishes to arrange a trip from a town outside London to a city on the western coast of the USA. Ideally these requirements could be submitted once to an online travel assistant, that could use its knowledge of the travel industry to plan an appropriate itinerary. As shown in Figure 5.4, this hypothetical travel assistant would use the traveller's profile and requirements, such as the destination, the duration of stay, etc. to locate, select and interact with the appropriate online information sources and services. For the current scenario, it would consult the global flight schedules to determine which airlines serve the chosen destination, and the airports from which they depart. Information from the national UK rail timetables could then be used to determine what time the traveller would need to leave home to reach the airport with adequate time to check-in. The assistant could also arrange hotel accommodation for the duration of the traveller's stay, and provide a list of local attractions that might interest the traveller during his/her stay. Once a trip has been planned, the PTA could then be instructed to monitor news sources for any events that might conceivably delay or jeopardise the journey. For instance, before the trip, the PTA could report on current airport weather conditions and flight delays; and once the traveller has left for the airport, the PTA could monitor motorway traffic congestion reports and send messages to the traveller's hands-free mobile phone.

The promise is that the PTA will negotiate with other software agents representing the interests of the different suppliers and select the most appropriate product/services based on the interests and preferences of the traveller. This way, the traveller's itinerary gets generated with minimal or possibly without any human intervention, unless changes to the itinerary are required.

If the above claims are valid, then agents should have been used in the following areas in tourism:

- Tourists will employ software agents to help them identify and locate travel products and services that they require;

- Tourists and suppliers will empower and trust their agents (to varying degrees) to negotiate electronically on their behalf in order to automate various electronic transaction activities.

However, where are these agents?

5.3.2 The Problematic Reality

The lofty scenario portrayed above generates some critical challenges. If agents are to be introduced with such a revolutionary approach in electronic commerce in tourism, then it is necessary to increase agent sophistication to a higher level and allow multi-agent interoperability among consumer and supplier agents.

The travel industry involves many components such as service providers, intermediaries, tourist offices, travel-related content providers, etc. typically from many different companies. In applying agents to electronic commerce in tourism, various implementations from various suppliers must interoperate and dynamically discover each other as different services come and go. Currently there are several significant technical problems. Most of these problems are rooted in the poor or non-existent interoperability between the many and diverse heterogeneous supplier systems that would be involved in the provision of an integrated service.

5.3.2.1 Agent-to-Agent Communication

When the PTA tries to locate a flight reservation service on the Internet, it must *communicate* with the reservation system. There are three prerequisites for making an effective communication.

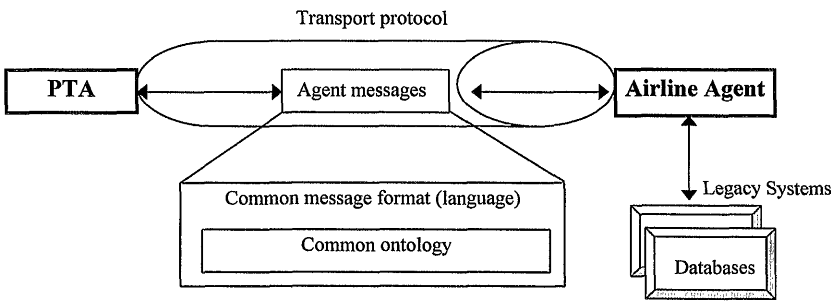


Figure 5.5 Prerequisites for Inter-Agent Communication

Firstly, there must be some means of transmitting the request to the airline agent. Both parties must adopt a common transport protocol that enables the request and results to be transmitted, as shown in Figure 5.5. Though the TCP/IP protocol can handle low-level transfer of information across the Internet at the moment, a *higher-level protocol* is needed to facilitate agent message transfer.

In addition, for both parties to understand one another, their messages must be in a common language that is grounded in a shared ontology (the ontology problem is discussed in the next section). Therefore, the next challenge is to devise a *common communication language* to express the structure and content of messages. This problem requires the communicating parties to agree on what instructions, assertions, requests, etc. will be supported, and what syntax is used. Currently, HTML is the standard format used that enables browser software to interpret pages on the WWW. However, HTML Web pages are designed for presentation of information for humans, and so valuable information is typically intermingled with formatting instructions, necessitating 'wrapper induction' software programs (Kushmerick, 1997), to parse the pages and extract the information from the formatting instructions. Clearly, in the travel scenario, a language geared towards direct machine-to-machine communication is preferable.

As a matter of fact, a number of inter-agent communication languages have been proposed, namely KQML (Finin & Labrou, 1997) and FIPA ACL (FIPA, 1999). These languages are based on speech act theory performatives (Searle, 1969), wherein the speaker's intent of the effects of a message on the hearer is communicated by specifying the type of the message, e.g. ask, tell, etc. However, these ACLs have yet to make their way into any major commercial product. Implementations have primarily been restricted to prototypical demonstrations. Hence, the question of testing the conformance and stability of ACLs is still kept open.

5.3.2.2 Common Ontology

Even with the transport protocol and structure of messages agreed, a final challenge remains – there must be some ways for the PTA and the airline agent to agree on the list of terms to be used in the content of messages, and the meanings of these terms. This obstacle is known as the *ontology* problem, and is perhaps the main problem preventing widespread interoperability of heterogeneous multi-agent systems.

For the PTA and the airline agent to interact, both systems must agree on and share a common definition of travel-related concepts. Once a message is received, the airline agent will use its knowledge of the ontology to translate the request into the terms used within the flight database. The ontology (or concept definitions) specifies the terms each party must understand and use during communication, e.g., definition of a journey leg, identities of airlines, types of fare, etc. Creating an ontology involves explicitly defining every concept to be represented. Take 'journey legs' as an example. One needs to know what the attribute of a leg is, and what each attribute means pragmatically. One must also ask what the constraints on valid attribute values are, and how the attributes of one concept are related to those of another.

While some general purpose ontologies have been proposed for use in inter-agent communication, like CYC (Lenat, 1995); the current trend is towards the provision of editors for creating domain-specific ontologies and converters for translating between ontologies (Gruber, 1993). The disadvantage of this approach is that, firstly, most general-purpose ontologies are unlikely to include the intricacies of all possible domains; and secondly that they are likely to be bloated and unnecessarily complex for most applications.

However, current research (FIPA, 1997) truly fails to appreciate the magnitude of this problem. The problem stems from what is sometimes referred to as the interaction problem between domain and task. The issue is, when designing a multi-agent system, in order to define the domain ontology, one needs to know the purpose or task for which it will be used. For example, to plan a trip, an agent needs to be aware of ontology concepts such as planes, flights, airports that may be different in other tasks. Worse still, even communicating or requesting the help of another agent requires some clear bilateral understanding of the context or task, so that the right ontology gets used. If the airline agent does not recognise that the PTA is trying to plan a trip, then it would not know how to interpret 'first leg'. 'First leg' has a clear meaning within the travel context, but it may mean something else in another context. In short, there is much unavoidable interaction between the communication layer, the task layer and the content (or domain) layer during agent-to-agent communication. Hence, there is no prospect for true agent interoperability without domain ontology standardisation.

At this moment, there is not a well-known ontology built on travelling. To make matters more complicated, the travel ontology does not exist by itself. Separation and cross-references to other ontologies such as Aviation, Banking, Geography, Entertainment, Tourism, etc. is needed. Apart from the universal ISO (International

Organisation for Standardisation) standards for basic concepts on country, currency, date and time, quantity and unit, etc. that are already in use, there is no concerted effort to tackle this problem. Early effort has been done by FIPA (1997) to define a limited travel ontology like origin, destination, budget, preferences, etc. Open Travel Alliance[2] (McNulty, 1999) is a recent attempt to arrive at a common dictionary for the travel industry. CommerceNet[3] (1998) and member organisations are working towards a common ontology for electronic commerce. However, it is still an open question how terms should be universally defined and who should manage their evolution.

FIPA is increasingly becoming aware of the ontology problem, but perhaps not its magnitude. Clear enough, there is no 'complete' answer to this problem in the short term.

5.3.2.3 Legacy Software Integration

Legacy systems are large pieces of software based on older technologies, and hence they are generally not designed for interoperability with other systems. The airline databases in Figure 5.5 are examples of legacy software. The legacy system problem involves devising some mechanism that enables legacy systems to communicate with external systems like the PTA. The solution is likely to involve a proxy, e.g., the flight reservation system, capable of translating requests made using the shared ontology into queries in the internal language of the legacy system, and then translating the results received from the legacy system back into the shared ontology.

Genesereth and Ketchpel (1994) discuss the problem of integrating legacy software with agent systems, and suggest three possible solutions to the problem[4]. Firstly, it is possible to rewrite the software, but this is a costly approach. Secondly, integration

[2] See footnote 8 in chapter one.

[3] See footnote 9 in chapter one.

[4] See section 'Future Challenges' in chapter three for details.

can be done through the use of a separate piece of software called a transducer that acts as an interpreter between the agent communication language and the native protocol of the legacy system. Lastly, the wrapper technique can be used where the legacy program is augmented with code that enables it to communicate using the inter-agent language. Again, this relates squarely to the ontology problem. There are only a few efforts around the globe attempting to automate this process. Clearly, with a large number of legacy systems out in the field, this is a pressing concern.

5.3.2.4 Inertia

To sum up, building a multi-agent PTA system is by no means straightforward. It requires an agent-independent inter-agent communicating language that the agents use to communicate with one another, and a common travel ontology that defines the application domain concepts being communicated between the agents. In addition, agent systems may need to interface with legacy systems such as airlines' databases. The combined effects of all these problems will create a strong inertia because the barriers involved in achieving this idealistic goal seem so insurmountable. Travel suppliers not only have to make big investments in upgrading their technologies, hiring new supporting employees, etc, but also engage in a certain scale of company re-organisation.

Of course, the benefits of the PTA system to small suppliers such as small hotels are obvious because agents level the playing field by increasing their exposure and provide equal opportunities for doing business on the Web. However, such a big investment will put any small company out of reach. As to large suppliers, the advantages they would derive from the system are not so compelling. Some lessons can be learned from the experience of Internet 'shopbots' here. For example, research (Guttman et al., 1998) shows that one-third of the online CD merchants assessed by BargainFinder[5] blocked all of its price requests. The merchants did not want to

[5] BargainFinder is a 'shopping' agent for online price comparison on music CDs.

compete on price. In other words, increasing market transparency will decrease price differential, which is less to large suppliers' benefits, especially when they are already well-known to the customers.

Under this situation, the most likely outcome is that large suppliers do not want to change unless there is some strong market force. Without the financial power, small companies will wait quietly, not to mention customers.

5.4 Design Principles and Strategies for PTA

The difficulties inherent in the multi-agent approach for electronic commerce in tourism suggest that rather than having a revolutionary impact on tourism as was hyped, the promises of the approach would be realised in an *evolutionary* manner. The design principles of PTA are to (1) pave a migration path (how to build agents); and (2) accelerate penetration (what features to consider).

5.4.1 Paving the Migration Path

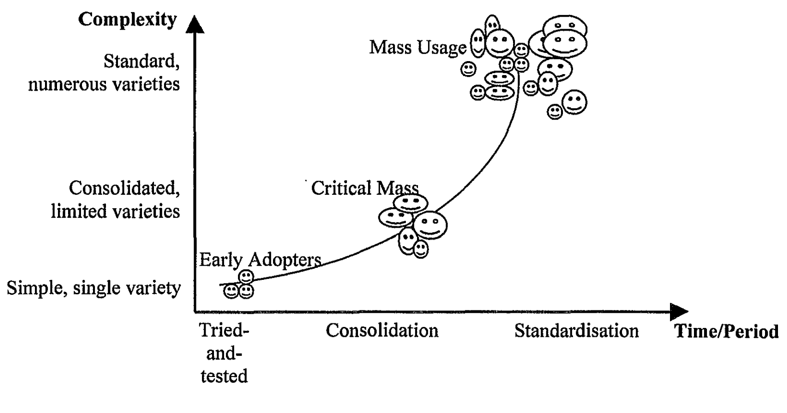


Figure 5.6 The Agent Migration Path

The roadmap for the development of agents can be viewed as characterised by three landmark phases. The first phase is the *'tried-and-tested'* period. A simple, cheap and useful product is needed to attract early adopters. They are mostly individual travellers who would like to try out the new technology, and small suppliers who want to provide added-value services to acquire competitive advantages. Large suppliers are not too keen because they do not want to increase market transparency and reduce their market share. It is likely that this product will be available free with practical, though limited functions and minimum cosmetic features. At this stage, only a single variety will exist. If this product proves to be useful, then more customers will join in, pushing the demand up.

The market will start to enter the second phase – *consolidation.* By consolidation, it means that defects of the 'first' product will be weeded out. A characteristic of this phase is the graduation of the 'first' into a 'proven' product. When more and more customers use the agents, it will benefit the small suppliers because of the increase of market exposure and hence higher demand for their products. Software developers will also see the potentially lucrative market and they will develop more sophisticated products with more functionality. This results in a more thorough exploration and hence a better understanding of the capabilities of the technology. There should also be further development of more structured methodological approaches for using the technology. Hence, this phase will see the emergence of different varieties to suit different applications. Demand is believed to get to a point – critical mass – where the large suppliers cannot afford to stay aloof and are being drawn into the game.

The third and final phase is *standardisation.* With more structured methodologies, it may bring along formal standards. Once standards are established, the technology will just take off. A result of standardisation is generally mass and routine usage. It is expected that a myriad of agent varieties, simple and complicated, cheap and costly, will be designed to suit different needs.

How to create this migration path?

5.4.1.1 Exploit Existing Technologies

The key factor that will determine the speed and depth to which multi-agent systems penetrate the commercial marketplace is the ease with which applications can be developed. The current agent applications that exist are hand-crafted from scratch for each new problem. This means that they have a high overhead as the relatively complex infrastructure for agent computing needs to be put in place before the rest of the application can be constructed. Hence, valuable time (and hence money) is often spent implementing libraries and software tools that, in the end, do little more than exchange KQML-like messages across a network. By the time these libraries and tools have been implemented, there is frequently little time, energy, or enthusiasm left to work on the agents themselves.

When developing any agent system, the percentage of the design that is agent-specific (e.g., doing co-operation or negotiation, or learning a user's profile) is comparatively small. This conforms to the raisin bread view of system development (Etzioni, 1996) in which the parts of the system which can be considered agent-based conform to the small percentage of raisins and the more standard technology needed to build the majority of the system conforms to the significantly larger amount of bread. Given these relative percentages, it is important that *conventional technologies and techniques are exploited wherever possible.* Such exploitation speeds up the development process, avoids re-inventing the wheel, and enables sufficient time to be devoted to the value-added agent component. This point may seem obvious, but many agent projects fail to take it on board.

5.4.1.2 Start Simple

When one builds an agent application, there is an understandable temptation to focus exclusively on the agent specific aspects of the application. After all, these are seen as the justification for the project in the first place. If one does this, then the result is often an agent framework that is too overburdened with experimental AI techniques to be usable. This problem is fuelled by a kind of 'feature envy', where one reads about agents that have the ability to plan, or communicate in natural language, etc., and imagines that such features are essential in one's own agent system. It is rather as if a musician became so enamoured of new instruments capable of generating novel sounds that he lost all interest in seeking the kind of disciplined musical inspiration that makes his art finally worthwhile. In general, a more successful strategy is to build agents with a minimum of AI techniques; as success is obtained with such systems, they can be progressively evolved into richer systems. This is what Etzioni (1996) calls the *'useful first' strategy.*

5.4.1.3 Build 'True' Agents

While at one extreme, there are developers obsessed with developing agent systems that employ only the most sophisticated and complex AI techniques available (and as a consequence fail to provide a sufficiently robust basis for the system), at the other, there exists so-called agents that do nothing to justify the use of the term. A common example is the practice of referring to WWW pages that have any behind the scenes processing as 'agents'. Such practices are unhelpful, it may lead to disappointment of the users who hold great expectations, just to find out that they are no more than a very conventional piece of software. Therefore, the way ahead will be to start simple, but be useful and extensible.

5.4.2 Accelerate Penetration

To facilitate large-scale realisation of the collaborative agent approach to electronic commerce in tourism, it is necessary to design first-wave software to build up a critical mass of early users. These agents are simpler, (and hence cheap or even free), easy-to-use but potentially useful and flexible. The ultimate goal is to *break the inertia and accelerate penetration.*

'You can imagine thinking of an intelligent landscape inhabited not only by humans but by smartifacts - artifacts that are intelligent enough to have some degree of autonomy. [It] will be decades and decades before we have agents or devices intelligent enough to make people nervous. But we already have devices today that are sufficiently autonomous that they do things for us that are practical.' (Saffo, 1996)

5.4.2.1 Simplicity

In order for a technology to be used with eagerness, there must be a clear notion of its purpose and usefulness. Only if some clear gains are realised during the course of its use will any further progress be made. Slick and expensive commercial packages are not required at this stage, but it must be possible to use cheap or public domain software reasonably easily, even if this has limited functionality. One outcome of this process - if it is successful - is a demand for better tools and methods. At the same time, particular tools can have the effect of increasing experience and thus demand. This is well-proven by the history of expert system shells. Limited as they were, expert system shells popularised the technology widely, and laid the basis for a second generation of more sophisticated tools.

5.4.2.2 Low-cost

Users' ability to adopt a new technology is limited by their ability to understand the potential opportunity or returns offered by the technology. Typically, widespread adoption requires some early adopters to create the breakthrough thinking and try out the product. If the software comes *cheap*, there is little overhead in trying it out. Just like something available free for download (e.g., Java), which experience shows (as stated in chapter one) that customers are unlikely to have much hesitation.

On the other hand, adopting a new technology is far more complicated to a supplier. Most often, it depends on the ability to re-engineer organisational processes, and even the whole organisation, as quickly as the technology changes. Unfortunately, few companies are able to do this. Therefore, cheaper software is necessary to keep development costs low and provide a greater chance to make tangible returns on investments.

5.4.2.3 Flexibility

However, being cheap alone is not enough to attract the consumers. The software needs to be potentially useful. It must have a future, i.e., be designed with future extensions in mind. In other words, simpler software now will have the capability of evolving into a more complicated technology. Rather than having extensive hand-coded knowledge and a high degree of intelligence, PTA will be kept simple intentionally and hence cheap. However, to be useful, it possesses the ability to learn in a collaborative manner to achieve *compound intelligence*. It is believed that such a strategy makes it possible for an agent to start simple, but augment to a highly intelligent assistant with experience. These agents will be able to achieve an up-and-running status via knowledge sharing and consolidation within a reasonably short duration.

5.4.2.4 Compatibility

As mentioned earlier, interoperability is a major barrier to the PTA system. Standards are necessary and will occur somehow sometime. The idea of utilising 'existing' HTTP servers is a means to get around the interoperability problem during the transition period. It also avoids unnecessary overheads and keeps development costs low. However, the prototype should be flexible enough to allow future replacement or upgrade with little difficulty in the future. With the absence of a custom infrastructure, the prototype has the flexibility to allow for future deployment of agent servers, agent mobility, and the evolution of agents with increasing intelligence and knowledge. The idea is to create a *migration path* to ease the transformation from HTTP servers to agent servers. Only in this way will the software be used without resistance from customers and suppliers.

5.5 Related Work

The review below is split into three categories: (1) multi-agent building tools; (2) real-world systems in electronic commerce; and (3) multi-agent systems in tourism.

5.5.1 Multi-Agent Building Tools

Over the past few years, a multitude of multi-agent system frameworks have emerged in the market, mostly developed by non-profit organisations like universities. Instead of providing a compendium of such systems, a few systems, that use an ACL for inter-agent communication, were selected for further illustration. All the systems below use some variants of KQML as their ACL. As of the spring of 1998, there were no published, deployed systems claiming to use FIPA ACL.

5.5.1.1 Infosleuth

Infosleuth (Nodine & Unruh, 1997) emphasises the semantic integration of heterogeneous information in an open dynamic environment. The communicating agents, primarily written in Java, make use of an infrastructure of basic services (agents) for authentication, brokering, monitoring, and visualisation of the agents' interaction. An integral part of the architecture is the ontology agent, which assists with the semantic integration of the information handled. Infosleuth agents engage in conversations rather than single-message exchanges.

5.5.1.2 KAoS

Knowledgeable Agent-Oriented System (Bradshaw et al., 1997) is a Boeing project aimed at providing an infrastructure for agent development. Kaos emphasises persistent interaction between agents that take into account not only the particular communication primitive but also the content of the message and the applicable conversation policies. The system allows the design of agents that support specialised suites of interactions.

5.5.1.3 Infomaster

Infomaster (Geresereth et al., 1997) is an information integration system from Stanford University that uses a KQML variant with KIF as its content language. The resulting language does not observe the distinction between the content layer and the message layer. Infomaster integrates structured information sources, giving the illusion of a centralised, homogeneous information system.

5.5.1.4 JATLite

Java Agent Template, Lite - JATLite (1997) - is a package of Java programs, developed at Stanford University, that allow users to create communicating agents quickly. Agents run as applets launched from a browser, and for that reason, all agents register with an agent message router facilitator that handles message delivery.

5.5.1.5 JAFMAS

The Java-based Agent Framework for Multi-Agent Systems (Chauhan & Baker, 1998) is a set of classes that support implementing communicating agents in Java. Developed at the University of Cincinnati, JAFMAS supports directed (point-to-point) communication as well as subject-based, broadcast communications.

5.5.1.6 Jackal

Jackal (Cost, 1998), developed at University of Maryland, Baltimore County, is another Java package that allows applications written in Java to communicate via an ACL. KQML is currently the ACL of choice for this package, but it could easily support FIPA ACL. Jackal is currently in use in the CIIMPLEX project (Peng, 1998), a project that involves planning and scheduling for manufacturing. Jackal strongly emphasises conversations between agents. It includes support for registration, naming, and control of agents.

The review of these projects shows:

- ***Limited scope:*** Though these tools may be the results of hard work, they are inevitably quite narrow in scope. Some of them may just be frameworks that do little more than supporting the transfer of KQML messages across a network. As mentioned before, building an infrastructure from scratch wastes a lot of time which may be spent on the actual building part of agents.

- ***Multiple alternative tools:*** There is a lack of a solid, agreed-upon standard. The existence of these tools shows that there are available standards for this purpose already, so the problem is not of a technical nature. Simply, the market forces have not reached an agreement on which format, or language, or technique to use in these areas.

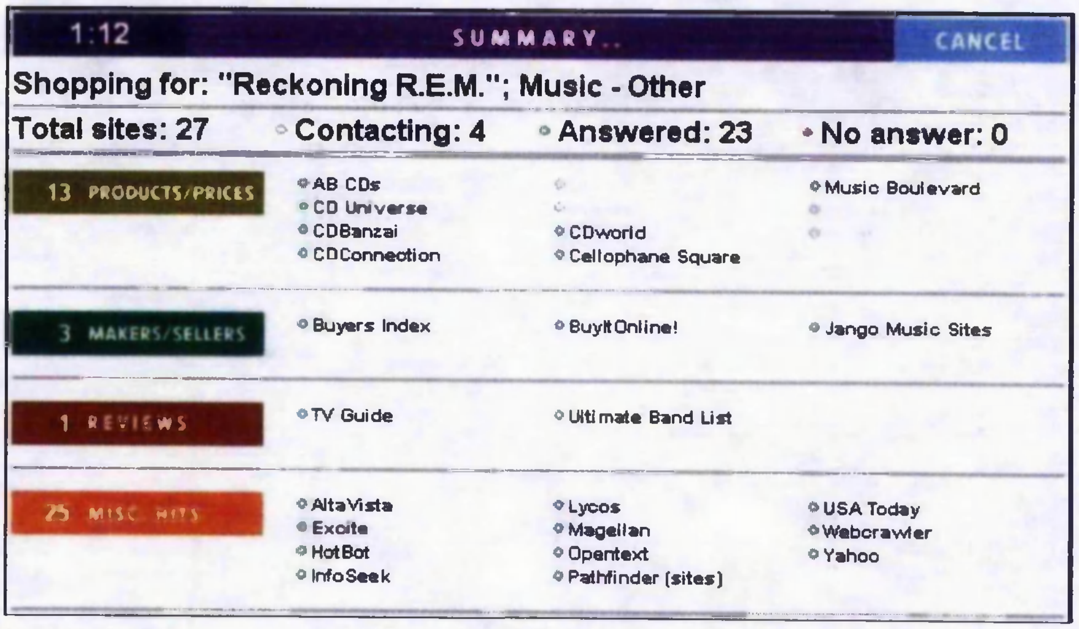
- ***No real solution to ontologies:*** there is no attempt to provide users with means of automating the task of defining custom application-specific ontologies.

- ***Popularity of Java:*** The popularity of Java (Arnold & Gosling, 1997) shows its strength as a portable machine-independent language. It is only recently that computer networks became popular and the need to connect computers within the network arose. Java is an excellent language for network programming and hence naturally stands out as the most suitable language for applications on the Internet.

5.5.2 Electronic Commerce Systems

Agent technologies are already used by some of the larger enterprises at this stage. Agent systems are playing visible roles in retail markets (e.g., Firefly, Jango, etc.) as well as stock markets (e.g., E-Trade). Jango, PersonaLogic and Firefly are typical examples.

5.5.2.1 Jango



Shopping for: "Reckoning R.E.M."; Music - Other

Total sites: 27 ◦ Contacting: 4 ◦ Answered: 23 • No answer: 0

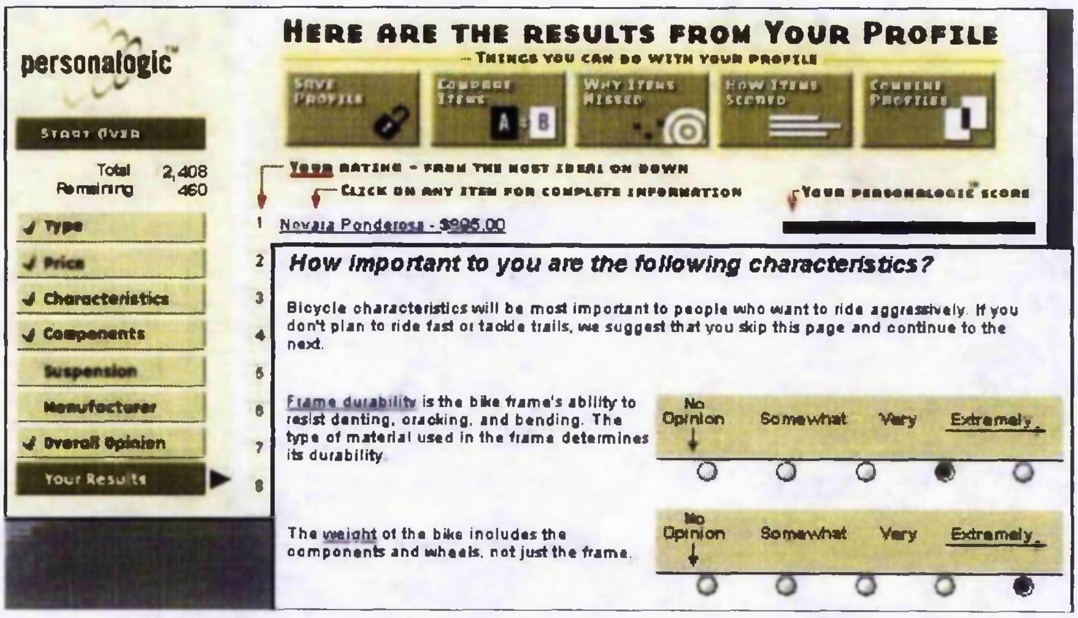
13 PRODUCTS/PRICES	AB CDs CD Universe CDBanzai CDConnection	CDworld Cellophane Square	Music Boulevard
3 MAKERS/SELLERS	Buyers Index	BuyItOnline!	Jango Music Sites
1 REVIEWS	TV Guide	Ultimate Band List	
25 MISC. HITS	AltaVista Excite HotBot InfoSeek	Lycos Magellan Opentext Pathfinder (sites)	USA Today Webcrawler Yahoo

Figure 5.7 Jango's Product Search

Excite's Jango[6] (1996) is a partial merchant comparison engine. Once a shopper has identified a specific product, e.g., music CDs, Jango can simultaneously query merchant sites for its availability, price, and related information. These results allow a shopper (or agent) to compare merchant offerings on price and other criteria. However, Jango does not help shoppers identify which product(s) to purchase in the first place. Moreover, search is primarily based on only one attribute of merchants' offerings, i.e., price.

[6] Jango, formerly known as 'Netbot', is a 'shopping' agent of Excite, Inc.

5.5.2.2 PersonaLogic



How important to you are the following characteristics?

Bicycle characteristics will be most important to people who want to ride aggressively. If you don't plan to ride fast or tackle trails, we suggest that you skip this page and continue to the next.

Frame durability is the bike frame's ability to resist denting, cracking, and bending. The type of material used in the frame determines its durability.

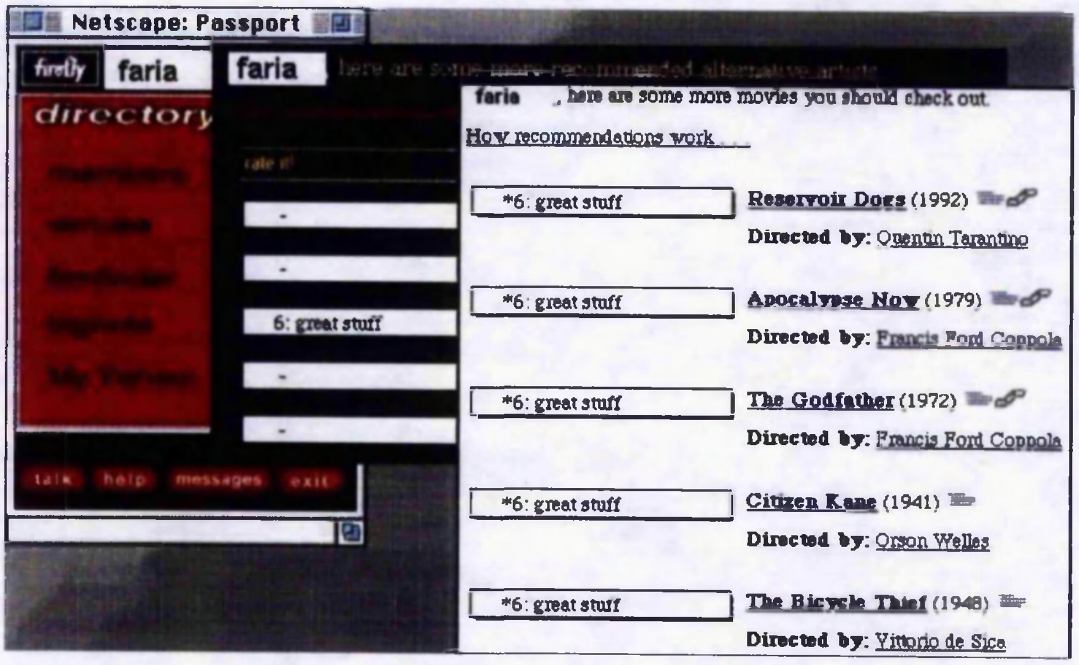
The weight of the bike includes the components and wheels, not just the frame.

Figure 5.8 PersonaLogic's Product Recommendation

Unlike Jango, which compares merchant offerings, PersonaLogic[7] (1999) compares products. PersonaLogic filters out unwanted products within a given domain, e.g., bicycles, by allowing shoppers to specify constraints on a product's features. A constraint satisfaction engine then returns a ranked list of only those products that satisfy all of the constraints. Like Jango, PersonLogic helps fulfil only one stage of shopping and does not help identify product domains. In addition, PersonaLogic does not offer reputation or merchant-specific features (e.g., price, warranty, etc.) to be constrained.

[7] PersonaLogic is a division of America Online (AOL).

5.5.2.3 Firefly



faria , here are some more movies you should check out.

How recommendations work . . .

*6: great stuff — **Reservoir Dogs** (1992)
Directed by: Quentin Tarantino

*6: great stuff — **Apocalypse Now** (1979)
Directed by: Francis Ford Coppola

*6: great stuff — **The Godfather** (1972)
Directed by: Francis Ford Coppola

*6: great stuff — **Citizen Kane** (1941)
Directed by: Orson Welles

*6: great stuff — **The Bicycle Thief** (1948)
Directed by: Vittorio de Sica

6: great stuff

Figure 5.9 Firefly's Movie Recommendation

Like PesonaLogic, Firefly[8] (1999) finds products. However, instead of filtering products based on features, Firefly recommends products via a 'word-of-mouth' recommendation mechanism called Automatic Collaborative Filtering (ACF). ACF first compares a shopper's product ratings with those of other shoppers. After identifying the shopper's 'nearest neighbours', ACF recommends products that these neighbours rated highly but may not yet have been rated by the shopper – potentially resulting in serendipitous finds. Firefly offers a different approach to locating products than PersonaLogic, but falls within the same single stage of shopping and is thus subject to the same limitations.

Neglecting the individual limitations mentioned above for each of these systems, it is obvious that they also share a more important common drawback – all of these are

[8] Firefly is a movie and music recommendation system. It was acquired by Microsoft in August, 1999.

supplier-oriented closed single-agent systems that are ***non-interoperable***; and hence ***expensive*** and ***non-extensible*** because of the limitations of custom-build methodologies and dedicated infrastructures.

The two general goals of electronic commerce in business are *interoperation* and *automation*. In many cases, there is a dependency of automation upon interoperation. As mentioned before, in order to help automate the management of supply chains, there needs to be a semantically interoperable language and protocol for co-ordinating the parties involved. Ideally, these electronic commerce intermediary services (e.g., Jango, PersonaLogic, Firefly) would be highly interoperable to offer a broader range of shopping experience than any one individually. To achieve this, players, products, and intermediaries must be codified into a common language and share a cross-industry developed common ontology.

Unfortunately, there is currently a lack of common languages and ontologies for inter-business interoperation. In fact, these systems use proprietary 'wrapper– techniques to 'scrape' Web pages for product and merchant content. Though HTML Web scraping may suffice for certain problems (e.g., product information retrieval in retail markets), it is very sensitive to minor changes and peripheral format changes, and hence not sufficiently robust to base important business processes on. Of course, it is true that these standards should appear in the long term. However, there is still a long way to go before standards are adopted to succinctly and universally define goods and services, consumer and merchant profiles, value-added services, secure payment mechanisms, etc.

One may be curious why these mega-sites do not share their existing agent system techniques and make it available to everybody, and hence making heterogeneous systems interoperable. The intention of these mega-sites is quite obvious, and partly confirms the speculation mentioned earlier in this chapter. These mega-sites choose to keep the techniques to themselves because they do not want to help their competitors to build agent systems and benefit from their efforts. They do not want to destroy this very profitable monopoly.

As opposed to these sleek but isolated systems, the goal of the thesis is to build a ***simple*, *low-cost, consumer-driven*** product which is made ***flexible*** and ***extensible*** via the sensible use of existing HTTP servers and the innovative collaborative learning strategy. It is worth remarking that in spite of the wild speculation about the revenues to be accrued from agent technology by the turn of the century[9], it remains unclear who will profit from agents. Most of these start-up companies like Jango and Firefly are still loss leaders[10]. Therefore, having a consumer-driven product to break the inertia and increase penetration appears to make a lot of sense. Only after a critical mass has been achieved can one go one step further to extract the next level of system requirements - to enter a detailed design phase for a truly distributed but scalable system.

5.5.3 Travel Systems

There is little related work using agents to provide integrated travel assistance based on combining services from disparate sources.

5.5.3.1 Tabican

Tabican (1997) is the only real-world (commercial) application in the travel industry so far. It uses the Aglets mobile agent platform. Unfortunately, both the Web site and related documentation are presented in Japanese, so it is beyond the author's linguistic knowledge to make any constructive comments. As a general comment, the advantages of using mobile agents are unfounded. A group of programmers have recently set up a petition to make the Aglets software open source in order to keep it

[9] Ovum (a UK market research company) predicted that 'agents will generate US$2.6 billion in revenue by the year 2000' (Ovum, 1994).

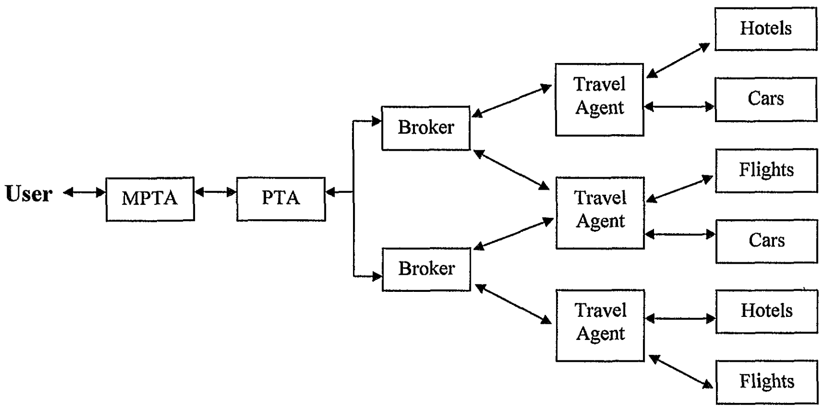
[10] Despite the reported loss, the recent strategic buy-out of Jango (by Excite) and Firefly (by Microsoft), however, showed the recognised importance of agent technologies by large companies.

alive. It is known that the project has been stagnating for a while and gradually slipping out of the limelight in IBM (Aglets Petition, 1999). Needless to say, the argument for a ***simple, user-driven*** product over sleek and complicated systems appears to prove itself.

5.5.3.2 FIPA's PTA

FIPA is an important initiative. It acts as a sanctioning body for the standardisation of agent technology and has made available a series of specifications to direct the development of agent systems. FIPA's PTA is a conceptual framework for researchers and developers to use as guidelines of agent applications. It is not intended to be a working prototype. The importance of FIPA's PTA is that it clearly shows the suitability of using a multi-agent approach to deliver personal travel assistance. As it claims that the travel industry 'presents a prime example to showcase the benefits of agent technology. Agents operating on behalf of their users can provide assistance in the pre-trip planning phase, as well as during the on-trip execution phase of a trip' (FIPA, 1997).

In FIPA's PTA, each player in the travel industry is represented by its own agent, e.g., the traveller has a main PTA and a MPTA (Mini-PTA) that interact with broker agents, travel agency's agents, airline agents, hotel agents, car agents, etc. All these agents communicate in FIPA ACL and use FIPA's travel ontology. The FIPA PTA is modelled as below:



Source: FIPA 1997 Specification, Part 4

Figure 5.10 The Architecture of FIPA Personal Travel Agent

FIPA did make true attempts to specify a common inter-agent communication language – FIPA ACL. However, FIPA ACL does not have a community of users because no FIPA ACL application has appeared yet, and thus it is untested in practice. FIPA ACL will be put to the test as applications that use it are deployed. However, in the end, there is a final question: Is it really necessary to have a performative-based ACL or is it better to work without one, e.g., by standardising collaboration protocols only?

FIPA also attempted to define a limited travel ontology. However, defining limited domain ontologies for limited tasks within limited contexts will only bring along limited and short-lived successes, but not solve the problem outright. Concerted research towards the ontology problem is absolutely crucial.

After all, does all the FIPA effort indirectly show that the concerns for interoperability and ontology make a lot of sense?

5.5.3.3 MOTIV

The German MOTIV (Mobility and Transport in Intermodal Traffic) (Völksen et al., 1997) aims to use collaborative agent technology to provide city-wide personal travel assistance, particularly trip planning and trip execution monitoring. The program plans to integrate many distributed travel oriented services such as hotel and flight reservation systems, restaurant information systems, motorway and local traffic information systems, and parking management systems. In addition, it aims to exploit a number of different communication media and end-user devices.

As the project is commercially sensitive and still in its infancy, it is not clear what progress has been made and hence no real evaluation is possible at this juncture. However, as a general point, this system is somewhat like the idea of a centralised mega-site which includes agent technologies for search assistance. Hence it generates the same question outlined earlier: As the formation of such a system still has to tackle the fundamental problems of interoperability and ontologies, it is a super-scale project with significant development costs because the enabling infrastructure has to be built from scratch. Even if there are enough investments for this purpose, the break-even point will take a long time to reach. This begs the final question: *Who* is going to develop user agents and put them into the hands of the travellers?

5.6 Conclusion

From the survey results of the previous chapter, it can be seen that the electronic travel market is well-cultivated and rich of opportunities. However, with opportunities come more challenges. This chapter suggests that agents are highly suitable to act as mediators to remove the 'friction' in electronic commerce in tourism, especially when they are implemented in a multi-agent system. However, the difficulties inherent in the multi-agent approach prove that agents will be best accepted (and subsequently take off) if they are introduced in an evolutionary rather than a revolutionary manner.

Only *real-world demand* will drive the development of commercial agents, so the availability of agent software that users find useful will help break the inertia and build up a critical mass. This software must have low overheads, achieve maximum impact with minimum disturbance to the status quo, and hence should not require a dedicated infrastructure. It is reasonable to conclude that the revolutionary vision held by most existing agent approaches is somewhat unrealistic at the moment. Needless to argue, the absence of a real-world multi-agent travel system, despite its clear potential, is the best evidence of these sorely neglected issues.

The prototype – PTA - designed to be used on existing HTTP servers to minimise initial investment, is a logical step to bridge this transitional gap. Such an approach will untie the dead-knot of the interoperability problem in the short term, and at the same time allow enough flexibility via collective learning to evolve agents to the next stage of development.

5.7 References

Aglets Petition, (1999), http://luckyspc.lboro.ac.uk/Aglets/Petition/index.html.

Arnold, K. and Gosling, J., (1997), *The Java Programming Language,* 2nd edition, Addison-Wesley.

Bradshaw, J. M., Dutfield, S., Benoit, P. and Woolley, J. D., (1997), KAoS: Toward an Industrial Strength Open Agent Architecture, in Bradshaw, J. M. (ed.) *Software Agents,* MIT Press, Cambridge, Mass., pp. 375-418.

Chauhan, D. and Baker, A.D., (1998), JAFMAS: A Multi-agent Application Development System, in Wooldridge, M and Finn, T. (eds.), *Proceedings of Second ACM Conference on Autonomous Agents,* Minneapolis, MN, pp. 100-107.

CommerceNet, (1998), *Success of XML for E-Commerce Accelerated by Advanced Knowledge Representation Techniques,* Palo Alto, CA, October 9, 1998, http://www.commercenet.com.

Cost, R. S., (1998), Jackal: A Java-Based Tool for Agent Development, *Working Papers of the AAAI '98 Workshop on Software Tools for Developing Agents,* AAAI Press.

Etzioni, O, (1996), Moving up the Information Food Chain: Deploying Softbots on the World Wide Web, *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI '96),* Portland, OR.

Expedia, http://www.expedia.com.

FIPA, (1997), *Specification, Part 4.*

FIPA, (1999), *Specification, Part 2.*

Firefly, (1999), http://www.firefly.com.

Genesereth, M. R., Keller, A. M. and Duschka, O. M., (1997), Infomaster: An Information Integration System, *Proceedings of the ACM Sigmod International Conference on Management of Data,* ACM Press.

Gruber, T. R., (1993), A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition,* Volume 2, pp. 199-220.

Gilbert, D, Aparicio, M, Atkinson, B, Brady, S., Ciccarino, J., Grosof, B., O'Connor, P., Osisek, D., Pritko, S., Spagna, R. and Wilson, L., (1995), *The Role of Intelligent Agents in the Information Infrastructure,* IBM White Paper, US.

Guttman, R. H., Moukas, A. G. and Maes, P., (1998), Agent-Mediated Electronic Commerce: A Survey, *Knowledge Engineering Review,* 13(3), June 1998.

Hopken, W., (1999), *Reference Model of an Electronic Tourism Market,* Workshop, The Sixth ENTER 99 International Conference on Information and Communication Technologies in Tourism, Innsbruck, Austria, 20-23 January, 1999.

JATLite, (1997), http://java.standford.edu.

Jango, (1996), http://www.jango.com.

Lenat, D. B., (1995), CYC: A Large-Scale Investment in Knowledge Infrastructure, *Communications of the ACM,* 38(11), pp. 33-38.

Mansfeld, Y., (1992), From Motivation to Actual Travel, *Annals of Tourism Research,* 19(3), pp. 399-419.

McNulty, M. A., (1999), Suppliers Seek XML Standard, *Business Travel News,* May 3, 1999.

Middleton, V. T. C., (1988), *Marketing in Travel and Tourism,* Butterworth-Heinemann, Oxford.

Moutinho, L., (1987), Consumer Behaviour in Tourism, *European Journal of Marketing,* 21(10), pp. 1-44.

Nodine, M. and Unruh, A., (1997), Facilitating Open Communication in Agent Systems: The Infosleuth Infrastructure, in Singh, M, Rao, A and Wooldridge, M. (eds.), *Proceedings of the 4th International Workshop on Agent Theories, Architectures and Languages,* Springer-Verlag.

Open Travel Alliance, (1999), http://www.opentravel.com.

Ovum, (1994), *Intelligent Agents: The Next Revolution in Software,* http://www.ovum.com.

Peng, Y., (1998), A Multi-Agent System for Enterprise Integration, *Journal of Applied Artificial Intelligence,* 1(1).

PersonaLogic, (1999), http://www.personalogic.com.

Preview Travel, http://www.previewtravel.com.

Saffo, P., (1996), Published Interview, *IBM Networking Press Room,* http://www.networking.ibm.com.

Tabican, (1997), http://www.tabican.ne.jp.

Travelocity, http://www.travelocity.com.

Volksen, G., Dieterich, H. and Steiner, D., (1997), Intelligent Agents for Personal Travel Assistance, *Proceedings of ITS Congress ' 97,* Berlin.

W3C, (1997), *Extensible Markup Language (XML),* World Wide Web Consortium Working Draft, 17 November, 1997.

Chapter Six

The Personal Travel Assistant (PTA) System

In this chapter, a prototype of the Personal Travel Assistant (PTA) system was implemented using the Java language[1]. The idea is to provide a *stepping stone* to accelerate the adoption of agents in the online travel market in order to bridge the gap between the present and the ideal situation. The design principles outlined in chapter five – (1) simplicity, (2) low-cost, (3) flexibility and (4) compatibility were adhered to. To keep the system simple but useful and flexible, multi-agent collaborative learning was used and it successfully demonstrated how PTA acquired new knowledge from other participating agents. To achieve compatibility but at the same time keep development costs low, the interoperability problem was handled by exploiting existing HTTP servers. It is believed that this first step is crucial to realise the ultimate objective to provide personalised journey information and assistance to travellers.

6.1 Requirement Analysis of Agent Development Tools

With the goals of PTA established in previous chapters, the next task is to select the most appropriate development tool and language, which will have a major impact on the software architecture of PTA. It is necessary to look for suitable development

[1] See Appendix C for program listing.

tools with future prospects in the long term rather than those just sufficient for the current implementation of the PTA prototype.

6.1.1 Low Cost

A direct requirement of low cost software is that the development tools must also be low cost. Consumers will not be willing to pay for the software before PTA proves its benefits. There are also insufficient suppliers (who are interested) at the initial stage to contribute towards the cost of expensive development tools. This rules out expensive tools such as Telescript[2]. The *Java 2 Development Kit*[3] is *free*, thanks to new commercial strategies that have been emerging over the age of the Internet. There are also various agent building tools either in the public domain, or available for free, to build the PTA prototype. Several notable tools are also written in Java, which proves that Java and its development kit is a feasible option.

6.1.2 Interoperability

Agents are inherently distributed. They must be able to execute anywhere on the network without prior knowledge of the target hardware and software platform. The most likely solution is a common language available for all popular platforms. Earlier languages such as C/C++ and Prolog are standardised in the core language only. Most programs will reuse libraries of pre-written codes to complete the current task at hand. Unfortunately, it cannot be guaranteed that the same library function behaves exactly the same on different machines, if the function is available at all. More importantly, system calls are <u>mandatory</u> when requesting system resources such as file input/output. However, system calls are different for different operation systems by definition.

[2] See 'Telescript' section in chapter three for details.

[3] Java is a registered trademark of Sun Microsystems, Inc.

Java provides the advantages of *architecture neutrality* and *portability* to agent developers. Java tackles the interoperability problem by the concept of virtual machine. A piece of platform specific software called the *Java Virtual Machine (JVM)* runs on each machine and interprets Java codes. Therefore Java codes are guaranteed to behave exactly the same on different platforms. The solution that the Java system adopts to solve the binary-distribution problem is a 'binary code format' that is independent of hardware architectures, operating system interfaces, and window systems. The format of this system-independent binary code is architecture neutral. If the Java run-time platform is made available for a given hardware and software environment, an application written in Java can then execute in that environment without the need to perform any special porting work for that application. The Java compiler does not generate 'machine code' in the sense of native hardware instructions, rather it generates bytecodes: a high-level, machine-independent code for a hypothetical machine that is implemented by the Java interpreter and run-time system.

The primary benefit of the interpreted byte code approach is that compiled Java language programs are portable to any system on which the Java interpreter and run-time system have been implemented. The architecture-neutral aspect discussed above is one major step towards achieving portability, but there is more to it than that. C and C++ both suffer from the defect of designating many fundamental data types as 'implementation dependent'. Java eliminates this issue by defining standard behaviour that will apply to the data types across all platforms. Java specifies the sizes of all its primitive data types and the behaviour of arithmetic on them.

Furthermore, the Java Virtual Machine comes with a vast amount of *Application Programming Interface*[4] *(API)* covering all aspects of general programming. API can be thought of as system calls, but they are identical on all Java Virtual Machines, and

[4] An API is an agreed-upon input/output format for a particular application program (APP). Humans, or application programs, may rely on that format and in particular use it to 'call' the APP and to interpret the output returned by the APP.

perform much more application-related task such as sorting. More importantly, GUI (Graphical User Interface) are complex and used to be platform-specific. The Swing package in Java provides state of the art GUI that is not only machine-independent, but also selectable in look-and-feel at the change of a flag.

The idea of virtual machine codes and interpreters is actually as old as the computer industry. However, they have never been popular (except in machines with small memories) because of their slow speed over compiled codes. The arrival of the Internet as a universal network changes the situation. Portability is key and speed is not so important when the codes have to travel around the network to client machines to execute. At the same time, machines are getting much faster than the first BASIC interpreters.

6.1.3 Network-Savvy

In any MAS, agents residing on different machines and environment need to communicate information and knowledge about their goals, beliefs and intentions to each other, in order to co-ordinate and co-operate so as to bring about a coherent solution. Thus communication is a very important aspect in the development of any MAS. Java especially lends itself as an extremely suitable choice in this regard. It offers an extensive library of classes and routines supports sending both broadcast and directed messages across the network.

6.1.3.1 Support for Broadcast

Java provides a multicast datagram socket class which is useful for sending and receiving multicast packets. A Multicast Socket has the capabilities for joining 'groups' of other multicast hosts on the Internet. This is an extremely useful feature for implementing a MAS. The ability to send broadcast messages helps in creating a truly scalable and flexible agent framework because the agents do not need to register with a centralised directory service to be able to receive messages from one another when they come on line. Moreover, as the agents do not need to know the identity of the receiver agents in order to be able to send messages to them, there is no necessity for any start-up protocol. It also saves network bandwidth when the same message has to be sent to all the agents in the group (e.g., a sub-factory).

6.1.3.2 Remote Method Invocation

The feature of RMI (Remote Method Invocation) is extremely helpful while creating a family of collaborating agents. The process of creating network connections is extremely easy especially when compared to other languages. After all, Java was 'invented' while Internet technology was getting more mature and popular.

RMI enables the programmer to create distributed Java-to-Java applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. It is possible to generate mobile code using RMI as it is possible to transport objects between client and server. A Java program can make a call on a remote object once it obtains a reference to the remote object, either by looking up the remote object in the bootstrap naming service provided by RMI, or by receiving the reference as an argument or a return value. A client can call a remote object in a server, and that server can also be a client of other remote objects. RMI uses the Object Serialisation feature of Java to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism. Methods invoked in remote server objects by the clients are reachable through the TCP/IP protocol.

6.1.4 Multithreading

Just as the human world is full of multiple events all happening at once, the agent world should work the same way. Unfortunately, writing programs that deal with many things happening at once can be much more difficult than writing conventional single-threaded programs in other languages. The major problem with explicitly programmed thread support is that one can never be quite sure that the locks one needs have been acquired and released at the right time so the situation often ends in a deadlock.

Multithreading is traditionally supported by the operating systems of most main frames and mini computers in the form of concurrent processes. Operating systems by definition do not know in advance what applications they will be running and what resources the applications will need. Concurrency is a way to provide timely response to applications. For example, a desktop user may be waiting for some worksheet calculations to finish in the background while working on some word processing. A user-friendly and efficient way to share the machine will be to make the applications run independently, and apparently at the same time.

Traditional processes are commonly described as heavy weight since the system resources, e.g., memory and runtime overheads, needed to support unpredictable and uncoordinated applications are significant. Later operating systems support lightweight processes within a single application, which use the same system resources as a single heavyweight process. The lightweight processes can only communicate with each other in the application. The next logical step is to support threads in the language itself to guarantee portability across operating systems.

Built-in support for *threads* is one of the most powerful tools in Java, not only to improve interactive performance of graphical applications but also to run multiple events concurrently. Multithreading is the way to obtain fast, lightweight concurrency within a single process space. The Java library provides a Thread class that supports a rich collection of methods to start a thread, run a thread, stop a thread, and check on a thread's status.

6.1.5 Security

Agents will run as guests in foreign host machines. A user may send his/her mobile agent to the network using a handheld device through a wireless connection, disconnect and then reconnect at leisure to receive progress reports from the agent. On the other hand, a vendor's agent may be running on the customer's desktop. The latter is particularly important because it allows dumb servers to make use of agent technology. Existing HTTP servers can be used to download Java codes to the client machine. The server agents can then communicate back with the server via files or conventional CGI programs. Under this situation, security is therefore of prime importance because of the need to execute foreign unknown codes in one's own machines.

Java is intended for use in networked or distributed environments. Toward that end, a great deal of emphasis has been placed on security. Java has a *security manager* to enable browsers to run untrusted applets[5] in a trusted environment. It is responsible for enforcing the applet restrictions to prevent applets from inspecting or changing files on the client file system. It also prevents applets from using network connections to circumvent file protections or people's expectations of privacy. The key to this approach is the fact that Java is based on an *interpreted language* that facilitates detailed control over the capabilities of the agents running on top of it.

In the sandbox model, shown in the following diagram, local code is trusted to have full access to vital system resources, such as the file system, but downloaded remote code (an applet) is not trusted and can access only the limited resources provided inside the sandbox.

[5] An applet is a Java program that is run from inside a Web browser. The HTML page loaded into the Web browser contains an `<applet>` tag, which tells the browser where to find the Java codes contained in files with .class extensions.

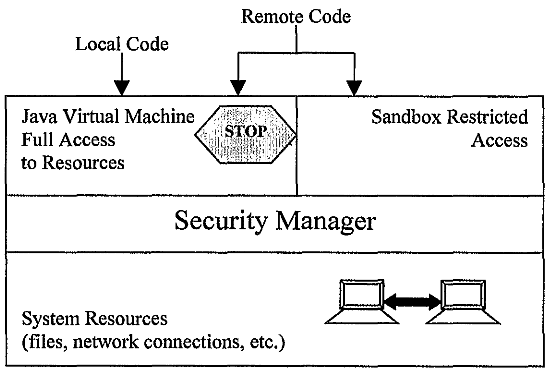


Figure 6.1 Java Security Manager

Java also provides the basic technology for loading and authenticating signed classes. This provides the basic certification mechanisms where agents can be signed with the owner's key. A digitally signed applet is treated like local code, with full access to resources, if the public key used to verify the signature is trusted. Unsigned applets are still run in the sandbox.

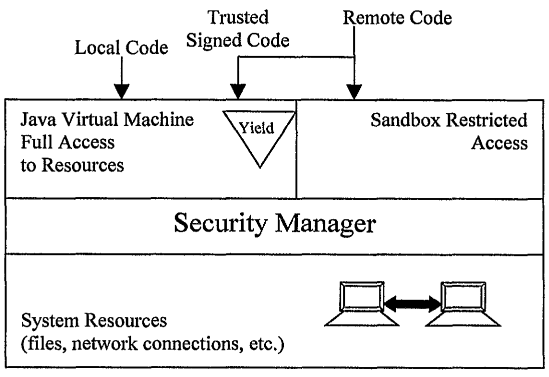


Figure 6.2 Authentication by Digital Signatures

In addition to this, Java provides tools for finer-grained control of flexible security policies. The security policy defines the set of **permissions** available for code from various signers or locations and can be configured by a user or a system administrator. Each permission specifies a permitted access to a particular resource, such as read and write access to a specified file or directory or connect access to a given host and port. The runtime system organises code into individual **domains**, each of which encloses a set of classes whose instances are granted the same set of permissions.

