Agent based application service providing system

Li, Bing

2005

Li, B. (2005). Agent based application service providing system. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/4615>

<https://doi.org/10.32657/10356/4615>

Nanyang Technological University

Downloaded on 26 Apr 2023 17:13:01 SGT

An Agent Based Application Service Providing System

Li Bing

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005

Acknowledgements

I would like to express my gratitude to my supervisors, Assistant Professor Miao Yuan and Associate Professor Yang Zhonghua for their patience and enthusiasm in helping me to complete this research project. This has been an enormous project that would not have been possible without their guidance and support throughout my work.

I also would like to give special thanks to all the staff in Information Communication Institute of Singapore (ICIS) laboratory for their kind assistance and support.

An Agent Based Application Service Providing System

Table of Contents

ACKNOWLEDGEMENTS	i
TABLE OF CONTENTS	ii
LIST OF FIGURES	v
SUMMARY	viii
CHAPTER 1 INTRODUCTION	1
1.1 MOTIVATION	1
1.2 OBJECTIVES	4
1.3 MAJOR CONTRIBUTION OF THE THESIS	5
1.4 ORGANIZATION OF THE THESIS	6
CHAPTER 2 LITERATURE REVIEW	8
2.1 ASP OVERVIEW	8
2.1.1 Definition	9
2.1.2 ASP Component	10
2.1.3 Evolution	12
2.1.4 Benefits of Using ASP Model	13
2.1.5 Challenges Faced by Current ASP model	17
2.2 AGENT OVERVIEW	19
2.2.1 Definition	20
2.2.2 Basic Features of Agent	22
2.2.3 Agent Architecture	23
2.2.4 Agent Based Applications	24
2.3 CURRENT ASP ARCHITECTURES	25
2.3.1 Windows Terminal Service Technology	25

An Agent Based Application Service Providing System

2.3.2 Client/Server Technology (Fat Client).....	27
2.3.3 Server-based Technology (Thin Client)	29
2.3.4 WebServices Technology	30
2.3.5 Hybrid Approach	33
2.4 SOME ASP MODELS.....	33
2.4.1 Hacigumus's NetDB2 Model	33
2.4.2 Furht's ASP Model	34
2.5 AGENT BASED APPLICATIONS IN SERVICE PROVISION	37
2.5.1 Agentfor Service Browsing and Retrieval	38
2.5.2 Agentsfor Distributed System	41
2.6 AGENT BASED ASP	42

CHAPTER 3 AN AGENT BASED ASP MODEL 45

3.1 DESIGN CONSIDERATIONS	45
3.1.1 ASP Challenge 1: Application Installation	46
3.1.2 ASP Challenge 2: Real Time Technical Support.....	47
3.1.3 ASP Challenge 3: Automation	47
3.1.4 ASP Challenge 4: Personalization and Customization	48
3.2 WHY AGENTS ARE SUITABLEFOR THE ASP MODEL.....	48
3.2.1 Automatic and Real Time Technical Support	48
3.2.2 Personalization	49
3.2.3 Usage Monitoring	49
3.3 DESIGN	50
3.3.1 An Agent Based ASP Model	50
3.3.2 Functions of Each Component.....	54
3.3.3 Communication among the Agents	62
3.3.4 Knowledge Base Design	66
3.4 KEY FUNCTIONALITIES OF THE PROPOSED MODEL	76
3.5 COMPARISON TO RELATED WORKS	79
3.5.1 Methodsfor Accessing ASP Services	79

An Agent Based Application Service Providing System

3.5.2 Types of Services Provided	80
3.5.3 Efficiency of Technical Support	80
3.5.4 Time Needed to Implement a Software Service.....	81
3.5.5 Personalization and Customization	82
CHAPTER 4 APPLICATIONS AND IMPLEMENTATIONS	83
4.1 AUTO APPLICATION INSTALLATION.....	83
4.1.1 Objective	84
4.1.2 Implementation.....	84
4.2 INTELLIGENT PERSONAL ASSISTANT	97
4.2.1 Objective	98
4.2.2 Implementation.....	99
4.3 OTHER POSSIBLE APPLICATIONS	110
CHAPTER 5 CONCLUSION AND RECOMMENDATIONS	111
5.1 CONCLUSION	111
5.2 RECOMMENDATIONS FOR FURTHER RESEARCH	113
AUTHOR'S PUBLICATION	115
BIBLIOGRAPHY	116
APPENDIX	126

An Agent Based Application Service Providing System

List of Figures

FIGURE 2.1: THE ASP COMPONENT [12]	11
FIGURE 2.2: WITH AND WITHOUT AN ASP	15
FIGURE 2.3: SIMPLE AGENT-ENVIRONMENT INTERACTION [26].....	22
FIGURE 2.4: CLIENT/SERVER COMMUNICATIONS IN WTS TECHNOLOGY	26
FIGURE 2.5: SERVER-BASED ASP ARCHITECTURE [43]	29
FIGURE 2.6: WEB SERVICES TECHNOLOGY [73]	30
FIGURE 2.7: HACIGUMUS'S NETDB2 MODEL [47]	34
FIGURE 2.8: FURHT'S ASP MODEL [15]	35
FIGURE 2.9: TRADITIONAL CLIENT/SERVER APPROACH VERSUS AGENT APPROACH IN SERVICE DISCOVERY AND DOWNLOADING [36]	38
FIGURE 2.10: MENA'S MODEL FOR SOFTWARE RETRIEVAL SERVICE [35]	40
FIGURE 2.11: CARCHIOLO'S AGENT BASED PLATFORM FOR A SERVICE PROVIDER [48]	41
FIGURE 3.1: FUNCTIONALITY OF PROPOSED AGENT BASED ASP MODEL	50
FIGURE 3.2: AN AGENT BASE ASP MODEL	53
FIGURE 3.3: REQUEST MESSAGE FORMAT	65
FIGURE 3.4: RESPONSE MESSAGE FORMAT	66
FIGURE 3.5: THE CASE-BASED REASONING MODEL	66
FIGURE 3.6: CBR FOR ABASPM	70
FIGURE 3.7: A DIALOG IN INSTALLATION PROCESS	74
FIGURE 4.1: CLIENT AGENT – LOG ON	86
FIGURE 4.2: CLIENT AGENT – SERVICE SELECTION	86
FIGURE 4.3: CLIENT AGENT – APPLICATION SELECTION	87
FIGURE 4.4: CLIENT AGENT – APPLICATION DOWNLOADING	87
FIGURE 4.5: APPLICATION INSTALLATION – DIALOG 1 (CLIENT SIDE)	88
FIGURE 4.6: APPLICATION INSTALLATION – SOLUTION NOT AVAILABLE IN KB (CLIENT SIDE) ...	89
FIGURE 4.7: TECHNICIAN AGENT - NEW TASK ARRIVAL	89

An Agent Based Application Service Providing System

FIGURE 4.8: TECHNICIAN AGENT-START RECORDING CASE	90
FIGURE 4.9: TECHNICIAN AGENT - APPLICATION INSTALLATION DIALOG 2	91
FIGURE 4.10: TECHNICIAN AGENT- APPLICATION INSTALLATION DIALOG 3	92
FIGURE 4.11: TECHNICIAN AGENT – APPLICATION INSTALLATION DIALOG 4	92
FIGURE 4.12: TECHNICIAN AGENT – APPLICATION INSTALLATION DIALOG 5	93
FIGURE 4.13: TECHNICIAN AGENT-STOP RECORDING CASE	93
FIGURE 4.14: TECHNICIAN AGENT – SAVE CASE	94
FIGURE 4.15: CLIENT AGENT B –APPLICATION INSTALLATION DIALOG 1	95
FIGURE 4.16: CLIENT AGENT B –APPLICATION INSTALLATION DIALOG 2	95
FIGURE 4.17: CLIENT AGENT B –APPLICATION INSTALLATION DIALOG 3	96
FIGURE 4.18: CLIENT AGENT B -APPLICATION INSTALLATION DIALOG 4	96
FIGURE 4.19: CLIENT AGENT B -APPLICATION INSTALLATION DIALOG 5	97
FIGURE 4.20: IPA – START RECORDING	99
FIGURE 4.21: IPA - RECORDING.....	100
FIGURE 4.22: IPA – RECORDING 2	100
FIGURE 4.23: IPA –RECORDING 3	101
FIGURE 4.24: IPA-RECORDING 4	101
FIGURE 4.25: IPA – RECORDING 5	102
FIGURE 4.26: IPA -RECORDING 6.....	102
FIGURE 4.27: IPA – RECORDING 7	103
FIGURE 4.28: IPA –RECORDING 8.....	103
FIGURE 4.29: IPA –RECORDING 9	104
FIGURE 4.30: IPA –STOP RECORDING	104
FIGURE 4.31: IPA - New CASE 1	105
FIGURE 4.32: New CASE - New CASE 2	105
FIGURE 4.33: New CASE - New CASE 3	106
FIGURE 4.34: IPA –LOAD CASES 1	107
FIGURE 4.35: IPA - LOAD CASES 2	108
FIGURE 4.36: IPA- LOAD CASES 3	108

An Agent Based Application Service Providing System

FIGURE 4.37: IPA - LOADCASES 4..... 109

FIGURE 4.38: IPA - SAVE TO KB 109

Summary

The concept of Application Service Provider (ASP) has been proposed at the end of last millennium and aroused comprehensive concern. However, availability of the services is still a major concern. Here the availability refers to the availability of the service that customer has subscribed and the availability of technical support that the customer may need in real time. As the Internet is a heterogeneous environment, different customers have various requests. However, current ASP model could not meet these requirements of ASP customers. In addition, current ASPs are not able to provide real-time technical support to their customers.

In order to solve these problems, we designed an agent based ASP model. The agent based application service providing system provides application services via software agents. Each client puts up requests to his/her agent. Then the agent communicates with coordinator agent, checks the availability of the service, configures the service and supervises the usage of the service. Being autonomous in nature, the agent based ASP system is able to provide services autonomously with enhanced availability. The system maintains a Knowledge Base (KB) so that the agent can work intelligently. A case based reasoning model for the agent based ASP model was proposed to make past knowledge reusable. To illustrate the feasibility of the proposed model, a prototype system has been implemented.

Chapter 1

Introduction

This chapter gives an overview of the research project and the organization of the thesis. It includes:

- The **Motivation** section gives an overview of ASP and agent technology. It also explains the circumstance under which this research project was proposed;
- The **Objectives** section elaborates the goals that we planed to reach;
- The **Major Contribution of the Thesis** section states the main contribution of this research project;
- The **Organization of the Thesis** section briefly summarizes the content of each chapter and the organization of the chapters.

1.1 Motivation

New computing architectures have emerged over the past several years as a result of two major events: the rapid adoption of the Internet as a business tool and increasingly higher availability of broadband connectivity [1]. As a

An Agent Based Application Service Providing System

consequence, more and more organizations are using computers to run applications that carry out specific business functions.

The most crucial business applications are usually the most sophisticated and costly. Thus most Small and Medium Enterprise (SMEs) (and even some large corporations) face difficulties in implementing full integration of business applications because of their limited in-house Information Technology (IT) capabilities [2]

Usually organizations spend a serious amount of money to buy software. Whether it is a desktop application that has to be supplied to thousands of users, or a server-based program of which only needs a few copies, the overall cost is still very high, sometimes extremely high. This is just the beginning. Applications are typically upgraded every 12 to 24 months [3]. Upgrades cost money, regardless of whether software is purchased anew or provided under an annual subscription or maintenance fee.

In the face of the limited in-house IT capabilities and high software costs, a new industry has emerged with startling speed: ASPs. An ASP is a third party entity that hosts software-based services and solutions on remote servers, and delivers the hosted applications to its customers across Wide Area Network (WAN). Applications are delivered in exchange for a rental fee. ASPs can provide a wide range of applications, from email and messaging, Web hosting, financial and accounting applications, to the more complicated Customer Relationship Management (CRM), Enterprise Resource Planning (ERP) and Supply Chain Management (SCM) applications [2][4][5]. Outsourcing these services to

An Agent Based Application Service Providing System

ASPs can help companies reduce operating costs. For example, a company might wish to use Microsoft Exchange as its groupware application, but might not have the in-house expertise to manage the product. Outsourcing to an ASP takes advantage of the ASP's services and technical expertise. Costs may also be reduced in terms of software licenses and hardware.

Mission critical business applications have only recently ventured onto public and WANs, leaving many challenges to be resolved. Thus, the increased complexity and growth of ASP is creating a demand for better performance and more intelligent internetworking. On the other hand, more and more client users are requesting customized services to fulfill their personalized needs. Thus guaranteeing the availability of these IT services and being able to provide customized services are critical to the success of the ASP business model. In the following chapters, we are going to present a novel approach of providing application services, mainly address the challenges faced by current ASPs today.

On the other hand, intelligent agents are a new paradigm for developing software applications. More than this, agent-based computing has been hailed as “the next significant breakthrough in software development”, and “the new revolution in software” [6]. Because of its characteristics of being autonomous, reactive, pro-active, etc., agent has been used in an increasingly wide variety of applications, ranging from comparatively small system such as email filters to large, open, complex, mission critical systems. And in the past few years, some Multi-agent System (MAS) standard [7][8] has been proposed by several independent industrials and research groups. This makes agents much more

An Agent Based Application Service Providing System

widely used in building complex software system. With intensive literature survey on intelligent agent and agent applications, we believe that by applying agent technology to ASP model, we could solve the challenges stated above. A comprehensive survey and discussion on agent technology is presented in the following chapters.

In this project, we aim to propose an agent based application service providing model, to guarantee the availability of ASP services, with a high degree of being autonomous and customization.

1.2 Objectives

The objective of this research is to propose a novel application service providing model, to guarantee the availability of ASP services, with a high degree of being autonomous and customization. To achieve this goal, the following works have to be done:

- Study and analyze current ASP models

By reviewing current ASP models, we can find the advantages and disadvantages of current ASP systems. Then we can make use of the good aspects of it and improve those aspects which are not so good.

- Research and study relevant technologies.

This is the first step before we could apply those technologies to ASP systems.

An Agent Based Application Service Providing System

- Propose a novel ASP model to solve the availability and customization problems of current ASP system.

This is the main objective of this project. This ASP model should be able to provide high availability application services. At the same time, it should be able to provide customized services to fulfill different users' needs.

- Implement a prototype system to prove the feasibility of the proposed model.

In this project, we implemented a prototype system to prove the feasibility of the proposed model.

1.3 Major Contribution of the Thesis

This research aims to propose a novel application service providing model, to guarantee the availability of ASP services, with a high degree of being autonomous and customization. In this research, we have achieved the research objectives. More specifically, the following contributions have been made:

- An agent based ASP model was proposed in this thesis, named ABASPM. This model enhanced the availability of ASP services, compared to current server-based ASP model. The model contains three fundamental building blocks, i.e., Client Agent, Server Agency and external components.

An Agent Based Application Service Providing System

- The model incorporates a KB, so that agents can work intelligently. In case of exception, it will seek help from human being, who normally are technical expert.
- We proposed a Case-Based Reasoning (CBR) model for ABASPM. Compared to traditional CBR model, which usually assume enough past experiences are already available in the KB, the CBR for ABASPM can build up the KB from zero.

All above contributing elements are demonstrated by two prototype applications – the Automatic Application Installation application and the Intelligent Personal Assistant application. The prototype applications serve as a proof of the ABASPM.

1.4 Organization of the Thesis

The thesis is organized as follows:

This chapter, first of all, introduces the background and motivation of the research project. Because nowadays client users more prefer to get customized services and availability is a major problem in current ASP system, we aim to propose a novel application service providing system which is able to solve those challenges faced by current systems.

Chapter 2 provides an intensive literature survey on ASP paradigm, intelligent agents, and current ASP models proposed by other researchers. Applications of

An Agent Based Application Service Providing System

software agents in related field are also presented.

Chapter 3 discusses the reason why agent is suitable for our ASP model, and the important design considerations for an agent based ASP model. The proposed agent based ASP model is also presented in this chapter, including functionalities of all the components in the model, communication protocol and message format between all the components and the detailed design of the KB.

Chapter 4 discusses possible applications that could be implemented by applying our proposed model. Two of those applications, Automatic Application Installation (AAI) application and Intelligent Personal Assistant (IPA) application are elaborated in detail together with their prototype implementations.

Chapter 5 states further works that could be done to enhance proposed framework and concludes this thesis.

Chapter 2

Literature Review

Previous chapter stated that the first mission of this thesis is to carry out an intensive survey on the related technologies. This chapter reviews the essential topics listed as follows:

- **ASP Overview** – ASP definition, ASP components, ASP evolution, the benefits of using ASP model, and challenges faced by current ASP model;
- **Agent Overview** – Agent definition, basic agent features, agent architecture, and agent based applications;
- **ASP Architectures.** Three commonly used technologies will be discussed in this section;
- **Agent applications** in ASP-related field.

2.1 ASP Overview

With companies relying on IT more and more for critical business process, the visibility of the corporate IT function has increased dramatically. At the same time, IT has moved from “behind the scenes” to being a highly visible

An Agent Based Application Service Providing System

function, one charged with leading business growth by delivering new revenue and increased profitability. Because of this increased reliance on IT systems, the pressure being placed on IT departments has increased significantly.

These increased pressures have led IT organizations to show a renewed interest in outsourcing tasks. In the past, this usually meant hiring another party to build a solution that was eventually handed over to the client for day-to-day operation and maintenance. In other cases, solutions created by the company were handed over to another party who took over operations and maintenance [9].

Driven by lower connectivity costs and scarce IT resources, companies have begun to take the outsourcing model one step further, relying on other companies to deliver software functionality as services. The concept of accessing functionality provided by a third party, and running at another location, forms the basis for ASP model.

In this section, we are presenting an intensive literature survey on ASP: its definition, its composing components, its evolution, the benefits it brings to us and challenges it is facing.

2.1.1 Definition

The ASP model is a recent development in the IT arena, with the name first appearing in an International Data Report in July 1998. According to Alexander [10], an ASP is a business that offers multiple customers access to business applications over the network (e.g., the Internet) while charging rental or

An Agent Based Application Service Providing System

subscription-based fees for services used. Or, deciphering this definition, an **ASP** is a business that:

- Delivers application services over the network;
- Delivers services to many customers with a wide range of requirements;
- Charges rental or subscription-based fees;
- Provides customer-specific service guarantees.

In essence, the ASP acquires and maintains the software. It also acquires the server hardware and the network and rents it to the client, typically for a fixed period of time at a specified price. This arrangement, somewhat akin to leasing a car, means that the client does not have to bear the up-front capital costs of the hardware and software but instead can pay for their usage more gradually.

There are a lot of well-known ASPs. Hewlett-Packard, SAP, and Qwest have formed one of the first major alliances for providing ASP services. They plan to make SAP's popular R/3 applications available at "cybercenters" that will serve the applications to other companies [11]. Microsoft is also allowing some companies to offer its BackOffice products, including SQL Server, Exchange and Windows NT Server on a rental, pay-as-you-use basis [11].

2.1.2 ASP Component

The whole concept of hosting and delivering applications to users encompasses a wide range of IT infrastructure needs, from hardware and networking requirements to software development. Usually, an ASP alone does not provide for all the necessary infrastructure and software. Therefore, ASPs need to

An Agent Based Application Service Providing System

work closely with other infrastructure partners, or enablers, to facilitate the process. The relationship among these various components is shown in Figure 2.1.

ASPs are the ones that host applications on centralized servers. Ideally, ASPs also provide the crucial application technical support.

The Internet Service Providers (ISPs) and Network Service Providers (NSP) provide basic communications, server center resources, and Internet protocol services. They take care of the connectivity issues and infrastructure. Hardware manufacturers provide the equipment, including storage, servers, etc.

Figure 2.1 : The ASP Component [12]

Independent Software Vendors (ISVs) are the ones that provide the applications to be hosted by ASPs. Traditional software vendors include the large ERP vendors such as Oracle, PeopleSoft and SAP, CRM companies. Siebel Systems is another leading ISV. In addition to these enterprise software vendors, there

An Agent Based Application Service Providing System

are many horizontal, vertical and niche players that are looking towards the ASP business model as a way of offering their software over the Internet [13].

All these components come together to form the ASP application provision and delivery model. However, end users (companies or individuals) usually only deal with ASPs. The other supporting elements are transparent to them.

Figure 2.1 merely shows the common relationship and interaction between various ASP components. However, this figure does not define ASP architecture. The current ASP architecture is discussed in section 2.3.

2.1.3 Evolution

In [9], Microsoft examined past and present challenges faced by IT community, and how they are driving the evolution and adoption of the ASP model. The ASP model, in its brief life, has evolved through several overlapping phases. The first phase, which is known as desktop phase, focuses on the delivery and management of desktop applications. The main technologies used in this phase are Windows Terminal Service (WTS), Microsoft Office, IntelliMirror management technologies, System Management Server, and products from other vendors. The benefit to the organization is mainly cost saving.

The second phase in the development of the ASP model, known as line-of-business phase, focuses on the delivery and management of server-based applications. With ERP and CRM being the most commonly provided applications, they provide messaging and line-of-business solutions. In this

An Agent Based Application Service Providing System

phase, ASP models are generally based on large, established client/server applications that have been adapted for delivery over the Internet. They usually communicate with the user's desktop via Web browser. As a result, they are unable to take full advantages of the range of capabilities offered by the client desktop.

The first two phases in the evolution of the ASP model have already started. Most ASPs are offering server-based application services today. The architecture of server-based ASP will be discussed in detail in section 2.3.

2.1.4 Benefits of Using ASP Model

The ASP model's benefits derive from distributing its software application across multiple servers rather than across multiple clients. A combination of a rental commercial model, a component-based application architecture, and a server-based thin client computing environment provides the greatest benefits [14]-[17].

2.1.4.1 Benefits to Software Vendors and Service Providers

No distribution cost

The ASP model eliminates the need to print manuals, press disks, order thousands of colorful cardboard boxes, manage stock, and operates a returns procedure.

An Agent Based Application Service Providing System

No user installation

Removal of an installation procedure eliminates the expense of providing user support for it.

Faster time to market

ASPs can generally deliver software to a client more rapidly than in-house resources or an external system integrator. There are two reasons for this. One is the fact that the ASP model eliminates the need to print manuals, press disks, order thousands of colorful cardboard boxes, etc. The second reason is that the ASP is putting the software on its own hardware, a configuration it is familiar with. For example, a typical ERP installation at a client site can take from six months to two years. Enterprise ASPs can typically deliver ERP solutions in 90-120 days [18].

Reduced piracy

Because the bulk of software is residing on the server, users cannot copy and distribute the full version.

Instant upgrades

Suppliers can implement bug fixes and new features automatically, without waiting for users to discover, download, and install the new codes.

Consistent user base

Automatically updated software that resides on the host reduces or completely

An Agent Based Application Service Providing System

eliminates the proliferation of different application versions and released levels.

Usage monitoring

Suppliers can monitor usage to gain a greater understanding of user interaction with the product, discovering the most and least popular features, identifying which features appear to cause the most problems, and determining which features must be streamlined to improve productivity.

Potentially consistent revenue stream

A steady stream of rental fees releases suppliers from the need to create annual releases simply to generate revenue.

2.1.4.2 Benefits to ASP Customers

Figure 2.2: With and without an ASP

An Agent Based Application Service Providing System

Turnkey solution

ASPs bundle software, hardware, and system development, integration, and management into one offering. The result is that the role of general technology contractor shifts from the company to the ASP. The result is a significant reduction in the decision making and administrative burden on a company looking for IT solutions (Figure 2.2).

Limitless choice

The Internet gives users access to every rental applications available online. The rental model gives users unbounded potential to access and combine services.

No installation hassle

ASP software can be used immediately, bypassing the time-consuming and potentially troublesome client installation. No compatibility hassle.

No support overhead

Users do not need to employ expensive administration and support staff to operate complex software installations and the equipment required to run them. The service provider takes care of these issues and builds the cost of doing so to the application's subscription price.

Improve focus on core competencies

Using an ASP allows an IT department to focus more of its attention on more

An Agent Based Application Service Providing System

critical software and system problems that are important for the company to function.

Reduced downtime

Most online service providers do a better job of ensuring 24 hours availability than customers can because they depend on their products' reliability to stay in business.

Predictable costs and lower up-front investment

ASPs typically offer their services using contractual monthly fees. Because of the lower up-front costs, an ASP can allow a company to get into a more sophisticated software product than if purchased outright. Prices may vary depending on the type of software, the level of customization, the number of users, and the length of the contract. If there is an up-front cost, it will usually relate to the implementation. The license cost and ongoing maintenance and connectivity cost are part of the monthly fee.

Better performance

Usually ASPs can do a better job of ensuring applications availability and reliability than customers could, because that is what they do best, to stay in business.

2.1.5 Challenges Faced by Current ASP model

While the current ASP model offers many benefits, there are a few key

An Agent Based Application Service Providing System

concerns:

Availability

The availability refers to the availability of the service that customer has subscribed to and the availability of technical support that the customer may need in real time. Most clients are worrying about not being able to get the connectivity of the applications and support they need [19].

Automation for inexperienced user

Some Personal Computer (PC) users have problems in using some complicated software applications, thus they might need usage support in real time. Current ASPs are using the normal way to provide this kind of service, i.e. telephone line. This may cause some problems when the two parties misunderstand each other.

Security

This addresses the security of client data. Security in terms of connection is not a major concern, as today's virtual private network, encryption and firewall technology are sufficient for highly secure connection and access. The major concern is that clients are wary about handing over critical operations and sensitive data to external firms and worry they may not get the connectivity and access they need [19].

An Agent Based Application Service Providing System

Bandwidth needs

Since a large part of potential users might access applications hosted simultaneously, there will be a huge increase in the bandwidth demand.

Personalization and Customization

Multiple clients connected to the ASP would have different preferences. Each client might require personalized service. However, server-based application hosting systems is difficult to do, because applications are hosted at the server side, and it is not feasible to host thousands of different versions of applications to fulfill each customer's needs. To be truly successful, a service must not only be easy to learn, but also be able to store individual user preference and customization [19].

Payments

ASPs require an efficient and accurate way of monitoring and recording usage of applications. This is for billing purposes if the rental is based on usage rather than fixed-monthly charges.

2.2 Agent Overview

Today's information society is becoming ever increasingly sophisticated. Some of the hallmarks of this sophistication including a steady growth in the production and consumption of information, heterogeneous and distributed information environments; a diversity of information processing needs and

An Agent Based Application Service Providing System

products, and, in general, a greater reliance on computer-based information processing systems to provide not only accurate and timely, but also value added solutions to complex commercial and industrial problems. This has led to a state of affairs where traditional IT systems are increasingly hard-pressed to meet the challenges of, for example, integrating information from several distributed and heterogeneous sources and processing this information to facilitate decision-making, while reducing the information overload on users. An agent-based technology is rapidly evolving to meet the demands of this new information era.

2.2.1 Definition

Despite the phrase “agent-based” in describing much current software technology [20]-[23][27], there is, as yet, no commonly agreed definition of exactly what an agent is. In this thesis, we adopt the definition of agent as:

“a self-contained program capable of controlling its own decision-making and acting, based on its perception of its environment, in pursuit of one or more objectives” [24]

Wooldridge and Jennings [25] distinguish two general usages of the term “agent”: the first is weak, and relatively un-contentious; the second is stronger, and potentially more contentious.

An Agent Based Application Service Providing System

2.2.1.1 Weak Notion of Agent

The most general way in which the term agent is used is to denote a hardware or software-based computer system that enjoys the following properties:

- **Autonomy:** agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- **Social ability:** agents interact with other agents (and possibly humans) via some kind of agent-communication languages.
- **Reactivity:** agents perceive their environment (which may not be 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.
- **Pro-activeness:** agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative.

2.2.1.2 A Stronger Notion of Agent

For some researchers, particularly those working in Artificial Intelligence (AI), the term “agent” has a stronger and more specific meaning than that sketched out above. These researchers generally mean an agent to be a computer system that, in addition to having the properties identified above, is either conceptualized or implemented using the concepts that are more usually applied

An Agent Based Application Service Providing System

to humans. For example:

- Mobility is the ability of an agent to move around an electronic network.
- Veracity is the assumption that an agent will not knowingly communicate false information.
- Benevolence is the assumption that agents do not have conflicting goals, and that every agent will therefore always try to do what is asked of it.
- Rationality is the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved – at least insofar as its belief permit

2.2.2 Basic Features of Agent

Figure 2.3: Simple Agent-Environment Interaction [26]

The basic feature of an agent is its ability to communicate with other agents and the environment. A simple agent-environment interaction diagram is shown below. The agent senses its environment, uses what it senses from its

An Agent Based Application Service Providing System

environment, and then performs the action through its effectors. Sensory input can include received messages, and action can be the sending of messages.

2.2.3 Agent Architecture

There are two famous agent architectures, which are Reactive agent and Belief-desire-Intention (BDI) agent.

2.2.3.1 Reactive Agent

A reactive agent is the simplest kind to build, since it does not maintain information about the state of its environment but simply reacts to current perceptions. In [26], Jose et al. designed such an agent in 2001. It encapsulates a collection of behaviors, known as plans, and the means for selecting an appropriate one. Each behavior fires when it matches the environment. The user is responsible for ensuring that at least one behavior will match for every environment. This can be achieved by defining a default behavior that matches all inputs but is inhibited by all other behaviors that match.

2.2.3.2 BDI Agent

One of the most successful agent theories is the BDI model [28]. As the name suggests, the internal state of a BDI agent is composed of three key data structures, which are intended to correspond loosely to beliefs, desires and intentions. An agent's beliefs are intended to represent the information it has about the world. An agent's desires may be thought of as the tasks allocated

An Agent Based Application Service Providing System

to it. While an agent's intentions represent desires that it has been committed to achieving.

The advantage of this approach is that the interpreter can stop the program at any time, save state, and execute some other plan, or intention, if it needs to.

2.2.4 Agent Based Applications

Agent-oriented programming is suitable for solving many complex problems [30][70][71]. This is because agents are task-oriented, whereby developers only need to define a solution to a problem in terms of agent behaviors. Developers can design and think of a solution at a higher level of abstraction. For example, by encapsulating the communication protocols in an agent, developers need not be concerned about the interaction between agents. They can simply focus on defining the actions to be taken by each agent under specific circumstances. The process of finding a solution becomes the process of defining agents' condition-action rules. Application can be represented in natural way, which makes it easier for programmers to understand applications because the roles of the agents are very clear.

Distributed computing models and applications made up of numerous interacting components are the most suitable implementation areas of the agent paradigm [31]. Agent technology is well-suited for use in applications that reason about messages received over a network, which is why agent-based approaches are so popular in applications that utilize the Internet. Agent-based applications are endowed with the capability of distributed, asynchronous

An Agent Based Application Service Providing System

and parallel processing. Also, agents are ideal for applications that require high-level of automation. It is the inherent characteristics of agents that appeal to the above types of applications. Examples of the most popular fields of applications that benefit from the agent paradigm are: e-commerce [32], personal assistance [33][34], distributed information retrieval [35]-[37], secure brokering, telecommunication network services, workflow applications [38], monitoring and notification processes [39], information dissemination and parallel processing [40].

2.3 Current ASP Architectures

There are five common architectures for the ASP delivery model: 1) Windows Terminal Service Technology; 2) Client/Server Technology (Fat Client); 3) Server-based Technology (Thin Client); 4) Web Services Technology; 5) A hybrid approach

2.3.1 Windows Terminal Service Technology

Windows Terminal Service technology [41] is one of the application delivery approaches. In WTS technology, applications are deployed, managed, supported and executed on the server side. Only screen images are transmitted from the server side to the client side.

An Agent Based Application Service Providing System

Figure 2.4: Client/Server Communications in WTS Technology

For example, there are basically three components to Windows 2000 Terminal Services: 1) the Terminal Service Server; 2) the communication between the server and the clients; 3) the client.

Nearly all of the computing resources that will be used in the Terminal Service networking environment are residing on the server. The server will receive the process, keystrokes and mouse movements that take place at the client PC.

The second component is the communication between the server and clients. This communication takes place via the Remote Desktop Protocol (RDP) [42]. RDP is an application-layer protocol replying on TCP/IP. It is optimized for moving graphical information quickly and efficiently across the network.

The third component to Terminal Services is the client. Using a minimal amount of client software, a PC on the network is able to open a window containing a terminal session. The remote desktop running on the server is in this window. Applications and windows opened on this desktop are actually running on the server.

The benefits of using WTS are mainly in two ways:

An Agent Based Application Service Providing System

1) Cost. The cost benefits stem mostly from the fact that you can run a Terminal Server session on a PC that would otherwise not meet the minimum hardware requirements for running locally. There are other cost savings in terms of time savings, such as when deploying new applications. Simply installing a new application on the server automatically makes it available to the clients you specify.

2) Administration. Besides application deployment, Terminal Services allow administrators to log on the server as though they are sitting right at the console, so there is much less need to go to server room for administration purpose. Another administrative benefit is in user support. Support technicians can monitor a client session, and also remote control a client session, which results huge benefits when taking support calls.

The drawbacks of using WTS are:

1) A certain portion bandwidth is exclusively occupied by each client throughout the session. Thus a huge bandwidth is need if many clients are using the service at the same time.

2) As all the Graphical User Interface (GUI) are generated at server side, a respectively high computational consumption is expected.

2.3.2 Client/Server Technology (Fat Client)

In client/server (fat client) approach, two pieces of software are needed to enable the service: one server side software, and one client side software. The

An Agent Based Application Service Providing System

responsibilities of the client side software are: 1) handle the user interface; 2) translate the user's request into the desired protocol; 3) send the request to the server; 4) wait for the server's response; 5) translate the response into "human-readable" results; 6) present the results to the user. The responsibilities of the server side software are: 1) listen for a client's query; 2) process the query; 3) return the results back to the client.

A typical client/server interaction goes like the follows:

1. The user runs client software to create a query.
2. The client connects to the server.
3. The client sends the query to the server.
4. The server analyzes the query.
5. The server computes the results of the query.
6. The server sends the results to the user.
7. Repeat as necessary.

Flexible user interface development is the most obvious advantages of client/server computing. It is possible to create an interface that is independent of the server hosting the data. On the other hand, the server has more computing resources to spend on analyzing queries and disseminating information, since the user interface is the responsibility of the client.

However the client PC must meet a minimum hardware specification to run the client software. At the same time, different version of client application

An Agent Based Application Service Providing System

software must be provided by the ASP for different platforms, e.g. Windows, Solaris and Linux.

2.3.3 Server-based Technology (Thin Client)

Figure 2.5: Server-based ASP Architecture [43]

Most current ASPs are applying server-based ASP architecture. The server-based ASP architecture caters for existing applications. In this case, ISVs concentrate on developing the applications, while the hosting and delivery work are left for the ASPs. ASPs do not need to have much concern about how the applications are designed. Thus, the ASP is independent of the ISV. In this ASP architecture, the ASP is an intermediary between its clients and ISVs, as shown in Figure 2.5.

An Agent Based Application Service Providing System

In this architecture, there are three parties: clients, ASP and ISVs. ISVs provide the applications to the ASP. ASP in turn allows multiple clients to access the applications that are hosted by it. The main focus is on the design of the components of the ASP side, and the various services and functions that are provided by the ASP in support of application hosting. Examples of server-based ASPs are Corio [44], mySAP [45] and Futurelink [46].

2.3.4 Web Services Technology

According to W3C, a Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP message, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [73].

Figure 2.6: Web Services Technology [73]

An Agent Based Application Service Providing System

The Web Services approach is based on a maturing set of standards that are widely accepted and used. It makes it possible for client and services to communicate and understand each other across a wide variety of platforms and across language boundaries. In the following paragraph, the protocols and technologies that constitute these Web Services Standards will be discussed.

XML

Extensible Markup Language (XML) is the standard for describing data to be exchanged on the Web. As its name indicates, XML is a markup language. It makes use of tags that “mark up” the contents of a document. An XML tag identifies the information of a document, as well as the structure of the information.

WSDL

WSDL (Web Service Definition Language), and XML-based technology, defines Web services interfaces, data and message type, interaction pattern, and protocol mappings.

A WSDL documents defines a Web Service using these major elements: <portType>, <message>, <types> and <binding>. The <portType> element is the most important WSDL element. It defines a Web Service, the operations that can be performed, and the message that are involved. The <message> element defines the data elements of an operation. The <type> element defines the data types that are used by the Web Service. For maximum platform neutrality, WSDL uses XML schema syntax to define data types. The

An Agent Based Application Service Providing System

<binding> element defines the message format and protocol details for each port.

SOAP

Though the meaning and structure of XML tags makes the use of XML an effective way of exchange data, it is not sufficient for data interchange over the Web. An agreed-upon protocol for formatting XML is needed so that the receiver understands what the message is. SOAP (Simple Object Access Protocol) is an XML-based protocol for exchanging information in a distributed environment. SOAP provides a common message format for exchanging data between clients and services.

UDDI

UDDI (Universal Description, Discovery, and Integration), a Web services registry and discovery mechanism, is used for storing and categorizing business information and for retrieving pointers to Web services interfaces.

The main idea of Web Services is to combine Web and distributed objects into a single framework where the user-to-component as well as component-to-component interactions are conducted by using standard Web technologies. In this way, integration between different Web Service based systems is much easier compare to traditional systems. Some researchers have already applied Web Services technology in application service provision field [74]-[76]. They mainly focused on solving integration issue faced by current ASP.

2.3.5 Hybrid Approach

Some ASP models are proposed based on the combination of the technologies discussed above. Hybrid approach takes advantages of various technologies and provides different ways to access the ASP services. An example could be Furht's ASP model, which will be discussed in section 2.4.2.

2.4 Some ASP Models

In this section, some ASP models will be discussed.

2.4.1 Hacigumus's NetDB2 Model

NetDB2 is a server based database service built on top of DB2 that provides users with tools for application development, creating and loading tables, and performing queries and transactions. The basic NetDB2 system is implemented based on 3-tier server-based architecture, namely: the presentation layer (consists of the end user's Web browser at NetDB2's HTTP server), the application layer (consists of Java Servlets managed by a Servlet engine), and the data management layer (consists of a database manager and a backup/recovery server). There are two purposes of separating NetDB2 into layers. The first is the insulation of software components of one layer from another; the other is the separation of concerns that help achieve better interoperability and higher scalability.

An Agent Based Application Service Providing System

Figure 2.7: Hacigumus's NetDB2 Model [47]

In this model, the only way to access the ASP services is through Web browsers.

Such a model alleviates the need for organizations to purchase expensive hardware and software, deal with software upgrades, and hire professionals for administrative and maintenance tasks which are taken on by the service provider. This is also one of the main reasons why people are adopting ASPs. However, this model is specially designed only for providing database systems as a service.

2.4.2 Furht's ASP Model

Furht's ASP model was proposed based on the combination of the technologies discussed in section 2.3. It uses the following technologies developed by Citrix:

- Independent Computing Architecture (ICA)
- Windows-based Terminal (WBT)
- Software component that allow consumers to access applications from their Internet browser (Charlotte), and
- Software component that allows interactive applications on the Web (Verigo)

An Agent Based Application Service Providing System

Figure 2.8: Furht's ASP Model [15]

ICA is a Windows presentation service protocol that turns any client device (thin or fat) into a thin client.

A WBT is a thin-client hardware device that connects to Citrix server-based system software. The WBT does not require downloading of the Operating System (OS) or applications and there is no local processing of applications at the client.

This model also allows consumers and business to access software applications from their Internet browsers. This is provided using Citrix's software Charlotte. In addition, software Vertigo allows more interactive applications on the Web. This software allows customized Web pages such as electronic trading accounts

An Agent Based Application Service Providing System

to be updated automatically without hitting the refresh button on the computer.

This proposed architecture is platform independent and allows non-Windows and specialized ICA devices to run an applications server farm. An application server farm is a group of application servers that are linked together as a single system to provide centralized administration and scalability.

In this model, users can access the ASP services either through Web browsers or using windows-based terminal.

Advantages:

- Single-point management
- Universal application access
- Wide choice of client devices

Disadvantages:

- Technical support

It is stated in ASP Island that “a complete application service consists of three unique service components: application hosting, application delivery and application technical support” [12]. Application hosting, application delivery and application technical support are all very important in making an ASP model more desirable. However, in this model, efficient technical support is not provided. That means only traditional technical support, hot phone line, for example, is used.

An Agent Based Application Service Providing System

Some may argue that for server-based application hosting model, technical support can be provided solely at the server side. It is true for application installation, updates and maintenance. But people might encounter problems when they are using complex software applications. Therefore, efficient software usage support should also be provided to make an ASP attractive.

- Security.

Here the security refers to the security of critical applications and sensitive data. This is a common problem existing in all server-based computing models. Such ASP client users might wary of putting their sensitive data at the server side. Thus, giving users option of putting data at local PC will be very sought-after.

- Personalization.

Multiple clients connected to the ASP would have different preferences. Each client might require personalized service. However, for a server-based application hosting systems this is difficult to do, because applications are hosted at the server side, and it is not feasible to host thousands of different versions of applications to fulfill the need of all customers.

2.5 Agent Based Applications in Service Provision

In this section, we will review some agent applications in the service provision field.

An Agent Based Application Service Providing System

2.5.1 Agent for Service Browsing and Retrieval

The mobility of software agent is very useful for service discovery and downloading. A mobile agent is able to move from one server to another while having access to all required resources.

Mobile agent is very useful for service browsing and retrieval. The most challenging issue in service browsing is how to choose the desired service from thousands of available services in a relatively short time. Doing this manually is very time consuming. A detailed comparison between manually browsing and agent based service retrieval service can be found in papers [35][36].

The advantages of using agents to search and download information are illustrated in the following figure. The black areas represent brisk periods of remote access.

Figure 2.9: Traditional Client/Server Approach versus Agent Approach in Service Discovery and Downloading [36]

An Agent Based Application Service Providing System

In the traditional client/server approach, many spurts of data transfer occur because bits of information are regularly downloaded to the client in response to each user's action (for example, mouse clicking on links). These frequent occurrences of data transfer means that a constant network connection needs to be maintained throughout the service-selection process. Whereas by using mobile agents to encapsulate user's preferences/instructions, the total number of remote accesses needed is reduced dramatically due to the local interaction between the mobile agent and user, and local interaction between mobile agent and remote data source. Occasionally, the mobile agent migrates back to the client, bringing intermediate results. Network connection need not be maintained between A to B, C to D, and E to F. Thus even though the total time required for the service-selection process might be longer, the charges for network connection is reduced.

The benefit of using mobile agents for software downloading is due to the fact that the movement of code portions can be done transparently. Various commercially available agent platforms provide transparent mechanisms for code mobility, which are generally encapsulated in a single primitive or method. Software agents for service downloading can make this process transparent to the end users.

An Agent Based Application Service Providing System

Figure 2.10: Mena's Model for Software Retrieval Service [35]

The ability of retrieving and installing software in an easy and efficient way confers competitive advantages on computer users. In [35], Mena proposed an agent based software retrieval system which allows users to select, retrieve and install software anywhere and at any time. From the implementation point of view, the service was based on the use of mobile agents because: 1) they allow the management of knowledge used to solve problems and to take appropriate decisions, and 2) they are able to travel from one computer to another. From the point of view of the functionalities that the system provides: 1) it incorporates a majordomo agent that acts on behalf of the user, and so relieves the user from many tasks, 2) it incorporates a ontology-based mechanism that catalogs software and takes care of new releases and new products, 3) it helps users in

An Agent Based Application Service Providing System

the task of finding the adequate software, and 4) it incorporates mechanisms to get concrete software, carry it to the mobile computer and install it. The problem targeted was how users (especially ones with mobile devices) can find and acquire the exact program needed and suitable for their computer, while trying to shorten the total time in which the remote connection needs to be open.

2.5.2 Agents for Distributed System

Agents can provide more reliable services in a distributed system [71][72]. Carchiolo et al. [48] proposed a platform for a service provider that can provide reliable services. The main feature is to make it possible to serve clients who do not need special software to access the service they require, but are only equipped with an engine for network access.

Figure 2.1 1: Carchiolo's Agent Based Platform for a Service Provider [48]

In this model, Virtual Service Provider (VSP) is a set of heterogeneous hosts

An Agent Based Application Service Providing System

connected by the Internet. The consistency of the data distributed over the hosts in the VSP is guaranteed by a distributed database, called Virtue Database (VDB), which contains the data to manage both the VSP and the services the VSP offers. This information, contained in the VDB, includes a list of the hosts making up the VSP at any one time and the services they can offer.

The VSP is made up of a VDB and a set of agents: an agent for each of the services of the VSP providers (SA_i), and a Service Manager Agent. When a client requests a service, the Client Agent will migrate to the VSP. And the request is received by all the hosts making up the VSP, as shown in Figure 2.10. Then a Service Agent will be chosen to provide the service, by applying some predefined policies. After completing its task, the Client Agent returns to the source location, called the Home Location, with the result of the service.

The main advantage of this model is that this ASP model is able to provide reliable services. To achieve this, the author designed a platform based on use of a pool of heterogeneous hosts, each of which provides some of the services required. A number of policies have been made to choose which server is able to provide the service to the end user.

2.6 Agent Based ASP

After looking at the ASP concept and properties of software agents, it is prudent to have an initial evaluation of the viability and suitability of deploying an

An Agent Based Application Service Providing System

agent based ASP model. It is observed that the characteristics of software agents enable software agents to fit well in the ASP model in several aspects.

- Providing remote technical support at the client side. Agents can migrate or be dispatched to an ASP client's computer and take over the whole process of setting up the client-side program (for accessing the server side application). Whenever a technical problem occurs, an agent, which can be dispatched from the ASP or could have already been residing on the client's computer, will carry out the technical support steps for fixing the problem. This feature greatly shortens the response time and saves the travel time that would otherwise be required of the ASP staff.
- Automation for inexperienced users. Many ordinary pc users are inexperienced in the details of application installation, configuration and upgrades, and thus do not welcome the fact that they have to deal with these issues. An agent-based approach has the necessary autonomy for taking over those tasks transparently, based on pre-defined rules.
- Realizing complex systems. Software agents are an innovative technology for the efficient and intelligent realization of complex, distributed and highly interactive heterogeneous systems and applications. An ASP model is exactly that type of system – complex, distributed and with highly interactive components. The complex model can be decomposed into multiple, autonomous components that interact.

An Agent Based Application Service Providing System

- Handling of persistent and repetitive tasks. An ASP system as a whole is a collection of various persistent services that cater to the needs of clients. Such services include billing, information management, reporting and logging, client requests processing and handling, etc. These services are repetitive and do not require much innovative contemplation. Software agents can conveniently take over these tasks.

Therefore, software agents can potentially provide valuable enhancement to realize a more efficient and automated ASP model.

Chapter 3

An Agent Based ASP Model

Previous chapter gives an intensive literature review on ASP, intelligent agent, current ASP Architecture and models and agent applications in service provision. At the end of chapter 2, we have easily derived that agent technology could help us to solve the two challenges faced by current ASP model: to guarantee the availability of ASP services, with a high degree of being autonomous and customization.

In this chapter, we are going to present an agent based ASP model. The proposed model is able to solve the challenges addressed above. This chapter is organized as follows: firstly we discuss the design considerations. Then we give our proposed model. Follow by detailed explanation of the proposed model, including functionality of each composing agent, communication protocol and message format between those agents and the KB design.

3.1 Design Considerations

The Agent Based Application Service Providing Model (ASASPM) aims to solve two challenges faced by existing model: 1) to guarantee the availability of

An Agent Based Application Service Providing System

ASP services; 2) to be able to provide customized services.

Availability problem is the key issue to be addressed in this project. And the availability refers to the availability of the service that customer has subscribed and the availability of technical support that the customer may need in real time. So during the design process, we are focusing on two points: firstly, how to improve the availability of ASP services; Secondly, how to make it autonomous, i.e. less human interference.

We are going to address the customization problem by incorporate a KB. The KB helps our ASP system work intelligently.

3.1.1 ASP Challenge 1: Application Installation

The common way of providing services by most ASPs is to host the applications at the server side. In this case, the clients usually access the services by using Web browsers, without downloading and installing the application software. However, we believe that, most of the time there is a need for the client to download and install the application or part of it at his/her local computer. One reason would be the security issue. It is because of the fact that client may be wary about handing over critical operations and sensitive data to external firms. Secondly, availability is also their concern. Clients may worry that they may not get the steady connectivity and be able to access the applications when they need to do so. Lastly, when a client knows that he is going to use an application frequently for a specific period, he would prefer to download and install it, rather than remotely accessing the application every

An Agent Based Application Service Providing System

time. Therefore, we believe a good ASP model should provide also desktop-based application installation service besides server-based application hosting service.

3.1.2 ASP Challenge 2: Real Time Technical Support

Nowadays, one of the main reasons that SMEs adopt ASPs is because it eliminates the need to employ expensive administration and support staff to operate various technical supports. For example, software installations and upgrading, complex software usage support, etc. However, the common way of providing this kind of technical support is still inefficient. They usually give a telephone call to the ASP center and ASP technical staff will either tell users what to do in the phone call or come down to the customer side personally. This is very time consuming. And sometimes misunderstandings may occur when they talk by telephone line. One of the main purposes of this project is to propose an ASP model to provide real time technical support autonomously. The complex service providing process is transparent to the end user.

3.1.3 ASP Challenge 3: Automation

Most common PC users are inexperienced in the details of application installation, configuration, and upgrades. Thus, automation of all these tasks will be tempting. In our agent based ASP model, the Client Agent should be able to handle all these tasks automatically, transparent to the end user.

3.1.4 ASP Challenge 4: Personalization and Customization

Multiple clients connected to the ASP would have different preferences. Each client might require personalized services. However, for a server-based application hosting system this is difficult to do, because applications are hosted at the server side, and it is not feasible to host thousands of different versions of applications to fulfill each customer's needs. Our desktop-based application installation service, which is the main part of our proposed model, will be able to provide personalized application services to the end users.

3.2 Why Agents are Suitable for the ASP Model

Referring back to our design considerations discussed in section 3.1, the characteristics of agents presented in section 2.2, and agent applications discussed in service provision (section 2.4), we will see why agents are suitable for our ASP model.

3.2.1 Automatic and Real Time Technical Support

The fact that agents are autonomous, together with other attributes, contributes to their potential usage in the ASP model for providing IT services. Agents can migrate or be dispatched to an ASP client's computer and take over the whole process of setting up the client-side program (for accessing the server side application). Whenever a technical problem occurs, an agent, which can be dispatched from the ASP or could have been residing on the client's

An Agent Based Application Service Providing System

computer, will provide the technical support for fixing the problem by itself. This feature greatly shortens the response time and saves the travel time that would otherwise be required of the ASP staff. After all, many technical procedures are routinely repeated. These can be assigned to and handled by agents.

This is the main reason why we apply agent technology into our ASP model design. The details of advantages of using agents in this model can be found in a later section.

3.2.2 Personalization

As we discussed in part 3.1.5, multiple clients connected to the ASP would have different preferences. Each client would require personalized service. In this aspect, agents are suitable for providing the personalization touch [49][50]. This is because of the fact that agents are highly customizable. Their behaviors can be set to depend on recorded/learned user settings and preferences. It is easy to express user preferences by keeping user profiles.

3.2.3 Usage Monitoring

Being persistent in nature, an agent that is deployed at the client's side is suitable for monitoring and "remembering" the exact usage of applications by users at the client side. It will be able to generate the exact billing report to be sent to the ASP side at specific interval of times.

3.3 Design

In this section, we are presenting our proposed model. The detailed functionality of each composing agent, communication protocol and message format between those agents are also discussed in this section. Last but not least, the most important part, KB design will be discussed.

3.3.1 An Agent Based ASP Model

Based on the literature review on ASP and software agent, **an** agent based ASP model is proposed. The proposed ASP model has the following functions:

Figure 3.1: Functionality of Proposed Agent Based ASP Model

User authentication and reporting

Customer's personal information, preferences and application usage history are

An Agent Based Application Service Providing System

stored for several reasons: 1) user authentication; 2) to provide customized services based on users' preferences; 3) for billing purpose.

There are three ways to store those data: 1) stored at client side; 2) store at server side; 3) a hybrid method. Each way has its advantages and disadvantages.

We applied the hybrid approach in our model. We put application session data at client PC. While other sensitive personal information at the server side. In this way, the customer can use any machine at any location to access the same customized applications.

Application hosting

Our ASP provides application hosting services, meaning that the applications are hosted at servers and can be accessed by users without downloading the applications.

Automatic application installation

We believe that for several reasons, users might prefer to download and install some applications to their PCs. 1) data privacy and security; 2) if an application will be used frequently for a period. Therefore, we also provide desktop-based application downloading and installation services.

Automatic technical support

One of the main reasons why users adopt ASPs is because of the technical support. Therefore, this is a main factor for deciding on ASPs.

An Agent Based Application Service Providing System

Personal assistant

Nowadays more and more customers are requesting customized service.

Personal assistants are used in our model to provide customized application services.

Based on the design considerations and functionality needed, the following agent based ASP model is proposed.

An Agent Based Application Service Providing System

Figure 3.2: An Agent Base ASP Model

3.3.2 Functions of Each Component

3.3.2.1 Client-Side Component

There is only one component at the client side, which is the agent that is responsible for assisting the user. It is called the Client Agent. The Client Agent will be available at the user's machine when he installs the ASP client application. One Client Agent is created and assigned to each user session with ASP.

Client Agent

Client Agent is the most important component, because from the user's point of view, the Client Agent is the only way he/she can interact with and obtain the ASP services. The Client Agent should be highly automated, so as to handle as many tasks as possible for the user. From the ASP center's point of view, each Client Agent is a user. All service requests received by the ASP center are coming from the Client Agents. The functions for the Client Agent are:

Communication

Client Agent is responsible for establishing network connection with the ASP side. Further actions include sending and receiving messages, instructions and notification, retrieving and updating user profile stored at the ASP side, requesting technical support and the authentication process with the ASP.

An Agent Based Application Service Providing System

Application management

Whenever the user intends to access an application, the Client Agent is responsible for requesting sessions from the ASP. If the user wishes to obtain an application to be installed locally, Client Agent should automatically handle the tasks of application downloading, installation and troubleshooting. These tasks are enabled by a KB approach described in section 3.3.4. Client Agent automates the installation process by executing the installation file, monitor and sense the appearance of installation dialog boxes, identify the dialog box and fetch the appropriate mouse/keyboard commands from the KB server, and executing the fetched commands. This method is a cycle of “event identification” and “perform corresponding action”.

Personalization

The Client Agent is responsible for collecting user preferences and sending it to the authentication server for storage. As described in section 3.2.2, the Client Agent has to consider the user’s preferences when it is deciding on its next action. For example, this may include the preferred installation path, installation options (full installation, custom installation, minimum installation), whether automatic application update is desired by the user and frequency of other notification messages from the ASP, and additional services customized to preferences.

An Agent Based Application Service Providing System

Extensive information feedback

There must be a way of letting the user know what is happening and what is being done by the Client Agent. Extensive log and display of information is needed. The user should have the assurance of knowing that he can always check the action of his Client Agent, and the Client Agent is not performing anything “secretly”.

Application usage monitoring

The Client Agent also needs to monitor the exact usage of any downloaded applications, for billing purposes.

3.3.2.2 ASP-Side Components

There are 10 components at the ASP side: Usher Agent, ASP Coordination Agent, Authentication Agent, Service Directory Agent, File Transfer Protocol (FTP) Agent, ASP Server Pool Agent, KB Agent, Technician Agent, Information Dissemination Agent, and the Work Coordination Agent. For some agents, like Authentication Agent, Service Directory Agent, FTP Agent, ASP Server Pool Agent, and KB Agent, there is a corresponding server which they are residing on.

Usher Agent and ASP Coordination Agent

The ASP Coordination Agent's main tasks are for the ASP to interact with all other components external to the ASP (including ISV, other ASPs, proprietary payment systems, will be discussed in section 3.2.2.3), not including the

An Agent Based Application Service Providing System

Client Agents. This interaction with external components are very few and occasional. Thus, the ASP coordination agent also takes care of any coordination that is required among the internal ASP-side components. Even though the ASP coordination agent could also have interaction with the Client Agents, it is not desirable. Instead, an Usher Agent is used for interaction with Client Agents. The reasons are explained as follows:

The Usher Agent is the access point of all Client Agents when communicating with the ASP. As opposed to the ASP Coordination Agent, this Usher Agent should perform as little work as possible. The consideration is because, at any time, the Usher Agent would be handling a lot of requests from Client Agents. This is going to present a potential bottleneck. Therefore, the message dispatch agent does not do anything whatsoever, except to authenticate the incoming connection requests from Client Agents and to dispatch all received messages to the intended ASP components. Authentication is done in collaboration with the Authentication Agent. In order to speed up the Usher Agent, it does not decide which ASP component is the recipient for each message. The recipient has to be indicated in the message. Therefore, instead of just sending a message to the ASP side and expecting the message to be handled accordingly, the Client Agent must know to which ASP internal component the message must be sent. In this way, message queues at the Usher agent are eliminated, and broken down into smaller message queues at the ASP components.

An Agent Based Application Service Providing System

Authentication Agent

The main purpose of the Authentication Agent is to store all user profiles. For this purpose, it has an Authentication Server attached which provides information storage. User data and preferences collected by the Client Agent are sent to the Authentication Agent, and stored in the Authentication Server. On the other hand, the Client Agent will retrieve the previously stored user profile, in order to provide personalized services. This server-centric storage of user profile enable multi-point access by users, meaning a user can use any machine to connect to the ASP and still receive the same personalized service. The Authentication Agent also collaborates with the Usher Agent to authenticate and subsequently approve a connection request from Client Agent. The types of user information stored in the Authentication Server are: personal information, payment information, history of application usage, and preferences.

Service Directory Agent

The Service Directory Agent resides on the Service Directory Server, which stores all application related information: application parameters, location of application at FTP Server and ASP Server Pool, and application usage history. There should be a software cataloging system, whereby ontology is used to categorize all the available applications and to organize information about the characteristics of available software [51][52] The purpose is to make it easier for users to search for applications. This information will also be requested by Client Agent, during application downloading or remote access to the ASP server pool. Application usage history keeps a record of who are the users

An Agent Based Application Service Providing System

that have accessed an application, the frequency of access and any comment given.

FTP Agent and ASP Server Pool Agent

Users can choose to either download then install an application, or to directly access the application hosted by ASP. The FTP Agent provides the applications for downloading through FTP protocol [53][54], whereas application hosting is done at the ASP Server Pool which consists of many servers that maintain sessions for all the remote accesses.

KB Agent

The KB Agent together with KB Server are the most important parts of the ASP, which enable the Client Agent to perform many automated tasks. To this end, the KB is required to store a collection of all previously encountered events at the customers' PCs. There are a few types of events, or cases, including application installation dialog boxes, application warning dialog boxes, errors, and also user queries about how to perform certain actions like resizing a photograph, etc.

Whenever a new event is encountered by the Client Agent, it will inform the technician (through Technician Agent). The technician will handle the event. The solution procedures are automatically recorded into the KB by the Technician Agent. This new event and the corresponding solution will be saved as a new case in the KB server. Gradually, the KB builds up to comprehensively include more cases from different users. Then, when a

An Agent Based Application Service Providing System

previous case is again encountered, the correct solution procedures are already available in the KB. The Client Agent will fetch the solution procedures and automatically fix the problem by performing the procedures.

We need a clear definition of how to describe cases, how to describe the solution procedures and how to manage the organization of the knowledge. This will be discussed in section 3.3.4.

Technician Agent

When a help request is received from the Client Agent, the Technician Agent alerts a human technician. An actual screenshot of the problematic dialog box is sent by the Client Agent to the Technician Agent. Through the Technician Agent, the human technician views the screenshot, and performs the necessary mousekeyboard actions. Those actions are automatically recorded and formatted to conform to the KB server solution procedure format. Finally, this technician recommended solution is sent to the Client Agent for execution. A new use case is created in the KB server for recording this dialog box problem solution. The combination of KB Agent and Technician Agent allows remote application installation by the Client Agent, which save a lot of time and traveling otherwise. Eventually, we believe this system can be extended to include other troubleshooting processes.

Information Dissemination Agent

The role of the Information Dissemination Agent is to provide automatic notification services. This is essentially generation of emails for informing

An Agent Based Application Service Providing System

users about billing, application usage dateline, reminder for downloaded application deletion after dateline, etc. A system that notifies users about application updates is applied. But, users have the option of choosing whether this notification process is an information push style or pull style. This is because some users may not appreciate receiving update notifications frequently. Also, a recommendation system is employed, whereby new or previously-unused applications are recommended to users. Collaborative filtering mechanism is used in the recommendation [55]. The principle is that, applications that are used and highly-rated by a group of users with similar user characteristics (occupation, age, etc), are recommended to another similar user who has never tried that application before. Thus, application recommendations can be broadcast to those what will be potentially interested in it, without annoying other users [56].

Work Coordination Agent

Some ASP components need to serve multiple Client Agents for a “significant period”, where “significant period” is defined as processes that need to perform multiple successive interactions with the client. For example, a process that authenticates a user or retrieves user information (from authentication server) is short-lived, since this process only involves a single message exchange with the client. For sessions like query for technician support, it will involve multiple successive interactions between the technician and the Client Agent over a “significant period”. Thus, many Technician Agents should be available for each technician support session in order to efficiently serve multiple clients

An Agent Based Application Service Providing System

concurrently. This is also the case for ASP Server Pool, where sometimes the limited number of application-hosting-servers is receiving many access requests at once. Therefore, Work Coordination Agents is present to distribute the client requests evenly to the Technician Agents or ASP server pool.

3.3.2.3 External Components

There are three external components that may interact with an ASP. These are: ISVs, other ASPs, and external payment systems. Interaction is through the ASP coordination agent. ISVs need to have a channel to add or upgrade applications they previously provided for hosting at the ASP. Also, an ASP may collaborate with other ASPs for offering a wider range of applications, since different ASPs may offer different applications. For payment purposes, an ASP will use the services of external professional payment systems. This is done through payment gateways built by legal professional organizations like credit card companies and banks. Upon receiving the correct payment information, the payment gateway performs transactions between the ASP's bank and the customer's bank.

3.3.3 Communication among the Agents

In section 3.3.2, we have designed the functions of each agent. In this section, we will focus on the communication among those agents. We divide this section into two parts. The first is the communication protocol design. The second is the communication message format.

An Agent Based Application Service Providing System

3.3.3.1 Communication Protocol

The communication protocols between the agents are specified in Table 1. The objective of the design is to use as little bandwidth as possible. For normal communications, like the communication between the Client Agent and the Usher Agent, we use connectionless HTTP protocol. This is because the main function of the Usher Agent is to forward the request of the Client Agent to other components at the ASP side. There is no need to have a session between the Client Agent and the Usher Agent. From Table 1, we can see that the communication protocol between the ASP Server Pool Agent and the Client Agent is SOCKET. The reason behind it is that the ASP Server Pool Agent provides application hosting services. The Client Agent needs continuously send the events to the ASP Server Pool Agent, and the ASP Server Pool Agent also needs to send the corresponding results back. Thus the communication between Client Agent and ASP Server Pool Agent will be very frequent. In this case, ASP Server Pool Agent will create a particular session for each client, as long as he is still using the hosted service.

An Agent Based Application Service Providing System

	Client Agent	Usher Agent	ASP Server Pool Agent	Information Dissemination Agent	Authentication Agent	Service Directory Agent	FTP agent	Knowledge-base Agent	Technician Agent
Client Agent	-	HTTP	SOCKET	-	-	-	-	-	-
Usher Agent	HTTP	-	HTTP	HTTP	HTTP	HTTP	HTTP	HTTP	-
ASP Server Pool Agent	SOCKET	HTTP	-	-	-	-	-	-	-
Information Dissemination Agent	-	HTTP	-	-	HTTP	-	-	-	-
Authentication Agent	-	HTTP	-	HTTP	-	-	-	-	-
Service Directory Agent	-	HTTP	HTTP	HTTP	-	-	HTTP	-	-
FTP agent	FTP	HTTP	-	-	-	-	-	-	-
Knowledge-base Agent	-	HTTP	-	-	-	-	-	-	HTTP
Technician Agent	-	HTTP	-	-	-	-	-	HTTP	-

Table 3.1: Communication Protocols between Agents

An Agent Based Application Service Providing System

3.3.3.2 Communication Message Format

XML is widely used in messaging in various fields [57][58][59]. In our ABASPM, we use XML to define our message format. Basically, there are two types of messages: Request and Response. Base on this concept, we defined two Document Type Definitions (DTDs). For Request message (as shown in Figure 3.1), we have “type”, “from”, “to”, “body”. The “type” element indicates what kind of Request Message it is. In this way, receiver would be able to know which child elements it should fetch from the message body, and operate on it

```
<?xml version="1.0"?>

<!DOCTYPE Request [
  <ELEMENT Request (type, from, to, body)>
  <ELEMENT type (#PCDATA)>
  <ELEMENT from (#PCDATA)>
  <ELEMENT to (#PCDATA)>
  <ELEMENT body (authentication?, application?, screen?, dialog?)>
  <ELEMENT authentication (username, password)>
  <ELEMENT username (#PCDATA)>
  <ELEMENT password (#PCDATA)>
  <ELEMENT application (#PCDATA)>
  <ELEMENT screen (#PCDATA)>
  <ELEMENT dialog (static_text, edit_box, list, button)>
  <ELEMENT static_text (x_coordinate, y_coordinate, caption, style)>
  <ELEMENT edit_box (x_coordinate, y_coordinate, caption, style)>
  <ELEMENT list(x_coordinate, y_coordinate, caption, style)>
  <ELEMENT button (x_coordinate, y_coordinate, caption, style)>
  <ELEMENT x_coordinate (#PCDATA)>
  <ELEMENT y_coordinate (#PCDATA)>
  <ELEMENT caption (#PCDATA)>
  <ELEMENT style (#PCDATA)>
]
>
```

Figure 3.3: Request Message Format

Figure 3.3 is our DTD for Response message. In this DTD, there is a “status” attribute for Response. It indicates the availability of the application service or case bases in the ASP side. For example, when user searches for some software, and if it is not available at the ASP server, the Service Directory Agent will response a message with the “status” “not available”.

An Agent Based Application Service Providing System

```

<?xml version="1.0"?>
<!DOCTYPE Response [
  <ELEMENT Response (type, from, to, body)>
  <ELEMENT type (#PCDATA)>
  <ELEMENT from (#PCDATA)>
  <ELEMENT to (#PCDATA)>
  <ELEMENT body (url?,ftp?, action?)>
  <ELEMENT url (#PCDATA)>
  <ELEMENT ftp (ip, port, username, password)>
  <ELEMENT ip (#PCDATA)>
  <ELEMENT port (#PCDATA)>
  <ELEMENT username (#PCDATA)>
  <ELEMENT password (#PCDATA)>
  <ELEMENT action (left_monse_up, right_monse_up, left_mouse_drag, keyboard)>
  <ELEMENT left_mouse_up (#PCDATA)>
  <ELEMENT right_mouse_up (#PCDATA)>
  <ELEMENT left_mouse_drag (#PCDATA)>
  <ELEMENT keyboard (#PCDATA)>
  <!ATTLIST Response status CDATA #REQUIRED>
]
>

```

Figure 3.4: Response Message Format

3.3.4 Knowledge Base Design

We mainly adopt Case-based reasoning [60][61][62] for our KB. The basic theory of Case-based reasoning is that past experiences could be retrieved and reused when a similar situation occurs. In this section, we are going to present the theory of Case-based reasoning and how we adopt it into our model.

3.3.4.1 Case-Based Reasoning - Theory

Figure 3.5: The Case-Based Reasoning Model

An Agent Based Application Service Providing System

Case-Based Reasoning (CBR) is an approach to develop knowledge-based systems that are able to retrieve and reuse solutions that have worked for similar situations in the past. The first and most important prerequisite is a collection of experiences, embodied in so-called cases, and stored in a case base. Every single case consists at least of a problem description part, called the problem, and a solution part, called the solution. In addition to the case base, there may be some general knowledge in the form of models or rules or constraints available and used. This will be discussed in section 3.3.4.3

In 1995, Macher, Balachandran and Zhang brought forward the case-based reasoning model [60]. Problem solving with CBR proceeds as follows (Figure 3.5): a new problem comes into existence and is described as the problem part of a new case, sometimes also called the query. Then, old cases containing problems that are similar to the new problem are retrieved and the most suitable one of their solutions is suggested to become the solution of the new problem. This solution is then tested in reality which may lead to a revised solution worth to be stored as a new case. This last step is a form of incremental learning that enables CBR systems to adapt to changing environments rather smoothly.

The overall algorithm for case based reasoning can be summarized as:

```

begin
  get new problem specifications;
  identify indexing attributes
  retrieve a set of cases that match attributes;
  select one case:
  repeat
    modify case;
    evaluate solution;
  until solution is satisfactory;
end

```

An Agent Based Application Service Providing System

1. Case Base

Developing a case base includes a definition of each cases and the organization of the cases.

The content of each case

The content of each case is basically a description of the previously solved problem. In order for the cases to be useful in solving the new problem, a description of a case should include a representation of the problem and solution. In addition to that, case memory should also include knowledge for identifying a similar situation.

The organization of the cases is the way that the cases are organized.

2. Recalling cases

Recalling a case from case memory is a pattern-matching problem that is based on the specification of a new problem.

Organize Specification Features. There are two tasks in organization of the specification features. First is to identify features of specification, and the second is to assign a weight to each feature according to their importance in identifying a unique dialog.

3. Retrieve

The retrieval task in CBR searches case base for matches can be based on perfect match, where the pattern is found exactly, or on partial matches. If

An Agent Based Application Service Providing System

partial matches are retrieved, a threshold will be set to determine when a partial match is close enough. Otherwise, if there is no case in case base which is above the threshold, the problem will be handled by other methods. The approach to case retrieval is given below:

```
begin
  get pattern for matching;
  determine relevant cases for comparison;
  repeat for all relevant cases
    determine how close the case matches the pattern;
    if case match above threshold;
      then add case to retrieved list;
  output list of retrieved cases;
end
```

4. Select

The selection task determines which case is the best match. If it is a perfect match, then the corresponding solution is selected immediately. Otherwise, if it is a partial match, the highest “sum of product” of features and weights is selected.

5. Adapting cases

Adapting a case from case base to solve a new problem requires additional knowledge.

3.3.4.2 Case-Based Reasoning – Adoption to our model

Our KB is designed to provide software application installation service, technical support and customized application services.

An Agent Based Application Service Providing System

Classic CBR was proposed under the assumption that a huge case base was already available. Thus, when a new problem is encountered, normally a similar case could be recalled from the case base, if not exactly the same case.

However, such assumption is too idealistic. Hence, in our CBR proposal, an initial empty KB is considered a default precondition.

Figure 3.6: CBR for ABASPM

There are five steps for reusing a case or creating a new case:

1. When a new problem is encountered, we firstly recall similar cases from our case base. This step is done by browsing our case base catalogue. It is a manual step done by client user.
2. If there is no similar case in the case base, a new case must be created. The representation of the case is as follows: 1) Problem description: the problem is described by the client user. For easy search, keyword description is used. For example, a client user wishes to enter student marks and calculate their average.

An Agent Based Application Service Providing System

The keyword description of this problem could be: Excel, Average; 2) Solution: the solution of the problem is the sequence of windows events, such as the mouse clicks and keyboard inputs done by the client user. Please note that the mouse click is not recorded by the position where the mouse down event occurs. Instead, the control where mouse clicks together with its parent window ID is stored. This is because the relative position might be different on different PCs for different computer settings, while the windows ID can uniquely identify a window.

In our CBR, we use a sequence of windows events as our problem solution procedure. We call this sequence an Action Flow. The Client Agent interprets the Action Flow and performs accordingly, while ignoring the description. The XML format description allows new preference attributes to be easily created and added to the Action Flow. Client agent can then choose the desired mousekeyboard actions based on values of its user's preferences. An example of the preference attributes is "installation path".

The window ID used together with the problem solution uses properties of all the controls in a dialog box. Those properties may include control caption, position, and type of control. We stored them in a text stream which is comparably very small in size and efficient in uniquely identify a dialog.

Some proposes to use screenshots to represent a dialog in some similar implementations. There are two main disadvantages. One is that screenshots are usually very large in terms of file size, which will bring a lot of network problem. Secondly, screenshots are very dependent on Windows setting.

An Agent Based Application Service Providing System

Client users might have different setting on their PCs. For example, some might set resolution to be 1024x768 and some 800x600. This raises a problem. Same dialog might look different on different PCs. So same problem solved for 1024x768 clients can not be used on 800x600 clients.

Some others propose a Hash-Window concept to identify screenshots of dialog boxes. The main advantage of this concept is that it greatly reduces the size for screenshots. In the Hash-Window concept, first, a screenshot of the dialog box is captured by the client agent. Instead of directly sending the screenshot to the KB server as the identification symptom, a hash code is generated from the screenshot, by using a one-way hash function. The properties of a hash function are such that it is almost impossible to have two screenshots that produce the same hash code. **So**, the unique hash code for each screenshot will serve as a “tag” for that screenshot. The client agent does not have to send the screenshot to the KB server for match comparison; instead it just sends the hash code, which is very much smaller than the actual screenshot. The RIPEMD-160 or SHA-1 hash functions always yield 160-bit hash codes [63][64]. However, the problem discussed in previous method still exists in this Hash-Window concept.

3. If a similar case could be found in the case base, it will be retrieved. The problem description and solution will be presented to the user. Each operation of the solution will be listed. Details of the operation such as the window name, object (control) type, object caption and event are displayed to help the user to identify each operation. One big problem faced in case retrieval is how to make an existing case solution feasible to new problems, which is called adaptation

An Agent Based Application Service Providing System

problem. In the next chapter, we raise a simple example to illustrate this problem. As for adding new profile into Outlook, every user wants to use his/her own user name, while the rest of the operations of tasks are just the same. In our proposal, a customization option is provided to users to let them modify a particular step only according to their own preference without recording a new solution all over again. Thus, the solution is adapted for the new problem.

4. When a new case is created and stored in the case base, our KB agent will check the similarity between this new case and the rest of the cases in the case base. If there is overlap between any two cases, our KB agent will inform the client user on this. Based on the consideration whether the overlap portion is potentially useful to future cases, client user can choose whether the overlap portion should be extracted into a new case.

5. When a new case is adapted from a similar case, the adapted solution together with some environment characteristics such as the user name and Operating System (OS) version are saved back into the KB. In the future, during the case retrieval, the solution is generated according to the particular user's preference and the environment characteristics.

Our proposed CBR system can work well with problems which need less human intervention. For example, people might need to work on different PCs. The procedure of setting up the environment according to user's preference is very tedious. As for setting up Microsoft Outlook, one needs to memorize and fill in the address of the Exchange Servers, set the font and signatures, and

An Agent Based Application Service Providing System

add rules to process the received emails, etc. By using our system, one only needs to do all the settings once and they will be applied to all his/her PCs by one click.

To illustrate our CBR model, we give a simple example. Suppose a client user request to install Acrobat Reader, a dialog asking you to choose the installation folder may look like Figure 3.7. For the inexperienced user, he may not sure about which folder he should put in. In this case, the representation of problem and solution will be as follows:

Figure 3.7: A Dialog in Installation Process

An Agent Based Application Service Providing System

Windows ID is described as follows:

```
<dialog>
  <static-text>
    <caption>Please choose the installation folder.</caption>
    <x-coordinate>50</x-coordinate>
    <y-coordinate> 100</y-coordinate>
  </static-text>

  <static-text>
    <caption>Path:</caption>
    <x-coordinate>50</x-coordinate>
    <y-coordinate> 150</y-coordinate>
  </static-text>
  .....
  <edit_box>
    <x-coordinate>50</x-coordinate>
    <y-coordinate>180</y-coordinate>
  </edit_box>

  <button>
    <caption>ok</caption>
    <x-coordinate>300</x-coordinate>
    <y-coordinate>250</y-coordinate>
  </button>
</dialog>
```

The action flow is stored as follows:

```
<action>
  <left_mouse_up>
    <x-coordinate>51</x-coordinate>
    <y-coordinate>181</y-coordinate>
    <wait_milliseconds>1000</wait_milliseconds>
  </left_mouse_up>

  <keyboard>
    <ASCII_string>C:\Program files\Acrobat\Acrobat 5.0</ASCII_string>
    <wait_milliseconds>1000</wait_milliseconds>
  </keyboard>

  <left_mouse_up>
    <x-coordinate>301</x-coordinate>
    <y-coordinate>251</y-coordinate>
    <wait_milliseconds>1000</wait_milliseconds>
  </left_mouse_up>

  <user_id>
    <first_name>Bing</first_name>
    <last_name>Li</last_name>
  </user_id>
</action>
```

Thus, the problem solution is the combination of Windows ID and action flow.

3.4 Key Functionalities of the Proposed Model

Compatibility with Current ASP Model

As we know, whenever a new technology comes to real application, compatibility is an important factor which determines whether this technology is able to survive or not. That means this new technology must be compatible with the current technology people have been using all along. This is because most end users need some time to transit from the old to the new system.

Our ASP model is designed to be compatible with the current server-based application hosting ASPs.

Automatic Application Installation

As we discussed in previous sections, most common PC users, who treat applications merely as convenient tools, do not wish to deal with the details of application installation, configuration and upgrades. In our model, Client Agent takes care of those tasks by taking the corresponding actions from KB. Being autonomous in nature, our agent is able to take over all the tasks with the least interference with the end user. In comparison, Mena's ASP model cannot provide automatic application installation service, although it also makes use of agent technology. As Mena's model is specifically designed to provide software retrieval service, it is not able to provide customized application services.

An Agent Based Application Service Providing System

Real Time Technical Support

Efficient technical support is very important for a good ASP model. Current ASP models can not provide real time technical support, while our proposed agent based ASP model is able to do so. The details of our real time technical support can be found in the following paragraphs.

Faster Implementation of Application Services

Automatic application installation and real time technical support are only some of the most important services that our ASP model can provide. Besides that, some services, such as application selection service, electronic commerce interaction service, might also needed for some client users. Our model is able to implement these services rapidly without having to implement each of the services from scratch.

Synchronization

Data synchronization is very important in our ASP model. Unlike the normal ASP approach, which only provides hosted application services at the server side, our ASP model provides both server-based application hosting service and desktop-based application downloading service. Thus data must be consistent in both services. In this model, we apply version control and some other rules to ensure data synchronization.

Knowledge Base Approach

Generally, the primary objective of having a Knowledge Management (KM)

An Agent Based Application Service Providing System

system is for an organization to generate value from what they have already known. Past experiences and knowledge are stored in knowledge Bases, so that this knowledge could be utilized in the future when the same circumstances arise. We extend this concept to our proposed agent system. A knowledge-based approach is heavily deployed in our model, in order to add the element of automation to various agent tasks.

Unlike the model proposed by Mena in chapter 2, our model is applying the Knowledge Base approach. In Mena's agent based service retrieval system, they use the normal approach in getting the application service ready. While in practice, an ASP may need to provide some of the application services. In this case, it will be very troublesome and time consuming to implement one by one. In our design, once a skeleton application is implemented, a new application service can be implemented pretty fast. We can implement new modules, make use of existing modules and tell our Knowledge Base Server the way to combine those modules to form the application. In this way, the time to implement duplicated modules is eliminated.

Another important function of Knowledge Base Server covers agent-assisted application installation, configuration and troubleshooting. In our ASP model, initially all technical support procedures have to be performed by humans, i.e. technicians, to solve client's problems. We exploit the fact that these procedures are always repeated in the future when the same problem occurs. Therefore, it is possible to record these procedures in a knowledge base, so that in the future when the same or similar problem is encountered, an agent could

An Agent Based Application Service Providing System

identify the problem and access the knowledge base to retrieve the corresponding solution procedures to be performed. This knowledge base is essentially the condition-action rules that provide the element of automation to agents. As more and more new problems are solved by human technicians and recorded, the knowledge base will grow and become more comprehensive and efficient. We believe this approach is useful in application installation, configuration, troubleshooting, as well as in providing customized applications. In our model, the knowledge base server is the repository for stored procedures.

We have also proposed a novel Knowledge Re-Organization Algorithm. By applying the proposed algorithm, our cases will be “atomic” as we have more and more cases in our KB. In this way, reusability of old cases is increased and finally when we have “enough” cases in our KB, our goal of providing fully automatic services will be almost achieved.

3.5 Comparison to Related Works

There are some related models proposed by other research groups or organizations. A comparison between these models and our proposed ABASPM is made.

3.5.1 Methods for Accessing ASP Services

Today most ASPs are adopting server-based technology. Users normally access

An Agent Based Application Service Providing System

ASP services through Web browsers.

Our ASP model provides both server-base application hosting service and desktop-based application downloading service. Thus, user can either use Web browsers to access ASP services or install the application service to his/her local PC. What's more, the tedious application installation, configuration and upgrades are taken care of by the client agent.

3.5.2 Types of Services Provided

Some ASP models can only one or several types of services. For example, Hacigumus's NetDB2 Model can only provide database service. Mena's model discussed in section 2.5.2 can only provide software retrieval service. And some ASP models can only provide software applications provided by the third party.

Our proposed ASP model can provide various application services, including applications provided by the third party, automatic application installation services and intelligent personal assistant services. Some prototype implementations can be found in chapter 4.

3.5.3 Efficiency of Technical Support

One of the main reasons that SMEs adopt ASPs is because it eliminates the need to employ expensive administration and support staff to operate various technical supports. For example, software installations and upgrading, complex software usage support, etc. However, the common way of providing this kind

An Agent Based Application Service Providing System

of technical support is still inefficient. They usually give a telephone call to the ASP center and ASP technical staff will either tell users what to do in the phone call or come down to the customer side personally. This is very time consuming. And sometimes misunderstandings may occur when they talk by telephone line.

Our proposed agent based ASP model is able to provide real time technical support by storing all the knowledge in the KB. This greatly reduces time needed to provide support and eliminates the misunderstandings that may occur when they talk over telephone line.

3.5.4 Time Needed to Implement a Software Service

Unlike the ASP models discussed in chapter 2, our model is applying the Knowledge Base approach. No matter Mena's agent based service retrieval model, Furht's ASP Model, and Hacigumus's NetDB2 Model, they all use the normal approach in getting the application service ready. While in practice, an ASP may need to provide quite some of the application services. In this case, it will be very troublesome and time consuming to implement one by one. In our design, once a skeleton application is implemented, a new application service can be implemented pretty fast. We can implement new modules, make use of existing modules and tell our Knowledge Base Server the way to combine those modules to form the application. In this way, the time to implement duplicated modules is eliminated

An Agent Based Application Service Providing System

3.5.5 Personalization and Customization

Multiple clients connected to the ASP would have different preferences. Each client might require personalized services. However, for service-based application hosting system this is difficult to do, because applications are hosted at the server-side, and it is not feasible to host thousand of different version of applications to fulfill each customer's needs.

In our proposed ABASPM, the Client Agent is able to store user preferences. In this way, application services can be provided according to each user's preference. On the other hand, our agent system together with the KB, are able to provide customized applications. For example, intelligent personal assistant, which is a highly customized application, could be provided.

Chapter 4

Applications and Implementations

In previous chapters, we have given an intensive overview of other ASPs and their related technologies. We have also explained how we could apply agent technology in an ASP model and given our proposed agent based ASP model. In this chapter, we are going to discuss about possible applications that could be implemented based on our agent based ASP model. Among all applications, “Auto Application Installation” application and “Intelligent Personal Assistant” application will be discussed in detail together with our prototype implementation.

4.1 Auto Application Installation

As we have discussed in previous chapter, we believe that in order to provide more “available” services, we decided to allow software applications to be installed in the client user’s PC. In this way, end users will not be relying too much on the Internet connection. Instead, they can work on the applications they subscribed offline. However, this arrangement raises another problem – users must install the software in their own PC, which is found tedious and troublesome for most of PC users.

An Agent Based Application Service Providing System

In order to overcome this problem as a result of making services more “available”, we designed and developed Auto Application Installation (AAI) application based on our proposed model. When a user chooses an application provided by ASP, the chosen application is then downloaded and installed on the user’s PC with the help of the Client Agent.

By applying the agent based ASP model, the scenario of getting the automatic application installation service is as follows:

A client user first chooses an application provided by ASP. The chosen application is then downloaded to the user’s PC. If this is the first time for the ASP server to install this application, Technician will do the installation for the client. And KB Agent will record the installation procedure automatically to the KB. Then next time when the same application is requested, Client Agent is able to fetch the installation procedure from KB and install it to the client PC automatically.

4.1.1 Objective

By developing this application service, we aim to provide a way to automate the application installation process for our client users.

4.1.2 Implementation

In this section, we will demonstrate our “Automatic Application Installation” Service prototype. We are going to use “MSN Messenger” as a demo

An Agent Based Application Service Providing System

application. The scenario of the demo is as follows: client user A requests to download and install “MSN Messenger”. He firstly selects this application. Then “MSN Messenger” is downloaded to his local PC. Assume this is the first time the ASP Server installs this application, i.e. the application installation procedures are not stored in the KB. Technician in ASP Server will be alerted and requested to do the installation for client user A. Then client user A’s application installation dialog will be transferred to the technician’s desktop. Technician will take over the whole installation process by sending the mouse click and keypad input to the client PC. The whole process is transparent to the client user. Next time when client user B wants to install the same application, his client agent is able to fetch the application installation procedures from KB and does the installation automatically. The sequence diagrams of all the use cases are attached in the appendix at the end of the thesis.

The User Interface (UI) of client logging in page is shown in Figure 4.1. After successfully logged on to ASP, client user A is able to choose services provided by ASP. In this demo, only automatic application installation service is provided (Figure 4.2). Click on the “Launch Service” button, a dialog is then popped up to allow user select applications available in our ASP. “MSN Messenger” is selected in this demo (Figure 4.3).

An Agent Based Application Service Providing System

Figure 4.1 Client Agent – Log on

Figure 4.2: Client Agent – Service Selection

An Agent Based Application Service Providing System

Figure 4.3: Client Agent – Application Selection

Figure 4.4: Client Agent – Application Downloading

An Agent Based Application Service Providing System

After “MSN Messenger” application is selected, “MSN Messenger” set up file is then downloaded to the client PC through FTP protocol. And it will run at the client PC automatically (Figure 4.5). Client Agent will then try to fetch the installation procedures from ASP. But since it is the first time our ASP installs “MSN Messenger”, the solution is not available in our KB. A message box is then popped up asking client user whether he wants to seek help from our technician agent (Figure 4.6). By clicking “No”, client user A chooses to do the installation by himself. And clicking “Yes”, his installation dialog will then be transmitted to the technician at ASP Side (Figure 4.7).

Figure 4.5: Application Installation – Dialog 1 (Client Side)

An Agent Based Application Service Providing System

Technician Agent runs at the ASP server all the time. It persistently listens to incoming messages. When a new message comes, a new task dialog will pop up to alert technicians. In this case the dialog shows “user Li Bing requests to install MSN Messenger” (Figure 4.7). Technician will then click “Yes” to take over the task. The MSN installation dialog 1 shown in Figure 4.5 will be transmitted to the PC where Technician Agent resides. Technician opens Technician Agent’s menu and starts recording case (Figure 4.8).

Figure 4.8: Technician Agent – Start Recording Case

The mouse movement and keyboard input in Figure 4.8 – 4.12 are subsequently sent to the client PC. And the installation process is finished by clicking “Finish” button in the last installation dialog (Figure 4.12). Technician can now stop recording case (Figure 4.13). A pop up window will appear at the

An Agent Based Application Service Providing System

technician's PC asking him whether he wants to save the case (Figure 4.14). By clicking "Yes", the case is saved in the KB for future uses.

Figure 4.9: Technician Agent – Application Installation Dialog 2

An Agent Based Amlication Service Providine Svstem

Figure 4.10: Technician Agent – Application Installation Dialog 3

Figure 4.1 1: Technician Agent – Application Installation Dialog 4

An Agent Based Application Service Providing System

Figure 4.12: Technician Agent – Application Installation Dialog 5

Figure 4.13: Technician Agent – Stop Recording Case

An Agent Based Application Service Providing System

Figure 4.14: Technician Agent – Save Case

Assuming now client user B would like to download and install “MSN Messenger” now. His Client Agent is able to fetch the installation solution in KB and do the installation automatically. See the sequence of screenshots obtained from client user B’s desktop (Figure 4.15 – 4.19).

An Agent Based ApplicationService Providing Svstem

Figure 4.15: Client Agent B – Application Installation Dialog 1

Figure 4.16: Client Agent B – Application Installation Dialog 2

An Agent Based Application Service Providing System

Figure 4.17: Client Agent B – Application Installation Dialog 3

Figure 4.18: Client Agent B – Application Installation Dialog 4

An Agent Based Application Service Providing System

Figure 4.19: Client Agent B – Application Installation Dialog 5

In this way, application installation procedures are reused for future cases. As time goes on, our KB grows together with more and more application installations are encountered. In the end, our system will be fully automated. Most of or commonly used applications can be automatically installed to the client user's PC. And less manpower is required at the ASP side to support this kind of service.

4.2 Intelligent Personal Assistant

The concept of Intelligent Personal Assistant (IPA) has been proposed at the end of last millennium. Since then many Artificial Intelligence (AI) researchers

An Agent Based Application Service Providing System

have been working on it, and a lot of PA models have been proposed [65]-[69].

IPAs are software agents that can represent individuals. They help users in their day-to-day activities, such as prepare your morning newspaper, help you surf net by filtering out junk advertisements, keep searching the web day after day for things you want, participate in on-line actions, learn your interests and bring you back valuable information, take care of repetitive chores, answer e-mail, negotiate the date and time of a meeting, and more.

Already there are simple personal agents to advise us on how to use particular software products, and arrange meetings within corporate workgroups. However, most of these agents cannot take more than one aspect of our activities into account, nor do they adapt easily to our preferences.

In this section, we will present an IPA implementation. We use this application to illustrate that the CBR for ABASPM proposed in chapter 3, which can be used in IPA design. Together with our proposed agent based ASP model, personalized application service can then be provided by our model.

4.2.1 Objective

The objective of IPA application is to provide personalized application service to end users. The objective of the prototype application is to demonstrate the feasibility of the IPA application.

An Agent Based Application Service Providing System

4.2.2 Implementation

Our IPA is shown in Figure 4.20. It is running in the system tray. It has two main functions: 1) Load cases; 2) Start recording. By clicking “Load Cases”, old cases stored in the ASP KB will be retrieved. You can then browse the case catalogue. If there is similar case available, you can simply use it or adapt it if needed. Otherwise, if it is not available in the JSB, you may click on “Start Recording” and record your own case.

Assume that user A wants to add new Microsoft Outlook user profile. He can record it to the KB, so that it can be reused later. After clicking on the “Start Recording” menu, he can now start setting up his profile.

Figure 4.20: **IPA** – Start Recording

An Agent Based Application Service Providing System

The user profile will then be set up by user A, as shown from Figure 4.21 – 4.29.

Figure 4.2 1:IPA – Recording 1

Figure 4.22: IPA - Recording 2

An Agent Based Application Service Providing System

Figure 4.23: IPA – Recording 3

Figure 4.24: IPA – Recording 4

An Agent Based Application Service Providing System

Figure 4.25: IPA -Recording 5

Figure 4.26: IPA – Recording 6

An Agent Based Application Service Providing System

Figure 4.27: IPA – Recording 7

Figure 4.28: IPA – Recording 8

An Agent Based Application Service Providing System

Figure 4.29: **IPA** – Recording 9

Figure 4.30: IPA – Stop Recording

An Agent Based Application Service Providing System

Figure 4.3 1:IPA - New Case 1

Figure 4.32: New Case - New Case 2

An Agent Based Application Service Providing System

Figure 4.33: New Case - New Case 3

After all the set up is done, user A clicks on the “Stop Recording” menu (Figure 4.30). A new case dialog will pop up. The record operations including keyboard input and mouse click will be displayed to the user. To save the case to the KB, the user must provide the case name and the category it belongs to. For some cases which include personal information like user password, user could set it so that the case could be view by himself only (Figure 4.31 - 4.33). In this case, we assume that user A would like to share the case with others.

Now assume user B would like to set up Microsoft Outlook user profile as well. He will firstly “load cases” (Figure 4.34) and the browse the case catalogue (Figure 4.35). The “add new profile for exchange server” case is then selected, and the list of operations will be displayed to user B. User B will check through the operations and find that he need to customize the “new profile” to his

An Agent Based Application Service Providing System

email user name. He can do so by checking the “new profile” check box. By doing so, the case will run automatically until it reaches the “new profile” operation, where user B could do by himself. After user B finishes verifying his user name, he can simply press Ctr + Alt + F2 to resume the rest of the set up operation.

Figure 4.34: IPA -Load Cases 1

An Agent Based Application Service Providing System

Figure 4.35: IPA – Load Cases 2

Figure 4.36: IPA– Load Cases 3

An Agent Based Application Service Providing System

Figure 4.37: IPA – Load Cases 4

Figure 4.38: IPA - Save to KB

An Agent Based Application Service Providing System

After all the case operation has been executed, a small dialog will pop up asking user if he wishes to save the adapted case, next time when he loads the same case, it will be executed according to his preference. Otherwise, the default user A's preference will be loaded.

4.3 Other Possible Applications

There are many other applications which can be implemented based on our proposed model. Our proposed agent based ASP model best suits applications which have repeated procedures. Our agents have the ability of learning base on past experience and the ability to solve the problem when it is encountered again. Both Automatic Application Installation and Intelligent Personal Assistant have the characteristics of having repeated procedures.

An Example of other possible applications could be Real Time Technical Support. The idea of realizing real time technical support is very much similar to the Automatic Application Installation implementation. But the scope of technical support is much wider than application installation, so the corresponding KB will be much larger.

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

In this thesis report, we firstly give an intensive literature review of ASP, agent, current ASP architectures, and agent applications in service provision. After that we have presented our proposed agent based ASP model. We discussed about the functionality of each components in our model and the communication protocol and message format between them. The key characteristic of the proposed model is the knowledge-based approach. The knowledge bases provide the necessary automation and personalization ingredients for agents. We also proposed a new CBR for our ABASPM, which make our knowledge reusable. Last but not least, we discussed possible applications based on our model, among which two implementations are discussed in detail.

In summary, the main advantages of our proposed agent based ASP model are:

Application Hosting and Application Installation

Our agent based ASP model supports both application hosting and application installation. This is because we believe that most of the time there is a need

An Agent Based Application Service Providing System

for the client to download and install the application at his local computer. One reason would be the security issue. It is because of the fact that client may worry about handing over critical operations and sensitive data to external firms. Secondly, availability is also their concern. Clients may worry that they may not get the connectivity and be able to access the applications when they need to do so. Lastly, when a client knows that he is going to use an application frequently for a specific period, he would prefer to download and install it, rather than remotely accessing the application every time.

Real Time Technical Support

Our agent based ASP system is able to provide real time technical support. Both automatic application installation service and real time technical support are achieved by storing problem description and solution procedures in our KB.

Automation

Most common PC users are inexperienced in the details of application installation, configuration, and upgrades. Thus, automation of all these tasks will be tempting. In our agent based ASP model, the Client Agent is able to handle all these tasks automatically, transparent to the end user. Automatic Application Installation application is implemented to prove the feasibility of this point.

Personalization and Customization

Multiple clients connected to the ASP would have different preferences. Each client might require personalized service. However, in server-based

An Agent Based Application Service Providing System

application hosting systems this is difficult to do, because applications are hosted at the server side, and it is not feasible to host thousands of different versions of applications to fulfill each customer's needs. Our desktop-based application installation service, which is the main part of our proposed model, is able to provide personalized application services to the end users. The IPA prototype is implemented to prove the feasibility of this point.

5.2 Recommendations for Further Research

This research field is a challenging but beneficial field. We achieved the objective of designing an agent based ASP model, to guarantee the availability of ASP services, with a high degree of being autonomous and customization. Some further researches we intend to recommend are listed as follow:

- With the development of Artificial Intelligence (AI) and Natural Language Processing (NLP), the capability of our CBR for ABASPM can be further improved. For example, when NLP is mature, user can use his/her nature language to describe a task instead of the UI we currently proposed.
- Web Services technology can be used to implement the communication and coordination between the agents in the proposed ABASPM. For example, the communications between the agents in the ABASPM can be defined by WSDL. Web services technology can enhance the integrability and compatibility of the proposed model. However, further

An Agent Based Application Service Providing System

research is needed to incorporate the technology to the proposed system.

- The future ASP developers can choose many useful and beneficial domains to develop application services, especially IPA applications which are commonly used in our daily life.

Author's Publication

Y. Miao, **B. Li** and R.K.L. Gay, “An Agent Based Application Service Providing Model”, *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision*, Kunming, China, 2004.

Bibliography

- [1] R. Terdiman, "Evaluating ASPs: Dig Out the Value Propositions", *Gartner Group Research Note*, 2000.
- [2] R. Comerford, "The New Software Paladins", *Spectrum, IEEE*, Vol. 37, No. 6, 2000, pp. 56-61.
- [3] M. Holohan, "Application Service Providers",
<http://www.computerworld.com/softwaretopics/crm/story/0,10801,49858,0,0.html>
- [4] S.J. Shepard, "IT Shops Take Stock of Application Service Providers", *IT Professional*, Vol. 2, No. 2, 2000, pp. 12-16.
- [5] A. Susarla, A. Barua, and A.B. Whinston, "Myths about Outsourcing to Application Service Providers", *IT Professional*, Vol. 3, No. 3, 2001, pp. 32-35.
- [6] W.M. Shen, D.H. Norrie and J.P. Barthes, *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*, Taylor & Francis, 2001.
- [7] M. Wooldridge, *Introduction to MultiAgent Systems*, Chichester, West Sussex: J. Wiley, 2002.
- [8] Jade, <http://jade.tilab.com>
- [9] Application Service Providers: Evolution and Resources, White Paper,

An Agent Based Application Service Providing System

Microsoft Corporation, 2000.

[10] A. Factor, Analyzing Application Service Providers, Prentice Hall PTR, c2002.

[11] “Application Service Provider”,
http://searchwebservicestechtarget.com/sDefinition/0,,sid26_gci213801.00.html

[12] “ASP Island’s Application Service Model”,
<http://www.aspisland.com/services/rmodel.asp>

[13] P. Seltsikas and W.L. Currie, “Evaluating the Application Service Provider (ASP) Business Model: The Challenge of Integration”, *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2002.

[14] S.J. Shepard, “It Shops Take Stock of Application Service Providers”, *IT Professional*, Vol. 2, No. 2, 2000, pp. 12-16.

[15] B. Furht, C. Phoenix, J. Yin and Z. Aganovic, “An Innovative Internet Architecture for Application Service Providers”, *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[16] A. Plepys, “Software Renting - Better Business, Better Environment: The Case of Application Service Provider (ASP)”, *IEEE International Symposium on Electronics and Environment*, 2002.

[17] L.X. Tao, “Shifting Paradigms with the Application Service Provider

An Agent Based Application Service Providing System

Model”, *Computer, IEEE*, Vol. 34, No. 10, 2001, pp. 32-39.

[18] C.H. Sng, “A Framework for Business-to-Business e-Commerce for Application Service Providers”, *International Journal of Information Technology*, Vol. 7, No. 2, 2001.

[19] G. Flammla, “Application Service Providers: Challenges and Opportunities”, *Intelligent Systems, IEEE*, Vol. 16, No. 1, 2001, pp. 22-23.

[20] L. Monoston and B. Kadar, “Agent-Based Control of Manufacturing Systems”, Proceedings of the 2nd International Conference on Intelligent Processing and Manufacturing of Materials, 1999.

[21] P.C. Weinstein, W.P. Birmingham and E.H. Durfee, “Agent Based Digital Libraries: Decentralization and Coordination”, *Communications Magazine, IEEE*, Vol. 37, No. 1, 1999, pp. 110-115.

[22] H. Little and A. Esterline, “Agent-Based Transaction Processing”, *Proceedings of the IEEE Southeastcon*, 2002.

[23] R. Okada, E.S. Lee and N. Shiraton, “Agent-Based Approach for Information Gathering on Highly Distributed and Heterogeneous Environment”, *International Conference on Parallel and Distributed Systems*, 1996.

[24] N. Jennings and M. Wooldridge, “Software Agents”, *IEE Review*, 1996, pp. 17-20.

[25] M. Wooldridge and N.R. Jennings, “Intelligent Agents: Theory and

An Agent Based Application Service Providing System

- Practice”, *Knowledge Engineering Review*, Vol. 10, No. 2, 1995, pp. 115-152.
- [26] J.M. Vidal, P.A. Buhler and M.N. Huhns, “Inside an Agent”, *Internet Computing, IEEE*, Vol. 5, No. 1, 2001, pp. 82-86.
- [27] M. Wooldridge, “Agent-Based Software Engineering”, *IEE Proceedings of Software Engineering*, 1997.
- [28] A.S. Rao and M.P. George, “BDI Agent: from Theory to Practice”, *Proceedings of the 1st International Conference on Multi-Agent Systems*, San Francisco, 1995.
- [29] G Yamamoto and H. Tai, “Agent Server Technology for Next Generation of Web Applications”, *4th International Conference on Computational Intelligence and Multimedia Applications*, 2001.
- [30] N.R. Jennings and S. Bussmann, “Agent-Based Control Systems: Why are They Suited to Engineering Complex System?”, *Control Systems Magazine, IEEE*, Vol. 23, No. 3, 2003.
- [31] R. Okada, E.S. Lee and N. Shiraton, “Agent-Based Approach for Information Gathering on Highly Distributed and Heterogeneous Environment”, *Proceedings of International Conference on Parallel and Distributed Systems*, 1996.
- [32] A. Khamis, M. Aboul-Ela and M. Aywany, “Multi-Agent Collaboration: an E-Commerce Example”, *ACS/IEEE International Conference on Computer*

An Agent Based Application Service Providing System

Systems and Applications, 2003.

- [33] J. Kozma, "Intelligent Agents", *Potentials, IEEE*, Vol. 17, No. 2, 1998, pp. 16-19.
- [34] M.D. Huhms and M.P. Singh, "Personal Assistant", *Internet Computing, IEEE*, Vol. 2, No. 5, 1998, pp. 90-92.
- [35] E. Mena, A. Illarramendi and A. Goni, "A Software Retrieval Service Based on Knowledge-Driven Agents", *Proceedings of the 7th International Conference on Cooperative Information Systems*, 2000.
- [36] E. Mena, A. Illarramendi and A. Gofli, "Customizable Software Retrieval Facility for Mobile Computers using Agents", *Proceedings of the 7th International Conference on Parallel and Distributed Systems*, 2000.
- [37] E. Mena, J.A.Royo, A. Illarramendi and A. Gofli, "An Agent-Based Approach for Helping Users of Hand-Held Devices to Browse Software Catalogs", *Proceedings of the 16th International Workshop on Cooperative Information Agents VI*, 2002.
- [38] S.S. Foster, D. Moore, M.J. Flester and B.A. Nebesh, "Performance Tuning Mobile Agent Workflow Application", *Proceedings of Technology of Object-Oriented Languages and Systems*, 1999.
- [39] C. Zhan, B. Odger and M. Schroeder, "An in-service Agent Monitoring and Analysis System", *Proceeding of 11th IEEE International Conference on Tools with Artificial Intelligence*, 1999.

An Agent Based Application Service Providing System

- [40] C. Panayiotou, G. Samaras, E. Pitoura and P. Evripidou, “Parallel Computing Using Java Mobile Agent”, *Proceeding of 25th EUROMICRO conference*, 1999.
- [41] “Windows 2000 Terminal Service Technical Overview”,
<http://www.microsoft.com/windows2000/technologies/terminal/default.asp#section8>
- [42] “Remote Desktop Protocol”,
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/termserv/termserv/remote_desktop_protocol.asp
- [43] T. Okuda, M. Takano, K. Tajima, S. Shimojo, S. Yamaguchi and H. Miyahara, “Realizing Continuous and Transparent Service Using Agents”, *IEEE Pacific Rim Conference on Communication, Computers and Signal Processing*, 2001.
- [44] Corio, <http://www.corio.com>
- [45] SAP, <http://www.sap.com>
- [46] FutureLink, <http://futurelink.com>
- [47] H. Hacigumus, B. Iyer and S. Mehrotra. “Providing Database as a Service”. *Proceeding of the 18th International Conference on Data Engineering*, 2002.
- [48] V. Carchiolo, M. Malgeri and G.. Mangioni, “An Agent Based Platform for a Service Provider”, *Proceedings of 2nd IMACS International*

An Agent Based Application Service Providing System

Conference on Circuits, Systems and Computers, 1998.

- [49] M. Berger, B. Bauer and M. Watzke, "Towards an Agent-Based Infrastructure for Distributed Virtual Organizations", *Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2001.
- [50] E.A. Kendall, P.V.M. Krishna, C.B. Suresh and C.G.V. Pathak, "An Application Framework for Intelligent and Mobile Agents", *ACM Computing Surveys*, Vol. 32, No. 1, 2002.
- [51] C.M. Papaterpos, N.P. Georgantis and T.S. Papatheodorous, "An Ontology for Modeling Ill-Structured Domains in Intelligent Educational Systems", *Proceedings of IEEE International Conference on Advanced Learning Technologies*, 2001.
- [52] Y. Tzitzikas, N. Spyrtos and P. Constantopoulos, "Mediators over Ontology-Based Information Sources", *Proceedings of the 2nd International Conference on Web Information Systems Engineering*, 2001.
- [53] "FTP Protocol Overview", <http://www.freesoft.or,g/CIE/ToPics/69.htm>
- [54] "File Transfer Protocol (FTP)", <http://www.w3.org/Protocols/rfc959/>
- [55] P. Maes, R.H. Guttman and A. G. Moukas, "Agents That Buy and Sell", *Communications of the ACM*, Vol. 42, No. 3, 1999. pp. 81-91.
- [56] S. Case, N. Azarmi, M. Thint and T. Ohtani, "Enhancing E-Communities

An Agent Based Application Service Providing System

with Agent-Based Systems”, *Computer*, Vol. 34, No. 7, 2001, pp. 64-69.

[57] L. Ye, Y. Luo and M. Nagata, “XML Based Message Queuing”, *IEEE International Conference on Systems*, 2000.

[58] B. Zhang, K. Ding and J. Li, “An XML-Message Based Architecture Description Language and Architectural Mismatch Checking”, *25th International Computer Software and Applications Conference*, 2001

[59] P. Widener, G. Eisenhauer and K. Schwan, “Open Metadata Format: Efficient XML-Based Communication for High Performance Computing”, *Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing*, 2001.

[60] M.L. Maher, M.B. Balachandran and D.M. Zhang, *Case-Based Reasoning in Design*, Mahwah, N.J.: Lawrence Erlbaum Associates, 1995.

[61] B. Bartsch-Sporl, M. Lenz and A. Hubner, “Case-Based Reasoning - Survey and Future Directions”, *Knowledge-Based Systems-Survey and Future Directions*, 1999

[62] M.M. Richter, “The Knowledge Contained in Similarity Measures”, *Invited Talk at International Conference on Case-Based Reasoning*, 1995.

[63] H. Dobbertin, A. Bosselaers and B. Preneel, “RIPEMD-160-A Strengthened Version of RIPEMD”, *Fast Software Encryption, 3rd International Workshop*, 1996.

[64] “Hash Function RIPEMD-160”,

An Agent Based Application Service Providing System

<http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html>

- [65] M.N. Huhns and M.P. Singh, "Personal Assistants", *Internet Computing, IEEE*, Vol. 2, No. 5, 1998, pp. 90-92.
- [66] J. Kozma, "Intelligent Agents", *Internet Computing, IEEE*, Vol. 17, No. 2, 1998, pp. 16-19.
- [67] F. Enembreck and J.P. Barthes, "Personal Assistant to Improve CSCW", the 7th International Conference on Computer Supported Cooperative Work in Design, 2002.
- [68] J.Y.C. Pan and J.M. Tenenbaum, "An Intelligent Agent Framework for Enterprise Integration", *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 21, No. 6, 1991, pp. 1391-1408.
- [69] S. Hedberg, "Agents for Sale: First Wave of Intelligent Agents Go Commercial", *Expert, IEEE*, Vol. 11, No. 6, December 1996, pp. 16-19.
- [70] W. Lim, "An Agent-Based Approach for Programming Mobile Robots", *Proceedings of IEEE International Conference on Robotics and Automation*, 1994.
- [71] K. Sycara, M. Paolucci and S. Maveen, "Dynamic Discovery and Coordination of Agent-Based Semantic Web Services", *Internet Computing, IEEE*, Vol. 8, No. 3, 2004, pp. 66-73.
- [72] J.W. Baek and H.Y. Yeom, "E-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval", *IEEE Transaction on*

An Agent Based Application Service Providing System

Consumer Electronics, Vol. 49, No. 1, 2003, pp. 115-122.

[73] “Web Services Architecture”,

<http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>

[74] H.N. Xu and P. Seltsikas, “Evolving the ASP Business Model: Web Service Provision in the Grid Era”, *Proceedings of 2nd International Conference on Peer-to-Peer Computing*, 2002.

[75] A. Umar, “The Emerging Role of The Web for Enterprise Applications and ASPs”, *Proceedings of the IEEE*, Vol. 92, No. 9, 2004, pp. 1420-1438.

[76] P. Seltsikas, and W.L. Currie, “Evaluating The Service Provider (ASP) Business Model: The Challenge of Integration”, *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2002.

An Agent Based Application Service Providing System

Appendix

Sequence Diagram for Automatic Application Installation Application:

Sequence Diagram 1: (Access Request)

Sequence Diagram 2: (Installation Request)

An Agent Based Application Service Providing System

Sequence Diagram 3: (Inform Bill)

Sequence Diagram 4: (Authentication)

Sequence Diagram 5: (New Software or Software Update)

An Agent Based Application Service Providing System

Sequence Diagram 6: (Rent Software (not available))

Sequence Diagram 7: (Technical Support (with KB only))

Sequence Diagram 8: (Technical Support (with help of Technician Agent))

An Agent Based Application Service Providing System

Sequence Diagram 9: (Software Downloading Request)

Sequence Diagram 10: (Software Installation with Technician Agent)