N°d'ordre : année 2006

THESE

en Cotutelle entre HUST et l'Université Paul Verlaine de Metz

Présentée en vue de l'obtention des

Doctorat de l'Université Paul Verlaine de Metz

Spécialité : Automatique

Et

Doctorat de l'Université de HUST

Spécialité : System integration & Analysis

Par

Anbo MENG

CONTRIBUTION À LA MODÉLISATION ET L'IMPLÉMENTATION D'UN SYSTÈME D'E-ÉDUCATION BASÉ SUR LES MULTI-AGENTS

"Contributions to the Modeling and Implementation of Multi-Agent Based e-Education System"

Soutenue le 15 novembre 2006 à Wuhan devant la commission d'Examen :

Rapporteurs : MM. Weiyou CAI
Tiebing JIANG
Bernard LAGET

Examinateurs : MM. Zhaohui LI
Pierre PADILLA
Jean RENAUD
Daniel ROY
Luqing YE

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy in Science

CONTRIBUTION TO THE MODELING AND IMPLEMENTATION OF MULTI-AGENT BASED E-EDUCATION SYSTEM (MAGE)

Ph.D. Candidate:	MENG Anbo
Major:	System Analysis & Integration
Supervisor:	Prof. YE Luqing
	Prof. PADILLA Pierre
	Dr. ROY Daniel

Huazhong University of Science and Technology
Wuhan, 430074, P.R.China

nov. 2006

TABLE OF CONTENT

CHAPTER 1 INTRODUCTION

This chaptered is focused on the background, motivation, objectives as well as some key issues considered in this dissertation. Eventually, the thesis organization is also given.

1.1 BACKGROUND & MOTIVATION

The world is transforming into a global village with the rapid development of Information and Communication Technology (ICT) (Nabil et al. 1997). In the past decades, the Internet and Web have rapidly developed into the main platform of software applications in a wide range of domains. Especially the advance of ICT has potentially brought about drastic change in the instructional process by enhancing the way information and knowledge are represented and delivered to learners. As one of the killer internet applications, the emerging e-Education turns out to be an important aspect for the educational area (Lennon et al. 2003) as well as for companies as part of a holistic knowledge management approach (Hasebrook 2001). Most of literatures reveal that e-Education paradigm has the potential to revolutionize the basic tenets of learning by making it individual rather than institution-based, eliminating clock-hour measures in favor of performance and outcome measures, and emphasizing customized learning solutions rather than generic, one-size-fits-all instruction. consequently, it is not surprising that e-Education is now becoming more acceptable to corporations, society, and academia which offer various kinds of online learning environments in support of the flexible, just-in-time, work-on-hand ,on-demand learning or training programs regardless of geographical, temporal, physical, social, and economical constraints. As pointed out in (ADL 2004): “E-learning paradigms and implementations have brought many advantages to technology-based distance education.” it may be a frontier for new methods of communication and new technologies giving rise to innovative teaching and learning practices that may not be possible in traditional face-to-face classroom based education. In fact, today’s e-Education may have a greater impact on the nature of higher education than any innovation since the invention of the printing press and is being regarded as a force for change in higher education, extending and improving education in general.

Although the benefits and potentials of the new generation of e-Education are obvious and exiting, unfortunately, so far the great potential of e-Education has been far from being taken full of advantage. Apparently, the current e-Education does not seem to fulfill its promise to become the most important learning paradigm, especially in the context of the

increased role of continuous and life-long learning. This is often explained by the fact that, despite their recent impressive developments, most of the currently available e-Education systems and environments are still less appealing than the traditional face-to-face teaching methods for both learners and tutors. From the learner perspective, they often complain about the lack of flexible performance tools in support of personalized and tailored learning, value-added reflection, mutual simulative knowledge sharing, on-demand expertise finding, just-in-time peer help as well as efficient and timely tutor guidance. From the tutor perspective, the main drawback of current e-Education systems is that they tend to require more effort in terms of authoring learning materials and preparing tests or examinations than their classical counterparts do. The necessity of mastering technology-intensive teaching tools and the lack of the tutor's computer literacy often make tutors reluctant to participate in online teaching activity. Based on the analysis of the aforementioned limitations existing current e-Education systems, it is obvious that on one hand, we need to provide learner with more intelligent learning environment that supports various customized learning services as needed, on the other hand, we need innovative mechanism to alleviate tutor workload in terms of facilitating the development of learning contents and test/exam by hiding as much technique details as possible. Nowadays more and more educators believe that the above-identified factors are a key to the future successful e-Education and thus naturally become the focus and motivation of this dissertation. Other motivations come from the gap between existing and ideal e-Education system that is identified as several key issues considered in this dissertation, which we can observe in next section. Nevertheless, the direct motivation of this dissertation derived from the DUO-FRANCE project co-jointly initiated by ENIM & HUST aiming at developing an intelligent, flexible, personalized and open e-Education environment in order to improve learning outcomes and teaching efficiency. To achieve such goal, we explored, and adopted a series of innovative methodologies, theories, algorithms, and technologies derived from multiple disciplines such as Multi-Agent System (MAS), Learning Object (LO), Cognitive Theory (CT), Knowledge Management (KM), Genetic Algorithm (GA), eXtensible Markup Language (XML), J2EE and so on. In particular, we, in this dissertation, concentrate on the approach of MAS as a container and supporting environment to integrating and encapsulating the above mentioned technologies and methodologies, as well as to modeling and implementing several typical e-Education applications at different levels and different contexts in terms of content authoring, individual and collective learning, expertise peer help finding, and test generation, delivery, assessment in distributed learning environment.

1.2 KEY ISSUES CONSIDERED IN THIS DISSERTATION

The following issues at high level are identified as important considerations in this dissertation, reflecting the motivation and research concerns from different perspectives through deliberate analysis of the state of the art on e-Education.

- *Educational theory*: There is an increasing recognition that successful e-Education requires not only rational and sound technology but an appropriate educational theory, which has also had an impact on using ICT as technology, is increasingly looked to as an enabler of learning. Consequently, e-Education has to consider didactical theories in terms of behaviorism, cognitivism and constructivism (Dietinger 2003) as well as psychological aspects like cognitive styles, learning strategies, etc. (Blöchl et al. 2003). The rise of constructivism as the predominant post-modern learning theory and the recognition of the importance of the social context for learning are changing curricula and teaching practice. Currently, most of the WEB-based learning systems and environments are prominently based on objectivist school of thought. Their approaches to the online learning environment usually transfer traditional classroom instruction to an online setting, recasting reading materials as web-based materials without taking into consideration the learner's different learning style, preference, cognitive ability etc. from our point of view, these are basically mere Internet-based correspondence courses which just reflect low-level learning. In this dissertation, we model MAGE on constructivism basis, taking into consideration the different learner profiles in individual and collective learning contexts, respectively.
- *Interoperability*: Due to the necessity of high-quality content, interoperability issues like transferability and reusability of content as well as the usage of learning object repositories have to be considered (Qu et al. 2002). In particular, standards for describing and exchanging e-Education content should be supported by an e-Education environment. In fact, the standardization process in the field of e-Education is still in progress and only a few specifications are standardized. Hence, many aspects of e-Education content can only be described with proprietary specifications, which, additionally, do not fully support learner-centered adaptivity as shown in (Mödritscher et al. 2004). In such case, most e-Education system cannot interoperate. In MAGE, our practice is, on one hand, to assure interoperation as high as possible through incorporating mainstreaming e-Education specification (e.g. LOM specification), on the other hand, to enhance the adaptivity by extending existing specification.
- *Adaptivity*: the lack of adaptivity is considered one important drawback in most

e-Education systems. A paradigm shift from consumption of static learning contents to well tailored; highly personalized learning sessions is needed (Garcia et al. 2004). Adaptability has been considered as an important aspect for characterizing and comparing different systems' behaviors. Therefore, an e-Education environment has to provide methods to adapt to the learner as well as to the teacher. In MAGE, we distinguish four categories of adaptivity: Adaptive Interface (AI), Adaptive Content Delivery (ACD), Adaptive LO Discovery and Assembly (ALODA) and Adaptive Collaboration Learning (ACL). AI refers to adaptations that take place at the system's interface and are intended to facilitate or support the user's interaction with the system, without, however, modifying in any way the learning "content" itself, ACD refers to adaptations that are intended to tailor a course content to the individual learner, ALODA refers to the application of adaptive techniques in the discovery and assembly of learning object from potentially distributed LO repositories and ACL is intended to capture adaptive support in learning processes that involve communication between multiple users (and, therefore, social interaction), and, potentially, collaboration towards common objectives. This is an important dimension to be considered as we are moving away from "isolationist" approaches to group learning, which are at odds with what modern learning theory increasingly emphasizes the importance of collaboration, cooperative learning, and communities of learners, social negotiation, and apprenticeship in learning (Wiley, 2003). Adaptive techniques can be used in this direction to facilitate the communication / collaboration process, ensure a good match between peer learners, etc.

- *Reusability*: the conversion from stand-alone Computer Based Instruction (CBI) to Web-based learning content was direct adaptations of existing products from CD-ROM to online delivery. The Web was used initially as little more than a replacement distribution medium. Content was still static and monolithic (i.e., designed to address one specific set of learning objectives as a contiguous whole, and not easily broken into components with significant reuse potential). The situation obviously prevents the reusability and share of content objects in other application contexts. Furthermore, failing to separate content and the logic controlling the display and presentation further aggravate the situation. Therefore, reusable, sharable learning objects and intelligent authoring methodologies, and adaptive learning strategies should be taken into consideration. In MAGE, we developed a flexible learning object model and designed a powerful framework to assure several possibilities of reusability applied to different contexts.

- *Social Intelligence*: the application of Artificial Intelligent to education is not a new thing, including several paradigms, such as Intelligent Computer Aided Instruction (ICAI), Micro-world, Intelligent Tutoring Systems (ITS), and Intelligent Learning Environment (ILE). Nevertheless, their intelligence is often restricted in single-user, single-computer systems because of the lack of dealing with large-scale social communication ability among different intelligent components in distributed e-Education context. From my personal point of view, the higher level of intelligence should be obtained during the process of cooperation, collaboration, negotiation, even competition among distributed agents on behalf of different particular services, resources, and human agents. The intention of achieving such intelligence forces us to have to consider new technology and architecture. Fortunately, the recent emerging MAS technology seems to fulfill our need and ambition for the purposes of the provision of efficient mechanism and approach to promoting the intellectual exchange, collective learning, collaborative endeavors, and socialization. The next section will further elaborate on the rational of MAS applied in this dissertation.
- *Scalability, Robustness and Maintainability*: these issues are also considered important factors that contribute to the success of an e-Education system. From the literature, we can see that most of traditional e-Education systems are characterized by data-centered, content-oriented, computing-concentrated, function-interdependent, and logic-coupled. This obviously impairs the possibility of scalability, reusability, robustness, and maintenance reliability from technology and system development perspective. How to address these issues is still an open question. However, from the next section, we can see that MAS has the most promise to deal with the aforementioned issues.

1.3 OBJECTIVES

The overall goal of this thesis attempts to build a flexible, scalable, adaptive and intelligent e-Education system aiming at providing an efficient mechanism to personalize the learner's learning process, diversify the learning paradigms, help find tailored expertise and peers and facilitate the development of learning materials and tests through the integration of some advanced technologies such as MAS, MA, LO, GA, XML and KM as well as well-established teaching and learning theory.

More specifically, we pursue the following create objectives:

- To develop a multi-agent based learning management system in order to provide an efficient mechanism to personalize the learner's learning process, evaluate learner's performance, diversify the learning scenarios, offer adaptive course sequencing and

navigation etc.

- To develop a multi-agent enabled e-Education object & course authoring system in order to provide an efficient and powerful mechanism to facilitate the developing process of courses and learning objects including searching the appropriate courses and EEOs, subscription to LCMA as well as negotiation between course and EEO developers.
- To develop a distance assessment system. In particular, to implement the automatic test generation system (GAMASTG) in order to facilitate for the tutor or system agent to automatically compose tests according to the desired parameters (adaptive or specified by tutor) associated with a test.
- To develop a peer help system in e-Education context in order to provide seamless access for learners to a variety of distributed help resource including human resources, like peer help and tutor advice, as well as electronic resources, like threads in discussion forums, FAQ entries, and web-resources.
- To develop a KM based collaborative learning environment to facilitate collective thinking, collaborative endeavor, knowledge sharing.

1.4 THESIS ORGANIZATION

The dissertation is composed of eight chapters described as follows:

Chapter 1 introduces the background, motivation, objectives from the general perspective of the whole thesis.

Chapter 2 focuses on examining the state of the art on e-Education and related pedagogic theories, methodologies and technologies (MAS in particular) in order to identify the key components that can be served as the foundation of this dissertation from pedagogic and technology perspectives.

Chapter 3 analyzes the e-Education reference model based on LTSA of IEEE LTSC and presents a recommended architecture of multi-agent e-Education system, which consists of three types of agents: learner-side agents, server-side agents and learning content-side agents. With this architecture, it is convenient to facilitate the construction of various flexible learning scenarios and the development of learning contents.

Chapter 4 puts forward an architecture of multi-agent enabled course authoring model based on e-Education object (MEEOCAS), under support of this subsystem, the course designers may conveniently develop their courses through assembling the ready-made EEOs instead of creating them from the scratch. As far as learners are concerned, in MEEOCAS-enabled environment, they may choose among several available learning patterns (e.g.

course-oriented learning, EEO-oriented learning, self-paced learning, teacher-centered learning or collective learning pattern etc.).

Chapter 5 proposes a MAS based integrated framework in support of adaptive and active learning. The aim is to address the adaptive learning issues such as how to dynamically generate learning path and present tailored learning objects catering for a learner's knowledge state and learning preference, how to find appropriate help resources (e.g. peer learners, learning materials, or other applications) for a learner when s/he encounters difficulty in learning certain domain concept or topic, how to build collective learning environment in support of constructivist learning, e.g. a learner can take the initiative to construct a desired learning group for his or her particular purpose

Chapter 6 puts forward a new approach to building the e-assessment subsystem applying MAS.. One of the key innovative points is that the core functionality is mostly carried out by relative mobile agents, as compared with the traditional client-server computing paradigm, the advantages is obvious such as: communication latency and bandwidth, asynchronous execution, protocol encapsulation and parallel execution.

Chapter 7 aims at implementing part of models proposed in previous chapters in order to verify and validate their feasibility and efficiency. The simulation results show the feasibility and efficiency of the models proposed in this dissertation.

Chapter 8 draws the conclusions and conducts some perspectives.

CHAPTER 2 STATE OF THE ART: E-EDUCATION, PEDAGOGIC THEORIES & MAS

This chapter focuses on examining the state of the art on e-Education and related pedagogic theories, methodologies and technologies(MAS in particular) in order to identify the key components that can be served as the foundation of this dissertation from pedagogic and technology perspectives

2.1 INTRODUCTION

E-Education paradigms and implementations have brought many advantages to technology-based e-Education as pointed out in (ADL 2004). Since e-Education on its own is accompanied by many disadvantages, it is recommended that technology-based learning should be combined with conventional courses (Garcia et al. 2004). Besides, e-Education has to consider pedagogic theories in terms of behaviorism, cognitivism and constructivism (Dietinger 2003) as well as psychological aspects like cognitive styles, learning strategies, etc. (Blöchl et al. 2003). Furthermore, it is believed that adaptation, personalization and socialization will greatly improve learning quality and enhance learning efficiency. Therefore, e-Education is a big picture, which involves several cross disciplines such as computers, psychology, pedagogy and so on. To go deep into this thesis, it is necessary to introduce and examine the relative topics involving e-Education and related pedagogic theories, methodologies and technologies, which will applied to the relevant architectures, models or algorithms in the following chapters.

2.2 LITERATURE ON E-EDUCATION

2.2.1 DEFINITION

There are many terms for e-Education. Frequently used vocabularies include: e-Education, e-learning, distributed learning, distance education, distance learning, online learning, virtual education, Internet-based education, Web-based education, and education via computer mediated communication (CMC) etc. Although they can be considered as interchangeable and equivalent at most cases, sometimes, they make us confused. From the literature observation, the commercial community tends to use e-learning while the

academia tends to use distance education. The possible reason is that e-learning providers often focus on course content, while distance education institutions emphasize on the whole range of educational services. In this dissertation, we tends to use e-Education throughout this dissertation except that we have to respect the original references, the reason is based on the following considerations:

- The research scope is constrained within online education via Internet, the concept distance appears not to be anymore important;
- E-Education not only includes e-learning but e-teaching. More specifically, an ideal e-Education system needs to provide channels to facilitate both learning and teaching.

Since e-Education is our focus in this dissertation, it is necessary to examine its definitions from mainstream literature. Actually, there exist no acknowledged standard definition; the following lists of definitions are used to interpret the concept e-Education from different perspectives (note that we referred to the original terms for e-Education put forward by the authors).

According to Desmond Keegan's (1988), distance education is characterized by:

- the separation of teachers and learners which distinguishes it from face-to-face education;
- the influence of an educational organization which distinguishes it from self-study and private tutoring;
- the use of a computer network to present or distribute some educational content;
- the provision of two-way communication via a computer network.

DerekStockley (2004) defines e-learning as "the delivery of a learning, training or education program by electronic means. E-learning involves the use of a computer or electronic device in some way to provide training, educational or learning material."

Porter (1997) shared that distance learning was education or training offered to learners who are in a different location than the source or provider of instruction. Porter went on to say that the technologies used in distance learning, the structure of a course or program, and the degree of supervision for a distance learning course can be varied to meet a particular's group's needs or interests.

Taking into consideration these definitions it can be summarized that all of them comprise the combination of the following basic components: learning activities and teaching via different electronic media. That is why it becomes very important to be aware of some instructional and technological aspects during the development process of any e-Education system. In this context, the main goal of this dissertation is to investigate the

didactical aspects of e-Education as it concerns the development of e-Education environments as well as it refers to the use of these environments and all proper technological tools. The strong expectations that high-level technological tools will increase the quality of any e-Education course often follow to an underestimation of the educational objectives being set. According to our experience, the crucial question is not what technological tools are to be used during the development process of an e-Education system? The core problem is how to design and plan an e-Education course that ensures the achievement of the educational objectives?

2.2.2 EVOLUTION OF E-EDUCATION

The history of E-Education is also a history of communication technologies. As new communication technologies have been developed, they have joined the repertoire of the distance educator. Generally, each of the emerging educational delivery technologies has been incorporated into different e-Education systems, resulting in a total multimedia-based e-Educational system comprised of various generations of distance technology and media. In other words, the different technologies and media have complemented and supported each other, rather than replaced existing ones. Historically, e-Education operations have evolved through the following four generations: first, the Correspondence Model based on print technology; second, the Multi-media Model based on print, audio and video technologies; third, the Tele-learning Model, based on applications of telecommunications technologies to provide opportunities for synchronous communication; and fourth, the Flexible Learning Model based on online delivery via the internet. Although the latter approach is still gaining momentum, as we stride into the new millennium, there is already emerging the fifth generation of e-Education based on the further exploitation of new technologies. The fifth generation has the potential to decrease significantly the cost of online tuition and thereby increase significantly access to education and training opportunities on a global scale. Through the application of automated response systems, which entail the use of software that can scan the text of an incoming electronic message and respond intelligently- without human intervention. In fact, the fifth generation of e-Education is a derivation of the fourth generation, which aims to capitalize on the features of the Internet and the Web. To place the fifth generation Intelligent Flexible Learning Model into a meaningful conceptual framework, it is first worth reviewing briefly certain features of the previous four generations of e-Education. Some of the characteristics of the various models of e-Education that are relevant to the quality of teaching and learning are summarized in Table 2-1, along with an indicator of institutional variable costs (Taylor et al. 1993). In traditional e-Education delivery, the distribution of packages of self-instructional

materials (audiotapes, videotapes, etc) is a variable cost, which varies in direct proportion to the number of learners enrolled. Internet-based delivery, however, changes significantly the institutional costs associated with learners gaining access to learning experiences. For example, a key consideration for the fifth generation is the use of automated response systems to reduce the variable cost of computer-mediated communication (CMC), which in the fourth generation is quite resource- intensive.

Table 2-1: Models of E-Education - A Conceptual Framework

Models of E-Education and Associated Delivery Technologies	**Characteristics of Delivery Technologies**				**Institutional Variable Costs Approaching Zero**
	Flexibility			**Advanced Interactive Delivery**	
	Time	**Place**	**Pace**		
First Generation - Correspondence Model					
• Print	Yes	Yes	Yes	No	No
Second Generation - Multi-media Model					
• Print	Yes	Yes	Yes	No	No
• Audiotape	Yes	Yes	Yes	No	No
• Videotape	Yes	Yes	Yes	No	No
• Computer-based learning	Yes	Yes	Yes	Yes	No
• Interactive video (disk and tape)	Yes	Yes	Yes	Yes	No
Third Generation - Telelearning Model					
• Audio teleconferencing	No	No	No	Yes	No
• Videoconferencing	No	No	No	Yes	No
• Audio graphic Communication	No	No	No	Yes	No
• Broadcast TV/Radio and Audio teleconferencing	No	No	No	Yes	No
Fourth Generation – Flexible Learning Model					
• Interactive multimedia (IMM)	Yes	Yes	Yes	Yes	Yes
• Internet-based access to WWW resources	Yes	Yes	Yes	Yes	Yes
• Computer mediated communication	Yes	Yes	Yes	Yes	**No**
Fifth Generation – