Application of Mobile Agents in Web-based Student Modelling

Hong Hong

Abstract

In recent years, educational information on the web has increased exponentially, and web-based learning environments are becoming mainstream applications on the Internet. But these environments face some common deficiencies, such as slow access, no adaptivity to individual student, limitation by bandwidth, and so on, which need to be resolved. Meanwhile, the research in Intelligent Agents technology has received a lot of attention in Information Systems Research and Development area. This project investigated mobile agents technology and its benefits, and applied this technology to address the problems that limit the potential of web-based learning environments.

This project has developed a system, using mobile agents technology, to capture interactions over the Internet and to provide a continuous interaction pattern for a given student, even in off-line mode or in the case of unreliable connection. The mobile agents technology is used as the communications channel between client and server instead the traditional approaches. The system uses two-step student modelling architecture, which consists of the local and central individual student models and central group student model. There are primarily three parts of student model in the system: local individual student model that resides in student's machine, central individual student model that resides on the central server, and central group student model that sits on the central server. This two-step modelling mechanism largely improves capturing interactions of a given student in the web-based learning environment, even in off-line mode, and enables the system to provide adaptation at different granularity.

The combination of two-fold student modelling and mobile agents technology provides an attractive alternative to implement and improve web-based learning environments. The methodology used in this system addresses the problem of adaptation, which is one of the main bottlenecks that limit the development of web-based intelligent educational systems.

Acknowledgements

I would like to thank my supervisor, Dr. Kinshuk for his guidance and help throughout this research.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 The Context of Web-based Environments

The proliferation of computers, the advent of Internet and the steady gain in popularity of Distance Education greatly influence our educational environment. Educational information on the web has been increasing exponentially, and Web-based learning is currently an important research and development area. Web-based learning environments are strongly driven by information revolution and the Internet. The educational systems, especially the Web-based learning environments, are becoming mainstream applications on the Internet, but they have a number of common deficiencies, such as the lack of adaptivity for individual student, connection limitation and slow access to course, which need to be resolved. A number of attempts, such as static intelligent interface agents, have been made to solve some of these problems, but solution to one problem often impedes solutions to the remaining problems. The emerging intelligent mobile agents have huge potential to address those deficiencies.

1.2 Intelligent Agents and Mobile agents

Intelligent agent is a computational entity, which acts on behalf of other entities in an autonomous fashion, performs its actions with some level of proactivity and/or reactiveness, and possesses some key attributes such as learning, co-operation and mobility. The concept of agents came in mid 1970s (Hewitt, 1977). These few years, the strong trends in the Internet technology and distributed systems have lead to the point, where agents technology, in particular the mobile agents technology, is one of the "hot" topics in Information Systems Research and Development.

Application domains, in which agent solutions are being applied to or investigated include workflow management, network management, air-traffic control, business process re-engineering, data mining, information retrieval/management, electronic commerce, education, personal digital assistants (PDAs), e-mail, digital libraries, command and control, smart databases, scheduling/diary management, and so on.

The newcomers – mobile agents, which emerged in the mid 90s, can move from one computer to another. These few years mobile agents technology has been highlighted by many big research groups, e.g. Telescript (White, 1996), AgentTCL (Gray, 1997), Aglet system (Chang and Lange, 1996), Bee-gent and Plangent (URL1, 2001), Hive (Minar, 2000). There are various reasons why mobile agents are highlighted more than static agents in recent years. These include their potential to address the problems of latency and bandwidth of client-server applications and the vulnerability of network disconnection, to fit into the coming dynamic and mobile age of computing, etc. They are bringing together telecommunications, software, and distributed system technologies to create new ways of building computing systems. The benefits of mobile agents, especially on the web, directly address the problems of Web-based learning environment as discussed in chapter 5.

1.3 Motivation for the Research

1.3.1 The Context of The Research

In general sense, a Web-based learning environment should interact with the students, adapt to the needs of individual students, support interaction with teachers and other students, and be user-friendly to the authors. A prototype of Web-based Intelligent Tutoring System, Student Modelling and Adaptivity in Web-based Learning System (SMAWLS) has been built successfully to exploit the student adaptivity in Web-based environment by using traditional client-server technology (Han, 2001). This system benefits from the collaborative learning on the web by having two separate student models: *Individual students model* for each student; and *Group student model* for generalizing the attributes of a group of students. However, the potential of the mechanism of the individual student model and group student model in the SMAWLS was not exploited fully, and it also has the common deficiencies of Web-based learning environments.

1.3.2 Motivations

Although it is possible to propose an alternative, based on an existing technology, to almost every mobile agent-based function (Chess et al., 1995), in certain cases mobile agents have significant advantages over conventional approaches at the design, implementation and execution stages. The motivation for using mobile agents stems from the following anticipated benefits:

- *Efficiency and reduction of network traffic:* Mobile agents consume fewer network resources since they move the computation to the data rather than the data to the computation. Also mobile agents can package up a conversation and ship it to a destination host, where the interactions can take place locally, hence reducing the network traffic (figure 1.1).

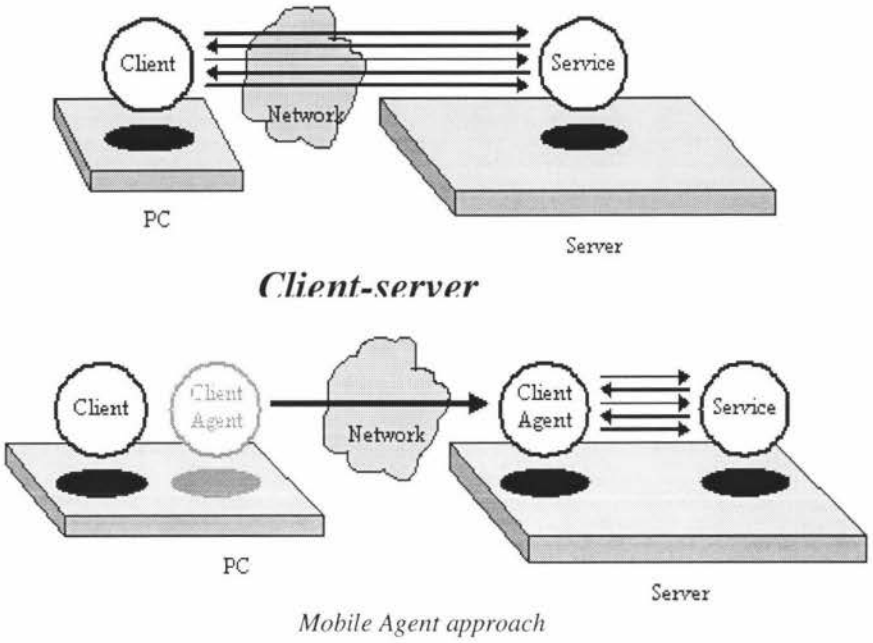


Figure 1.1: Client-server paradigm vs. Mobile Agent approach

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

- *Interaction with real-time entities:* Real-time entities require immediate responses to changes in their environment. Controlling these entities from across a potentially large network will incur significant latencies. Mobile agents offer an alternative to reduce or even eliminate network latency.

- *Local processing of data:* Processing of vast volumes of data over the network becomes inefficient when the data is stored at remote locations. Mobile agents allow the processing to be performed locally, instead of transmitting the data over a network.

- *Support for heterogeneous environments:* Both the computers and networks on which a mobile agent system is built are heterogeneous in character. As mobile agent systems are generally computer and network independent, they support transparent operation.

- *Convenient development paradigm:* The design and construction of distributed systems can be made easier by the use of mobile agents. Mobile agents are inherently distributed in nature and hence are natural candidates for such systems.

In term of motivation, this research attempts to empower the mechanism of individual and group student models, and investigate possibility and potential of using mobile agent technology to facilitate the communication between student models to improve the student adaptivity in Web-based learning environments.

1.4 The Research Steps

The project is broken into five phases, as described below:

- Phase 1: overviews the development of Web-based learning environments and discusses its student adaptivity and its existing problems.
- Phase 2: overviews the intelligent agents and has a closer look at mobile gents, and matches the benefits with the problems of Web-based learning environments.
- Phase 3: investigates and exploits the mobile agent framework – Beegent framework.
- Phase 4: investigates and develops the architecture of communication between central and local student models using mobile agent.
- Phase 5: develops a prototype that implement the mechanisms of communication between central and local student models by employing Beegent framework, Java 2, and InstantDB.

1.5 Structure of the Thesis

The structure of the thesis follows closely the phases in the research steps. Chapter 2 contains the review of the Web-based learning environments. In Chapter 3 the intelligent agent history and technology are reviewed. Then the details of the mobile agents technology is discussed. Chapter 4 reviews the existing application of intelligent agents in educational systems and matches the benefits of mobile agents with the problems of Web-based learning environments. Chapter 5 describes the context system of application of mobile agent, and discusses the architecture of using Beegent to implement the communication between central and local student models. Chapter 6 contains the overview of main technologies used by the prototype, and the implementation of the prototype that demonstrates the use of mobile agents in student adaptivity in the Web-based learning environments. Chapter 7 presents the early evaluation and concludes the thesis by reviewing the work done by the project and discussing further research directions.

Chapter 2

Review of Web-based Learning Environments

This chapter overviews the history of computer assisted learning environments, and Web-based and standalone environments. The related architectures and technologies are also reviewed, with particular focus on the adaptation and student model in both standalone systems and Web-based environments. The difficulty and bottleneck factors in developing Web-based environments are then described in detail.

2.1 History of Computer Assisted Learning Environments: Overview

The educational systems, especially the Web-based learning environments, are becoming mainstream applications on the Internet. To discuss the Web-based learning environments, it is necessary to have a brief review of computer assisted learning environments or instruction.

Over the years there has been a large array of names used more-or-less interchangeably to denote computer assisted systems: Computer-Based Instruction (CBI), Computer-Based Training (CBT), Computer Aided Learning (CAL), Computer Mediated Education (CME), or Computer Assisted Instruction (CAI). While CBI, CBT, CAL, CME, and CAI all refer to computer software that aims at instructing, Intelligent Tutoring Systems (ITS) add the intelligent to it. ITS were to provide individualized instruction by adapting its teaching strategy to the students of different abilities and learning styles and diverse educational and cultural backgrounds. The term Intelligent Tutoring Systems (ITS) has persisted and is still widely used in literature although computer programs do not possess human-like intelligence.

The growing popularity of the Internet in the 1990s resulted in the emergence of a number of Adaptive Hypermedia Systems (AHS), which are based on ITS and add adaptive hypermedia techniques to suit the Web-based environments. In the recent years, there are factors, such as the Internet development, lifelong learning, education in the workplace and distance education, driving Web-based learning environment to become an important research and development area and opening new ways of learning for many people.

2.2 Overview of the Existing Educational Systems

2.2.1 Standalone (Non Web-based) Systems

The development of educational systems is bound tightly to the evolution of computer technologies and the discovery of educational theories. The early computers didn't have the storage capacity or the computational power to support the development of detailed user models. In addition, the educational theories were just developing and the majority of the systems built in the 1960s were based on behaviourism, which explains human cognition in terms of simple response to external stimuli. These educational systems were inflexible in teaching strategies but, by restricting them to building skills, the systems proved to be effective.

In early 1970s, drill-and-practice systems with adaptive elements were developed. The Intelligent Tutoring Systems (ITSs) development took a big step towards more student-centred systems. What to be presented to students would depend on the students' past performance. In these systems, the "buggy" libraries were created to store students' possible responses and instructional paths that depended on the students' responses. For example, BUGGY employed an extensive library of "bugs" to evaluate the students' responses in doing arithmetic operations (Brown and Burton, 1978), and its improved versions are DEBUGGY and IDEBUGGY (Burton, 1982). SOPHIE (SOPHisticated Instructional Environment) (Brown et al., 1982) was a reactive learning environment, which searched the student responses in order to find the closest match. SOPHIE's domain was electronic circuit.

Research in the 1980s resulted in more efficient ITSs. The educational theories had also evolved by that time from the initial behavioural model. One particularly influential theory of learning was Anderson's Adaptive Control of Thought (Anderson, 1983), which considered the student's problem-solving behaviour as being goal-oriented, not as behavioural respose to stimuli. The Geometry Tutor (Anderson et al., 1985) and the LISP Tutor (Anderson et al., 1989) are notable examples built on Anderson's theory.

The architecture of standalone intelligent tutoring systems has been established well and used popularly. At a bird's-eye level of abstraction, the intelligent tutoring system architecture consists of fours basic modules: a tutorial component, a student model for keeping track of student progress over time, a cognitive simulation of an expert problem solver, and the user interface. Figure 2.1 (Burns and Capps, 1988) shows the basic architecture of the traditional intelligent tutoring systems.

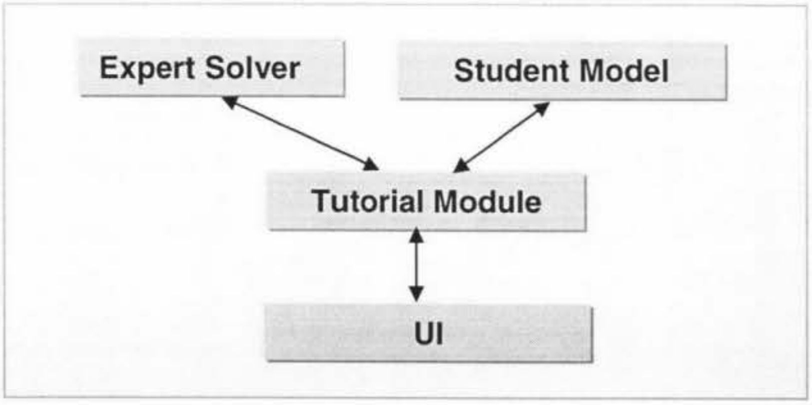


Figure 2.1: Architecture of traditional standalone intelligent tutoring systems

2.2.2 Web-based Environments

Duchastel (1992) has provided a good overview of various possibilities of how hypermedia and intelligent tutoring could shake hands and the recent developments in distance education research include the facilitation of intelligent tutoring on the Internet. The growing popularity of the World Wide Web in the 1990s has resulted in a number of Adaptive Hypermedia Systems. AHSs apply different forms of user models to adapt the content and the links of hypermedia pages to the user. Most of adaptive and intelligent technologies applied in Web-based adaptive and intelligent educational systems (AIES) were directly adopted from either the ITS area or the adaptive hypermedia area. Advantages of Web-based AHS over other ITS include potentials for world-wide use, greater level of communication and collaboration (Kinshuk and Patel, 1997), and the ability to dramatically extend the domain with static and dynamic materials from around the globe (Vassileva, 1997a). A number of standalone (i.e., non Web-based) adaptive educational hypermedia systems were built between 1990 and 1996. The initial Web-based adaptive and intelligent tutoring systems that use adaptive hypermedia technologies were reported in 1996 (Brusilovsky et al, 1996; De Bra, 1996). The Adaptive Electronic Textbooks (Brusilovsky et al, 1996a) facilitate transfer of normal textbooks existing in electronic form into adaptive electronic textbooks. The system contains a domain model and a learner model. The domain model can either be a set of domain concepts or a network with nodes corresponding to domain concepts and with links reflecting several kinds of relationships between concepts. The learner model is of overlay type storing estimatioin values for each concept. The concepts are visually annotated representing learner's state of interaction with them and systems's recommendation at a point of time.

The system - Dynamic Course Generator (DCG) (Vassileva and Deters, 1998; Vassileva, 1998; Vassileva, 1997b) generates individual courses according to the learner's goals and previous knowledge and dynamically adapts the course according to the learner's success in acquiring knowledge. The teaching material is kept in other with various types of semantic relations. Schoch et al (1998) described a personal teacher agent, called ADI integrated within an adaptive statistics tutor (AST). ADI supports the learner at the time of browsing through the domain content of the system. It is adaptable to the learner'' individual preferences and is adaptive to the learner's current state of knowledge.

Web Recourse, the Web Retargetable Course Generation System (Lemone, 1996) facilitates creation and sharing of courses on the Internet and easy update reworkingo the course over time. Yum and Crawford (1996) described a generic interoperable Intelligent Tutoring System (ITS) architecture based on Open Database Connectivity (ODBC). They accepted that though ODBC supports interoperable data transfer, the specifications of the ITS domain data and multimedia information need to be agreed upon to enable implementations by external parties.

Nakabayashi et al. (1996, 1997) proposed CALAT based on client-server architecture, which consists of a WWW server integrated with a tutoring system and a WWW client equipped with a multimedia scene viewer. The system makes use of text, sound, animated graphics and other media, as well as simulation techniques for hands-on experience.

Those systems used different technologies to implement the Web-based intelligent tutoring systems, which mainly include following solutions:

- Java-only solution: the whole system resides in a Java applet that can be downloaded from its web site, then run on the student's computer. All tutorial processes happen on the client side;

- HTML-CGI solution: the student interacts with HTML forms in the client side web browser, and information is sent to the web server (e.g. Apache, IIS) by requests that are passed to the CGI program (such as PHP, PERL), then replies by HTML pages. All tutorial processes reside on the server side;

- Traditional client-server solution: a downloadable Java applet contains the client interaction portion and uses the socket connection technique to communicate with the server application developed with Java (e.g. Servlet). Some of tutorial processes reside in the client side computer, others in the server side. However, they share the similar high-level architecture (figure 2.2).

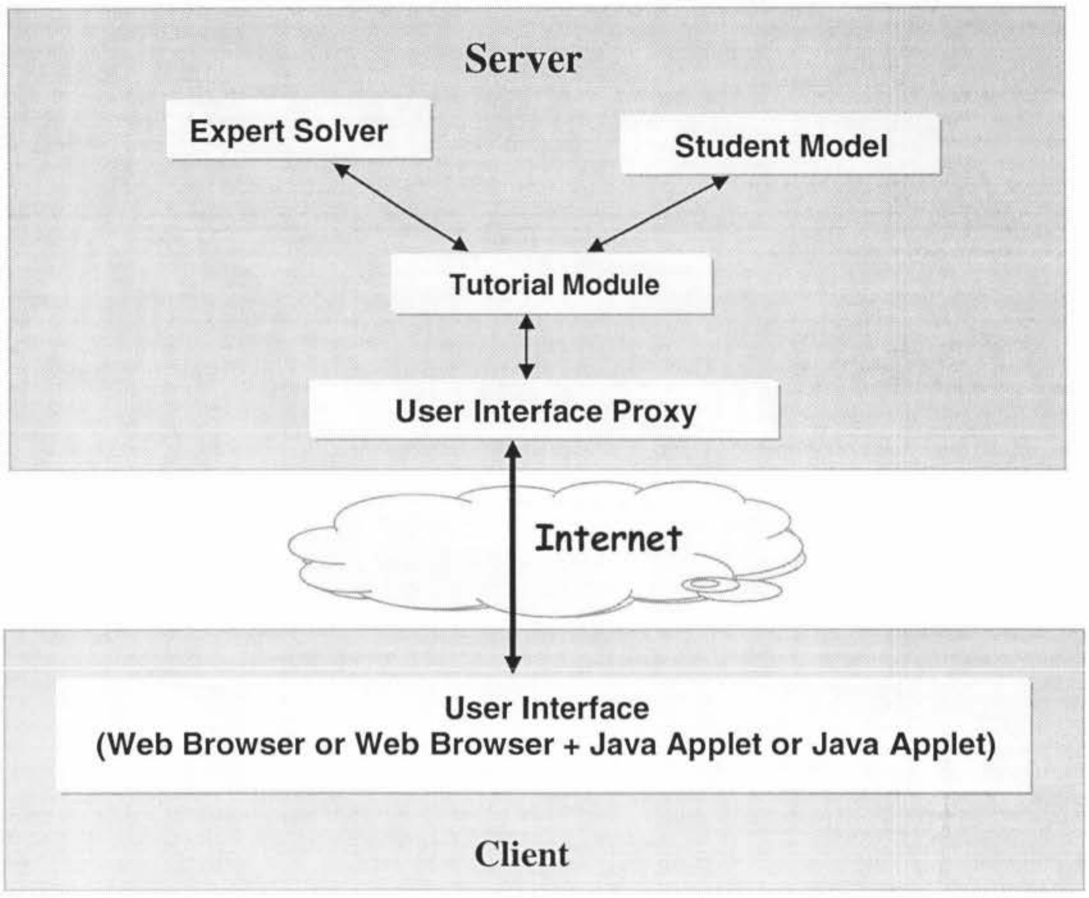


Figure 2.2: Basic high level Web-based architecture for ITSs

In 1999, Kinshuk, et al. (1999) presented Hyper-ITS, which is a Web-based architecture for evolving and configurable learning environment. The Hyper-ITS architecture is based on data driven technology which enables the applications to be relatively domain independent and allow easy incorporation of additional domain content and modification of existing material. The Hyper-ITS has already been implemented and tested in the form of Byzantium ITTs (Intelligent Tutoring Tools) – a consortium of six universities, and has been found successful in providing adaptive and interactive learning environment to the students similar to the traditional one-to-one tutoring.

Currently, the web has become the primary platform for developing educational adaptive and intelligent hypermedia systems, and a number of Web-based environments have been developed and are developing. Web-based learning environments are currently an important research and development area and opens new ways of learning for people.

2.3 Benefits of Web-based Environments

The information revolution has induced the boom of knowledge economy. This boom is supported by the updating technologically skilled workforce. Fischer (1996) pointed out that the earlier notions of a lifetime divided into neat phase of education followed by work are no longer tenable and learning can no longer be dichotomized either spatially or temporally into a place and time to acquire knowledge (school/college) and a place and time to apply knowledge (workplace). Lifelong learning integrates and mutually enriches both the cultures of education and work. According to a survey, conducted by the International Foundation of Employer Benefits Plans, employees rank continuing education as more important than childcare, flexitime and family leave (Petersons, 1999). It has also been recognized that lifelong learning, education in the workplace and distance education are major factors in achieving the technologically skilled workforce.

Driven by the new education environment, the Internet is now becoming integral to all types of education and training. The global learner community includes users of all age and backgrounds with different interests and motivations for using computer networks. Internet has generally been successful in keeping up with these motivations and interests by providing useful information and a greater sense of enjoyment through new ways of cognition. In the current educational environment with an increasing demand on education sectors, alternative approaches such as distance education are becoming more attractive for academic institutions. Leigh (1996) observed that the Internet can provide an instantaneous dissemination of information to a wider audience and education at a distance can meet the financial needs of both the institution and the learner often proving a very effective alternative to traditional instruction or training. However, the students do not prefer to learn by the traditional distance education as they value the presence of an informal interaction that provides active learning (Simonson, 1996). In this connection, the Internet has proved more beneficial than the traditional distance learning approaches as it has the following facilities:

- Accommodating far more interactions – involving information in textual, graphical, audio and video forms, in a cost effective manner

- Providing mechanism for asynchronous interactions through discussion forums, newsgroups and email

- Providing synchronous interactions through Electronic White Boards, Web Phone and Web Meeting

In some ways, this medium of instruction surpasses even the traditional classroom as it facilitates communications between academic communities located in different parts of the world.