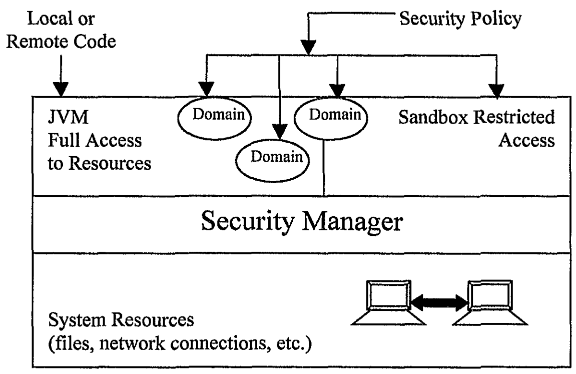


Figure 6.3 Security Policy – Domain Access

Telescript is the only non-Java based building tools that provides or makes use of interpretation. However, it is too costly and extremely memory-hungry. For other tools without interpretation, security management will be difficult.

6.1.6 Dynamic Objects

Object-oriented design is a very powerful concept because it facilitates the clean definition of interfaces and makes it possible to provide reusable 'software ICs (Integrated Circuits)'. It is a technique that focuses design on the data and on the interfaces to it. It is also the mechanism for defining how modules 'plug and play'.

Object-oriented programming is an old concept. Object-oriented languages such as Simula and Small Talk have been around for quite a while. The popularity of these languages might have been fuelled by the need for the development of more sophisticated GUIs. From the look of the program listings, it is rather daunting to write C programs for earlier GUIs such as Motif. Later, development tools for Windows are popular with the prefix Visual, such as Visual C++, Visual Basic, etc. These tools are object-oriented to make writing complex GUIs easier.

Though there are many languages that are object-oriented in nature, the implementations vary. One can write a piece of program that can be compiled by both C++ and Java compilers. C++ emphasises traditional values such as minimising object size and runtime overhead. C++ objects are static in the sense that one has to know the structure of the object at compile time in order to use the object at runtime. In contrast, one can find out the type, or class name, of a Java object at runtime, even the fields and methods within the object, via the Java mechanism of Reflection. This is important for communication amongst agents. For example, a heterogeneous list of objects can be passed to another agent without the need to tell what the types of objects are.

6.1.7 Legacy Integration

Agent building tools should support legacy systems such as database, which is essential in electronic commerce. Therefore, links between agent building tools and the database system are needed.

Java provides ways to call already compiled code in other languages (*native* methods). The Java Database Connectivity (JDBC) kit lets Java programmers connect to any relational database, query it, or update it using the industry standard query language (SQL). This is a very useful feature as databases are among the most common uses of software and hardware today.

Other than legacy codes, native methods are necessary when it is essential to maximise speed of code. Java may be fast enough for communicating over the Internet or for running GUI in the client machine where speed is not the priority. However, it will be hard to imagine running a heavy-duty database the scale of a GDS in a virtual machine with an interpreted language.

6.1.8 Knowledge and Reasoning

Agents must have basic knowledge and reasoning capability to assist consumers. In considering building tools for the agent system, it is necessary to understand how consumers use various decision rules or heuristics to simplify their task of alternative evaluation to arrive at a decision. Generally, the rules can be used in combination (mixed strategy) and may vary with the stages of the decision process. They are commonly classified as *compensatory* and *non-compensatory* rules (Mansfield, 1992, Moutinho, 1987).

6.1.8.1 Non-compensatory Rules

In non-compensatory rules, trade-offs between attributes do not occur. For example, the tourist may evaluate the alternative products on a few key criteria and eliminate products that fall short on these criteria. There are three types of non-compensatory rules, namely the conjunctive, disjunctive and the lexicographic rule. In a conjunctive rule, a product is selected if it meets the minimum standard on all the criteria. In a disjunctive rule, a product is selected if it is perceived to be excellent on one or more key criteria. The lexicographic rule uses all criteria but in a stepwise manner. Products are evaluated on the most important criteria first. If there is a tie, then the choice is based on the next most important criteria, and so on. In the case of non-compensatory decision rules, the potential traveller evaluates each alternative separately, then compares the utility values of each alternative, and eventually chooses one alternative. An unattractive aspect of one attribute is not allowed to be compensated by an attractive aspect of another attribute.

6.1.8.2 Compensatory Rules

Compensatory rules, on the other hand, assume a complex cognitive structure and demand the greatest effort on the part of the decision maker of all the decision rules. It requires a systematic and balanced evaluation of all the alternatives on all the criteria and then summing up the evaluations to derive an overall preference score for each alternative. Thus the weakness of an alternative on one criterion can be compensated for by strength on another. The final choice of a preferred product is the outcome of a decision rule based subjectively on an evaluation score. Low score or negative scores on one or more attributes can be compensated (at least partially) by high scores on one or more of the other attributes, which is basically a trade-off situation. The weighted linear compensatory rule is similar, with the exception that the criteria ratings are weighted by their importance before they are summed.

At the root of every choice is the assignment of utility values to various parts of the alternative. These parts are referred to as 'product attributes'. The set of attributes is constructed in the consumers' mind as a result of perceived needs and expectations derived from a given product, constraints to be faced, and the information collected while pursuing a product choice process. Each attribute within this set is assigned either a positive or negative utility value. Once the perceived attributes and utilities have been structured by the individual, a measurement scale needs to be constructed that enables the weighting of the utility values of each attribute. This preference scale, together with a decision criterion, forms the basis of alternative selection process.

6.1.8.3 Rules Representation

Consumers used 'mixed strategies' - a combination of rules in a sequence to arrive at the final choice. First, those products that are totally unacceptable are eliminated. In this stage, a non-compensatory conjunctive rule is used to eliminate alternatives quickly and arrive at a manageable manner, all of which possess a set level of the evaluative criteria. Second, a compensatory rule is applied for a more thorough evaluation of the subset of alternatives before making the final purchase decision.

Non-compensatory rules are well represented by logic languages such as Prolog. On the other hand, Fuzzy Logic, or even conventional languages apparently better represent compensatory rules.

In a multi-agent context, an agent may learn by receiving new rules from other agents, testing the rules for a period, then assigning importance levels based on their performance, or simply deleting unreliable rules. A disadvantage of logic languages is that one cannot manipulate individual rules. For example, Prolog is designed to deduce the right conclusions from an aggregate of rules and facts, or to find the conditions that make the rules valid. Hence, there is no apparent way to refer to an individual rule.

Another disadvantage of logic languages is that, without the notion of time, they are almost unsuitable for writing anything other than rules. For example, it seems impossible to write any protocols including ACL.

For Java, rule objects can be manipulated just as any other objects. They can be transported across the network and invoked by remote agents without knowing what the rule contents are. It may be difficult to write complex reasoning using only if-then-else statements, but rule objects can be used systematically to build complex reasoning structures such as rule trees.

6.2 The PTA Architecture

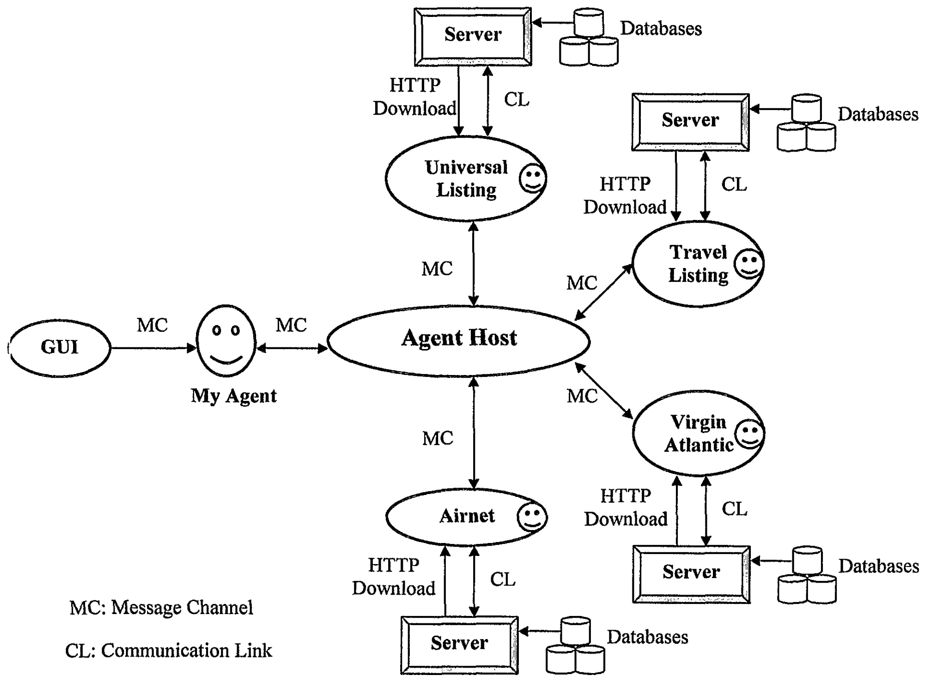


Figure 6.4 The Architecture of PTA

The figure above depicts the architecture of the PTA system. To meet the goal that existing dumb HTTP servers can be used immediately with minor supporting CGI programs, all agents involved in a transaction run on the user's machine.

- The Agent Host provides services common to all agents — download and activate agents, establish communication channels, validate and route messages to the addressed recipients, etc.

- My Agent is a special agent that represents the user. Therefore My Agent controls the GUI. My Agent is automatically started at software initialisation. After some interactions with the user, My Agent will typically invite other agents to visit and conduct businesses.

- Foreign agents represent suppliers and intermediaries. As with all Java objects, these agents reside in a file with '.class' extension. The address of each agent is the full URL of the class file. Once these foreign agents are activated in the host machine, they may invite other agents or communicate back to their servers using custom communication links (CLs). At a minimum, foreign agents can communicate with their base via the existing Common Gateway Interface (CGI). Typically, CGI programs provide, or link to, database services at the suppliers' or intermediaries' sites.

All agents including the host are threads because they run independently of each other. The GUI, which can be seen as an object of all graphical objects, is also a thread by definition. The Java mechanism for synchronising threads is implemented here as message channels (MCs). For all message channels, the forward and return channels have different characteristics. My Agent invokes GUI methods directly rather than sending messages to the GUI. All other forward links (agent to host) share a common channel, thus forming a message queue for the host to process. All return links (host to agent) have dedicated channels to maintain privacy and simplicity.

6.2.1 The User Interface

In designing the interface, issues of 'look and feel' was de-emphasised because it is more important to focus on how to leverage PTA's learning capabilities to increase the interface's expressive power and flexibility. Specifically, the interface of PTA embodies the following ideas:

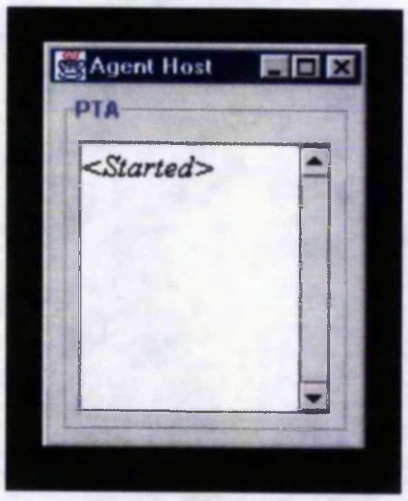
- Goal-oriented: A request indicates what the traveller wants. PTA is responsible for deciding *how* and *what* to satisfy the request.
- Forgiving: A request is not a complete specification of the traveller's goal, but a clue or a hint that PTA attempts to deduce and then satisfy.
- Generic: PTA provides a single, expressive, and uniform interface to a wide variety of services and utilities. PTA does not need prior knowledge of individual products.

The following sections show the process of a typical purchase - a flight ticket.

6.2.1.1 Initialisation

At initialisation, the Agent Host gets started. Then, the Agent Host loads the first agent, My Agent, into the system and activates it. The Agent Host provides an Agent Monitor to display the activities of all active agents. The Agent Monitor maintains a small display area for highlighting the activities of each agent such as the messages it transmits and receives. The message display is in stack form with the last message shown at the top. The display area, shown in Figure 6.5, can be scrolled down to the very first message.

Though the Agent Monitor is mainly designed for development purposes, it provides the user with some idea about what is going on. The Agent Monitor window may be closed to avoid distractions without affecting normal operations.



<Started>

Figure 6.5 Agent Monitor Window

At start-up, My Agent - Personal Travel Assistant - shows up in an unobtrusive window. The combo box can be pulled down to review several options of services. Now, let us walk through the option: 'Find Travel Products'.

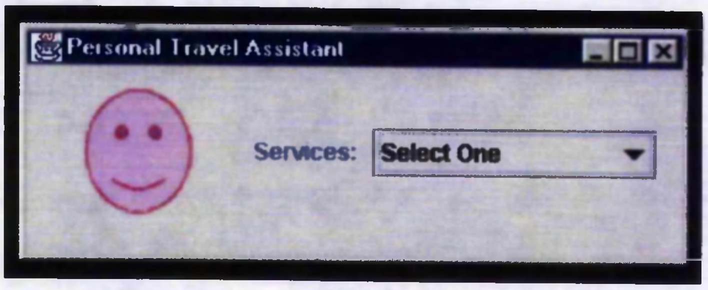


Figure 6.6 PTA Start-up Window

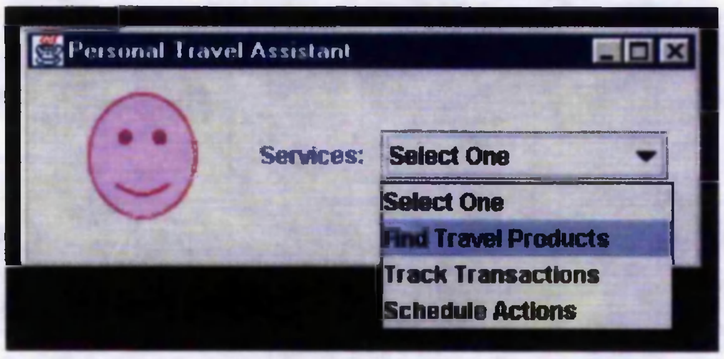
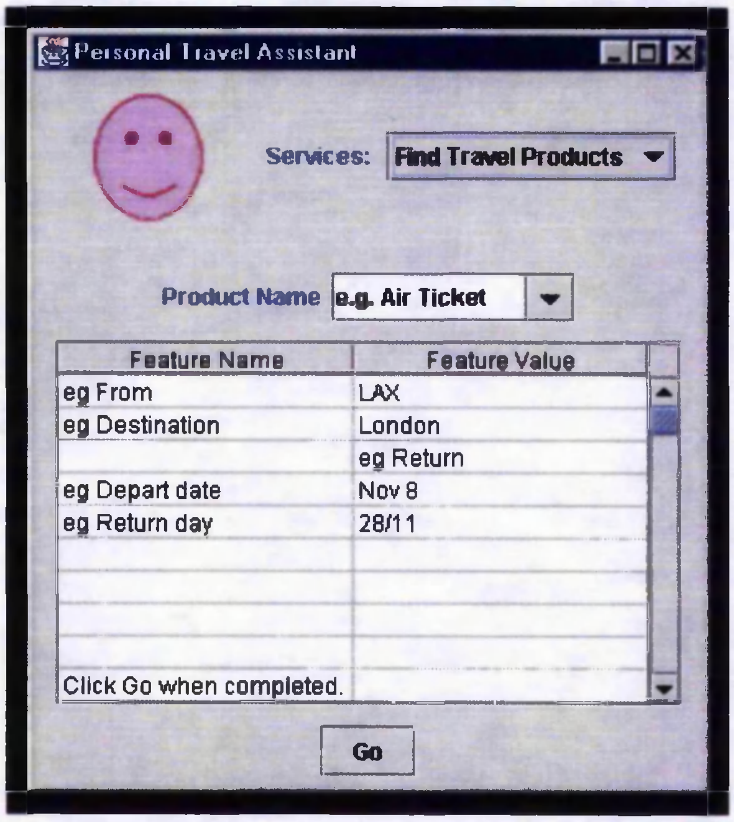


Figure 6.7 Service Combo Box Options

6.2.1.2 Specification

Being a new assistant, PTA starts as a blank sheet. It does not have any prior information of the product requested by the user. Here, a relational model was used, i.e., a table, for the user to enter the product specifications.



Product Name: e.g. Air Ticket

Feature Name	Feature Value
eg From	LAX
eg Destination	London
	eg Return
eg Depart date	Nov 8
eg Return day	28/11
Click Go when completed.	

Figure 6.8 Product Specification

First, the user has to enter the 'name' and 'attribute-value pairs' of the product in his/her own words. The table is fully editable by using the mouse and keys as in any popular editors. An example is shown in the initial product specification table. The user requests for a return air ticket from LAX to London, departing on Nov 8 and returning on 28/11.

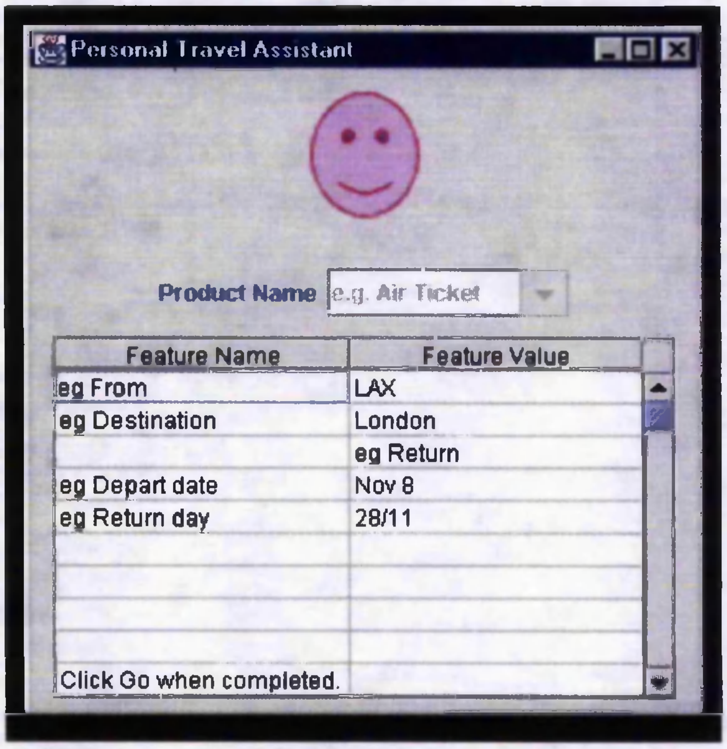
The interface needs to be flexible enough to understand phrases that a normal person will probably understand. Typically, the user will use different date and time formats in an unrestricted entry field. The user may know terms such as non-stop and direct flight, but may not know how to formally describe or name the attributes or features. However, the information in the table can be submitted exactly as it is to PTA without confusion. It is believed that, through collaborative learning, the multi-agent system as a whole should be able to understand these user inputs. Moreover, it is entirely up to the user to decide how detail he/she wants when entering the attribute-value pairs. At a minimum, the user enters the name of the product in his/her own words. Once PTA has the chance to contact other agents, the proper terms will be learned and stored for later use.

The table can be scrolled, the window resized, and the column spacing modified by using only the mouse. This is important since it is assumed that PTA does not know the attributes of a ticket in advance and hence it will not be able to format the table in an ideal way for the user to see clearly. The Java Swing GUI package provides a class of table objects that makes the concept of accepting unknown product attributes possible without tedious programming.

By choosing the relational model, it is assumed that all attributes of the products can be flattened into a table form. However, this may not be true for complex products. For example, even a simple trip may include one or more indirect flights. The alternative of a hierarchy model, i.e., a tree, was investigated. Visual tree objects are also provided in the Swing package, so there is not much technical difficulty in building the user interface. The relational model was chosen over the hierarchy model because it is expected that users are more familiar with data entries in tables. In this prototype, the feasibility of a relational model, which is apparently more natural to the user, was investigated. If a relational model proves to be inadequate, it is always possible to convert to a hierarchy model.

6.2.1.3 Searching

Once the user is satisfied with the table entries, the specifications are submitted to PTA by clicking on the 'Go' button. When PTA is searching for alternatives, the user is prevented from making changes to the table that might cause confusion. For example, the Product Name combo box is dimmed and disabled, and the 'Go' button disappears altogether. It is easy to provide these fine controls in GUI using the Java Swing package.



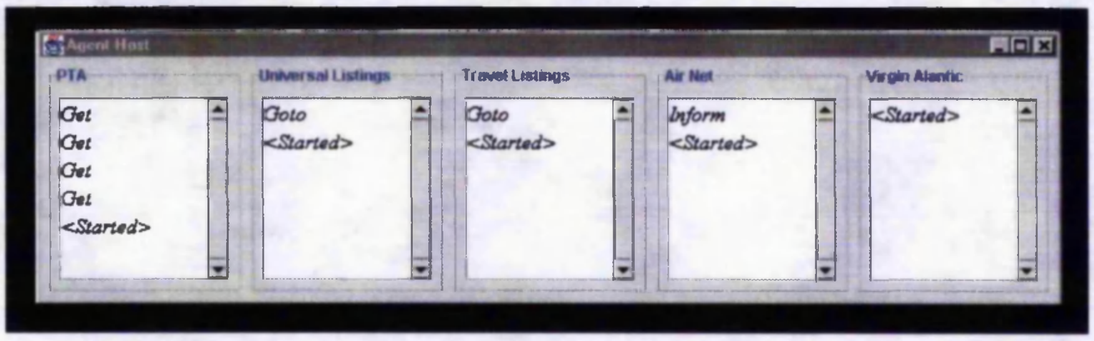
Feature Name	Feature Value
eg From	LAX
eg Destination	London
	eg Return
eg Depart date	Nov 8
eg Return day	28/11
Click Go when completed.	

Figure 6.9 Search in Progress

When PTA is searching for the requested ticket, the Agent Monitor, shown in Figure 6.10 provides the user with some ideas on what is happening in the background. PTA needs at least the address of one general-purpose category listing agent to search for all products, which is a company called Universal Listings[6]. PTA sends a 'Get' message to Universal Listings, containing all the product specifications. The Agent Host then downloads the Universal Listings Agent. After analysing the name of the product, the Universal Listings Agent consults its server database and comes to a conclusion that the requested item is a travel product. It then replies with a 'Goto'

6 In contrast to mega search engines such as Yahoo, multi-agent systems allow distributed processing. Small but concise and well-maintained information sources are preferred. There is no urgency for the agents to complete their tasks and no effort is required from the users.

message containing the address of another company called Travel Listings. PTA then invites the Travel Listings Agent to visit by sending another ‘Get’ message. From its database, the Travel Listings Agent locates two suppliers - Air Net and Virgin Atlantic - and replies to PTA with their addresses. PTA then sends ‘Get’ messages in turn to Air Net and Virgin Atlantic. These supplier agents then reply with ‘Inform’ messages, containing a list of flight details that meet the user’s specifications.

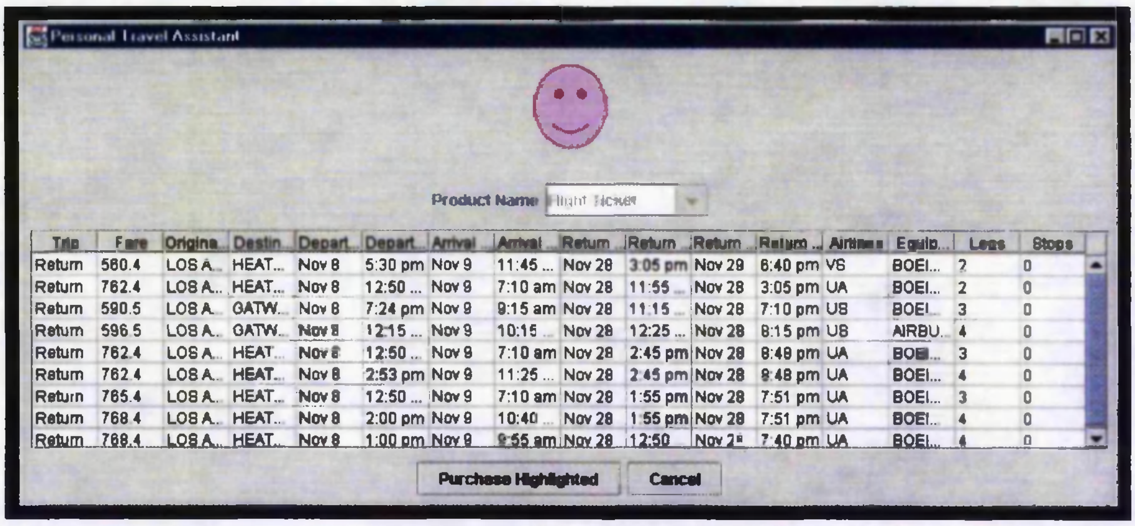


PTA	Universal Listings	Travel Listings	Air Net	Virgin Atlantic
Get	Goto	Goto	Inform	<Started>
Get	<Started>	<Started>	<Started>	
Get				
Get				
<Started>				

Figure 6.10 Snapshot of Agent Monitor

It is up to the invited agents to decide when to leave the host. Typically, the listing agents will leave after replying because they will not likely be questioned again. It is worthwhile for the supplier agents to stay and hold on to the current information in case PTA queries again for further information. The agent protocol and Agent Host should protect the client machine and provide fail-safe operations rather than relying on the goodwill of foreign agents.

Each agent will analyse the user’s specifications using its own ontology database. For example, the Universal Listings Agent will associate ‘Ticket Air’ with ‘Flight Ticket’ and substitute the product name in its reply. If the Travel Listings Agent understands ‘Flight Ticket’, it may reply with the same or a different, but more appropriate term. In this way, as long as some agents understand the phrase ‘Ticket Air’, a more popular term, used and understood by most agents, will be singled out automatically. PTA will hence end up using standardised terms through these multi-agent interactions.



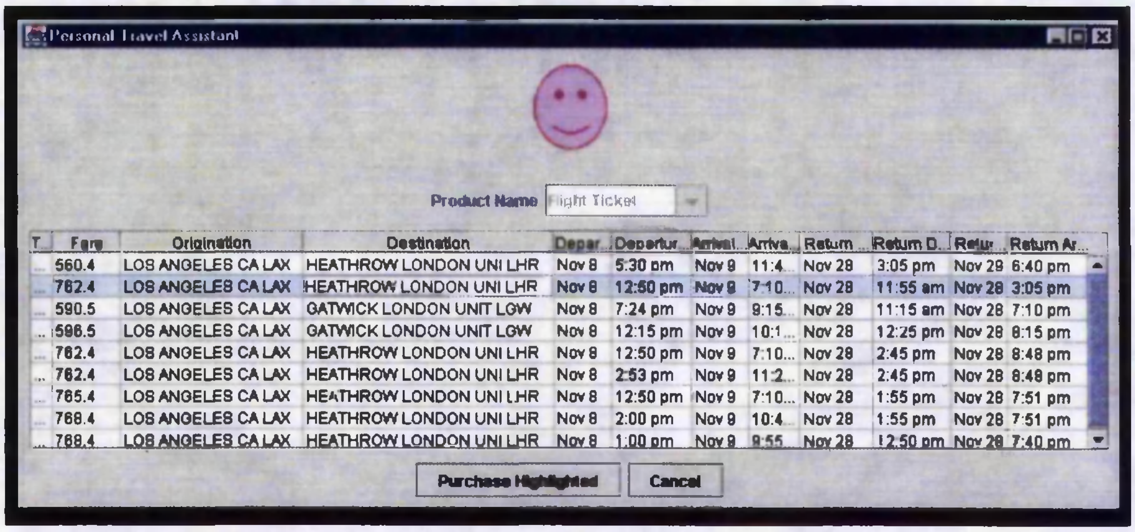
Trip	Fare	Origina...	Destin...	Depart...	Depart...	Arrival ...	Arrival ...	Return	Return ...	Return ...	Return ...	Airlines	Equip...	Legs	Stops
Return	560.4	LOS A...	HEAT...	Nov 8	5:30 pm	Nov 9	11:45 ...	Nov 28	3:05 pm	Nov 29	6:40 pm	VS	BOEI...	2	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	11:55 ...	Nov 28	3:05 pm	UA	BOEI...	2	0
Return	590.5	LOS A...	GATW...	Nov 8	7:24 pm	Nov 9	9:15 am	Nov 28	11:15 ...	Nov 28	7:10 pm	US	BOEI...	3	0
Return	596.5	LOS A...	GATW...	Nov 8	12:15 ...	Nov 9	10:15 ...	Nov 28	12:25 ...	Nov 28	8:15 pm	US	AIRBU...	4	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	3	0
Return	762.4	LOS A...	HEAT...	Nov 8	2:53 pm	Nov 9	11:25 ...	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	4	0
Return	765.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	3	0
Return	768.4	LOS A...	HEAT...	Nov 8	2:00 pm	Nov 9	10:40 ...	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	4	0
Return	768.4	LOS A...	HEAT...	Nov 8	1:00 pm	Nov 9	9:55 am	Nov 28	12:50 ...	Nov 28	7:40 pm	UA	BOEI...	4	0

Figure 6.11 Search Results

When the search is completed, PTA acquires a list of available tickets and presents it to the user in a table form. Attribute names or column names are likely to be propagated after the search process. Understood names are replaced or untouched, while new column names may be added. Each agent will add rows to the table when the flights meet the user's specifications. If the agent does not understand a column name, the field will be left blank.

In this way, there is no need to have a universally agreed-upon format on the database or a common ontology. The only requirement is that every supplier flattens all product features into a table form. For example, an indirect flight will show that a return trip has three legs instead of two rather than showing details on the arrival and departure information on the intermediate destinations. It is expected that the relational model should contain sufficient information, which is essential for users and agents to make intelligent decisions. Details that do not fit in the table can be supplied on request or attached to each row as additional informative resources to the user. In other words, PTA can virtually display the information in whatever format the supply agent submits such as rich text formats and graphics.

Just like the case in product specification, it is important that the display table is flexible. The window is resizable, the table is scrollable and the column width adjustable with simple and standard mouse manipulations. Figure 6.12 shows the user zooming in on the airport and timing details.



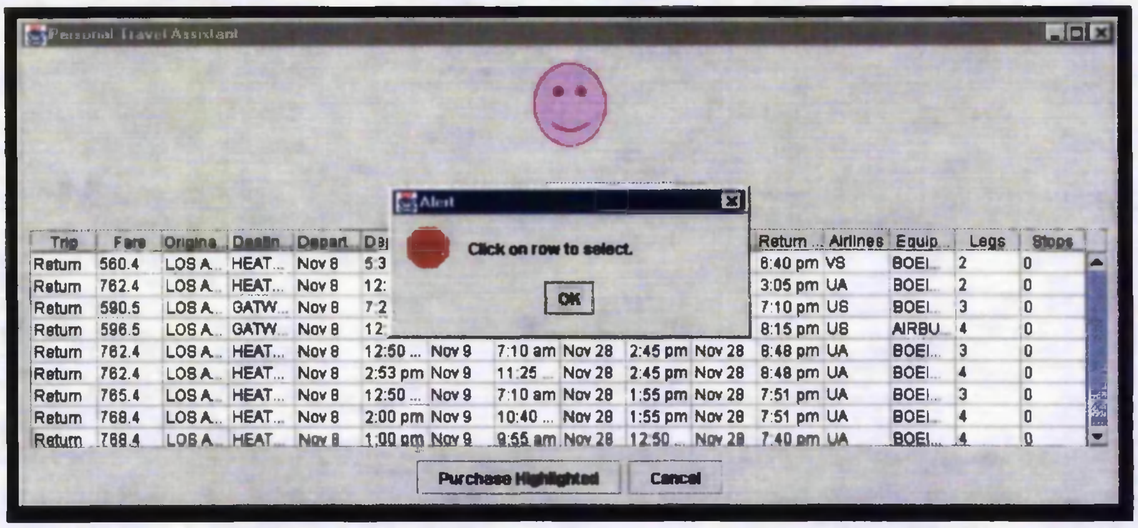
T...	Fare	Origination	Destination	Depar...	Departur...	Arrival...	Arriva...	Return ...	Return D...	Retur...	Return Ar...
...	560.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	5:30 pm	Nov 9	11:4...	Nov 28	3:05 pm	Nov 29	6:40 pm
...	762.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	12:50 pm	Nov 9	7:10...	Nov 28	11:55 am	Nov 28	3:05 pm
...	590.5	LOS ANGELES CA LAX	GATWICK LONDON UNIT LGW	Nov 8	7:24 pm	Nov 9	9:15...	Nov 28	11:15 am	Nov 28	7:10 pm
...	596.5	LOS ANGELES CA LAX	GATWICK LONDON UNIT LGW	Nov 8	12:15 pm	Nov 9	10:1...	Nov 28	12:25 pm	Nov 28	8:15 pm
...	762.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	12:50 pm	Nov 9	7:10...	Nov 28	2:45 pm	Nov 28	8:48 pm
...	762.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	2:53 pm	Nov 9	11:2...	Nov 28	2:45 pm	Nov 28	8:48 pm
...	765.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	12:50 pm	Nov 9	7:10...	Nov 28	1:55 pm	Nov 28	7:51 pm
...	768.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	2:00 pm	Nov 9	10:4...	Nov 28	1:55 pm	Nov 28	7:51 pm
...	768.4	LOS ANGELES CA LAX	HEATHROW LONDON UNI LHR	Nov 8	1:00 pm	Nov 9	9:55	Nov 28	12:50 pm	Nov 28	7:40 pm

Figure 6.12 Exploring Field Details

The user is not expected to go through the table in detail. PTA ranks the alternatives and the display order is arranged in a way so that the user will most likely pick the first row. This is especially true when PTA has acquired sufficient knowledge from other agents and starts to make increasingly competent decisions over time.

6.2.1.4 Decision

When PTA finishes its job, the user has to decide which product to buy in the sorted table. The user can either cancel the task, or click on one row to highlight it and then select the 'Purchase' button. If no row is selected, an error message will pop up as in Figure 6.13.



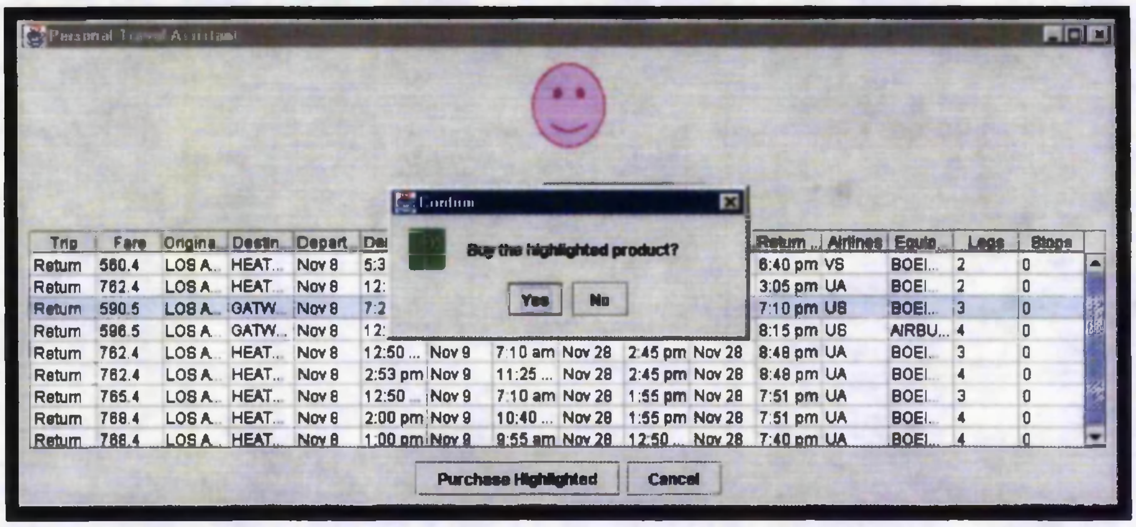
Click on row to select.

OK

Trip	Fare	Origina	Destin	Depart	De						Return ...	Airlines	Equip	Legs	Stops
Return	560.4	LOS A...	HEAT...	Nov 8	5:3						6:40 pm	VS	BOEI...	2	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:						3:05 pm	UA	BOEI...	2	0
Return	590.5	LOS A...	GATW...	Nov 8	7:2						7:10 pm	US	BOEI...	3	0
Return	596.5	LOS A...	GATW...	Nov 8	12						8:15 pm	US	AIRBU	4	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	3	0
Return	762.4	LOS A...	HEAT...	Nov 8	2:53 pm	Nov 9	11:25 ...	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	4	0
Return	765.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	3	0
Return	768.4	LOS A...	HEAT...	Nov 8	2:00 pm	Nov 9	10:40 ...	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	4	0
Return	768.4	LOS A...	HEAT...	Nov 8	1:00 pm	Nov 9	9:55 am	Nov 28	12:50 ...	Nov 28	7:40 pm	UA	BOEI...	4	0

Figure 6.13 Error Message — No Product Selected to Purchase

When the user selects a ticket, PTA will ask the user to confirm it before sending a 'Buy' message back to the supplier. The confirmation dialog is shown in Figure 6.14.

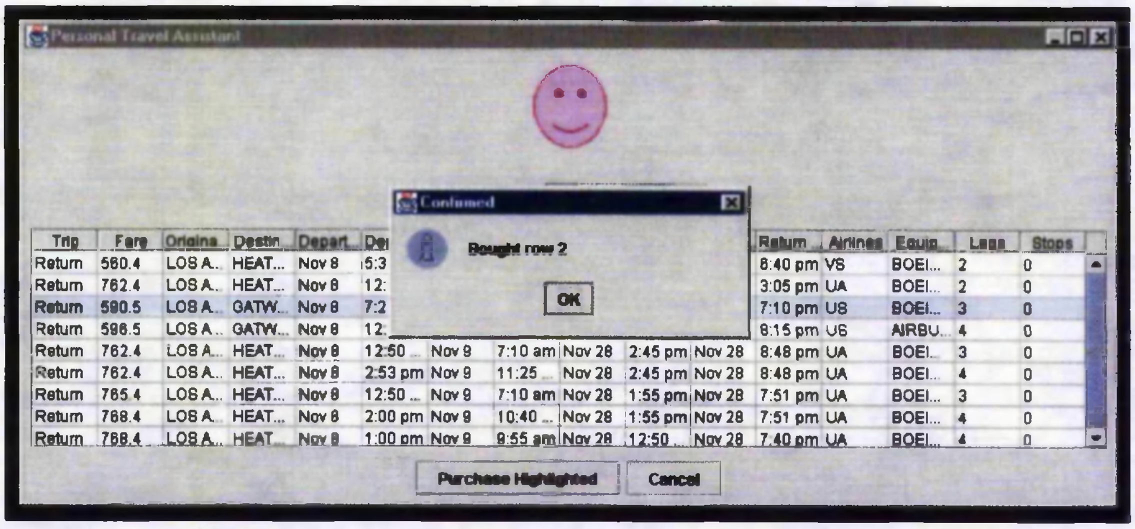


Buy the highlighted product?

Trip	Fare	Origina	Destin	Depart	De						Return ...	Airlines	Equip	Legs	Stops
Return	560.4	LOS A...	HEAT...	Nov 8	5:3						6:40 pm	VS	BOEI...	2	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:						3:05 pm	UA	BOEI...	2	0
Return	590.5	LOS A...	GATW...	Nov 8	7:2						7:10 pm	US	BOEI...	3	0
Return	596.5	LOS A...	GATW...	Nov 8	12:						8:15 pm	US	AIRBU...	4	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	3	0
Return	762.4	LOS A...	HEAT...	Nov 8	2:53 pm	Nov 9	11:25 ...	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	4	0
Return	765.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	3	0
Return	768.4	LOS A...	HEAT...	Nov 8	2:00 pm	Nov 9	10:40 ...	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	4	0
Return	768.4	LOS A...	HEAT...	Nov 8	1:00 pm	Nov 9	9:55 am	Nov 28	12:50 ...	Nov 28	7:40 pm	UA	BOEI...	4	0

Figure 6.14 Confirm Product Selection

When PTA finalises the purchase with the supplier agent, the result is reported back to the user. Figure 6.15 shows that the ticket is purchased successfully.

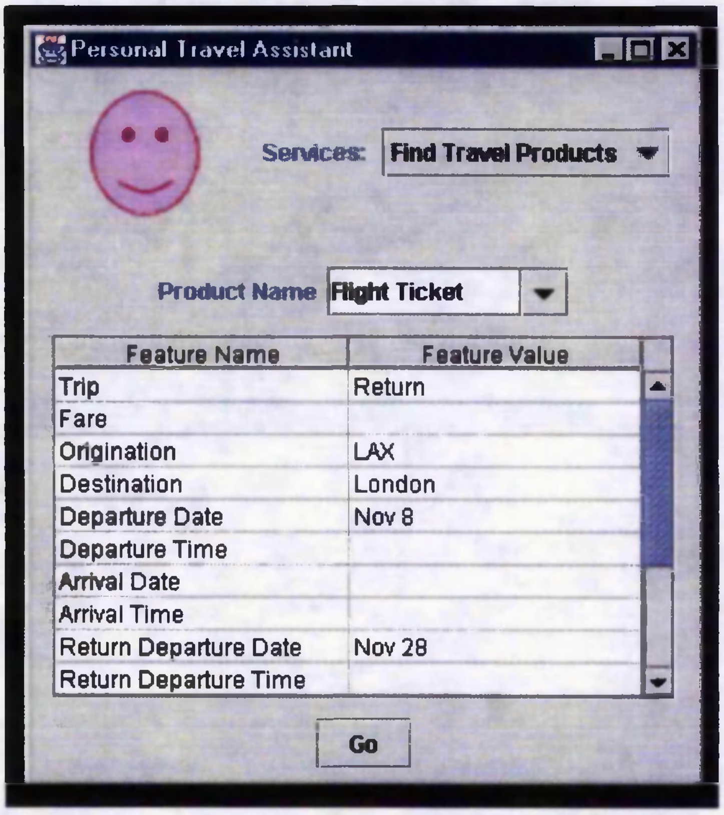


Bought row 2

Trip	Fare	Origina	Destin	Depart	De						Return	Airlines	Equip	Legs	Stops
Return	560.4	LOS A...	HEAT...	Nov 8	5:3						8:40 pm	VS	BOEI...	2	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:						3:05 pm	UA	BOEI...	2	0
Return	590.5	LOS A...	GATW...	Nov 8	7:2						7:10 pm	US	BOEI...	3	0
Return	596.5	LOS A...	GATW...	Nov 8	12:						8:15 pm	US	AIRBU...	4	0
Return	762.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	3	0
Return	762.4	LOS A...	HEAT...	Nov 8	2:53 pm	Nov 9	11:25 ...	Nov 28	2:45 pm	Nov 28	8:48 pm	UA	BOEI...	4	0
Return	765.4	LOS A...	HEAT...	Nov 8	12:50 ...	Nov 9	7:10 am	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	3	0
Return	768.4	LOS A...	HEAT...	Nov 8	2:00 pm	Nov 9	10:40 ...	Nov 28	1:55 pm	Nov 28	7:51 pm	UA	BOEI...	4	0
Return	768.4	LOS A...	HEAT...	Nov 8	1:00 pm	Nov 9	9:55 am	Nov 28	12:50 ...	Nov 28	7:40 pm	UA	BOEI...	4	0

Figure 6.15 Purchase Confirmation

The PTA will learn the 'consensus' ontology from various agents in this transaction. The attributes and the user's input are captured and stored for use on future bookings. Whenever the user wants to book travel products again, the Product Name combo box will show 'Flight Ticket' (instead of 'Ticket Air') in the pull down menu. When this item is selected, the attribute names learned from past experience and the user's previous specifications are displayed in the product specification table automatically, as shown in Figure 6.16.



Feature Name	Feature Value
Trip	Return
Fare	
Origination	LAX
Destination	London
Departure Date	Nov 8
Departure Time	
Arrival Date	
Arrival Time	
Return Departure Date	Nov 28
Return Departure Time	

Figure 6.16 Learned Product Specifications

6.2.2 Agent Communication Protocol

It has often been discussed whether an agent communication language is really necessary. Definitely, a comprehensive, rigid and fully standardised ACL will not meet the low-cost design philosophy unless it is available now at reasonable costs. Therefore, a pragmatic approach was taken to implement agent communication by standardising message protocols instead. Since all agents are written in Java, message objects can be passed directly from agents to agents. An agent message is defined in Java as follows:

```
class AgentMessage {
      AgentAddress src;
      AgentAddress dst;
      int act;
      Object obj;
      //… constructor
      }
}
```

The necessary information is the sender and recipient addresses. Agent addresses must be globally unique. URL provides part of the solution. An address consists of the full URL of the host where the agent can be found, and a filename of the class file. An example for the Universal Listings Agent address is:

http://www.universalistings.com/agent/www_universalistings_com_agent.class

Since the filename is also the mandatory class name of the agent object, it has to be derived from the globally unique host URL. This will eliminate the chance that two invited agents have the same class name, or identity, in the Agent Host.

The 'ACT' field specifies the message type, which is vaguely related to FIPA ACL performatives. The message content is simply a Java object.

The Universal Listings Agent only knows three acts. When a 'Get' message is received, the Java object contains a product specification. The response is either a 'Goto' message containing a list of supplier agent addresses, or a 'Refuse' message stating that no address can be supplied. This is represented in the following protocol diagrams.

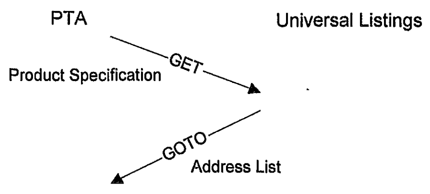


Figure 6.17 Get-Goto Protocol

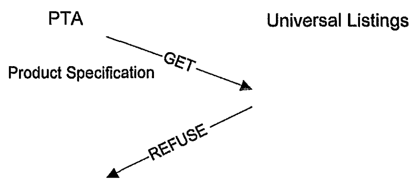


Figure 6.18 Get-Refuse Protocol

For a supplier agent, two more message types are needed for replying with a list of products and knowing what the user wants to buy. An 'Inform' message contains a list of products that meets the PTA's specifications.

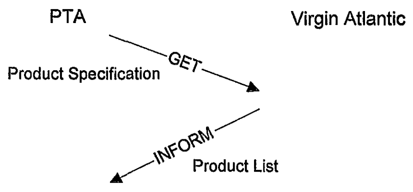


Figure 6.19 Get-Inform Protocol

When the user decides on a product, PTA sends a 'Buy' message that contains a specific product received from the supplier earlier. In response, the supplier agent checks if the product is valid and available. It will respond with either an 'Inform' or a 'Refuse' message to indicate the outcome of the transaction.

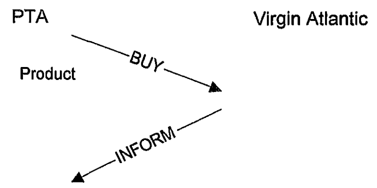


Figure 6.20 Buy-Inform Protocol

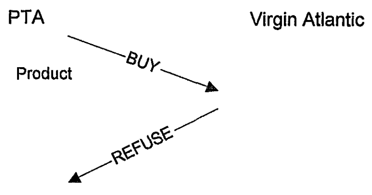


Figure 6.21 Buy-Refuse Protocol

It can be seen that agent communication can be simple, five message types can complete a ticket transaction:

Message Type	Meaning
Get	Product information
Goto	Address redirect
Inform	Positive
Refuse	Negative
Buy	Propose transaction

Table 6.1 Agent Message Types

There are only two types of message content involved. The product and product specifications share the same table form. The agent address is a two-part string as defined above. The only message content currently left out in the prototype is the payment method. The minimum will be a credit card number attached to the 'Buy' message, and a transaction tracking number attached to the 'Inform' message.

It can be seen that a simple protocol can be used in the early phase to enable an open multi-agent system. The standardisation effort is minimal. To avoid being obsolete, it is desirable for existing agents to learn or adapt to new protocols. To achieve this, the standard should include ways for PTA to receive refined protocols from the supplier as protocol objects. PTA then invokes the methods in the protocol object to complete the transactions. For example, the supplier may insist that the user must be personally involved in the final phrase of the transaction. This can be done by sending PTA a dialog object as part of the protocol. Once the protocol object is invoked by PTA, the user will have to respond before the transaction is final.

6.2.3 Ontology

Ideally, agents should use the same term to mean the same thing. For example, every agent should use 'Flight Ticket' instead of 'Airline Ticket' for product names. This will require much more standardisation work than protocols. It is also doubted if the standardisation approach is sufficiently flexible because agents should be allowed to learn new terms.

In the context of tickets, it is obvious to humans that 'From' and 'Origination' means the same thing. The narrower the context, the more likely it will be for humans and agents to use the same words. Therefore, the concept of ontology trees was suggested. For example, the Virgin Atlantic Agent defines a 'Flight Ticket' as:

```
"Flight Ticket" (
      ticket (
            fly
            air
            airline
            flight
            plane
      )
)
```

This definition means that 'Flight Ticket' means 'ticket' AND one of the words 'fly' OR 'air' OR 'airline' OR 'flight' OR 'plane'. The order of words is unimportant and unknown words are ignored. The syntax can be fully nested to provide a complex tree definition. For example:

```
"Return Departure Date" (
      "return departure date"
      return (
            date (
                  depart
                  leave
                  from
            )
```

```
                day (
                      depart
                      leave
                      from
                )
                day
                date
          )
    )
```

'Return Departure Date' matches 'return day' or 'leave date on return'. In this way, it is highly likely that agents will understand each other without using exactly the same terms.

Each agent is armed with an independent ontology database file containing definitions as above. On initialisation, tree branches are built from the definitions in files and then put into an ontology map which enables fast matching. When a foreign term is received, the phrase is parsed into individual keywords. If a keyword is contained in the ontology map, a set of ontology trees can be found. All trees in the set contain the keyword in their branches. The complete foreign term is then compared with each tree in the set to find the first definition that matches. This pre-processing strategy simplifies programming since one can write codes as if all other agents use the same term for the same thing.

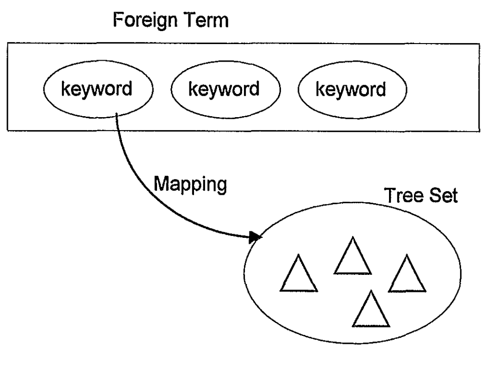


Figure 6.22 Ontology Matching

PTA does not have an ontology database at the beginning and it is not necessary to have one. It is, however, advantageous for PTA to build an ontology from responses of other agents. For example, agents may insist that their terms are the most appropriate and do not want to change. Hence, PTA will continue to see different terms such as 'Flight Ticket' and 'Airline Ticket' interchangeably, depending on who supplies the ticket. By building an ontology database of its own, PTA will use consistent terms. The user will then see only one term for a particular product. More importantly, PTA acts as a depository for ontology trees. Conflicting definitions can be detected and resolved by sending error messages to agents involved.

6.2.4 Learning

The goal of PTA is to learn what a ticket is and how to select the best available ticket for the user. As PTA will start as a clean sheet, the knowledge has to be obtained from other agents. By pooling knowledge from many agents, the learning process can be speeded up.

Central to learning is the definition of a product specification:

```
class productSpec {
      String      productName;      // E.g. Flight Ticket
      String[]    columnNames;      // Table heading
      List        maps;             // Table Rows
      List        rules;
      List        importance;
      //... methods
}
```

This definition contains the mechanism for two types of learning described in the following sections.

6.2.4.1 What is a ticket?

The 'columnNames' field and 'maps' field relate to the table as seen before in the user interface. The array of columnNames is the attributes of the product. The table data are represented by a list of maps, each representing one row of data. A map is made up of key-value pairs in which the keys are the column names. There are also some hidden keys that are not included in the array of column names. An example of a hidden key is the address of the agent that supplies the ticket, since each row of data can be offered by different agents.

Initially, the user requirements are translated into a table with one row. That is, the table consists of the ideal product if it can be found. The same specification object is passed from agent to agent, each may add products that are close to the user's ideal. Therefore the number of rows in the table grows. The number of columns may also grow when agents find that the number of attributes they used is larger than the number of columns in the tables they receive. That is, agents' definitions of a product vary in details. PTA ends up with a table of potential products from different suppliers. The final column names are the most detailed attributes that PTA can find. All names are likely to be popularly used in the industry. Therefore it can be said that PTA learns what a ticket is from collaborating agents.

6.2.4.2 How to select tickets?

When agents receive product specifications, they may append rule objects to the specifications just as they append ticket data to the table. Given a rule object from some agent, PTA can do three things as defined in the Rule class definition:

```
abstract class Rule {
      // ... internal objects and methods
      public void score(){}
      public void pass(){}
      public void calibrate(Object selected) {}
}
```

Each rule is allowed to work on the complete product table to make decisions. PTA can ask whether any product is rejected by invoking the pass method. PTA can also ask for the score of each product by invoking the score method. When the user finally decides on a product, each rule is informed of the choice via the calibrate method.

Each rule then adjusts its internal variables so that the scoring and rejection algorithms will be improved. The Rule class is declared abstract because rules from different agents will implement their own scores, pass and calibrate methods in different ways.

The pass method largely relates to non-compensatory rules that select or reject base on a particular attribute. For example, 'I am willing to consider auxiliary airports if the price is significantly cheaper'. This rule will reject all flights using auxiliary airports, with the exception that the fare is calculated to be lower than a threshold.

The score method is largely related to compensatory rules that provide scores for each attribute, allowing one attribute to compensate for another in the overall score. For example, 'I prefer direct flights'. It is not possible to reject all indirect flights because this is probably not what the user wants. Instead, it is necessary to provide a tentative score for direct flights so that the higher score in direct flights will compensate for lower scores elsewhere.

Given the user's final choice, the 'airport rule' can check its correctness. If the user's choice was rejected by the rule, it means that the rejection threshold has to be lowered. Similarly, if the user's choice is an indirect flight, the tentative score for direct flights in the 'direct flight rule' has to be lowered.

Rules are intelligence directly transferred to PTA, which has to combine all rules in order to rank the products. An effective combination simply adds the scores of all flights that are not rejected by any rules. PTA learns by measuring performance of individual rules over time based on their empirically determined accuracy. To this end, each rule is associated with an importance value. If a rule consistently makes correct rejections, it will be given a higher importance value. Similarly, if a rule gives the user's choice highest score, its importance will be raised.

More complicated forms of learning by the PTA have been considered, but the simple strategy adopted fits the consumer decision-making model discussed earlier. It should be noted that there is no limit to the amount of intelligence in the rules that are transferred to PTA. A rule object may contain an entire expert system shell and a large rule base.

6.3 Evaluation

How can the benefits and drawbacks of a multi-agent system approach be assessed in an objective manner? As discussed in chapter three (section 3.7.3.3), there is no formal technique for multi-agent evaluation available so far. Hence, evaluation seems to be a problem in itself for such a new technology.

In trying to find ways of assessing the contribution of a multi-agent system, one will argue that this is only really with experience. Strictly speaking, it is necessary to know how good or bad the multi-agent approach is in comparison with the best alternative solutions. Nearly always, the answer is unknown because resources do not allow multiple approaches to be tried. Moreover, in the real world there is usually a number of different ways of solving a particular problem and it is impossible to compare them in any objective fashion.

One may say that methods and tests are needed to verify and validate multi-agent systems. However, success or failure is not always related to technical superiority, so it is hard to measure prospects, chances and relevance of different techniques and technologies. Again, the comparison with expert systems or neural nets is illuminating. In the former case, the benefits were expected to be in the area of efficiency (because of fewer experts needed) but turned out to be in the area of effectiveness (with better use of existing experts and more widespread distribution of expertise). In the case of neural nets, some comparisons have been made with equivalent mathematical and statistical techniques, and the benefits are less in performance and more in the time and skill required for development and extension.

In the absence of an objective method, some criteria for judging the strength of the PTA multi-agent approach have to be defined. In doing this, it is necessary to revisit some of the claims, challenges as well as the design principles discussed in chapter five.

6.3.1 Meeting Customers' Needs

It has been claimed that agents have advantages in meeting customers' needs in the electronic travel market because it reduces user involvement, provide intelligence support and save bandwidth. Now let us evaluate PTA and see what benefits it offers to the travellers.

- Does PTA reduce user involvement?

The biggest advantage of PTA is its ability to automate previously manual operations. PTA demonstrates that travellers are able to delegate the task of an air ticket purchase with minimal human intervention. Though the prototype is primitive to start, it has the prospects of evolving into a fully competent travel assistant through its learning ability. It may take some time for PTA to mature, but the potential is certain. Even when PTA is relatively unsophisticated, its processing ability is beyond what a human can realistically do, and this has the major impact on reducing workload.

- Does PTA offer intelligence support?

The ability to *learn* is the major strength of PTA. Though the issue of personalisation was not addressed, PTA demonstrates its ability to learn and acquire domain-specific knowledge, and to a certain extent ontology. This learning ability, however, is not restricted to domain knowledge acquisition. PTA can apply this ability in various ways and will eventually learn about the make-up of their user's preferences as it matures.

- Does PTA offer speedy service?

With PTA, travellers no longer need to access the Web themselves because all tasks can be delegated. Hence, no time will be wasted in browsing and they are freed to concentrate on their core activities. In addition, PTA has incredible processing speed beyond the capability of any human. PTA also works by exchanging messages and information with other agents, and hence no more browsing is involved and this proves to be more efficient because a lot of bandwidth is saved.

6.3.2 Value-Added Solution

- Does PTA offer a value-added solution to existing approaches?

This simply requires the definition of value-added which could range from excellent (where no solution is probable without a multi-agent approach), through minimal (where a multi-agent solution is marginally of value) to poor (where conventional approaches offer better solutions). It is not exaggerating to say that there is no other approach that will deliver the combined benefits of a multi-agent approach. As mentioned in chapter five, a multi-agent system is the only workable solution if the highest degree of user automation is to be achieved. If the aim is to make searching easier, then there are numerous alternatives available, like a mega-site agent and an open market using XML standard. Apart from all the limitations associated with these alternatives, PTA outshines these alternatives by one outstanding advantage – it belongs to the traveller, and hence trusted. Instead of relying on the goodwill of suppliers, PTA acts solely for the interests of its owner. Rather than just making information search easier, it makes search totally disappear. What makes it distinctive is that it puts the power into the hands of the travellers directly. It is these combined benefits that make the multi-agent approach stand out from the rest.

6.3.3 Design Principles Addressed

PTA fulfils its design objectives by being:

- Simple: PTA is written in pure Java. It demonstrates that complex agent building tools are not necessary. Java packages are very useful. With the availability of extensive API calls, the objects built are simple and small.

- Easy to use: travellers interact with PTA via an entry form where they specify their requirements. PTA is straightforward and it has no non-functional cosmetic features to divert the travellers.

- Practical: PTA and Agent Host can actually be deployed 'as is' immediately. The supplier agents need only a bridge such as JDBC to query the suppliers' databases.

- Useful: PTA, in its current state, has the potential of scrapping many products from many suppliers, filtering most of them and suggesting the best option.

- Flexible: the strength of collaborative learning puts PTA in an advantageous position. Due to its low-cost requirement, the software is very simple to start with, but PTA demonstrates its ability to learn new products, decision rules, and possibly protocols.

- Extensible/Scalable/Compatible: PTA runs on HTTP servers, so it can easily be upgraded to more complex technology in the future because it is not attached to a custom infrastructure.

- Cheap and low-cost: PTA fully satisfies this criterion by exploiting existing technologies such as HTTP servers and public domain software like Java.

6.3.4 Challenges Tackled

Instead of skirting around the interoperability and ontology problems, PTA tackles them in a direct and positive manner. It is therefore expected that PTA would successfully live up to its challenges as a <u>transitional</u> product. With all its advantages, PTA holds great potential in accelerating the adoption of agents in the transitional period.

6.3.4.1 *Interoperability*

The idea of exploiting existing HTTP servers renders itself a very promising approach to tackle the interoperability problem. In the PTA prototype, HTTP servers are used to download suppliers' agents onto the customers' computers. This allows dumb servers to make use of agent technologies and the prototype clearly shows that the idea works well.

PTA also demonstrates that the standardisation of a simple set of protocols will be quite sufficient for communications between buying and selling agents in order to handle everyday electronic commerce activities. Instead of taking the extreme approach of creating a full-blown agent communication language, such a pragmatic approach may prove to be more useful in the initial period.

6.3.4.2 *Ontology*

While this might be quite unintentional, PTA comes up with a workable solution towards the ontology problem. This is done by a decision-tree classification technique via multi-agent learning (see section 6.2.4). In the PTA system, each agent builds up its ontology by sharing, gathering, refining and extending its own ontology within the agent community. New vocabularies are added to the ontology if the agent finds them to be valid. This approach works well in the PTA prototype.

6.3.4.3 *Legacy Software Integration*

One of the reasons that Java is chosen as the development tool is that it is a powerful language that allows for numerous extensions on the prototype. As mentioned earlier, the Java Database Connectivity kit supports connection to any relational database, which is a very powerful feature as far as legacy software is concerned. Though this capability has not been implemented into the current version of the prototype, the addition of such a bridge is by all means workable.

Chapter Seven

Impacts on the Travel Industry

The growth of the Internet and its potential as an electronic commerce channel was surveyed in chapter four and there appears to be sufficient evidence that travel businesses were migrating online rapidly. Just by looking at the title of the thesis, it suggests the inter-relationship among three entities - Electronic Commerce, Travel and Agents. Travel, by itself, is an ever-changing and highly complex industry. Adding electronic commerce to the scene makes it more unpredictable and it is already challenging a lot of traditional marketing theories. Now, agents move onto the picture – how are they going to change Web commerce in travel?

> *'[...] it often is impossible to identify the effects of a technology. Consider the now ubiquitous computer. In the mid-1940s, when digital computers were first built, leading pioneers presumed that the entire country might need only a dozen or so. In the mid-1970s, few expected that within a decade the PC would become the most essential occupational tool in the world. Even fewer people realised that the PC was not a stand-alone technology, but the hub of a complex technological system that contained elements as diverse as on-line publishing, e-mail, computer games and electronic markets' (Zachary, 1996).*

Though it is difficult to have any 'black-and-white' predictions on the future, it is likely that the relationship among the various players would change. Agents may bring about a situation where players on all levels of the value chain have an equal chance to play, while success or failure will depend on an individual's ability to adapt to the changing environment in an intuitive way. As a general statement, agents will make the 'virtual' world 'realistic' by allowing users to work in the same natural, usual manner as they do in the physical world. That is, once more, they are able to delegate, let specialised groups sort things out, concentrate on their core activities, and live in a world of 'survival of the fittest'.

In this chapter, the future impacts of agents[1] at each stage of their development along the migration path will be studied. First, a rough chronology of expected developments is sketched to give an idea of the micro impacts, both short and long term, with respect to different groups of players. Then, the macro impacts on the electronic market as a whole will be discussed to complete the picture.

7.1 A Stepping-Stone

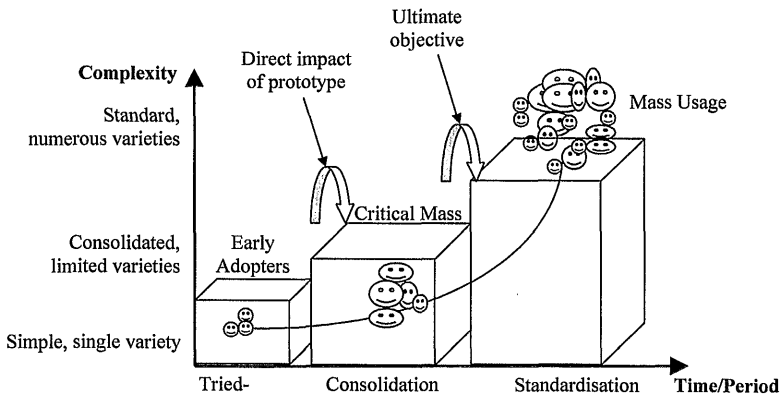


Figure 7.1 Stepping Stones on Migration Path

The prototype may be considered as the first <u>stepping-stone</u> towards:

[1] The word 'agents' always refers to software agents in this chapter. Online travel agents, which have traditional physical storefronts are referred to as travel agents. Online virtual travel agents, e.g., Expedia, are those that have no physical existence in the offline world.

1. using more sophisticated agent software in the short term; and
2. developing open standards in the long term.

Its aim is to accelerate the penetration of agents in the electronic travel market by making things easy, giving a 'product tasting', and causing no fuss. The ultimate goal is to reach the highest step where agent technology will be used transparently by everybody everywhere, even without them knowing that they are using it at all.

> *'Whenever people learn something sufficiently well, they cease to be aware of it. When you look at a street sign, for example, you absorb its information without consciously performing the act of reading ... Computer scientist, economist, and Nobelist Herb Simon calls this phenomenon "compiling"; philosopher Michael Polanyi calls it the "tacit dimension"; psychologist TK Gibson calls it "visual invariants"; philosophers Georg Gadamer and Martin Heidegger call it "the horizon" and the "ready-to-hand", John Seely Brown at PARC calls it the "periphery". All say, in essence, that only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals' (Weiser, 1991).*

Now, let us re-walk the migration path portrayed in chapter five. The path is divided into two periods:

- *'short term'*, relating to the migration from the 'Tried-and-Tested' to the 'Consolidation' period;
- *'long term'*, relating to the graduation from the 'Consolidation' to the 'Standardisation' period.

The beginning of the short term period is characterised by some early adopters who are most likely to be tourists, small and medium sized travel suppliers or intermediaries such as smaller hotels, B&Bs, small specialised travel agents, etc. Local, regional and national tourist offices may also show interests. These pioneers are believed to gain more exposure from routine usage of PTA. They will be able to provide better customer services that lead to higher customer satisfaction. The results will be growth in business and profits. Once the benefits of PTA are revealed, more users will try out the new technology. In the short term, it is believed that the agent software is originated from technological companies that aim to promote agent technology and their brands. It may also be available from non-profit foundations which develop public domain software. Other interested companies may be willing to give away free software as long as their business models justifies this as a sound marketing tactic. The end of this period will be a period of consolidation where a critical mass will be formed. Taking a broader view beyond the travel industry, it is expected that the popularity of agents will be intensified by the adoption of other industry sectors which sell simpler commodity products[2] such as music CDs, books, etc.

The continuous usage of agents by users in the 'Consolidation' period leads to increase in confidence and experience. Hence, it is likely that a proliferation of products will emerge when de facto standards start to evolve. At the beginning of the long term period, the different agent types of the short term, will now start to mature. Significant usage of these systems should be expected across the whole industry after major large suppliers join in. More agent applications will be offered by a significant number of developers/vendors because of the prospects and profits of the technology.

It is also by this time that the outlines of the most important agent-related standards should become clear. However, instead of having standards being imposed in a 'top-down' manner, it may happen in very unexpected ways, for instance, some products

[2] The popularity of agents in commodity markets has been proven in a lot of academic research projects such as those from MIT. Kasbah (Chavez & Maes, 1996) is a typical example. It creates a 'virtual' marketplace where students can exchange products by creating their own buying and selling agents. Unfortunately, the project is limited to a consumer-to-consumer application context.

that are successful become the de facto standards. It is believed that the architecture that is used by (or best supports) the most popular product will become the prevailing architecture, and will set the standard for future developments and applications.

7.2 Micro Impacts on Key Players

PTA is predicted to *reinforce* many of the trends that already exist in the electronic travel market. Though all the traditional key players in the industry are predicted to stay, an evolution of their respective roles in the value chain will happen. PTA may also disrupt the balance of power among different groups of players and *transform* the rules of the game. It may also *create* the need for new players or cause existing players to evolve into new forms. It is worth remarking that, what we mean by 'impacts' is an indication of how PTA is going to change the way that these players conduct business in a multi-agent context, under the consumer-driven approach. In other words, how these players evolve to adapt to user automation by taking advantage of the functionality of a multi-agent system. It is different from how these players make use of PTA for marketing purposes which should be obvious given all the general advantages of software agents, such as personalisation, intelligence support, etc.

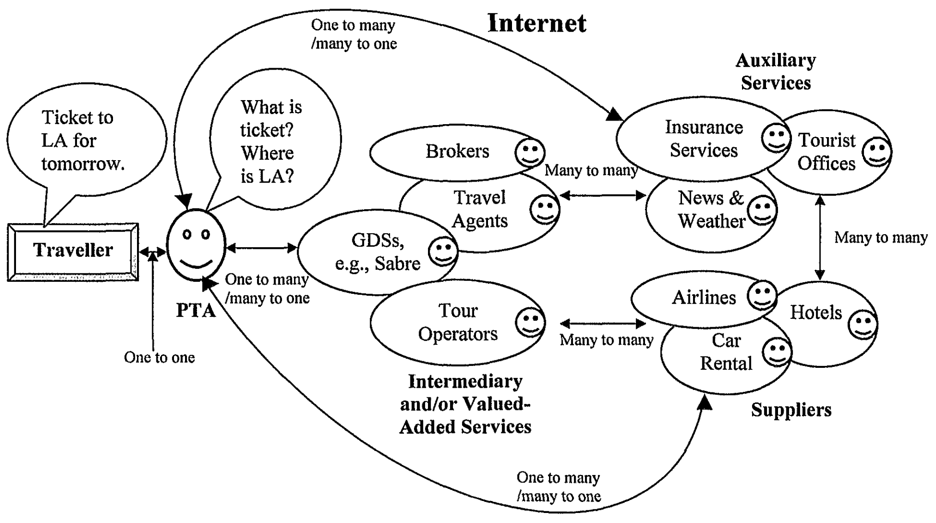


Figure 7.2 The PTA-Mediated Electronic Travel Market

As seen above, PTA can interact with players at all levels in the online travel market. It maintains a one-to-one relationship with the traveller. It has the flexibility of contacting the travel suppliers directly, or it can select specialised brokers or intermediary services for a more complicated task to spread the workload. Below, the impacts of PTA with respect to each of these groups of players are analysed:

- Tourists/customers;
- Suppliers/producers, e.g., airlines, hotels, car rental companies, etc.
- Intermediaries – travel agents, GDSs, etc.;
- Auxiliary service providers, e.g., tourists offices, content sites, etc.
- Newcomers – Yellow Page agents.

7.2.1 Tourists

Customers will certainly stand out as the winners of the game. PTA, being cheap, easy and practical, will accelerate the penetration of software agents via a consumer-driven approach. Customers will certainly benefit from PTA in the immediate term. The most important impacts are:

- Choice, and
- User empowerment.

The new technology broadens their choice and increases their ability to obtain products and services that exactly match with their needs and preferences. PTA disrupts the balance of power between the suppliers, the intermediaries and the customers by putting the control directly into customers' hands. In the long term, PTA will deliver a powerful increase in that twin driving force of commerce: increased customer choice and improved value for money. This *customer empowerment* is quite unprecedented in the history of tourism. It has macro impacts on catalysing the evolution of the tourism *market* into a customer-driven market in the long term. Apart from transforming the market, PTA will also impact the *product*. It is likely that highly customised packages will be offered to suit the individual taste of the customers. Customers' PTAs will specify what they want and the agents of suppliers or brokers will dynamically aggregate the products to suit the demand. All these macro impacts will be discussed further in the next section.

7.2.2 Travel Suppliers

Let us recap the recent developments and market strategies of this group of participants. The leading incentives for suppliers to develop Web sites are two-fold:

1. to have a direct link to consumers; and

2. to avoid the commissions paid to travel agents, thus reducing distribution costs.

There are numerous examples to support this. British Midland was the first airline to create a booking system for its flights over the Internet, closely followed by Alaska Airlines in the United States. Since then, there has been a profusion of airline sites on the Web. Some recent development in the airline sector clearly showed the airlines' intention of selling direct to the customers to lighten their travel agent overhead cost load. In 1998, most major airlines decided to cap the commissions they paid online travel agents at 5%, or US$10 maximum per ticket. This US$10 cap was less than what the airlines pay 'real' ('face-to-face') travel agents, which was 8%/US$50 on most round-trip domestic tickets. Recently, many major airlines started to encourage their customers to book online - but on their own Web sites, not the online agents'. They offered cyberfares, Internet-only pricing, two-for-one deals and free frequent flyer miles in order to entice their customers to book online[3]. This dual-action strategy - commission cap plus Web pricing - makes it clear that airlines want to sell tickets online to their customers themselves rather than via travel agents who may steer their customers to a better deal.

Within the hotel industry, large chains are creating their own sites, such as Hilton and Marriott. More importantly, there are new services that enable customers to access hotel room inventory information and conduct reservation transactions electronically. For example, Pegasus Systems[4] processes Internet hotel reservations via its online booking service - TravelWeb. The company maintains a database of more than 28,000 hotel properties (PhocusWright, 1999). Apart from providing direct access for the customers, it opens up a new distribution channel especially for those small

[3] American Airlines (1996) has put deep discounted fares for weekend travel up for sale exclusively to Internet users. These fares are around 70-80% lower than the carrier's already discounted 21-day advance purchase fares. United (1998) has announced a bonus of 20,000 frequent-flier miles (nearly enough for a round-trip ticket) to customers who use its Web site to book 10 trips. Northwest (1998) is luring frequent fliers to its Web site by making frequent-flier award travel bookable online. Delta (1998) has floated the idea of Net-only discounts for certain trips booked exclusively at its Web site.

[4] Pegasus Systems was formerly known as THISCO (The Hotel Industry Switch Company).

establishments that were previously left to their own means because they could not afford the GDSs. WorldRes is another service that provides hotel reservation facility on the Internet without the need of going through a GDS. The system is based on Internet database and client server technology, and allows hotels to update rates, availability and content via a Web browser interface. WorldRes maintains multiple connections with partner sites such as Travelocity and Preview Travel of which the hotels can choose where they want to receive their reservations (Dombey, 1998). However, both Pegasus and WorldRes still have to rely on the supply of related information on partner sites to add value to their hotel bookings.

How is PTA going to accelerate this trend? It is necessary to sidetrack a bit to talk about the role of the GDSs. Suppliers are still maintaining their link with the GDSs instead of going full swing to direct sale. Clearly, suppliers are trying their best to reach out to their customers as direct marketing in the 1980's has proven to be an effective marketing tool to enhance customer loyalty and increase sales. So, why are they still 'inside' the GDSs? Simple enough, the GDSs represent one single source of aggregated data in a global way which is unsurpassable by any single site. As mentioned, the Internet, though offering an additional global distribution channel, has, because of its open nature, information sources distributed in an unmanaged manner. Integrating disparate information sources becomes a priority to make searching manageable to the customers. There are two major barriers that cause suppliers' hesitation:

- they need to be found;
- they need to offer customers an alternative to integrate disparate information sources.

Here, the greatest impacts of PTA are:

- <u>distributed computing</u>, and

- proactive marketing.

PTA provides an alternative solution to the GDSs with the advantage of distributed computing. Since customers' PTAs will integrate the information sources on distributed databases of various suppliers, it makes the option of selling on suppliers' own Web sites feasible. Instead of having their data maintained by GDSs' centralised databases, suppliers have the flexibility of manipulating their own data. An additional impact is active marketing. For the travel suppliers, business is getting intensively competitive on the Internet. Instead of passively waiting for customers to track down their Web sites and products, supplier agents can take the initiative to visit customers' PTAs and push customised materials to increase awareness and motivate customers. The additional ability of addressing customers' needs directly might contribute to customer loyalty.

Among all the suppliers, airlines may be slower than the others to make full use of PTA because of their own interests in the GDSs and their closer link to the travel agents. It is unlikely that airlines will get out of their GDSs in the short term. However, the impacts on other suppliers like hotel chains and car rental companies are more direct. It is predicted that their dependence on travel agents and GDSs will be reduced significantly in the short term. Hotel representation groups like TravelWeb should be in the best position to pick up the benefits of software agents. They will find them extremely useful because data needed from other segments to combine with a hotel booking are now available without relying on the relationship of partner sites[5]. In the long term, airlines may eventually get out of the GDS distribution systems. This is the biggest impact that PTA will create – the *feasibility of distributed databases*.

For independent small suppliers that have no affiliation with any reservation system, PTA's impact will be felt by levelling the playing field. These smaller suppliers will

[5] For example, Pegasus Systems has partner sites such as Expedia and Preview Travel.

be able to gain more exposure because it is now easier for customers' PTAs to locate them. It is likely that there will be an increase in the number of small service and content producers, and hence a wider diversity of goods and services. As opposed to the prediction that airlines or mega travel agents (Jupiter Communications, 1999b) will eventually squeeze out the independent smaller suppliers such as single-product, single-market sites, there may be the emergence of a more 'assorted' market as smaller sites become more 'visible' and financially viable. This will have a macro impact on reducing the pressure of market consolidation in the long term which will be discussed in the next section.

7.2.3 Intermediaries

The sudden potential for intermediaries, brokers and other go-between services seems quite ironic, as only a few years ago media and research reports suggested that the Internet would spell the end of such 'middlemen' and intermediary services. When all the information and services one could possibly want are just a click away on the Internet, who needs brokers?

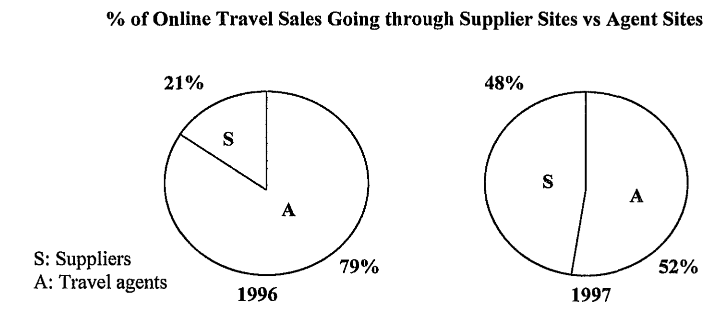
However, intermediaries exist in the real world for a reason, and those reasons do not go away on the Internet. Obviously, middlemen who are merely distributors will be less needed (e.g., booking travel agents). It is expected that the *types* and *nature* of intermediaries will evolve to suit the customer-driven market.

7.2.3.1 Independent Small Travel Agents[6]

After the launch of the commission cap by major airlines, it is certain that online travel agents will no longer be living on airline commission revenue in the future. There are clear trends that travel suppliers started to go to direct customer sales. The share of direct sales by service providers (as opposed to those sales via online travel

[6] These are those small- or medium-sized travel agents that have migrated online, but have their traditional physical locations in the real world.

agents) grew from 21% in 1996 to 48% of total online sales in 1997 (Jupiter Communications, 1999b).



Source: Jupiter Communications (1999b)

Figure 7.3 Suppliers Moving to Direct Online Sale

Apparently, travel agents have to sell value-added services to retain customers in the absence of airline-subsidised revenue. Hence, the identification of market niches will become the crucial point for survival. For example, adventure travel specialists and specialty retailers are packaging and promoting ever more specialised holidays than the old-fashion 'air and hotel' bookings. In fact, this is what customers are demanding. Travel agents will best survive in these highly specialised markets such as adventure tours like mountaineering trips to Nepal or the Himalayas, specialty or sports tours like ski, horse-riding, tennis and golf, nature or eco tours like trips to the Costa Rica's rainforests.

Those travel agents that remain will be travel professionals, adaptable to change, not simply booking processors. And as the processors fall by the way, more opportunities will arise for the professionals. There is an upside to these travel agents. The travel jobs of the future are going to be more knowledge-based. Travel agents are going to be more knowledgeable of destinations, cruises, and specialty tours. They are likely to develop long term relationships with clients.

In what way will these travel agents evolve to work in a multi-agent environment? As mentioned, the evolution of travel agent services is already underway in a positive fashion and one can find extensive evidence on the WWW. In the immediate term, PTA will intensify this evolution by giving the small specialised travel agents more exposure. Basically, the core service of these travel agents will remain the same, i.e., advise on complex tours. The major difference lies in the interface, which is now agent to agent. It is predicted that customers will use PTAs for information search, so these travel agents may take a proactive approach to reach these customers, e.g., use their software agents to make readily 'bundle-up' specialty packages available for free download on their sites. Another approach is to use software agents as sales agents to aggregate products dynamically to meet the needs of customers' PTAs on their visits. These travel agents may also use software agents to target their customers aggressively. Software agents can 'wrap up' packages, contact potential customers' PTAs and upload themselves onto customers' computers.

7.2.3.2 Virtual Travel Agents[7]

The availability of the booking interface to GDSs has created a new group of players in the last three years such as BizTravel.com, Microsoft Expedia, Internet Travel Network, Preview Travel and Trip.com.

There is widespread speculation that their market share will continue to increase in the future and the market will be increasingly dominated by this group (Jupiter Communications, 1999b). However, there is a common recent trend for these mega intermediaries to move away gradually from predominantly air ticket sales to promote 'packaged' travel products, including package holidays and cruises. For example, Expedia has re-designed its page to highlight 'special package offer' to get rid of distressed inventory.

[7] These are those new entrants into the online travel market. They do not have a travel background nor do they have a physical location.

How will PTA impact this group? In the short term, PTA will increase market participation because it makes smaller businesses more viable. However, as long as these players are enjoying the data source from the GDSs, the consolidation situation may continue to prevail for some time. In the long term, as PTA continues to attract more small suppliers, and at the same time, when major suppliers eventually get out of their GDSs, a more balanced marketplace will emerge. These joint forces will significantly broaden the market base and the large intermediaries may evolve into new specialised services. It is possible for these intermediaries to take advantage of their buying power, e.g., buying bulk inventory for resale to consumers, or to act as centralised markets for suppliers to get rid of distressed inventory.

7.2.3.3 GDSs

There are two key players who have a direct link with GDSs: (1) travel suppliers, and (2) travel agents. Now, let us trace back what has been changing on the Web with respect to their relationships so far.

How GDSs respond to the threat of disintermediation so far? With the popularity of the Internet, many suppliers started to reach out directly to customers. Like other travel suppliers, GDSs have exploited the Web as another means of distribution. The GDSs have created Web-based booking interfaces with their databases, which have signalled a key change in the direction for both travel distribution and a riposte to the threat of disintermediation. With the exception of Galileo[8], all GDSs have their Web presence through their own branded sites. They are in a perfect position to partner with companies strategically to consolidate their position. This is already evident in the market. Sabre has crossed the line into the retail world by launching Travelocity in 1996 and generated US$285 million in 1998 (Sabre, 1999). It has since acquired 25 powerful partners (PhocusWright, 1998a), with Yahoo and Netscape among the

[8] In February 1998, Galileo International signed a letter of intent to form a strategic alliance with Internet Travel Network (ITN) (now renamed getthere.com) to help develop and integrate ITN's corporate booking system with the Galileo and Apollo computer reservation systems.

largest. Travelocity seeks partners that have a significant Internet reach and a user base that has a propensity to buy travel online. Therefore, what GDSs may aim to achieve eventually is to evolve from a wholesale to a retail sales tool.

How is their relationship with travel agents? Obviously, GDSs' dependency on traditional travel agents will be reduced. However, as long as the agents can still help the airlines to maintain their market shares, the link will still exist. For example, Sabre introduced to travel agents a product called 'Web Reservation' which allows any Sabre agent to issue tickets booked on Travelocity and to be selected online. Travel agents can also contact their Sabre account manager to integrate the Sabre booking engine on their Web sites. As of 1998, Sabre has created 12,500 travel agent Web sites using 'Web Reservation' (Sabre, 1999). Similarly, Amadeus' 'Private Label' allows any Amadeus participant (both providers and subscribers) to host their Web site by Amadeus or to integrate Amadeus booking engine on their current site. WorldSpan also launched an Internet-based system for agencies called Odyssey.

How is PTA going to affect the balance of power in the long term? As mentioned before, the GDSs manage to keep their power simply because of one single important reason: It is a *centrally managed aggregate data source*. The combination of volume of data available and direct links to travel suppliers' inventory for real-time data creates a powerful position, which is unrivalled even with the presence of the Internet as a direct distribution channel.

PTA will have its impacts by providing an *alternative* to the GDSs. This is where the powerful impact of *distributed computing* comes into action. First, PTA is a multi-agent system that is best suited to solve distributed problems, like integrating disparate sources of information. It has the power of removing the need for a data aggregate source completely. In a PTA system, multiple agents belong to suppliers and other intermediaries or auxiliary service providers will be able to interoperate with customers' PTAs. Obviously, the advantage of centralising data at one source does not have any more ground. PTA makes it feasible and worthwhile for various groups of players to maintain their own databases. In addition to this, having distributed databases has a higher fault tolerance than a centralised one. So, the picture, though long term, is quite complete, suppliers and all others will be offered a <u>cheap</u> option to get out of the GDSs, which is hard to resist. With this gradual, but continuous shift of power, it may be logical to speculate on the eventual downfall of the GDSs.

7.2.4 Auxiliary Service and Content Providers

This group is seen as being the obvious organisation to secure and maintain high quality destination information and to provide access to the market place for smaller local and regional companies. Unfortunately, travellers have to take the initiative to visit the Web sites and retrieve all the relevant information themselves. Hence, there are two major concerns:

- to increase customer awareness, and
- to add value to the shopping experience.

The early adoption of agents for this group is rather straightforward. One approach to help increase awareness is the online mimicking of familiar physical-world travel consultancy. For example, tourist offices can use their agents to 'wrap up' destination information, proactively contact potential travellers' PTAs, and upload the information onto travellers' computers. These may include audio and visual information[9] of destinations, accommodations, etc., to give the traveller a preview of destinations and products.

[9] It is expected that state-of-the art technologies would be widely available to allow agents to 'understand' *multi-media* information like sounds, images, video and text in order to facilitate commerce within complex multimedia information systems. Even now, there is already a lot of progress in recognition of images and of speech, and in understanding natural language. As that progress manifests itself in lower-cost, more widely deployed software, it is natural that agents will start using it more routinely.

Software agents can also be exploited to provide instant customer services when they receive requests for information from customers' PTAs. Based on the specified requirements, these customer service agents can process the real-time requests and dynamically package available relevant materials in the most efficient way.

7.2.5 Newcomers - Yellow Pages/Broker Agents

As mentioned before, there is a trend for re-intermediation on the Web. In an environment where there are numerous distributed information sources, the need for intermediaries is inevitable. As travellers' PTAs must obtain travel information such as timetables and seat availability information from the databases of appropriate travel companies, they need some means of discovering what relevant resources exist and where they can be found. One may say that it is the job of a customer's PTA to go out and search information all on its own. However, given the frequency that Internet addresses change, it is wise to avoid hand-coding the addresses of resources into agents. Even then, the central idea of PTA is not for 'searching' because this is a low-level job, and so burdening it with 'searching' is not a good design. It seems logical to relieve travellers from the burden to go out and find out who are offering and/or seeking certain information and services; why should PTAs be burdened with exactly the same task?

Clearly, the Internet equivalent of Yahoo that provides 'yellow pages' type service listings, rather than Web site directories, is needed. Specialised Yellow Pages software agents are expected to emerge as *new* intermediaries to maintain indices of available resources that serve the whole agent community. By inserting yellow page agents into the picture, the burden is lifted from individual PTAs.

One most interesting issue here is that the idea of distributed computing can be applied. It is most likely that there will be a few general-purpose Yellow Page agents who compile service category listings. When a customer's PTA searches for information, it will automatically go to one of these Yellow Page agents first for further referral service. The PTA will be referred to the 'Travel' Yellow Page agent who will further refer it to various specialised Yellow Page agents, e.g., 'exotic places' Yellow Page agent, 'hiking tour' Yellow Page agent, etc.

These Yellow Page agents will be increasingly popular. They provide channels in which travel service providers advertise their services by describing explicitly the service that they provide. They discover new information and keep indices current by deciding which sources are no longer reliable or up-to-date. They may also add value to the information, e.g. by sorting or giving recommendations. Upon the request of a PTA, these Yellow Page agents will dynamically match the query request with an optimal match between supply and demand. In return, these agents may charge a fee for their services, and this is why they are also called 'broker' agents.

7.3 Macro Impacts on Electronic Travel Market

PTA will impact travel distribution to a great extent because of a single powerful concept – Distributed Computing. In other words, it removes the need for a centralised database by integrating disparate information sources. Customers and suppliers will certainly welcome this change and gain benefits from this new technology. Intermediaries, instead of the widespread speculation about their disappearance, will more likely evolve into new forms. Apart from changing the economics of the distribution channels on a micro level, PTA will also affect the evolution of the online market on a macro scale. All the macro impacts below are intertwined with one another, e.g., the consumer-driven market will lead to more intense supplier competition, which in turn will push suppliers and intermediaries to offer add-value services and tailored-made products in order to gain competitive advantage.

7.3.1 The Market

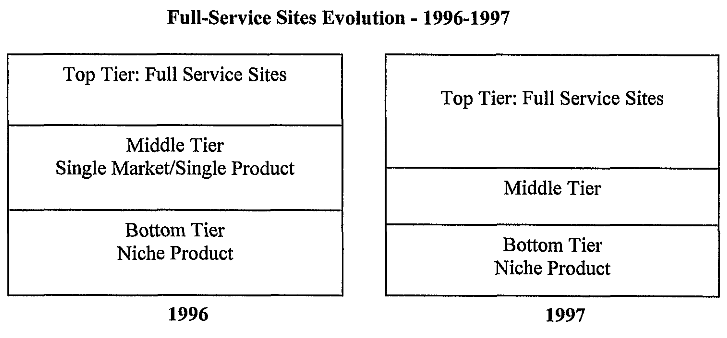
7.3.1.1 Customer-Driven Market

At present, the trend towards a customer-driven market is clearly illustrated by Priceline.com's 'demand collection' model. Customers are invited to name their prices for airline tickets, which airlines with otherwise empty seats can take or leave. However, it is still far from a perfect information market at this moment due to the difficulty of finding and retrieving information. Nevertheless, customers already have more 'say' in choosing their products. PTA will accelerate this trend because of the total market transparency that it will bring along.

In the long term, PTA will continue to reinforce the existing 'power shift' by turning the online travel market into a truly buyer's market. As more and more agents are working on the Internet, the more downward price pressure there will be, and the more level the playing field. PTA will be able to obtain comprehensive information about a supplier or an intermediary, its products and its pricing relative to nearly every one of its competitors. As a result, suppliers are watching the power they gained from information inequality wrested away. Any mechanism that spreads information more widely will, from an economic theory point of view, make real markets correspond more closely to idealised markets. It increases efficiency of the marketplace, and will tend to provide goods at better prices for the consumers.

7.3.1.2 Consolidation?

Recently, the most marked market trend is the beginning of the stage of consolidation (Jupiter communications, 1999b) where top-tier mega-sites continue to gain market share, dominate the online scene and squeeze out many of the middle-tier smaller suppliers. The bottom tier is characterised by smaller sites that serve niche markets.


Full-Service Sites Evolution - 1996-1997
Top Tier: Full Service Sites
Middle Tier
Single Market/Single Product
Bottom Tier
Niche Product
1996
Top Tier: Full Service Sites
Middle Tier
Bottom Tier
Niche Product
1997

Source: Jupiter Communications, 1999

Figure 7.4 The Trend towards Consolidation

The short history of the online travel market clearly showed two distinct phases of development. In the first phase, there were few entrance barriers. It offered a comparative advantage for the small suppliers, allowing them a direct participation in the online market (and closing the cycle from planning, marketing to selling). As a result, there was a proliferation of Web-based information systems built by different players.

By now, the industry is just preparing to enter the second phase. The field of major players is consolidating quickly, as rapid growth in traffic to the large full-service mega-sites combined with high operating costs created by that traffic make the top tier a place for only the giants. Internet travel is currently dominated by six mega-travel Web sites including Travelocity, Expedia, Preview Travel, Internet Travel Network, BizTravel.com and Trip.com. They account for 75% of the Internet revenue in the travel industry (TIA, 1998).

While forecasts (Jupiter Communications, 1999b) predict that consolidation will continue to intensify and gather momentum, PTA may release the pressure by changing the balance of power between the mega-sites and smaller mid-tier suppliers.

Though mega-sites may continue to gain market share in the short term, there may be a return of more small and mid-sized suppliers in the longer term when software agents are used more earnestly by the market players. Instead of the prediction of a consolidation of a cottage industry, PTA enables fair play, and hence reduces the pressure of further concentration.

7.3.2 The Product

7.3.2.1 Valued-Added Service

As mentioned in chapter four, there has been a strong growth of online travel revenues. However, the sales are dominated by air-ticket sales at this moment, which is a straightforward commodity product. PTA will enable more complicated and expensive products traded on the market, such as customised package holidays. It is expected that there will be a shift from 'customer focus' to 'customer-driven marketing', where consumers, via their PTAs, actively specify their precise requirements. Suppliers' agents will be able to offer highly customised packages by dynamically aggregating the products in their inventory to suit these customers' specifications. For example, a customer's PTA can download several software agents from selective suppliers to arrange a package for its owner. While attempting to satisfy the goal of each party, supplier agents may strategically create dynamic business partnerships that exist only as long as necessary. In other words, transactions may emerge in the form of dynamic relationships among previous unknown parties. It is at this stage where companies will be at their most agile and markets will approach perfect efficiency.

7.3.2.2 Tailored-Made Products

As mentioned before, the ultimate advantage of PTAs is their ability to maintain a one-to-one relationship with the customers. Currently, all players including airlines, hotels, travel agents, tour operators, GDSs, destinations, etc. already have their online and direct electronic marketing strategies. Though these are all different, they all have one single common important element, that is, precisely 'personalisation'. To quote Sabre (1999), 'we are not yet in the world of online recommendations for totally customised travel'.

PTA enjoys a one-to-one relationship with the customer. Direct feedback is thus possible with immediate response. So, the dream of 'total' personalisation is within reach. Suppliers' and intermediaries' agents are able to get not just segment-level but individual-level understanding of customers' habits and preferences. At the same time, they are able to tailor the products they deliver to suit individual customers' interests. It allows suppliers to expand the choices they offer all customers, and tailor their individual offerings more precisely to what a given customer wants.

7.3.2.3 Variable Pricing

On one hand, it is predicted that PTA will encourage the online market approaching perfect efficiency because of the market transparency that it will bring along. However, there may be counter actions from suppliers by manipulating net pricing. The Internet provides a cheap means of changing the price dynamically according to market demand. This is demonstrated by the recent proliferation of airline ticket auctions, e.g., Priceline.com introduced its real-time travel auction service on the WWW in 1996 to allow customers to bid for airline seats, hotel rooms, and other travel-related products. To go one step further, suppliers may use software agents to leverage net pricing strategies to keep the price differentials, e.g., agents can dynamically change the composition of products to manipulate the price. As price comparison is only possible under situations where customers compare like with like, suppliers may succeed in keeping their price differentials in this way. Besides, it is also possible to manipulate demand via variable pricing on the same product. It has been proven by the strategy in the Kasbah project (Chavez & Maes, 1996) where software agents can efficiently monitor demand condition - raise price when demand is high or reduce price when demand is low.

7.4 Conclusion

The Internet, the development of tourist sites on the WWW, and the resultant change in the nature and buying behaviour of the tourists are joint forces leading to the dramatic changes in the tourist service system over the last few years.

PTA will reinforce many of the changes that already exist in the online market. It will intensify the degree of customer orientation, value-added services and market efficiency. It will transform the trend of market consolidation by balancing the power between small and large players. However, the most important impact of PTA is that it creates an opportunity for *distributed computing.* This strong force will enable all players to maintain their own separate databases and make the option of direct sale feasible and worthwhile, potentially reducing travel suppliers' dependence on travel agents and GDSs. Hence, it casts doubts on the future of the GDSs. Another big impact of PTA is to prove the need of *new intermediaries.* While many predict there will be a bypass of the GDSs and travel agents, PTA may induce these players to evolve into specialised roles which may prove to be more productive for the whole value chain.

7.5 References

BizTravel.com, http://www.biztravel.com.

Chavez, A. and Maes, P., (1996), Kasbah: An Agent Marketplace for Buying and Selling Goods, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technologies (PAAM-96),* London, U. K., pp. 75-90.

Dombey, A., (1998), Separating the Emotion from the Fact – the Effects of New Intermediaries on Electronic Travel Distribution, in Buhalis, D and Jafari, J, (eds.), *Information and Communication Technologies in Tourism,* The Fifth ENTER 98

International Conference on Information and Communication Technologies in Tourism, Istanbul, 21-23 January, 1998, pp. 129-138.

Expedia, http://www.expedia.com.

Internet Travel Network, http://www.itn.net.

Jupiter Communications, (1999b), *Consumer Market Forecast and Research,* Applied Track, The Sixth ENTER 99 International Conference on Information and Communication Technologies in Tourism, Innsbruck, Austria, 20-23 January, 1999.

Pegasus Systems, Inc., http://www.pegsinc.com.

PhoCusWright (1999), Internet Hotel Sales to Reach Almost US$4 Billion by 2001, *Hotel And Hospitality: The Sleeping Giant Of Internet Travel?*, 28 April, 1999.

PhocusWright, (1998a), *Insights, Analysis & Commentary for the Online Internet Travel Marketplace,* 1(1), September, 1998.

Preview Travel, http://www.previewtravel.com.

Priceline.com, http://www.priceline.com.

Sabre, (1999), *Marketing the Consumer in the Digital Age – New Approaches, New Opportunities,* Applied Track, The Sixth ENTER 99 International Conference on Information and Communication Technologies in Tourism, Innsbruck, Austria, 20-23 January, 1999.

Travelocity, http://www.travelocity.com.

TravelWeb, http://www.travelweb.com.

Trip.com, http://www.trip.com.

Weiser, M., (1991), The Computer for the 21st Century, Ubiquitous Computing Paper, *Scientific American,* September, 1991.

WorldRes, http://www.worldres.com.

Chapter Eight

Conclusions and Future Work

With the Internet and the WWW becoming more and more popular as a medium for electronic commerce, the possibility of arranging an entire trip itinerary from a traveller's personal computer, using information gathered from the WWW, based on the traveller's own personal preferences, seems a dream come true.

Unfortunately, the Web, as interesting as it might be, does not integrate well with the heavily segmented travel industry. At present, the information necessary to plan a comprehensive trip using different transport and accommodation services tends to be distributed across the Internet on the Web sites of individual companies. While the Web enables direct access to various information sources and services, integrating these services effectively to provide a total solution for a complex trip remains a challenging task that requires significant human intervention. In other words, though the WWW is an ideal medium for disseminating information to individual users, it is very difficult to find and integrate information from multiple Web sites. This is not surprising given that the WWW was developed to allow people to interact with remote computers. When computers need to communicate with each other, they do so directly. This raises the question: why are the PCs on users' desks unable to contact the appropriate travel companies directly and arrange a journey?

This thesis describes the idea of using agents to personalise travel services through media such as the Internet by integrating distributed travel services. However, before reaching such an ideal situation, there are some major technical and economic barriers that need to be crossed to make a multi-agent application feasible. This research identifies the importance of a transitional system to accelerate the penetration of agent technology. A collaborative learning strategy was employed to speed up prototyping. Then, the feasibility of the idea was demonstrated with a prototype - Personal Travel Assistant (PTA) - which was implemented in the Java language. The idea is to provide a *stepping stone* to bridge the gap between the present and the ideal situation. It is believed that this first step is crucial to realise the ultimate objective of providing personalised journey information and assistance to travellers. In this last chapter, some future work is suggested for further refinements.

8.1 Extensions and Future Work

8.1.1 Security

Technically, the prototype should be ready for use in a real-time environment. However, as PTA has to download 'outside' agents onto the travellers' computers, there are risks that the host system may be subject to a variety of attacks by malicious agents. The traditional methods of attack include eavesdropping, masquerading, message tampering, message replay and viruses. Hence, security features need to be added to provide *safety* and *privacy.*

It is possible to take advantage of the built-in security features of Java. Java has a security manager[1] to enable browsers to run untrusted applets[2] in a trusted environment. The same security measures are applicable for running Java program codes loaded from the Internet. The key to this approach is the fact that Java is based on an <u>interpreted</u> language that facilitates detailed control over the capabilities of the unknown codes (e.g., agents) running on top of it. By default, downloaded codes are considered untrusted unless they are signed by an identity marked as trusted in the

[1] See 'Requirement Analysis' section of chapter six for details on the security manager.

[2] An applet is a Java program that is run from inside a Web browser. The HTML page loaded into the Web browser contains an `<applet>` tag, which tells the browser where to find the files containing Java program codes.

identity database of the host. This is very important because cryptography and related certification and authentication services will play vital roles in agent-mediated electronic commerce. Currently, PTA makes use of default security features that need to be refined.

The security manager is also responsible for enforcing restrictions to prevent downloaded codes from inspecting or changing files on the client file system. In Java-enabled browsers, untrusted applets cannot read or write files at all. It also prevents applets from using network connections to circumvent file protections or people's expectations of privacy. In addition, applets loaded over the net are prevented from starting other programs on the client. Applets loaded over the net are also not allowed to load libraries, or to define native method calls (If an applet could define native method calls, it would give the applet direct access to the underlying computer).

However, there is a fine line between a secure system and an unusable one. Therefore, it is important to choose carefully the trade-offs between flexibility and security.

8.1.2 Trust

In the previous section, some security measures are proposed to safeguard against 'outside' agents. Do users need to protect themselves against their 'own' agents? This is the issue of trust which has become a major discussion point in the domain of agents. As mentioned in chapter three, delegation implies relinquishing user control, and unless users trust their own agents, they do not feel comfortable with letting them handle more important tasks. This is, however, not a problem of the technology. It is important to get the design right in the first place. For example, a traveller may send his/her PTA out to look for travel deals, and it may even do a good job from start, but he/she will definitely want the final say on whether to buy the package or not. In order to make users trust their PTAs, it is necessary to insist that humans remain in the loop in the initial period. Within this model, a user will always have to approve the purchase or agreement that an agent initiated. This introduces some inefficiency into the overall system, but people are likely to demand it for a long time.

To make sure that agents do not surpass the goals they are instructed to fulfil, future refinements of PTA will introduce this kind of 'warranty' with a two-stage process: (1) involving contingent commitment in the first stage, and (2) final (human-approved) commitment in the second. In other words, humans always have the ability to have final approval of what their agents are agreeing to in their names. Such an approach should be enough to handle everyday electronic commerce transactions.

When users are getting more familiar with their PTAs, and at the same time, when the learning ability of PTAs transforms them into highly competent assistants, users' levels of trust may be high enough to remove this 'human-approval' requirement. Here, a more promising approach is to design a PTA that learns to discriminate those transactions that it can confidently handle from those it cannot. PTA can distinguish cases in which it can trust its learned rules from those in which it should not by relying on past performance to attach confidences to individual rules. It can assign 'competence thresholds' to each of the rules. For example, in making decisions with rules with 'low confidence level', PTA will need to seek its owner's approval before making any commitment - 'Ask' threshold. In situations where the applicable rule has been quite accurate in the past, PTA will try to automate the transaction without consulting the traveller – 'Action' threshold. In this way, PTA will handle situations for which it has high confidence, while referring difficult indecisive cases to the traveller. This allows more flexibility into the system because travellers can always change the amount and form of feedback, and finally reduce it to some minimal level. Of course, people will want to know all the actions taken for them at the beginning, but after they have come to trust the actions, they may not want complete reporting in simple, straightforward situations, especially when the decision is time-critical.

These built-in safeguards to prevent runaway computation are very important to ensure that people feel in control. It will allow a gradual transfer of authority and release of supervisory control as both the traveller and PTA gain confidence in its evolving capabilities.

8.1.3 Ontology

As far as ontology is concerned, PTA has made some preliminary attempts to solve the problem. This is done by a decision-tree classification technique via a learning mechanism. In the PTA system, each agent builds up its ontology by sharing, gathering, refining and extending its own ontology within the agent community. When deciding whether to add a new vocabulary to its ontology, the agent checks the consistency and sees if the addition will conflict with any existing definitions. If the new item proves to be valid, PTA will merge the new vocabulary into its ontology.

This approach works well for a limited domain as evident in the PTA prototype. The same decision-tree classification technique is inherently suitable for real world applications. However, refinements will be needed for issues such as the validation, conflict resolution and merging of large ontology branches.

8.1.4 Negotiation

Negotiation is not common in the real world because it wastes a lot of time and hence accrues transaction costs that may be too high for either consumers or suppliers. PTA can be exploited to automate the process of negotiation (also related closely to auctions in the next section). The process of negotiation can be described in terms of *protocols* and *strategies*. The protocols of a negotiation comprise the rules, i.e., the valid actions, of the game. For a given product, a bidder uses a rational strategy, i.e., a plan of action, to maximise his/her utility.

PTA can be extended to include a simple set of protocols for communications between buying and selling agents. Here, instead of creating a full-blown agent communication language, a more pragmatic approach may prove to be more useful. It is possible to establish a minimum set of parameters (e.g., clearing times, method for resolving bidding ties, the number of sellers permitted, etc.) to be used as a common protocol.

To be able to negotiate effectively, PTA will need some form of strategy to guide its actions. PTA can act as selling agents for the suppliers and buying agents for the travellers. For example, Kasbah[3] uses a crude 'price-raise and decay' negotiation strategy for its buying agents. Each buying agent has three strategies – anxious, cool-headed and frugal which corresponds to a linear, quadratic, or exponential function respectively for increasing its bid for a product over time. Similarly, selling agent can use the same techniques to reduce price over a period of time.

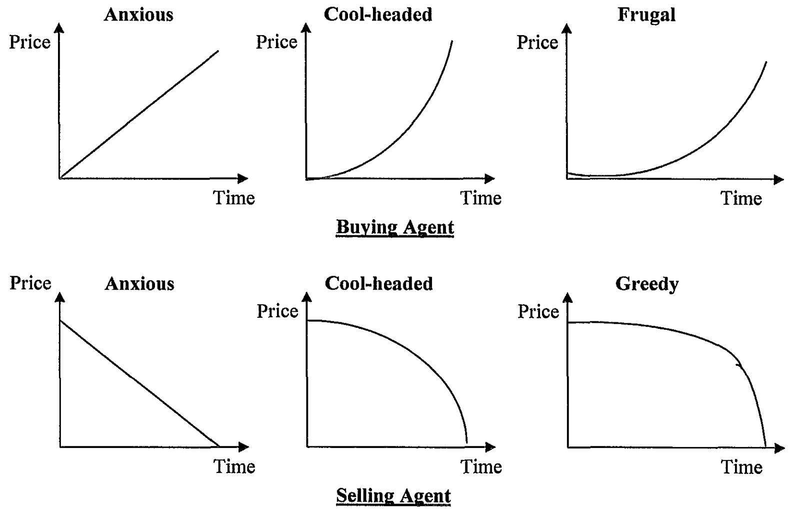


Figure 8.1 Negotiation Strategies in Kasbah

Similar to Kasbah, a lot of research projects use price as a basic parameter for negotiation. Such negotiation strategies are straightforward and easy to understand, however, it is too crude to be used in real-world situations. First, price is only one of

[3] See 'Examples of Agent Systems' in chapter three for details.

the many features of a product. In addition, the variety of strategies is too limited and restrictive. For example, if a buying agent's strategies are limited to a few options, the selling agent can detect the pattern easily because the variations of its bidding behaviour are too predictable.

Since PTA operates in an open, multi-agent system, the use of even simple negotiation strategies will bring great benefits. Holding simultaneous conversations with several suppliers will arm PTA with last-minute market information. Pitching suppliers against suppliers is always a very effective strategy. The adjustments on 'offer price' will depend on the number of suppliers found who are willing to negotiate, and the rate of price decrease. On the supplier side, the agent will set the 'asking price' with respect to instantaneous stock levels, revenue targets and the number of deals and customer enquiries. These generic negotiation algorithms are not difficult to add to PTA since no product-specific attributes are needed. However, instead of competing solely on price, suppliers are likely to react with more diversified strategies, e.g., by offering more flexible fare rules, class upgrades, etc. Under these situations, PTA will be able to learn rules from other agents to evaluate the product-specific attributes and decide on the optimal product.

This is an effective way to match demand and supply. To the travellers, the opportunity cost of haggling is non-existent because no time will be wasted. A result of this is that limited resources are allocated fairly, i.e., to those who value them most.

8.1.5 Auctions

There has been a growing popularity of various types of auctions on the Web in the last few years. For example, Travelfacts[4] is a travel auction site where travellers can bid for airline tickets, hotel rooms, cruises and other travel-related products. However, it requires a great deal of human intervention at this moment because

[4] Travelfacts: http://www.travelfacts.com.

travellers have to monitor the auction and do the bidding themselves. The actual bidding may be too complicated or frustrating for the average travellers. Auctions also occur over an extended period of time which does not cater to impatient or time-constrained travellers. Hence, means for automated negotiation are a natural complement to online travel auctions in the future.

The learning mechanism of PTA can be extended to handle auctions and bidding without prior domain-specific knowledge programmed into the software. Also, the user may not be aware that PTA is involved in auctions until final approval for the actual purchase is needed. The advantage is that the user does not need to understand the differences and rules of the myriad of auctions, and also PTA can handle new forms of auctions whenever they are launched on the Internet. For example, when PTA is instructed to shop for a flight ticket, it will first contact the Yellow Page agent to discover those agents in the community that have the required product in their inventory. When PTA visits a conventional supplier, it will learn the definitions and features of the product. Similarly, if the visited site is an auction site, the rules of auctions and tactics of bidding are learned by PTA. For example, a price may be associated with a valid duration. It is expected that some basic built-in knowledge of auctions can be added to enable PTA to learn and handle all varieties of auction modes like English Auctions, Dutch Auctions, Double Auctions, etc.

8.1.6 Multi-Product

Being a proof-of-concept demonstration prototype, the current function of PTA is limited to flight arrangements at this moment. However, multiple products such as 'fly-drive' packages can also be handled as a single integrated product if they are readily bundled up by suppliers or intermediaries.

The next step is to extend PTA to be capable of integrating multiple travel products into a total package, which does not require major alteration on the prototype. The existing architecture is scalable and can accommodate further addition of supplier agents, like hotel agents, car agents, etc.

Given the flexibility of the learning mechanism, arranging a multi-component product is a similar job as what has been done in the single-product demonstration. Just as PTA acquires learned rules from the flight agent, the same learning process will be applicable to all other types of agent. In requesting a multiple product package, the user will enter specifications, i.e., attribute-value pairs, for each required product one by one. Related attribute names of separate products such as 'Sales-trip arrival date' for flight ticket and 'Sales-trip start date' for car hire are used to link up the products. PTA will use learned rules from suppliers or intermediaries to reason, plan and co-ordinate the sequence and composition of the purchase. Some built-in generic co-ordinating rules in PTA are desirable. This incremental work will mean a big improvement on the usefulness of PTA.

8.1.7 Multi-Stage Decision-Making

The current version of PTA concentrates on a single decision-making phase - product comparison. This stage typically occurs after the traveller has already identified the required product, e.g., an air ticket in the prototype, and the decision is basically on *which* alternative to choose. However, it must be noted that the functionality of PTA can be extended to cover other decision-making phases, such as product identification. At this stage, a traveller has to decide *what* to buy, e.g., destination choice.

At present, there are two ways to identify a product:

- Information Pull where a traveller will take the initiative to browse various travel Web sites, and

- Information Push where the suppliers will choose what information to send to the travellers.

Both methods are far from ideal. Information pull is hardly customised and at the same time laborious and time consuming. Information Push is too restrictive and the quality of the information will depend on how well the suppliers understand the travellers.

PTA, being a multi-agent system, can merge the advantages of information push and pull. PTA can filter information, which is obtained from suppliers' agents, before submitting it to the traveller. With multi-agent co-operation, the information will likely be organised in a highly customised manner. Moreover, information can be retrieved in various forms such as formatted text and images. Web pages can be bundled up and directed to the browser. Any Java objects can be embedded as part of the information just as applets are embedded in Web pages. In addition to text-based information, multi-media materials such as interactive dialogue can be gathered by PTA and started by the user at leisure. Programs triggered by the dialogue can then connect back to the information source to perform additional tasks.

8.2 Concluding Remarks

Artificial intelligence as a field has arguably suffered a great deal from over optimistic claims about its potential. Most recently, perhaps, the expert systems experience vividly illustrates the perils of overselling a promising technology. The problem usually does not lie in the technology itself, but the overblown expectations by software developers about the capability of the technology. It is why one of the design principles of PTA is to <u>build 'true' agents</u>. The current situation on the Internet shows that there is a risk of software agents being overused for marketing. Many people equate agents with anthropomorphic characters or an interface with an animated face with no real functionality, which is truly an alarming trend of development.

What many people forget is that agents are not the end product to accomplish the miraculous. They are *tools* to achieve the *goal*. There are a number of good reasons for supposing that agent technology will enhance the ability of software engineers to construct complex, distributed applications: other considerations aside, agents are a powerful and natural metaphor for conceptualising, designing and implementing many systems. In short, agents may make it easier to solve certain classes of problems, (and there are good arguments for supposing that this is the case), but they do not make the impossible possible.

The other aspect of overselling is to equate agents with intelligent problem solving. Those unfamiliar with the achievements (and failures) of Artificial Intelligence often believe that agents are capable of human-like reasoning and acting. Obviously, this is not the case: such a level of competence is well beyond the state of the art in AI. Thus agents may sometimes exhibit smart problem solving behaviour, but it is still very much limited by the current state of the art in machine intelligence.

Regardless of the beauty of PTA, it is good practice to refrain from inflating the travellers' expectations beyond agents' capabilities as giving false hopes will lead to a lack of trust in agent systems. PTA starts humble, but as a true agent, it has unlimited potential to learn. Its mission is to increase awareness in the electronic travel market, pave the migration path to carry the evolution to the next development stage. Of course, there are some social and legal issues that remain to be sorted out which are well beyond the scope of this thesis. However, it is expected that PTA would serve its job well and become more than yet another temporary technological innovation.

'A little use of intelligence goes a long way.'

Appendix A

Methodology

The following excerpt is taken from ActivMedia's report – The Real Numbers Behind Net Profits, 1997.

Background

The survey is based on ActivMedia's Fourth Semi-Annual Study of Net Marketers, a tracking study conducted among managers responsible for marketing decisions on the Web. It builds on earlier studies with the longitudinal profile of eight market segments and doubled sample size to enhance depth exploration of Web business sectors.

To be included, respondents must have indicated a primary Web site product or service category among these eight generic business groups:

- COMPUTER (hardware / peripherals / software / support)
- CONSUMER (audio / video / CD / books / entertainment / gifts / clothes / home / hobby / sports / adult)
- BUSINESS & PROFESSIONAL (general & specific B&P services / products)
- FINANCE (banking / investment / insurance)
- INDUSTRY (manufacturing / transport / telecom / energy / construction / agriculture / mining)

- INFORMATION (publications and magazines / information providers)
- REAL ESTATE (sales / rental / operations / maintenance / investment)
- TRAVEL (destinations / agents / services)
- OTHER (reclassified to nearest standard industry group)

Excluded from this analysis (reported separately) are Web sites devoted solely to Web infrastructure, whose contributions are part of the cost structure for traditional business online:

- INTERNET SERVICE (access / site host / ISP / network support / directory / search)
- INTERNET PROMO (Web design / mall / ad design / Web marketing consulting)

Sampling

In late December 1996, ActivMedia compiled a comprehensive global directory of all URLs listed in twelve worldwide structured directory systems (including Yahoo [US, Canada, UK, Germany, France and 6 US Cities], Open Market [now defunct], Big Yellow [excluded for excessive overlap], Lycos [top 5% Web sites] and Hoover). An additional half-dozen local Pacific Rim/Asian directories were also evaluated but eventually excluded due to excessively narrow industry focus. Yahoo Japan is notably absent from the study due to anticipated language barriers (kanji character limitations in e-mail) in later study steps. The criterion for inclusion was broad coverage of URL's dedicated to business over the Web. An internal 'Directory Unduplication Study' was conducted to guide final sampling.

Web sites were chosen randomly (Nth name) to ensure samples were balanced to Global proportions. Agent technology was used to visit Web sites and collect e-mail addresses (within 3 levels of the starting point URL).

Using a split-sample method, two survey versions were deployed to a net effective mailing of 28,000 and 6,580 Web sites respectively. The first sample, e-mailed in late January 1997, offered a US$100 incentive awarded by random selection to 10 respondents.

Net effective return rates after validation study and second mailing were 18.5% (2,440 surveys/13,159 effectively targeted surveys) after adjusting for respondents that (a) never saw the survey, (b) never read past the first line (equivalent to telephone hang-up before qualification), (c) were not in the desired target group or (d) never received the survey due to e-mail system non-performance.

A second survey wave in late February targeted 6,580 respondents. Similar non-response adjustments result in a net effective return of 27.2%, derived from raw return rates of 12.1%. Split samples were evaluated for matching demographics and consistency across multivariate indicators, then combined (other than divergent questions) to comprise the final sample set of responses.

In total nearly 4,000 surveys were received, of which 3,541 were sufficiently complete for tabulation. Of these, 79% (2,811) were self-classified into one of the eight Web Commerce categories and included in this report; the balance were classified as Internet Providers and Internet Promoters.

All surveys were conducted in English, the most common denominator language of today's Web (future studies will be multi-lingual). Japanese Yahoo Web sites were not sampled, and study results refer to non-Japan global statistics. While low incidence precludes traditional statistical margin of error analysis, consistent findings across split samples and compared over time with the Third Wave (July '97, May '97 data collection) suggest that results are repeatable and generally reliable indicators of Web progress. Study reviewers should beware of inherent limitations in representation of global projections to secondary audiences. Nevertheless, study findings remain the best estimate available today of Web site practice and performance as viewed from the perspective of the executives responsible for Web site decisions.

Appendix B

Data Tables

This is an updated version of the survey on consumer Internet access in chapter four. Results show that the projections given in chapter three closely coincide with the actual figures. As of August 1999, the figures for consumer access is as follows:

Region	**Access in millions**
World Total	**195**
Africa	1.72
Asia/Pacific	33.61
Europe	46.39
Middle East	0.88
USA & Canada	107.3
South America	5.29

Table B1 Consumer Internet Access (1999)

Below are the data tables for Worldwide and the United States.

Worldwide

Date	Access in millions*	% total pop	Source
August 1999	195.19	4.64	Nua Ltd
July 1999	185.2	4.41	Nua Ltd
June 1999	179	4.27	Nua Ltd
May 1999	171.25	4.09	Nua Ltd
April 1999	163.25	3.9	Nua Ltd
March 1999	159	3.89	Nua Ltd
February 1999	153.5	3.75	Nua Ltd
December 1998	150	3.67	Nua Ltd
Sept 1998	147	3.6	Nua Ltd
July 1998	129.5	3.17	Nua Ltd
December 1997	101	2.47	Nua Ltd
September 1997	74	1.81	Nua Ltd
December 1996	55	1.34	Nua Ltd
1995	26	.63	Nua Ltd

* Figures include both adults and children

Table B2 Consumer Internet Access – Worldwide (1995-1999)

United States

Country	Date	Access in millions	% total pop	Source
U.S.	July 1999	106.3	39.37	*** NielsenNetRatings
U.S.	May 1999	101	37.4	*** NielsenNetRatings
U.S.	April 1999	95.8	35.4	*** NielsenNetRatings
U.S.	March 1999	83	30.7	* IntelliQuest
U.S.	January 1999	79.4	29.3	* IntelliQuest
U.S.	October 1998	73	27.8	* IntelliQuest
U.S.	February 1998	62	23.0	* IntelliQuest
U.S.	November 1997	56	21.0	* IntelliQuest
U.S.	June 1997	51	19.17	* IntelliQuest
U.S.	April 1997	40 - 45	16.16	* FIND/SVP
U.S.	November 1996	31	11.15	* CommerceNet/Nielsen
U.S.	1995	18	6.7	* CommerceNet/Nielsen

* Figures quoted are for Adult Population only (Age 16 and over). They do not include number of children online.

** Figures are for Internet users age 12 and older

*** The NielsenNetRatings Internet universe is defined as all members (2 years of age or older) of U.S. households which currently have access to the Internet.

Table B3 Consumer Internet Access – United States (1995-1999)

Appendix C

Software and Hardware Requirements

- Java Development Kit 1.2.2 (Java 2)
- Windows 95
- Pentium 133 MHz
- Memory 32 MB

File Contents

1. Agent Host

StartHost.java
Host.java
HostFrameManager.java
AgentLoader.java
URLloader.java
URLAlias.java
SimpleClassLoader.java

2. Agents

Agent.java
MyAgent.java
WWW_universalist_com_agent.java
Universalist.txt
WWW_travelist_com_agent.java
Travelist.txt
WWW_airnet_com_agent.java
WWW_virginatlantic_com_agent.java
Virginatlantic.txt
Database.java
Virginatlantic.data
Database.java

3. Agent Communication Protocol

AgentMessage.java
AgentAddress.java

Program Listings

Acts.java

```
// Messages types for agent communication protocol.

interface acts{
      int get=0;
      int say=1;
      int where=2;
      int refuse=3;
      int inform=4;
      int go=5;
      int bye=6;
      int buy=7;
      String[] name={"Get", "Say",
"Where","Refuse","Inform","Goto","Bye", "Buy"};
};
```

Agent.java

```
// Common Agent definitions

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

abstract class Agent extends Thread {
      String name;
      AgentAddress address;
/*
Name is for display purposes and need not be unique.
Address is the unique URL except the file name.
E.G. www.universalist.com/travel
The complete URL is
http://www.universalist.com/travel/www_universalist_com_travel.class
The class name has to agree with the filename.
*/

      Channel in;
      Channel out;
      JTextArea textArea;
      JScrollPane areaScrollPane;

    void initAgent(Channel o) {
            out = o;
            in = new Channel();
      }

    abstract public void run(); {
      }

      void bye(){
            try{
                  sleep(3000); // Appear on screen
            }
            catch(Exception e){}
            out.put(new AgentMessage(address,null,acts.bye,null));
      }
}
```

AgentAddress.java

```
// Global naming convention

class AgentAddress {
      String      path;       // all puntuation in URL
      String      fileName;   //no class
      AgentAddress(String p, String f){
            path = p;
            fileName = f;
      }
      String both(){
            return (path+fileName);
      }
}
```

AgentLoader.java

```
// Load agent class file dynamically

import java.net.*;

class AgentLoader {
      Agent loadAgent( AgentAddress address, Channel common ) {

            Agent ag = null;
            String cName=address.fileName;
            try {

                  URLloader scl = new URLloader();
                  Class cl = scl.loadClass(address);
                  ag = (Agent) cl.newInstance();
            ag.initAgent(common);
          }
        catch (Exception e) {
            System.out.println("Load Agent Error: " + e);
            System.exit(1);
        }
            return ag;
      }
}
```

AgentMessage.java

```
// Definition of agent message

class AgentMessage {
      AgentAddress src;
      AgentAddress dst;
      int act;
      Object obj;

      AgentMessage(){}

      AgentMessage(AgentAddress s, AgentAddress d, int a, Object o){
            src = s;
            dst = d;
            act = a;
            obj = o;
      }
}
```

AirportRule.java

```
// Decision rule concerning airport
```

```
/*
If the price is significantly cheaper,
I will consider auxiliary airports.
*/

import java.util.*;

/*
If the price is close to lowest price, I will consider auxiliary
airports.

Auxiliary airports carry a "AUX" tag in database.
*/

class airportRule extends Rule {
      double threshold=1.1;
      double ratio;
      String ori = "Origination";
      String dst = "Destination";
      String price = "Fare";
      String aux = "AUX";
      boolean allAUX = true;

      airportRule(){
            id = "Airport Rule 101 VA";
      }

      public void pass() {
            double lowestPrice = findMin(price);

            // User may choose an auxiliary airport
            for(int i=0; i<data.size(); i++){
                  Map row = (Map) data.get(i);
                  if( ! containString(row.get(dst),aux) )
                        allAUX = false;
            }

            if( ! allAUX ) {
                  for(int i=0; i<data.size(); i++){
                        Map row = (Map) data.get(i);
                        boolean select = false;
                        if( query.containsKey(ori) ) {
                              if( ! containString(row.get(dst),aux) )
                                    select = true;
                              else {
                                    ratio = ((Double) row.get(price)
).doubleValue() / lowestPrice;
                                    if ( ratio <= threshold )
                                          select = true;
                              }
                        }
                        if( query.containsKey(dst) ) {
                              if( ! containString(row.get(dst), aux)
)
                                          select = true;
                              else {
                                    ratio = ((Double) row.get(price)
).doubleValue() / lowestPrice;
```

```
                                        if ( ratio <= threshold )
                                             select = true;
                                   }
                         }
                         boolean x = ((Boolean)
row.get("_selected")).booleanValue();
                         x = x && select;
                         row.put("_selected", new Boolean(x) );
                         row.put("_selected"+id, new Boolean(select)
);
                    }
               }
     }

     public void calibrate(Object selected){
          Map data = (Map) selected;
          boolean select
=((Boolean)(data.get("_selected"))).booleanValue() ;
          boolean isAUX = containString( data.get(ori),aux)
                         ||        containString( data.get(dst),aux)
;
          if( isAUX && (! select) && (! allAUX) )  // raise
threshold to ratio
                  threshold = ratio;
     }
}
```

Channel.java

```
// Message channels using standard thread save constructs

public class Channel {
    private Object contents;
    private boolean available = false;

    public synchronized Object get() {
        while (available == false) {
            try {
                wait();
            } catch (InterruptedException e) { }
        }
        available = false;
        notifyAll();
        return contents;
    }

    public synchronized void put(Object value) {
        while (available == true) {
            try {
                wait();
            } catch (InterruptedException e) { }
        }
        contents = value;
        available = true;
        notifyAll();
    }
}
```

Database.java

```
// Database building and query in pure Java

import java.util.*;
import java.io.*;

class database {
      String[] fieldNames;
      List rows=new ArrayList();

      database(String fileName)
            throws IOException {
            List names=new ArrayList();
            FileReader fr = new FileReader(fileName);
            StreamTokenizer tok = new StreamTokenizer(fr);
            while( tok.nextToken() != tok.TT_EOF ) {
                  if( tok.ttype == tok.TT_WORD || tok.ttype == '"' )
{
                        if( tok.sval.equals("DATA") )
                              break;
                        else
                              names.add(tok.sval);
                  }
            }

            fieldNames = new String[names.size()];
            for(int i=0; i<names.size(); i++) {
                  fieldNames[i] = (String) names.get(i);
            }
            while( ! makeRow(tok) ) ;
            fr.close();
      }

      String[] fields(){
            return fieldNames;
      }

      boolean makeRow(StreamTokenizer tok)
            throws IOException {
            boolean eot=false;
            Map mp = new HashMap();
            for(int i=0; i<fieldNames.length; i++) {
                  if( tok.TT_EOF == tok.nextToken() ) {
                        eot = true;
                        break;
                  }
                  if( tok.ttype == tok.TT_NUMBER) {
                        if( fieldNames[i].equals("Legs") ||
                              fieldNames[i].equals("Stops") )
                              mp.put(fieldNames[i] , new
Integer((int)tok.nval));
                        else
                              mp.put(fieldNames[i] , new
Double(tok.nval));
                  }
```

```
				else
					mp.put(fieldNames[i], tok.sval);
			}
			if(! eot) rows.add(mp);
			return eot;
		}

		List search(Map mp){
			List lst = new ArrayList();

			if( mp.containsKey("Destination") &&
				mp.containsKey("Origination") )  {
				for(int r=0; r<rows.size(); r++){
					if( matchrow( mp, (Map) rows.get(r) ) ) {
						lst.add( rows.get(r) );
						//System.out.println("match row "+ r);
					}
				}
			}
			return lst;
		}

		boolean matchrow(Map mpSpec, Map mp) {
			int match=0;
			Object[] keys = mpSpec.keySet().toArray();

			modifyDate(mpSpec, mp);

			for(int m=0; m<mpSpec.size(); m++){
				//System.out.println("matching " +  keys[m] );
				if( mp.containsKey( keys[m] ) )
					if( matchKey(keys[m], mpSpec, mp ) ) {
						match++;
						//System.out.println("match " +
	mpSpec.size() + " " + match);
					}
			}
			boolean result=false;
			if(match==mpSpec.size())
				result=true;
			return result;
		}

		boolean matchKey(Object key, Map mpSpec, Map mp) {
			boolean result = false;
			Object require = mpSpec.get(key);
			Object available = mp.get(key);
			String reqS = require.toString().trim().toLowerCase();
			String avaS = available.toString().trim().toLowerCase();
			//exact match
			if( reqS.equalsIgnoreCase(avaS) )
				result=true;
			// ignore direct price request
			if( key.equals("Fare") ) {
				//System.out.println("match fare");
				result=true;
			}
			// match airport
```

```
            if( key.equals("Origination") ||
key.equals("Destination") )
                  if( avaS.indexOf(reqS) >= 0 )
                        result=true;
            // date
            parseDate parser = new parseDate();
            if( key.equals("Departure Date")){
                  Date dd = parser.go(reqS,false);
                  if (dd == null) dd = new Date();
                  mpSpec.put("Departure Date", parser.fmt(dd,false)
);
                  if( mpSpec.get("Departure Date").equals(
                        mp.get("Departure Date") ) )
                        result=true;
            }
            if( key.equals("Return Departure Date")){
                  Date dd = parser.go(reqS,false);
                  if (dd == null) dd = new Date();
                  mpSpec.put("Return Departure Date",
parser.fmt(dd,false) );
                  if( mpSpec.get("Return Departure Date").equals(
                        mp.get("Return Departure Date") ) )
                        result=true;
            }
            if( key.equals("Arrival Date")){
                  Date dd = parser.go(reqS,false);
                  if (dd == null) dd = new Date();
                  mpSpec.put("Arrival Date", parser.fmt(dd,false) );
                  if( mpSpec.get("Arrival Date").equals(
                        mp.get("Arrival Date") ) )
                        result=true;
            }
            if( key.equals("Return Arrival Date")){
                  Date dd = parser.go(reqS,false);
                  if (dd == null) dd = new Date();
                  mpSpec.put("Return Arrival Date",
parser.fmt(dd,false) );
                  if( mpSpec.get("Return Arrival Date").equals(
                        mp.get("Return Arrival Date") ) )
                        result=true;
            }
            if( key.toString().indexOf("Time") >=0 )
                  result = true;
            if( key.equals("Airlines") ||
                  key.equals("Equipments") ||
                  key.equals("Legs") ||
                  key.equals("Stops") )
                  result = true;

            return result;
      }

      void modifyDate(Map mpSpec, Map mp){

            Calendar rightNow = Calendar.getInstance();
            Calendar tomorrow = Calendar.getInstance();
            tomorrow.add(Calendar.DATE,1);

            boolean nextDay =
            mp.get("Arrival Date").equals("1");
        boolean returnNextDay =

            mp.get("Return Arrival Date").equals("1");

        parseDate parser = new parseDate();

        Calendar departDate= Calendar.getInstance();
        if( ! mpSpec.containsKey("Departure Date") ) {
            departDate.setTime(tomorrow.getTime());
        }
        else {
            Date dd = parser.go( mpSpec.get("Departure
Date").toString() ,false );
            if(dd == null)
                departDate.setTime(tomorrow.getTime());
            else
                departDate.setTime( dd );
        }
        mp.put("Departure Date", parser.fmt(departDate.getTime(),
false) );

        Calendar arriveDate = Calendar.getInstance();
        arriveDate.setTime(departDate.getTime());
        if( nextDay )
            arriveDate.add(Calendar.DATE,1);
        mp.put("Arrival Date", parser.fmt(arriveDate.getTime(),
false) );

        // Arrival Date Override
        if( mpSpec.containsKey("Arrival Date") ) {
            Date ad = parser.go( mpSpec.get("Arrival
Date").toString() , false);
            if( ad != null ) {
                arriveDate.setTime(ad);
                departDate.setTime(ad);
                if( nextDay )
                    departDate.add(Calendar.DATE,-1);
                mp.put("Departure Date",
parser.fmt(departDate.getTime(), false) );
                mp.put("Arrival Date",
parser.fmt(arriveDate.getTime(), false) );
            }
        }

        if( mp.get("Trip").equals("Return") ) {
            Calendar returnDepartDate = Calendar.getInstance();
            returnDepartDate.setTime(departDate.getTime());
            returnDepartDate.add(Calendar.DATE,7);

            if( mpSpec.containsKey("Return Departure Date") ) {

                Date rdd = parser.go(
                    mpSpec.get("Return Departure
Date").toString(),false);
                if( rdd != null )
                    returnDepartDate.setTime(rdd);
            }
```

```
                    mp.put("Return Departure Date",
                          parser.fmt(returnDepartDate.getTime(), false)
);

                    Calendar returnArriveDate = Calendar.getInstance();

      returnArriveDate.setTime(returnDepartDate.getTime());
                    if( mp.get("Return Arrival Date").equals("1") )
                          returnArriveDate.add(Calendar.DATE,1);
                    mp.put("Return Arrival Date",
                          parser.fmt(returnArriveDate.getTime(), false)
);

                    //return arrival override
                    if( mpSpec.containsKey("Return Arrival Date") ) {
                          Date rad = parser.go(
                                mpSpec.get("Return Arrival
Date").toString() ,false);
                          if( rad != null ) {
                                returnArriveDate.setTime(rad);
                                returnDepartDate.setTime(rad);
                                if( returnNextDay )

      returnDepartDate.add(Calendar.DATE,-1);
                                mp.put("Return Departure Date",

      parser.fmt(returnDepartDate.getTime(), false) );
                                mp.put("Return Arrival Date",

      parser.fmt(returnArriveDate.getTime(), false) );
                          }
                    }
             }
      }
}
```

Directory.java

```
// Listing agent's ontology database and addresses

package lexical;
import java.util.*;
import java.io.*;

class entry {
      List adr;
      boolean     isDirectory;
      List keyTree;

      entry(List a, boolean is, List k, Map mp){
             adr = a;
             isDirectory = is;
             keyTree = k;
             Iterator it = keyTree.iterator();
             enter(keyTree, mp);
      }
      void enter(List lst, Map mp){
            Iterator it = lst.iterator();
            while( it.hasNext() ){
                  node nd = (node) it.next() ;
                  put(mp, nd.name);
                  enter(nd.branches, mp);
            }
      }

      void put(Map mp, String s){
            if( mp.containsKey(s) ) {
                  Set st = (Set) mp.get(s);
                  st.add(this);
            }
            else {
                  Set st = new HashSet();
                  st.add(this);
                  mp.put(s,st);
            }
      }
}

public class directory{
      Map mp;

      public directory(String fileName)
            throws IOException {
            mp = new HashMap();
            parseTree p = new parseTree(fileName);
            Iterator ent = p.treeList.iterator();
            while( ent.hasNext() ) {
                  node nd = (node) ent.next();
                  boolean isDir;
                  if(nd.name == "directory")
                        isDir = true;
                  else
                        isDir = false;
                  List adrProd = nd.branches;
                  List adrLst = ((node)adrProd.get(0)).branches;
                  List pdrLst = ((node)adrProd.get(1)).branches;
                  new entry(adrLst, isDir, pdrLst, mp);
            }
      }

      public Set search(String allcase) throws IOException {
            String s = allcase.toLowerCase();
            StreamTokenizer parser = new StreamTokenizer(new
StringReader(s));
            parser.lowerCaseMode(true);
            Set addrSet = new HashSet();
            while( parser.nextToken() != parser.TT_EOF ) {
                  if(parser.ttype == parser.TT_WORD )   { //skip
numbers
                        Set entrySet = (Set) mp.get(parser.sval);
                        if (entrySet != null)
                              addrSet.addAll( match(entrySet,s) );
                  }
            }
            return addrSet;
```

```
	}

	Set match(Set st, String s){

		Set matchSet = new HashSet();
		Iterator itEntry = st.iterator();
		while( itEntry.hasNext() ) {
			entry ent = (entry) itEntry.next();
			if( matchEntry(ent,s) )
				matchSet.add( ent.adr );
		}
		return matchSet;
	}

	boolean matchEntry(entry ent, String s){
		 return matchString(ent.keyTree, s);
	}

	boolean matchString(List lst, String s){
		Iterator it = lst.iterator();
		boolean result = false;

		while( it.hasNext() ) {
			node nd = (node) it.next();
			if( s.indexOf(nd.name) >= 0 ) {
				if( nd.branches.isEmpty() )
					result= true;
				else
					result= matchString(nd.branches,s);
			}
		}
		return result;
	}
}
```

DirectRule.java

```
// Decision rule for direct flight
// I prefer direct flights

import java.util.*;

//I prefer direct flights

class directRule extends Rule {
	double addScore=0.3;

	directRule(){
		id = "Direct Flight Rule 103 VA";
	}

	public void score(double weight) {
		String ori = "Origination";
		String dst = "Destination";
		String price = "Fare";

		for(int i=0; i<data.size(); i++){
```

```
                  double doubResults=0.0;
                  Map row = (Map) data.get(i);
                  boolean roundTrip =
containString(row.get("Trip"),"Return");
                  int legs = ((Integer)row.get("Legs")).intValue();
                  if( roundTrip ) {
                        if( legs == 2 )
                              doubResults = addScore;
                  }
                  else {
                        if( legs == 1 )
                                    doubResults = addScore;
                  }

                  double x = ((Double)
row.get("_score")).doubleValue();
                  x = x + doubResults * weight;
                  row.put("_score", new Double(x) );
                  row.put("_score"+id, new Double(doubResults) );
                  //System.out.println("Legs " + legs + " score " +
doubResults);
            }
      }

      public void calibrate(Object selected){
            Map mp = (Map)selected;
            boolean roundTrip =
containString(mp.get("Trip"),"Return");
            int legs = ((Integer)mp.get("Legs")).intValue();
            if(roundTrip) {
                  if( legs > 2 )
                        addScore = addScore * 0.8;
            }
            else {
                  if( legs > 1 )
                        addScore = addScore * 0.8;
            }
      }
}
```

FareRule.java

```
// Decision rule for fare
// I like low price.

import java.util.*;

// I like low price
class fareRule extends Rule {

      fareRule(){
            id = "Fare Rule 102 VA";
      }

      public void score(double weight) {
            for(int i=0; i<data.size(); i++){
                  Map row = (Map) data.get(i);
                  double price =
((Double)row.get("Fare")).doubleValue();
                  double low = findMin("Fare");

                  double doubResult = low/price * weight;

                  double x = ((Double)
row.get("_score")).doubleValue();
                  x = x + doubResult;
                  row.put("_score", new Double(x) );
                  row.put("_score"+id, new Double(doubResult));
            }
      }
}
```

FeatureTableModel.java

```
// A bridge between PTA's product attribute table and Swing's table

import javax.swing.table.*;

class featureTableModel extends AbstractTableModel {
      productSpec prod;
      boolean canEdit;

      featureTableModel(productSpec pd ){
            super();
            prod = pd;
            canEdit=true;
      }

    public String getColumnName(int col) {
        return prod.getColumnName(col);
    }
    public int getRowCount() { return prod.rowCount() ; }
    public int getColumnCount() { return prod.columnCount() ; }

    public Object getValueAt(int row, int col) {
        return prod.get(row,col);
    }

    public boolean isCellEditable(int row, int col)
        { return canEdit; }

    public void setEditable(boolean b){
      canEdit=b;
    }

    public void setValueAt(Object value, int row, int col) {
        prod.set(row,col,value);
            fireTableCellUpdated(row, col);
      }

}
```

Host.java

```
// Agent Host

import java.util.HashMap;

class Host extends Thread {
      public void run() {
            String newline = System.getProperty("line.separator");
            HostFrameManager frman = new HostFrameManager();

            Channel common = new Channel();
            HashMap AgentMap = new HashMap(10);
            AgentLoader ldr = new AgentLoader();
            Agent ag;

            AgentAddress agentAddress = new
AgentAddress("localHost/","myAgent");
            ag = ldr.loadAgent(agentAddress,common);
        AgentMap.put(agentAddress.both() , ag);
            frman.initAgent(ag);
            ag.start();

            while(true) {
                  AgentMessage msg = (AgentMessage) common.get();
                  if( msg.dst != null ) {
                        System.out.println("from " + msg.src.fileName
+ " to " + msg.dst.fileName);
                  if( ! AgentMap.containsKey(msg.dst.both()) ) {
                              ag = ldr.loadAgent(msg.dst, common);
                              AgentMap.put(msg.dst.both() , ag);
                              frman.initAgent(ag);
                              ag.start();
                        }
                        ag = (Agent)AgentMap.get(msg.dst.both() );

                        Channel ch =  ag.in;
                        ch.put(msg);
                  }

                  ag = (Agent) AgentMap.get(msg.src.both() );
                  ag.textArea.insert( acts.name[msg.act] + newline, 0
);
                  switch( msg.act ){
                        case acts.say :
                              ag.textArea.insert( (String)(msg.obj) +
newline, 0);
                              break;
                        case acts.bye :
                              frman.remove(ag);
                              AgentMap.remove(msg.src.both());
                              break;
                        default:
                              //
                  }
                  ag.textArea.requestFocus();
            }
        }
}
```

HostFrameManager.java

```
// Agent Monitor indicates agent presence and their status

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class HostFrameManager {
        JFrame frame;
        JPanel panel;
        String newline = System.getProperty("line.separator");

        HostFrameManager(){
                try {
                        UIManager.setLookAndFeel(

        UIManager.getCrossPlatformLookAndFeelClassName()
                        );
                }
                catch (Exception e){
                }

                frame = new JFrame("Agent Host");
                panel = new JPanel();
                frame.getContentPane().add(panel, BorderLayout.CENTER);
                frame.setVisible(true);
        }

        void initAgent(Agent ag){
                ag.textArea = new JTextArea("<Started>"+newline);
                ag.textArea.setEditable(false);
                ag.textArea.setFont(new Font("Serif", Font.ITALIC, 16));
                ag.textArea.setLineWrap(true);
                ag.textArea.setWrapStyleWord(true);

                ag.areaScrollPane = new JScrollPane(ag.textArea);
                ag.areaScrollPane.setVerticalScrollBarPolicy(
                        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
                ag.areaScrollPane.setPreferredSize(new Dimension(150,
180));
                ag.areaScrollPane.setBorder(
                        BorderFactory.createCompoundBorder(
                                BorderFactory.createCompoundBorder(

        BorderFactory.createTitledBorder(ag.name),

        BorderFactory.createEmptyBorder(5,5,5,5)),
                                        ag.areaScrollPane.getBorder()));
                panel.add(ag.areaScrollPane);

                ag.textArea.requestFocus();
                frame.pack();
                frame.setVisible(true);
```

```
	}

	void remove(Agent ag) {
		panel.remove(ag.areaScrollPane);
		frame.pack();
	}
}
```

MyAgent.java

```
// Personal travel assistant

import java.util.*;
import javax.swing.*;

class myAgent extends Agent {

	Channel GUIch;
	myAgentGUI gui;
	Map ruleMap;
	Map importanceMap;

	myAgent(){
		address = new AgentAddress("localHost/","myAgent");

		name = "PTA";
		GUIch = new Channel();
		gui = new myAgentGUI(GUIch);
		ruleMap = new HashMap();
		importanceMap = new HashMap();
	}

	public void run() {
		while(true) {
			AgentMessage msg = (AgentMessage) GUIch.get();	//
react to GUI interface
			switch( msg.act ) {
				case acts.get:
					get(msg);
					break;
			}
		}
	}

	void get(AgentMessage msg){
		gui.pp.remove(gui.pp.go);
		gui.tp.actionPane.setVisible(false);
		gui.pack();
		msg.src = address;
		List lst = new LinkedList();
		lst.add(
			new AgentAddress (
				"www.universalist.com/agent/",
				"www_universalist_com_agent"
			)
		);
		getProductList(msg,lst);
```

```
        productSpec pd = (productSpec) msg.obj;
        List maplst = new ArrayList();
        maplst.add(pd.maps.get(0));
        productSpec spec = new productSpec(pd.productName,
              pd.columnNames, maplst, pd.rules, pd.importance);
        gui.pp.remove(gui.pp.tblPane);

        Iterator iter = pd.rules.iterator();
        Iterator iterImp = pd.importance.iterator();
        while( iter.hasNext() ){
              Rule r = (Rule)iter.next();
              Double imp = (Double) iterImp.next();
              if( ! ruleMap.containsKey(r.id) ) {
                    ruleMap.put(r.id,r);
                    importanceMap.put(r.id, imp);
              }
        }

        gui.pp.productName.setSelectedItem(pd.productName);
        gui.pp.remove(gui.pp.tblPane);
        gui.pp.remove(gui.pp.go);

        // add new table to map
        if( pd.maps.size() > 1 )
              gui.pp.addItem(spec);

        pd.rank(ruleMap, importanceMap);


        gui.pp.createProductTable(pd);
        gui.pp.add(gui.pp.prodTblPane);
        gui.pp.add(gui.pp.decision);
        gui.pack();


        // Waiting for user reply
        AgentMessage reply = (AgentMessage) GUIch.get();
        if( reply.act == acts.inform ) {
              Map mp = (Map) reply.obj;
              pd.calibrate(mp);
              reply.src = address;
              reply.dst = (AgentAddress) mp.get("_agent");
              reply.obj = reply.obj;
              reply.act = acts.buy;
              out.put(reply);
              AgentMessage confirmBuy = (AgentMessage) in.get();
              if( confirmBuy.act == acts.inform )
                    gui.pp.prodTblPane.confirm(true);
              else
                    gui.pp.prodTblPane.confirm(false);
        }
        gui.restart();
  }

  void getProductList(AgentMessage msg, List lst){
        Iterator iter = lst.iterator();
        while( iter.hasNext() ){
              msg.dst = (AgentAddress) iter.next();
              out.put(msg);
```

```
                    AgentMessage reply=(AgentMessage) in.get();
                    if( reply.act == acts.refuse) {
                    }
                    else if (reply.act == acts.inform) {
                          msg.obj=reply.obj;
                    }
                    else if (reply.act == acts.go) {
                          getProductList(msg, (List) reply.obj);
                    }
              }
        }
}
```

MyAgentGUI.java

```
// User interface

import javax.swing.*;
import javax.swing.table.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

class bottomPane extends JPanel implements ActionListener{
      myAgentGUI ag;
      bottomPane(myAgentGUI a){
            ag = a;
            JButton     go = new JButton("Go");
            go.addActionListener(this);
            add(go);
      }
      public void actionPerformed(ActionEvent e){
            productPane ppane = ag.pp;
            ppane.productName.setEnabled(false);
            productSpec pd = new productSpec( (String)
ppane.selected,
                  ag.pp.tblPane.tbl );
            AgentMessage msg = new AgentMessage();
            msg.act = acts.get;
            msg.obj = pd;
            ag.chan.put(msg);
      }
}

class topPane extends JPanel implements ActionListener {
      myAgentGUI frame;
      JPanel actionPane;
      JComboBox actionList;
      String[] actions={      "Select One",
                              "Find Travel Products",
                              "Track Transactions",
                              "Schedule Actions",
                              "Exit"
                        };

      topPane(myAgentGUI f){
            frame = f;

            ImageIcon smile=new ImageIcon("smile.gif");
            JLabel title = new JLabel(smile);

            title.setPreferredSize(new Dimension(90,70));

            JPanel titlePane = new JPanel();
            titlePane.add(title);
            JLabel actionLabel = new JLabel("Services: ");
            actionList = new JComboBox(actions);
            actionList.setSelectedIndex(0);
            actionList.addActionListener(this);
            actionPane = new JPanel();
            actionPane.add(actionLabel);
            actionPane.add(actionList);
        add(titlePane, BorderLayout.WEST);
        add(actionPane, BorderLayout.CENTER);
        setPreferredSize(new Dimension(340,100));
    }

    public void actionPerformed(ActionEvent e) {

        JComboBox cb = (JComboBox)e.getSource();
      String selectedAction = (String)cb.getSelectedItem();

      if( selectedAction == actions[4] )
            System.exit(0);

      if( selectedAction == actions[1]  ) {
            if( frame.pp.visible==false ){
                  frame.pp.visible = true;
                  frame.getContentPane().add( frame.pp );
            }
      }

      else if( selectedAction == actions[0] ) {
            if( frame.pp.visible==true ) {
                  frame.pp.visible=false;
                  frame.getContentPane().remove( frame.pp );
            }
      }

      else {
            JOptionPane.showMessageDialog(frame,
                        "Service under development.", "Alert",
                        JOptionPane.ERROR_MESSAGE );
                  actionList.setSelectedIndex(0);
                  if( frame.pp.visible==true ) {
                  frame.pp.visible=false;
                  frame.getContentPane().remove( frame.pp );
            }
      }
            frame.pack();
    frame.repaint();
    }
}
```

```
class featureTablePane extends JPanel {
      featureTableModel tbl;
      String[] columnNames = {"Feature Name", "Feature Value"};
      int rows=30;
      int cols=2;
      productSpec prod;
      featureTablePane(){
            prod = new productSpec("eg Ticket Air", columnNames,
100);

            prod.set(0,0, "eg From");
            prod.set(0,1, "LAX");
            prod.set(1,0, "eg Destination");
            prod.set(1,1, "London");
            prod.set(2,1, "eg Return");
            prod.set(3,0, "eg Depart date");
            prod.set(3,1, "Nov 8");
            prod.set(4,0, "eg Return day");
            prod.set(4,1, "28/11");
            prod.set(9,0, "Click Go when completed.");
            JTable featureTable = new JTable(
                  (tbl=new featureTableModel(prod))
            );
            JScrollPane scrollPane = new JScrollPane(featureTable);
            featureTable.setPreferredScrollableViewportSize(new
Dimension(300, 170));
            add(scrollPane);
      }
      featureTablePane(productSpec pd){
            //turn table 90 deg
            productSpec newpd = new productSpec(pd.productName,
columnNames,

      pd.columnNames.length+10);
            for(int i=0; i<pd.columnNames.length; i++) {
                  newpd.set( i, 0, pd.columnNames[i] );
                  newpd.set( i, 1, pd.get(0,i) );
            }

            JTable featureTable = new JTable(
                  (tbl=new featureTableModel(newpd))
            );
            JScrollPane scrollPane = new JScrollPane(featureTable);
            featureTable.setPreferredScrollableViewportSize(new
Dimension(300, 170));
            add(scrollPane);
      }
}

class productTablePane extends JPanel {

      myAgentGUI gui;
      featureTableModel tbl;
      String[] columnNames;
      Object[][] data;
      productSpec prod;
      int selectedRow = -1;
      productTablePane(productSpec pd, myAgentGUI g){
            gui = g;
            prod = pd;
            tbl=new featureTableModel(pd);
            tbl.setEditable(false);
            JTable featureTable = new JTable(tbl);

      featureTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTI
ON);

            ListSelectionModel rowSM =
featureTable.getSelectionModel();
            rowSM.addListSelectionListener(new
ListSelectionListener() {
                public void valueChanged(ListSelectionEvent e) {
                    ListSelectionModel lsm =
                        (ListSelectionModel)e.getSource();
                    if (lsm.isSelectionEmpty()) {
                        System.out.println("no rows are selected");
                    }
                    else {
                        selectedRow = lsm.getMinSelectionIndex();
                        System.out.println("selected " +
selectedRow);
                    }
                }
            });

            JScrollPane scrollPane = new JScrollPane( featureTable,
                  JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
                  JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED
            );
            featureTable.setPreferredScrollableViewportSize(new
Dimension(800, 150));

            TableColumn column = null;
            for ( int i = 0; i < tbl.getColumnCount(); i++) {
                column = featureTable.getColumnModel().getColumn(i);
                column.setPreferredWidth(100);
            }

            add(scrollPane);
      }

      void buy(){
            if( selectedRow < 0 )
                  JOptionPane.showMessageDialog(gui,
                        "Click on row to select.", "Alert",
JOptionPane.ERROR_MESSAGE );
            else {
                  int n = JOptionPane.showConfirmDialog(gui,
                         "Buy the highlighted product?", "Confirm",
                              JOptionPane.YES_NO_OPTION
                              );
                  if( n == 0 ){
                        AgentMessage msg = new AgentMessage();
                        msg.act = acts.inform;
                        msg.obj = prod.maps.get(selectedRow);
                        gui.chan.put(msg);
                        System.out.println("buy");
                  }
```

```
                        else {
                                AgentMessage msg = new AgentMessage();
                                msg.act = acts.refuse;
                                gui.chan.put(msg);
                                System.out.println("not buy");
                        }
                }
        }

        void confirm(boolean success){
                if( success )
                        JOptionPane.showMessageDialog(gui, "Bought row " +
selectedRow , "Confirmed",
                                                        JOptionPane.INFORMATION_MESSAGE
);
                else
                        JOptionPane.showMessageDialog(gui,
                                "Product not available.", "Alert",
JOptionPane.ERROR_MESSAGE );
        }
}

class decisionPane extends JPanel {
        myAgentGUI frm;
        decisionPane(myAgentGUI f){
                frm = f;
                JButton     buy = new JButton("Purchase Highlighted");
                JButton cancel = new JButton("Cancel");
                buy.addActionListener( new ActionListener()  {
                        public void actionPerformed(ActionEvent e){
                                System.out.println("buy");
                                frm.pp.prodTblPane.buy();
                        }
                });
                cancel.addActionListener( new ActionListener() {
                        public void actionPerformed(ActionEvent e){
                                System.out.println("cancel");
                                AgentMessage msg = new AgentMessage();
                                msg.act = acts.refuse;
                                frm.chan.put(msg);
                        }
                });
                add(buy);
                add(cancel);
        }
}

class productPane extends JPanel{
        bottomPane go;
        myAgentGUI frm;
        JPanel prodlbl;
        JComboBox productName;
        String defaultSel = "e.g. Air Ticket";
        String selected = defaultSel;
        featureTablePane tblPane;
        productTablePane prodTblPane;
        java.util.List    productList;
        java.util.Map     tblMap;
        decisionPane      decision;
```

```
	java.util.List featLst;
	boolean visible;

	productPane(myAgentGUI f){

		productList = new ArrayList();
		tblMap = new HashMap();
		go = new bottomPane(f);
		frm = f;
		featLst = new ArrayList();
		visible=false;
		JLabel productLabel = new JLabel("Product Name");
		String[] none = {selected};
		productName = new JComboBox(none);

		productName.addActionListener( new ActionListener() {
			public void actionPerformed(ActionEvent e){
				//System.out.println("detected");
				remove(go);
				remove(tblPane);
				frm.pack();
				JComboBox cb = (JComboBox)e.getSource();
			selected = (String)cb.getSelectedItem();
			if( tblMap.containsKey(selected) )
					tblPane = (featureTablePane)
tblMap.get(selected);
				else
					tblPane = (featureTablePane)
tblMap.get(defaultSel);
				add(tblPane);
				add(go);
				frm.pack();
			}
		});

		productName.setEditable(true);

		prodlbl = new JPanel();
		prodlbl.add(productLabel);
		prodlbl.add(productName);

		tblPane = new featureTablePane();
		tblMap.put(selected, tblPane);
		decision = new decisionPane(frm);

		BoxLayout contentBox = new BoxLayout(this,
BoxLayout.Y_AXIS);
		setLayout(contentBox);
		add(prodlbl);
		add(tblPane);
		add(go);
	}

	void createProductTable(productSpec pd){
		prodTblPane = new productTablePane(pd,frm);
	}

	void createFeatureTable(productSpec pd){
		tblPane = new featureTablePane(pd);
```

```
            tblMap.put(pd.productName, tblPane);
      }

      void addItem(productSpec pd){
            if( ! tblMap.containsKey(pd.productName) ) {
                  productName.addItem(pd.productName);
                  remove(tblPane);
                  remove(go);
                  tblPane = new featureTablePane(pd);
                  tblMap.put(pd.productName, tblPane);
            }
      }
}

class myAgentGUI  extends JFrame {
      productPane pp;
      Channel chan;
      topPane tp;

      myAgentGUI(Channel ch){

            super("Personal Travel Assistant");
            chan = ch;

            pp = new productPane(this);
        tp = new topPane(this);

        JPanel contentPane = new JPanel();
            BoxLayout contentBox = new BoxLayout(contentPane,
BoxLayout.Y_AXIS);
            contentPane.setLayout(contentBox);
            contentPane.add(tp);
            setContentPane(contentPane);
            addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                  System.exit(0);
            }
            });
            pack();
            setVisible(true);
      }
      void restart(){
            pp.remove(pp.prodTblPane);
            pp.remove(pp.decision);
            pp.add(pp.tblPane);
            pp.add(pp.go);
            pp.productName.setEnabled(true);
            getContentPane().remove( pp );
            tp.actionPane.setVisible(true);
            tp.actionList.setSelectedIndex(0);

            pack();
      }

}
```

Node.java

```
// Tree node definition

package lexical;
import java.util.*;

public class node {
      public String     name;
      List  branches;
      public node(String s, List l) {
            name = s;
            branches = l;
      }
}
```

Ontology.java

```
// Match ontology trees

package lexical;
import java.util.*;
import java.io.*;

class wordDef {
      node keyTree;

      wordDef(Object nd, Map mp){
            keyTree = (node) nd;
            put(mp, keyTree.name.toLowerCase());
            enter(keyTree.branches, mp);
      }

      void enter(List lst, Map mp){
            Iterator it = lst.iterator();
            while( it.hasNext() ){
                  node nd = (node) it.next() ;
                  put(mp, nd.name);
                  enter(nd.branches, mp);
            }
      }

      void put(Map mp, String s){
            if( mp.containsKey(s) ) {
                  Set st = (Set) mp.get(s);
                  st.add(this);
            }
            else {
                  Set st = new HashSet();
                  st.add(this);
                  mp.put(s,st);
            }
      }
```

```
}

public class ontology{
      Map mp;

      public ontology(String fileName)
            throws IOException {
            mp = new HashMap();
            parseTree p = new parseTree(fileName);
            Iterator ent = p.treeList.iterator();
            while( ent.hasNext() )
                  new wordDef(ent.next(), mp);
      }


      public String search(String allcase) throws IOException {
            String s = allcase.toLowerCase(); // Root def need upper
case
            StreamTokenizer parser = new StreamTokenizer(new
StringReader(s));
            parser.lowerCaseMode(true);
            String def = null; // Def not found
            while( parser.nextToken() != parser.TT_EOF ) {
                  if(parser.ttype == parser.TT_WORD )  { //skip
numbers
                        Set entrySet = (Set) mp.get(parser.sval);
                        if (entrySet != null) {
                              def = match(entrySet,s);
                              if( def != null )
                                    break; // first match is enough
                        }
                  }
            }
            return def;
      }

      String match(Set st, String s){
            Iterator itEntry = st.iterator();
            String def=null;
            while( itEntry.hasNext() ) {
                  wordDef ent = (wordDef) itEntry.next();
                  /*
                  if( s.indexOf(ent.keyTree.name.toLowerCase()) >=0 )
{ // near exact match
                        def = ent.keyTree.name;
                        break;
                  }
                  */
                  if( matchEntry(ent,s) ) { // single match
                        def = ent.keyTree.name;
                        break;
                  }
            }
            return def;
      }

      boolean matchEntry(wordDef ent, String s){
             return matchString(ent.keyTree.branches, s);
      }
      boolean matchString(List lst, String s){
            boolean result = false;

            Iterator it = lst.iterator();
            while( it.hasNext() ) {
                  node nd = (node) it.next();
                  if( s.indexOf(nd.name) >= 0 ) {
                        if( nd.branches.isEmpty() )
                              result= true;
                        else
                              result= matchString(nd.branches,s);
                  }
            }
            return result;
      }
}
```

Parse.java

```
// Parse a string into tokens

package lexical;

import java.util.*;
import java.io.*;

public class parse {

      public List wordList;

      public parse(String s)
      throws IOException {
            wordList = new ArrayList();
            StreamTokenizer p = new StreamTokenizer(new
StringReader(s));
            p.lowerCaseMode(true);
            while( p.nextToken() != p.TT_EOF )
                  if(p.ttype == p.TT_WORD )
                        wordList.add(p.sval);
      }
}
```

ParseDate.java

```
// Flexible date and time format

import java.util.*;
import java.text.*;

class parseDate {
      String[] datefmt = {"d MMM y", "d M y", "d/M/y", "d.M.y",
                  "d M", "d/M", "MMM d", "d MMM"};
      String[] timefmt = {"h:m a", "h a", "HHmm", "H m", "H"};
      Date go(String s, boolean findtime) {
```

```
		SimpleDateFormat df;
		Date mydate = new Date();
		String[] fs = datefmt;
		if(findtime) fs = timefmt;
		for(int i=0; i<fs.length; i++) {
			df = new SimpleDateFormat(fs[i]);
			ParsePosition pos = new ParsePosition(0);
			try{
				Date d = df.parse(s,pos);
				mydate = d;
				String a = df.format(mydate);
				//System.out.println( a );
				break;
			}
			catch( Exception e) {
			}
		}
		return mydate;
	}

	String fmt(Date dd, boolean time) {
		String datefmt = "MMM d";
		String timefmt = "h:m a";
		String fmt;
		if( time ) fmt = timefmt;
			else fmt = datefmt;
		SimpleDateFormat formatter = new SimpleDateFormat(fmt);
		return formatter.format(dd);
	}

	public static void main(String[] args) {
		parseDate p = new parseDate();
		//System.out.println(p.go(args[0],true).toString());
	}
}
```

ParseTree.java

```
// Ontology tree builder

package lexical;

import java.util.*;
import java.io.*;

public class parseTree {

	public List treeList;

	public parseTree(String fileName)
		throws IOException {
		treeList = new ArrayList();
		FileReader fr = new FileReader(fileName);
		StreamTokenizer p = new StreamTokenizer(fr);
		p.lowerCaseMode(true);
		p.wordChars('_','_');
		p.wordChars('(','(');
            p.wordChars(')',')');
            p.wordChars('/','/');
            p.whitespaceChars('<','<');
            p.whitespaceChars('>','>');
            p.quoteChar('"');

            buildTree(p,treeList);
            fr.close();
      }

      void buildTree(StreamTokenizer tok, List lst)
            throws IOException {
            while( tok.nextToken() != tok.TT_EOF ) {
                  if(tok.ttype == tok.TT_WORD || tok.ttype == '"' )
{ //skip numbers
                        if( tok.sval.indexOf( "(" ) >= 0 ) {
                              buildTree(tok,
((node)lst.get(lst.size()-1)).branches );
                        }
                        else if (tok.sval.indexOf( ")" ) >= 0 )
                              break;
                        else
                              lst.add( new node(tok.sval, new
LinkedList() ) );
                  }
            }
      }

      public void displayTree(List lst){
            ListIterator it = lst.listIterator();
            while( it.hasNext() ) {
                  node nd = (node) it.next();
                  //System.out.println((String) nd.name);
                  if( ! nd.branches.isEmpty() ) {
                        System.out.println( '{' );
                        displayTree((List)nd.branches);
                        System.out.println( '}' );
                  }
            }
      }
}
```

ProductSpec.java

```
// Product specification definition

import java.util.*;
import javax.swing.table.*;

class productSpec {
      String            productName;
      String[]    columnNames;
      List        maps;
      List        rules;
      List        importance;
      compareScore cmp;
      Map         rlMap;
```

```
	Map		imMap;

	productSpec(String s, String[] names, int rows ){
		productName=s;
		columnNames = names;
		maps = new ArrayList();
		rules = new ArrayList();
		importance = new ArrayList();
		for(int i=0; i<rows; i++){
			maps.add( new HashMap() );
		}
		cmp = new compareScore();
	}

	productSpec(String name, String[] cols, List mp, List rl) {
		productName = name;
		columnNames = cols;
		maps = mp;
		rules = rl;
		cmp = new compareScore();
	}

	productSpec(String name, String[] cols, List mp, List rl, List
imp) {
		productName = name;
		columnNames = cols;
		maps = mp;
		rules = rl;
		importance=imp;
		cmp = new compareScore();
	}

	productSpec (String name, TableModel tbl ){
		productName = name;
		maps = new ArrayList();
		rules = new ArrayList();
		importance = new ArrayList();
		Map mp = new HashMap();
		for(int r=0; r<tbl.getRowCount(); r++){
			Object fld = tbl.getValueAt(r,0);
			Object val = tbl.getValueAt(r,1);
			if( fld == null && val == null )
				break; // end of table
			if( fld==null )
				fld = "unknown" + r;  // make unique key
			mp.put(fld,val);
		}
		maps.add(mp);
		Object[] obj = mp.keySet().toArray();
		columnNames = new String[obj.length];
		for(int i=0; i<obj.length; i++)
			columnNames[i] = (String) obj[i];
		cmp = new compareScore();
	}


	void set(int row, int col, Object obj){
		((Map)maps.get(row)).put(columnNames[col],obj) ;
	}
```

```
Object get(int row, int col ) {
      return ((Map)maps.get(row)).get(columnNames[col]);
}

String getColumnName(int col){
      return columnNames[col];
}

int rowCount(){
      return maps.size();
}

int columnCount(){
      return columnNames.length;
}

void rank(Map ruleMap, Map importanceMap){
      rlMap = ruleMap;
      imMap = importanceMap;
      Map spec = (Map) maps.get(0);
      maps.remove(0);
      for( int i=0; i<maps.size(); i++ ){
            Map row= (Map) maps.get(i);
            row.put("_score", new Double(0.0) );
            row.put("_selected", new Boolean (true) );
            row.put("_key", new Integer(i) );
      }

      Set keys = ruleMap.keySet();
      Iterator iter = keys.iterator();
      while( iter.hasNext() ) {
            String id = (String) iter.next();
            Rule r = (Rule) ruleMap.get(id);
            Double imp = (Double) importanceMap.get(id);
            r.input(spec,maps);
            r.pass();
            r.score(imp.doubleValue());
            System.out.println( imp.doubleValue() + " " + id );
      }

      Collections.sort(maps,cmp);
      for(int i=0; i<maps.size(); i++) {
            Map row = (Map) maps.get(i);
            System.out.print( row.get("_score") );
            System.out.println( " " + row.get("_selected") );
      }
}

void calibrate(Map record){
      Set keys = rlMap.keySet();
      Iterator it = keys.iterator();
      while(it.hasNext()) {
            String id = (String) it.next();
            Rule r = (Rule) rlMap.get(id);
            Double im = (Double) imMap.get(id);
            r.calibrate( record );
            if( topScore( r,record ) ) {
                  double x = im.doubleValue();
```

```
                        x = x * 1.1;
                        imMap.put(id, new Double(x));
                    }
                }
            }

        boolean topScore( Rule rl, Map data ) {
                boolean result = true;
                double max = ( (Double)data.get("_score"+rl.id)
).doubleValue() ;
                Iterator it = maps.iterator();
                while(it.hasNext()) {
                        Map mp = (Map) it.next();
                        boolean pass = ( (Boolean)
mp.get("_selected"+rl.id)).booleanValue();
                        double score = ( (Double)
mp.get("_score"+rl.id)).doubleValue();
                        if( ! pass )
                                result = false;
                        else if (max <=0.0 )
                                result = false;
                        else if ( score > max )
                                        result = false;
                        //System.out.println( rl.id + " " + max + " " +
score + " " + result);
                }
                return result;
        }
}

class compareScore implements Comparator {
        public int compare(Object o1, Object o2){
                int result=-1;
                double first = ((Double)
((Map)o1).get("_score")).doubleValue();
                double second = ((Double)
((Map)o2).get("_score")).doubleValue();
                Boolean in1st = (Boolean) ((Map)o1).get("_selected");
                Boolean in2nd = (Boolean) ((Map)o2).get("_selected");
                if( in1st.booleanValue() == in2nd.booleanValue() ) {
                        if( first < second ) // desending order
                                result = 1;
                        else if( first == second )
                                result = 0;
                        else
                                result = -1;
                }

                else {
                        if( in2nd.booleanValue() )  // second selected, 1st
< 2nd
                                result = 1;
                        else
                                result = -1;
                }

                return result;
        }
}
```

Rule.java

```
// Common rule definitions

import java.util.*;

abstract class Rule {
	String	id;
	Map query;
	List data;

	public void input(Map q, List d){
		query=q;
		data=d;
	}

	public void score(double weight)	{
		Iterator it = data.iterator();
		while( it.hasNext() ) {
			Map record = (Map) it.next();
			record.put("_score"+id, new Double(0.0) );
		}
	}

	public void pass()	{
		Iterator it = data.iterator();
		while( it.hasNext() ) {
			Map record = (Map) it.next();
			record.put("_selected"+id, new Boolean(true) );
		}
	}
	public void calibrate(Object selected) {}

	double findMin(String key) {
		Iterator it = data.iterator();
		double val;
		double min = Double.MAX_VALUE;
		while(it.hasNext()){
			val = ((Double)
((Map)it.next()).get(key)).doubleValue();
			if( val < min )
				min = val;
		}
		return min;
	}

	boolean containString(Object obj, String s){
		return	obj.toString().indexOf(s) >= 0 ;
	}
}
```

SimpleClassLoader.java

```
// Load class from file
```

```
public class SimpleClassLoader extends ClassLoader {

      public SimpleClassLoader() {
    }

    public synchronized Class loadClass(String className)
    throws ClassNotFoundException {
        return super.findSystemClass(className);
    }
}
```

StartHost.java

```
// Main program entry point

public class StartHost {
    public static void main(String[] args) {
      Host h= new Host();
        h.start();
    }
}
```

Travelist.txt

```
// Ontology + addresses for Travel Listings Agent

supplier (
      address (
            www.airnet.com/agent/
            www_airnet_com_agent
      )
      products (
            ticket      (
                  fly
                  air
                  airline
                  plane
                  flight
            )
      )
)

supplier (
      address (
            www.virginatlantic.com/agent/
            www_virginatlantic_com_agent
      )
      products (
            ticket      (
                  fly
                  air
                  airline
                  plane
                  flight
            )
      )
)
```

Universalist.txt

```
// Ontology + addresses for Universal Listings Agent

directory (
      address (
            www.travelist.com/agent/
            www_travelist_com_agent
      )
      products (
            ticket      (
                  fly
                  air
                  airline
                  plane
                  sea
                  cruise
                  train
                  flight
            )
            holiday (
                  ski
                  summer
                  cruise (
                        carribean
                  )
            )
      )
)

supplier (
      address (
            www.localserve.com/agent/
            www_localserve_com_agent
      )
      products (
            hotel
            car (
                  hire
                  rental
            )

      )
)
```

URLAlias.java

```
// Map hypothetical agent addresses to real agent addresses

import java.net.*;
```

```
class URLAlias {
      URL u;
      URLAlias(URL v){
            String fl = v.getFile();
            try {
                  u = new URL( "http://" +
"www.jewelrywonderland.com" + fl );
            }
            catch(Exception e){
                  System.out.println("URL Alias Error " + e );
                  System.exit(1);
            }
      }

      URL alias(){
            return u;
      }
}
```

URLloader.java

```
// Load class via http servers

import java.net.*;
import java.io.*;

class URLloader {
      Class loadClass(  AgentAddress adr ) {
            SimpleClassLoader scl = new SimpleClassLoader();
            Class cl=null;
            try {  // local or foreign agent already loaded in case
demo server fails
                  cl = scl.loadClass(adr.fileName);
                  System.out.println("Loaded class: " + "localHost/"
+ adr.fileName );
            }
            catch (ClassNotFoundException e) {
                  try {
                        URL u = new URL( "http://" + adr.path +
adr.fileName + ".class" );
                        URLAlias v = new URLAlias(u);
                        InputStream in =  v.alias().openStream();
                        FileOutputStream out = new
FileOutputStream(adr.fileName + ".class" );
                        int by;
                        while  (  (by = in.read()) != -1 )
                        out.write( by );
                        in.close();
                        out.close();
                        cl = scl.loadClass(adr.fileName);
                        System.out.println("Loaded class: " +
adr.path + adr.fileName );
                  }
                  catch(Exception err){
                        System.out.println("Load Error: " + err );
                        System.exit(1);
                  }
            }
            return(cl);
      }
}

/*java -Dhttp.proxyHost=proxyhost [-Dhttp.proxyPort=portNumber]
URLReader
*/
```

VirginAtlantic.txt

```
// Supplier ontology

"Flight Ticket" (
      ticket (
            fly
            air
            airline
            flight
            plane
      )
)

"Airlines" (
      airlines
      airline
      air (
            line
      )
)

"Destination" (
      destination
      to
      dest
      dst
      arrive (
            location
            place
            airport
      )
)

"Origination" (
      origination
      from
      origin
      depart (
            location
            place
            airport
      )
)

"Return Departure Time" (
      "return departure time"
      return (
            time
            time (
                  depart
                  leave
                  from
            )
      )
)

"Departure Time" (
      "departure time"
      time (
            depart
            leave
            from
      )
)

"Return Arrival Time" (
      "return arrival time"
      return (
            time (
                  arrival
                  arrive
                  to
            )
      )
)

"Arrival Time" (
      "arrival time"
      time (
            arrival
            arrive
            to
      )
)

"Return Departure Date" (
      "return departure date"
      return (
            date (
                  depart
                  leave
                  from
            )
            day (
                  depart
                  leave
                  from
            )
            day
            date
      )
)

"Departure Date" (
      "departure date"
      date (
            depart
            leave
            from
      )
      day (
            depart
            leave
            from
      )
)

"Return Arrival Date" (
      "return arrival date"
      return (
            date (
                  arrival
                  arrive
                  to
            )
            day (
                  arrival
                  arrive
                  to
            )
      )
)

"Arrival Date" (
      "arrival date"
      date (
            arrival
            arrive
            to
      )
      day (
            arrival
            arrive
            to
      )
)

"Equipments" (
      equipments
      equipment
      equipt
      plane (
            type
            body
      )
)

"Fare" (
      cost
      price
      fare
)

"Trip" (
      trip
      featureof (
            single
            return
            round (
                  trip
            )
      )
)

"Single" (
      valueof (
            single
            one ( way )
      )
)

"Return" (
      valueof (
            return
            round (
                  trip
            )
      )
)

"Legs" (
      leg
      featureof (
            direct
            indirect
      )
)

"Direct" (
      valueof (
            direct
      )
)

"Indirect" (
      valueof (
            indirect
      )
)

"Stops" (
      stops
      featureof (
            stop ( non no )
            "non-stop"
            nonstop
      )
)

"Non-stop" (
      valueof (
            stop ( non no )
            "non-stop"
            nonstop
```

```
        )
)
```

VirginAtlantic.data

```
// Flight database of Virgin Atlantic

Trip                Fare
Origination         Destination
"Departure Date"    "Departure Time"
"Arrival Date"            "Arrival Time"
"Return Departure Date" "Return Departure Time"
"Return Arrival Date"   "Return Arrival Time"
Airlines            Equipments
Legs                Stops

DATA

Return      560.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"    "5:30 pm"    "1"    "11:45 am"
"0"    "3:05 pm"    "1"    "6:40 pm"
VS     "BOEING 747 AIRBUS JET"
2      0

Return      590.5
"LOS ANGELES CA LAX "    "GATWICK LONDON UNIT LGW AUX "
"0"    "7:24 pm"    "1"    "9:15 am"
"0"    "11:15 am"   "0"    "7:10 pm"
US      "BOEING 767 BOEING 757"
3      0

Return      620.5
"LOS ANGELES CA LAX "    "GATWICK LONDON UNIT LGW AUX "
"0"    "12:15 pm"   "1"    "10:15 am"
"0"    "12:25 pm"   "0"    "8:15 pm"
US      "AIRBUS BOEING 767 BOEING 757"
4      0

Return      762.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"    "12:50 pm"   "1"    "7:10 am"
"0"    "11:55 am"   "0"    "3:05 pm"
UA      "BOEING 777"
2      0

Return      762.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"    "12:50 pm"   "1"    "7:10 am"
"0"    "2:45 pm"    "0"    "8:48 pm"
UA      "BOEING 777 BOEING 737"
3      0

Return      762.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"    "2:53 pm"    "1"    "11:25 am"
"0"    "2:45 pm"    "0"    "8:48 pm"
```

```
UA     "BOEING 777 BOEING 737"
4      0

Return      765.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"   "12:50 pm"  "1"    "7:10 am"
"0"   "1:55 pm"   "0"    "7:51 pm"
UA     "BOEING 777 MCDONNELL DC10"
3      0

Return      768.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"   "2:00 pm"   "1"    "10:40 am"
"0"   "1:55 pm"   "0"    "7:51 pm"
UA     "BOEING 777 MCDONNELL DC10"
4      0

Return      768.4
"LOS ANGELES CA LAX "    "HEATHROW LONDON UNI LHR "
"0"   "1:00 pm"   "1"    "9:55 am"
"0"   "12:50 pm"  "0"    "7:40 pm"
UA     "BOEING 767 BOEING 777 BOEING 747"
4      0
```

WWW_airnet_com_agent.java

```
// Agent at www.airnet.com

import lexical.*;
import java.util.*;

class www_airnet_com_agent extends Agent {

      www_airnet_com_agent(){
            address= new AgentAddress("www.airnet.com/agent/",
"www_airnet_com_agent" );
            name = "Air Net";
      }

      public void run() {
            AgentMessage msg = (AgentMessage) in.get();
            if( msg.act == acts.get )
                  search(msg);
            else
                  out.put(new
      AgentMessage(msg.dst,msg.src,acts.refuse,"?"));
            bye();
      }

      void search(AgentMessage msg){
            productSpec prod = (productSpec) msg.obj ;
            Set st = new HashSet();  // empty


            AgentMessage reply = new
AgentMessage(address,msg.src,acts.inform,msg.obj);
            out.put(reply);
      }
}
```

WWW_travelist_com_agent.java

```
// Listings agent for travelling services

import lexical.*;
import java.util.*;

class www_travelist_com_agent extends Agent {

      www_travelist_com_agent(){
            address= new AgentAddress("www.travelist.com/agent/",
"www_travelist_com_agent" );
            name = "Travel Listings";
      }

      public void run() {
            AgentMessage msg = (AgentMessage) in.get();
            if( msg.act == acts.get )
                  where(msg);
            else
                  out.put(new
      AgentMessage(msg.dst,msg.src,acts.refuse,"?"));
            bye();
      }

      void where(AgentMessage msg){
            productSpec prod = (productSpec) msg.obj ;
            Set st = new HashSet();  // empty
            try {
                  directory d = new directory("travelist.txt") ;
                  st = d.search(prod.productName);
            }
            catch(Exception e){
                  //set empty on exception
            }

            AgentMessage reply = new
AgentMessage(address,msg.src,acts.refuse,null);
            if( st.isEmpty() ) {
                  reply.act = acts.refuse;
                  reply.obj = "No supplier found";
            }
            else {
                  reply.act = acts.go;
                  Iterator it = st.iterator();
                  List agrLst= new ArrayList();
                  while(it.hasNext()){
                        List adr = (List) it.next();
                        agrLst.add(
                              new AgentAddress(
((node)adr.get(0)).name , ((node)adr.get(1)).name )
                        );
                  }
                  reply.obj = agrLst;
            }
            out.put(reply);
      }
}
```

WWW_universalist_com_agent.java

```
// Listing agent at www.universalist.com

import lexical.*;
import java.util.*;

class www_universalist_com_agent extends Agent {

      www_universalist_com_agent(){
            address= new AgentAddress("www.universalist.com/agent/",
"www_universalist_com_agent" );
            name = "Universal Listings";
      }

      public void run() {
            AgentMessage msg = (AgentMessage) in.get();
            if( msg.act == acts.get )
                  where(msg);
            else
                  out.put(new
      AgentMessage(msg.dst,msg.src,acts.refuse,"?"));
            bye();
      }

      void where(AgentMessage msg){
            productSpec prod = (productSpec) msg.obj ;
            Set st = new HashSet();  // empty
            try {
                  directory d = new directory("universalist.txt") ;
                  st = d.search(prod.productName);
            }
            catch(Exception e){
                  //set empty on exception
            }

            AgentMessage reply = new
AgentMessage(address,msg.src,acts.refuse,null);
            if( st.isEmpty() ) {
                  reply.act = acts.refuse;
                  reply.obj = "No supplier found";
            }
            else {
                  reply.act = acts.go;
                  Iterator it = st.iterator();
                  List agrLst= new ArrayList();
                  while(it.hasNext()){
                        List adr = (List) it.next();
                        agrLst.add(
                              new AgentAddress(
((node)adr.get(0)).name , ((node)adr.get(1)).name )
                        );
```

```
                }
                reply.obj = agrLst;
            }
            out.put(reply);
        }
}
```

WWW_virginatlantic_com_agent.java

```
// Supplier agent at www.virginatlantic.com

import lexical.*;
import java.util.*;

//don't add rule if rule is there already. keep calibration

class www_virginatlantic_com_agent extends Agent {

        www_virginatlantic_com_agent(){
            address= new
AgentAddress("www.virginatlantic.com/agent/",
"www_virginatlantic_com_agent" );
            name = "Virgin Atlantic";
        }

        public void run() {
            AgentMessage msg = (AgentMessage) in.get();
            if( msg.act == acts.get )
                search(msg);
            else if ( msg.act == acts.buy )
                out.put(new
        AgentMessage(address,msg.src,acts.inform,"OK"));
            else
                out.put(new
        AgentMessage(msg.dst,msg.src,acts.refuse,"?"));
            bye();
        }

        void search(AgentMessage msg){
            try{
                ontology    ont = new
ontology("virginatlantic.txt");
                productSpec prod = (productSpec) msg.obj ;

                Map mp = (Map) prod.maps.get(0);
                Map newmp = new HashMap();  //cannot modify mp
during iteration
                Iterator it = mp.keySet().iterator();
                while( it.hasNext() ) {
                    String m = (String) it.next();
                    String v = (String) mp.get(m);
                    //System.out.println( m + " " + v);
                    if( m.indexOf("unknown")>=0 ) {  // empty key
                        //System.out.println("null " + v );
                        String fs = ont.search( "featureof " +
v );
                        if( fs != null ){
```

```
                                    newmp.put(fs,
ont.search("valueof" + v));
                                    System.out.println(fs + " " +
ont.search("valueof" + v));
                              }
                        }
                        else {
                              String s =  ont.search( m );
                              if( s != null )  // known key
                                    if( v != null ) // specified
                                          newmp.put(s, v );
                              System.out.println(s + " " + v);
                        }
                  }
                  database db= new database("virginatlantic.data");
                  //System.out.println("no. of data " +
db.rows.size());
                  //System.exit(0);
                  List results = db.search(newmp);
                  System.out.println("Match: " + results.size());
                  for(int i=0; i<results.size(); i++)
                        ((Map)results.get(i)).put("_agent",address);
                  results.add(0,newmp);  // first result is spec
                  String[] colNames =  db.fields();
                  List rulelst = new ArrayList();
                  rulelst.add( new airportRule() );
                  rulelst.add( new directRule() );
                  rulelst.add( new fareRule() );
                  List weightlst = new ArrayList();
                  Double weight = new Double(1.0);
                  weightlst.add(weight);
                  weightlst.add(weight);
                  weightlst.add(weight);
                  productSpec newprod = new productSpec(
                        ont.search(prod.productName), colNames,
results,
                        rulelst, weightlst
                  );
                  AgentMessage reply =
                        new
AgentMessage(address,msg.src,acts.inform,newprod);
                  out.put(reply);
            }
            catch (Exception e){
                  System.out.println("Internal database error: " +
e);
                  System.exit(0);
            }
      }
}
```

Bibliography

ActivMedia Incorporation, (1995a), *FutureScapes study,* Peterborough, NH, US, October, 1995, http://www.activmedia.com.

ActivMedia Incorporation, (1995b), *Executive Report,* Peterborough, NH, US, December, 1995, http://www.activmedia.com.

ActivMedia Incorporation, (1997a), *FutureScapes study,* Peterborough, NH, US, October, 1997, http://www.activmedia.com.

ActivMedia Incorporation, (1997b), *The Real Numbers behind 'Net Profits',* Peterborough, NH, US, 1997, http://www.activmedia.com.

Aglets Petition, (1999), http://luckyspc.lboro.ac.uk/Aglets/Petition/index.html.

Aglets, (1996), http://www.trl.ibm.co.jp/aglets.

American Society of Travel Agents, (1998), *Automation Survey,* May, 1998, http://www.astanet.com.

Andereck, K. L., (1992), Researching Consumer Information: Comments Researching Consumer Information, *Proceedings of the 23rd Annual Conference of Travel and Tourism Research Association,* June 14-17, 1992, Minneapolis, MN, pp. 50-52.

Angeline, P. J., (1995), Evolution Revolution: An Introduction to the Special Track on Genetic and Evolutionary Programming, *IEEE Expert,* June, pp. 6-10.

Arnold, K. and Gosling, J., (1997), *The Java Programming Language,* 2nd edition, Addison-Wesley.

Association of British Travel Agents, (1997), *IT Survey,* December, 1997, http://www.abtech.co.uk/itsurvey.

Aylett, R., Brazier, F., Jennings, N., Luck, M., Preist, C. and Nwana, H., (1998), Agent Systems and Applications, *The Knowledge Engineering Review,* 13(3), pp. 303-308.

Barbagallo, T., (1994), Intelligent Middleware: Data Overload Relief, *AI Expert,* September, pp. 38-41.

Barrett, E. D., (1986), Strategies for the Use of Artificial and Human Intelligence, *Business Quarterly,* No. 51, pp. 18-27.

Bates, J., (1994), The Role of Emotion in Believable Agents, *Communications of the ACM,* 37(7), July, pp. 122-125.

BizTravel.com, http://www.biztravel.com.

Boden, M. A., (1994), Agents and Creativity, *Communications of the ACM,* 37(7), July, pp. 117-121.

Bradshaw, J. M., Dutfield, S., Benoit, P. and Woolley, J. D., (1997), KAoS: Toward an Industrial Strength Open Agent Architecture, in Bradshaw, J. M. (ed.) *Software Agents,* MIT Press, Cambridge, Mass., pp. 375-418.

Bratman, M. E. Lsrael, D. J. and Pollack, M. E., (1988), Plans and Resource-bounded Practical Reasoning, *Computational Intelligence,* Volume 4, pp. 349-355.

Brown, C., Gasser, L., O'Leary, D. E., and Sangster, A., (1995), AI on the WWW: Supply and Demand Agents, *IEEE Expert,* August, pp. 50-56.

Bussmann, S. and Muller, J., (1992), A Negotiation Framework for Co-operating Agents, in Dean, S. M., (ed.), *Proceedings of CKBS-SIG,* Dake Centre, University of Keele, pp. 1-17.

Chang, D. and Lange, D., (1996), Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW, *Proceedings of the OOPSLA '96 Workshop*.

Chauhan, D. and Baker, A.D., (1998), JAFMAS: A Multi-agent Application Development System, in Wooldridge, M and Finn, T. (eds.), *Proceedings of Second ACM Conference on Autonomous Agents,* Minneapolis, MN, pp. 100-107.

Chavez, A. and Maes, P., (1996), Kasbah: An Agent Marketplace for Buying and Selling Goods, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-agent Technologies (PAAM-96),* London, U. K., pp. 75-90.

Cheaptickets, http://cheaptickets.com.

Chess, D., Grosof, B., Harrison, C., Levine, D., Parris, C. and Tsudik, G., (1995b), *Itinerant Agents for Mobile Computing,* Technical Report, October 1995, IBM, T.J. Watson Research Center, NY.

Chess, D., Harrison, C. and Kershenbaum, A., (1995a), *Mobile Agents: Are they a good idea?,* Technical Report, March 1995, IBM, T.J. Watson Research Center, NY.

Chin, D. N., (1991), Intelligent Interfaces as Agents, in Sullivan, J. and Tyler, S. (eds.), *Intelligent User Interfaces,* ACM Press, N. Y., pp. 177-205.

Chu-Carroll, J. and Carberry, S., (1995), Conflict Detection and Resolution in Collaborative Planning, in *Intelligent Agents II, Lecture Notes in Artificial Intelligent 1037,* Heidelberg: Springer Verlag.

Cohen, P., Morgan, J., Pollack, M. (eds.) (1990), *Intentions in Communication,* MIT Press, Cambridge, MA.

CommerceNet, (1998), *Success of XML for E-Commerce Accelerated by Advanced Knowledge Representation Techniques,* Palo Alto, CA, October 9, 1998, http://www.commercenet.com.

CommerceNet/Nielsen Media Research, (1999), *Top Web Site Properties and Advertisers for February 1999,* March 22, 1999, http://www.commercenet.com.

Concordia, (1997), http://www.concordia.mea.com

Cost, R. S., (1998), Jackal: A Java-Based Tool for Agent Development, *Working Papers of the AAAI '98 Workshop on Software Tools for Developing Agents,* AAAI Press.

Crabtree, B. and Stuart, S., (1998), Identifying and Tracking Changing Interests, *Journal of Digital Libraries.*

Crompton, J. L. and Ankomah, P. K., (1993), Choice Set Propositions in Destination Decisions, *Annals of Tourism Research,* 20, pp. 461-475.

Crompton, J. L., (1992), Structure of Vacation Destination Choice Sets, *Annals of Tourism Research,* 20, pp. 420-434.

Crowston, K. and Malone, T. W., (1988), Intelligent Software Agents, *BYTE,* 13(13), December, pp. 267-270.

Cypher, A., (1991), EAGER: Programming Repetitive Tasks by Example, *Proceedings of CHI 91,* ACM Press, N. Y., pp. 33-39.

DARPA, (1993), *Specification of KQML Agent Communication Language,* Technical Report, ARPA Knowledge Sharing Initiative, External Interfaces Working Group.

Datamonitor, (1998), *Travel will be Largest Online Product by 2002,* April 28, 1998.

Datamonitor, (1999), *Interactive Services Strategic Planning Programme,* 24 May, 1999, http://www.datamonitor.com.

DeBellis, M., (1995), User-Centric Software Engineering, *IEEE Expert,* February, pp. 34-41.

Dennet, D, (1978), *Brainstorms,* MIT Press, Bradford, 1978.

Dent, L. et al., (1992), A Personal Learning Apprentice, *Proceedings of the National Conference on Artificial Intelligence,* MIT Press, Cambridge, Mass., pp. 96-103.

Dombey, A., (1998), Separating the Emotion from the Fact – the Effects of New Intermediaries on Electronic Travel Distribution, in Buhalis, D and Jafari, J, (eds.), *Information and Communication Technologies in Tourism,* The Fifth ENTER 98 International Conference on Information and Communication Technologies in Tourism, Istanbul, 21-23 January, 1998, pp. 129-138.

Durfee, E. H. and Montgomery, T. A., (1990), A Hierarchical Protocol for Co-ordinating Multi-Agent Behaviours, *Proceedings of the 8th National Conference of Artificial Intelligence,* Boston, Mass., pp. 86-93.

Durfee, E., Lesser, V. and Corkill, D., (1989), Trends in Co-operative Distributed Problem Solving, *IEEE Knowledge & Data Engineering,* 1(1), pp. 63-83.

Edmonds, E. A., Candy, L., Jones, R. and Soufi, B., (1994), Support for Collaborative Design: Agents and Emergence, *Communications of the ACM,* 37(7), July, pp. 41-47.

Elias, E., (1999), *Internet Commerce Transforming the Travel Industry,* SRI Consulting, CommerceNet Research Report #99-34.

Eliot, L., (1994), Intelligent Agents are Watching You, *AI Expert,* August, pp. 9-11.

Ephrati, E. and Rosenschein, J. S., (1992), Constrained Intelligent Action: Planning under the Influence of a Master Agent, *Proceedings of the 10th National Conference on Artificial Intelligence,* San Jose, California, July, 1992, pp. 263-268.

Equinus, (1998), *Consumer Web Survey,* http://www.equinus.com.

Etzioni, O, (1996), Moving up the Information Food Chain: Deploying Softbots on the World Wide Web, *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI '96),* Portland, OR.

Etzioni, O. and Weld, D. S., (1994), A Software-Based Interface to the Internet, *Communications of the ACM,* 37(7), July, pp. 72-76.

Etzioni, O. and Weld, D. S., (1995), Intelligent Agents on the Internet: Fact, Fiction, and Forecast, *IEEE Expert,* August, pp. 44-49.

Expedia, http://www.expedia.com.

Farquhar, A., Fikes, R. and Rice, J., (1996), *The Ontolingua Server: A Tool for Collaborative Ontology Construction,* Technical Report KSL-96-26, Stanford Knowledge Systems Laboratory, Stanford, CA.

FIND/SVP, (1996), *The American Internet User Survey: New Survey Highlights,* http://www.findsvp.com.

FIPA (1998), *Specification, Part 12.*

FIPA (1999), *Specification, Part 2.*

FIPA, (1997), *Specification, Part 4.*

Firefly, (1999), http://www.firefly.com.

Fischer, K., Muller, J., Hemig, I. and Sheer, A-W, (1996), Intelligent Agents in Virtual Enterprises, *Proceedings of the First International Conference on the*

Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96), London, UK, 1996, pp. 75-90.

Foner, L. N., (1997), Entertaining Agents: A Sociological Case Study, *Proceedings of the First International Conference on Autonomous Agents (Agents 97),* Marina de Rey, CA, 1997, pp. 122-129.

Foner, L., (1997), *What's an Agent? Crucial Notions,* http://www.media.mit.

Forrester Research, Inc., (1998), *Resizing Online Business Trade,* Forrester Research, November, 1998, http://www.forrester.com.

Forslund, G., (1995), Toward Co-operative Advice-Giving Systems: A Case Study in Knowledge-Based Decision Support, *IEEE Expert,* August, pp. 56-62.

Franklin, S. and Graesser, A, (1996), Is It an Agent, or Just a Program? In Muller, J. P., Wooldridge, M. and Jennings, J. R. (eds.) *Intelligent Agents III* (LNAI Volume 1193), Springer-Verlag, Berlin, Germany, 1997, pp. 21-36.

Gaspari, M., (1995), An Open Framework for Co-operative Problem Solving, *IEEE Expert,* June, pp. 48-55.

Gazdar, G. and Mellish, C., (1989), *Natural Language processing in Prolog: An Introduction to Computational Linguistics*, Wokingham, Addison-Wesley Publishing Company.

Genesereth, M. R. and Files, R. E., (1992), *Knowledge Interchange Format,* Version 3.0, Reference Manual, Computer Science Department, Stanford University.

Genesereth, M. R. and Ketchpel. S. P., (1994), Software Agents, *Communications of the ACM,* 37(7), July, pp. 48-53.

Genesereth, M. R. and Nilsson, N. J., (1987), *Logical Foundations of Artificial Intelligence*, Los Altos, Morgan Kaufmann.

Genesereth, M. R., Keller, A. M. and Duschka, O. M., (1997), Infomaster: An Information Integration System, *Proceedings of the ACM Sigmod International Conference on Management of Data,* ACM Press.

Genesereth, M., Ginsberg, M. and Rosenchein, J., (1986), Co-operation without Communication, *Pròceedings of Annual Association of Artificial Intelligence,* Philadelphia, pp. 51-57.

Georgeff, M. P. and Lansky, A. L., (1987), Reactive Reasoning and Planning, *Proceedings of the Sixth National Conference on Artificial Intelligence, (AAAI '87),* Seattle, WA, pp. 677-682.

Georgeff, M., (1983), Communication and Interaction in Multi-Agent Planning, *Proceedings of the Third National Conference on Artificial Intelligence,* Washington, D.C., Morgan-Kaufmann, San Mateo, California, pp. 125-129.

Georgeff, M., (1984), A Theory of Action for Multi-Agent Planning, *Proceedings of the Fourth National Conference on Artificial Intelligence,* Austin, Texas, San Mateo, California, pp. 121-125.

Gilbert, D, Aparicio, M, Atkinson, B, Brady, S., Ciccarino, J., Grosof, B., O'Connor, P., Osisek, D., Pritko, S., Spagna, R. and Wilson, L., (1995), *The Role of Intelligent Agents in the Information Infrastructure,* IBM White Paper, US.

Gilster, P., (1994), *The Internet Navigator,* Second Edition, New York, John Wiley & Sons, Inc.

Gmytrasiewicz, P. J., Durfee, E. H. and Wehe, D. K., (1991), A Decision-Theoretic Approach to Co-ordinating Multi-agent Interactions, *Proceedings of the 12th International Joint Conference on Artificial Intelligence,* Sydney, Australia, pp. 62-68.

Gosling, J. and McGilton, H., (1995), *The Java Language Environment: A White Paper.* Sun Microsystems, 1995.

Grasser, L. and Briot, J. (1992), Object based concurrent programming in DAI, in Avouris, N. and Grasser, L. (eds.), *Distributed Artificial Intelligence: Theory and Praxis,* Kluwer, pp. 81-107.

Greif, I., (1994), Desktop Agents in Group-Enabled Products, *Communications of the ACM,* 37(7), July, pp. 100-104.

Gruber, T. R., (1993), A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition,* Volume 2, pp. 199-220.

Guha, R. V. and Lenat. D. B., (1994), Enabling Agents to Work Together, *Communications of the ACM,* 37(7), July, pp. 127-142.

Guilfoyle, C., Jeffcoate, J. and Stark, H. (1997), *Agents on the Web: Catalyst for E-commerce,* Ovum, Inc., April, 1997, http://www.ovum.com.

Guttman, R. H., Moukas, A. G. and Maes, P., (1998), Agent-Mediated Electronic Commerce: A Survey, *Knowledge Engineering Review,* 13(3), June 1998.

Hearst, M. A., (1997), Interfaces for Searching the Web, *Scientific American,* March, 1997, http://www.sciam.com.

Hensley, P., Metral, M., Shardanand, U, Converse, D. and Myers, M., (1997), *Proposal for an Open Profiling Standard,* Technical Note, W3C, June, 1997.

Hermans, L. A. and Schlimmer, J. C., (1993), A Machine-Learning Apprentice for the Completion of Repetitive Forms, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 164-170.

Hoffman, D. L., Kalsbeek, W. D. and Novak, T. P., (1996), Internet and Web Use in the United States: Baselines for Commercial Development, *Project 2000 Working Paper,* 10 July, 1996.

Hopken, W., (1999), *Reference Model of an Electronic Tourism Market,* Workshop, The Sixth ENTER 99 International Conference on Information and

Communication Technologies in Tourism, Innsbruck, Austria, 20-23 January, 1999.

Howard, L. M. and D'Angelo, D., (1995), The GA-P: A Genetic Algorithm and Genetic Programming Hybrid, *IEEE Expert,* June, pp. 11-15.

Huhns, M and Singh, M. P., (1998), *Readings in Agents,* Morgan Kaufmann Publishers, San Mateo, CA, 1998.

Intermarket Online, (1999), *The Internet Commerce Briefing,* August, 1999, http://www.intermarketgroup.com.

International Air of Transport Association, (1995), *Press Release,* November, 1995, http://www.iata.org.

Internet Travel Network, http://www.itn.net.

Jango, (1996), http://www.jango.com.

JATLite, (1997), http://java.standford.edu.

Jennings, N. (1994), *Co-operation in Industrial Multi-agent Systems,* World Scientific Press.

Jennings, N. R. and Mamdani, E. H., (1992), Using Joint Responsibility to Co-ordinate Collaborative Problem Solving in Dynamic Environments, *Proceedings of the 10th National Conference on Artificial Intelligence,* San Jose, CA, pp. 269-275.

Jennings, N. R., Sycara, K. P. and Wooldridge, M., (1998), A Roadmap of Agent Research and Development, *Journal of Autonomous Agents and Multi-agent Systems,* 1(1), July, 1998, pp. 7-38.

Jupiter Communications, (1996a), *Home Shopping Report,* 1996, http://www.jup.com.

Jupiter Communications, (1996b), *Travel and Interactive Technology Report,* 1996, http://www.jup.com.

Jupiter Communications, (1997), *Consumer Internet Report,* July, 1997, http://www.jup.com.

Jupiter Communications, (1998a), *Online Shopping Report,* 1998, http://www.jup.com.

Jupiter Communications, (1998b), *Online Travel – Five Years Outlook,* 1998, http://www.jup.com.

Jupiter Communications, (1999a), *Travel Suppliers Missing Online Market Potential,* May 17, 1999, http://www.jup.com.

Jupiter Communications, (1999b), *Consumer Market Forecast and Research,* Applied Track, The Sixth ENTER 99 International Conference on Information and Communication Technologies in Tourism, Innsbruck, Austria, 20-23 January, 1999.

Kautz, H. A., Selman, B. and Coen, M., (1994), Bottom-Up Design of Software Agents, *Communications of the ACM,* 37(7), July, pp. 143-146.

Kay, A., (1984) Computer Software, *Scientific American,* 251(3), pp. 53-59.

Kay, A., (1990), User Interface: A Personal View, in Laurel, B., (eds.) *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 191-208.

Kowalski, R., (1995), Using Meta-Logic to Reconcile Reactive with Rational Agents, in Apt, K. and Turini, F, (eds.) *Meta-Logic and Logic Programming,* MIT Press, 1995, pp. 227-242.

Kozierok, R. and Maes, P., (1993), A Learning Interface Agent for Scheduling Meetings, *Proceedings of ACM SIGCHI International Workshop on Intelligent User Interfaces,* ACM Press, N. Y., pp. 81-88.

Kozierok, R., (1993), *A Learning Approach to Knowledge Acquisition for Intelligent Interface Agents,* SM Thesis, Department of Electrical Eng. and Computer Science, MIT, May, 1993.

Krantz, (1997), Keeping Tabs Online, *TIME,* 10 November, 1997, pp. 44-45.

Kraus, S., (1993), Agents Contracting Tasks in Non-Collaborative Environments, *Proceedings of the 11th National Conference on Artificial Intelligence,* AAAI Press, Washington D. C., pp. 243-248.

Laasri, B., Lassri, H., Lander, S. and Lesser, V., (1992), A Generic Model for Intelligent Negotiating Agents, *International Journal of Intelligent and Co-operative Information Systems,* 1(2), pp. 291-317.

Lander, S. E. and Lesser, V. R., (1993), Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents, *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambery, France, pp. 438-444.

Lashkari, Y., Metral, M. and Maes, P., (1994) Collaborative Interface Agents, *Proceedings of the National Conference on Artificial Intelligence,* MIT Press, Cambridge, Mass., pp. 444-449.

Laurel, B., (1990), Interface Agents: Metaphors with Character, in Laurel, B. (ed.), *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 355-365.

Lenat, D. B., (1995), CYC: A Large-Scale Investment in Knowledge Infrastructure, *Communications of the ACM,* 38(11), pp. 33-38.

LowestFare, http://www.lowestfare.com.

Luce, R. and Raiffa, H., (1957), *Games and Decisions*, John Wiley & Sons, NY.

Maes, P. and Kozierok, R., (1993), Learning Interface Agents, *Proceedings of the AAAI'93 Conference,* MIT Press, Cambridge, Mass., pp. 459-465.

Maes, P., (1994), Agents that Reduce Work and Information Overload, *Communications of the ACM*, 37(7), July, pp. 31-40.

Mahling, D. E. and Craven, N., (1995), From Office Automation to Intelligent Workflow Systems, *IEEE Expert,* June, pp. 41-47.

Mansfeld, Y., (1992), From Motivation to Actual Travel, *Annals of Tourism Research,* 19(3), pp. 399-419.

Marcussen, C. H., (1999), *Internet Distribution of European Travel and Tourism Services: The Market, Transportation, Accommodation and Package Tours,* The Research Centre of Bornholm, Denmark, ISBN 87-90881-28-1.

Maulsby, W. R., (1992), The Phenomenal Influence of Media and Movies on Tourism: The Impact of Electronic Media on Consumer Travel Preferences, *Proceedings of the 23rd Annual Conference of Travel and Tourism Research Association,* June 14-17, 1992, Minneapolis, MN, pp. 53-55.

Maybury, M. T., (1992), Intelligent Multimedia Interfaces, *IEEE Expert,* February, pp. 4-11.

Mazur, J., (1994), *Learning and Behaviour*, Prentice Hall.

McCarthy, J. and Hayes, P., (1969), Some Philosophical Problems from the Standpoint of Artificial Intelligence, in Melzer, B. and Michie, D. (eds.), *Machine Intelligence 4,* Edinburgh University Press, 1969, pp. 463-502.

McNulty, M. A., (1999), Suppliers Seek XML Standard, *Business Travel News,* May 3, 1999.

Merz, M., Müller, K., Lamersdorf, W., (1995), Electronic Market Support for the Tourism Industry: Requirements and Architectures, in Schertler, B. et al. (eds.) *Information and Communications Technologies in Tourism, Proceedings of ENTER 95 International Conference in Innsbruck, Austria, 18-20 January, 1995,* Springer-Verlag Wien, New York, pp. 220-229.

Middleton, V. T. C., (1988), *Marketing in Travel and Tourism,* Butterworth-Heinemann, Oxford.

Minsky, M. and Riecken, D., (1994), A Conversation with Marvin Minsky About Agents, *Communications of the ACM,* 37(7), July, pp. 23-29.

Minsky, M., (1963), Steps Towards Artificial Intelligence, in Feigenbaum, E. and Feldman, J. (eds.) *Computers and Thought*, McGraw Hill.

Moukas, A., Guttman, R. and Maes, P. (1998), Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective, *Software Agents Group, MIT Media Laboratory,* Cambridge, MA, http://ecommerce.media.mit.edu.

Moutinho, L., (1987), Consumer Behaviour in Tourism, *European Journal of Marketing,* 21(10), pp. 1-44.

Mtichell, T., Caruana, R, Freitag, D, McDermott, J. and Zabowski, D., (1994), Experience with a Learning Personal Assistant, *Communications of the ACM,* 37(7), July, pp. 80-91.

Murnighan, J., (1991), *The Dynamics of Bargaining Games,* Englewood Cliffs, NJ, Prentice Hall.

National Research Council, (1994), *Realising the Information Future - The Internet and Beyond,* Washington D. C., US, http://www.nap.edu/nap/online/rtif.

Ndumu, D. T., Collis, J. C. and Nwana, H. S., (1998), Towards Desktop Personal Travel Agents, *BT Technology Journal,* 16(3), July 1998, pp. 69-78.

Negroponte, N., (1970) *The Architecture Machine: Towards a More Human Environment,* Cambridge, Mass., MIT Press.

Ng, F. Y. Y. and Sussmann, S., (1995), Need an Expert for Holiday Selection?, *Proceedings of Hospitality Information Technology Association Worldwide Conference,* New Orleans, Louisiana, 24-26 June, 1995.

Ng, F. Y. Y. and Sussmann, S., (1996), A Personal Travel Assistant for Holiday Selection - A Learning Interface Agent Approach, in Klein, S. et al. (eds.) *Information and Communications Technologies in Tourism, ENTER 96 International Conference,* Innsbruck, Austria, 17-19 January, 1996, Springer-Verlag, Wien, New York, pp. 1-10.

Ng, F. Y. Y. and Sussmann, S., (1998), Intelligent Agents in Electronic Travel Markets, *Proceedings of New Zealand Tourism and Hospitality Research Conference, Third Biennial Conference, 'Advances in Research',* Akaroa, New Zealand, 1-4 December, 1998, PART 2, Day Four, Group 2.

Ng, F. Y. Y., (1995a), Business Travellers' Expert Advisor, in Schertler, B. et al. (eds.) *Information and Communications Technologies in Tourism, Proceedings of ENTER 95 International Conference in Innsbruck, Austria, 18-20 January, 1995,* Springer-Verlag Wien, New York, pp. 288-298.

Ng, F. Y. Y., (1995b), Intelligent Agents as Personal Travel Assistants, in Nuryanti, W. (ed.) *Tourism and Culture: Global Civilisation in Change, 1995 Indonesian-Swiss Forum on Culture and International Tourism,* Yogyakarta, Indonesia, 23-26 August, 1995, pp. 360-371.

Ng, F. Y. Y., (1995c), The Expert Travel Counselling System: A Case Study, *Proceedings of the Fourth Annual CHME Research Conference (Volume 3),* The Hotel School City College, Norwich, 19-20 April, 1995.

Nodine, M. and Unruh, A., (1997), Facilitating Open Communication in Agent Systems: The Infosleuth Infrastructure, in Singh, M, Rao, A and Wooldridge, M. (eds.), *Proceedings of the 4th International Workshop on Agent Theories, Architectures and Languages,* Springer-Verlag.

Norman, D. A., (1994), How Might People Interact with Agents, *Communications of the ACM,* 37(7), July, pp. 68-71.

NUA, Inc., (1997-1999), *Internet Surveys,* http://www.nua.ie.

Nwana, H. S. and Ndumu, D. T., (1999), A Perspective on Software Agents Research, *Knowledge Engineering Review,* 1999, (To appear).

Nwana, H. S., (1998), Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints, *Proceedings of Agents '98,* Minneapolis, May 1998, pp. 189-196.

Nwana, H. S., Ndumu, D. T., Lee, L. C. and Collis, J. C., (1999), ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems, *Applied Artificial Intelligence Journal,* 13(1), 1999, pp. 129-186.

Nwana, H., (1996), Software agents: An Overview, *Knowledge and Engineering Review,* 11(3), November 1996.

Nwana, H., Lyndon Lee, L. and and Jennings, N., (1996), Co-ordination in Software Agent Systems, *BT Technology Journal*, 14(4), 1996, pp. 79-88.

O'Hare, G. and Jennings, N., (1996), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, NY.

Open Travel Alliance, (1999), http://www.opentravel.com.

O'Reilly/Trish, (1996), *Defining the Internet Opportunity,* O'Reilly & Associates/Trish Information Services, http://www.ora.com.

Ovum, (1994), *Intelligent Agents: The Next Revolution in Software,* http://www.ovum.com.

Patil, R., Fikes, R., Patel-Schneider, P., McKay, D., Finin, T., Gruber, T. and Neches, R. (1992), The ARPA Knowledge Sharing Effort: Progress report. In Nebel, B., Rich, C. and Swartout, W. (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92),* San Mateo, CA, November 1992, Morgan Kaufmann.

Pegasus Systems, Inc., http://www.pegsinc.com.

Peng, Y., (1998), A Multi-Agent System for Enterprise Integration, *Journal of Applied Artificial Intelligence,* 1(1).

PersonaLogic, (1999), http://www.personalogic.com.

PhoCusWright (1999), Internet Hotel Sales to Reach Almost US$4 Billion by 2001, *Hotel And Hospitality: The Sleeping Giant Of Internet Travel?*, 28 April, 1999.

PhocusWright, (1998a), *Insights, Analysis & Commentary for the Online Internet Travel Marketplace,* 1(1), September, 1998.

PhoCusWright, (1998b), *Travel E-commerce Survey,* November, 1998, http://www.phocuswright.com.

Plain, S. W., (1997), KQML at Your Service, *Computer Shopper,* March, 1997.

Porto, V. W., (1995), Alternative Neural Network Training Methods, *IEEE Expert,* June, pp. 16-22.

Preview Travel, http://www.previewtravel.com.

Priceline.com, http://www.priceline.com.

Raitz, K. and Dakhil, (1989), A Note about Information Sources for Preferred Recreational Environments, *Journal of Travel Research,* 27(4), pp. 45-49.

Ralston, L. S., (1993), The Association between the Need for Affiliation and Traveller Type with the Motivation for Travel, *Visions Leisure Business,* 12(3), pp. 24-41.

Ram, S., (1995), A Blackboard-Based Co-operative System for Schema Integration, *IEEE Expert,* June, pp. 56-62.

Rao, A. and Georgeff, M., (1995), BDI Agents: From Theory to Practice, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS '95),* San Francisco, USA, pp. 312-319.

Rasmusson, A. and Janson, S., (1996), Personal security assistance for secure Internet commerce, *Proceedings of the New Security Paradigms '96,* ACM Press, September 1996.

Riecken, D., (1994a), Intelligent Agents, *Communications of the ACM,* 37(7), July, pp. 18-21.

Riecken, D., (1994b), M: An Architecture of Integrated Agents, *Communications of the ACM,* 37(7), July, pp. 106-116.

Robie, C., Bateson, A. G., Ellison, P. A. and Figler, M. H., (1993), An Analysis of the Tourism Motivation Construct, *Annals of Tourism Research,* 20(4), pp. 773-776.

Rosen, C, (1999), *Online Sales Heat Market,* http://www.btnonline.com.

Rosenschein, J. and Genesereth, M., (1985), Deals among Rational Agents, *Proceedings of 9th Artificial Intelligence Conference: Intelligent Agents,* Los Angeles, pp. 91-99.

Rosenschein, J. S. and Zlotkin, G., (1994), *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers,* MIT Press, Boston, MA.

Russell, S. J. and Norvig, P., (1995), *Artificial Intelligence: A Modern Approach,* Englewood Cliffs, NJ: Prentice Hall.

Sabre, (1999), *Marketing the Consumer in the Digital Age – New Approaches, New Opportunities,* Applied Track, The Sixth ENTER 99 International Conference on Information and Communication Technologies in Tourism, Innsbruck, Austria, 20-23 January, 1999.

Saffo, P., (1996), Published Interview, *IBM Networking Press Room,* http://www.networking.ibm.com.

Saravanan, N., (1995), Evolving Neural Control Systems, *IEEE Expert,* June, pp. 23-27.

Schmid, B., (1993), Electronic Markets, *Electronic Markets - Newsletter of the Competence Centre,* St. Gallen, Switzerland, 3(9/10), October, 1993.

Schneiderman, B., (1983), Direct Manipulation: A Step Beyond Programming Languages, *IEEE Computer,* 16(8), pp. 57-69.

Searle, J., (1969), *Speech Acts*, Cambridge, MA, Cambridge University Press.

Seel, N. R., (1989), *Agent Theories and Architectures,* Unpublished PhD Thesis, Department of Electrical and Electronic Engineering, University of Surrey.

Selker, T., (1994), Coach: A Teaching Agent that Learns, *Communications of the ACM,* 37(7), July, pp. 92-99.

Sheth, B. and Maes, P., (1993) Evolving Agents for Personalised Information Filtering, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 345-351.

Sheth, B., (1994), *A Learning Approach to Personalised Information Filtering,* Unpublished SM Thesis, Department of Electrical Eng. and Computer Science, MIT, Feb., 1994.

Shoham, Y., (1993), Agent-oriented Programming, *Artificial Intelligence*, 60(1), pp. 51-92.

Skarmeas, N. and Clark, K., (1996), Process Oriented Programming for Agent-Based Network Management, *Proceedings of the Intelligent Agents for Telecoms Applications, ECAI '96 Workshop.*

Smith, D. C., Cypher, A. and Spohrer, J., (1994), KidSim: Programming Agents Without a Programming Language, *Communications of the ACM,* 37(7), July, pp. 55-67.

Smith, R. and Davis, R., (1981), Frameworks for Co-operation in Distributed Problem Solving, *IEEE Transactions on Systems, MAN and Cybernetics,* SMC-11, No. 1, January, 1981.

Soltysiak S. and Crabtree, B., (1998), Knowing Me, Knowing You: Practical Issues in the Personalisation of Agent Technology, *Proceedings of PAAM '98,* London, March 1998, pp. 467-484.

Soltysiak, S. and Crabtree, B., (1998), Automatic Learning of User Profiles - Towards the Personalisation of Agent Services, *BT Technology Journal,* 16(3), July 1998, pp. 110-117.

Stanfill, C. and Waltz, D., (1986), Towards Memory-Based Reasoning, *Communications of the ACM,* 29(12), December, pp. 13-28.

Sun Microsystems, (1995), *Java Tutorial,* http://java.sun.com.

Sussmann, S. and Ng, F. Y. Y., (1995), Business Travel Counseling, *Annuals of Tourism Research,* 22(3), pp. 688-690.

Sussmann, S. and Ng, F. Y. Y., (1995), The Expert Travel Counselling System - The Next Stage, *Progress in Tourism and Hospitality Research, 1(1),* pp. 43-52.

Sycara, K. P., Decker, K., Pannu, A., Williamson, M. and Zeng, D., (1996), Distributed Intelligent Agents, *IEEE Expert,* 11(6).

Sycara, K., (1989), Multi-Agent Compromise via Negotiation, in Gasser, L. and Huhns, M. (eds.), *Distributed Artificial Intelligence 2*, Morgan Kaufmann.

Tabican, (1997), http://www.tabican.ne.jp.

Tackett, W. A., (1995), Mining the Genetic Program, *IEEE Expert,* June, pp. 28-38.

Talbott, S. L., (1995), *The Future Does Not Compute - Transcending the Machines in Our Midst,* Sebastopol, CA, O'Reilly & Associates, 1995.

Tamburino, L. A. and Zmuda, M. A., (1995), Generating Pattern-Recognition Systems Using Evolutionary Learning, *IEEE Expert,* August, pp. 63-68.

Thomas, S. (1994), The PLACA Agent Programming Language, in Wooldridge, M. and Jennings, N. (eds.) *Proceedings of ECAI Workshop on Agent Theories, Architectures and Languages,* Amsterdam, The Netherlands, pp. 307-320.

Thompson, G., Frances, J., Levacic, R. and Mitchell, J., (1991), *Market, Hierarchies and Networks - the Co-ordination of Social Life*, Sage Publications.

Travel Industry Association of America, (1998), *Travel and Interactive Technology Report,* February, 1998, http://www.tia.org.

Travel Industry Association of America, (1999), *1998 Technology and Travel Survey,* January 1999, http://www.tia.org.

Travelfacts, http://www.travelfacts.com.

Travelocity, http://www.travelocity.com.

TravelWeb, http://www.travelweb.com.

Trip.com, http://www.trip.com.

Tschanz, N. and Zimmermann, H., (1996), The Electronic Mall Bodensee as Platform for the Development of Travel Services, in Klein, S. et al. (eds.) *Information and Communications Technologies in Tourism, Proceedings of ENTER 96 International Conference in Innsbruck, Austria, 17-19 January, 1996,* Springer-Verlag Wien, New York, pp. 200-210.

Tsvetovatyy, B. and Gini, M., (1996), Toward a Virtual Marketplace, *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96),* London, UK, 1996, pp. 75-90.

Tyler, S. W. et al., (1991), An Intelligent Interface Architecture for Adaptive Interaction, in Sullivan, J. and Tyler, S. (eds.), *Intelligent User Interfaces,* ACM Press, N. Y., pp. 85-109.

Um, S. and Crompton, J. L., (1992), The Roles of Perceived Inhibitors and Facilitators in Pleasure Travel Destination Decisions, *Journal of Travel Research,* 30(3), pp. 18-25.

Volksen, G., Dieterich, H. and Steiner, D., (1997), Intelligent Agents for Personal Travel Assistance, *Proceedings of ITS Congress ' 97,* Berlin.

Voyager, http://www.objectspace.com.

W3C, (1997), *Extensible Markup Language (XML),* World Wide Web Consortium Working Draft, 17 November, 1997.

Watkins, C., (1989), *Learning from Delayed Rewards,* PhD Thesis, University of Cambridge, England.

Wayner, P., (1994), Agents Away, *Byte*, May, 1994, pp. 113-118.

Weiser, M., (1991), The Computer for the 21st Century, Ubiquitous Computing Paper, *Scientific American,* September, 1991.

Werner, (1996), Co-operating Agents: A Unified Theory of Communication and Social Structure, in Gasser and Huhns, (eds.), *Distributed Artificial Intelligence,* Volume 2, Morgan Kaufmann.

White, J. E., (1995), *Telescript Technology: The Foundation for the Electronic Marketplace,* White paper, General Magic Inc.

Wiener, N., (1948), *Cybernetics,* Wiley & Sons, NY.

Woodside, A. G. and Carr, J. A., (1988), Consumer Decision-Making and Competitive Marketing Strategies: Application for Tourism Planning, *Journal of Travel Research,* 26(3), pp. 2-7.

Wooldridge, M. and Jennings, N. R., (1995), Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review,* 10(2), 1995, pp. 115-152.

WorldRes, http://www.worldres.com.

World Tourism Organisation, (1999), *Marketing Tourism Destinations Online: Strategies for the Information Age,* ISBN: 92-844-0328-6.

Zlotkin, G. and Rosenschein, J. S., (1989), Negotiation and Task Sharing Among Autonomous Agents in Co-operative Domains, *Proceedings of the 11th International Joint Conference on Artificial Intelligence,* Morgan Kaufmann, August, 1989, pp. 912-917.

Zlotkin, G. and Rosenschein, J. S., (1991), Incomplete Information and Deception in Multi-agent Negotiation, *Proceedings of the IJCAI 91 International Joint Conference on Artificial Intelligence,* Australia, pp. 225-231.

Zlotkin, G. and Rosenschein, J. S., (1993), A Domain Theory for Task Oriented Negotiation, *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambery, France, pp. 416-422.

Published Papers and Articles

1. Ng, F. Y. Y. and Sussmann, S., (1998), Intelligent Agents in Electronic Travel Markets, *Proceedings of New Zealand Tourism and Hospitality Research Conference, Third Biennial Conference, 'Advances in Research',* Akaroa, New Zealand, 1-4 December, 1998, PART 2, Day Four, Group 2.

2. Ng, F. Y. Y. and Sussmann, S., (1996), A Personal Travel Assistant for Holiday Selection - A Learning Interface Agent Approach, in Klein, S. et al. (eds.) *Information and Communications Technologies in Tourism, ENTER 96 International Conference,* Innsbruck, Austria, 17-19 January, 1996, Springer-Verlag, Wien, New York, pp. 1-10.

3. Ng, F. Y. Y., (1995), Intelligent Agents as Personal Travel Assistants, in Nuryanti, W. (ed.) *Tourism and Culture: Global Civilisation in Change, 1995 Indonesian-Swiss Forum on Culture and International Tourism,* Yogyakarta, Indonesia, 23-26 August, 1995, pp. 360-371.

4. Ng, F. Y. Y. and Sussmann, S., (1995), Need an Expert for Holiday Selection?, *Proceedings of Hospitality Information Technology Association Worldwide Conference,* New Orleans, Louisiana, 24-26 June, 1995.

5. Ng, F. Y. Y., (1995), The Expert Travel Counselling System: A Case Study, *Proceedings of the Fourth Annual CHME Research Conference (Volume 3),* The Hotel School City College, Norwich, 19-20 April, 1995.

6. Sussmann, S. and Ng, F. Y. Y., (1995), Business Travel Counseling, *Annals of Tourism Research,* 22(3), pp. 688-690.

7. Sussmann, S. and Ng, F. Y. Y., (1995), The Expert Travel Counselling System - The Next Stage, *Progress in Tourism and Hospitality Research, 1(1),* pp. 43-52.

8. Ng, F. Y. Y., (1995), Business Travellers' Expert Advisor, in Schertler, B. et al. (eds.) *Information and Communications Technologies in Tourism, Proceedings of ENTER 95 International Conference in Innsbruck, Austria, 18-20 January, 1995,* Springer-Verlag Wien, New York, pp. 288-298.

INTELLIGENT AGENTS IN ELECTRONIC TRAVEL MARKETS

Faria Ng & Silvia Sussmann
School of Management Studies for the Service Sector
University of Surrey
Guildford, Surrey, GU2 5XH
United Kingdom

ABSTRACT

Intelligent agents in electronic commerce is an area that currently gains increasing attention. Travel, as an information-intensive industry, may well be the most fertile ground to apply this technology. This paper starts with an analysis of the current situation of the online travel market. It then identifies the problems facing the Internet users and what they need in terms of software technologies. To address these deficiencies, it proposes a multi-agent system with an intention that information demand and supply can be brought together more efficiently in the online travel market. A prototype is currently under construction to demonstrate the technical and commercial viabilities of the multi-agent architecture.

1 INTRODUCTION

The introduction of new media of information and communication technologies, like the Internet and the World Wide Web, creates the opportunity for the formation of the 'Electronic Travel Market' as a new form of distribution channel. The Internet aims to fulfil the role of enhancing product information dissemination and eventually facilitating online market transactions. However, the current structure has led to disorientation of information consumers as well as suppliers.

2 THE ELECTRONIC TRAVEL MARKET OF TODAY

To many web travellers, the Internet - today's closest approximation of the electronic marketplace - is a valuable source of all sorts of travel information. Never before has an information source been available through which such massive amounts of information - and in such a broad range - can be gathered. And this information can be obtained conveniently and at very low costs, which too is something that seems unprecedented. The same story applies to travel information suppliers. The barriers, as well as the investments, needed to advertise online are relatively low.

Though this has been - and it still is - one of the biggest advantages of using the Internet to retrieve or disseminate travel information, it also has an important downside to it.

There is very little supervision on the ways in which information is offered, i.e., there are no rules stating in which format information should be offered.

Many travellers find that the amounts are becoming more and more overwhelming and beyond their control. A few years ago, finding travel information on the Internet was by casually 'surfing' the Net and the task was not that difficult. However, as the amount of travel information available began to grow at exponential rates, and the number of travel web sites began to increase as well, new ways of finding information were desperately needed. Using search engines such as Excite and Altavista (or directory services such as Yahoo) does not offer any help. When the result of querying a search engine is a list of thousands of links ('hits'), it seems like we are back to square one.

Suppliers of travel information, as opposed to travellers, have problems of their own. One of the biggest problems facing suppliers is how they can get their information to the target audience. With the growing number of information sites, travel suppliers begin to look into issues like how to stand out from the rest, and how to make sure the *right* people know about their locations. Submitting information (an 'advertisement') to search engines is a method that is getting less and less effective.

New means of both **offering** as well as **retrieving** information are needed. The whole process of information exchange through the electronic travel market will be enhanced and catalyzed by various new enabling technologies, including Push Technologies[1], Intelligent Agents, electronic intermediaries[2], etc. What is sure is that, no matter how the online travel market evolves, the process will be more personal as to fit the personal needs and preferences of each individual, be it traveller or supplier. Intelligent agents will certainly play an important role, though they will not be the only actors.

[1] Push Technology is the basic technique of server push. The idea is to push information to the user about topics he or she is interested in, whenever it suits him or her best, and preferably by using the medium and the format which is most appropriate and most convenient at that moment.

[2] An electronic intermediary is the outcome of the idea of information brokering. When it receives an information query from a consumer, it determines to which suppliers it should, could (perhaps), or should *not* send this query to. The intermediary will not store any of the actual information as it is offered by suppliers. In case it fails to obtain an answer (advertisement) to the query, it can delegate this task to a third party (e.g. specialised agents). After it has send out the query to the appropriate sources, the intermediary will collect the results of each individual source. Before sending these results to the consumer, it will enhance the results, e.g.,ranking them, sorting out double entries, etc.

3 THE REBIRTH OF INTELLIGENT AGENTS

The time never seems to be better to (re-)introduce a familiar concept to move things to the next stage of evolution, the concept being that of 'intelligent software agents'.

3.1 Defining Intelligent Agents

The term 'agent' has its background in the early work on AI when researchers concentrated on trying to create artificial entities that mimicked human abilities (Kay, 1984; Negroponte, 1970; Seel, 1989). In its strictest sense, the term can be applied to a wide range of entities including software systems that have become synonymous with the term as well as autonomous robots and biological organisms. In essence, therefore, an 'intelligent agent' is a computational entity that:

- acts on behalf of other entities in an autonomous fashion;
- performs its actions with some level of proactivity and/or reactiveness; and
- exhibits some level of the key attributes of learning, co-operation and mobility.

3.2 Applying Intelligent Agents

Many agree that agent - or at least agent-like - applications will become a necessity to be able to cope with the enormous amounts of information that is available through the Internet; the question does no longer seem to be <u>if</u> there will be a considerable usage of agents, but rather *when* this will happen.

> *'You can imagine thinking of an intelligent landscape inhabited not only by humans but by smartifacts - artifacts that are intelligent enough to have some degree of autonomy. [It] will be decades and decades before we have agents or devices intelligent enough to make people nervous. But we already have devices today that are sufficiently autonomous that they do things for us that are practical. (Saffo, 1996)*

Already, more and more 'agent' applications are launched onto the market and the term 'software agents' is used in wild abundance. It seems like there is no problem too big that cannot (or will not soon) be solved by intelligent software agents. What many of the organisations using the term agents seem to forget or overlook, is that agents are not the end products to accomplish the miraculous. They are a design model for applications, a tool to achieve the goal. Bearing this in mind, using intelligent agents to mediate in todays' chaotic online travel market is a very suitable application.

4 AGENTS' CAPABILITIES

'Technical innovation - the devising of new tools - is surely a desirable activity. But unless there is a balance between our fascination with tools and our concern for the ends they may help us achieve, the tool becomes tyrannical. What stares us in the face today is the startling fact that, not only has the balance been upset, but one of its terms has virtually disappeared. Technological innovation now proceeds for its own sake, driven by its own logic, without reference to human need. We are a society obsessed with new tools, but incapable of asking in an serious way, "What are we developing these tools for?"' (Talbott, 1995)

Our pressing need is not for more information, or faster access to information, or more connectivity, but rather an efficient way to make sense of the massive amount of unstructured, randomly distributed information in the online market. Intelligent agents will enable people to focus primarily on <u>what</u> it is they want to do (e.g., which information they need, which task they would like to get done), and much less on <u>how</u> they should best do this (e.g., where to look for information or where to offer it) and which applications, services and techniques should be best used to accomplish this. This focus shift is not only necessary because it saves time and makes life a lot easier, but also because it may be expected that many of the newcomers on the future online travel market place are non-technical by nature. If this marketplace is to be open and ready for everyone, it should not have high technological barriers to entry. Agents will contribute to making the focus shift from people complying with the technique to a situation where the technique is complying with the people.

4.1 Reducing User Involvement

Accessing online travel information requires *time* (to access each server independently), and *perseverance* (each server presents information in a different form, through different search mechanisms, and with different levels of detail). For example, there is no simple way for a traveller to access the fare information in an efficient way. When accessing the CRS data via gateway systems, the online systems generally cannot quote the fare until after the airline, date, and exact flight times and airports have been selected by the user. It takes an average of 15-20 minutes of hard hunting to get out data about all the fares between two points.

Individual and business travellers cannot afford to allocate time and energy away from their more important daily activities to navigate the massive networks and search for relevant information. Drifting through site after site in search of the key information is arduous and boring, and there is nothing inherently creative in the process.

Intelligent agents have the abilities to learn continuously from their users and act on their behalf in a competent manner. Users can delegate high-level tasks without direct manipulation. They act as the representative of the users' goals in the haphazard online travel market environment. The idea is to free the travellers to do more productive jobs - things they specialise at - rather than any tedious, unpleasant or time-consuming tasks like connecting to and searching the Web.

4.2 Providing Intelligence Support

Planning a multifaceted multi-component travel itinerary at a right price requires knowledge, including basic computer skills, knowing where the servers are located, product specifications, etc. From an information perspective, there is a limit to the number of information resources and the browsing behaviour a human being can deal with. As agents are trainable, they can learn continuously from their users. Apart from knowing where to go to search for information, more importantly, they pick up habits and preferences that may serve as bases for selecting and prioritising criteria for product comparison and evaluation. Their learning abilities also helps in cloning users' habits and preferences, enabling them to offer intelligence support with a personal touch that will benefit users in decision-making. Agents make it possible for the travellers to get good, sound, and creative advice with the disguise of a user-friendly interface.

4.3 Performing with Incredible Speed

Another undeniable reason to use agents is that they can perform at a speed which is beyond the capability of human beings. Due to the large number of heterogeneous online travel systems, the decentralised and open nature of the Internet, and the growing number of Internet users, travellers find it increasingly difficult to efficiently locate what they are looking for. Increasing response time and high network complexity result in serious disadvantages for the online bookers. Access to information via centralised index can be both cumbersome and slow. The situation is worse for travel sites embedding large graphics and images that take a long time to be downloaded on the screen.

4.4 Possessing Mobility

A mobile agent is a software entity that exists in a software environment. It can migrate from machine to machine in a heterogeneous network (White, 1995). Mobile agents consume fewer network resources since they move the computation to the data rather than the data to the computation. When vast volumes of data are stored at remote travel sites, the processing of this data can be performed local to the data, instead of transmitting it over the network. Tasks can be encoded into mobile agents and then dispatched. The online travel market is inherently a heterogeneous environment in nature, mobile agent systems can support transparent operation as they are generally computer and network independent.

5 THE ELECTRONIC TRAVEL MARKET FOR TOMORROW

A multi-agent architecture is proposed for the future electronic travel market. A prototype is being built to demonstrate the technical and commercial viabilities of the multi-agent system. Much of the work is based on existing technologies so as to help us envisage the evolution of the travel market.

5.1 Multi-Agent Systems

A multi-agent system can be defined as 'a loosely coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities' (O'Hare and Jennings, 1996).

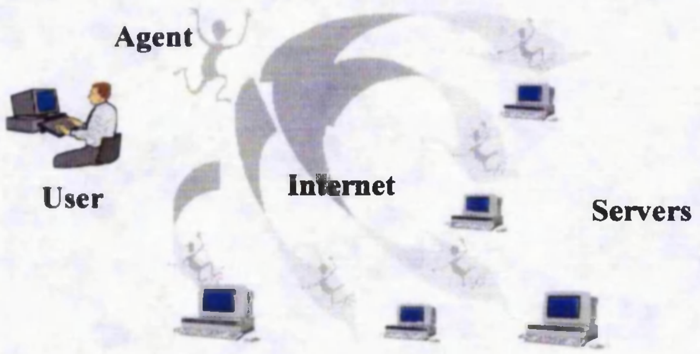


Figure 1 The Multi-agent Architecture

Figure 1 shows a client-server architectural design for the multi-agent approach. On one end is the user/traveller, who demands information and service via the Internet. On the other end are various servers that belong to travel suppliers or intermediates. Suppliers include tour operators, airlines, cruiselines or even medium sized hotels. Intermediaries include online reservation systems, travel agencies, or completely new forms as explained in footnote 2. The user agent and the server agents communicate, co-operate and negotiate to facilitate the information exchange between the two ends. The user agent itself could be a **multi-agent**, with each sub-agent responsible for a specific task. It **interacts** with the users, interprets user needs, and initiates enquiries to the server agents. The server agents interpret enquires, collate and select materials with relevance to the initial information request, etc. All these agents could be **mobile**, e.g., the user agent can transport itself to large quantities of data sources and process information at the servers' location. A server agent can also travel to another server to collect relevant materials.

The multi-agent system design was chosen due to its ability of:

- personalising client-server interface by adapting dynamically to user needs;
- dealing with vast volumes of data more efficiently;
- collecting information from diverse information sources where the expertise is distributed;

- reducing network traffic;
- supporting heterogeneous environments; and
- offering conceptual clarity and simplicity of design.

5.2 Design Considerations

User modelling

One principle consideration in constructing the user agent is how to gather accurate information regarding the user's interests, goals and general preferences. Machine Learning techniques such as Reinforcement Learning[3] (Stanfill and Waltz, 1986), Learning By Observations[4] (Kozierok and Maes, 1993), Instructional Learning[5] (Cypher, 1991), etc. might be useful in achieving this task.

Co-operation, Co-ordination, Negotiation

The problem solving performed by agents in a multi-agent system is termed Distributed Problem Solving and involves research in the areas of co-ordination, negotiation and communication (Durfee and Montgomery, 1990; Jennings and Mamdani, 1992; Lander and Lesser, 1993; Zlotkin and Rosenschein, 1989). In order for a multi-agent system to solve common problems coherently, the agents must communicate amongst themselves, co-ordinate their activities and negotiate once they find themselves in conflict. Conflicts can result from simple limited resource contention to more complex issue-based computations where the agents disagree because of discrepancies between their domains of expertise. Co-ordination is required to determine organisational structure amongst a group of agents and for task and resource allocation, while negotiation is required for the detection and resolution of conflicts.

[3] One fairly recent example of this type of approach to learning is Watkin's Q Learning algorithm (Watkins, 1989). Q Learning works by calculating an estimate for state-action pairs *Q(s,a)*, which are defined to be the expected discounted sum of taking action a in state s and pursuing an optimal policy from there on in. Once these values are learnt, the correct course of action can be determined at any state by taking the action with the highest *Q(s,a)* value.

[4] The program is expected to learn about the task simply by watching and copying users as they display their usual behaviour (Mazur, 1994).

[5] Users may employ simple techniques, like Programming By Examples, to demonstrate to an agent how to perform a particular task without explicit communication.

Agents will ultimately need a general and expressive language to communicate such as KQML[6]. A variety of protocols will be needed to accomplish the complex tasks.

Security

A mobile agent system is an open system (Chess et al., 1995). Therefore, just like in any open system, the host nodes are subject to a variety of attacks, both old and new. Attacks on host security fall into four main categories:

- *Leakage*: acquisition of data by an unauthorised party
- *Tampering*: alteration of data by an unauthorised party
- *Resource stealing*: use of facilities by an unauthorised party
- *Vandalism*: malicious interference with a host's data or facilities with no clear profit to the perpetrator

The traditional methods of attack include eavesdropping, masquerading, message tampering, message replay and viruses. A mobile agent can employ any of these methods of attack, which in turn, can be guarded against using standard techniques such as cryptography, authentication, digital signatures and trust hierarchies.

5.3 The Software Architecture

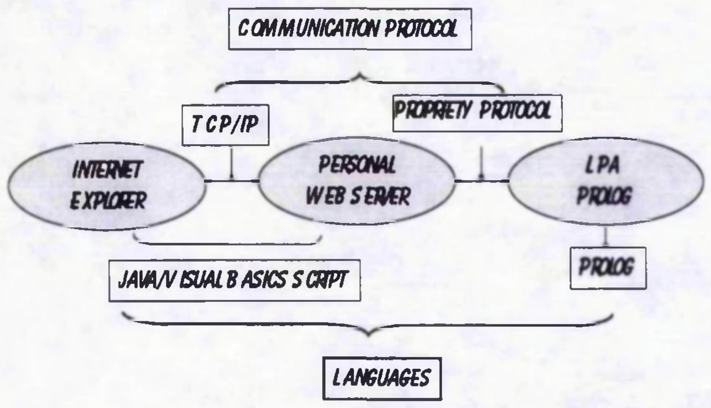


Figure 2 The User Agent Software Architecture

The user agent prototype consists of three major software components:

[6] KQML (Knowledge Query Manipulation Language) is an Agent Communication Language (ACL) produced by the ARPA Knowledge Sharing Effort (Patil et al., 1992). This language provides a message format and message handling protocol supporting run-time knowledge sharing and interaction among agents.

- MS Internet Explorer is used for the multi-media user interface;
- MS Personal Web Server is used for the execution of programming languages, such as Java script;
- LPA Prolog environment is used for logic programming.

Though a client-server software approach is used, the user agent is most likely to run on a single computer.

Prolog, the language for logic programming, has advantages in expressing reasoning and deduction (Dodd, 1990; Gazdar and Mellish, 1989; Sterling and Shapiro, 1986). Java, the native language to the browser and server, is most convenient in driving the user interface and other imperative procedures. Java scripts and Visual Basics scripts are also used to save programming effort in many cases. TCP/IP[7] is the communication protocol between the browser and the server. The prolog environment comes into play via a custom designed protocol.

The architecture for supplier agents is similar. Though no interactive component is needed, the browser is retained for monitoring and management purposes.

6 CONCLUSION

The foundation for the electronic travel market is well-cultivated, multi-agent technologies will act as the catalyst for its expansion. The research in architecture and software will help us to predict and define the evolution of the future electronic travel market. Agents are perceived to be able to bring along more freedom, personal choice and individuality. Just as machines extend our control of the physical world, agents will give us greater control over the world of information, software and networks - the world of the 21st century.

REFERENCES

Altavista, http://www.altavista.digital.com.

Chess, D., Harrison, C. and Kershenbaum, A., (1995), Mobile Agents: Are they a good idea, *Technical Report,* March 1995, IBM T.J. Watson Research Center, NY.

[7] Transmission Control Protocol/Internet Protocol (TCP/IP) is the set of protocols that drives the Internet, regulating how data is transferred between computers.

Cypher, A., (1991), EAGER: Programming Repetitive Tasks by Example, *Proceedings of CHI 91,* ACM Press, N. Y., pp. 33-39.

Dodd, T., (1990), *Prolog: A logical Approach,* New York, Oxford University Press.

Durfee, E. H. and Montgomery, T. A., (1990), A Hierarchical Protocol for Co-ordinating Multi-agent Behaviours, *Proceedings of the 8th National Conference of Artificial Intelligence,* Boston, Mass., pp. 86-93.

Excite, http://www.excite.com.

Gazdar, G. and Mellish, C., (1989), *Natural Language processing in Prolog: An Introduction to Computational Linguistics*, Wokingham, Addison-Wesley Publishing Company.

Jennings, N. R. and Mamdani, E. H., (1992), Using Joint Responsibility to Co-ordinate Collaborative Problem Solving in Dynamic Environments, *Proceedings of the 10th National Conference on Artificial Intelligence,* San Jose, CA, pp. 269-275.

Kay, A., (1984), Computer Software, *Scientific American,* 251(3), pp. 53-59.

Kozierok, R. and Maes, P., (1993), A Learning Interface Agent for Scheduling Meetings, *Proceedings of ACM SIGCHI International Workshop on Intelligent User Interfaces,* ACM Press, N. Y., pp. 81-88.

Lander, S. E. and Lesser, V. R., (1993), Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents, *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambery, France, pp. 438-444.

Mazur, J., (1994), *Learning and Behaviour,* Prentice Hall.

Microsoft, Inc., (1997), *Internet Explorer 4.0,* 1997.

Microsoft, Inc., (1997), *Personal Web Server,* 1997.

Negroponte, N., (1970), *The Architecture Machine: Towards a More Human Environment,* Cambridge, Mass., MIT Press.

Ng, F. Y. Y., (1995), Intelligent Agents as Personal Travel Assistants, in Nuryanti, W. (ed.) Tourism and Culture: Global Civilisation in Change?, *Proceedings of 1995 Indonesian-Swiss Forum on Culture and International Tourism,* Yogyakarta, Indonesia, 23-26 August, 1995, pp. 360-371.

O'Hare, G. and Jennings, N., (1996), *Foundations of Distributed Artificial Intelligence,* John Wiley & Sons.

Patil, R., Fikes, R., Patel-Schneider, P., McKay, D., Finin, T., Gruber, T. and Neches, R. (1992), The ARPA Knowledge Sharing Effort: Progress report. In Nebel, B., Rich, C. and Swartout, W. (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92),* San Mateo, CA, November 1992, Morgan Kaufmann.

Saffo, P., (1996), IBM Networking Forum, IBM Press.

Seel, N. R., (1989), *Agent Theories and Architectures,* Unpublished PhD Thesis, Department of Electrical and Electronic Engineering, University of Surrey.

Stanfill, C. and Waltz, D., (1986), Towards Memory-Based Reasoning, *Communications of the ACM,* 29(12), December, pp. 1213-1228.

Sterling, L. and Shapiro, E., (1986), *The Art of Prolog,* Cambridge, Mass, MIT Press.

Talbott, S. L., (1995), The Future Does Not Compute - Transcending the Machines in Our Midst, Sebastopol, CA, O'Reilly & Associates.

Watkins, C., (1989), Learning from Delayed Rewards, *PhD Thesis,* University of Cambridge, England.

White J., (1995), *Telescript Technology: The Foundation of the Electronic Market Place,* General Magic White Paper, 1995.

Yahoo, http://www.yahoo.com.

Zlotkin, G. and Rosenschein, J. S., (1989), Negotiation and Task Sharing Among Autonomous Agents in Co-operative Domains, *Proceedings of the 11th International Joint Conference on Artificial Intelligence,* Morgan Kaufmann, August 1989, pp. 912-917.

A Personal Travel Assistant for Holiday Selection — A Learning Interface Agent Approach

Faria Y. Y. Ng & Silvia Sussmann
Department of Management Studies, University of Surrey
Guildford, Surrey, GU2 5XH, U. K.
e-mail: f.ng@surrey.ac.uk

1 Introduction

An intelligent agent is a computer system that tries to fulfil its goals in a complex, dynamic environment. It is situated in the environment and interacts with the user in an autonomous manner. It operates adaptively and becomes more experienced overtime in achieving its goals. This paper attempts to trace the development of the concept, analyse the appropriate application area in tourism, survey the possible techniques in creating agents, and finally describe a conceptual framework for a learning interface agent for a holiday selection application. The idea is to employ Machine Learning techniques to customise an agent to the traveller's personal selection rules and preferences by observing his/her actions and receiving positive or negative feedback. This approach provides the traveller with the sophisticated control over the gradual delegation of holiday selection tasks to the agent.

2 What are Intelligent Agents?

The scientific study of agent behaviour and design predates Artificial Intelligence, going back to the early days of Cybernetics in the 1940's (Varela, 1979). As an interdisciplinary subject, Cybernetics generated theoretical attempts to define the behaviour and structure of abstract machines which had properties corresponding to biological, cognitive systems (Seel, 1989). From the early 60's, the dominant intellectual force in agent theory was to be found in Artificial Intelligence, and allied fields such as the philosophy of mind (Minsky, 1963), epistemic logics and natural language semantics. The idea of employing agents to delegate computer-based tasks goes back to research by Negroponte (1970) and Kay (1984). Kay traced the development of the concept:

> *'The idea of an agent originated with John McCarthy in the mid-1950's, and the term was coined by Oliver G. Selfridge a few years later, when they were both at the Massachusetts Institute of Technology. They have in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a "soft robot" living and doing its business within the computer's world.'*

Various names are associated with these intelligent agents, such as personal/automated assistants (Mitchell, et al., 1994), over-the-shoulder coaches (Selker, 1994), knowbots/softbots (Etzioni et al., 1994; Kautz et al. 1994), wizards, or sometimes just simply experts.

From a *functional* point of view, agents can be described as *'a class of interactive knowledge-based consultants that directly assimilate new knowledge by observing and analysing the problem-solving steps contributed by their users through their normal use of the system'* (Mitchell et al., 1994).

Maes & Kozierok (1993) defined agents with a *behavioural* approach. *'An interface agent is a semi-intelligent, semi-autonomous system that assists a user in dealing with one or more computer applications. Interface agents typically behave as personal assistants: they have knowledge about the tasks, habits and preferences of their users and use this knowledge to perform actions on their behalf.*

Laurel (1990) gave a *cognitive* account on agents. *'An intelligent agent can be defined as a character, enacted by the computer, who acts on behalf of the user in a virtual (computer-based) environment. Interface agents draw their strengths from the naturalness of the living-organism metaphor in terms of both cognitive accessibility and communication style. Their usefulness can range from managing mundane tasks like scheduling, to handling customised information searches that combine both filtering and the production (or retrieval) of alternative representations, to providing companionship, advice, and help throughout the spectrum of known and yet-to-be-invented interactive contexts.'* By capturing and representing the capabilities of agents in the form of character, the user is able to make accurate inferences about the internal traits, i.e., how the agent is likely to think, decide and act (values, heuristics, etc.) on the basis of some cues from its external traits, e.g., selection of the mode of representation, such as visual, audio, etc. Users can therefore predict what the agents are likely to do in a given situation on the basis of their character, and have full control on the actions.

To sum up, the central idea is that of a personal assistant who is collaborating with the user in the same work environment. Instead of user initiated interactions via commands and/or direct manipulation (Schneiderman, 1983), the user is engaged in a co-operative process in which human and computer agents both initiate communication, monitor events, and perform tasks. This has been referred to as 'indirect management' (Kay, 1990) in the field of autonomous agents. The assistant becomes gradually more confident and effective as it learns the users' interests, habits and preferences.

3 Why do We Need Agents?

3.1 Tourism Market and Trends

According to the 1994 Business Travel Survey, 'Approximately 2.3 million of the 39.8 million business travellers will fly 10 or more times this year on business', and '73 % of business travellers — 67% whom already travel with laptop computers — say they want the convenience of accessing travel schedules online ...,' (1994 Business Travel Survey, Business Travel News). Apart from the growing needs identified by the business travellers, there is also a steady growing trend towards independent travelling (Bennett, 1993; Cockerell, 1991; Hitchins, 1991). Even leisure travellers will now try to book their travel products direct rather than going to the travel agent for pre-arranged holidays. Coupled with this trend is the increasing number of computer-literate customers who will welcome more user-friendly online travel reservation systems.

3.2 Online Travel Reservation Systems

New York Times (26 February 1995) says the Internet connects at least 20 million users to 70,000 computer network world-wide, and both numbers are growing in leaps and bounds. The number of users of the online travel services is sure to increase. With the Internet and gateway systems — CompuServe, Prodigy, GEnie, Delphi, AOL, eWorld — becoming more and more popular, many online self-booking systems, such as easySabre (data from American Airlines' Sabre), Travelshopper (data from Delta, Northwest and TWA's Worldspan), Official Airline Guide (OAG), allow travellers to tap into the same powerful, complex computer systems airlines and travel agents use to book flights, rental cars and hotels. Now, some software providers, including Travelocity (an alliance between Sabre and Worldview Systems Corporation), Internet Travel Network, PCTravel, TraveLOGIX and Personal ExpertWare provide user-friendly real-time connectivity on the World Wide Web. Other online travel information systems include United Connection on CompuServe, and travel agents' and airlines' WWW browser, such as Virgin Atlantic, USAir, British Airways, Canadian Airlines, Northwest Airlines, Southwest Airlines, Cathay Pacific, Lufthansa, etc.

3.3 How Agents can Help?

However, while consumer online reservation services have been around for as long as a decade, they represent less than 5% of all airline tickets sold. Of the estimated 1.4 million subscribers to the largest and oldest service, American Airline owned easySabre, more than 80% are still lookers, not bookers — people who simply browse, or who make an electronic reservation but call an airline or travel agent to double-check fares and issue the ticket (Bly, 1995). Bly suggests, 'Booking flights takes more than cursory skills.' As the online services are aimed at computer-literate travellers going on business trips, 'they require an affinity for alphabet-soup fare codes and a zen-like patience when encountering such on-screen roadblocks as "incorrect response packet" and "could not connect to gateway host".' Access to information can be both cumbersome and slow. The traveller has to have a familiarity with computers, and a certain personality type.

The gap in the industry is clear. There is a huge market, both business and leisure, yearning for systems that are easy to use and without access barriers. Travellers want information to be accurate and easily understood. They want a service that makes a complicated process comfortable. Even Sabre has recognised the need to make easySabre easier, particularly for new computer users (Bly, 1995). It is obvious that online reservation systems are available, and travellers are ready to accept and try a new method of booking their products. What makes online booking unpopular is the difficulty in dealing with the systems and the time required for the booking process.

The construction of personal travel assistants as learning agents — interactive assistants that learn continuously from their users — is one approach that could help take up the routine booking procedure for the traveller and also select the most suitable products, based on the preferences and habits of the individual traveller. Some people argue that expert systems may suffice to solve the holiday selection problem. However, an expert system approach requires a huge amount of work from the knowledge engineer. He/She has to endow a system with a lot of domain-specific background knowledge about its application and about the users, and little of this knowledge or the system's control architecture can be reused when building systems for other applications. A second problem is that the major part of the knowledge base is fixed. It is possibly incorrect, incomplete, become obsolete overtime, and cannot be easily adapted or customised to individual user differences, or to the changing habits of one user. In highly personalised domains such as travel, the knowledge engineer cannot possibly anticipate how to best aid each user in each of their goals. On the contrary, personal learning agents can produce useful knowledge-based assistants with significant reduced manual development and maintenance of the knowledge base. It is possible to design an agent that is attractive to the users, able to capture useful training data, and capable of generalising from such data to learn a customised knowledge base competitive to hand-coded knowledge (Dent et al., 1992).

4 How to Create Agents?

Prior to the construction of intelligent travel assistants, some preliminary questions have to be considered, such as 'What are the qualities of the task that make it a good candidate for an agent-like interface?'; 'What kind of users will want them, and what are the differences among potential user populations?'; 'How might interface agents affect the working styles, expectations, productivity, knowledge, and personal power of those who use them?'.

4.1 Integration of Multi-Disciplines

The basic idea is to develop a system which engages and helps all types of end users. In understanding user needs and preferences, supports may be sought from psychology and behavioural sciences. In terms of design, disciplines such as dramatic theory and practice, literary criticism and storytelling, psychology and communication acts and sciences will be needed to select the appropriate set of traits for a given agent — traits that can form coherent characters, provide useful cues to users, and gives rise to all of the necessary and appropriate actions in a given context.

4.2 Machine Learning Approach

In terms of implementation, the Machine Learning approach can be adopted to minimise the workload of endowing the agent with sufficient knowledge. There are two major criteria to justify the Machine Learning approach:

- The use of the application has to involve a lot of repetitive behaviour; and
- This repetitive behaviour is very different for different users.

If the first condition is not met, an agent will not be able to learn anything because there are no regularities in the user's actions to learn about. If the second criterion is not met, i.e., the repetitive behaviour demonstrated by different users is the same, a knowledge-based approach might prove to yield better results than a learning approach (Maes and Kozierok, 1993). In the case of holiday arrangement, both criteria are met. A traveller will normally have a consistent, but personal pattern in choosing his/her travel products.

There are several ways that the travel assistant can learn:

Memory-Based Learning: The travel assistant learns by continuously 'looking over the shoulder' of the user as the user is making travel bookings. It can monitor the activities of the user, keep track of all of his/her booking records over long periods of time, find recurrent patterns and offer to automate these. The technique that the agent uses is *memory-based learning* (Stanfill and Waltz, 1986). The agent keeps a memory of everything the user does, stored as situation-action pairs, e.g., long-haul travel with morning flights, which are simply raw data about what happened. Situations are described as a set of attributes. When a new situation occurs, the agent will try to predict the action of the user, based on the situation-action pairs stored in its memory. The agent compares the new situation with the memorised situation-action pairs and tries to find a set of close matches. The most similar of these memorised situations contribute to the decision of which action to take or suggest in the current situation.

Reinforcement Learning: The travel assistant can learn from direct and indirect feedback from the user. The technique used is *reinforcement learning.* Positive feedback occurs when the booking arrangement made by the agent is accepted by the user. Negative feedback happens when the user ignores (indirect) or rejects (direct) the recommendations of the agent and takes a different action instead. The agent learns from negative feedback by adding the right recommendation for the situation as a new example in its memory, or adjusting the weightings of the attributes of the situation. This technique helps ensure that a similar mistake is less likely to occur in the future bookings.

Programming By Examples: If the user does not want to wait for the travel assistant to pick up a certain pattern, it is possible for the user to instruct the agent explicitly by creating a hypothetical booking session and show the agent what should be done. The technique used is *Programming by Examples* (Cypher, 1991). This involves adding the example to the agent's memory, including 'wildcards' for the attributes not specified in the hypothetical example. Smith et al. (1994) called this 'programming a software system in its own user interface'. The agent records the actions, tracks relationships among objects and changes the existing memory to incorporate the example that it is shown.

Mutli-Agent Collaboration: As travelling is not a frequent activity, a longer period will be needed to feed the travel assistant with sufficient information to postulate the preferences of the user and make accurate bookings. Even if the user uses hypothetical examples to train the agent, the agent's competence is necessarily restricted to situations similar to those it has encountered in the past. Collaboration between travel agents assisting different users can alleviate this problem. When faced with an unfamiliar situation, an agent consults its peers who may have the necessary experience to help it. Experienced agents can help a new agent to come up to speed quickly as well as help agents in unfamiliar situations. Such kind of collaboration allows agents of different users to cooperate to best aid their individual users. Agents thus have access to a much larger body of knowledge than that possessed by any individual agent. Overtime agents learn to trust the suggestions of some of their peers more than others for various classes of situations. Each agent also learns which of its peers is a reliable 'expert' for different types of situations. Multi-agent collaboration enables an inexperienced agent to make accurate predictions with high confidence as soon as it is activated as well as fill in gaps in even an experienced agent's knowledge. This will steepens the 'learning curve' and improve the handling of entirely novel situations (Lashkari et al., 1994). Though there are still problems involved with the willingness of agents helping out each other, it is not the intention of this paper to go into the discussion of multi-agent interactions under non-collaborative environment.

4.3 Evolutionary Computation

Existing AI techniques such as Evolutionary Computations (Angeline, 1995) can be used to evolve a population of personalised agents. An evolutionary computation selects a subset of the population to act as parents for a new population. Selection of parents is based on the relative worth of the candidates in the population as judged by a fitness function. It then applies a set of operators, often called mutations, to copies of the selected parents that alter their content. The resulting children comprise the subsequent population. Evolutionary Computations can be viewed as search in a space of genotypes for the ones that are the fittest (or the best adapted) to survive in the environment. Cycles of genetic variation followed by selection of the fittest produce a relatively fitter species with every generation (Sheth and Maes, 1993).

4.4 Prototyping and Testing

Finally, rapid prototyping techniques can be used to facilitate user testing and evaluation. It is possible, but difficult to determine if the program might be a trusted agent based on such properties as its correctness, completeness, efficiency, and reliability. Questions to consider in evaluation and testing should include (Riecken, 1994):

- How long does it take to become a useful assistant?
- How often does a user accept the apprentice's advice?
- How does the quality of learned advice compare with the quality that could be provided by hand-crafted rules?
- How many features (attributes) need to be considered to produce useful advice?

- Can learning keep pace with changes that occur overtime in a dynamic environment?

With the growing popularity of the *Internet*, a potentially large number of users can be involved in prototype testing. The prototype can be invoked and user comments received through, for example, the Worldwide Web.

5 Agents as Personal Travel Assistants

5.1 Agents' Tasks in Tourism

Agents can assist users in a range of different ways: they can hide the complexity of difficult tasks, perform tasks on the users' behalf, train or teach the users, help different users collaborate, and monitor events and procedures. Computer-related tasks that require expertise, skill, resources, or labour to accomplish some goals are appropriate for an agent because they are too complex for either straightforward algorithmic solutions or for complete parametric specification by the human user. One of the most difficult aspects of agent design is to define specific tasks that are both feasible using current technology, and are truly useful to the everyday user. In the travel industry, opportunities exist at all levels.

Table 1 Agents' Tasks

Information	Work	Learning (User)
Filtering	Advising	Coaching
Retrieval	Programming	Training
Navigation and Browsing	Reminding	Tutoring
Sorting and Organising	Scheduling	

A typical example is navigating a large airline database and selecting for the most suitable flight. This may involve basic linguistic, numeric, and computer skills, as well as a great deal of heuristics concerning fare bases, geography, time zones, routing systems of different airlines, etc. The nature of the complexity of such problems makes them excellent candidates for an agent application. In order to be effective, such agents require considerable detailing and subtlety in their character traits. They will need knowledge regarding the travel preferences (e.g., airline, seat, time, class, etc.) and constraints (e.g., time, budget, etc.) of the individual traveller, as well as knowledge of various travel products within the environment (e.g., frequent flyer program, fare bases, etc.).

5.2 The Scenario of a Personal Travel Assistant

The Personal Travel Assistant model can be illustrated by Figure 1.

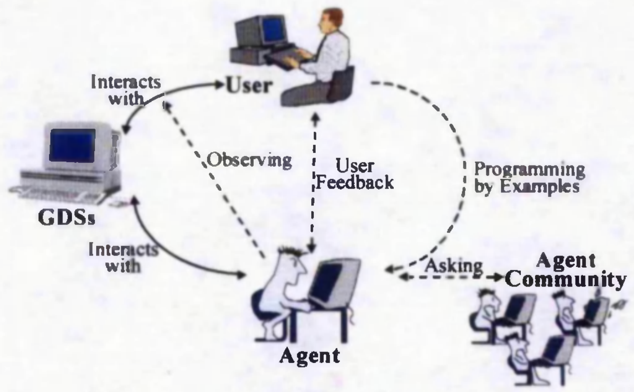


Figure 1 The Personal Travel Assistant Model

The scenario involves a non-intrusive Personal Travel Assistant (PTA) that provides viable suggested values for parameters in holiday reservations. The approach is to design a set of domain-independent reasoning modules driven by domain-specific knowledge bases. The necessary knowledge bases include explicit models of the user (i.e., the preferences of the traveller), the domain (e.g., the classification of airports, aircrafts, hotels and cars; the computation of units of time), and the interface (e.g., defaults, range and consistency checks, context-sensitive help, etc.). PTA is able to support a wide range of forms of user input and output, including natural language, graphics, voice, etc.; interpret and infer user intentions and directly assist users; modify interface features to best suit the current user and that user's ongoing tasks; and select the best form of presentation for the information based on the nature and intended use of the information.

5.3 Impacts on the Travel Industry

Customers — Changing Style: According to the August 1995 survey of travellers among Internet users (over 2,000 respondents from over 300 countries) conducted by the CIC Research, Inc. and Data Transfer Group of San Diego, over 57% of the respondents had flown twice or more on domestic business trips and 42% had flown on two or more international business trips in the past twelve months. 72% were currently shopping for a ticket and 70% were willing to deal directly through the Internet with an airline. 74% of respondents stated that the Internet was important to their travel planning and 88% stated that they were interested in using the Internet for booking their travel. The leisure market also conveys a similar picture. With the decline of standardised travel packages and inclusive tours, there is a growing trend for independent travelling. The availability of personalised intelligent systems can help with the selection and booking of travel products from an online source and ease the task in information gathering and product evaluation. Consumers will likely change their style in purchasing travel products, buying products online rather than going to the intermediaries.

Travel Agents — Changing Role: Travel agents will find their way to survive by serving in highly specialised niche markets. With changing customer taste towards individualised arrangements, the role of travel agents will certainly change from booking to counselling.

Travel Product Suppliers — Changing Distribution Strategies: Airlines, car rental companies and hotels rely on the travel agents to distribute their products to the customers, and they usually offer special discounts to the travel agents. With the availability of a new distribution channel through the Internet, these suppliers may take the opportunity to revolutionise their traditional distribution methods. The pricing strategies may change in order to make the products available online more attractive, e.g., airline database will not only include tickets with published fares, but also discounted ones. However, the actual outcome will still be very speculative.

6 Conclusion

This paper recognises the opportunities in deploying intelligent agents as personal travel assistants to assist direct consumers in selecting and booking travel products. Though the concept of agents has existed for decades, the tourism industry has yet to fulfil its potential to become one of the major application areas. Approaches such as learning techniques and Evolutionary Computations are suggested to generate adaptable knowledge-based travel consultants that will learn continuously from their users, automatically update their knowledge bases, and evolve overtime to endow themselves with genes that help contribute to the most reliable and accurate predictions.

References

Angeline, P. J., (1995) Evolution Revolution: An Introduction to the Special Track on Genetic and Evolutionary Programming, *IEEE Expert,* June, pp. 6-10.

Bennett, M. M., (1993) Information Technology and Travel Agency: A Customer Service Perspective, *Tourism Management*, August, pp. 259-266.

Bly, L., (1995) Electronic Explorer, *Los Angeles Times,* Sunday, March 12, 1995.

Business Travel News, 1994 Business Travel Survey.

CIC Research Inc. and Data Transfer Group of San Diego, August 1995 Survey.

Cockerell, N., (1991) Outbound Markets/Market Segment Studies — European Independent Travel, *Travel & Tourism Analyst*, No. 4, pp. 38-48.

Cypher, A., (1991) EAGER: Programming Repetitive Tasks by Example, *Proceedings of CHI 91,* ACM Press, N. Y., pp. 33-39.

Dent, L. et al., (1992) A Personal Learning Apprentice, *Proceedings of the National Conference on Artificial Intelligence,* MIT Press, Cambridge, Mass., pp. 96-103.

Etzioni, O. and Weld, D., (1994) A Software-Based Interface to the Internet, *Communications of the ACM,* 37(7), July, pp. 72-76.

Hitchins, F., (1991), The influence of technology on UK Travel Agents, *Travel & Tourism Analyst*, No. 3, pp. 88-105.

Kautz, H. A., Selman, B. and Coen, M., (1994) Bottom-Up Design of Software Agents, *Communications of the ACM,* 37(7), July, pp. 143-146.

Kay, A., (1984) Computer Software, *Scientific American,* 251(3), pp. 53-59.

Kay, A., (1990) User Interface: A Personal View, in Laurel, B., (eds.) *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 191-208.

Lashkari, Y., Metral, M. and Maes, P., (1994) Collaborative Interface Agents, *Proceedings of the National Conference on Artificial Intelligence,* MIT Press, Cambridge, Mass., pp. 444-449.

Laurel, B., (1990) Interface Agents: Metaphors with Character, in Laurel, B. (ed.), *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 355-365.

Maes, P. and Kozierok, R., (1993) Learning Interface Agents, *Proceedings of the AAAI'93 Conference,* MIT Press, Cambridge, Mass., pp. 459-465.

Minsky, M., (1963) Steps Towards Artificial Intelligence, in Feigenbaum, E. and Feldman, J. (eds.) *Computers and Thought*, McGraw Hill.

Mitchell, T., Caruana, R, Freitag, D, McDermott, J. and Zabowski, D., (1994) Experience with a Learning Personal Assistant, *Communications of the ACM,* 37(7), July, pp. 80-91.

Negroponte, N., (1970) *The Architecture Machine: Towards a More Human Environment,* Cambridge, Mass., MIT Press.

New York Times, 26 February, 1995.

Riecken, D., (1994) Intelligent Agents, *Communications of the ACM,* 37(7), July, pp. 18-21.

Schneiderman, B., (1983) *Direct Manipulation: A Step Beyond Programming Languages,* IEEE Computer, 16(8), pp. 57-69.

Seel, N. R., (1989) *Agent Theories and Architectures,* PhD Thesis, Department of Electrical and Electronic Engineering, University of Surrey.

Selker, T., (1994) Coach: A Teaching Agent that Learns, *Communications of the ACM,* 37(7), July, pp. 92-99.

Sheth, B. and Maes, P., (1993) Evolving Agents for Personalised Information Filtering, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 345-351.

Sheth, B., (1994) *A Learning Approach to Personalised Information Filtering,* SM Thesis, Department of Electrical Eng. and Computer Science, MIT, Feb., 1994.

Smith, D. C., Cypher, A. and Spohrer, J., (1994) KidSim: Programming Agents Without a Programming Language, *Communications of the ACM,* 37(7), July, pp. 55-67.

Stanfill, C. and Waltz, D., (1986) Towards Memory-Based Reasoning, *Communications of the ACM,* 29(12), December, pp. 1213-1228.

Varela, F. J., (1979) *Principles of Biological Autonomy,* Elsevier North Holland.

Intelligent Agents as Personal Travel Assistants

Faria Y. Y. Ng
University of Surrey
Guildford, Surrey
GU2 5XH, U. K.
e-mail: f.ng@surrey.ac.uk

1 Introduction

An intelligent agent is a computer system that tries to fulfil its goals in a complex, dynamic environment. It is situated in the environment and interacts with the user in an autonomous manner. It operates adaptively and becomes more experienced overtime in achieving its goals. This paper attempts to clear the terminology, survey the possible techniques in creating agents, analyse the appropriate application area in tourism, and finally describe a conceptual framework for a learning interface agent for a holiday selection application. The idea is to employ Machine Learning techniques to customise an agent to the traveller's personal selection rules and preferences by observing his/her actions and receiving positive or negative feedback. This approach provides the traveller with the sophisticated control over the gradual delegation of holiday selection tasks to the agent.

2 What are Intelligent Agents?

Intelligent Agents arose as an exploding research topic in the field of Artificial Intelligence. It has been the centre of discussion in the academic as well as the technical environment in recent years. Announcements of products like the Apple Newton with its agent software and General Magic's messaging agents are evidence of significant interest in agent research (Riecken, 1994). The idea of employing agents to delegate computer-based tasks goes back to research by Negroponte (1970) and Kay (1984). The research in this field is directed towards the idea of agents that have a high-level human-like communication skills and can accept high-level goals and reliably translate these to low-level tasks. Kay traced the development of the concept:

> *'The idea of an agent originated with John McCarthy in the mid-1950's, and the term was coined by Oliver G. Selfridge a few years later, when they were both at the Massachusetts Institute of Technology. They have in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a "soft robot" living and doing its business within the computer's world.'*

Various names are associated with these intelligent agents. Researchers have invented numerous new terms to better describe their agents, such as personal/automated assistants Mitchell, et al., 1994), over-the-shoulder coaches (Selker, 1994), knowbots/softbots

(Etzioni et al., 1994; Kautz et al. 1994), wizards, or sometimes just simply experts. Some popular definitions are:

> *'Learning apprentice systems are a class of interactive knowledge-based consultants that directly assimilate new knowledge by observing and analysing the problem-solving steps contributed by their users through their normal use of the system.' (Mitchell et al., 1994)*

> *'An interface agent is a semi-intelligent, semi-autonomous system that assists a user in dealing with one or more computer applications. Interface agents typically behave as personal assistants: they have knowledge about the tasks, habits and preferences of their users and use this knowledge to perform actions on their behalf.' (Maes & Kozierok, 1993)*

> *'An intelligent agent can be defined as a character, enacted by the computer, who acts on behalf of the user in a virtual (computer-based) environment. Interface agents draw their strengths from the naturalness of the living-organism metaphor in terms of both cognitive accessibility and communication style. Their usefulness can range from managing mundane tasks like scheduling, to handling customised information searches that combine both filtering and the production (or retrieval) of alternative representations, to providing companionship, advice, and help throughout the spectrum of known and yet-to-be-invented interactive contexts.' (Laurel, 1990)*

The central idea is that of a personal assistant who is collaborating with the user in the same work environment. Instead of user initiated interactions via commands and/or direct manipulation, the user is engaged in a co-operative process in which human and computer agents both initiate communication, monitor events, and perform tasks. This has been referred to as 'indirect management' (Kay, 1990) in the field of autonomous agents. The assistant becomes gradually more confident and effective as it learns the users' interests, habits and preferences (as well as those of his/her community).

3 Characteristics of Agents

3.1 Agents Have Adaptive User-Centred Interfaces

The currently dominant mode of interaction — direct manipulation (Schneiderman, 1983) — requires the user to initiate all tasks and interactions and monitor all events. This form of human-computer interface exhibits explicit responsiveness (Laurel, 1990). The user and computer communicate through a series of highly constrained, explicit terms dictated by the system. The motivation for having agents in the interface is that they provide a higher level of abstraction for user interaction. Agent-based interfaces involve more peer-to-peer interaction where the user and agent both initiate communication. Delegation of tasks is at a much higher level. The user delegate high-level goals, either explicitly stated or inferred by the system, and the agent determines how to execute these without much user consultation. The way an agent interprets and attempts to meet the goals constitutes implicit responsiveness (Laurel, 1990). One aspect of implicit responsiveness is the ability of an agent to tune its actions to the user's traits and preferences. Knowledge about the user can be obtained explicitly (by questioning) and inferred (by noticing). Users also change overtime. Even when the user's goals are explicitly the same from day to day, the way they should be interpreted changes. Responsiveness therefore requires that the agent have access to a dynamic model of the user, or at least, a record of his experience in a particular environment with rules for interpreting that experience when formulating actions.

3.2 Agents Have Expertise

An agent must have access to all the information and possible operations in its domain. It must possess (or be able to generate) both meta-knowledge and multiple representation Laurel, 1990). Meta-knowledge refers to knowledge about problem-solving in a particular domain. If the domain is an airline database, the agent must have the expertise to formulate a travel plan based on both domain information and the preferences of the traveller. The agent should also be capable of presenting the output in a usable and efficient way. Instead of displaying the results in a fixed manner, the agent reasons, based on its knowledge of the user and the user's current goals, as well as the nature of the information itself, how best to show the information so that the user detects the relevant relations (Tyler et al., 1991).

3.3 Agents Have Character

By capturing and representing the capabilities of agents in the form of character, the user is able to make accurate inferences about the internal traits, i.e., how the agent is likely to think, decide and act (values, heuristics, etc.) on the basis of some cues from its external traits, e.g., selection of the mode of representation, such as visual, audio, etc. Users can therefore predict what the agents are likely to do in a given situation on the basis of their character, and have full control on the actions. Agents are conceived by their users as coherent entities.

4 Agents as Personal Travel Assistants

Agents can assist users in a range of different ways: they can hide the complexity of difficult tasks, perform tasks on the users' behalf, train or teach the users, help different users collaborate, and monitor events and procedures. Computer-related tasks that require expertise, skill, resources, or labour to accomplish some goals are appropriate for an agent because they are too complex for either straightforward algorithmic solutions or for complete parametric specification by the human user. One of the most difficult aspects of agent design is to define specific tasks that are both feasible using current technology, and are truly useful to the everyday user. In the travel industry, opportunities exist at all levels ranging from navigating, filtering and sorting information, advising, programming, training and coaching.

4.1 Tourism Market and Trends

According to the 1994 Travel Data Center Survey, 'Approximately 2.3 million of the 39.8 million business travellers will fly 10 or more times this year on business', and '73 % of business travellers — 67% whom already travel with laptop computers — say they want the convenience of accessing travel schedules online ...,' (1994 Business Travel Survey, Business Travel News). Apart from the growing needs identified by the business travellers, there is also a growing trend towards independent travelling. Even leisure travellers will now try to book their travel products direct rather than going to the travel agent for pre-arranged holidays. Coupled with this trend is the increasing number of computer-literate customers who will welcome more user-friendly online travel reservation systems.

4.2 Online Travel Reservation Systems

New York Times (26 February) says the Internet connects at least 20 million users to 70,000 computer network world-wide, and both numbers are growing in leaps and bounds. The number of users of the online travel services is sure to increase. With the Internet and gateway systems — CompuServe, Prodigy, GEnie, Delphi, AOL, eWorld — becoming more and more popular, many online self-booking systems, such as Eaasy Sabre (data from American Airlines' Sabre), Travelshopper (data from Delta, Northwest and TWA's Worldspan), Official Airline Guide (OAG), allow travellers to tap into the same powerful, complex computer systems airlines and travel agents use to book flights, rental cars and hotels. Now, three software providers, including PCTravel, TraveLOGIX and Personal ExpertWare provide user-friendly real-time connectivity on the World Wide Web. PCTravel even offers the traveller all the necessary information with a convenient graphical presentation. Other online travel information systems include United Connection on CompuServe, and airlines' WWW browser, such as Virgin Atlantic, USAir, British Airways, Canadian Airlines, Northwest Airlines, Sothwest Airlines, Cathay Pacific, Lufthansa, etc.

4.3 How Agents can Help?

However, while consumer online reservation services have been around for as long as a decade, they represent less than 5% of all airline tickets sold. Of the estimated 1.4 million subscribers to the largest and oldest service, American Airline owned Eaasy Sabre, more than 80% are still lookers, not bookers — people who simply browse, or who make an electronic reservation but call an airline or travel agent to double-check fares and issue the ticket (Bly, 1995). As Bly (1995) suggests, 'Booking flights takes more than cursory skills.' As the online services are aimed at computer-literate travellers going on business trips, 'they require an affinity for alphabet-soup fare codes and a zen-like patience when encountering such on-screen roadblocks as "incorrect response packet" and "could not connect to gateway host".' Access to information can be both cumbersome and slow. The traveller has to have a familiarity with computers, and a certain personality type.

The gap in the industry is clear. There is a huge market, both business and leisure, yearning for systems that are easy to use and without access barriers. Travellers want information to be accurate and easily understood. They want a service that makes a complicated process comfortable. Even Sabre has recognised the need to make Eaasy Sabre easier, particularly for new computer users (Bly, 1995). It is obvious that online reservation systems are available, and travellers are ready to accept and try a new method of booking their products. What makes online booking unpopular is the difficulty in dealing with the systems and the time required for the booking process. The construction of personal travel assistants as learning agents — interactive assistants that learn continually from their users — is one approach that could help take up the routine booking procedure for the traveller and also select the most suitable products, based on the preferences and habits of the individual traveller. In order to be effective, such agents require considerable detailing and subtlety in their character traits. They will need knowledge regarding the travel preferences (e.g., airline, seat, time, class, etc.) and constraints (e.g., time, budget, etc.) of the individual traveller, as well as knowledge of various travel products within the environment (e.g., frequent flyer scheme, fare bases, etc.).

5 A Conceptual Model of a Personal Travel Assistant

The Personal Travel Assistant model can be illustrated by Figure 1.

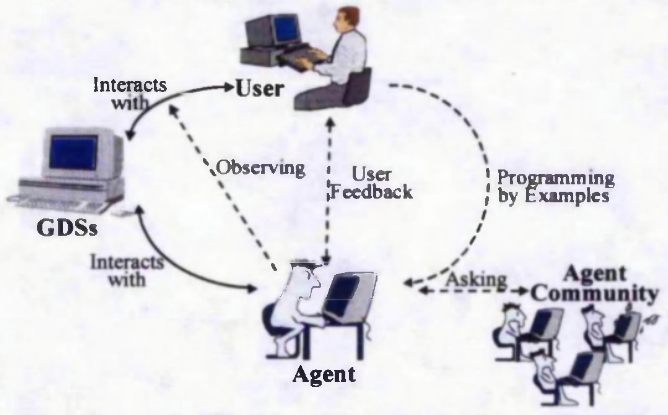


Figure 1 The Personal Travel Assistant Model

5.1 The Scenario

The scenario involves a non-intrusive Personal Travel Assistant (PTA) that provides viable suggested values for parameters in holiday reservations. The approach is to design a set of domain-independent reasoning modules driven by domain-specific knowledge bases. The necessary knowledge bases include explicit models of the user (i.e., the preferences of the traveller), the domain (e.g., the classification of airports, aircrafts, hotels and cars; the computation of units of time), and the interface (e.g., defaults, range and consistency checks, context-sensitive help, etc.). PTA is able to support a wide range of forms of user input and output, including natural language, graphics, voice, etc.; interpret and infer user intentions and directly assist users; modify interface features to best suit the current user and that user's ongoing tasks; and select the best form of presentation for the information based on the nature and intended use of the information.

PTA does not act as a layer between the user and the GDS. It behaves as a personal assistant that co-operates with the user in making holiday arrangements. The user is able to bypass the agent.

5.2 The User Interface

The user interface will display a reservation form. When the user has invoke the 'booking' command, the system will prompt the user for the parameters associated with this command. For example, when selecting a flight, there are some basic parameters that the user must supply, e.g., departure and arrival cities, travel date, and number of passengers. The status/values of the remaining attributes are inferred upon demand from the given parameters together with its background knowledge and the current date and time (e.g., previous travelling details on the same destination, other travel schedule in the same week or month of the year, etc.). PTA will also has knowledge about the user's personal details such as his/her name, address, telephone number, frequent traveller number, and credit card information.

To minimise the mistakes when the user enters a parameter, the system enforces an evolving set of legal values (range and inconsistency checks) during data entry, and provides automatic completion (defaults) where possible, thereby reducing the number of keystrokes necessary to complete a form and reducing the risk of errors (Hermens and Schlimmer, 1993).

The intelligent agent interface is attractive to users even before it has learned any knowledge, because it provides a convenient set of commands for making and changing reservations. Second, it is able to collect the training examples without imposing a significant burden on the user. Such spying agents could pick up the user's habits and preferences and perhaps make assumptions about his/her private life (Eliot, 1994). It could note what day of the week he/she prefers to travel, how long, on average, he/she usually stays and how the current situation will affect the rule tolerance (e.g., Advanced Purchase, Minimum/Maximum Stay, etc.), his/her preferred departure time, class and airlines, etc. Third, the interface is designed to maximise consistency in the data it obtains by providing automatic completion and checking of legal values (Dent et al., 1992).

6 How the Agent Learns and Works?

6.1 Learning through Observation

The interface agent learns by continuously 'looking over the shoulder ' of the user as the user is performing actions. The interface agents can monitor the activities of the users, keep track of all of his/her actions over long periods of time, find recurrent patterns and offer to automate these. The technique that the agent uses is memory-based learning (Stanfill and Waltz, 1986). The agent keeps a memory of everything the user does, stored as situation-action pairs, which are simply raw data about what happened. Situations are described as a set of attributes. When a new situation occurs, the agent will try to predict the action of the user, based on the situation-action pairs stored in its memory. The agent compares the new situation with the memorised situation-action pairs and tries to find a set of nearest neighbours (or close matches). The most similar of these memorised situations contribute to the decision of which action to take or suggest in the current situation.

The distance between a new situation and a memorised situation is computed as a weighted sum of the distances between the values of each attribute (Maes and Kozierok, 1993). The distance between attribute-values is based on a metric computed by observing how often in the example-base the two values in that attribute correspond to the same action. The agent also maintains a set of weightings for each attribute (Sypniewski, 1994). It helps to determine which attributes of the situation are most relevant to predicting the action. The weight given to a particular attribute depends upon the value for that attribute in the new situation, and is computed by observing how well that value has historically correlated with the action taken.

6.2 Learning from User Feedback

The learning technique used with feedback is reinforcement learning. Positive feedback occurs when the suggestion made by the agent is accepted by the user. Negative feedback happens when the user ignores the suggestion of the agent and takes a different action instead. The user can give direct negative feedback by telling the agent 'Don't do this action again'. The agent learns from negative feedback by adding the right action for the situation as a new example in its memory, or adjusting the weightings of the attributes of the situation. This technique helps ensure that a similar mistake is less likely to occur in the future.

6.3 Learning by Being Trained

The technique used is Programming by Examples (Cypher, 1991). The agent can learn from examples given by the user intentionally. The user can teach/train the agent by giving it hypothetical examples of events and situations and showing the agent what to do in those cases. The interface agent records the actions, tracks relationships among objects and changes the example base to incorporate the example that it is shown.

6.4 Learning by Asking Advice

If an agent does not know itself what action is appropriate in a certain situation, it can present the situation to other more experienced agents and ask what action they recommend for that situation.

7 Future Issues

Agent results will eventually draw from the results of many different disciplines. In essence, the study of agents presents a unique opportunity to integrate many significant results from many diverse research areas. Agent research also presents researchers with the opportunity to put technical results directly in the hands of the end users. In the past, many AI technologies have resulted in published papers and laboratory experiments. The application of numerous efforts, such as expert systems research, has only benefited specialised group of users. The basic idea of agent research is to develop software systems which engage and help all types of end users. Agent research needs the integration of many different research talents, and the results serve to directly benefit everyone.

In the theoretical aspect, work must proceed on the analysis of user needs and preferences vis-a-vis applications and environments. What are the qualities of a task that make it a good candidate for an agent-like interface? What kind of users will want them, and what are the differences among potential user populations? How might interface agents affect the working styles, expectations, productivity, knowledge, and personal power of those who use them?

In terms of design, the meatiest problem is developing criteria that will help select the appropriate set of traits for a given agent — traits that can form coherent characters, provide useful cues to users, and gives rise to all of the necessary and appropriate actions in a given context. Contributions will be needed from the disciplines of dramatic theory and practice, literary criticism and storytelling, and aspects of psychology and communication acts and sciences.

In the area of implementation, existing AI techniques must be explored and refined. The techniques from Artificial Life can be used to evolve a population of personalised agents (Sheth and Maes, 1993). The techniques of artificial evolution and the techniques of learning from feedback are combined to develop agents that dynamically adapt to the changing interest of the users. Evolution can be viewed as search in a space of genotypes for the ones that are the fittest (or the best adapted) to survive in the environment. Cycles of genetic variation followed by selection of the fittest produce a relatively fitter species with every generation. Techniques for constructing and enacting characters can also be imported from the field of computer game design. Expert-systems techniques can be applied to such 'soft' problem as learning and assimilating the user's style and preferences, developing navigational strategies, and creating alternate representations. Work on such technologies as language and speech processing, paralinguistic, story generation, image recognition, and intelligent animation can be refocused and revitalised by the agent's platform. Finally, rapid prototyping techniques must be developed to facilitate user testing and evaluation.

8 Conclusion

Personal learning agents can produce useful knowledge-based assistants with significant reduced manual development and maintenance of the knowledge base. It is possible to design an agent that is attractive to the users, able to capture useful training data, and capable of generalising from such data to learn a customised knowledge base competitive to hand-coded knowledge (Dent et al., 1992). Intelligent agents will certainly relieve users with some tasks that they are unable or unwilling to do.

References

Bates, J., (1994) The Role of Emotion in Believable Agents, *Communications of the ACM,* 37(7), July, pp. 122-125.

Bly, L., (1995) Electronic Explorer, *Los Angeles Times,* Sunday, March 12, 1995.

Boden, M. A., (1994) Agents and Creativity, *Communications of the ACM,* 37(7), July, pp. 117-121.

Chin, D. N., (1991) Intelligent Interfaces as Agents, in Sullivan, J. and Tyler, S. (eds.), *Intelligent User Interfaces,* ACM Press, N. Y., pp. 177-205.

Crowston, K. and Malone, T. W., (1988) Intelligent Software Agents, *BYTE,* 13(13), December, pp. 267-270.

Cypher, A., (1991) EAGER: Programming Repetitive Tasks by Example, *Proceedings of CHI 91,* ACM Press, N. Y., pp. 33-39.

Dent, L. et al., (1992) A Personal Learning Apprentice, *Proceedings of the National Conference on Artificial Intelligence,* MIT Press, Cambridge, Mass., pp. 96-103.

Edmonds, E. A., Candy, L., Jones, R. and Soufi, B., (1994) Support for Collaborative Design: Agents and Emergence, *Communications of the ACM,* 37(7), July, pp. 41-47.

Eliot, L., (1994) Intelligent Agents are Watching You, *AI Expert,* August, pp. 9-11.

Etzioni, O. and Weld, D., (1994) A Software-Based Interface to the Internet, *Communications of the ACM,* 37(7), July, pp. 72-76.

Genesereth M. R. and Ketchpel. S. P., (1994) Software Agents, *Communications of the ACM,* 37(7), July, pp. 48-53.

Gmytrasiewicz, P. J., Durfee, E. H. and Wehe, D. K., (1991) A Decision-Theoretic Approach to Coordinating Multiagent Interactions, *Proceedings of the 12th International Joint Conference on Artificial Intelligence,* Sydney, Australia, pp. 62-68.

Greif, I., (1994) Desktop Agents in Group-Enabled Products, *Communications of the ACM,* 37(7), July, pp. 100-104.

Guha, R. V. and Lenat. D. B., (1994) Enabling Agents to Work Together, *Communications of the ACM,* 37(7), July, pp. 127-142.

Hermans, L. A. and Schlimmer, J. C., (1993) A Machine-Learning Apprentice for the Completion of Repetitive Forms, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 164-170.

Johanser, J. T. and Harbidge, R. M., (1986) *Validating Expert Systems: Problems & Solutions in Practice,* Presented at KBS '86: Online Publications, Pinner, U. K., pp. 215-229.

Kautz, H. A., Selman, B. and Coen, M., (1994) Bottom-Up Design of Software Agents, *Communications of the ACM,* 37(7), July, pp. 143-146.

Kay, A., (1984) Computer Software, *Scientific American,* 251(3), pp. 53-59.

Kay, A., (1990) User Interface: A Personal View, in Laurel, B., (eds.) *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 191-208.

King, J. A., (1994) NDM and Knowledge Acquisition, *AI Expert,* October, pp. 14-16.

Kozierok, R. and Maes, P., (1993) A Learning Interface Agent for Scheduling Meetings, *Proceedings of ACM SIGCHI International Workshop on Intelligent User Interfaces,* ACM Press, N. Y., pp. 81-88.

Kozierok, R., (1993) *A Learning Approach to Knowledge Acquisition for Intelligent Interface Agents,* SM Thesis, Department of Electrical Eng. and Computer Science, MIT, May, 1993.

Kraus, S., (1993) Agents Contracting Tasks in Non-Collaborative Environments, *Proceedings of the 11th National Conference on Artificial Intelligence,* AAAI Press, Washington D. C., pp. 243-248.

Lander, S. E. and Lesser, V. R., (1993) Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents, *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambery, France, pp. 438-444.

Laurel, B., (1990) Interface Agents: Metaphors with Character, in Laurel, B. (ed.), *The Art of Human-Computer Interface Design,* Addison-Wesley, Reading, Mass., pp. 355-365.

Lieberman, H., (1993) Mondrian: A Teachable Graphical Editor, in Cypher, A. (eds.) *Watch What I Do: Programming by Demonstration,* Cambridge, Mass., MIT Press.

Maes, P., (1994) Agents That Reduce Work and Information Workload, *Communications of the ACM,* 37(7), July, pp. 31-40.

Maes, P. and Kozierok, R., (1993) Learning Interface Agents, *Proceedings of the AAAI'93 Conference,* MIT Press, Cambridge, Mass., pp. 459-465.

Malin, J. T. and Schreckenghost, D. L., (1993) User as Intelligent Agent: Interface Design for Intelligent Monitoring System, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 461.

Minsky, M. and Riecken, D., (1994) A Conversation with Marvin Minsky About Agents, *Communications of the ACM,* 37(7), July, pp. 23-29.

Mtichell, T., Caruana, R, Freitag, D, McDermott, J. and Zabowski, D., (1994) Experience with a Learning Personal Assistant, *Communications of the ACM,* 37(7), July, pp. 80-91.

Myers, B. A., (1986) Creating Highly-Interactice and Graphical User Interfaces by Demonstration, *SIGGRAPH 86,* 20(4), August 18-22, 1986.

Negroponte, N., (1970) *The Architecture Machine: Towards a More Human Environment,* Cambridge, Mass., MIT Press.

Ng, F. Y. Y., (1995) Business Travellers' Expert Advisor, in Schertler, B. et al. (eds.) *Information and Communications Technologies in Tourism,* ENTER 95 International Conference in Innsbruck, Austria, 18-20 January, 1995, Springer-Verlag Wien, New York, pp. 288-298.

Norman, D. A., (1994) How Might People Interact with Agents, *Communications of the ACM,* 37(7), July, pp. 68-71.

Riecken, D., (1994) M: An Architecture of Integrated Agents, *Communications of the ACM,* 37(7), July, pp. 106-116.

Riecken, D., (1994) Intelligent Agents, *Communications of the ACM,* 37(7), July, pp. 18-21.

Schneiderman, B., (1983) *Direct Manipulation: A Step Beyond Programming Languages,* IEEE Computer, 16(8), pp. 57-69.

Selker, T., (1994) Coach: A Teaching Agent that Learns, *Communications of the ACM,* 37(7), July, pp. 92-99.

Sheth, B. and Maes, P., (1993) Evolving Agents for Personalised Information Filtering, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, N. Y., pp. 345-351.

Sheth, B., (1994) *A Learning Approach to Personalized Information Filtering,* SM Thesis, Department of Electrical Eng. and Computer Science, MIT, Feb., 1994.

Smith, D. C., Cypher, A. and Spohrer, J., (1994) KidSim: Programming Agents Without a Programming Language, *Communications of the ACM,* 37(7), July, pp. 55-67.

Stanfill, C. and Waltz, D., (1986) Towards Memory-Based Reasoning, *Communications of the ACM,* 29(12), December, pp. 1213-1228.

Sypniewski, B. P., (1994) The Importance of Being Data, *AI Expert,* November, pp. 23-31.

Tyler, S. W. et al., (1991) An Intelligent Interface Architecture for Adaptive Interaction, in Sullivan, J. and Tyler, S. (eds.), *Intelligent User Interfaces,* ACM Press, N. Y., pp. 85-109.

Need an Expert for Holiday Selection?

Faria Y.Y. Ng & Silvia Sussmann

Department of Management Studies

University of Surrey

Guildford, Surrey, GU2 5XH, UK

1 Introduction

This article intends to review the results of a research project involving the construction of an expert system prototype: ***Business Travel Counsellor (BTC),*** aimed at providing assistance to travel agent staff in matching the available products with the needs of business trips. First, the concept of an expert system is defined. A review of the existing literature on expert system applications in tourism is undertaken, to set the context in which this research is carried out. Finally, the prototype is described in detail, with suggestions on the areas for future research now being actively pursued.

2 What are Expert Systems?

A recent definition by El-Najdawi and Stylianou highlights the most important features to be expected from viable commercial expert systems:

> *Expert Systems are computer programs that embody the knowledge of one or more experts in a narrow domain and can solve problems in that domain matching the expert's level of performance (El-Najdawi and Stylianou, 1993)*

An expert system is a computer-based system that makes decisions based on facts and rules embedded in a knowledge base. It can broadly be divided into three major components: the knowledge base, the inference engine, and the user interface.

The knowledge base is the nucleus of an expert system. It encapsulates the knowledge of human experts and represents their knowledge of facts, judgements, rules, intuition, and experience about a particular domain in some symbolic manner.

The inference engine manipulates and interrogates the knowledge base in order to test a particular line of reasoning and perform logical deductions. It contains the reasoning methods that may be used by human experts for solving problems. The reasoning process can be modelled either as deductive: given the facts and rules infer the conclusion, or inductive: analyse the actions and decisions and infer the rules from them.

The user interface provides the communication exchange between the system and the users. It is responsible for eliciting user requirements, and at the same time, transmitting expert advice. It may also give an explanation of the basis of the advice given.

3 Expert Systems in Tourism

The main economic rationale for expert system application is either scarcity, cost or availability of experts, along with the need to train new experts. In the field of tourism, there is a fairly limited literature which identifies suitable areas for the development of expert systems. Crouch (1991) made a comprehensive review of emerging possibilities, concluding that applications in tourism retailing should be given priority over other sectors, such as accommodation, transportation or tourist offices. McCool (1987) postulated that the greatest advantages accrue when the user is paired with the expert system and provides the overall problem direction. Bruce (1989) and Doll (1989) recognised the potential of expert systems in travel counselling in their reviews of Information Technology applications in the fields of tourism and transportation. Hruschka and Mazanec (1990) stated that *'an expert system designed for the travel agent to assist in travel counselling had not yet been introduced'*. They developed the first working prototype on leisure vacation travel counselling, which used two different approaches: an expert system — based on an expert system shell — and a retrieval system, written in Prolog. In both cases, the object was matching the traveller's concept of an ideal tour. Other examples found in the literature address the problems of travel scheduling or itinerary (Bodi and Zeleznikov, 1988; Bose and Pudala, 1987; Frietag and Biernath, 1988). More recently, Mulye and Rickard (1995) have proposed a decision support system to enhance the potential of counselling expert systems in the area of destination choice.

4 Travel Counselling as the Application Domain

A field research conducted in one of the travel management centres of a large multiple showed that the travel consultants performed below the performance standard required by the company and they need domain specific expertise to speed up their counselling service. As reflected in this research, the level of experience of consultants varies significantly. Though a large proportion (70%) has two to five years' experience, the rest of them showed a dimension ranging from less than one to more than 16 years. Therefore, the same customer with the same requirements may be recommended different products by different consultants. Moreover, human decisions are inconsistent by nature. The same inputs do not always produce the same decisions. This may be due to external distractions, forgetting to get key pieces of information and even changes in an individual's mood. Therefore, even the same consultant may have varied performance at different times. All these problems showed that there is no guarantee that the customer really gets the most suitable product.

An expert system can come to help by encoding the *pooled expertise* of the expert consultants into a computer. All the knowledge will then be agreed upon and any inconsistencies removed. It can offer speedy 'customer-product' matching because it thinks like a human expert, applying rules in a non-procedural manner, jumping to quick hypotheses and offering speedy conclusions. Not only will it *smooth individual variances*, but also help to *diminish the variances across decision-makers*. In this way, consistent and unbiased advice can be provided. Besides, capturing expertise in the form of an expert system gives *permanency* to the expertise and allows it to become *ubiquitous*. The private knowledge of experienced travel consultants can be transformed into a tangible item and distributed to everyone via a computer. Since an expert system can easily be duplicated, it is possible for inexperienced and new staff to gain specialised assistance without having to deal with the time and location limitations that restrict human experts.

5 The Business Travel Counsellor

An initial prototype was constructed with a reasonable but restricted set of options. The main focus of the prototype is to show the viability of a new powerful concept: the *integration of expert systems and Central Reservation Systems (CRSs)* in the airline industry. Such kind of integration has been recognised as an important functional element in future global distribution systems (Goeldner, 1994; Sheldon, 1992).

5.1 Knowledge Acquisition

Complex knowledge acquisition cannot be achieved by a single method; a 'mix and match' technique (Hart, 1991), *protocol analysis* plus *interviews,* was therefore employed. Knowledge was extracted with assistance from experienced travel consultants in the field research.

Protocol Analysis

Protocol analysis involves a knowledge engineer observing how a domain expert solves problems. The knowledge engineer may directly query the expert. Alternatively, the knowledge engineer may passively observe the expert 'speaking out loud' while working through a problem (Chadha et al., 1991; Kim and Courtney, 1988). In the case of BTC, multiple counselling sessions of a team of expert travel counsellors were recorded. From the actual dialogue between the clients and the consultants, it was possible to extract not only the knowledge needed for the expert system, but also a benchmark against which to compare the prototype once developed. The observations were supplemented by an analysis of transcripts produced from recorded notes of the 'speaking out loud' problem sessions. Interference with the expert consultants' decision-making process was thus minimised, by avoiding interruptions through questions.

Interviews

Follow-up interviews were conducted to complement the results from the protocol analysis. The experts were asked why they made specific decisions. Sometimes, they had formed such a close relationship with their clients that they could apply what they already knew from the customers' profiles, e.g., a customer prefers to fly with a particular airline every time, or a corporation will have price as its first priority when they book flights for its staff. Therefore, the consultants were asked to verbalise the thoughts behind their decisions.

5.2 Prototype Construction

BTC was constructed using a simple commercial expert system shell. The shell provides a backward-chaining inference engine and tools for building graphical user interfaces. The inference engine derives conclusions from both a rule base and a database. The rule base contains heuristic knowledge collected from the field research concerning how travel consultants make decisions. The database contains information which simulates the input from a CRS, in response to actions triggered by the inference engine. The user interface collects users' requirements such as date, time, class of travel, etc., and displays details of the recommended flights together with other information useful to the trip. The custom components of BTC are described in the following sections. The architecture of BTC is shown in Figure 1.

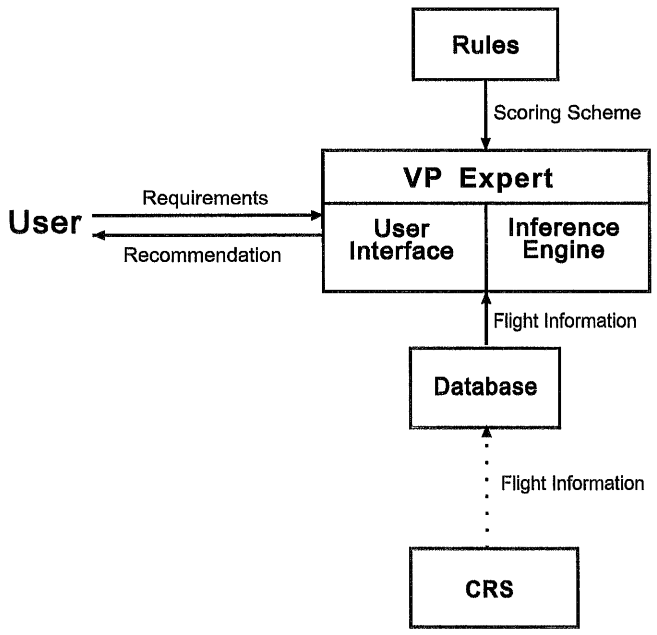


Figure 1 The Functional Architecture of BTC

The Database

After accepting the user's requirements, BTC will start processing them and send commands to the simulated CRS system calling for relevant information, typically a list of suitable flights. The familiar perception of a CRS system involves a travel consultant reading information from a terminal. For BTC, the information extracted from the CRS is assumed to be arranged in a *machine-readable format* that can be further analysed, i.e., a database. This takes place internally, in a form which is totally transparent to the user. Though the effective real time connection was not tested, the structure was accurately simulated, based on real CRS output collected in the field work.

The BTC database is *relational* and consists of several linked files. The database structure is designed to handle the full flight information supplied by the SABRE system, however, this initial prototype does not include all possible details. The problems involved in the SABRE emulation are not trivial, e.g., the representation of the complex fare structure of individual seats, which depends on the date of travel, length of stay, ticket restrictions, etc., or the representation of the seat plan which allows not merely the selection of aisle and window seats, but also the identification of adjacent seats.

The Rule Base

The complexity of human decision making is reflected by the *'scoring scheme'* in which rules are formulated to handle complex users' requirements and different weighting/importance levels. Simultaneous optimisation for all the requirements can be achieved. The principle of the scoring scheme is to assign a score for each unique option based on two aspects: *'matchability'* and *'weighting'*. Flights with the *lowest* overall score will be selected.

The 'matchability' criterion assesses how much the option can fit into the user's requirements. For example, if the customer wants to be at a destination before 1500, a flight arriving at 1400 will certainly have a lower score for the 'arrival time' than a flight reaching the airport at 1700. In BTC, the rule for this case can be illustrated as follows:

```
IF          preferred_arrival_type = before
AND         preferred_arrival_time > scheduled_arrival_time
THEN        time_score = (preferred_arrival_time - scheduled_arrival_time) x cost_scale)
```

The above means that the scores are worked out by calculating the discrepancies between the preferred arrival time and the scheduled arrival time plus an adjustment according to a 'cost scale' with £100 as the base fare. In the rule above, the cost scale is £2 per hour as the traveller can still manage to reach the destination before the preferred time, though a bit earlier than the exact hour he wants.

In another rule that handles the case when the flight is scheduled to arrive later than the preferred time, the scale is set at £5 per minute, because there would be much more inconvenience to the traveller.

```
IF          preferred_arrival_type = before
AND         preferred_arrival_time < scheduled_arrival_time
THEN        time_score = (scheduled_arrival_time - preferred_arrival_time) x cost_scale)
```

In an operational system, these cost scales will be predetermined by experts, who consider how much extra typical customers would pay for exactly what they prefer, by each £100 of the ticket cost. These costs are then adjusted in proportion to the actual fare. In more complex cases, cost scales can be determined by rules that take other circumstances into account. Since the scheduled arrival time is one of the fields in the database, the associated score is called a field score.

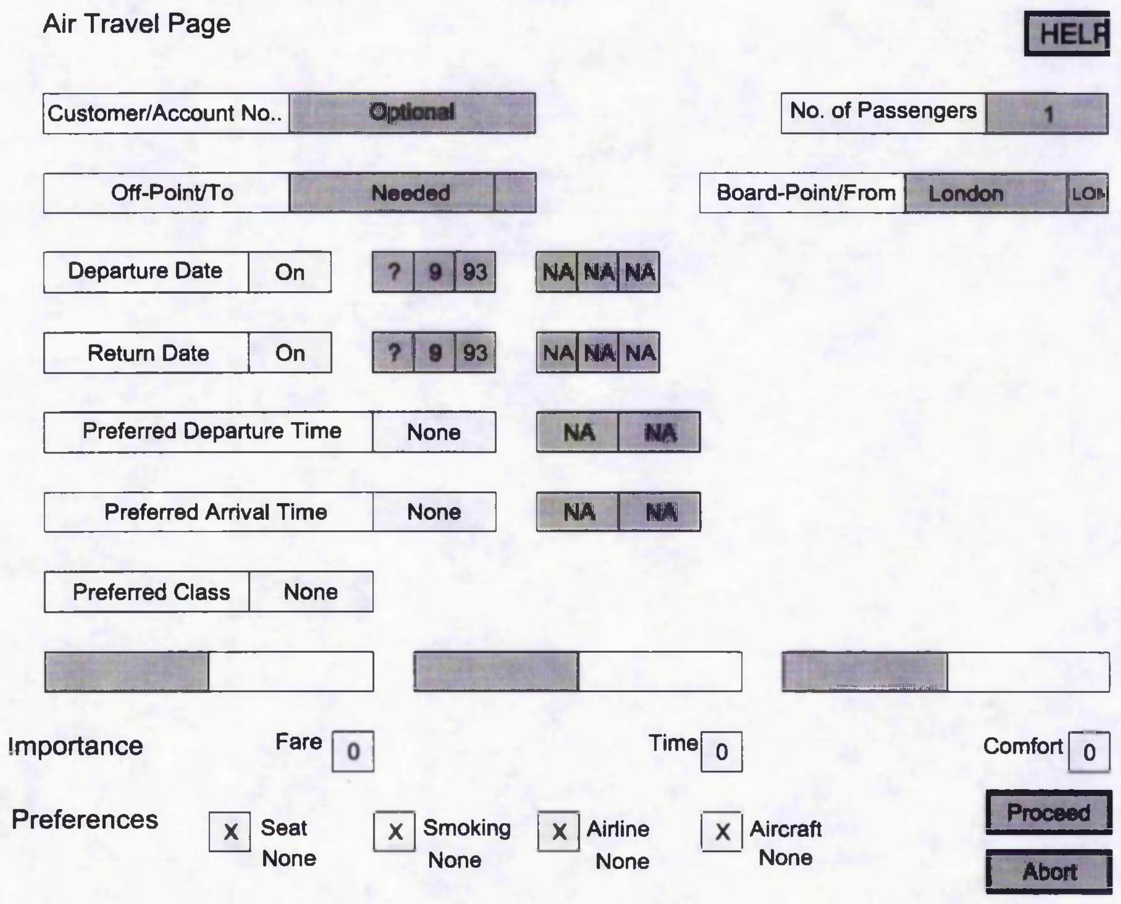
In terms of 'weighting', the importance that a particular traveller associates with a field is also considered as travellers will have individual preferences. The relative importance of the fields also depends on circumstances. For example, if the flight is short-haul, a smoking seat is much less important to a smoker. The arrival and departure times are less important in long-haul flights as absolute punctuality is not easy and the traveller will probably allow for some reasonable delay in such cases. The total score for each candidate flight is the sum of the products of individual field marks and their weights.

Though human experts do not work with numerical scores, they do consider 'matchability' and 'weighting' (Kattan, 1994; Sypniewski, 1994). BTC attempts to emulate human experts except that the final assessment is quantitative. The advantages of using a structured scoring scheme are consistency, ease of averaging expert opinions and simplicity of software maintenance.

The User Interface

The design principle for the user interface is: (1) to exploit the advantages of using an expert system to arrive at a user friendly interface; and (2) to design a user interface that exhibits expert behaviour in data collection. These design concepts integrate user interface and expert system design. It can improve expert system robustness and usability. Usability is improved by designs based on task-relevant information requirements. Robustness is improved by designs with support for user supervision and recovery (Malin and Schreckenghost, 1993). With careful considerations about the level of users' expertise, the graphical user interface is potentially suitable for direct access by customers.

BTC is designed to interact with users via an *intelligent form* to ease the task of data entry and to reduce errors. The use of a form allows information to be gathered in arbitrary sequences, without hindering natural conversation with customers, whilst operating BTC. The intelligent form is shown in Figure 2.


Air Travel Page

HELP

Customer/Account No.. Optional

No. of Passengers 1

Off-Point/To Needed

Board-Point/From London LON

Departure Date On ? 9 93 NA NA NA

Return Date On ? 9 93 NA NA NA

Preferred Departure Time None NA NA

Preferred Arrival Time None NA NA

Preferred Class None

Importance Fare 0 Time 0 Comfort 0

Preferences X Seat None X Smoking None X Airline None X Aircraft None

Proceed

Abort

Figure 2 The Intelligent Form

The form is divided into *windows*, each containing a number of *fields* for the operator to enter the data. Key information such as date and time of travel are entered in the main window. These are the details that customers are most likely to tell their consultant. The number of key fields is kept to a minimum, since an experienced consultant will not ask unnecessary questions. On the other hand, a large number of optional fields are present to satisfy customer preferences that are only occasionally specified, e.g., the type of aircraft. These optional fields are placed in a number of auxiliary windows that are only displayed when their corresponding buttons are selected on the main window.

Context sensitive fields are implemented to speed up form entry which is illustrated as shown for the preferred departure time in Figure 3.

Preferred Departure Time	Morning	NA	NA
	Afternoon	NA	NA
	Evening	NA	NA
	Around	?	NA
	Before	?	NA
	After	?	NA
	Between	?	?
	None	NA	NA

Figure 3 Context Sensitive Fields

If the customer wants to depart around a specific time, the BTC operator then points and clicks at the question mark before entering the time. If the highlighted word **Around** is selected, a *pull-down menu* appears which provide choices such as **Morning** and **Between.** The time field is dynamic, i.e., if **Morning** is selected, the field will disappear immediately, whilst an additional question mark will appear if **Between** is selected. This design avoids large number of unnecessary fields and labels obscuring the screen. Also, *default values* suggesting likely answers are supplied if possible and they may change dynamically. For example, the current month and year are the defaults for all dates, and the default month/year of the return trip are those of the outward trip.

Whenever a field changes value, special rules associated with the field are triggered. These rules are used to check likely data errors and, more importantly, potential conflicts amongst fields. Rules on individual fields are mostly range checks to discover mistakes such as trips to the past. More complex rules examine the interrelations amongst fields, e.g., if both preferred arrival and departure times are entered, the initial entry will be cleared since the fixed flight time will not allow both fields to be specified arbitrarily.

The relative importance levels, or weights, of the fields are determined by rules as explained above. Customers are unaware of individual weights, but three importance levels, **Fare**, **Time** and **Comfort** are provided to allow customisation. The default values of these importance levels are one, i.e., the three criteria are equally important to the customer. The importance levels can be changed by direct data entry, or by sliding horizontal gauges. Individual field weights are then derived from the importance levels by a set of rules. For example, if Time is greater than one, the schedule of the flight that matches the customer's requirement is given higher priority.

If there are no reported errors and all mandatory fields are completed, the user can leave the intelligent form and enter the consultation screen. The operator is presented with a split screen, on which the customers' requirements are shown alongside the recommended flights so that it is easy to check if the requirements were handled satisfactorily. If the customer is satisfied, the operator should then start the computerised booking procedures, otherwise, BTC can be aborted or the intelligent form can be restarted and modified. Finally, extra information is displayed in boxes containing ticket restrictions and destination information such as visa requirements.

Context sensitive help information is provided at all times implemented by hypertext tools of the expert system shell. Help menus relevant to the current user action are located in pop-up menus, error message boxes and all windows.

6 Future Research

Recent market analyses show a common and coherent trend: decrease in standardised travel packages and inclusive tours (Bennett, 1993; Hitchins, 1991). With travel information more and more readily available, the trends appear to point towards *independent travelling*, i.e., the customers will mix and match travel products for their self-planned holidays instead of going to the tour operators for a pre-arranged package. Coupled with this trend is the changing role of the travel agents, which could evolve from providing convenient booking to *giving customised advice* as a valued added service (Bruce, 1994). Therefore, research is currently continuing into the development of a more comprehensive expert system, based on the artificial intelligent language Prolog, to cater for both the business and independent travellers (Ng and Sussmann, 1994).

Personalised knowledge-based systems have not yet become widespread in the travel industry despite their potential. This is due, partly to the high costs of developing and maintaining customised knowledge bases. The future of expert systems depends on finding a solution to automate the knowledge acquisition process. An expert system does not mimic human intelligence unless it is adaptive, that is: *it learns*. Learning is perceived as an integral part of expert systems in the future. Intelligence must be viewed as an evolutionary adaptation (Rubin, 1993). Therefore, parallel research has attended to the possibility of employing intelligent agents for travel counselling. The construction of interactive personal assistants is one approach that could dramatically reduce the cost of knowledge-based advisors. In order to be effective, such an agent will need considerable knowledge regarding the travel preferences (e.g., airline, seat, hotel, etc.) and constraints (e.g., time, budget, etc.) of the individual traveller, as well as knowledge of various travel products within the environment (e.g., frequent flyer scheme, weekend discounts for hotels, etc.). The agent is programmed by examples at the start and it evolves by observing the user's actions and receiving direct user feedback. The idea is to employ Machine Learning techniques to customise itself to the user's personal selection rules and preferences. This approach provides the user with the sophisticated control over the gradual delegation of holiday selection tasks to the agent.

The use of *multi-media* has also shown its importance as a competitive tool (Sheldon, 1994). We are sure to expect an increasing number of multi-media booking and information systems emerging in the market, such as the Thomas Cook Travel Kiosk at their Marble Arch branch which shows still images of destinations (CD ROM User, 1994), etc. With the audio-visual output of product information, the system is able to appeal to the psychological needs of the customers, making the products more tangible by allowing customers to 'experience' them on the screen. An expert system with direct customer involvement through multi-media is going to be as important as the knowledge it encapsulates.

References

Bennett, M. M., (1993), Information Technology and Travel Agency: A Customer Service Perspective, *Tourism Management*, August, pp. 259-266.

Bodi, A. and Zeleznikow, J., (1988), CATA: A Computer Aided Travel Assistant, *Proceedings of Frontiers of Australian Tourism Conference,* Australian National University, Canberra, ACT, 29 June - 1 July.

Bose, P. K. and Pudala, A. M. R., (1987), Reasoning with Incomplete Knowledge in an Interactive Personal Flight Planning Assistant, *Proceedings Seventh Avignon Conference on Expert Systems,* Avignon, France, pp. 1077-92.

Bruce, M., (1994), Technological Change and Competitive Marketing Strategies, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 430-433.

CD ROM User, (1994), The Information Highway: Virtual Holidays, July 1994.

Chadha, S. R., Mazlack, L. J. and Pick, R. A., (1001), Using Existing Knowledge Sources (cases) to Build an Expert System, *Expert Systems*, November, 8(4), pp. 3-12.

Crouch, G. I., (1991), Expert Computer Systems in Tourism: Emerging Possibilities, *Journal of Travel Research,* 29(3), pp. 3-10.

El-Najdawi, M. K. and Stylianou, A. C., (1993), Expert Support Systems: Integrating AI Technologies, *Communications of the ACM*, 36(12), December, pp. 55-65.

Freitag, B. and Biernath, O., (1988), An Airtravel Expert Database, *Proceedings of the 3rd International Conference on Data and Knowledge Bases,* Jerusalem, June, pp. 32-46.

Goeldner, C. R., (1994), Tourism Information Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 178-182.

Hart, A., (1991), Report of Workshop on the Use of Knowledge Acquisition Techniques with Structured Systems Methods, Held at BP, London, 5 December 1990, *Expert Systems*, November, 8(4), pp. 38-40.

Hitchins, F., (1991), The influence of technology on UK Travel Agents, *Travel & Tourism Analyst*, No. 3, pp. 88-105.

Hruschka, H. and Mazanec, J., (1990), Computer-Assisted Travel Counselling, *Annals of Tourism Research,* Volume 17, pp. 208-227.

Kattan, M., (1994), Inductive Expert System vs. Human Experts, *AI Expert*, April, pp. 32-38.

Kim, J. and Courtney, J. M., (1988), A Survey of Knowledge Acquisition Techniques and their Relevance to Managerial Problem Domains, *Decision Support Systems,* 4, pp. 269-284.

Maes, P. and Kozierok, R., (1993), Learning Interface Agents, *Proceedings of the AAAI'93 Conference,* MIT Press, Cambridge, Mass., pp. 459-465.

Malin, J. T., (1993), User as Intelligent Agent: Interface Design for Intelligent Monitoring Systems, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, p. 461.

McCool, A. C., (1987), Some Considerations in Developing Expert Systems for the Hospitality Industry, *International Journal of Hospitality Management,* 6(4), pp. 191-198.

Mulye, R. and Rickard, J., (1995), A Decision Support System for Tourism Destination Choice, in Schertler et al. (eds.), *Information and Communications Technologies in Tourism, Proceedings of the ENTER 95 International Conference*, Innsbruck, Tyrol, Austria, 18-20 January, 1995, Sprincer Verlag, Vienna, pp. 299-307.

Ng, F., (1995), Business Travellers' Expert Advisor, in Schertler et al. (eds.), *Information and Communications Technologies in Tourism, Proceedings of the ENTER 95 International Conference*, Innsbruck, Tyrol, Austria, 18-20 January, 1995, Sprincer Verlag, Vienna, pp. 288-298.

Ng, F., (1995), Designing Expert Travel Counselling Systems with a Better End-User Involvement, *Proceedings of the 1995 Indonesian-Swiss Forum on Culture and International Tourism,* Yogyakarta, Indonesia, 23-26 August, 1995, (forthcoming).

Ng, F. and Sussmann, S., (1994), The Expert Business Travel Counsellor, Tourism: The State of the Art, University of Strathclyde, Glasgow, Scotland, 11-14 July, 1994.

Rubin, S. H., (1993), Machine Learning and Expert Systems, *AI Expert*, June, pp. 32-37.

Sheldon, P. J., (1992), Researching Consumer Information: Destination Information Systems: Examples of Electronic Market Places, *Proceedings of the 23rd Annual Conference,* Travel and Tourism Research Association, Minneapolis, MN, June 14-17, pp. 37-49.

Sheldon, P. J., (1994), Information Technology and Computer Reservation Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 126-130.

Sussmann, S. and Ng, F., (1995), Business Travel Counselling, *Annuals of Tourism Research,* 22(2).

Sypniewski, B. P., (1994), The Importance of Being Data, *AI Expert,* November, 23-31.

The Expert Travel Counselling System — A Case Study

Faria Ng
Department of Management Studies
University of Surrey
Guildford, Surrey
GU2 5XH

1 Introduction

Expert Systems are computer programs that embody the knowledge of one or more human experts in a narrow problem domain and can solve problems in that domain matching the expert's level of performance (El-Najdawi and Stylianou, 1993). Capturing expertise in the form of an expert system gives permanence to the expertise and allows it to become ubiquitous, which is termed as 'attaining corporate immortality' by McMullen (McMullen, 1987). This article describes the construction of an expert system prototype — **B**usiness **T**ravel **C**ounsellor (BTC). It is aimed to provide assistance to travel agent staff to match the available products with the needs of business trips. Since an expert system can easily be duplicated, it is possible for inexperienced travel consultants to gain specialised assistance without having to deal with the time and location limitations that restrict human experts (Hruschka and Mazanec, 1990). In other words, there will always be an expert around to help.
A powerful feature of BTC is the integration of expert systems and **C**entral **R**eservation **S**ystems (CRSs) in the airline industry. Such kind of integration has been recognised as an important functional element in future global distribution systems (Goeldner, 1994; Sheldon, 1992). It is also capable of generating a market profile on products as well as customers (Burke, 1986). Travel agents can, therefore, focus their follow-up marketing efforts on likely prospects. This would in turn save time and money, and increase the likelihood that they could successfully convert inquirers into paying customers. Such satisfied customers would represent the nucleus of a dedicated clientele group, and might also provide the agent with positive 'word-of mouth testimony' to potential customers.

2 Knowledge Acquisition

As no single method will suffice for complex knowledge acquisition, a 'mix and match' method (Hart, 1991) — Protocol analysis plus interviews was employed. Knowledge was extracted with assistance from experienced travel consultants, at a travel management centre of one of the large multiples.

Protocol Analysis
Protocol analysis involves a knowledge engineer observing how a domain expert solves problems. The knowledge engineer may directly query the expert. Alternatively, the knowledge engineer may passively observe the expert 'speaking out loud' while working through a problem (Chadha et al., 1991; Kim and Courtney, 1988, Sussmann and Ng, 1994). In the case of BTC, multiple counselling sessions of a team of expert travel counsellors were recorded. From the actual dialogue between the clients and the consultants, it was possible to extract not only the knowledge needed for the expert system, but also a benchmark against which to compare the prototype once developed. The observations were supplemented by an analysis of transcripts produced from recorded notes of the 'speaking out loud' problem sessions. Interference with the expert consultants' decision-making process, which might be caused when asking questions, was minimised.

Interviews
Follow-up interviews were conducted to complement the results from the protocol analysis. The experts were asked why they make 'such and such' decisions. Sometimes, they have formed a close relationship with their clients that they simply apply what they know from the customers' profiles, e.g., a customer prefers to fly with BA every time, or a company will put price in the first place when they book flights for their staff. Therefore, the consultants were asked to verbalise their immediate thoughts behind their decisions.

3 Prototype Construction

BTC was constructed using a simple commercial expert system shell. The shell provides the inference engine and tools for building graphical user interfaces. The inference engine derives conclusions from both a rule base and a database. The rule base contains heuristic knowledge collected from the field research concerning how travel consultants make decisions. The database contains information received from a CRS, in response to actions triggered by the inference engine. The user interface collects users' requirements such as date, time, class of travel, etc., and displays details of the recommended flights together with other information useful to the trip. The custom components of BTC are described in the following sections. The architecture of BTC is shown in Figure 1.

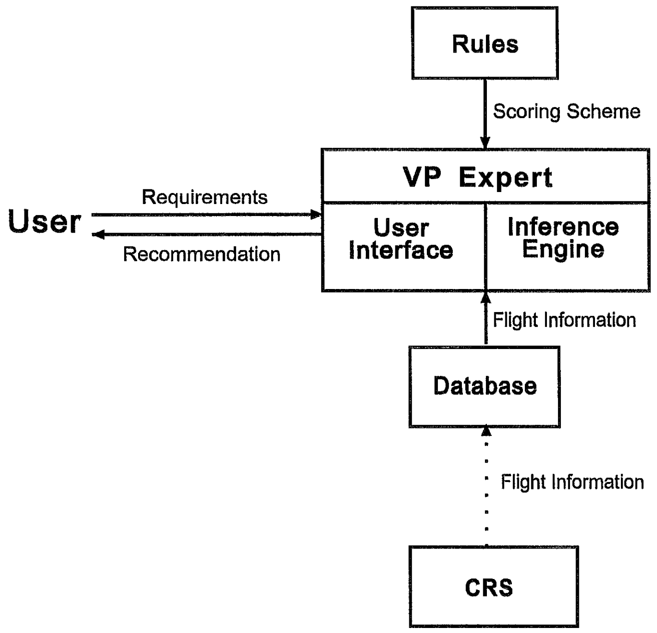


Figure 1 The Functional Architecture of BTC

The Database
After accepting the user's requirements, BTC will start processing them and then sent commands to the CRS system to call for relevant information, typically a list of relevant flights. The familiar perception of a CRS system is that of a travel consultant reading information from a terminal. For BTC, the information extracted from the CRS is assumed to be arranged in a *machine-readable format* that can be further analysed, i.e., a database. This, in fact, will take place internally without the user noticing at all. Though the effective real time connection was not tested, the structure was accurately simulated.
The BTC database is *relational* and consists of several linked files. The database structure is designed to handle the full flight information supplied by the SABRE system, though not all possible details are included in this prototype. The problems are not trivial, e.g., the representation of the complex fare structure of individual seats, which depends on the date of travel, length of stay, ticket restrictions, etc. Another problem is to represent the seat plan such that not only aisle and window seats can be selected, but adjacent seats can also be determined.

The Rule Base

The complexity of human decision making is reflected by the *'scoring* scheme' in which rules are formulated to handle complex users' requirements and different weighting/importance levels. Simultaneous optimisation for the requirements can be achieved. The principle of the scoring scheme is to assign a score for each unique option based on two aspects: *'matchability' and 'weighting'.* Flights with the lowest overall score will be selected.

By 'matchability', it assesses how much the option can fit into the user's requirements. For example, if the customer wants to be at the destination before 1500, a flight arriving at 1400 will certainly has a lower score for the 'arrival time' than a flight reaching the airport at 1700. In BTC, the rule for this case can be illustrated as follows:

```
IF          preferred_arrival_type = before
AND         preferred_arrival_time > scheduled_arrival_time
THEN        time_score = (preferred_arrival_time - scheduled_arrival_time) x cost_scale)
```

The scores are worked out by calculating the discrepancies between the preferred arrival time and the scheduled arrival time plus an adjustment according to a 'cost scale' with £100 as the base fare. In the rule above, the cost scale is £2 per hour (i.e., £1/30 per minute) as the traveller can still manage to reach the destination before the preferred time, though a bit earlier than the exact hour he wants.

In another rule that handles the case when the flight is scheduled to arrive later than the preferred time, the scale is set at £5 per minute, because there would be much more inconvenience to the traveller.

```
IF          preferred_arrival_type = before
AND         preferred_arrival_time < scheduled_arrival_time
THEN        time_score = (scheduled_arrival_time - preferred_arrival_time) x cost_scale)
```

In an operational system, these cost scales will be predetermined by experts, who consider how much extra typical customers would pay for exactly what they prefer if the ticket cost £100. These costs are then adjusted in proportion to the actual fare. In more complex cases, cost scales can be determined by rules that take other circumstances into account. Since the scheduled arrival time is one of the fields in the database, the associated score is called a field score.

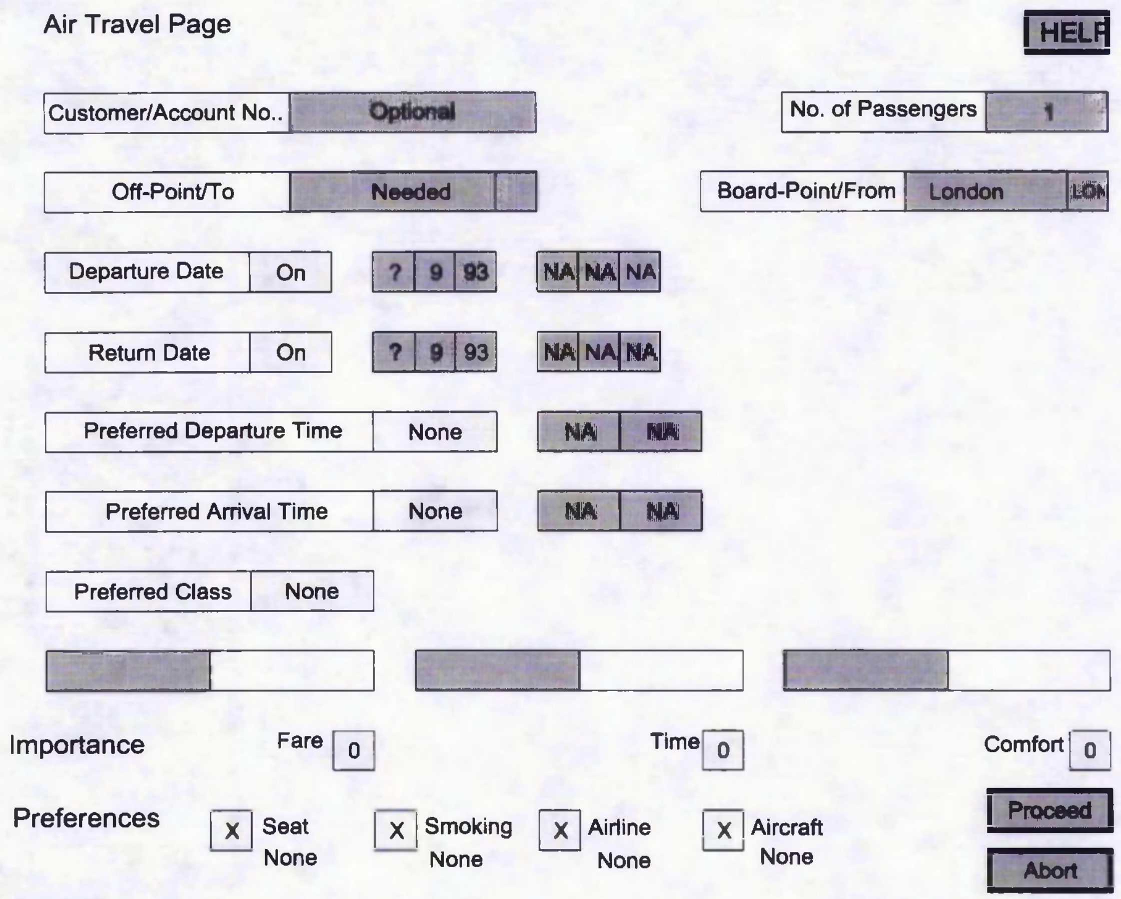
In terms of 'weighting', the importance that a particular traveller associates with a field is also considered as travellers will have individual preferences. The relative importance of the fields also depends on circumstances. For example, if the flight is short-haul, a smoking seat is much less important to a smoker. The arrival and departure times are less important in long-haul flights as absolute punctuality is not easy and the traveller will probably allow for some reasonable delay in such cases. The total score for each candidate flight is the sum of the products of individual field marks and their weights.

Though human experts do not work with numerical scores, they do consider 'matchability' and 'weighting' (Kattan, 1994). BTC attempts to emulate human experts except that the final assessment is quantitative. The advantages of using a structured scoring scheme are consistency, ease of averaging expert opinions and simplicity of software maintenance.

The User Interface

It is assumed that the users of BTC are travel agent staff (BTC operators) with little training on travel counselling. User acceptance depends on several aspects of consultation, including the number of questions that need to be answered, the order in which the questions are asked, whether the questions appear to be meaningful, and whether the user can answer the questions (Philip, 1993). The design principle for the user interface is: (1) to exploit the advantages of using an expert system to arrive at a user friendly interface; and (2) to design a user interface that exhibits expert behaviour in data collection. With careful considerations about the level of users' expertise, the graphical user interface is potentially suitable for direct access by customers.

BTC is designed to interact with users via an *intelligent form* to ease the task of data entry and to reduce errors. The use of a form allows information to be gathered in arbitrary sequences that does not hinder natural conversations with customers whilst operating BTC. The intelligent form is shown in Figure 2.



Air Travel Page

HELP

Customer/Account No..	Optional		No. of Passengers	1
Off-Point/To	Needed		Board-Point/From	London LON
Departure Date	On	? 9 93	NA NA NA	
Return Date	On	? 9 93	NA NA NA	
Preferred Departure Time	None	NA NA		
Preferred Arrival Time	None	NA NA		
Preferred Class	None			

Importance: Fare 0, Time 0, Comfort 0

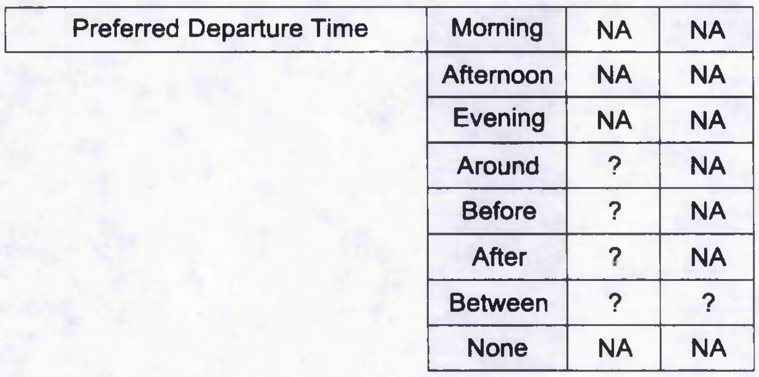
Preferences: X Seat None; X Smoking None; X Airline None; X Aircraft None

Proceed

Abort

Figure 2 The Intelligent Form

The form is divided into windows, each contains a number of fields in which the operator will enter the data. Key information such as date and time of travel are entered in the main window. These are the most likely details that customers will tell their consultant. The number of key fields is kept to a minimum as a consultant will not ask a lot of unnecessary questions. On the other hand, a large number of optional fields are present to satisfy customer preferences that are rarely specified, e.g., the type of aircraft. These optional fields are placed in a number of auxiliary windows that are only displayed when their corresponding buttons are selected on the main window.
Context sensitive fields are implemented to speed up form entry, which is illustrated as shown for the preferred departure time:



Preferred Departure Time	Morning	NA	NA
	Afternoon	NA	NA
	Evening	NA	NA
	Around	?	NA
	Before	?	NA
	After	?	NA
	Between	?	?
	None	NA	NA

Figure 3 Context Sensitive Fields

If the customer wants to depart around a specific time, the BTC operator then point and click at the question mark before entering the time. If the highlighted word **Around** is selected, a *pop-down menu appears which provide choices such as* **Morning and Between.** The time field is dynamic, i.e., if Morning is selected, the field will disappear immediately, whilst an additional question mark will appear if Between is selected. This design avoids large number of unnecessary fields and labels obscuring the screen. Also, *default values* suggesting highly likely answers are supplied if possible and they may change dynamically. For example, the current month and year are the defaults for all dates, and the default month/year of the return trip are those of the outward trip.

Whenever a field changes value, special rules associated with the field are triggered. These rules are used to check likely mistakes in the data and, more importantly, the likely conflicts amongst fields. Rules on individual fields are mostly range checks to discover mistakes such as trips to the past. More complex rules examine the interrelations amongst fields, e.g., if both preferred arrival and departure times are entered, the initial entry will be cleared since the fixed flight time will not allow both fields to be specified arbitrarily.

The relative importance, or weights, of the fields are determined by rules as explained above. Customers are unaware of individual weights, but three importance levels, **Fare, Time and Comfort** are provided to allow customisation. The default values of these importance levels are one, i.e., the three criteria are equally important to the customer. The importance levels can be changed by direct data entry, or by sliding horizontal gauges. Individual field weights are then derived from the importance levels by a set of rules. For example, if Time is greater than one, the schedule of the flight that matches the customer's requirement is given higher priority.

If there are no reported errors and all mandatory fields are completed, the user can leave the intelligent form and enter the consultation screen. The operator is presented with a split screen, on which the customers' requirements are shown alongside the recommended flights so that it is easy to check if the requirements are handled satisfactorily. If the customer is satisfied, the operator then starts the computerised booking procedures, otherwise, BTC can be aborted or the intelligent form can be restarted and modified. Finally, extra information is displayed in boxes containing ticket restrictions and destination information such as visa requirements.

Context sensitive help information is provided at all times implemented by hypertext tools of the expert system shell. Help menus relevant to the current user action are located in pop-down menus, error message boxes and all windows.

4 Future Work

Recent market analyses show a common and coherent trend — decrease in the importance of standardised travel packages and inclusive tour (Bennett, 1993; Hitchins, 1991). With travel information more and more readily available, there will be a trending towards *independent travelling*, i.e., the customers will mix and match travel products for their self-planned holidays instead of going to the tour operators for a pre-arranged package. Coupled with this trend is the changing role of the travel agents, which evolves from providing convenient booking to help giving customised advice as valued added services (Bruce, 1994). Therefore, research is currently continuing into the development of a more comprehensive expert system, using Prolog, to cater for both the business and independent travellers (Ng and Sussmann, 1994).

The future of expert systems depends on the cracking of the so-called knowledge acquisition bottleneck. The knowledge acquisition bottleneck limits the scalability of expert systems. While it is relatively straightforward to populate a small-scale knowledge base, it becomes more difficult to maintain consistency and validity as the knowledge base grows. Thus, it is important to automate the knowledge acquisition process (Rubin, 1993). An expert system in no way mimics human intelligence — unless it is adaptive — unless it learns. Learning is perceived as an integral part of expert systems in the future (by Kick in Rubin, 1993). Intelligence must be viewed as an evolutionary adaptation (by Fogel in Rubin, 1993). Therefore, the next version of BTC will address the 'learning' issues of knowledge elicitation.

The use of *multi-media* has also showed its importance as a competitive tool (Sheldon, 1994). We are sure to expect an increasing number of multi-media booking and information systems emerging in the market, such as SABREvision which can show coloured photos of hotel exteriors and interiors, deck and cabin of cruise ships (SABRE, 1993), the Thomas Cook Travel Kiosk at their Marble Arch branch which shows still images of destinations (CD ROM User, 1994), etc. With the audio-visual output of product information, the system is able to appeal to the psychological needs of the customers, making the products more tangible by allowing customers to actually 'feel' the products on the screen. An expert system with direct customer involvement through multi-media is going to be as important as the knowledge it encapsulates.

References

Bennett, M. M., (1993), Information Technology and Travel Agency: A Customer Service Perspective, *Tourism Management*, August, pp. 259-266.

Bruce, M., (1994), Technological Change and Competitive Marketing Strategies, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 430-433.

Burke, J. F., (1986), Computerized Management of Tourism Marketing Information, *Tourism Management,* December, pp. 279-289.

CD ROM User, (1994), The Information Highway: Virtual Holidays, July 1994.

Chadha, S. R., Mazlack, L. J. and Pick, R. A., (1001), Using Existing Knowledge Sources (cases) to Build an Expert System, *Expert Systems,* November, 8(4), pp. 3-12.

El-Najdawi, M. K. and Stylianou, A. C., (1993), Expert Support Systems: Integrating AI Technologies, *Communications of the ACM,* 36(12), December, pp. 55-65.

Goeldner, C. R., (1994), Tourism Information Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 178-182.

Hart, A., (1991), Report of Workshop on the Use of Knowledge Acquisition Techniques with Structured Systems Methods, Held at BP, London, 5 December 1990, *Expert Systems,* November, 8(4), pp. 38-40.

Hitchins, F., (1991), The influence of technology on UK Travel Agents, *Travel & Tourism Analyst,* No. 3, pp. 88-105.

Hruschka, H. and Mazanec, J., (1990), Computer-Assisted Travel Counselling, *Annals of Tourism Research,* Volume 17, pp. 208-227.

Kattan, M., (1994), Inductive Expert System vs. Human Experts, *AI Expert,* April, pp. 32-38.

Kim, J. and Courtney, J. M., (1988), A Survey of Knowledge Acquisition Techniques and their Relevance to Managerial Problem Domains, *Decision Support Systems,* 4, pp. 269-284.

McMullen, M., (1987), Artificial Intelligence in the Airline Industry, *IATA Review,* April/June, 1987, pp. 16-18.

Ng, F. and Sussmann, S., (1994), The Expert Business Travel Counsellor, *Tourism: The State of the Art, University of Strathclyde,* Glasgow, Scotland, 11-14 July, 1994.

Philip, G. C., (1993), Guidelines on Improving the Maintainability and Consultation of Rule-Based Expert Systems, *Expert Systems with Applications,* Volume 6, pp. 169-179.

Rubin, S. H., (1993), Machine Learning and Expert Systems, *AI Expert,* June, pp. 32-37.

SABRE Travel Information Network, (1993), *SABREvision,* 7 February 1993.

Sheldon. P. J., (1992), Researching Consumer Information: Destination Information Systems: Examples of Electronic Market Places, *Proceedings of the 23rd Annual Conference, Travel and Tourism Research Association,* Minneapolis, MN, June 14-17, pp. 37-49.

Sheldon, P. J., (1994), Information Technology and Computer Reservation Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 126-130.

Sussmann, S. and Ng, F., (1994), A Prototype Expert System for Business Travel Counselling, *Annuals of Tourism Research* (forthcoming).

Business Travel Counseling

Silvia Sussmann &Faria Y. Y. Ng
University of Surrey, U. K.

Expert systems are the most successful application of 'artificial intelligence' (AI), an area of information technology concerned with systems that attempt to function as human brains (McCool, 1987). AI has generally been applied to situations involving complex problem solution, reasoning, perception and planning. The success of expert system applications is based on the fact that they concentrate in the solution of problems that are complex but within a tightly restricted domain. A good conceptual definition of an Expert System is a 'computer system that uses a representation of human expertise in a specialist domain, in order to perform functions similar to those normally performed by a human expert' (Goodall, 1985).

The restricted domain addressed here is that of business travel counseling, an area where there is an ever-growing interest in providing effective competitive solutions. In practical terms, expert systems (ES) are computer programs that make - or suggest - decisions based on a set of rules and facts, the ES knowledge base. The most arduous part of any expert system development is the capture of this knowledge, generally carried out by interviewing and/or observing groups of experts in the particular field. Once the knowledge has been elicited from the experts, it becomes permanent. In the specific instance of travel counseling, the private knowledge of an experienced consultant can be transferred to inexperienced operators who feel that there is always an expert around to help. The 'facts', such as timetables, restrictions, availability, and requirements, will of course be in constant dynamic change, and so will the criteria by which they are selected, but the methodology that guides the selection of both facts and criteria should have been encapsulated in the expert system.

There are several different methodologies for addressing this critical part of the expert system development. Hruschka and Mazanec (1990) adopted a 'brainstorming session' technique with a group of travel counselors, where they attempted to elicit the property of a trip. In the area of business travel counseling, this concept is clearly defined, and most of the expertise lies in the choices surrounding the predefined trip, and the relative weight attached to each of these choices. Therefore, a modified form of 'protocol analysis' - where the experts are recorded 'thinking aloud' (Shah, 1994) - was chosen. Field research was conducted in one of the business travel management centers of a large UK tour operator, Thomas Cook, where several days were spent listening and recording the telephone counseling sessions of a team of expert business counselors. Some of them were later interviewed regarding their performance criteria and standards.

From this field research, it was possible to extract not only the knowledge needed to formulate the rules for the expert system, but also a benchmark against which to compare the prototype once developed. The rules were extracted from an analysis of the prominent attributes of the client requirements, such as minimum time spent on board, or lowest price for comparable service, and of the hierarchy of their preferences, which helped in establishing their relative weighting. The expertise of the travel counselors was then translated into the 'scoring scheme', which determines the final advice.

The prototype was constructed using a simple commercial 'expert system shell' - that is 'a constrained programming environment designed to help the development of expert systems' (Harmon, Maus and Morrisey, 1988). This appeared to be the fastest and most cost effective method of developing an initial prototype, aimed at testing the validity of the expert business travel counselor. The expert system shell provides the inference engine (mechanism for deriving conclusions from knowledge-based rules) and the user interface (framework for interaction with the expert system user) and is in turn connected to both the knowledge base for the expert rules and the database for the factual data.

As already noted, the rules constituting the 'scoring scheme' were formulated through the analysis of counseling sessions at the Thomas Cook Travel Management Center.

The structure of the database is modeled on the type of output produced by SABRE, which was examined and evaluated in detail as part of the field research. The concept tested was that of a relational database which automatically interfaces with the CRS output, and feeds real time flight and accommodation facts to the expert system.

The user interface was designed as a smart form. This involved a data collection form exhibiting expert behavior, such as fields automatically filled with the most likely answer, weight scoring offered as a horizontal gauge that can be displaced (with the form calculating the exact numeric equivalence), and some use of hypertext to compress the presentation to a single screen form. The user of the system completes the form with the flight and/or accommodation requirements, and is presented with a new split screen where his requirements are shown alongside the expert system recommendation. At this point, if the recommendation is accepted, a real-time reservation could be made, or a new alternative sought with additional requirements. Later, the prototype was validated against benchmark of real expert responses created during field research, and tuned until the responses were found to match, following a well-established expert system development procedure (Harmon, Maus and Morrisey, 1988).

In short, a prototype was constructed to test the feasibility of designing a business travel counselor expert system, with a direct 'real time' interface to CRSs like SABRE or GALILEO. The effective real time connection was not tested, but the structure was accurately simulated, based on field research conducted in one of the largest corporate travel centers in the United Kingdom. The rules incorporated in the expert system were inferred from recorded information of similar consultations in the same center, and validated against the experts' responses. The concept proved feasible and efficient, within the constraints of a small scale prototype.

Research is currently continuing into a comprehensive counseling system, to be developed using a specialist language like PROLOG, and integrating external databases, or even destination databases (Sheldon, 1993; Sussmann, 1992). Another area of interest is the incorporation of multimedia facilities (Sussmann, 1994), to make the product more tangible to end-users.

References

Goodall, A., (1985), *The Guide to Expert Systems,* Oxford: Learned Information.

Harmon, P., Maus, R. and Morrisey, W., (1988), *Expert System Tools and Applications,* New York: Wiley.

Hruschka, H and Mazanec, J., (1990), Computer Assisted Travel Counseling, *Annals of Tourism Research, 17*, pp. 208-227.

Martin, J. and Oxman, S., (1988), *Building Expert Systems: A Tutorial,* London: Prentice Hall.

McCool, A. C., (1987), Some Considerations in Building Expert Systems for the Hospitality Industry, *International Journal of Hospitality Management,* 6, pp. 209-215.

Shah, M., (1994), Knowledge Elicitation, *Computing,* March 17, p.35.

Sheldon, P., (1993), Destination Information Systems, *Annals of Tourism Research,* 20, pp. 633-649.

Sussmann, S., (1992), Destination Management Systems: The Challenge of the 1990s, in Cooper, C. (ed.), *Progress in Tourism, Recreation and Hospitality Management,* pp. 209-215.

Sussmann, S., (1994), The Impact of New Technological Developments on Destination Management, in Cooper, C. and Lockwood, A. (eds.), *Progress in Tourism, Recreation and Hospitality Management,* pp. 289-296.

The Expert Travel Counselling System — The Next Stage

Faria Ng and Silvia Sussmann

1 Introduction

Expert Systems are computer programs that embody the knowledge of one or more human experts in a narrow problem domain and can solve problems in that domain matching the expert's level of performance (El-Najdawi and Stylianou, 1993). Capturing expertise in the form of an expert system gives permanency to this expertise, it attains 'corporate immortality' (McMullen, 1987). This article describes the construction of an expert system prototype — **B**usiness **T**ravel **C**ounsellor (BTC). It is aimed to provide assistance to travel agency staff in matching the available products with the needs of business *trips. Since* expert systems can be duplicated, it is possible for inexperienced travel consultants to gain specialised assistance without having to deal with the time and location limitations that restrict human experts (Hruschka and Mazanec, 1990). That is, an ubiquitous expert is always available.

A powerful concept encapsulated in BTC is the integration of expert systems and **C**entral **R**eservation **S**ystems (CRSs) in the airline industry. Such kind of integration has been recognised as an important functional element in future global distribution systems (Goeldner, 1994; Sheldon, 1992). It is also capable of generating a market profile on products as well as customers (Burke, 1986). Travel agents can, therefore, focus their follow-up marketing efforts on likely prospects, by which inquirers are more likely to turn into satisfied customers.

2 Knowledge Acquisition

Complex knowledge acquisition cannot be achieved by a single method; a 'mix and match' technique (Hart, 1991), protocol analysis plus interviews, was therefore employed. Knowledge was extracted with assistance from experienced travel consultants, at a travel management centre of one of the large multiples.

2.1 Protocol Analysis

Protocol analysis involves a knowledge engineer observing how a domain expert solves problems. The knowledge engineer may directly query the expert. Alternatively, the knowledge engineer may passively observe the expert 'speaking out loud' while working through a problem (Chadha et al., 1991; Kim and Courtney, 1988, Sussmann and Ng, 1995). In the case of BTC, multiple counselling sessions of a team of expert travel counsellors were recorded. From the actual dialogue between the clients and the consultants, it was possible to extract not only the knowledge needed for the expert system, but also a benchmark against which to compare the prototype once developed. The observations were supplemented by an analysis of transcripts produced from recorded notes of the 'speaking out loud' problem sessions. Interference with the expert consultants' decision-making process was thus minimised, by avoiding interruptions through questions.

2.2 Interviews

Follow-up interviews were conducted to complement the results from the protocol analysis. The experts were asked why they made specific decisions. Sometimes, they had formed such a close relationship with their clients that they could apply what they already knew from the customers' profiles, e.g., a customer prefers to fly with a particular airline every time, or a corporation will have price as its first priority when they book flights for its staff. Therefore, the consultants were asked to verbalise the thoughts behind their decisions.

3 Prototype Construction

BTC was constructed using a simple commercial expert system shell. The shell provides the inference engine and tools for building graphical user interfaces. The inference engine derives conclusions from both a rule base and a database. The rule base contains heuristic knowledge collected from the field research concerning how travel consultants make decisions. The database contains information which simulates the input from a CRS, in response to actions triggered by the inference engine. The user interface collects users' requirements such as date, time, class of travel, etc., and displays details of the recommended flights together with other information useful to the trip. The custom components of BTC are described in the following sections. The architecture of BTC is shown in Figure 1.

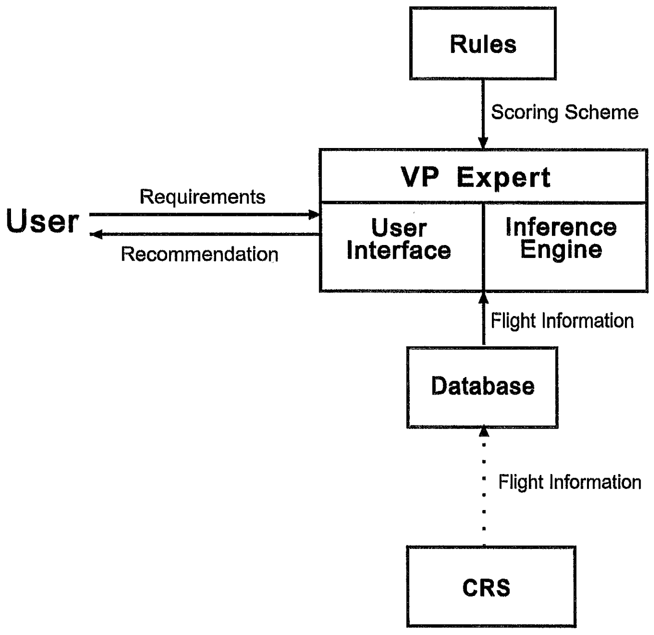


Figure 1 The Functional Architecture of BTC

3.1 The Database

After accepting the user's requirements, BTC will start processing them and send commands to the simulated CRS system calling for relevant information, typically a list of suitable flights. The familiar perception of a CRS system involves a travel consultant reading information from a terminal. For BTC, the information extracted from the CRS is assumed to be arranged in a machine-readable format that can be further analysed, i.e., a database. This takes place internally, in a form which is totally transparent to the user. Though the effective real time connection was not tested, the structure was accurately simulated, based on real CRS output collected in the field work.

The BTC database is relational and consists of several linked files. The database structure is designed to handle the full flight information supplied by the SABRE system, however, this initial prototype does not include all possible details. The problems involved in the SABRE emulation are not trivial, e.g., the representation of the complex fare structure of individual seats, which depends on the date of travel, length of stay, ticket restrictions, etc., or the representation of the seat plan which allows not merely the selection of aisle and window seats, but also the identification of adjacent seats.

3.2 The Rule Base

The complexity of human decision making is reflected by the 'scoring scheme' in which rules are formulated to handle complex users' requirements and different weighting/importance levels. Simultaneous optimisation for all the requirements can be achieved. The principle of the scoring scheme is to assign a score for each unique option based on two aspects: *'matchability' and 'weighting'.* Flights with the lowest overall score will be selected.

The 'matchability' criterion assesses how much the option can fit into the user's requirements. For example, if the customer wants to be at a destination before 1500, a flight arriving at 1400 will certainly have a lower score for the 'arrival time' than a flight reaching the airport at 1700. In BTC, the rule for this case can be illustrated as follows:

```
IF              preferred_arrival_type = before
AND             preferred_arrival_time > scheduled_arrival_time
THEN            time_score = (preferred_arrival_time - scheduled_arrival_time) x cost_scale)
```

The above means that the scores are worked out by calculating the discrepancies between the preferred arrival time and the scheduled arrival time plus an adjustment according to a 'cost scale' with £100 as the base fare. In the rule above, the cost scale is £2 per hour as the traveller can still manage to reach the destination before the preferred time, though a bit earlier than the exact hour he wants.

In another rule that handles the case when the flight is scheduled to arrive later than the preferred time, the scale is set at £5 per minute, because there would be much more inconvenience to the traveller.

```
IF              preferred_arrival_type = before
AND             preferred_arrival_time < scheduled_arrival_time
THEN            time_score = (scheduled_arrival_time - preferred_arrival_time) x cost_scale)
```

In an operational system, these cost scales will be predetermined by experts, who consider how much extra typical customers would pay for exactly what they prefer, by each £100 of the ticket cost. These costs are then adjusted in proportion to the actual fare. In more complex cases, cost scales can be determined by rules that take other circumstances into account. Since the scheduled arrival time is one of the fields in the database, the associated score is called a field score.

In terms of 'weighting', the importance that a particular traveller associates with a field is also considered as travellers will have individual preferences. The relative importance of the fields also depends on circumstances. For example, if the flight is short-haul, a smoking seat is much less important to a smoker. The arrival and departure times are less important in long-haul flights as absolute punctuality is not easy and the traveller will probably allow for some reasonable delay in such cases. The total score for each candidate flight is the sum of the products of individual field marks and their weights.

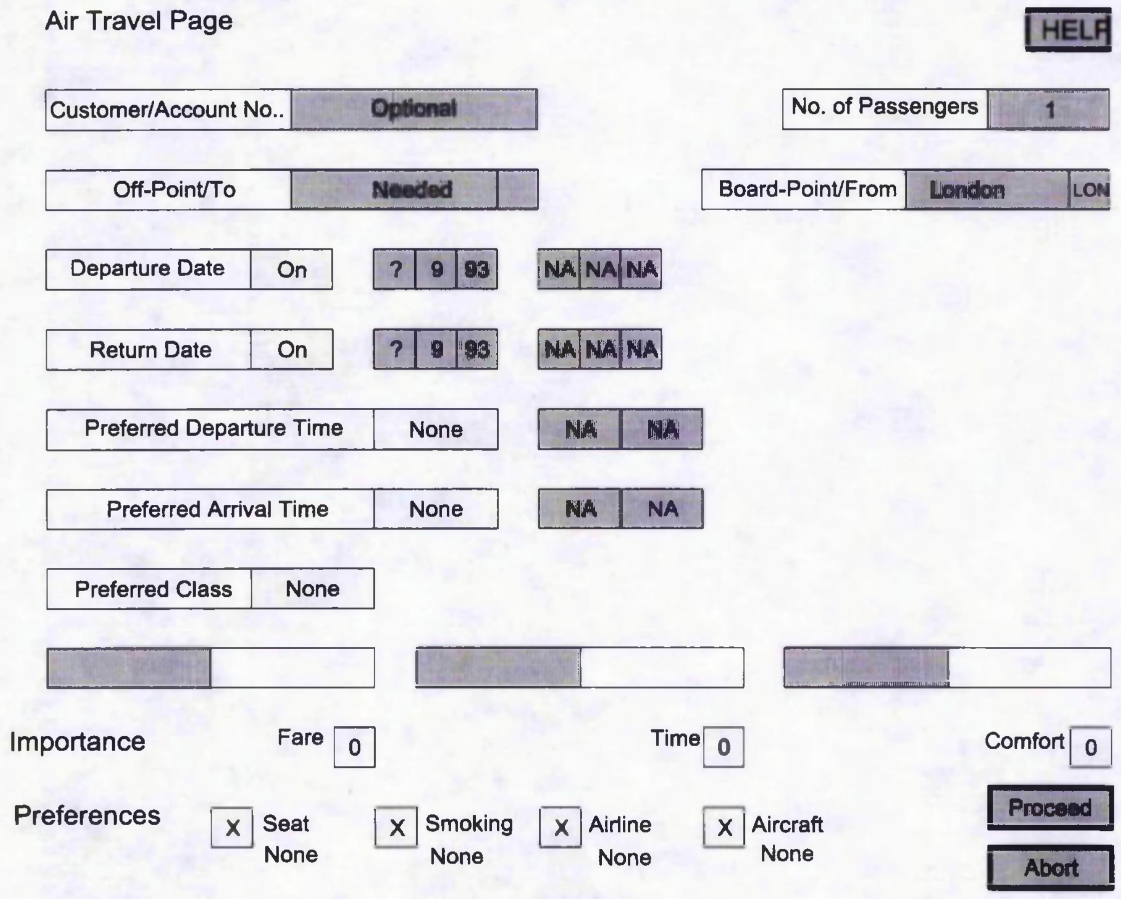
Though human experts do not work with numerical scores, they do consider 'matchability' and 'weighting' (Kattan, 1994; Sypniewski, 1994). BTC attempts to emulate human experts except that the final assessment is quantitative. The advantages of using a structured scoring scheme are consistency, ease of averaging expert opinions and simplicity of software maintenance.

3.3 The User Interface

It is assumed that the users of BTC are travel agent staff (BTC operators) with little training on travel counselling. User acceptance depends on several aspects of consultation, including the number of questions that need to be answered, the order in which the questions are asked, whether the questions appear to be meaningful, and whether the user can answer the questions (Philip, 1993). The design principle for the user interface is: (1) to exploit the advantages of using an expert system to arrive at a user friendly interface; and

(2) to design a user interface that exhibits expert behaviour in data collection. These design concepts integrate user interface and expert system design. It can improve expert system robustness and usability. Usability is improved by designs based on task-relevant information requirements. Robustness is improved by designs with support for user supervision and recovery (Malin and Schreckenghost, 1993). With careful considerations about the level of users' expertise, the graphical user interface is potentially suitable for direct access by customers.

BTC is designed to interact with users via an intelligent form to ease the task of data entry and to reduce errors. The use of a form allows information to be gathered in arbitrary sequences, without hindering natural conversation with customers, whilst operating BTC. The intelligent form is shown in Figure 2.



Air Travel Page

HELP

Customer/Account No..	Optional	No. of Passengers	1
Off-Point/To	Needed	Board-Point/From	London LON
Departure Date	On	? 9 93	NA NA NA
Return Date	On	? 9 93	NA NA NA
Preferred Departure Time	None	NA NA	
Preferred Arrival Time	None	NA NA	
Preferred Class	None		

Importance: Fare 0, Time 0, Comfort 0

Preferences: X Seat None; X Smoking None; X Airline None; X Aircraft None

Proceed

Abort

Figure 2 The Intelligent Form

The form is divided into windows, each containing a number of fields for the operator to enter the data. Key information such as date and time of travel are entered in the main window. These are the details that customers are most likely to tell their consultant. The number of key fields is kept to a minimum, since an experienced consultant will not ask unnecessary questions. On the other hand, a large number of optional fields are present to satisfy customer preferences that are only occasionally specified, e.g., the type of aircraft. These optional fields are placed in a number of auxiliary windows that are only displayed when their corresponding buttons are selected on the main window.

Context sensitive fields are implemented to speed up form entry, which is illustrated as shown for the preferred departure time:

Preferred Departure Time	Morning	NA	NA
	Afternoon	NA	NA
	Evening	NA	NA
	Around	?	NA
	Before	?	NA
	After	?	NA
	Between	?	?
	None	NA	NA

Figure 3 Context Sensitive Fields

If the customer wants to depart around a specific time, the BTC operator then point and click at the question mark before entering the time. If the highlighted word **Around** is selected, a *pull-down menu* appears which provide choices such as **Morning and Between**. The time field is dynamic, i.e., if Morning is selected, the field will disappear immediately, whilst an additional question mark will appear if Between is selected. This design avoids large number of unnecessary fields and labels obscuring the screen. Also, default values suggesting likely answers are supplied if possible and they may change dynamically. For example, the current month and year are the defaults for all dates, and the default month/year of the return trip are those of the outward trip.

Whenever a field changes value, special rules associated with the field are triggered. These rules are used to check likely data errors and, more importantly, potential conflicts amongst fields. Rules on individual fields are mostly range checks to discover mistakes such as trips to the past. More complex rules examine the interrelations amongst fields, e.g., if both preferred arrival and departure times are entered, the initial entry will be cleared since the fixed flight time will not allow both fields to be specified arbitrarily.

The relative importance levels, or weights, of the fields are determined by rules as explained above. Customers are unaware of individual weights, but three importance levels, **Fare, Time and Comfort** are provided to allow customisation. The default values of these importance levels are one, i.e., the three criteria are equally important to the customer. The importance levels can be changed by direct data entry, or by sliding horizontal gauges. Individual field weights are then derived from the importance levels by a set of rules. For example, if Time is greater than one, the schedule of the flight that matches the customer's requirement is given higher priority.

If there are no reported errors and all mandatory fields are completed, the user can leave the intelligent form and enter the consultation screen. The operator is presented with a split screen, on which the customers' requirements are shown alongside the recommended flights so that it is easy to check if the requirements were handled satisfactorily. If the customer is satisfied, the operator should then start the computerised booking procedures, otherwise, BTC can be aborted or the intelligent form can be restarted and modified. Finally, extra information is displayed in boxes containing ticket restrictions and destination information such as visa requirements.

Context sensitive help information is provided at all times implemented by hypertext tools of the expert system shell. Help menus relevant to the current user action are located in pop-up menus, error message boxes and all windows.

4 Future Research

Recent market analyses show a common and coherent trend: decrease in standardised travel packages and inclusive tours (Bennett, 1993; Hitchins, 1991). With travel information more and more readily available, the trends appear to point towards *independent travelling*, i.e., the customers will mix and match travel products for their self-planned holidays instead of going to the tour operators for a pre-arranged package. Coupled with this trend is the changing role of the travel agents, which could evolve from providing convenient booking to giving customised advice as a valued added service (Bruce, 1994). Therefore, research is currently continuing into the development of a more comprehensive expert system, based on the artificial intelligent language Prolog, to cater for both the business and independent travellers (Ng and Sussmann, 1994).

The future of expert systems depends on finding a solution to the so-called knowledge acquisition bottleneck. The knowledge acquisition bottleneck limits the scalability of expert systems. While it is relatively straightforward to populate a small-scale knowledge base, it becomes more difficult to maintain consistency and validity as the knowledge base grows. Thus, it is important to automate the knowledge acquisition process (Rubin, 1993). An expert system does not mimic human intelligence unless it is adaptive, that is: *it learns.* Learning is perceived as an integral part of expert systems in the future. Intelligence must be viewed as an evolutionary adaptation (Rubin, 1993). Therefore, the next version of BTC will address the 'learning' issues of knowledge elicitation.

The use of *multi-media* has also shown its importance as a competitive tool (Sheldon, 1994). We are sure to expect an increasing number of multi-media booking and information systems emerging in the market, such as SABREvision which can show coloured photos of hotel exteriors and interiors, deck and cabin of cruise ships (SABRE, 1993), or the Thomas Cook Travel Kiosk at their Marble Arch branch which shows still images of destinations (CD ROM User, 1994), etc. With the audio-visual output of product information, the system is able to appeal to the psychological needs of the customers, making the products more tangible by allowing customers to 'experience' them on the screen. An expert system with direct customer involvement through multi-media is going to be as important as the knowledge it encapsulates.

References

Bennett, M. M., (1993), Information Technology and Travel Agency: A Customer Service Perspective, *Tourism Management*, August, pp. 259-266.

Bruce, M., (1994), Technological Change and Competitive Marketing Strategies, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 430-433.

Burke, J. F., (1986), Computerized Management of Tourism Marketing Information, *Tourism Management,* December, pp. 279-289.

CD ROM User, (1994), The Information Highway: Virtual Holidays, July 1994.

Chadha, S. R., Mazlack, L. J. and Pick, R. A., (1001), Using Existing Knowledge Sources (cases) to Build an Expert System, *Expert Systems*, November, 8(4), pp. 3-12.

El-Najdawi, M. K. and Stylianou, A. C., (1993), Expert Support Systems: Integrating AI Technologies, *Communications of the ACM,* 36(12), December, pp. 55-65.

Goeldner, C. R., (1994), Tourism Information Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 178-182.

Hart, A., (1991), Report of Workshop on the Use of Knowledge Acquisition Techniques with Structured Systems Methods, Held at BP, London, 5 December 1990, *Expert Systems,* November, 8(4), pp. 38-40.

Hitchins, F., (1991), The influence of technology on UK Travel Agents, *Travel & Tourism Analyst,* No. 3, pp. 88-105.

Hruschka, H. and Mazanec, J., (1990), Computer-Assisted Travel Counselling, *Annals of Tourism Research,* Volume 17, pp. 208-227.

Kattan, M., (1994), Inductive Expert System vs. Human Experts, *AI Expert,* April, pp. 32-38.

Kim, J. and Courtney, J. M., (1988), A Survey of Knowledge Acquisition Techniques and their Relevance to Managerial Problem Domains, *Decision Support Systems,* 4, pp. 269-284.

Malin, J. T., (1993), User as Intelligent Agent: Interface Design for Intelligent Monitoring Systems, *Proceedings of the Ninth Conference on Artificial Intelligence for Applications,* IEEE Computer Society Press, p. 461.

McMullen, M., (1987), Artificial Intelligence in the Airline Industry, *IATA Review*, April/June, 1987, pp. 16-18.

Ng, F., (1995), Business Travellers' Expert Advisor, *Proceedings of the ENTER 95 International Conference on Information and Communications Technologies in Tourism,* Innsbruck, Tyrol, Austria, 18-20 January, 1995.

Ng, F. and Sussmann, S., (1994), The Expert Business Travel Counsellor, *Tourism: The State of the Art,* University of Strathclyde, Glasgow, Scotland, 11-14 July, 1994.

Philip, G. C., (1993), Guidelines on Improving the Maintainability and Consultation of Rule-Based Expert Systems, *Expert Systems with Applications,* Volume 6, pp. 169-179.

Rubin, S. H., (1993), Machine Learning and Expert Systems, *AI Expert,* June, pp. 32-37.

SABRE Travel Information Network, (1993), *SABREvision,* 7 February 1993.

Sheldon. P. J., (1992), Researching Consumer Information: Destination Information Systems: Examples of Electronic Market Places, *Proceedings of the 23rd Annual Conference, Travel and Tourism Research Association,* Minneapolis, MN, June 14-17, pp. 37-49.

Sheldon, P. J., (1994), Information Technology and Computer Reservation Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 126-130.

Sussmann, S. and Ng, F., (1995), A Prototype Expert System for Business Travel Counselling, *Annuals of Tourism Research* (forthcoming).

Sypniewski, B. P., (1994), The Importance of Being Data, *AI Expert,* November, 23-31.

Business travellers' expert advisor

Faria Y.Y. Ng

Department of Management Studies, University of Surrey

Guildford, Surrey, GU2 5XH, UK

Abstract. This paper describes the construction of an 'expert system' prototype — ***Business Travel*** **C**ounsellor (BTC) for business travel counselling. It is aimed to provide assistance to travel agent staff to match the available products with the needs of inquiring customers. Knowledge acquisition was done in consultation with experienced travel consultants to build the rule base. The complexity of human decision making is reflected by the *'scoring scheme'* in which rules are formulated to handle complex users' requirements and different weighting/importance levels. The structure of the SABRE system was analysed for the design of the database. Special emphasis was put on the feasibility of the *integration of expert systems and Central* ***R****eservation* ***S****ystems (CRSs)*. BTC serves to prove the concept of valuable applications of expert systems in travelling counselling. The aim is to provide efficient and reliable information to ensure a better customer-product match.

1 Introduction

Travel choices are relatively high-involvement decisions, and customers tend to engage in active information gathering (Moutinho, 1987). As the industry enters the mature market phase, competition for available tourists will increase. This mature market for the tourism industry will almost certainly usher in an era where price will no longer be sufficient to assure success. Rather, a strong reliance on 'expertise' to deliver individual services will become necessary for survival.

Expert Systems are computer programs that embody the knowledge of one or more human experts in a narrow problem domain and can solve problems in that domain matching the expert's level of performance (El-Najdawi and Stylianou, 1993). Capturing expertise in the form of an expert system gives permanence to the expertise and allows it to become ubiquitous, which is termed as 'attaining corporate immortality' by McMullen (McMullen, 1987). Since an expert system can easily be duplicated, it is possible for inexperienced travel consultants to gain specialised assistance without having to deal with the time and location limitations that restrict human experts (Hruschka and Mazanec, 1990). In other words, there will always be an expert around to help. The implementation of an expert system that makes sophisticated travel counselling possible for everybody and everywhere, therefore, represents an important opportunity for the travel agent industry (Crouch, 1991).

2 Problem identification

A field research was conducted in one of the travel management centres (TMCs) of Thomas Cook. The consultants are assisted by the SABRE Central Reservation System. They will look up the customer profile in SABRE once receive a call. Based on the information from the profile and the present requests of the customer, the consultants will send commands to SABRE to check the availability of the products that the customer wants. SABRE will then generate a list of alternatives. The consultants will advise on the option that is supposed to meet the requirements of the client most appropriately.

2.1 The performance gap

A *questionnaire analysis* on the characteristics of travel consultants showed that the level of experience of consultants varies significantly. Though a large proportion (70%) has two to five years' experience, the rest of them show a dimension ranging from less than one to more than 16 years. Therefore, the same customer with the same requirements may be recommended different products by different consultants. Moreover, human decisions are inconsistent by nature. The same inputs do not always produce the same decisions. This may be due to external distractions, forgetting to get key pieces of information and even changes in an individual's mood. Therefore, even the same consultant may have varied performance at different times. In other words, there is no guarantee that the customer really gets the most suitable product.

Based on a *performance review* carried out in the same TMC, the consultants were evaluated in terms of their efficiency and productivity.

Table 1 Details of the calls received by the observed team in the TMC

	No. of Calls	ABD*	ABD%	TSF%+
Monday	63	9	14.2	87
Tuesday	64	2	3.1	83
Wednesday	73	6	8.2	89
Thursday	57	2	3.5	83

* ABD is the number of calls abandoned.

+ TSF% is the amount of calls answered in 15 seconds after being put through

The consultants in this team were *interviewed*. It was found out that the team performed below average on Tuesday (83%) and Thursday (83%). The criterion for assessing performance is by the standard set by Thomas Cook, which is 85% of calls to be answered in 15 seconds for a day.

2.2 How expert systems fit in?

With the use of a *criteria checklist* (Liebowitz, 1989), it was found out that travel counselling meets all the major criteria as an appropriate narrow problem domain since the task requires strategic use of heuristics. As an expert system can encode the *pooled expertise of the experts into a computer, it will not only smooth individual variances, but also help to diminish the variances across decision-makers.* Moreover, an expert system can offer speedy 'customer-product' matching because it thinks like a human expert, applying rules in a non-procedural manner, jumping to quick hypotheses and offering speedy conclusions. By this, consultants will become more efficient and productive. The *ABD% can be reduced and TSF% increased as clients'* waiting time is minimised with the improvement in the speed in decision making.

3 The functional architecture of BTC

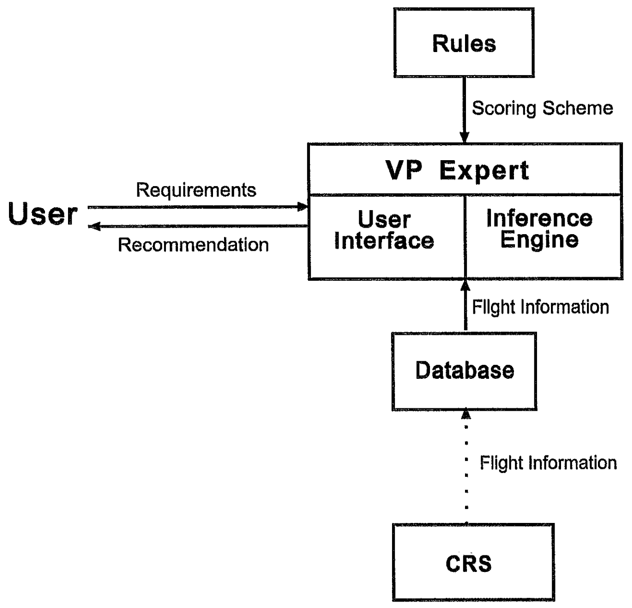


Figure 1 The functional architecture of BTC

BTC was constructed using a simple commercial expert system shell. The shell provides the inference engine and tools for building graphical user interfaces. The inference engine derives conclusions from the knowledge base that includes both a rule base and a database. The rule base contains heuristic knowledge collected from the field research concerning how travel consultants make decisions. The database contains information received from a CRS, in response to actions triggered by the inference engine. The user interface collects users' requirements such as date, time, class of travel, etc., and displays details of the recommended flights together with other information useful to the trip. The custom components of BTC will be explained in the following sections.

3.1 The database design

The BTC database structure is designed with the practical idea of interfacing to an external CRS. After accepting the user's requirements, BTC will start processing them and then sent commands to the CRS system to extract relevant information, typically a list of relevant flights. The information extracted is assumed to be rearranged in a *machine-readable format* that can be further analysed, i.e., a database. This, in fact, will take place internally without the user noticing at all. Though the real time connection was not tested, the structure was accurately simulated. Such kind of integration has been recognised as an important functional element in future global distribution systems (Goeldner, 1994; Martin and Oxman, 1988; Sheldon, 1992).

3.2 The analysis on SABRE

By entering the date, the destination pair and the preference for the departure time (if any), a list of flights available will be shown.

Table 2 SABRE — flight availability

```
1. NW 49 F4 C4 Y4 B4 M4 H4 Q0 V0    LGWBOS 1215 1435 747 DS 0
2. AA109 F7 C7 Y7 B7 M7 V7 X7 Q7 H7 LHRBOS 1130 1400 767 LS 0
3. VS 11 J4 W4 Y7 B7 L7 A7 S7       LGWBOS 1500 1710 747 LS 0
4. BA213 F7 J7 M7 S7 B7 K7 Q0 V0    LHRBOS 0955 1230 767 D  0
5. BA215 F7 J7 M7 S7 B7 K7 Q0 V0    LHRBOS 1645 1900 747 D  0
```

It shows information such as airline and flight no., e.g., NW49, the classes of seat available, e.g., F4 means four in first class, the departure and arrival airports, e.g., LGWBOS (London Gatwick and Boston), the departure and arrival (local) time, e.g., 1215, 1435, the type of flight, e.g., 747, meal services, e.g., DS (dinner and snack) and the number of stops, e.g., 0 means no stop.

Specific flight details such as the length and the total miles of the flight can be obtained. To deal with seat preferences, a seating plan showing the availability and particulars of seats can be displayed. Seats are shown relative to the position of toilets and the screen for movies. Information on fare bases can be requested to see if the customer is eligible for a particular fare base, e.g., youth and standard adult tickets. Each type of fare will impose different restrictions on the traveller.

3.3 The BTC database structure

The BTC database is *relational* and consists of four linked files, `namely SCHEDULE, CLASS, FARE, and SEAT.` The database structure is designed to handle full flight information supplied by SABRE, though not all possible details are included in this prototype. The problems are not trivial, e.g., the representation of the complex fare structure of individual seats, which depends on the date of travel, length of stay, ticket restrictions, etc. Another problem is to represent the seat plan such that not only aisle and window seats can be selected, but adjacent seats can also be determined.

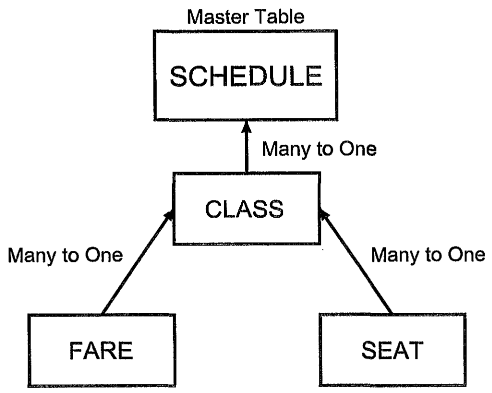


Figure 2 Structure of the relational database

General flight details are stored in SCHEDULE to which all three files are linked. Both FARE and SEAT are linked to CLASS. Their relationships are explained in Figure 2.

SCHEDULE shows flight availability. Basic flight information, such as Board and Off-Points, Airline and Flight No., Times of Departure and Arrival, No. of Stops and Equipment are stored. CLASS gives information on the types of class available for the flights. The field Availability gives information on the number of seats available for a particular class. FARE stores information on fare base. The fields Before Day and After Day denote the period within which a particular fare base may apply, i.e. season. SEAT shows the variety of seats (Seat Type), such as window, aisle, front and back seats. The field Vacant tells if a particular seat is available. The particulars of each seat are represented by specifying what lies on its right side (Seat No. on the Right). For example, if there is an aisle/a window on the right, it is an aisle/a window seat; if both sides are vacant seats, then the user may know that adjacent seats are available, and so on. Smoke distinguishes between smoking and non-smoking seats.

3.4 The rule base

Knowledge acquisition

As no single method will suffice for complex knowledge acquisition, a 'mix and match' method (Hart, 1991) , *protocol analysis plus interviews*, was employed.

Protocol analysis. Protocol analysis involves a knowledge engineer observing how a domain expert solves problems. The knowledge engineer may directly query the expert. Alternatively, the knowledge engineer may passively observe the expert 'speaking out loud' while working through a problem (Chadha et al., 1991; Kim and Courtney, 1988, Sussmann and Ng, 1994). In the case of BTC, multiple counselling sessions of a team of expert travel counsellors were recorded. From the actual dialogue between the clients and the consultants, it was possible to extract not only the knowledge needed for the expert system, but also a benchmark against which to compare the prototype once developed. The observations were supplemented by an analysis of transcripts produced from recorded notes of the 'speaking out loud' problem sessions. Interference with the expert consultants' decision-making process, which might be caused when asking questions, was minimised.

Interviews. Follow-up interviews were conducted to complement the results from the protocol analysis. The experts were asked why they make 'such and such' decisions. Sometimes, they have formed a close relationship with their clients that they simply apply what they know from the customers' profiles, e.g., a customer is a member of the frequent flyer program and prefers to fly with a particular airline every time, or a company will put price in the first place when they book flights for their staff. The consultants were asked to verbalise their immediate thoughts behind their decisions.

The 'scoring scheme'

The knowledge elicited are then translated into a 'scoring scheme'. The principle of the 'scoring scheme' is to assign an overall score for each unique option based on two aspects: **'matchability'** and **'weighting'.**

By 'matchability', it assesses how much the option can fit into the user's requirements, e.g., if the customer want to be at the destination before 1500, a flight arriving at 1400 will certainly has a lower mark for the 'arrival time' (fleld in database) than a flight reaching the airport at 1700.

```
RULE    arrive_before_b
IF      arrival_time_requirement = before
        and preferred_arrival_time > scheduled_arrival_time
THEN    time_score = ((preferred_arrival_time - scheduled_arrival_time) x cost scale)
```

The scores are worked out by calculating the discrepancies between the preferred arrival time and the scheduled arrival time plus an adjustment according to a *'cost scale'* with £100 as the base fare. These costs are then adjusted in proportion to the actual fare. For the flight arriving at 1400, the scale is £2 per hour as the traveller can still manage to reach the destination before the preferred time, though a bit earlier than the exact hour he wants. For the flight arriving at 1700, the scale will be £5 per minute, as late arrival would cause much more inconvenience to the traveller.

In terms of 'weighting', the importance that a particular traveller associates with a field is also considered as travellers will have individual preferences. The relative importance of the fields also depends on circumstances. For example, the arrival and departure times are less important in long-haul flights as absolute punctuality is not easy and the traveller will probably allow for some reasonable delay in such cases.

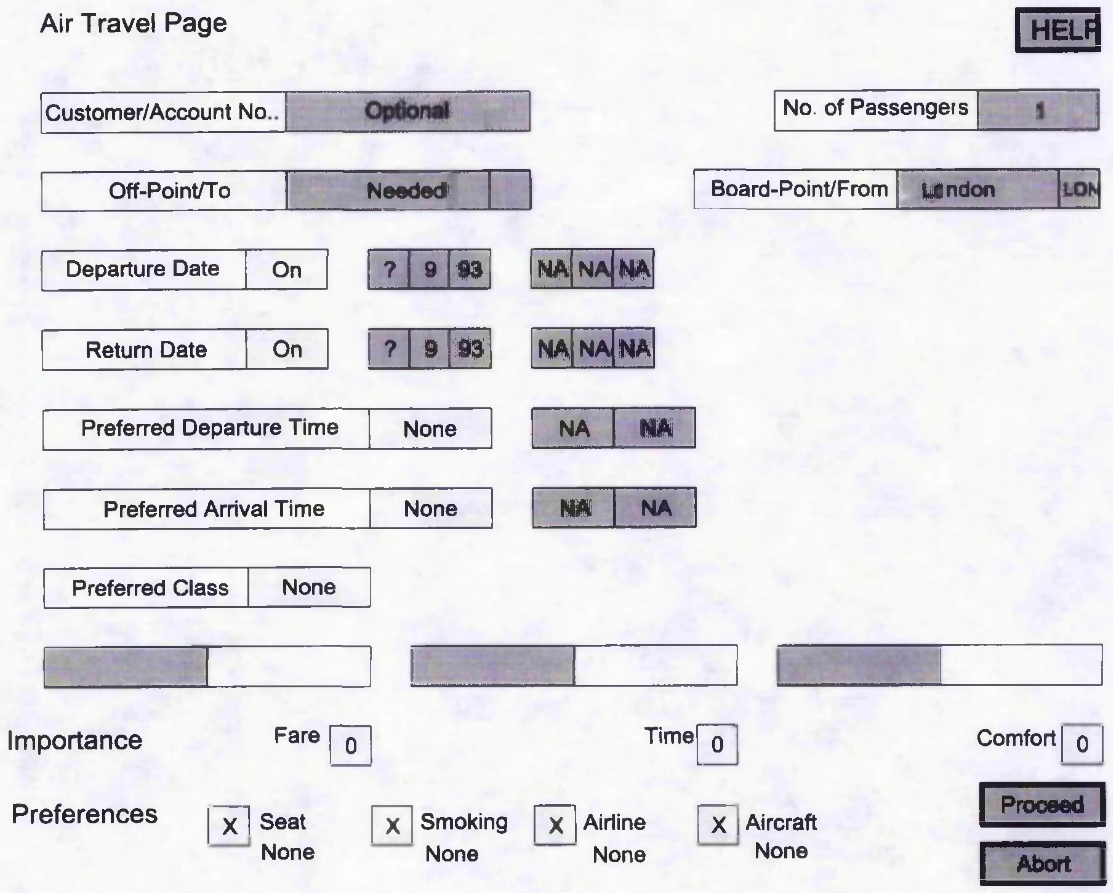
The total score for each candidate flight is the sum of the products of individual field marks and their weights. Though human experts do not work with numerical scores, they do consider 'matchability' and 'weighting' (Kattan, 1994). BTC attempts to emulate human experts except that the final assessment is quantitative. The advantages of using a structured scoring scheme are consistency, ease of averaging expert opinions and simplicity of software maintenance.

3.5 The user interface

The design principle for the user interface is: (1) to exploit the advantages of using an expert system to arrive at a user friendly interface; and (2) to design a user interface that exhibits expert behaviour in data collection.

The intelligent form

BTC is designed to interact with users via an *intelligent form* as shown in Figure 3. The use of a form allows information to be gathered in arbitrary sequences that do not hinder natural conversation with customers whilst operating BTC.



Air Travel Page HELP

Customer/Account No..	Optional	No. of Passengers	1
Off-Point/To	Needed	Board-Point/From	London LON
Departure Date	On	? 9 93	NA NA NA
Return Date	On	? 9 93	NA NA NA
Preferred Departure Time	None	NA NA	
Preferred Arrival Time	None	NA NA	
Preferred Class	None		

Importance Fare 0 Time 0 Comfort 0

Preferences X Seat None X Smoking None X Airline None X Aircraft None

Proceed Abort

Figure 3 The intelligent form

The form is divided into windows, each contains a number of fields in which the operator will enter the data. Key information such as date and time of travel, travel class, etc., are entered in the main window. These are the most likely details that customers will tell their consultant. The number of key fields is kept to a minimum as a consultant will not ask a lot of unnecessary questions. On the other hand, a large number of optional fields are provided to satisfy individual customer preferences that are occasionally specified, e.g., seat, smoking and airline preferences. These optional fields are placed in a number of auxiliary windows that are only displayed when their selection buttons are activated on the main window.

Context sensitive fields. Context sensitive fields are implemented to speed up form entry, which is illustrated as shown for the preferred departure time in Figure 4.

Preferred Departure Time	Morning	NA	NA
	Afternoon	NA	NA
	Evening	NA	NA
	Around	?	NA
	Before	?	NA
	After	?	NA
	Between	?	?
	None	NA	NA

Figure 4 Context sensitive fields

If the customer wants to depart around a specific time, the BTC operator then point and click at None to display the options. A pull-down menu appears which provide options such as Morning, Afternoon, Evening, Around, Before, After and Between. The time field is dynamic, i.e., if Morning is selected, the field will disappear immediately, whilst an additional question mark will appear for Around and two additional question marks for Between. This design avoids large number of unnecessary fields and labels obscuring the screen.

Defaults. Default values suggesting highly likely answers are supplied if possible and they may change dynamically. For example, the current month and year are the defaults for all dates, and the default month/year of the return trip are those of the outward trip.

Range and relationship checks. Whenever a field changes value, special rules associated with the field are triggered. These rules are used to check likely mistakes in the data and, more importantly, the likely conflicts amongst fields. Rules on individual fields are mostly range checks to guard against mistakes such as trips to the past. More complex rules examine the interrelations amongst fields, e.g., if both preferred arrival and departure times are entered, the initial entry will be cleared since the fixed flight (travelling) time will not allow both fields to be specified arbitrarily.

Horizontal gauge. The relative importance, or weights, of the fields are determined by rules as explained in section 5.2. Customers are unaware of individual weights, but three importance levels, *Fare, Time and Comfort* are provided to allow customisation. The default values of these importance levels are one, i.e., the three criteria are equally important to the customer. The importance levels can be changed by direct data entry, or by sliding horizontal gauges. Individual field weights are then derived from the importance levels by a set of rules. For example, if Time is greater than one, the schedule of the flight that matches the customer's requirement is given higher priority.

Auxiliary windows. The options for seat, smoking and airline preferences are provided in auxiliary windows, popping up on the blank space to the right hand side of the screen only if one of the buttons has been activated. An example on seat is shown in Figure 5.

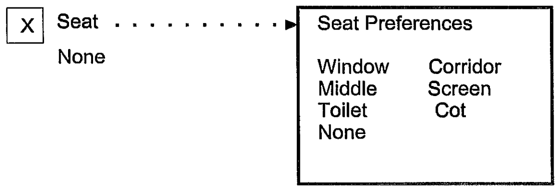
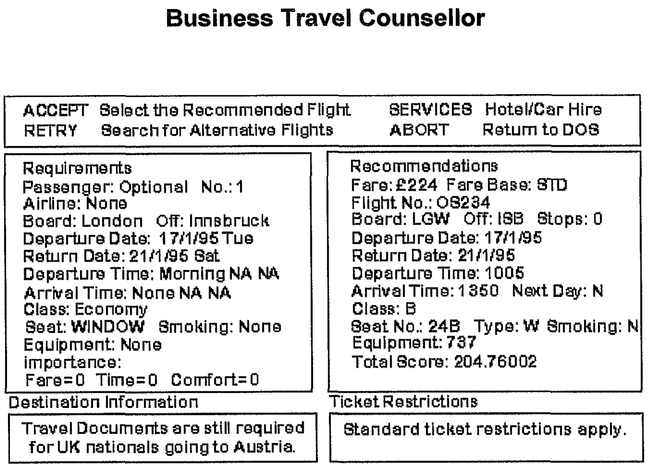


Figure 5 Auxiliary windows

Context sensitive HELP. Context sensitive help information is provided at all times implemented by hypertext tools of the expert system shell. Help menus relevant to the current user action are located in pop-up menus, error message boxes and all windows.

The consultation screen

If there are no reported errors and all mandatory fields are completed, the user can leave the intelligent form and enter the consultation screen. The operator is presented with a split screen, on which the customers' requirements are shown alongside the recommended flights so that it is easy to check if the requirements are handled satisfactorily. An example is shown in Figure 6.



Business Travel Counsellor

```
ACCEPT  Select the Recommended Flight      SERVICES  Hotel/Car Hire
RETRY   Search for Alternative Flights     ABORT     Return to DOS
```

```
Requirements                               Recommendations
Passenger: Optional  No.: 1                Fare: £224  Fare Base: STD
Airline: None                              Flight No.: OS234
Board: London  Off: Innsbruck              Board: LGW  Off: ISB  Stops: 0
Departure Date: 17/1/95 Tue                Departure Date: 17/1/95
Return Date: 21/1/95 Sat                   Return Date: 21/1/95
Departure Time: Morning NA NA              Departure Time: 1005
Arrival Time: None NA NA                   Arrival Time: 1350  Next Day: N
Class: Economy                             Class: B
Seat: WINDOW  Smoking: None                Seat No.: 24B  Type: W  Smoking: N
Equipment: None                            Equipment: 737
Importance:                                Total Score: 204.76002
Fare=0  Time=0  Comfort=0
Destination Information                    Ticket Restrictions
Travel Documents are still required        Standard ticket restrictions apply.
for UK nationals going to Austria.
```

Figure 6 The consultation screen

In this case, the customer wants to go to Innsbruck from a London airport on 17/1/95 and return on 21/1/95. The preferred departure time is morning and economy class is needed. The only personal preference is a window seat. The OS234 flight is recommended, which departs from London Gatwick to Innsbruck at 1005 and reaches the airport at 1350. A non-smoking window seat is allocated.

If the customer is satisfied, the operator then starts the computerised booking procedures, otherwise, BTC can be aborted or the intelligent form can be restarted and modified. Extra information is displayed in boxes highlighting ticket restrictions, destination information, visa requirements, etc.

4 Future work

Recent market analyses show a common and coherent trend — decrease in the importance of standardised travel packages and inclusive tour (Bennett, 1993; Hitchins, 1991). With travel information more and more readily available, there will be a trending towards *independent travelling*, i.e., the customers will mix and match travel products for their self-planned holidays instead of going to the tour operators for a pre-arranged package. Coupled with this trend is the changing role of the travel agents, which evolves from providing convenient booking to help giving customised advice as valued added services (Bruce, 1994). Therefore, research is currently continuing into the development of a more comprehensive expert system, using Prolog, to cater for both the business and independent travellers (Ng and Sussmann, 1994).

The future of expert systems depends on the cracking of the so-called knowledge acquisition bottleneck. The knowledge acquisition bottleneck limits the scalability of expert systems. While it is relatively straightforward to populate a small-scale knowledge base, it becomes more difficult to maintain consistency and validity as the knowledge base grows. Thus, it is important to automate the knowledge acquisition process (Rubin, 1993). An expert system in no way mimics human intelligence — unless it is adaptive — unless it learns. Learning is perceived as an integral part of expert systems in the future (by Kick in Rubin, 1993). Intelligence must be viewed as an evolutionary adaptation (by Fogel in Rubin, 1993). Therefore, the next version of BTC will address the 'learning' issues of knowledge elicitation.

The use of *multi-media* has also showed its importance as a competitive tool (Sheldon, 1994). We are sure to expect an increasing number of multi-media booking and information systems emerging in the market, such as SABREvision which can show coloured photos of hotel exteriors and interiors, deck and cabin of cruise ships (SABRE, 1993), the Thomas Cook Travel Kiosk at their Marble Arch branch that shows still images of destinations (CD ROM User, 1994), etc. Though BTC was designed to assist the travel agent staff, it has the capacity of involving the customers as direct end-users. With the audio-visual output of product information, the system is able to appeal to the psychological needs of the customers, making the products more tangible by allowing customers to actually 'feel' the products on the screen. An expert system with direct customer involvement through multi-media is going to be as important as the knowledge it encapsulates.

References

Bennett, M. M., (1993), Information Technology and Travel Agency: A Customer Service Perspective, *Tourism Management,* August, pp. 259-266.

Bruce, M., (1994), Technological Change and Competitive Marketing Strategies, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 430-433.

CD ROM User, (1994), The Information Highway: Virtual Holidays, July 1994.

Chadha, S. R., Mazlack, L. J. and Pick, R. A., (1991), Using Existing Knowledge Sources (cases) to Build an Expert System, *Expert Systems,* November, 8(4), pp. 3-12.

Crouch, G. I., (1991), Expert Computer Systems in Tourism: Emerging Possibilities, *Journal of Travel Research,* 29(3), pp. 3-10.

El-Najdawi, M. K. and Stylianou, A. C., (1993), Expert Support Systems: Integrating AI Technologies, *Communications of the ACM,* 36(12), December, pp. 55-65.

Goeldner, C. R., (1994), Tourism Information Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 178-182.

Hart, A., (1991), Report of Workshop on the Use of Knowledge Acquisition Techniques with Structured Systems Methods, Held at BP, London, 5 December 1990, *Expert Systems,* November, 8(4), pp. 38-40.

Hitchins, F., (1991), The influence of technology on UK Travel Agents, *Travel & Tourism Analyst,* No. 3, pp. 88-105.

Hruschka, H. and Mazanec, J., (1990), Computer-Assisted Travel Counselling, *Annals of Tourism Research,* Volume 17, pp. 208-227.

Kattan, M., (1994), Inductive Expert System vs. Human Experts, *AI Expert,* April, pp. 32-38.

Kim, J. and Courtney, J. M., (1988), A Survey of Knowledge Acquisition Techniques and their Relevance to Managerial Problem Domains, *Decision Support Systems,* 4, pp. 269-284.

Liebowitz, J., (1989), Problem Selection for Expert Systems, in Liebowitz, J. and De Salvo, D. A. (eds.), *Structuring Expert Systems, Domain, Design, and Development,* Yourdon Press Computing Series, Prentice Hall, Englewood Cliff, NJ, pp. 3-23.

Martin, J. and Oxman, S., (1988), *Building Expert Systems — A Tutorial,* London, Prentice-Hall International (U.K.) Limited.

McMullen, M., (1987), Artificial Intelligence in the Airline Industry, *IATA Review,* April/June, 1987, pp. 16-18.

Moutinho, L., (1987), Consumer Behaviour in Tourism, *European Journal of Marketing,* 21(10), pp. 1-44.

Ng, F. and Sussmann, S., (1994), The Expert Business Travel Counsellor, *Tourism: The State of the Art,* University of Strathclyde, Glasgow, Scotland, 11-14 July, 1994.

Rubin, S. H., (1993), Machine Learning and Expert Systems, *AI Expert,* June, pp. 32-37.

SABRE Travel Information Network, (1993), *SABREvision,* 7 February 1993.

Sheldon. P. J., (1992), Researching Consumer Information: Destination Information Systems: Examples of Electronic Market Places, *Proceedings of the 23rd Annual Conference, Travel and Tourism Research Association,* Minneapolis, MN, June 14-17, pp. 37-49.

Sheldon, P. J., (1994), Information Technology and Computer Reservation Systems, in Witt and Moutinho (eds.) *Tourism Marketing and Management Handbook,* Prentice Hall, Hemel Hempstead, pp. 126-130.

Sussmann, S. and Ng, F., (1994), A Prototype Expert System for Business Travel Counselling, *Annuals of Tourism Research* (forthcoming).