Since these different factors driving, thousands of Web-based course and other educational applications have been made available on the web within the last five years. But most of them are nothing more than a network of static electronic pages. Such electronic textbooks are non-adaptive, i.e., students with different abilities, knowledge and background receive the same educational material.

Obviously, the Web-based learning environment shows a number of benefits, its potentials and importance in the current and future education. However, the existing Web-based environments are facing some deficiencies and problems that need to be addressed, because the characteristics of Web-based environment are distinguished with the traditional learning environment, such as Internet connection problems, adapting to individual. In the following section, the problems of Web-based learning environments are going to be discussed.

2.4 Student Adaptivity in Web-based Learning Environments

Adaptivity features within intelligent tutoring systems have largely improved those learning environments, where the student modeling is the key technology to implement the adaptivity features. Before discussing the student models, the concept of adaptation is described briefly.

2.4.1 Adaptation

The concept of 'adaptation' is one of core research topics in intelligent learning environments (Nikov and Pohl, 1999; Hoschka, 1996). Adaptation of a system is based on the assumption that the system is able to adapt itself to the goals and tasks of the user by evaluating his/her behavior (Oppermann, 1994). The ambition of adaptation is that not only "every one should be computer literate" but also that "computers should be user literate" (Browne et al., 1990).

Milne et al (1996) have pointed out that the application of adaptation can provide better learning environments in intelligent tutoring systems but many issues need to be resolved before an effective and efficient adaptation in learning systems is possible. There have been a number of attempts in last decade to improve the correspondence between student, task and system characteristics and the student's efficiency by adding adaptation features into systems.

Based on the purposes of adaptation, two kinds of systems have been developed for improving the capability of adaptation within systems, which are adaptable systems and adaptive systems.

- Adaptable systems: allow the user to change certain system parameters and adapt their behavior accordingly;

- Adaptive systems: adapt to the user automatically based on the system's assumptions about the user needs.

Figure 2.3 shows the whole spectrum of the concept of adaptation in computer systems (Oppermann et al., 1997).

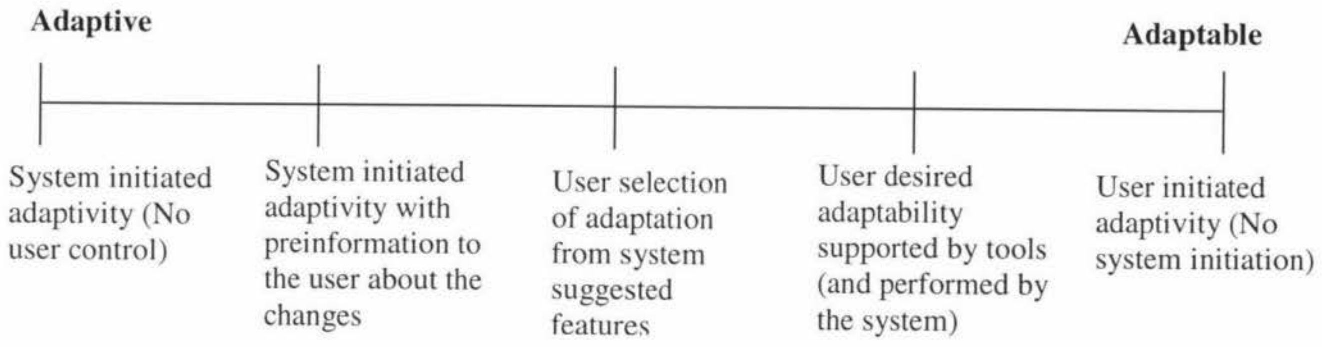

Adaptive
Adaptable
System initiated adaptivity (No user control)
System initiated adaptivity with preinformation to the user about the changes
User selection of adaptation from system suggested features
User desired adaptability supported by tools (and performed by the system)
User initiated adaptivity (No system initiation)

Figure 2.3: the whole spectrum of the concept of adaptation

2.4.2 Student Models

2.4.2.1 Definition

The student model is the main component within the adaptive non-Web-based or Web-based ITS, and it contains the information about the student's skill, knowledge and practice. It obtains student's information by dynamically observing and recording the student's behaviour, answers and problem-solving strategies, and analyzing them to deduct the level of the student's understanding of the domain material at any given moment. Because the student is acquiring knowledge continuously, the student model also works dynamically (Nielsen, 1990). The student model enables the individualized instruction and the dynamic representation of student skills and proficiencies (Vassileva, 1995), for example, when a student answers a question or completes an exercise, the student model will record the student's behavior and the system will use the student model to determine the best possible instructional path (that is a typical case of adaptation).

Wenger (1987) defined three major roles that the student model plays:

- To collect important quantitative information about the student

- To provide a presentation of the collect information and convert it into some qualitative pattern analysis, such as the student's ability to absorb new materials, learning habits, speed, accuracy, level of confidence and so on, and

- To perform a diagnosis of the student's knowledge and use it as the basis for selecting the optimal teaching strategy. The qualitative analysis enables the system to concentrate on the student's strengths and work around the weakness.

The student model has become the basic part of ITSs. Since the student knowledge is the most important source of adaptation in ITS, the student models are classified into three types according to the assumptions about the student's knowledge: overlay, differential and perturbation (Sleeman and Brown, 1982).

- *Overlay model:* it considers the student's knowledge as part of the expert's knowledge. The essence of this model is that the student's knowledge is overlaid on the expert knowledge. The learning system attempts to bring the student closer to the expert's knowledge used in the system. Most ITSs use the overlay model (Brusilovsky and Pesin, 1994; Brusilovsky et al., 1993; Brusilovsky and Zyryanov, 1993; De Rosis et al., 1993). In this model, the student knowledge is represented as a set of 'concept-value' pairs: Boolean (knows/doesn't), qualitative (good/average/poor), or quantitative (percentage). The overlay model was first report in WUMPUS (Stansfield, 1976). This model is usually initialized by pre-tests or default settings.

- *Differential model:* it examines how the student's knowledge differs from the expert's knowledge. It differentiates between the domain knowledge that students lack and the knowledge about which no conclusion can be made based on the state of the student model (Kass and Finin, 1989). The system determines whether students possess knowledge or not by checking how the students use knowledge that the system defines. A notable example of using this model is the WEST system (Burton and Brown, 1982).

- *Perturbation model:* it takes into account that the student's knowledge can extend beyond the expert knowledge. This model provides more flexibility as new perturbations can be added into an existing model when needed, while the overlay and differential models consider the students' knowledge as a subset of the expert knowledge. However, the perturbation model brings more difficulty. This model was implemented in DEBUGGY and IDEBUGGY system (Burton, 1982).

2.4.2.2 Content of Student Model

Self (1994) pointed out that a comprehensive student model should contain information about the student's domain knowledge prior to the use of the educational system, the learner's progress, preferences, interests, goals, and any other information related to the learner.

Based on the dependence upon the subject domain, the content held in student models consists of two parts: domain specific information and domain independent information (Brusilovsky, 1994).

- *Domain specific information:* it is also named as the student knowledge model (SKM) by Brusilovsky (1994), which represents a reflection of the student's state and level of knowledge and skills in term of a particular subject domain.

- *Domain independent information:* it is slightly different from system to system. The domain-independent information about a student may include learning goals, cognitive aptitudes, measures for motivation state, preference about the presentation method, factual and historic data, etc.

2.4.2.3 Constructing

The collected student model information can be done in three ways: explicit, in response to questions and tasks given, or initial testing to analyse the student's behaviour, and stereotyping.

Explicit questioning: the systems directly question the user to gather his/her relevant information, which is used to construct the student model. This is an effective effect way to obtain student's general information (Paiva, 1995).

Initial testing (pre-test): before using the system, the student is asked to take some tests, then the system analyses the student's test results to gather information for constructing the student's model.

Stereotyping: it is defined by collecting common attributes among a certain group of people, which should share the same interests according to a set of criteria. The information in a stereotype is used as default parameters to initialise a student's model. Sleeman (1984) criticises the use of stereotypes because of the coarse graining.

2.4.2.4 Updating

The student model is used to enable the individualised instruction and improve the adaptation capability of the system. Thus, it must keep up to date student's profile, that is the student model needs to be updated timely and dynamically.

The update of the student model is achieved based on the analytic results, which are obtained by analysing student responses to the questions presented by the system, problems solving processes and operations within the system. These analytic processes are called *cognitive diagnosis* that is the process of inferring a person's cognitive state from his or her performance (Paiva, 1995). These analytic processes can be implemented in four major ways: implicit, explicit, structural and historical acquisition (Kinshuk, 1996; Brusilovsky, 1994).

1. *Implicit acquisition,* by observing the actions of a student while being engaged in learning processes

2. *Explicit acquisition,* by direct dialogues between the system and the student e.g. explicit questioning.

3. *Structural acquisition*, by analysing the interrelations between curriculum elements, e.g. if curriculum element A is a prerequisite of B, mastery of B means mastery of A, and lack of knowledge on A implies lack of knowledge on B.

4. *Historical acquisition,* by generalised estimate of the student knowledge and capabilities or the student previous experience.

2.4.3 Student Adaptivity in Web-based Learning Environment

Most of the adaptive and intelligent technologies applied in Web-based learning environment were directly adopted from either the ITS area or the adaptive hypermedia area.

2.4.3.1 Adaptive ITS Technologies in Web-based Learning Environment

The adaptation ITS technologies related to Web-based learning environments are curriculum sequencing, intelligent analysis of student's solutions, interactive problem solving support, example-based problem solving support, and collaboration support (Brusilovky, 1996).

- *Curriculum sequencing:* it helps the student to find an optimal path through the learning material.
- *Intelligent analysis of student solutions:* it deals with students' final answers to educational problem.
- *Interactive problem solving support:* it provides the student with intelligent helps on each step of problem solving.
- *Example-based problem solving support:* it helps students to solve new problems by suggesting them relevant successful problem solving cases from their earlier experience.

In the web context, these technologies are implemented in ELM-ART (Brusilovsky et al 1996) and ELM-ART- II (Weber and Specht, 1997).

2.4.3.2 Adaptive Hypermedia Technologies in Web-based Learning Environment

Adaptive hypermedia systems apply different forms of user models to adapt the content and the links of hypermedia pages to the user. Brusilovsky (1999) distinguishes two major technologies in adaptive hypermedia based systems: adaptive presentation and adaptive navigation support.

- *Adaptive presentation support*: it is used to adapt the content of a hypermedia page to the users' goals, knowledge and other information stored in the user model. The PT (Kay and Kummerfeld, 1997), AHA (De Bra and Calvi, 1998) are the sample systems, which implemented the adaptive presentation technology.

- *Adaptive navigation support (ANS)*: it supports the student in hyperspace orientation and navigation by changing the appearance of visible links. The most popular forms of ANS on the web are direct guidance, annotation and disabling. The following Web-based environments implemented the ANS technology: ELM-ART- II (Weber and Specht, 1997), Interbook (Brusilovsky et al, 1997) and DCG (Vassileva, 1997), CALAT (Nakabayashi, 1997).

2.4.3.3 Student Models in Web-based Learning Environments

To create Web-based learning environments that adapt themselves to each individual student, the student model plays the same important role as in traditional standalone ITS. The student model within Web-based environments contains the same or similar student information as standalone ITS, which has been discussed early in this section.

However, since the platform and running environment for Web-based environments are different from standalone systems and much more complicated than standalone systems, the difficulty of tracing and recording student's behavior and responses in Web-based environments has resulted in no significant attempts to provide adaptation in Web-based learning. This is one of the problems that this project attempts to address.

2.5 Problems in the Web-based Environments

The information revolution has induced the boom of knowledge economy. This strong trend is supported by the updating technologically skilled workforce, which is achieved by lifelong learning, education in the workplace and distance education. Those are the prime factors driving the Web-based education.

Institutions with long-standing involvement in distance education, such as Open University in United Kingdom, are incorporating Web-based elements in their instruction. Although Web-based course materials have advantages over conventional textbooks and lecture notes, they have a number of common deficiencies:

- Slow access to course materials, because the Internet connection bandwidth limitation, in particular home users or remote area student.

- Inadequate adaptation to individual students, because the interactions between client and server normally take place using Hypertext Transfer Protocol (HTTP). HTTP is a stateless protocol, which makes it difficult to track the students progress and hence to analyze the mental processes of the student (Kinshuk and Patel, 1997).

- Hard to achieve continuous real time interaction between student and system, because of the connection unreliability and bandwidth limitations or student may be in off-line mode.

A number of attempts have been made to attack some of these problems, but solution to one problem often impedes solutions of the remaining problems. For example, InterBook (Brusilovsky et al., 1996a) supports adaptivity and authoring, but all adaptation and page generation takes place at the central server, risking access delays. QuestWriter (Bogley et al., 1996) supports authoring, and has built-in client-side and servers side interactivity, but does not adapt presentations to individual students. ADE (Shaw et al., 1999) uses an animated pedagogical agent, who guides and assesses students, to develop an agent-assisted learning environment. The Adele agent basically belongs to an intelligent interface agent, which still concentrates on providing helps to students. In particular, a number of efforts have been made to implement the concept of adaptation in learning systems (e.g. Crampes, 1999; Oppermann and Specht, 1999), but there have been no significant attempts to provide adaptation in Web-based learning environment.

In general sense, a Web-based learning environment should interact with the students, adapt to the needs of individual students, support interaction with teachers and other students, and be user-friendly to the authors. The emerging mobile agent technology is not only capable to facilitate these benefits but also attempts to address the problems mentioned above.

2.6 Summary

The standalone intelligent tutoring systems have been developed with sophistication , and their classic architecture consists of an expert module, a student model module, a tutoring module, and a user interface module.

The Internet has driven the Web-based learning environments to becoming mainstream applications, in particular since 1996 when the initial Web-based adaptive and intelligent tutoring systems were reported.

In Web-based learning environments, the standalone ITS is extended to create different client-server architecture, such as replicated, centralized or distributed. Most of the adaptive and intelligent technologies applied in Web-based learning environments are directly derived from either the ITS area or the adaptive hypermedia area. However, the existing bottleneck factors in developing Web-based environments, such as inadequate adaptation to individual students, hard to achieve continuous real time interaction and slow access to course material, means the Web-based environments do not exploit their full potential. These common deficiencies in Web-based environments are the major concerns of this research.

In the next chapter, the agents technology, which is the key technology used by this research, is discussed in detail.

Chapter 3

Overview of Agents Technology

This chapter briefly reviews the concepts and history of agents technology. Then the classification of agents is discussed in detail.

3.1 Concepts of Agents

In the agents developer community, experts define agents as:

- Entities that can communicate in an agent communication language (Genesereth et al., 1994).
- Computer systems in a complex environment that realize a set of tasks and goals they were designed for (Maes, 1995).
- Systems capable of autonomous purposeful action in the real world (Brustoloni, 1991).

Standards bodies define agents as:

- Computer programs act autonomously on behalf of a person or organization (OMG MASIF)(OMG, 2001).
- Computational processes implement an application's autonomous communicating functionality (FIPA, 2001).

These definitions are slightly different one another and have no consensus on definition, but they reflect important types of agents, such as autonomous, adaptive, reactive, mobile, cooperative, interactive, and delegated.

3.2 History of Agents

The concept of an agent was initiated by Carl Hewitt in 1977 (Hewitt, 1977). In the concurrent Actor model of his report, Hewitt proposed the 'actor', which is described as "a computational agent, which has a mail address and a behavior. Actors communicate by message-passing and carry out their actions concurrently" (Hewitt, 1977). The actor has following features:

- self-contained, interactive and concurrently-executing;
- had some encapsulated internal state; and
- capable of responding to messages from other similar actors.

From 1977, research in agents technology concentrated mainly on deliberative-type agents with symbolic internal models, and most of the work focused on issues such as the interaction and communication between agents, the decomposition and distribution of tasks, coordination and cooperation, conflict resolution via negotiation, and so on. The main goal was to specify, analyze, design and integrate systems comprising of multiple collaborative agents, such as MACE (Gasser et al., 1987), MICE (Durfee et al.,1989), and MAS/DAI planning and game theories (Rosenschein et al.,1994).

In early 1990s, another distinct trend to the research and development work on intelligent agents emerged, with focus on the investigation of the diversification in the types of agents. A number of agent types were identified such as User Interface Agents, Information Agents, Multi Agent System, Mobile Agents and so on. Mobile agents are perhaps the most exciting ones, which originally emerged from advances made in distributed systems research (Cardelli, 1995). These few years, many prominent research groups have been seduced by the research on Mobile Agents, which has resulted in a number of projects such as Telescript (White, 1996), AgentTCL (Gray, 1997), Aglet system (URL4, 2001), Bee-gent and Plangent (URL1, 2001), and Hive (Minar, 2000).

3.3 Classification of Agents

It is evident from recent literature and discussions in relevant mailing lists that there is no consensus on the classification of software agents. It is therefore no surprise that one can find researchers arguing how to classify the agents and attempts are still continuing to find out a more reasonable classification.

In reality, software agents exist in a truly multi-dimensional space, even though some research papers try to collapse these multiple-dimensions to a single list. Agents can be classified in at least four dimensions: Quantitative, Mobility, Responsiveness, Behavioral (figure 3.1).

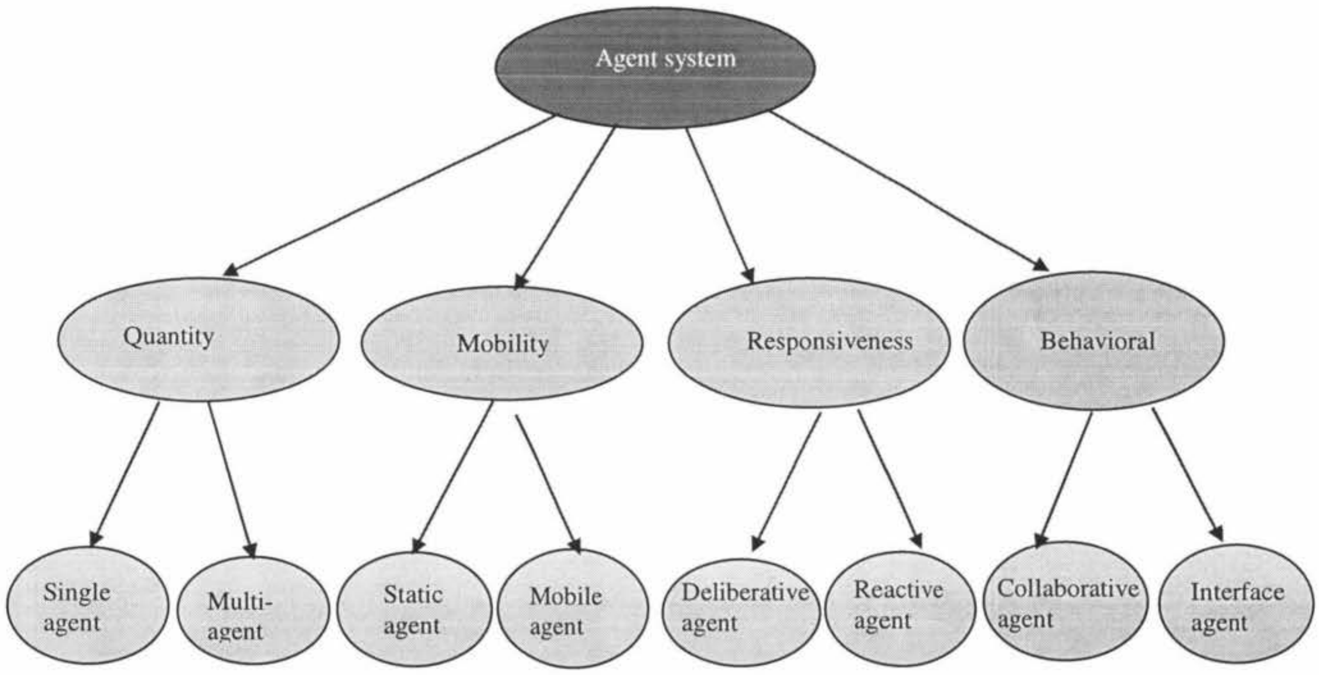


Figure 3.1: Multi-dimensional classification of software agent system

3.3.1 Quantitative

Various agent technologies existing today can be classified as being either single-agent or multi-agent systems. In single-agent systems, an agent performs a task on behalf of the user or some process. While performing its task, the agent may communicate with the user as well as with local or remote system resources, but it will never communicate with other agents. In contrast, the agents in a multi-agent system (MAS) not only communicate with the user and system resources, but they also extensively communicate and work with each other, solving problems that are beyond their individual capabilities.

3.3.2 Mobility

Agents can also be classified by their mobility, i.e. by their ability to move around the networks. This yields the classes of static or mobile agents. Static agents are constrained to a single computer; early agents were all static. The newcomers – mobile agents can move from one computer to another. They are bringing together telecommunications, software, and distributed system technologies to create new ways of building computing systems.

3.3.3 Responsiveness

According to the means or the response method that an agent uses to react to any request, they may be classified as either deliberative or reactive.

- Deliberative agents derive from the deliberative thinking paradigm: the agents possess an internal symbolic, reasoning model and they engage in planning and negotiation in order to achieve coordination with other agents.

- Reactive agents originate from research carried out by Brooks (1986) and Agre and Chapman (1987). They do not have any internal, symbolic models of their environment, and they act using a stimulus/response type of behaviour by responding to the present state of the environment in which they are embedded (Ferber, 1994).

3.3.4 Behavioral

Agents can also be classified along three primary characteristics: autonomy, learning and cooperation.

- Autonomy: autonomy refers to the principle that agents can operate on their own without the need for human guidance. A key element of their autonomy is their proactiveness.

- Learning: learning refers to an agent's ability to *learn* as they react, and/or interact with their external environment, then modify its own behavior.

- Cooperation: cooperation among multiple agents denotes the ability to interact with other agents and possibly humans via some communication language (Wooldridge et al., 1995), and coordinate to enhance their ability.

With regard to these three characteristics, an agent may have one or two of them. An agent with all three would be ideal.

In principle, by combining the above four constructs, i.e. quantitative (single/ multi-agent), mobility (static/ mobile), responsiveness (reactive/ deliberative), and behavioral (collaborative agents/ interface agents/ etc.), we could have single static reactive interface agents, such as Firefly's static agent (Firellly, 1996); multi-agent static deliberative collaborative agents, such as Bargain finder system (Krulwich, 1995); multi-agent mobile deliberative collaborative agents, such as Hive system (Minar et al., 2000), and so on. Single agent usually performs its task according to the preset state, and there is more emphasis on autonomy and learning than on cooperation. This kind of agent is not so flexible, but is portable, and simpler. Thus, it is suitable to be applied on personal day-to-day task assistants, and information filtering. Multi-agent systems have to plan, negotiate and cooperate with other agents regarding how to go about performing a task; there is more emphasis on autonomy and cooperation than learning. They are more flexible and complicated than single agent and the focus in their design and development is on coordination, negotiation, and communication. Such systems, which have already been applied in the fields of sensors, aircraft and robots, especially mobile agent systems, are also very ideal for today's electronic commerce systems and embedded network service systems, which are the next generation of computation. Today's research and investigation is showing that this area has huge potentials and is another revolution in computing.

In addition, agents may sometimes be classified by their roles (what they do), e.g. WebCrawlers (WebCrawlers, 2001), Lycos (Lycos, 2002) and Spiders (Spiders, 2002), which help manage the vast amount of information in wide area networks like the Internet. They are called *information* or *Internet agents*.

It is needed to be mentioned further that the newcomers – mobile agents, are becoming primary focus on the new underpinning technologies, and are highlighted extensively in recent years. They emerge from the shadow of artificial intelligence research. to create new era of computing.

3.4 A Closer Look at the Mobile Agents

After reviewing the general concepts of agent technology above, we shall discuss the detail of mobile agents technology, which is the key technology applied to implement our ideas in the research.

3.4.1 Overview of the Mobile Agents

Mobile agents technology originally emerged from advances made in distributed systems research (Cardelli, 1995). Mobile agents are computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming 'back home' having performed the duties set by its user (owner).

But for a mobile agent to work on the different host environments on a network, each host must have a software environment in which mobile agent can exist. We call this the mobile agent environment, which is built on top of a host system. This environment supports services that relate to the mobile agent environment itself, services pertaining to the environments on which the mobile agent environment is built, services to support access to other mobile agent systems, and finally support for openness when accessing non-agent-based software environments.

There are some reasons that mobile agents are highlighted more than static agents in recent years. Firstly, although much of the earlier work done on static agents has been important for facilitating single-user environments, the benefit of these intelligent agents or interface agents did not scale up well when they were placed alongside large real-time applications, including manufacturing, network management, and distance learning. In addition, previous research has failed to comprehensively address the behavior of agents serving the telecommunications and applications infrastructure.

Secondly, mobile agents can work against the problems of latency and bandwidth of client-server applications and reduce vulnerability of network disconnection.

Thirdly, www and the Internet, which are growing and becoming ubiquitous, dynamic and mobile, are driving mobile agents. People believe that current trends in Internet technology and usage lead to the use of mobile agents. Mobility (users, devices, and programs) becomes important paradigm, in particular e-commerce and mobile commerce. Agent system must be integrated with the Internet and www. Figure 3.2 shows how the mobile agent fits into the coming age of computing.

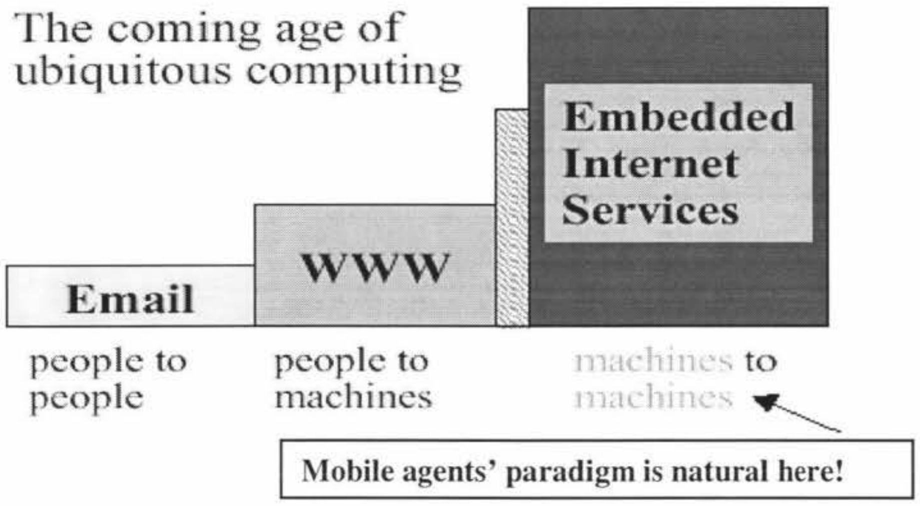

The coming age of ubiquitous computing
Email
WWW
Embedded Internet Services
people to people
people to machines
machines to machines
Mobile agents' paradigm is natural here!

Figure 3.2: Mobile agent fits into the embedded Internet services

Fourthly, mobile agents also provide a single, general framework in which distributed, information-oriented applications can be implemented efficiently and easily, with the programming burden spread evenly across information, middleware, and client providers.

3.4.2 Applications of Mobile Agents

Recent years mobile agent systems have become sufficiently well developed to be considered as using them as tools in other areas, such as gathering information, network management, electronic commerce, and mobile user and device.

Information search and retrieval remote filtering and compression: the nature of today's work enhances mobile agents' attractions because they offer better ways to deal with information and manipulating it. Agents can act as sophisticated assistants for exposing information systems, freeing people to interact with information indirectly and efficiently. For example, the Stormcast system (Stormcast, 2002) uses mobile agents created at the client side and sent to servers to retrieve and process huge amounts data. Another scenario is to use mobile agent approach instead of traditional client-server approach in search engines. The mobile agent approach, shown at figure 3.3, processes data at data source.

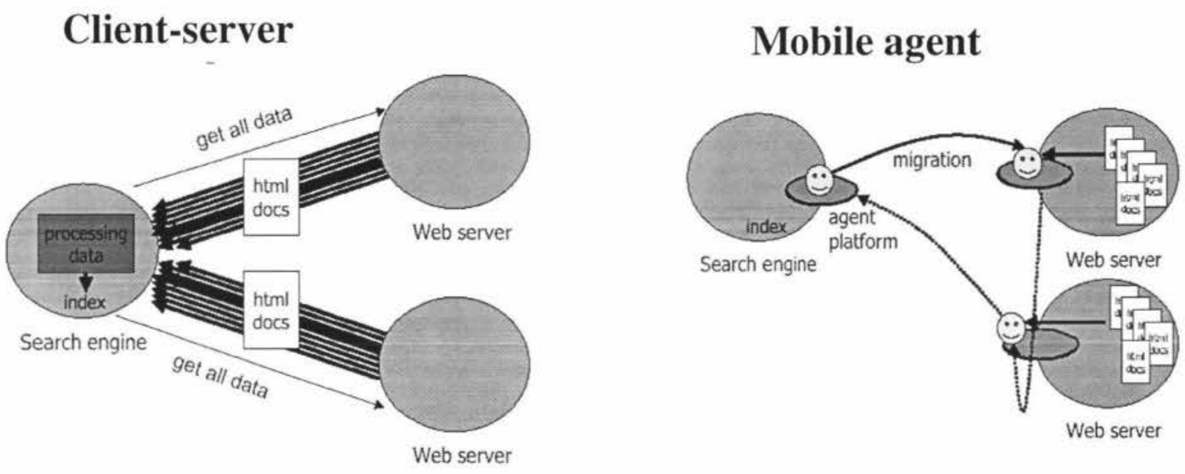


Figure 3.3: Comparison of traditional and mobile agent approach at search engine

Various applications of mobile agents are as follows.

Maintenance: Maintenance includes monitoring network management, and remote installation of software components. There are some telecommunications service providers, such as Lucent Technologies Inc. and Bell Atlantic Corp., who have tried developing products by using mobile agents to facilitate their network management. They believe that mobile agent technology can detect and react to network faults, and enable installable/extensible/modifiable services, and so on. The system, such as Perpetuum Mobile Procura, uses mobile agents - Netlets to enable advanced network management (Perpetuum, 2002).

Electronic commerce: Electronic commerce parties as a seller, buyer, trader, and broker, send personal agent on a shopping tour, such as in InAMos project (InAMosh-handy, 2002). The project consists of mobile agents to represent consumers in the market environment, and other ordinary mobile agents and services. They attempt to bring electronic market places to the mobile user.

Mobile devices and mobile users: Mobile devices are one of the hottest areas of growth in the computer industry. Devices from laptops to palmtops to electronic books, from cars to telephones to pagers, will access Internet services to accomplish user tasks. Usually, these devices have unreliable, low-bandwidth, high-latency cable or wireless network connections. As the mobile environment grows in complexity, more users will use diverse devices to work on several heterogeneous networks. The use of mobile agents, which have migrate-and-disconnect style of operations, has a lot of appeal due to their capability of sorting through to bear the burden of sorting through the service and delivery options.

3.4.3 Review of the Existing Mobile Agents Platforms

This section reviews a couple of enabling technologies on which a mobile agents system is built, and then proceeds to review five of the most important mobile agents systems in use today. Mobile agents technology has been under serious development since about 1994. However useful mobile agent systems appear only recently. Most of today's mobile agent systems are built on the top of the Java technology.

3.4.3.1 Enabling Technologies

Java and Obliq (Cardelli, 1995) are the two main technologies, which enable the building of mobile agent frameworks.

- Obliq is a lexically-scoped un-typed interpreted language that supports distributed object-oriented computation. It is one of the oldest mobile agent systems around. It is impractical as an implementation system.

- Java environment is a new approach to distributed computing. It is an interpreted, object-oriented language and library set. Java and its run-time system have combined them to produce a flexible and powerful programming system which supports distributed computing. It has recently received a lot of attention, which has resulted in significant revisions of the important RMI and OS facilities.

3.4.3.2 The Existing Mobile Agents Systems

Today, there are many mobile agents systems, some of them are still in the research phase, some are commercial systems. Five of the most important mobile agents systems in use or research today are reviewed in this section. They are Bee-agent (Toshiba, 2001), Aglets (IBM, 2001), Hive (Minar, 2000), AgentTcl (Gray, 1997, Kotz et al., 1996), and Telescript (White, 1996).

Bee-gent

Bee-gent, which is developed by Computer & Network Systems Laboratory Corporate Research & Development Center, TOSHIBA Corporation in 1999, is a new type of development framework in that it is a 100% pure agent system. As opposed to other following systems, which make only some use of agents, Bee-gent completely "Agentifies" the communication, that takes place between software applications. The applications become agents, and all messages are carried by agents. Bee-gent allows developers to build flexible open distributed systems that make optimal use of existing applications, and is able to work with Java 2 environment. In this research, Bee-gent is used to implement the mobile agents within the Web-based prototype developed by using Java 2. The details of Bee-gent are discussed in Chapter 6.

Aglet Software Development Kit

The aglet system (Chang et al., 1996) represents the next leap forward in the evolution of executable content on the Internet, introducing program code that can be transported along with state information. Aglets are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it brings along its program code as well as its data. Aglets are possibly one of the most applicable mobile agent technology at the present moment.

Since Aglets Software Development Kit with JDK 1.1 is entirely written in Java, it can only run on a platform that supports JDK 1.1.x. The current ASDK does not work with JDK 1.2, because JDK 1.2 is not fully compatible with JDK 1.1.x.

Hive System

Hive system is a decentralized system for building applications by networking local system resources. It is applied to the problem of networking "Things That Think" (ThingsThatThink, 2002) embedding computation and communication in everyday objects and appliances. Hive consists of three components: cells-nodes in the decentralized network, shadows-local resources, and agents-active computation. Its mobility is in many ways a reimplementation of systems such as Telescript, AgentTcl, Aglets, and Mole. But the research intends to create a simple version of mobility to the hard problems.

AgentTcl

AgentTcl is a simple platform independent mobile agents system. Currently it lacks the same degree of sophistication that the Aglets system has achieved. It is built on top of the Tcl/tk scripting system and lacks the power and flexibility of a full computer language such as Java. The security is quite basic and is relatively untested at this stage, as AgentTcl has had little exposure outside of Dartmouth College.

Telescript

Telescript is an object-oriented mobile agents language. It claims to be *"a platform that enables the creation of active, distributed network applications"*. Telescript is a whole new platform-independent system consisting of a language, and interpreter called the Telescript engine. But supporting company, General Magic Inc, withdrew telescript, and in 1997 introduced a new product Odyssey (Odyssey, 2001), which is a java-based system.

3.5 Summary

Recent research literature indicates that there is no consensus on the definition and the classification of software agents. But from the different definition, agents' essential properties can still be identified clearly, such as autonomous, goal directed, environment awareness, and communication/collaboration. In addition, agents essentially exist in a truly multi-dimensional space; the multi-dimensional classification can facilitate idea sharing among agents' community during research, design, and development. From this classification, we can clearly determine what characteristics agents have, and what they can do.

Recent years have seen various reasons for the emergence of mobile agents, such as limitation of static agents, www and Internet, and so on. Mobile agents have been applied in different areas, such as information search and retrieval, remote filtering and compression, maintenance, e-commerce, mobile devices and mobile users.
Java and Obliq are enabling technologies for building mobile agent frameworks. In particular, Java has become the mainstream of enabling technology, and some existing systems were rewritten using Java.

There are many existing mobile agents systems. Each has its own strengths. The five most important systems were reviewed in this chapter: Bee-gent, Alget, Hive, AgentTcl and Telescript.

In the next chapter, the application of intelligent agents in educational environments will be discussed and then we will match the potentials of mobile agents with deficiencies faced by Web-based learning systems.

Chapter 4

Mobile Agents and Web-based Learning

In Chapter 2 and Chapter 3, the Web-based learning environment and mobile agent technologies have been reviewed in detail. In this chapter, we shall review the existing works that applied intelligent agents in intelligent learning environments, and then discuss how mobile agents technologies can be employed to address the problems of Web-based learning environment.

4.1 Existing Application of Intelligent Agents in Learning Environments

The concept of agent was initiated in late 70's, and about 10 years later the intelligent agents technology was introduced into educational environments. Since then, the application of intelligent agents in educational environments has been developed rapidly, from only one agent to multiple agents and the new comer – animated pedagogical agents. In the following, we are going to review the existing situation of application of intelligent agents in learning environments.

4.1.1 Overview of Intelligent Agents in Educational Environments

Despite the concept of intelligent tutor has been pursued by Carbonell in 1970 (Carbonell, 1970), which the basic assumption was that one-on-one tutoring offers individual treatment tailored to individual student's needs. Until late 80's and early 90's, Chan and Baskin (Chan and Baskin, 1988; Chan and Baskin, 1990) proposed the idea that adds agents into ITSs and calls it the learning companion of the student, where student may interact with the learning companion in a different way. For example, the student can collaborate with or argue with the learning companion since the learning companion can make mistakes. A tutor agent there may interrupt, if needed. Thus, these two agents (learning companion and tutor) plus one student form an environment that can provide richer social context through different types of learning activities. Based on the theory of agent in learning environments, some systems have been developed, such as the Distributed West system (Chan, et al., 1992), the Reciprocal Tutoring (Chan, 1996), and my animal companions (Chang, et al., 2000).

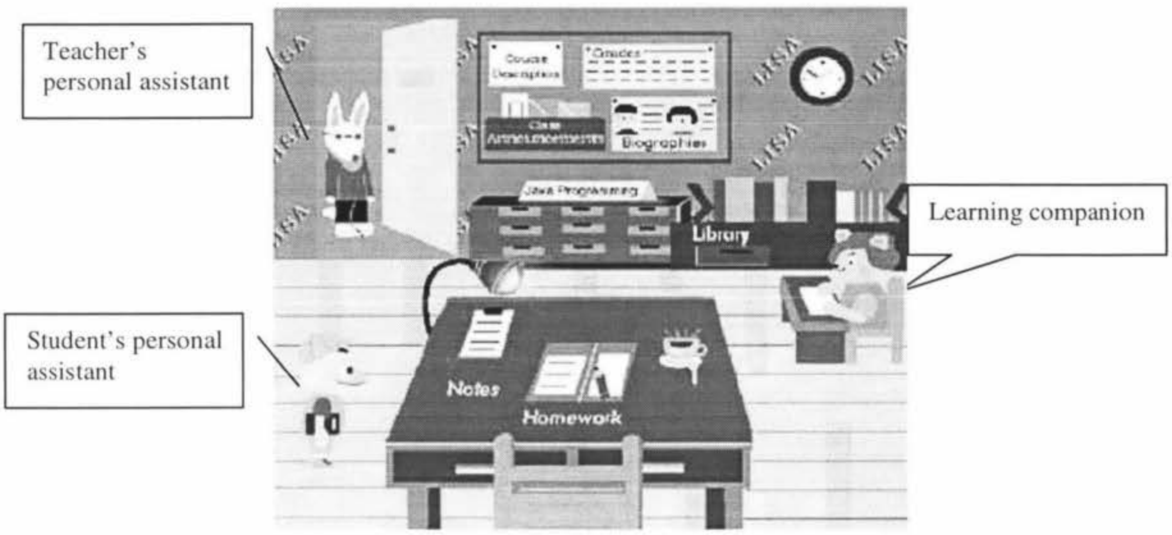


Figure 4.1: Intelligent educational agents (Chou, 1999)

As shown in Figure 4.1, Chou (1999) summarised that intelligent agents can be student's personal assistant, or teacher's personal assistant, or student's learning companion.

In recent years, a new paradigm has grown significantly for education and training: face-to-face interaction with intelligent, animated agents in interactive learning environments, which is called animated pedagogical agents. The new trend combines two distinct research areas: animated interface agents and knowledge-based learning environments. Pedagogical Agents (André, et al., 1997; Johnson, et al., 1998; Lester and Stone, 1997) are autonomous agents that facilitate human learning by interacting with students in learning environments. They have sufficient understanding of the learning context and subject matter to allow them to perform useful roles in computer-based learning environments. They can also manage their own behaviour in a consistent manner, responding to a variety of environmental stimuli. Animated pedagogical agents appear to the student as animated characters. They can engage in a continuous dialogue with the student, and emulate aspects of dialogue between a human teacher and student in instructional settings. They are able to give the user an impression of being lifelike and believable, producing behaviour that appears to the user as natural and appropriate to the role of a virtual instructor or guide. It is important that agents behaviours appear knowledgeable, attentive, helpful, concerned, etc (Johnson, et al., 1998; Lester, et al., 1999).

4.1.2 Examples of Animated Pedagogical Agents

Animated characters in the interface of pedagogical systems have become increasingly popular in the recent years. These characters are based on either cartoon-style drawings, real video or geometric 3D-models (André, et al., 1997). They may either inhabit a virtual world, or a constrained environment that consists solely of the agent. There are many examples of pedagogical agents constructed in research laboratories around the world.

USC/Information Sciences Institute's Centre for Advanced Research in Technology of Education (CARTE) has developed two agents: *Steve* (Soar Training Expert for Virtual Environments) and *Adele* (Agent for Distance Learning – Light Edition).

- *Steve* (Elliot, et al., 1999) is designed to interact with students in networked immersive virtual environments. It has been applied to naval training tasks such as operating engines aboard US Navy surface ships. Students can see the agent in stereoscopic 3D and hear him speak. The agent monitors the student's position and orientation in the environment. *Steve* software (Figure 4.2) is combined with 3D display and interaction software, simulation authoring software, speech recognition and generation software to produce a rich virtual environment in which students and agents can interact.

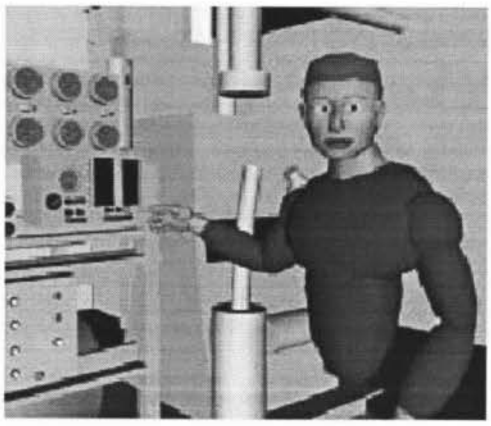


Figure 4.2: Steve demonstrates how to operate equipment US Navy ship (Steve, 2002).

- *Adele* (Johnson, et al., 1998), on the other hand, was designed to run on desktop platforms with conventional interfaces. The motivation behind the design of Adele was to broaden the applicability of pedagogical agent technology. *Adele* (Figure 4.3) is created to integrate into Web-based learning material and runs on a student's Web browser in a separate window. CARTE has developed Adele-based course material for medical education in family medicine and graduate level geriatric dentistry.

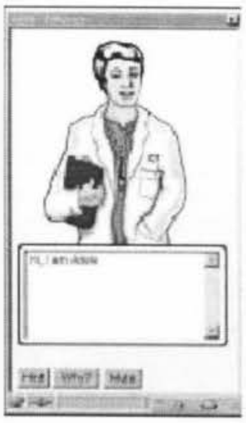

Hi, I am Adele

Figure4.3: Adele introducing herself (Johnson, et al., 1998)

North Carolina State University's Multimedia Laboratory has also developed two pedagogical agents: *Herman the Bug* and *Cosmo*. Both agents were developed to inhabit virtual environments.

- *Herman the Bug* (Elliot, et al., 1999; Lester, et al., 1999) is a 2D cartoon figure that inhabits Design-a-Plant, a design-centred learning environment to help middle school students understand botanical anatomy and physiology (Figure 4.4). This antenna-bearing creature advises students as they design plants to survive in various hypothetical environments. In the process of explaining, he performs a broad range of activities including walking, flying, swimming, shrinking and bungee jumping.

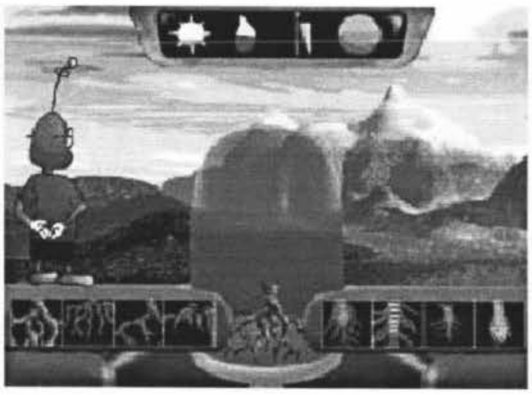


Figure 4.4: Herman the Bug inhabits Design-a-plant environment (IntelliMedia Initiative Projects, 2002)

- *Cosmo* (Lester, et al., 1997; Towns, et al., 1998), in contrast, is a 3D character that occupies the Internet Advisor, a learning environment for the domain of Internet packet routing. The agent assists students to solve problems such as finding a route avoiding high-traffic to transmit packets between network hosts. *Cosmo* has been used to investigate how to combine various agent behaviours in order to enhance deictic believability. Deictic believability is the ability to refer to objects in their environment through judicious combinations of speech, locomotion and gesture, in a manner similar to humans.

Herman the Bug has been used to investigate managing mixed initiative dialogues (Lester, et al., 1999). It was also used in large-scale empirical evaluation studies that have demonstrated the effectiveness of pedagogical agents in facilitating learning.

4.2 How Mobile Agents address the problems of Web-based learning

We have discussed the application of intelligent agents in educational environments. Most of them are used to provide help or company to users. In the research, the mobile agents technology was investigated to improve student's adaptivity in on-line or off-line learning environments.

In previous chapters, we have examined the benefits of mobile agents, and considering the limitations of Web-based learning environments, it becomes clear how mobile agents technology addresses those limitations very naturally. We discuss below how mobile agents can improve Web-based learning environments.

- In a Web-based learning environment, as shown in figure 4.5, mobile agents can be used to pre-fetch the domain content that will be requested shortly by the student, and report student's performance to the central server. This pre-fetch process is based on the real-time analysis of the student's behavior, and calculation of the probability of a request. Each student behavior, as they work in the Web-based learning environments, can be monitored and recorded by an intelligent interface agent, which runs on the student computer. Depending on the state of the network, an immediate request or a reservation can be placed by the mobile agent. In this way, end-to-end quality of the service can be improved for the delivery of distributed multimedia material, such as that represented by distance education. This agents technology avoids unnecessary networking delays, cope the bandwidth limitation and adapt the representations to students, based on the student performance.

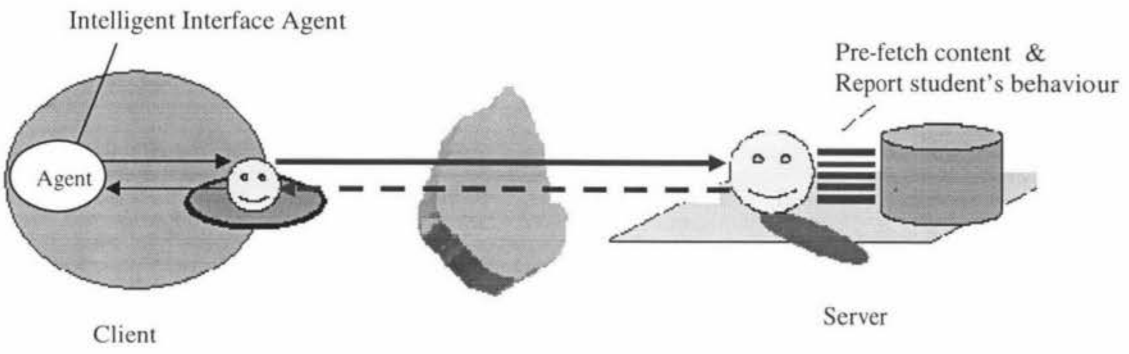


Figure 4.5: Scenario of mobile agent working

- In terms of mobile users, portable-computing devices such as laptops, palmtops, and electronic books will access Web-based learning environments. These devices have unreliable, low-bandwidth, high-latency telephone or wireless network connections. Mobile agents will be essential tools for allowing such access.

- Mobile agents offer application developers a new programming paradigm with higher-level abstraction and unified "process" and "object". In terms of scalability of the system and easy authoring, these features of mobile agents offer a flexible and effective philosophy on learning environment development, design, and scalability.

- In the future, Web-based learning environments can share resources through different systems, as shown in figure 4.6. Both the computers and networks on which Web-based systems are built are also heterogeneous in character. As mobile agent systems are generally computer and network independent, they support distributed systems and resources sharing.

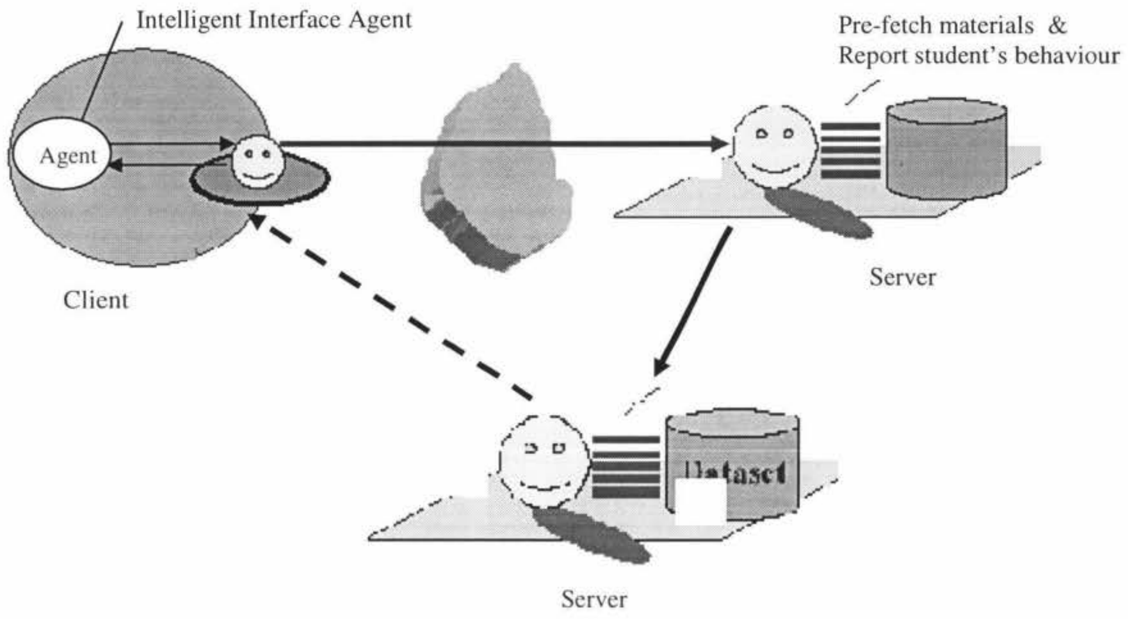


Figure 4.6: Scenario of mobile agent moving around distributes systems

- On another note, Web-based learning environments are ideal test beds for mobile agents technology. Till now, electronic commerce has been treated as the only important target for mobile agents technology. Money is involved, so the security is a key factor for agent technology. There are three types of security: agent-agent security, host-agent security, and agent-host security. Existing techniques can be successfully applied to protect agents from malicious agents, and hosts from malicious agents, but it is currently very hard to protect the agents from malicious hosts due to the possibility of the host changing the programming code of agent for some vested interests. This factor is one of main obstacle affecting mobile agents applications in the real world. However, in education sector, the agent-host security is not as important as in electronic commerce. This makes the Web-based learning environments as the most suitable test bed for mobile agents technology.

4.3 Summary

In this chapter, we briefly review the development of application of intelligent agents in educational environments. As discussed in the review, most of intelligent agents used in educational systems provided help or company to student, more like humanized assistants. However, the research investigated to use mobile agents technology to address some common deficiencies faced by Web-based learning environments, such as the adaptivity for individual student, connection limitation and slow access to course. Therefore, the advantages of mobile agents technology were discussed to match the deficiencies of Web-based learning environments.

In the next chapter, the system architecture with mobile agents will be discussed at a high level view.

Chapter 5

System Architecture

This chapter overviews the existing prototype - Student Modelling and Adaptivity in Web-based Learning System (SMAWLS) (Han, 2001), which is the platform used by this project to apply the mobile agent technology to empower the learning environment. An improved architecture and its adaptivity mechanisms are presented on Section 5.2.

5.1 The Basic Context of This Prototype

5.1.1 Architecture Overview of SMAWLS

The SMAWLS system is implemented with a *three-tier, client-server* architecture (Figure 5.1) (Han, 2001).

- Client tier: a web browser runs on the student computer. This tier is mainly responsible for handling adaptive presentation of course materials and dynamic tracking of the student working process.

- Middle tier: it resides at the server side, where it handles the student model initialisation and update logic. It is responsible for receiving client requests, processing the data contained in the requests, applying student model initialisation and update logic to the data, and generating a client response based on the updated student model.

- Database tier: the backend database resides on the server side and stores the data including student model, which is required by the middle tier.

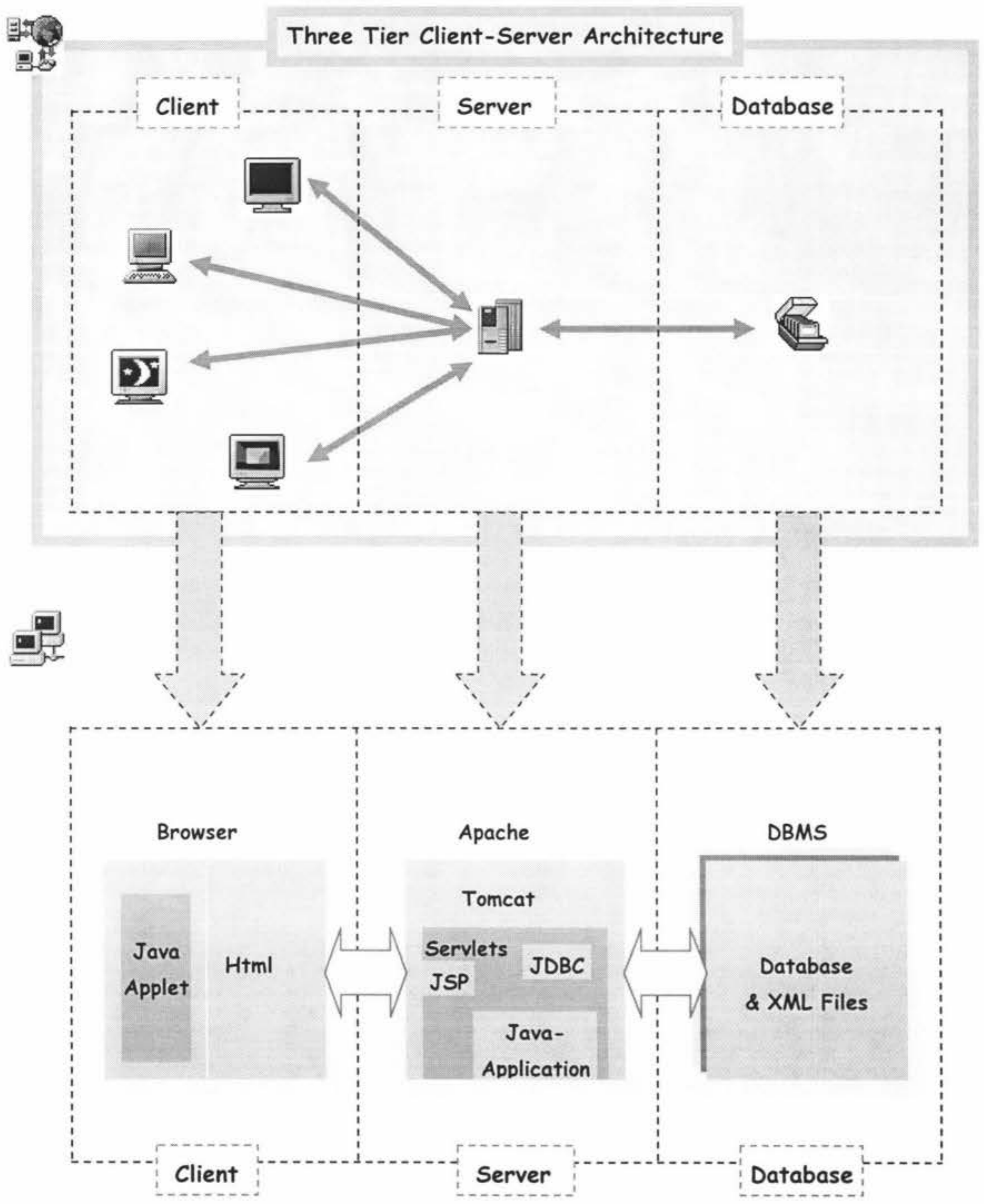


Figure 5.1: the architecture of SMAWLS (Han, 2001)

5.1.2 Adaptivity and Communication of SMAWLS

To implement the adaptivity within the system, the SMAWLS uses two students model databases: individual student model database and group student database.

- Individual student model: it is represented by an *overlay model*, in which the current state of a student's knowledge level is described as a subset of the domain model. The domain independent data within individual model is initialised by explicit questioning, while the domain specific data by using default values. Student can modify his/her domain independent data, but the domain specific data is updated by tracking student's study.

- Group student model: it is constructed by averaging corresponding values in the models for the individual students within a group. The statistical data with group student model is produced dynamically from individual model databases.

Both individual student model and group student models reside on the server.

The communication between clients and server is mainly implemented with HTTP socket techniques. When clients want to send a request to server, an HTTP socket connection is opened, and the request is sent. Then the client waits for the response from server.

5.2 The Architecture and adaptation mechanisms of the System

Although a number of attempts have been made to implement the concept of adaptation in learning systems, there have been no significant attempts to provide adaptation in Web-based learning. The circumstance largely limits the dynamic interaction in the Web-based learning systems, and it is one of the main reasons the Web-based systems do not exploit their full potential.

Kinshuk et al. (1999) pointed out that improving the dynamic adaptation in the system requires consideration of following criteria:

(a) adaptation with respect to current domain competence level of the learner;

(b) suitability with respect to domain content;

(c) adaptation with respect to the context in which the information is being presented.

Moving the adaptation mechanisms from standalone Intelligent Tutoring Systems to the Web-based learning systems, we have developed a suitable mechanisms to capture interactions over the Internet and to provide a continuous interaction pattern for a given student, even in off-line mode or in case the unreliable connection. This is the one of main concerns of this research. From this purpose, the high-level architecture of adaptation in the system is identified and presented as figure 5.2 below.

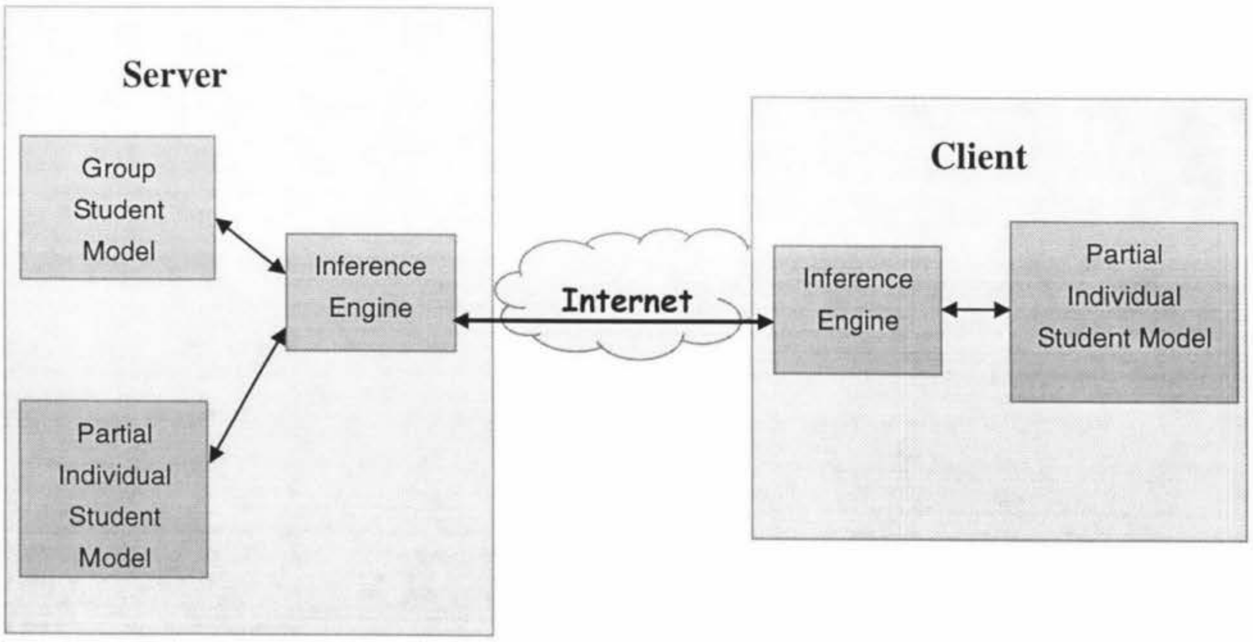


Figure 5.2: Architecture of adaptation mechanism

From figure 5.2, the system uses two separate student models:

1. *Individual student model*, which serves an individual student and contains the detailed information about particular student's domain competence level, preferences, interaction information and other relevant details. A partial individual student model resides on the central server and another partial individual student model on the user's machine (local server).
2. *Group student model*, which generalises various attributes over a number of students and attempts to classify students in various categories (for various attributes), resides on the host base.

This two-step modelling mechanisms will largely improve capturing interactions of a given student on the Web-based learning environment, and enables the system to provide adaptation at different granularity by having the following advantages.

- It allows the system to be much more flexible, particularly in the Web-based environment, where connectivity between host and client is not always guaranteed, and the quality of the connection often suffers from traffic congestion.
- It facilitates off-line adaptation with intermittent update of server side student model as and when possible;
- Student also can access his/her model and courses on-line from other computers with intermittent update of client side student model as and when possible;
- The host side student model allows adaptivity based on new information available from the domain experts and better adaptation procedures resulting from group student model.

The inference engines at both sides are employed to control the two way updates. The engine at server side is also used to extract common attributes from individual student models, process recommendations and initialisation for individual student.

5.3 High-level Architecture of Using Mobile Agents Technology in the System

Based on the architecture and mechanisms of adaptation in this system, the Mobile agents technology was used to implement the adaptation mechanisms. Use of mobile agents has facilitated a number of benefits in this system. Those main benefits are showed below:

- Mobile agents can package up a conversation and ship it to a destination host, where the interactions can take place locally, so it can avoid networking delays, improve the bandwidth limitation and adapt the representations to student based on the student performance. Although static agents can monitor and help student, but in term of web base environment, it still is limited by the same bottleneck as traditional approaches.

- Mobile agents technology naturally supports mobile devices and mobile users, it can promote learning environment's usability.

- Mobile agent systems are generally computer and network independent, they support distributed systems and resources sharing. So it can solve the heterogeneous problem of computers and networks on which web-base systems are built.

- Much of the earlier work made on static agents has been important for facilitating single-user environment. But the benefit of these intelligent agents or interface agents did not scale up well when they were placed alongside large real time applications, such as distance learning. But the characteristics of mobile agents technology are suitable for distributed environments.

Figure 5.3 describes the implementation of mobile agents technology in adaptation mechanism.

In high-level view, from figure 5.3, basically mobile agent interacts with client side inference engine to pick up data, which rely on individual student model in client side. Then mobile agent moves to host (or server) side. At host side, mobile agent performs all the processes needed, such as updating the partial individual student model based on summary of client side student model which brought by mobile agent, interacting with group student model to be updated if required. After mobile agent finishes all behaviors in host side, it gathers all information it need, and return to client side. Then it updates the client side individual student model. In addition, mobile agent approach can works in intermittent connectivity between client and host because mobile agent can be dispatched and works autonomously. The more details of how mobile agents work and its implementation are going to be discussed in chapter 6.

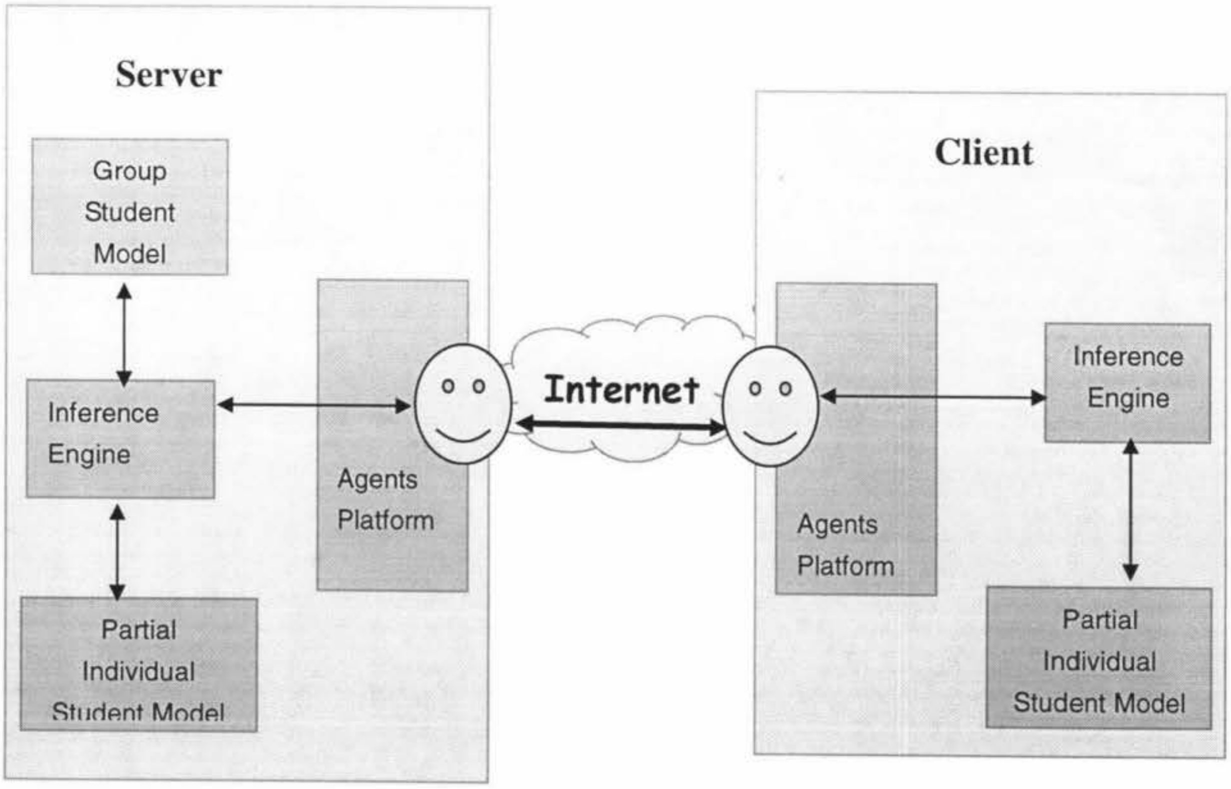


Figure 5.3: Architecture of adaptation mechanism using mobile agents

5.4 Summary

This chapter reviewed the architecture (figure 5.1) and adaptation of the Student Modelling and Adaptivity in Web-based Learning System (SMAWLS) (Han, 2001), where mobile agents technologies are used to facilitate the adaptation mechanisms in the system. Then the mechanism and the architecture, which use mobile agents to improve student adaptivity in Web-based learning systems were described at a high level view.

In the following chapter, main technologies used in the system are discussed, and the implementation details using Bee-gent framework are described.

Chapter 6

Toward the Prototype of the Web-based Learning System using Mobile Agents - Bee-gent

Based on the architecture and design of the system, this chapter discusses the techniques used in the system, followed by the details of the system implementation.

6.1 Technologies Applied in the Research

This section discusses the main technologies used to implement the prototype, including Bee-gent framework, Java, CodeWarrio, JDBC and InstantDB, and XML.

6.1.1 Bee-gent Framework

There are many mobile agent frameworks currently in existence, and Bee-gent framework is chosen to implement this project. Its characteristics are discussed in following.

6.1.1.1 Purposes of the Bee-gent Framework

With the advent of the "network society", a great variety of different software applications, such as WWW servers, databases, software packages, legacy systems, etc. are increasingly being connected together through use of the Internet, intranet, and other network systems. It is necessary to co-ordinate these disparate entities within a cohesive framework, which creates problems of network communication.

Though standard distributed systems are able to solve these problems, but, they rely on message based exchange with fixed connections resulting in inefficiencies when large numbers of, or widely varying, applications are used. Such situations are prohibitively expensive in both time and resources, unless network applications co-operate effectively through an appropriate communication infrastructure. The Bonding and Encapsulation Enhancement Agent (Bee-gent) provides a solution to address these network communication.

6.1.1.2 The Bee-gent Framework

Bee-gent, first release in 1999, is developed by Computer & Network Systems Laboratory, TOSHIBA Corporation. The Bee-gent framework is a new type of pure agent development framework for the advanced network society and its communication framework is based on the multi-agent model. Bee-gent provides applications with autonomous network behavior by "agentifying" them (i.e. providing an agent interface), and supports agent-based inter-application communication, facilitating co-operation and problem-solving.

The Bee-gent framework is comprised of two types of agents: agent wrappers and mediation agents.

- Agent Wrappers are used to agentify existing applications. The agent wrappers manage the states of the applications, which are wrapped around, and invoke the applications when necessary.

- Mediation Agents support inter-application co-ordination by handling all communications between applications. The mediation agents move from the site of an application to another where they interact with the agent wrappers.

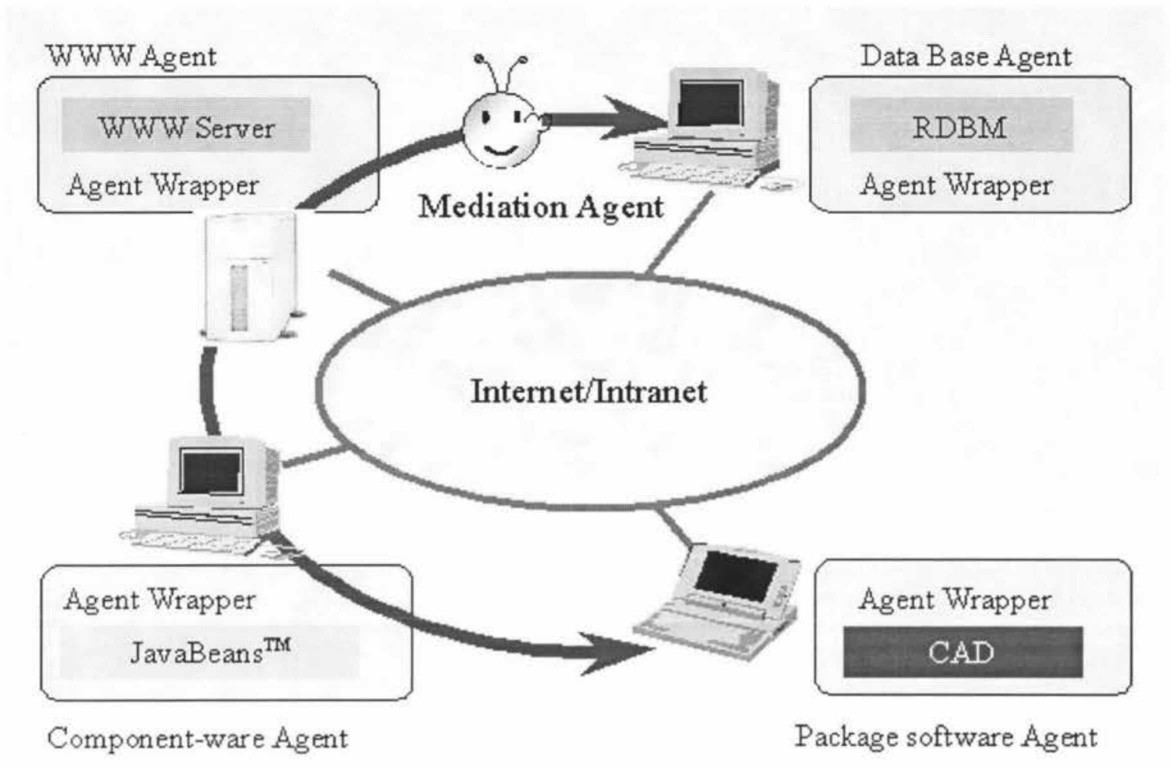


Figure 6.1: the scenario of Bee-gent framework existing

Inter-application co-ordination is handled through the agent wrappers generating and receiving requests, which are transported around by the mediation agents (figure 6.1).

6.1.1.3 Mechanisms of the Bee-gent Framework

In the high-level view, the Bee-gent framework uses agent wrappers and mediation agents to accomplish its functionality. Communication within the network is managed through the agent wrappers generating and receiving requests, and the mediation agents as vehicle to transport the messages.

The agent wrappers work as the masters, which manage the applications wrapped and decide when to send out mediation agents that carry request messages. The agent wrappers also are responsible for receiving from mediation agents or talking with mediation agents.

The mediation agents are the mobile agents that move from one place to another carrying their programs and data and carry out the processing. Instead of communicating with the others remotely, they move to the location of the others and communicate with them locally through agent wrappers (figure 6.2). Therefore the communication lines can be disconnected after the agents move. In addition, the frequency of the communication with the others can be reduced. The mediation agents do more than just transport the messages; they are able to respond to the nature of the request to determine the best course of action (Kawamura et al., 1999). The relationship of the existing applications, the agent wrappers and the mediation agents is shown in the figure below.

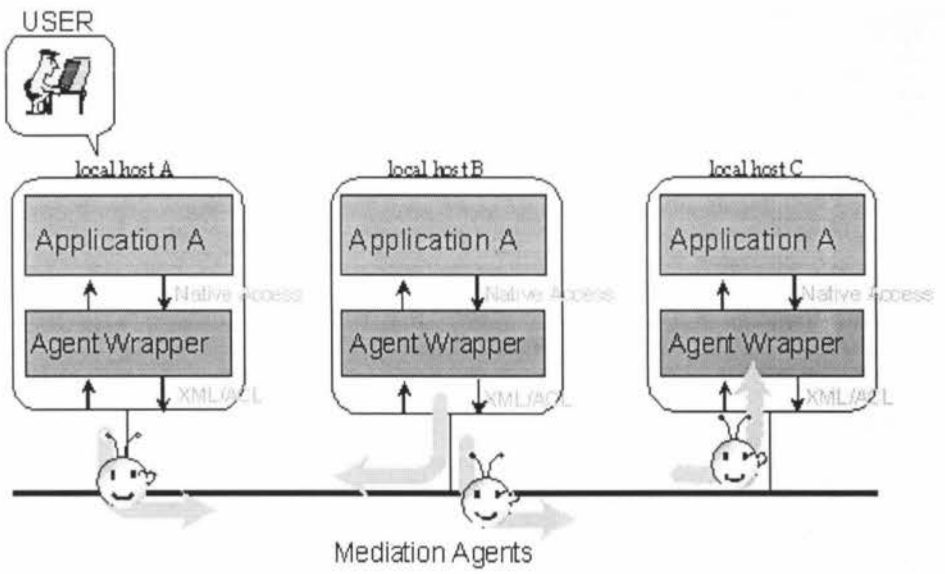


Figure 6.2: the relationship between agent wrapper, mediation agents and applications

In the low-level view, the Bee-gent resides on the top of Java technology. The Bee-gent Framework includes the basic class library for implementing the mediation agents and the agent wrappers, which are compatible with Java environment. The framework also provides the Rapid Application Development (RAD) tool based on the design patterns, in which actions of the mediation agents can be displayed in the form of state transition diagrams. Interaction patterns between the mediation agents and the agent wrappers can be embedded (figure 6.3).

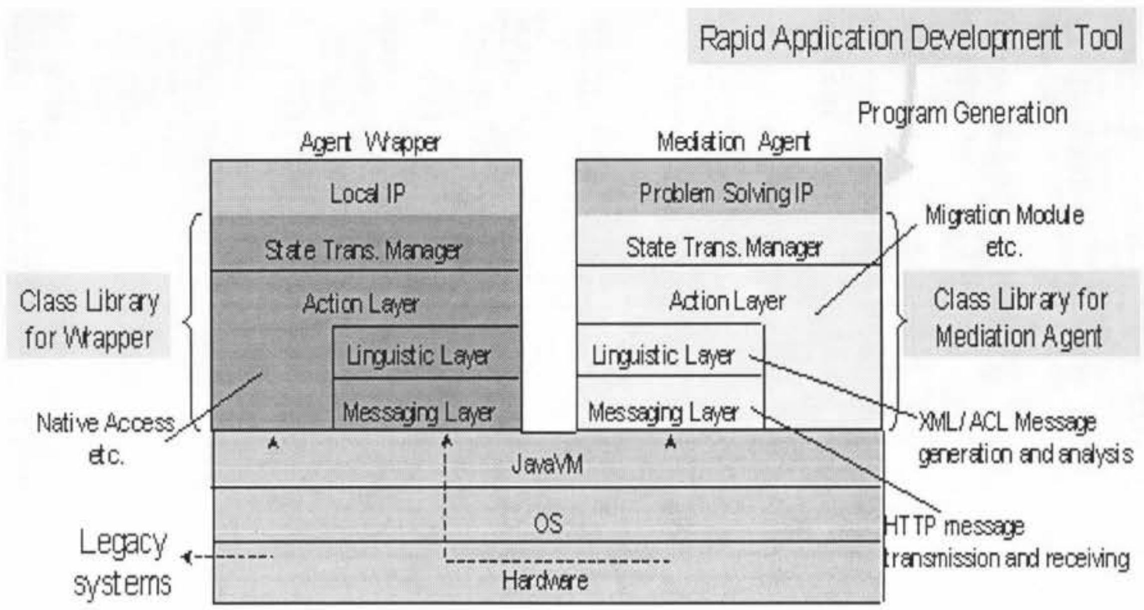


Figure 6.3: the layer structure of Bee-gent and its working environment.

6.1.1.4 Technological Characteristics

Some main technological features of Bee-gent framework shall be described as follows.

Mobile Agents - The Mediation Agents

The mediation agents are developed as mobile agents that can migrate around the network by themselves, keeping their coordination procedures, data and states, as they travel. This mechanism gives the mediation agents ability to execute their programs upon arrival at the destination automatically. Thus mediation agents can exchange information with the local application agents at the destination and avoid consuming network resources. Figure 6.4 shows the scenario how the mediation agent saves the network traffic.

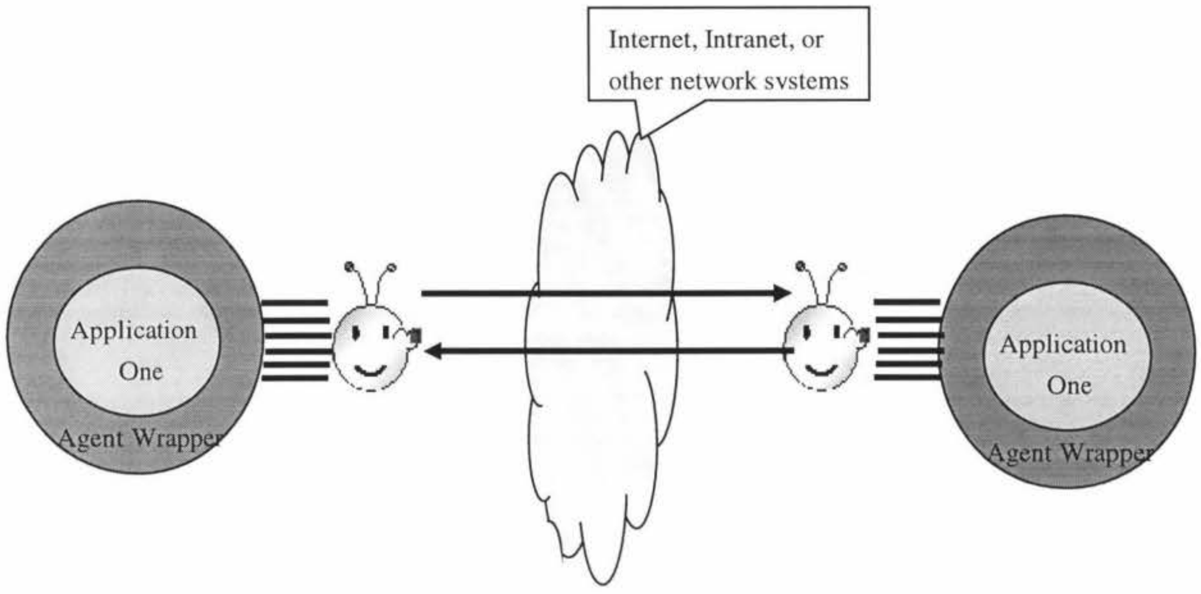


Figure 6.4: mediation agent as mobile agent

There are a number of reasons that can cause network load reduction:

A). Compared to the traditional message-based systems where conversations consist of many requests and correspondences, communication frequency in the Bee-gent environment is less because the conversations can take place locally.

B). The mediation agents are able to decide themselves when they should move to their next destination, whether they have to go back to their initiator, and when. This autonomous ability of mediation agents also can reduce the network traffic.

C). The mobile agents enable the communication connections to be disconnected after the launch of the mediation agents. This capability of mediation agent reduces frequency of communication significantly.

Interaction protocols with Bee-gent

In agents' society, ongoing conversations between agents often fall into typical patterns. "In such cases, certain message sequences are expected, and, at any point in the conversation, other messages are expected to follow. These typical patterns of message exchange are called *interaction protocols"* (IPs, 2002). In Bee-gent, all co-ordination procedures for goals are realised by conversations between mediation agents and the wrapper agents. For this purpose, the behaviour of both of the mediation agents and the agent wrappers is defined by the interaction protocols. The interaction protocols are represented by state diagrams, which consist of states and transitions, and are defined by specifying the preconditions of the possible states, the actions and the transition rules. (Figure 6.5)

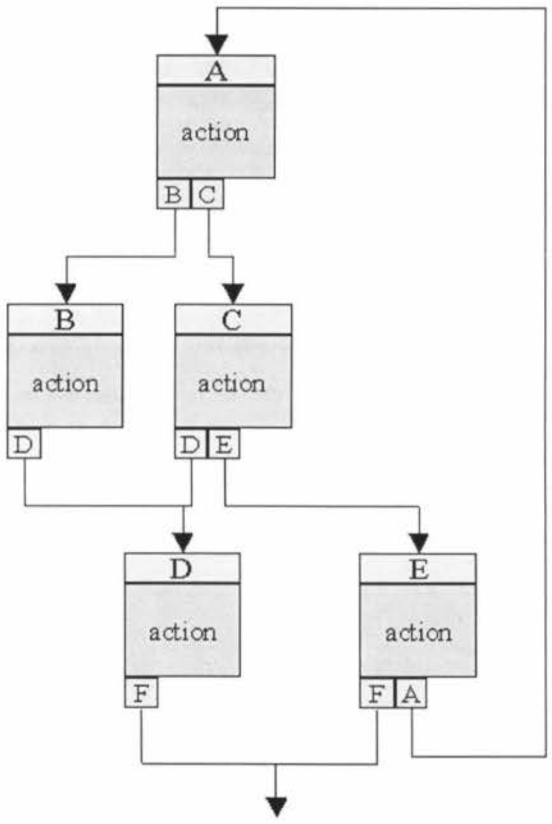


Figure 6.5: the state diagram of interaction protocols

The green blocks are the preconditions, which determine changing to a specified state. Action defined in a state takes place when current state coincides with the precondition. According to the result of the action, the transition rules define the next state to move further.

High consistency of co-ordination activities and separation of the local and global procedures

Interactive Protocols (IPs) are descriptions of interaction processes involving the exchange of messages, allowing problem solving via different system components. It is challenging but necessary to ensure consistency of the IPs among the all components.

In IPs within the Bee-gent framework, the interaction process for problem solving is described within the mediation agent in a unified and non-distributed fashion, and the local processes are described in a distributed manner in the agent wrappers. Thus, the Bee-gent environment can easily ensure consistency of the co-ordination procedures, and the global problem solving processes and locally dependent processes are completely separated. As a result, the systems developed using Bee-gent framework can be maintained and enlarged efficiently.

A Pure Agents System

Bee-gent framework is a 100% pure agent system. Bee-gent wraps all applications within the agent systems. Within a Bee-gent distributed system, the applications become agents, all messages are carried around by agents, and all communication that takes place between software applications carries out by agents. This pure agent framework offers developers flexibility to build and maintain real open distributed systems, and facilitates autonomous behavior on the part of each system component. Bee-gent framework realizes an open structure in which new applications (databases, WWW servers, and so forth) can be introduced dynamically.

Message Exchange among Agents: XML/ACL

Information exchange is performed using eXtensible Markup Language (XML) notation. XML is the standard way to identify and describe data on the web, and enables a universal standard syntax for exchanging data (XML, 2002). XML specifies a rigorous, text-based way to represent the structure inherent in data so that it can be authored and interpreted unambiguously. It is widely implementable and becoming the standard protocol for data exchange.

Bee-gent framework makes it compatible with other systems and flexible in the systems development. The agents can easily interpret XML in order to understand the contents of the information that they handle.

The information exchanged around is represented using XML, and the content of the information uses Agent Communication Language (ACL), that is agents use it to communicate.

ACL is an agent communication language based on the speech-act theory. The Speech act theory was developed and published by the Oxford philosopher J. L. Austin as "How to Do Things with Words" in 1975 (Austin, 1975). In this theory, the "performative" means purposeful actions performed during conversations between the communicators. Sending a performative in a message to the partner results in an intention of an action. The international standardization organization for agent, Foundation for Intelligent Physical Agents (FIPA) produces software standards for heterogeneous and interacting agents and agent-based systems (FIPA, 2001a), has specified their standards with the ACL. By using the ACL standards within Bee-gent framework, the systems developed by using Bee-gent framework are able to communicate and coordinate with other agents systems that use the FIPA standards.

Compatible with Java Technology

Bee-gent framework was built on the top of Java technology. The current version can work with Java 2, and offers the APIs that are easily used with Java environment. Based upon the Java technology, Bee-gent framework can be used widely because

Java has become the mainstream programming technology, especially in an Internet and other network systems environments.

6.1.2 Java

Java is a powerful, more or less platform-independent programming language. The initial commercial version of Java was released in 1995 (Sun Microsystems, Inc., 2002). Since then, Java, called as the Internet language, has grown rapidly in popularity and usage, because of its benefits of being simple, object-oriented, distributed, interpreted, robust, secure, architecturally neutral, portable, and dynamic.

There are three editions: Java 2 Platform Micro Edition (J2ME technology), Java 2 Platform, Standard Edition (J2SE technology), and the Java 2 Platform Enterprise Edition (J2EE technology).

- The J2SE platform is a fast and secure foundation for building and deploying client-side enterprise applications.
- The J2EE simplifies enterprise applications by basing them on standardized, modular and re-usable components Enterprise JavaBeans (EJB).
- The J2ME specifically addresses the range of extremely tiny commodities such as smart cards or pagers.

The J2SE technology is used in this system because the prototype developed in this project is application.

6.1.3 CodeWarrior

The CodeWarrior developed by Metrowerks (Metrowerks, 2002) is an integrated Java and C/C++ development environment (IDE) that provides tools and wizards for easy and rapid application development, such as a sophisticated project manager and build system, text editor, class browser, C/C++ and Java™ compilers, linkers and debuggers.

Within the same integrated development environment, applications in Java, C and C++ can be developed. The CodeWarrior supports Java applications using the latest JDK, and provides visual drag and drop rapid application development (RAD) tools that automatically generate your code when you create the design. The CodeWarrior 5.5 was used to develop this system.

6.1.4 JDBC and InstantDB

Java Database Connectivity (JDBC) is an API that provides facilities for accessing the tabular data sources from Java programming language (Sun Microsystems, Inc., 2002a). JDBC provides cross-DBMS connectivity to a wide range of SQL databases and connectivity to other tabular data sources, such as spreadsheets or flat files.

The core technology for supporting JDBC is JDBC drivers, which are engines and bridges that work between JDBC technology and database management systems or other tabular data sources. In this prototype, JDBC technology was applied to implement data access and retrieval from the database.

The database is implemented in the InstantDB (Lutris Technologies, Inc., 2002), which is a relational database management system. InstantDB is written purely in Java and designed specifically for embedding with Java applications. It provides numerous benefits to application developers, mainly including:

- it integrates seamlessly with any Java application and is implemented as a library of Java classes;
- JDBC2.0 driver;

- it supports Windows, WinCE, Linux, and popular Unix platforms

6.1.5 XML

Extensible Mark-up Language (XML) is a mark-up language for documents containing structured information that contains both content (words, pictures, etc.) and some indication of what role that content plays (XML, 2002a).

The XML supports richly structured documents usage over the web, because the other alternatives, such as HTML and SGML, are not practical for this purpose.

The XML specification defines a standard way to add mark-up to documents. It is defined by a number of related specifications (W3C, 2002):

- XML 1.0: defines the syntax of XML. The XML specification is the primary focus of this article.
- XML pointer language (Xpointer) and XML Linking Language (Xlink): defines a standard way to represent links between resources.
- Extensible style language (XSL): defines the standard stylesheet language for XML.

Since XML powerfully supports an easy and rich structured document exchange over the web, it has become the ad hoc standard for data exchange over the Internet. Java is an ideal companion to XML and both provide a natural match for the creation of applications that exploit the web of information. Java provides Java API for XML Parsing (JAXP) (Sun Microsystems, Inc., 2002b) to enables basic functionality for reading, manipulating, and generating XML documents through pure Java APIs. The JAXP provides a standard way for any XML-conformant parser to be accessed by an application.

XML is used in this prototype to implement the message and data exchange.

6.2 The System Design Using Bee-gent

The Bee-gent framework is used to implement the mobile agent in the system. The wrapper agents are used to wrap the client and server sides systems, and the mediation agents are used to perform the communication and exchange information with the wrapper agents (figure 6.6)

From figure 6.6, the main process scenarios within the system are:

A) At server side, the inference engine interacts with the group student model and partial individual student model through the database interface, and checks whether something has been updated within the database and the update has been sent to client periodically. If something needs to be sent to clients, the inference engine notices the wrapper agent with the information is going to send. The wrapper agent creates a mediation agent carrying the information and related program and launches the mediation agent. If the mediation agent can not find the target clients, it will back to notice the update process fails. If the mediation agent reaches the target client, it will communicate and exchange information with the client's wrapper agent and the client's inference agent communicates with its wrapper agent to get the data and processes them to update the client side student model.

B) To update the group student model at the server side, the server side engine has to extract information from individual student model.

C) At the client side, the procedure is similar as those described above, except the inference engine only interacts with the partial individual student model.

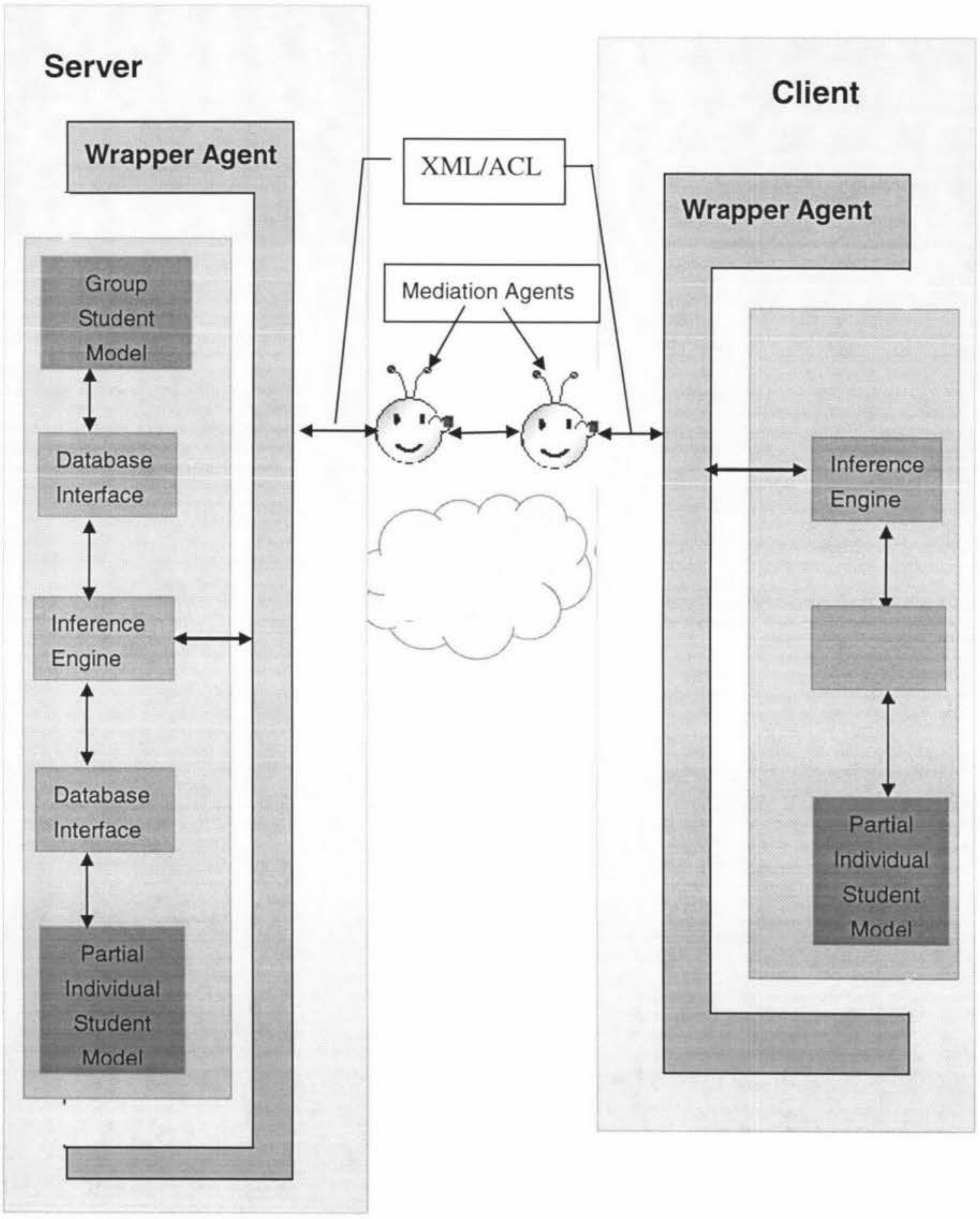


Figure 6.6: Architecture of adaptation mechanism using Bee-gent

6.3 Requirements

6.3.1 Requirements for mediation agents

The functionality of mediation agents, whenever they are created by server or client, is the same. The requirements for mediation agent are:

- Communicating with destination servers and exchanging information with servers;
- Notifying its creator when the connection is not available, or the update processes fail;
- Moving to different servers;
- Once launched, they are not affected by the situation that sender is on-line or off-line.

6.3.2 Requirements for Wrapper Agents

Both server side and client side wrapper agents simply are the interfaces between mediation agents and inference engines.

When wrapper agents receive visiting mediation agents, they act as translators, which pass the information back and forward between mediation agents and inference engines.

When wrapper agents receive demands from inference agents to send mediation agents, wrapper agents create mediation agents and launch them.

6.3.3 Requirements for Inference Engines

The inference engines are the core parts of the system. They initialize most processes and control the process flow.

Requirements of client side inference engine are:

- Periodically checking the update information in individual student model and determining whether the update of server side individual student model has to take place. If yes, the data must be passed to wrapper agent for creating mediation agents.
- If update fail because the connection problems or server can not be found, the inference engine has to inform the student model database.
- Receiving information from its corresponding wrapper agents and update the student model.

Requirements of server side inference engine include the requirements for client side inference engine, and plus the following points:

- Periodically extracting and generalizing the student model attributes from all individual student models within one group to update the group student model dynamically.
- Initializing a new student model based on the group student model by sending a mediation agent to client.
- Classifying students within the group in group student model.

6.3.4 Requirements for Databases

Requirements of both server side and client side individual student model are the same. They are based on the student model database in the SMAWLS system (Han, 2001), add the mechanisms to indicate the state of update.

Group student model stores the common attributes for the same group and can dynamically add some more columns depended on the results of inferring by inference engine.

6.4 The Implementation Details

6.4.1 Services between Client and Server

- Figure 6.7 presents the services, which happen between client and server, when client intends to update the individual student model on the server. For this prototype, a timer is set up to fire the event to notice the inference engine to check the individual student model on the client. The basic algorithm is described below.

```
If      the inference engine receives an event from the timer
Then    check the new data flag within the student model
        If      the new data flag is No
        Then    do nothing
        Else    get the new data and set the new data flag to No
                send request to agent wrapper asking to create mobile agent
                agent wrapper creates and launch a mediation agent with the new data

                If      client is offline
                Then    notice inference to set the new data flag back to Yes
                Elseif  server not found
                Then    notice inference to set the new data flag back to Yes
                Else    update is done

                        If      more than one server
                        Then    mediation agent moves to next server
                        If      server wants to send data to client by the agent
                        Then    the mediation agent get the data and back home
End If
```

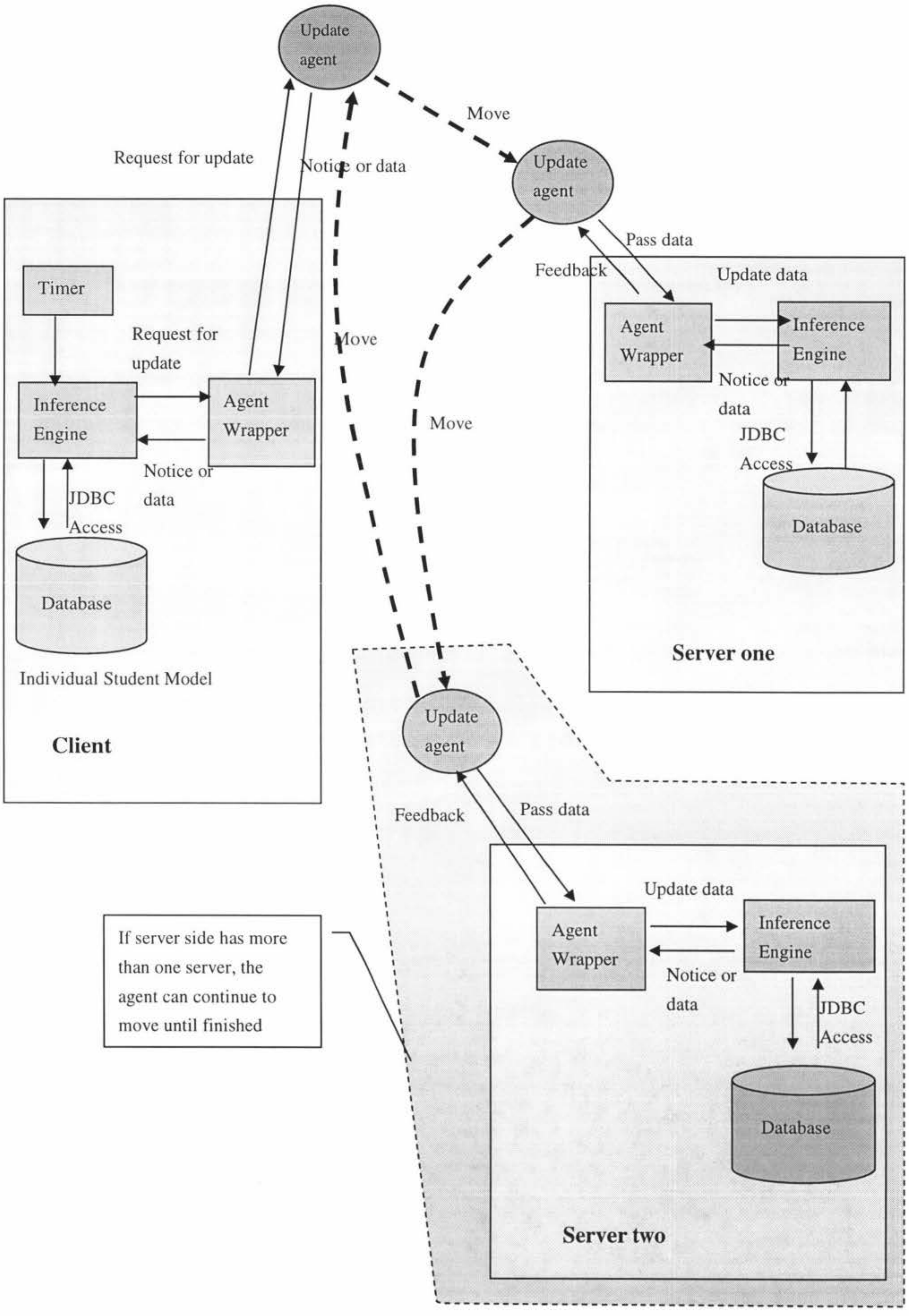


Figure 6.7: the summary of interaction protocols from client to server

- In the situation that server's individual student model has to update the client side corresponding student model, the mechanism and procedure are almost the same as the above case that client side want to update the individual student model on the server side.

- Initializing the new student's profile: in the SMSWLS system, the individual student model database consists of domain-independent data and domain-specific data. The domain independent data is initialised by explicit questioning or the default value when the student chooses not to specify his/her preferences. With generalising the domain-independent data of individual student model within a group, these can be used to initialise the new student's domain independent data with his/her student model (see figure 6.8). The basic algorithm for initialising the new student's model is presented below.

```
If the new student does not define his/her preference in domain independent data
Then send a request by mobile agent from client to server to retrieve the domain
        independent data from group student model, which is generalized from the
        individual student model within the group
        If      more than one server
        Then    mediation agent moves to next server until finished
End If
```

In the future work, the group student model and the inference engine could be empowered to generalise student performance and behaviour to provide better adaptation when presenting courseware and guiding student through study. In the SMSWLS system, the domain-specific data are initialised using default values. Ideally, the initialisation of the domain-specific data should be based on the data produced and improved dynamically according to the student study performance and result within the group. In this system, these basic ideas are experimented surface.

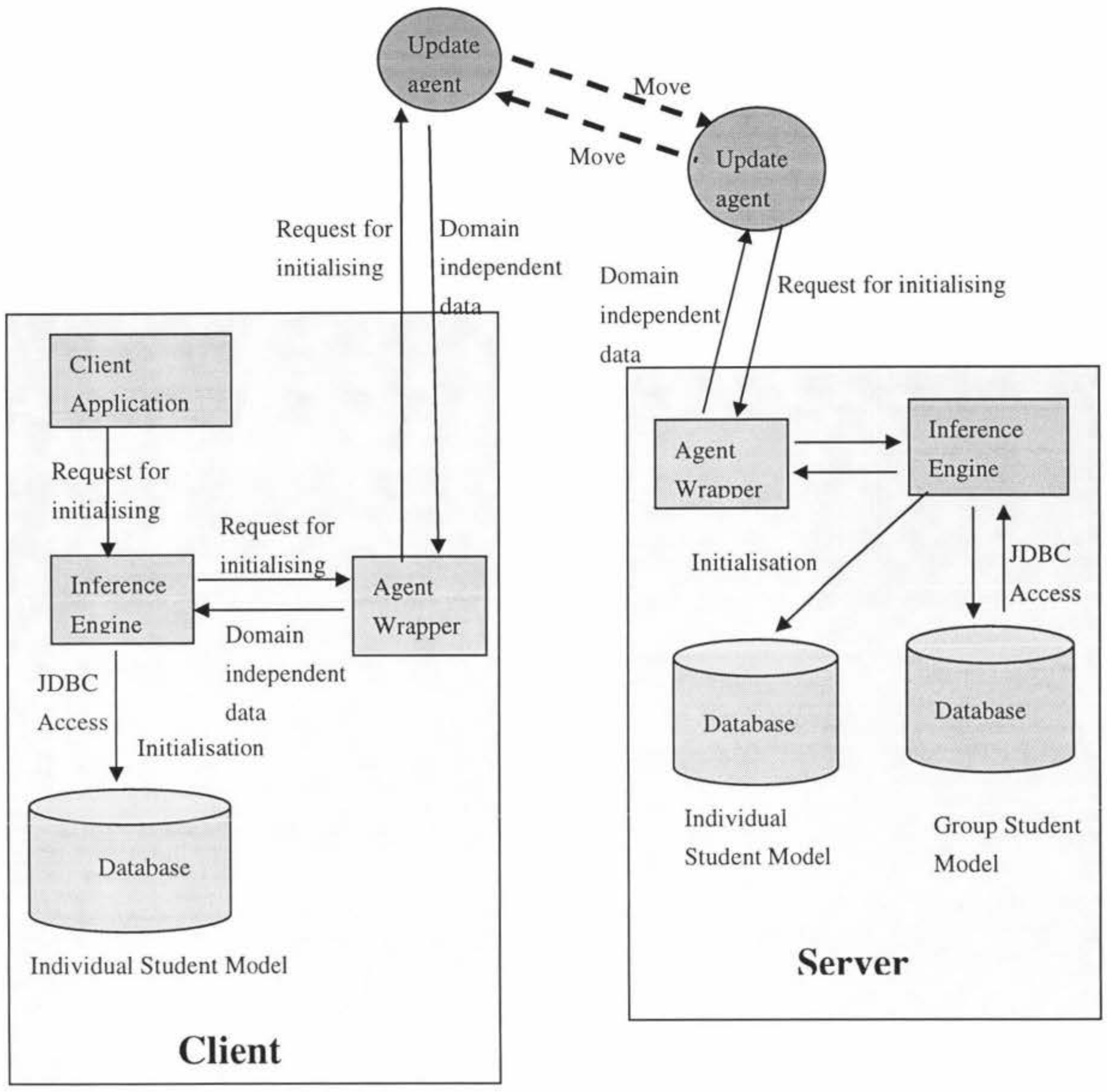


Figure 6.8: initialising the individual student model by using group student model data

6.4.2 Interaction Protocols: for individual student model update

Interaction protocols define the behaviour of the different parties within the system, which means system functionality is based on the interaction protocols and achieved by message exchanges between parties. Thus, the communication and how to perform the tasks within the system is programmed based on the interaction protocols.

The interaction protocols are presented by state transition diagrams. The state transition diagrams of each agent wrapper and the mediation agent when client intends to update the individual student model on the server are shown in Figure 6.9.

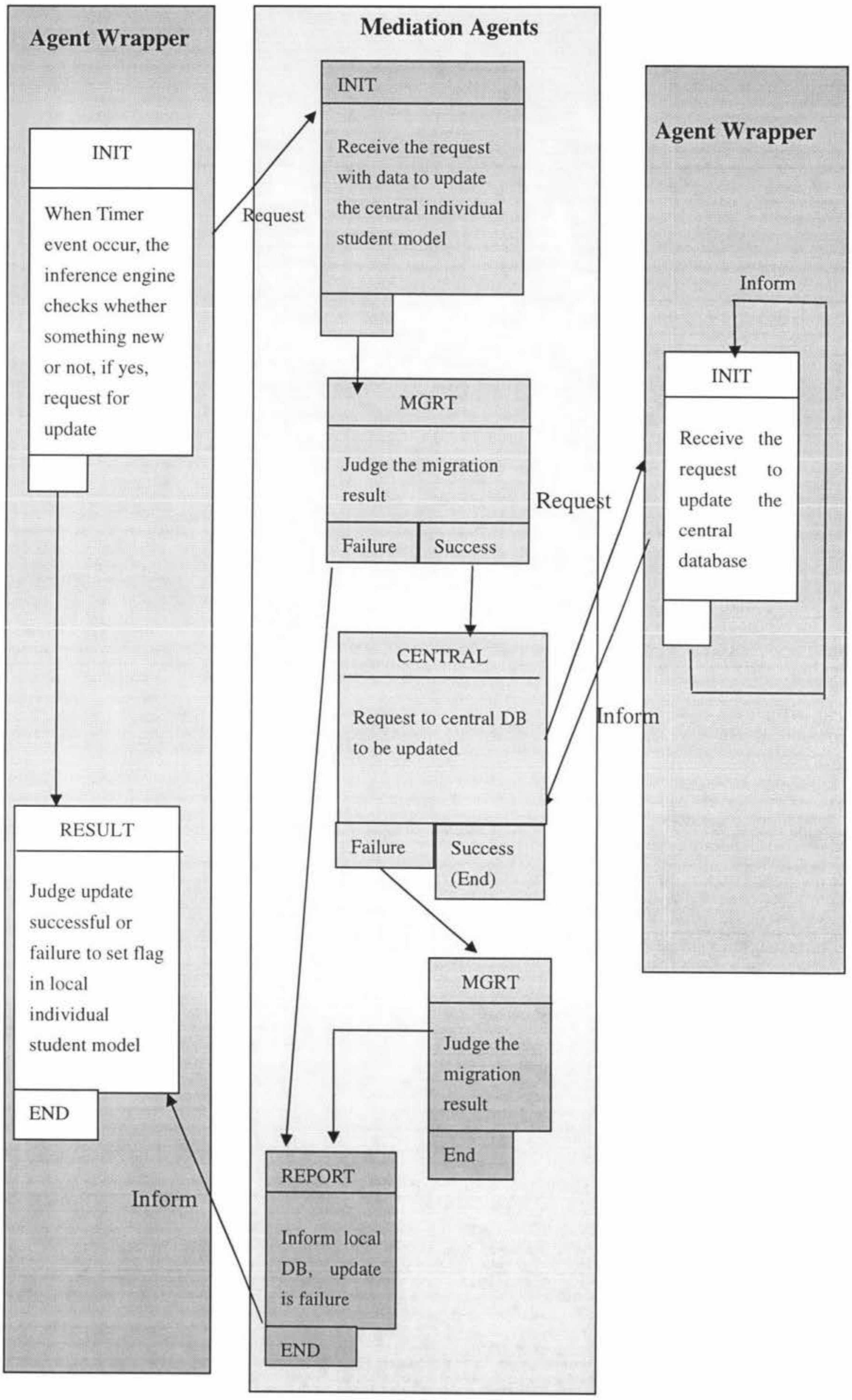

Agent Wrapper
Mediation Agents
Agent Wrapper
INIT
When Timer event occur, the inference engine checks whether something new or not, if yes, request for update
Request
INIT
Receive the request with data to update the central individual student model
MGRT
Judge the migration result
Failure
Success
Request
Inform
INIT
Receive the request to update the central database
CENTRAL
Request to central DB to be updated
Inform
Failure
Success (End)
RESULT
Judge update successful or failure to set flag in local individual student model
END
MGRT
Judge the migration result
End
REPORT
Inform local DB, update is failure
END
Inform

Figure 6.9: the state transition diagram for procedure from client to server

From server side's individual student model to update the client side corresponding student model, the state transition diagram is the similar as the one from client side to server side, the only difference is that it goes another opposite way from server to client.

6.4.3 Interaction protocols: for individual student model initializing from group student model

When a new student chooses not to define his/her preferences for domain independent data, the corresponding domain independent data within group student model generalized from students' common preferences will initialize the student's preferences. Figure 6.10 is the state transition diagrams of each agent wrapper and the mediation agent.

6.5 Organisation and Detail of Program Files

This prototype consists of two relatively separate subsystems: one runs on local or client side, and another runs on central or server side. Both subsystems are organised in similar way, but have different functionality, especially at the very lower level, although some similar functions exist. Each subsystem has the following parts: interface, local database related processes, wrapper agent between local processes and mobile agent, and mobile agent. In addition, each subsystem also comes with XML files and address registration files, which are important when deploying the system.

Since the main differences between client and server side occur only when the system updates or processes the data, the following sections mainly present the client side system.

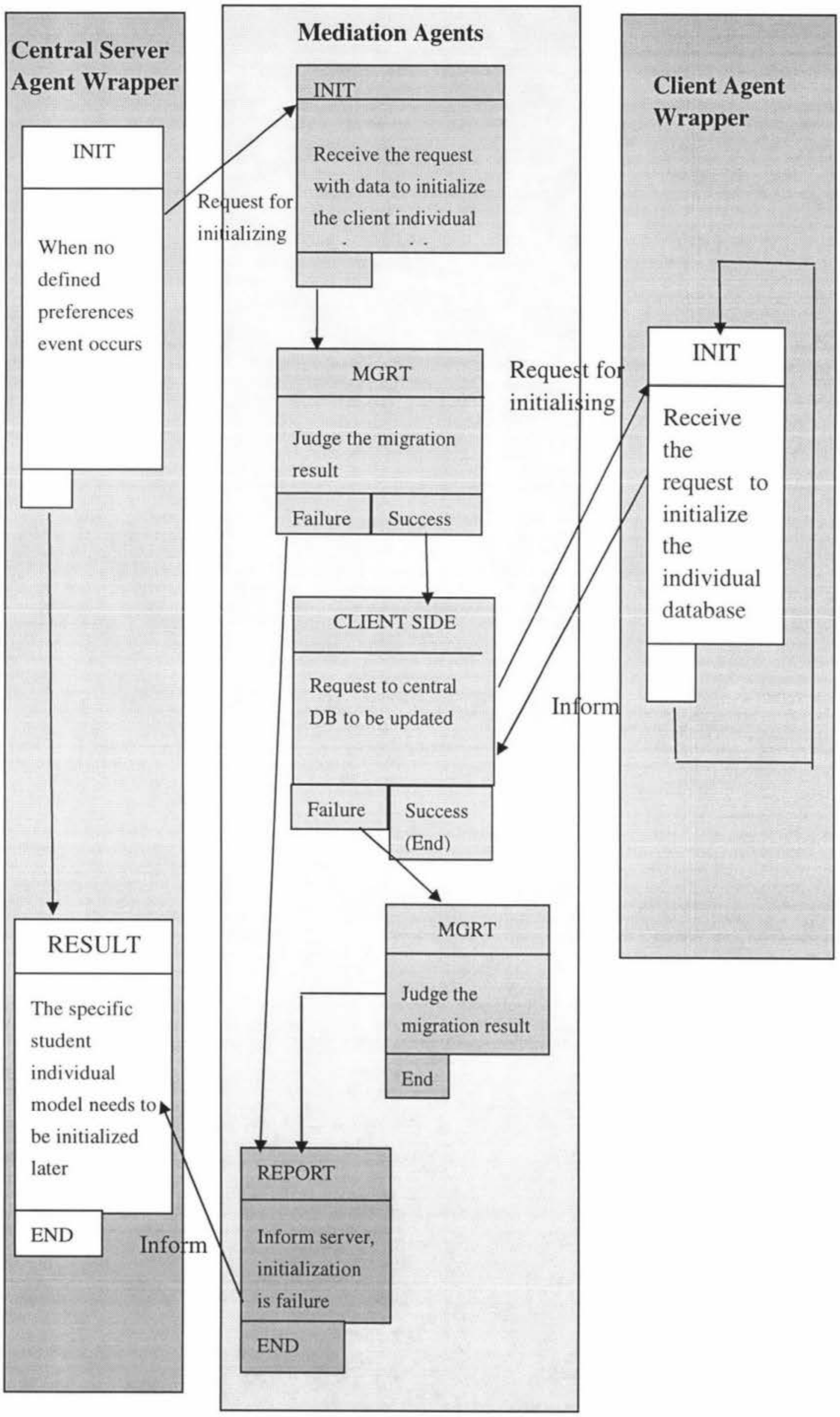

Central Server Agent Wrapper
INIT
When no defined preferences event occurs
Request for initializing
Mediation Agents
INIT
Receive the request with data to initialize the client individual
MGRT
Judge the migration result
Failure
Success
CLIENT SIDE
Request to central DB to be updated
Failure
Success (End)
MGRT
Judge the migration result
End
REPORT
Inform server, initialization is failure
END
Client Agent Wrapper
INIT
Receive the request to initialize the individual database
Request for initialising
Inform
RESULT
The specific student individual model needs to be initialized later
END
Inform

Figure 6.10: the state transition diagram for initialising individual student model

6.5.1 Interface

In term of the actual mobile agent system, there is no interface needed in the system. This is because the intended system is responsible for communicating between client and server and updating the databases of both sides automatically and these processes happen in the background. However, a simple interface is designed and developed for demonstration purposes where users can interact and system displays the status of mobile agents.

6.5.1.1 Functionality

Both client side and server side have similar functionality (figure 6.11):

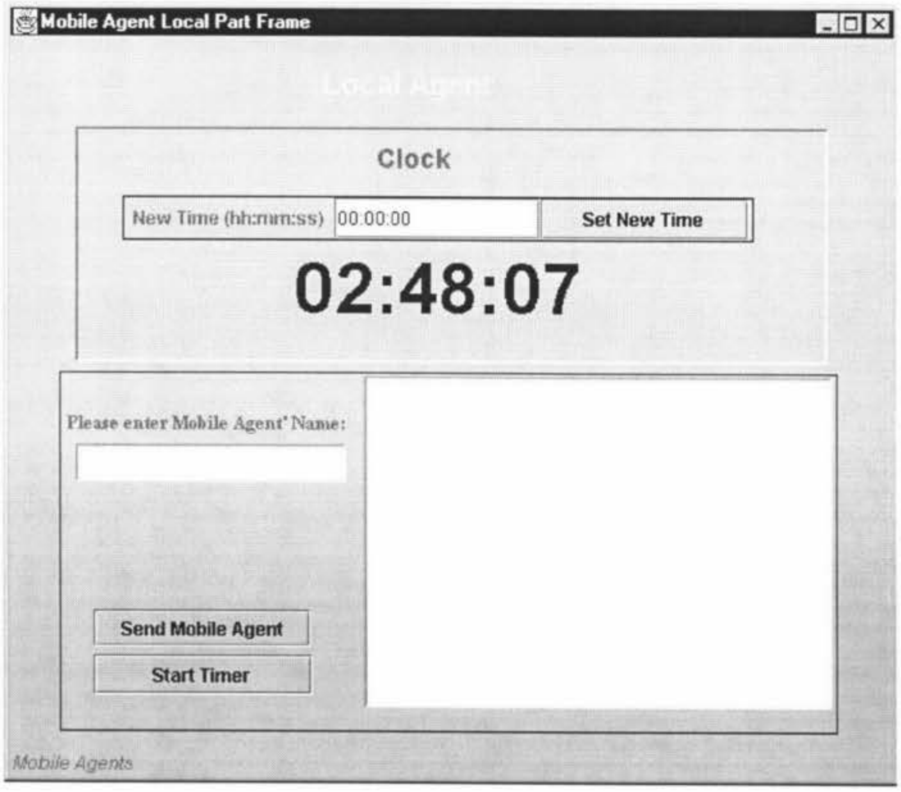


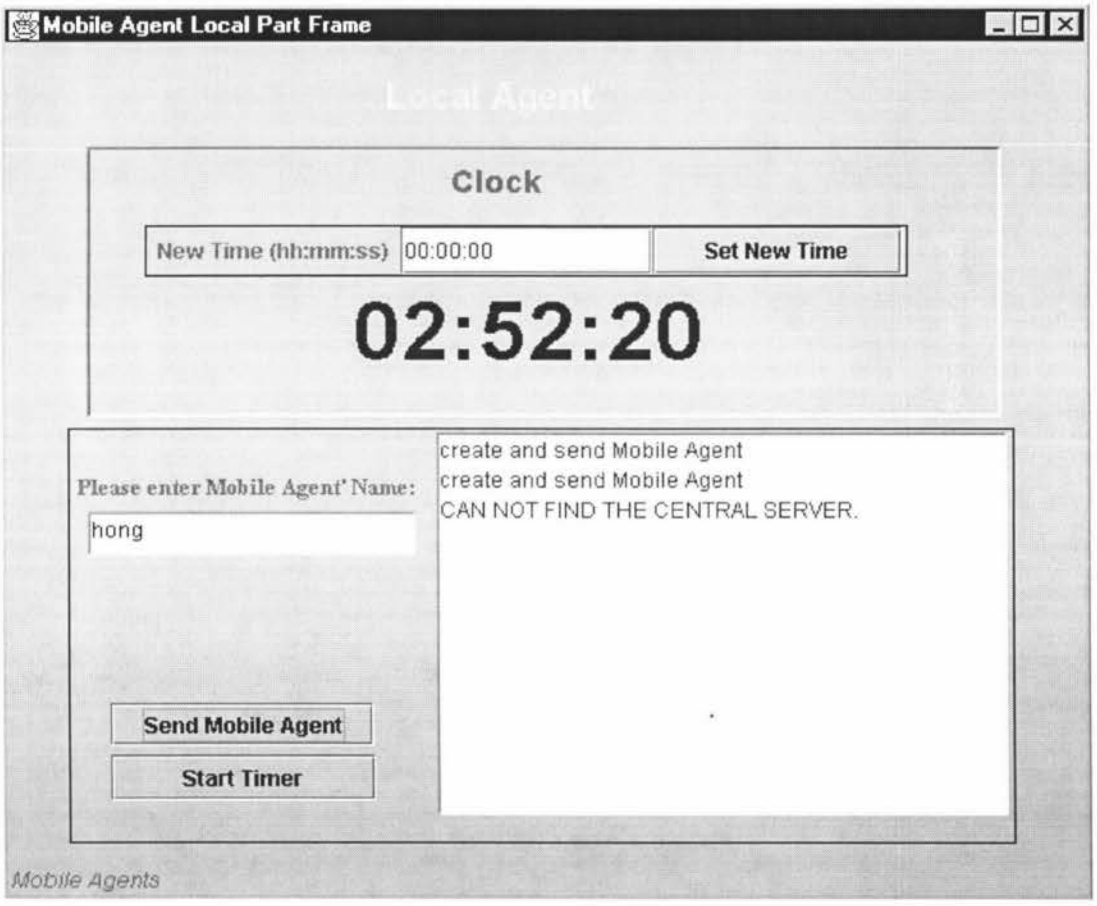
Figure 6.11: The screenshot of local side interface when it is open up.

- Checking any changes occurred in the database, and deciding to send a mobile agent by click of a button.

- Starting a timer, while repeatedly fires an event at a set interval of delay time. The event causes serial processes that include referring to database to check whether there has been any update of data, and decide whether to send a mobile agent or not. If yes, then a mobile agent will be created and sent.

- Displaying the status of mobile agent (figure 6.12).

- A clock can be used to adjust the start time and display the time elapsed.



```
create and send Mobile Agent
create and send Mobile Agent
CAN NOT FIND THE CENTRAL SERVER.
```

Figure 6.12: The screenshot after sending a mobile agent

6.5.1.2 Main Classes:

MA_LocalFrame: create and set up the whole window, which includes creation of a clock, a timer, a user entry text box, and a display area.

Clock: construct a clock with setup and adjustment functions.

6.5.2 Database Access Classes

The database access classes include the classes used for performing database-related functions, which include query, insert, and update. The main classes are described as follows.

DBControl: it is a supper class that is supposed to be extended by other database operation classes. It controls the JDBC connection and contains the query and update functions. The following code is part of this class.

```
class DBControl {
  public DBControl() {
      try {
            Class.forName("org.gjt.mm.mysql.Driver");
      }
      catch(Exception e){
            e.printStackTrace();
      }
 }
 public ResultSet executeQuery(String sql) {
    try {
     conn=
     DriverManager.getConnection("jdbc:mysql://localhost/
     MA_LOCAL?user=abrf&password=abrf");
     Statement stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
   }
   catch(SQLException ex) {
      System.err.println("db.executeQuery:"+
      ex.getMessage());
   }
   return rs;
}
public boolean executeUpdate(String sql) {
   try {
      conn=
      DriverManager.getConnection("jdbc:mysql://localhost/
      MA_LOCAL?user=abrf&password=abrf");
      Statement stmt = conn.createStatement();
             int rowCount = stmt.executeUpdate(sql);
             if(rowCount != 0)
                 bupdate = true;
   }
   catch(SQLException ex) {
      System.err.println("db.executeUpdate:"+
      ex.getMessage());
  }
  return bupdate;
  }
}
```

DBAccess: it extends the DBControl class, and is responsible for the detailed data operations that include the data checking, data retrieving, data updating, and so on.

StudentData: it is used to wrap the student data that is to be transferred. This class implements the serializable interface (e.g. java.io.Serializable), because objects passed by mobile agent are requested to be serialised.

6.5.3 Wrapper Agent Classes

Wrapper agents are more like communication bridges between local programs and mobile agents. All the conversations from local programs to mobile agents or from mobile agents to local programs have to go through wrapper agents. The following classes are the main wrapper agent classes.

LocalAW: it extends AgentWrapper that is a class from com.toshiba.beegent package, and can talk with local processes and mobile agent. Some parts of code for this class are presented below.

```
public class LocalAW extends AgentWrapper{
 public static void main(String[] argv) throws Exception{
         LocalAW aw = new LocalAW();
         aw.setName("localDB");
         aw.setPassword("honghong");
         aw.addIPStates(new AwrIPState1());
         aw.addIPStates(new AwrIPState2());
         aw.addIPStates(new AwrIPState3());
         try{
              aw.startIP();
         }catch(Exception e){
              System.out.println("cannot startIP");
         }
     }
}
```

AwrIPState1, AwrIPState2, and AwrIPState3: these classes define the behaviour of wrapper agent. The following code is from the AwrIPState1 class.

```
class AwrIPState1 extends AwrIPState{
   MA_LocalFrame mf;
   AwrIPState1(){
     setPrecond("INIT");
     mf = new MA_LocalFrame();
     mf.setActionListener(new sendActionListener());
     mf.setVisible(true);
  }

  public void action(){
     while(waitXML(0)){
           XmlAcl xa = getXML();
           String perf = xa.getTag2Value("performative");
           String content = xa.getTag2Value("content");
           String sender = xa.getTag2Value("sender");
           if(perf.equals("inform")){
                if (content.equals("finished")){
                    mf.setDisplay2("Receive feedback from
                    central server by Mobile Agent:\n");
                    mf.setDisplay2("\t--"+sender+"--\n");
                }
           }
           else if(perf.equals("failure")){
               mf.setDisplay2("CAN NOT FIND THE CENTRAL
               SERVER.\n\n");
           }
    }//while
    setPostcond("INIT");
}//action
//the following class is a inner class

class sendActionListener implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
     try{
          createBee("Bee1", mf.getName());//"carrier");
         mf.setDisplay2("create    and    send    Mobile
         Agent\n");
     }catch(Exception e){
         Debug.printLog(getMyname(),"failed   to   create
         Bee.");
         setPostcond("END");
         return;
    }
    XmlAcl xa = new XmlAcl();//set xa and send it to Bee1
    xa.setTag2Value("performative","request");
    xa.setTag2Value("sender"      ,getMyname());
    xa.setTag2Value("receiver"    ,mf.getName());
    xa.setTag2Value("args"        ,"message");
    if(!sendXML(xa)){
          Debug.printLog(getMyname(),"failed   to   send
          XML/ACL.");
          setPostcond("END");
          return;
    }
   }
 }
}//AwrIPState1
```

6.5.4 Mobile Agent Classes

Mobile agents are the core components within this system. They travel and exchange messages among client and central server. Those messages include request, inform, data, and so on. The main classes for mobile agents in this system are as follows:

Bee1: it extends the supper class Bee and implements the interface I_Bee, which come from com.toshiba.beegent package. Its function is to construct mobile agents that come with a number of behaviour, such as migrating, interacting with central server or client, and so on. The sample code for this class is presented below.

```
public class Bee1 extends Bee implements I_Bee{
      public void init() throws InstantiationException{
            addIPStates(new BeeIPStateINIT());
            addIPStates(new BeeIPStateMGRT());
            addIPStates(new BeeIPStateMGRT2());
            addIPStates(new BeeIPStateBACK());
            addIPStates(new BeeIPStateBACKCentral());
            addIPStates(new BeeIPStateWAIT());
            addIPStates(new BeeIPStateCENTRAL());
            addIPStates(new BeeIPStateLOCAL());
      }
}
```

BeeIPStateINIT: it extends the supper class BeeIPState and implements the interface I_BeeIPState, which comes from com.toshiba.beegent package. This class is compulsory and defines the initial function when a mobile agent is created. In this case, it waits to receive request or message from wrapper agents, and decides what to do and when to move to next state.

```
class   BeeIPStateINIT   extends   BeeIPState   implements
I_BeeIPState{
  BeeIPStateINIT(){
    super("INIT", "MGRT");
  }// constructor
 public void action{

    while(waitXML(0)){
      XmlAcl xa      = getXML();
      String perf    = xa.getTag2Value("performative");
      String sender = xa.getTag2Value("sender");
      String action = xa.getTag2Value("action");
      String content = xa.getTag2Value("content");

      if(perf.equals("request")){
          putBaggage("localDB",sender);
          putBaggage("update",action);
          putBaggage("message",content);
          migrateBee("centralDB");
          break;
      }
    }
  }
}
```

In addition, the following classes: *BeeIPStateMGRT, BeeIPStateMGRT2, BeeIPStateBACK, BeeIPStateBACKCentral, BeeIPStateWAIT BeeIPStateCENTRAL BeeIPStateLOCAL* define the mobile agents' behaviour based on different context or scenario.

6.5.5 Samples of XML Files

As discussed in chapter 6, the conversation among wrapper agents and mobile agents is based on the Speech-Act theory and an Agent Communication Language (ACL). The logical structure of an ACL expression is represented by XML, and therefore the ACL is called XML/ACL. The section briefly discusses the XML files used by the system.

The ACL specifies conversation flow. For example in the request-inform protocol, the inform message, the failure message or the not-understood message should be returned for the request message. XML is used to wrap and exchange ACL messages. The following files are samples of DTD file and XML document for inform message.

- DTD file:

```
<?xml encoding="US-ASCII"?>
<!ELEMENT inform (sender,receiver,content,in-reply-
to,language,protocol,conversation-id,ontology)>
<!ELEMENT sender (#PCDATA)>
<!ELEMENT receiver (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT in-reply-to (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT protocol (#PCDATA)>
<!ELEMENT conversation-id (#PCDATA)>
<!ELEMENT ontology (#PCDATA)>
```

- XML document:

```
<?xml version="1.0"?>
<!DOCTYPE inform SYSTEM "inform.dtd">
<inform>
```

```
    <sender>TBD</sender>
    <receiver>TBD</receiver>
    <content>TBD</content>
    <in-reply-to>TBD</in-reply-to>
    <language>TBD</language>
    <protocol>TBD</protocol>
    <conversation-id>TBD</conversation-id>
    <ontology>TBD</ontology>
</inform>
```

6.6 Summary

According to the Bee-gent framework technology and its compatibility with Java 2, this prototype is designed as client part (client side programs, client side wrapper agents), mobile agents, and central server parts (server side programs, server side wrapper agents) (as shown at figure 6.6). From this architecture, the prototype actually consists of two relatively separate subsystems: client side and central server side.

Based on the designed architecture, the system is implemented using Bee-gent framework technology and other technologies such as Java 2, JDBC, XML and so on. In the next chapter, we describe the early evaluation of the prototype and then conclude this research and its contributions, and also briefly mention some possible future work.

Chapter 7

Evaluation and Conclusions

This chapter concludes the research. Firstly, the contributions of this thesis are presented, and then the further work is discussed briefly.

7.1 Evaluation

In this section the initial evaluation of the prototype is described. The evaluation was given to 3 postgraduate students and 1 undergraduate student. Three of them are male and one female. This group of student are aged between 25 – 35. After working through the prototype, they were asked to complete a questionnaire, which contained 3 questions.

The goals for the evaluation were to discover how convenient the prototype is in Web-based learning environments and to confirm the improvement of student adaptivity in Web-based environments by using mobile agents technology.

Q1. Is the prototype convenient to be used in Web-based learning environments?

All evaluators thought the prototype was very convenient to be used, because it provides on-line and off-line learning supports. Students are able to access the system in various contexts, in particular, it would be helpful for students who live in remote area or do not have good quality of the Internet connection. In addition, all students are very impressed by the capability of the system that student can still automatically receive the feedback later when the Internet was connected again, after they submitted some requests and disconnected the Internet connection immediately.

Q2. Is the system useful as a Web-based leaning system?

Three of the students thought the system would be quite useful to support current Web-based learning that users are more mobile than before. Students can study from their own machine, and they also are able to study from any other machines if the Internet connection is available. One student who prefers to study from his own machine thought the system did not make any different for him. Another student said it is hard to adjust its usability from her experience.

Q3. What is the most important aspect within the prototype that is needed to be improved?

Three evaluators thought it would be a good idea to provide certain display to indicate the processes happened behind the system, such as updating the server side or the client side data. One student suggested that the system should periodically remind student, who works in local machine without connecting to the central server, to provide the Internet connection so that the local data can update the central data. Otherwise, student would not access his or her up-to-date profile from other machines through the Internet.

Based on the above comments from evaluators, certain possibilities of enhancement occurred. Also, this early evaluation would be very helpful when we conduct a formal evaluation.

7.2 Contributions

In this research, the prototype with local and central individual student models and central group model was investigated and designed, and has been implemented with mobile agents technology – Bee-gent framework, Java, and other related technologies. The goals of the system are to enhance the quality of adaptation and address some common deficiencies in Web-based learning environments

The main contributions in this research are described below.

7.2.1 Improving Adaptivity in Web-based Learning Environments

The main contribution of this thesis is to improve in Web-based learning environments is the main contribution in this thesis. It is achieved by the following two approaches:

A) By applying the architecture of the local and central individual student models and central group student models: there are totally three parts of student models in the system: local individual student model that resides in student's machine, central individual student model that resides on the central server and is a copy of its corresponding individual student model in local individual, and central group student model that sits on the central server (see figure 7.2).

   This two-step modelling mechanism largely improves capturing interactions of a given student on the Web-based learning environment, even in off-line, and enables the system to provide adaptation at different granularity.

B) By applying mobile agents technology as the communications channel between client and server instead the traditional approaches. Mobile agents technologies are applied successfully in the prototype to exploit the potentials of Web-based learning environment by addressing the common deficiencies of the Web-based learning systems.

   The Bee-gent framework is the mobile agents technology used by this prototype. Because mobile agents are inherently distributed in nature and its characteristics in nature, such as pre-fetching data/local processes of data and autonomous procedures, it is an ideal technology to implement the mechanisms of local and central individual student models and central group student model built in this research.

The mechanisms of student models implemented by mobile agent technology in this prototype address the inadequate adaptation to individual students in the Web-based learning environment by improving the capturing of interactions between student and systems.

7.2.2 Providing a Flexible Framework of Maintain Ability and Scalability

Mobile agents, in particular the Bee-gent framework, is implemented with higher-level abstraction and unified "object". This prototype provides a flexible and effective environment on future scaling and maintaining the systems. For example, to bind the existing systems into this prototype does not affect it, instead of only creating another agent wrapper to wrap the existing system and mediation agents to exchange data.

7.2.3 Free from Heterogeneous Environments

Ideally, the development of Web-based learning environment should share the existing resources through different systems and different platforms. The prototype built with mobile agent is really computer and network or web platforms independently. Thus, it can fit into the web environment naturally.

7.2.4 Facilitating Mobile Students

In the new scenario of information era, mobile users and portable-computing devices are getting popular and in terms of Web-based education the situation will become even obviously. These scenarios usually have unreliable, low-bandwidth, high-latency telephone or wireless network connections. The student model architecture and mobile agent technology essentially facilitate the Web-based learning systems to be used properly.

7.3 Future Work

The prototype was mainly built in a narrow scope to investigate the student model architecture with local and central individual student model and central group student model, and exploit the application of mobile agent technology within the Web-based learning environment. Its usability and effectiveness should be evaluated by the real world educational practices and wider scope.

Based on the evaluations, further improvements can be attempted. The iterative development circle (evaluation – modification) should be carried out until the system can be satisfying.

To further develop and improve the prototype, the following works may be needed to be carried out.

7.3.1 Modifying the Prototype to fully Work with SMAWLS

For evaluating purposes, the prototype needs to be modified to fully work with SMAWLS.

The SMAWLS is the basic context of this thesis, but its emphasising points are different from this research. This research and the prototype mainly exploited how to set up and create the structure and mechanism of local and central individual student model and central group student mode, and applied mobile agent technology. These developments all work in the background of SMAWLS, and have to modify the student model database to enable this student model architecture.

Since SMAWLS and this prototype are developed based on different hypothesises, both systems need to be modified to build a more completed system for further evaluating and developing.

7.3.2 Formal Evaluating the Prototype in Large Environments

After combining with SMALS successfully and referring to the initial evaluation, a real testing environment has to be set up for assessing how effective and accurate the student models and mobile agents are. Based on the assessment result, the system can be improved further.

7.3.3 Scaling the system with the other existing systems

To find out how easily to scale up the system and its scalable potential, we can select an existing learning system and use the Bee-gent framework to implement the merging of systems.

7.4 Conclusion

Although a number of attempts have been made to improve the adaptivity in Web-based learning environments, the Web-based systems do not exploit their full potential, in particular, facilitation of dynamic interaction. The mechanisms of the student models in this thesis and the mobile agent technologies provide an attractive alternative to implement and improve Web-based learning environments.

References

Agre, P. E. and Chapman, D. (1987), *Pengi: An Implementation of a theory of activity.* Proceedings of the 6th National Conference on Artificial Intelligence, San Mateo, CA: Morgan Kaufmann, 268-272.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge. MA: Harvard University Press.

Anderson, J. R., Boyle, C. F. and Yot, G. (1985). *The Geometry tutor.* In Soshi, A. (Ed.), Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Los Angeles: Morgan Kaufmann, 1-7.

Anderson, J. R., Conrad, F. G., and Corbett, A. T. (1989). *Skill acquisition and the LISP tutor.* Cognitive Science, 13(4), 467-505.

André, E., Rist, T. and Müller, J., (1997). *WebPersona: A Life-Like Presentation Agent for Educational Applications on the World Wide Web.* Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan

Austin, J. L. (1975). *How to do things with words.* Harvard University Press, Cambridge, Massachusetts, USA. (ISBN 0-674-41152-8).

Bogley, W. A., Dorbolo, J., Robson, R. O., and Sechrest, J. A. (1996). *New pedagogies and tools for Web based calculus.* Proc. Of WebNet96, 33-39.

Brooks, R. A. (1986). *A robust layered control system for a mobile robot.* IEEE Journal of Robotics and Automation 2 (1), 14-23.

Brown, J. S. and Burton, R. R. (1978). *An investigation of computer coaching for informal learning activities.* Technical Report, Defense Advanced Research Projects Agency, Human Resources Lab., Lowry AFB, CO.

Brown, J. S., Burton, R. R., and deKleer, J. (1982). *Pedagogical natural language, and knowledge engineering techniques in SOPHIE I, II and III.* In D. Sleeman and J. S. Brown (Eds.), Intelligent tutoring systems, New York: Academic Press, 227-282.

Browne, D. P., Totterdell, P. A. and Norman, M. A. (1990). *Adaptive user interfaces.* London: Academic Press.

Brusilovsky, P. (1994). *The construction and application of student models in intelligent tutoring systems.* Journal of Computer and System Sciences International, 32(1), 70-89.

Brusilovsky, P. (1996). *Methods and techniques of adaptive hypermedia.* User Modeling and User-adapted Interaction, 6(2-3), 87-129.

Brusilovsky, P. (1999). *Adaptive and intelligent technologies for web-based education.* in: C. Rollinger and C. Peylo (Eds.) Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching, 4, 19-25.

Brusilovsky, P. L., Ekund, J., and Schwar, E. (1997). *Adaptive navigation support in educaional hypermedia on the World Wide Web.* in: Proceedings of INTERACT97, the 6th IFIP World Conference on Human-Computer Interaction, Sydne, Australia, 14-18 July, 1997. New York: Chapman and Hall, 278-285.

Brusilovsky, P. and Pesin, L. (1994). *ISIS-Tutor: An adaptive hypertext learning environment.* Proc. Of JCKBSE'94, Japanese-CIS Symposium on knowledge-based software engineering. Pereslavl-Zalesski, May 10-13, Tokyo, 83-87.

Brusilovsky, P., Pesin, L., and Zyryanov, M. (1993). *Towards an adaptive hypermedia component for an intelligent learning environment.* In Bass L.J., Gornostaev J. and Unger C. (Eds.) Human-Computer Interaction. Lecture Notes in Computer Science #753, Springer-Verlag, Berlin, 348-358.

Brusilovsky, P. L., Schwar, E. and Weber, G. (1996). *ELM-ART: An intelligent tutoring system on World Wide Web.* In: Frasson, C., Gauthier, G. and Lesgold, A. (Eds.) Intelligent tutoring systems. Lecture Notes in Computer Science, Vol. 1086. Springer Verlag, Berlin, 261-269.

Brusilovsky, P. L., Schwar, E. and Weber, G. (1996a). *A tool for developing adaptive electronic textbooks on WWW.* Proceedings of WebNet'96 – World Conference of the Web Society, October 16 –19, 1996. San Francisco, CA, AACE, 64-69.

Brusilovsky, P. and Zyryanov, M. (1993). *Intelligent tutor, environment and manual for physical geography.* In PEG'93. Proceedings of the Seventh International PEG Conference. Edinburgh, 63-73.

Brustoloni, J. (1991). *Antonomous agents: characterization and requirements.* Technical Report CMU-CS-91-204, School of Computer Science, Carnegie Mellon University, November 1991.

Burns, H. L. and Capps, C. G. (1988). *Foundations of intelligent tutoring systems: an introduction.* In M. C. Polson and J. J. Richardson (Eds.), foundations of Intelligent Tutoring Systems, Hillsdale, NJ: Lawrence Frlbaum.

Burton, R. R. (1982). *Diagnosing bugs in a simple procedural skill.* In D. H. Sleeman, and J. S. Brown (Eds.), Intelligent tutoring systems, New York: Academic Press, 157-183.

Burton, R. R. and Brown, J. S. (1982). *An investigation of computer coaching for informal learning activities.* In Sleeman D. and Brwon J. S. (Eds.), Intelligent Tutoring Systems. New York: Academic Press, 79-98.

Carbonell, J. R. (1970). *AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction.* IEEE Transaction On Man-Machine Systems, MMS-11(4), 1970, pp 190-202.

Cardelli, L. (1995). *A language with distributed scope.* In Computing Systems, 8(1):27-59, 1995.

Chan, T.W. (1996). *Learning Companion Systems, social Learning Systems, and the Global Social Learning Club.* Jl. of Artificial Intelligence in Education. 7(2), 125-159.

Chan, T.W. and Baskin, A.B. (1988). *Studying With the Prince: The Computer as a Learning Companion.* Proceedings of Intelligent Tutoring Systems, Montreal, Canada.

Chan, T.W. and Baskin, A.B. (1990). *Learning Companion Systems.* Intelligent Tutoring Systems, 6-33, C. Frasson, G. Gauthier (Ed.), Ablex, New Jersey.

Chan, T.W., Chung, I.L., Ho, R.G., Hou, W.J., and Lin, G.L. (1992). *Distributed Learning Companion System: WEST Revisited.* Intelligent Tutoring Systems 1992: 643-650

Chang, D. and Lange, D. (1996). *Mobile agents: a new paradigm for distributed object computing on the WWW.* 1996, OOPSLA'96 Workshop.

Chang, L. J., Wang, J. C., Hsu, B. Y., and Chan, T. W. (2000). *Four applications of student modeling - my animal companions.* Proceedings of The Third Global Chinese Conference on Computers in Education, Macau, 366-70.

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

Chou, C. Y., (1999). Some future perspectives of learning companions. In proceedings of the 10th World Conference on Artificial Intelligence in Education, AI-ED 99. Le Mans, France, 80-87

Crampes, M. (1999). *User controlled adaptivity versus system controlled adaptivity in intelligent tutoring systems.* Artificial Intelligence in Education (Eds. S. P. Lajoie and M. Vivet), IOS Press, Amsterdam, 173-180 (ISBN 90 5199 452 4)

De Bra, P. M. E. (1996). *Teaching hypertext and hypermedia through the Web.* Journal of Universal computer Science 2, 12 (1996) 797-804.

De Rosis, F., De Carolis, N., and Pizzutilo, S. (1993). *User tailored hypermedia explanations.* In INTERCHI'93 Adjunct Proceedings, Amsterdam, 24-29 April, 1993 169-170.

IntelliMedia Initiative Projects (2002). *Design-A-Plant,* http://www.csc.ncsu.edu/eos/users/l/lester/www/imedia/DAP.html

Duchastel. (1992). *Integrating hypermedia into intelligent tutoring.* Interactive Multimedia Learning Environments (Ed. M. Giardina), Springer, Berlin, 67-74

Durfee, E. H. and Montogomery, T. A. (1989). *MICE: a flexible testbed for intelligent coordination experiments.* In Proceedings of the 1989 Distributed Artificial Intelligence Workshop, 25-40.

Elliot, C., Rickel, J. and Lester, J. (1999). *Lifelike Pedagogical Agents and Affective Computing: An Exploratory Synthesis.* Artificial Intelligence Today, Lecture Notes In

Artificial Intelligence (Subseries of Lecture Notes in Computer Science), Special Volume 1600, M. Wooldridge & M. Veloso (Eds.), pp. 195-212, Springer-Verlag, Berlin.

Ferber, J. (1994). *Simulating with reactive agents.* In Hillebrand, E. and Stender, J. (Eds.), Many Agent Simulation and Artificial Life, Amsterdam: IOS Press, 8-28.

FIPA. (2001). *The foundation for intelligent physical agents.*
http://www.fipa.org

FIPA. (2001a). *The foundation for intelligent physical agents.*
http://www.fipa.org/resources/index.html

Firefly. (1996). *Firefly*
http://www.firefly.com

Fischer, G. (1996). *Making learning a part of life – beyond the 'Gift Wrapping' approach to technology.* WLF Position Paper based on presentation at the NSF Symposium on Learning and Intelligent Systems, June 26, Available: http://www.cs.colorado.edu/~13d/presentations/gf-wlf

Gasser, L., Braganza, C. and Herman, N. (1987). *MACE: a flexible testbed for distributed AI research.* In Huhns, M. (ed.), Distributed Artificial Intelligence, Research Notes in Artificial Intelligence, London: Pitman, Chapter 5, 119-152.

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

Gray, R. S. (1997). *AgnetTCL: a flexible and secure mobile-agent system.* PhD thesis, Dept. of Computer Science, Dartmouth College, Hanover, N. H.

Han, B. (2001). *Student modelling and adaptivity in Web-based learning system.* Master thesis, Dept. of Information Systems, Massey University, Palmerston North, New Zealand

Hewitt, C. (1977). *Viewing control structures as patterns of passing messages.* Artificial Intelligence 8(3), 323-364.

Hoschka, P. (1996). *Computers as assistants: a new generation of support systems.* Lawrence Erlbaum Associates Publishers, Mahwah, NJ, New Jersey, 336-340 (ISBN 0-8085-3391-9).

IBM. (2001). *Aglets*
http://www.trl.ibm.co.jp/aglets/

InAMosh-handy. (2002). *InAMosh-handy: a shopping scenario for mobile phones based on intelligent agents.*
http://dai.cs.tu-berlin.de/english/forschung/projekte/InAMoSHandy/main.html

IPs. (2002). *Interaction protocol.*
http://www.fipa.org/specs/fipa00025/PC00025C.html

Johnson, W. L., Shaw, E. and Ganeshan, R. (1998). *Pedagogical Agents on the Web.* Workshop on WWW-based Tutoring, ITS '98, San Antonio, Texas.

Kass, R. and Finin, T. (1989). *The role of user models in cooperative interactive systems.* In international journal of intelligent systems, Vol. 4, 81-112.

Kawamura, T., Hasegawa, T., Ohsuga, A., and Honiden, S. (1999). *Bee-gent: bonding and encapsulation enhancement agent framework for development of distributed systems.* Proceedings of the 6th Asia-Pacific Software Engineering Conference (APSEC 99), IEEE, 260-267.

Kinshuk. (1996). *Computer-aided learning for entry-level accountancy students*. PhD Thesis, De Montfort University, England.

Kinshuk and Patel, A. (1997*). A conceptual framework for Internet based intelligent tutoring systems*. Knowledge Transfer (Volume 2) (Ed. A. Behrooz), pAce, London, 117-124 (ISBN 1-900427-015-X)

Kinshuk, Oppermann, R., Patel, A. and Kashihara, A. (1999). *Multiple representation approach in multimedia based intelligent educational systems*. Artificial Intelligent in Education (Eds. S. P. Lajoie and M. Vivet), IOS Press, Amsterdam, 259-266

Kinshuk, Patel, A. and Russell, D. (1999). HyperITS*: A Web-based architecture for evolving an configurable learning environment.* Staff and Educational Development International Journal, 3 (3), 265-280 (ISSN 0971-9008).
http://fims-www.massey.ac.nz/%7Ekinshuk/publications.html

Kotz D., R. Gray and D. Rus,(1996), *Transportable Agents Support Worldwide Applications*. In Proceedings of the 7th ACM SIGOPS European Workshop, September 1996, Connemara, Ireland.

Krulwich, B. (1995). *Bargain finder agent prototype*. Technical report. Anderson Consulting. http://bf.cstar.ac.com/bf/

Leigh, D. (1996). The Internet and distributed learning: instructional designer's medium and tool. http://mailer.fsu.edu/~dleigh/superflux/words/dleigh/netdistlearn.html.

Lemone, K. A. (1996). *Retargetable course generation – a methodology for reusability in distance education.* ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal, June 10, 1996.

Lester, J. and Stone, B. (1997). *Increasing Believability in Animated Pedagogical Agents.* Proceedings of the First International Conference on Autonomous Agents, pp. 16-21, California.

Lester, J., Voerman, J., Towns, S. and Callaway, C. (1997). *Cosmo: A Life-like Animated Pedagogical Agent with Deictic Believability.* Working Notes of the IJCAI '97 Workshop on Animated Interface Agents: Making them Intelligent, pp. 61-69, Japan.

Lester, J., Strone, B.A. and Stelling, G.D. (1999). *Lifelike Pedagogical Agents for Mixed-Initiative Problem Solving in Constructivist Learning Environments.* User Modeling and User-Adapted Interaction, 9(1-2), pp. 1-44.

Lutris Technologies, Inc. (2002). *InstantDB.*
http://www.lutris.com/products/instantDB/index.html

Lycos. (2002). *Lycos*
http://www.lycos.com/

Maes, P. (1995). *Intelligent Software.* Scientific American 273 (3), September

Metrowerks. (2001). *Metrowerks.*
http://www.metrowerks.com/products
Milne, S., Shiu, E. and Cook, J. (1996). *Development of a model of user attributes and its implementation within an adaptive tutoring system.* User modeling and user-adapted interaction, 6, 303-335.

Minar, N., Gray, M., Roup, O., Krikorian, R. and Maes, P. (2000). *Hive: distributed agents for networking things.* IEEE concurrency, 24-33.

Nakabayashi, K., Maruyama, M., Koike, Y., Fukuhara, Y. and Nakamura, Y. (1996). *An intelligent tutoring system on the WWW – Supporting interactive simulation environment with a multimedia viewer control mechanism.* World Conference of the Web Society, October 16-19, 1996. San Francisco. CA. Available: http://www.calat.com/calat_e/about/papers/webnet96/paper_camera.html

Nakabayashi, K., Maruyama, M., Koike, Y., Kato, Y., Touhei, H. and Fukuhara, Y. (1997). *Architecture of an intelligent tutoring system on the WWW* – Proceedings of the workshop on intelligent educational systems on the world wide web, 8th World. Available: http://www.calat.com/calat_e/about/papers/aied97/paper_camera.html

Nielsen, J. (1990). *Hypertext and hypermedia*. San Diego, Academic Press.

Nikov, A. and Pohl, W. (1999). *Combining user and user modeling for user-adaptivity systems*. Human Computer Interaction – Ergonomics and User Interaces (Eds. H.-J. Bullinger and J. Ziegler).

Odyssey. (2001). *Odyssey.*
http://www.genmagic.com/

OMG. (2001). *Object management group.*
http://www.omg.org

Oppermann, R. (1994). *Adaptively supported adaptability.* International Journal of Human Computer Studies, 40, 455-472.

Oppermann, R., Rashev, R. and Kinshuk. (1997). *Adaptability and adaptivity in learning systems.* Knowledge Transfer (volume II) (Ed. A. Behrooz), pAce, London, 173-179.

Oppermann, R. and Specht, M. (1999). *Adaptive mobile museum Guide for information and learning on demand.* Human Computer Interaction – Communiction, Cooperation, and Application Design (Eds. H. – J. Bullinger and J. Ziegler), Lawrence Erlbaum Associates, New Jersey, pp642-646 (ISBN 0-8058-3392-7).

Paiva, A.M. (1995). *About user and learner modeling –an overview.* Available: http://www.cbl.leeds.ac.uk/amp/MyPapers/overviewSUMChap.ps

Perpetuum. (2002). *Perpetuum mobile procura project.* http://www.sce.carleton.ca/netmanage/perpetuum.shtml

Petersons. (1999). Web site: http://www.petersons.com/

Rosenschein, J. S. and Zlotkin, G. (1994). *Rules of encounter: designing conventions for automated negotiation among computers.* Cambridge: MIT Press.

Schoch, V., Specht, M. and Weber, G. (1998*). ADI – an empirical evaluation of a tutorial agent.* Proceedings of Ed-Media/Ed-Telecom 98, AACE, VA, 1242-1247

Self, J.A. (1994). *Formal approach to student modeling*. Student Modeling: the Key to Individualized Knowledge-Based Instruction. (Ed. McCalla,G.I., and Greer, J.E.), Spring-Verleg, Berlin.

Shaw, E., Ganeshan, R., Johnson, W. L. and Millar, D. (1999). *Building a case for agent-assisted learning as a catalyst for curriculum reform in medical education.* Paper presented at the AI-ED'99 Workshop on Animated and Personified Pedagogical Agents.

Simonson, M. (1996). *Distance education: trends and redefinition.* Paper presented at the 1996 Frontiers in Education Conference, November 6-9, Salt Lake City, Utah

Sleeman, D. H. (1984). *Intelligent tutoring systems: A review*. (Report No. IRO11683). Stanford, CA: Stanford University, School of Education and Department of Computer Science. (ERIC Document Reproduction Service No. ED 257450).

Sleeman, D. H. and Brown J. S. (1982). *Intelligent tutoring systems*. London: Academic Press.

Spiders. (2002). *Spiders*
http://www.cnet.com/Resources/Info/Glossary/Terms/spider.html

Stansfield, J. C., Carr, B. and Goldstein, I. P. (1976). *Wumpus advisor I: A first implementation of a program that tutors logical and probabilistic reasoning skills*. AI Lab Memo 38. MIT, Cambridge, Massachusetts.

Steve (2002). http://www.isi.edu/isd/carte/carte-demos.htm

Stormcast. (2002). *Stormcast 5.0*
http://www.cs.uit.no/DOS/StormCast/

Sun Microsystems, Inc. (2002). *Java*.
http://java.sun.com/

Sun Microsystems, Inc. (2002a). *JDBC data access API.*
http://java.sun.com/products/jdbc/

Sun Microsystems, Inc. (2002b). *Java API for XML processing (JAXP).*
http://java.sun.com/xml/jaxp/index.html

ThingsThatThink. (2002). *Things that think.*
http://www.media.mit.edu/ttt/

Toshiba. (2001). *Bonding and encapsulation enhancement agent.*
http://www2.toshiba.co.jp/beegent/index.htm

Toshiba. (2001a). *Plangent*
http://www2.toshiba.co.jp/plangent/index.htm

Towns, S., Callaway, C., Voerman, J. and Lester, J. (1998). *Coherent Gestures, Locomotion, and Speech in Life-Like Pedagogical Agents.* Proceedings of the Fourth International Conference on Intelligent User Interfaces, pp. 13-20, San Francisco.

Vassileva, J. (1995). *Dynamic courseware generation: at the cross point of CAL, ITS and Authoring.* Proceedings of the AI-ED Workshop on Authoring Shells for Intelligent Tutoring Systems.

Vassileva, J. (1997a). *DCG + WWW: Dynamic courseware generation on the WWW.* Proceedings of AIED'97, Kobe, Japan, IOS Press, 498-505

Vassileva, J. (1997b). *Dynamic courseware generation.* Communication and Information Technologies, 5(2), 87-102.

Vassileva, J. (1998). *DCG + GTE: Dynamic courseware generation with teaching expertise.* Instructional Science, 26 (3/4), 317-332.

Vassileva, J. and Deters, R. (1998). *Dynamic courseware generation on the WWW.* British Journal of Educational Technologies, 29 (1), 5-14.

W3C. (2002). *XML specifications.*
http://www.w3.org/TR/

Webcrawlers. (2001). *Webcrawlers.*
http://www.webcrawlers.com/

Weber, G. and Specht, M.(1997). *ELM-ART.* In: A. Jameson, C. Paris and C. Tasso (Eds.), Proc of 6th International Conference on User Modeling, Chia Laguna, Sardinia, Italy, June 2-5, 1997. Springer-Verlag, Wien, 289-300.

Wenger, E. (1987). *Artificial intelligent and tutoring systems: computational and cognitive approached to the communication of knowledge.* Los Altos, Morgan Kaufmann Publishers, Inc.

White, J. E., (1996), *Telescript Technology: Mobile Agents.* Software agents, Jeffrey Bradshaw, ed., AAAI Press/MIT Press, Cambridge, Mass.

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

XML. (2002). *Extensible markup language.*
http://xml.com

XML. (2002a). *Extensible markup language.*
http://xml.com/pub/a/98/10/guide0.html

Yum, K. K. and Crawford, J. R. (1996). *On the feasibility of an interoperable tutorial machine to support the development and delivery of teaching.* Position Paper for ITS'96 Workshop on Architecture and Methods for Designing Cost-Effective and Reusable ITSs, June 10th, Montreal, Canada.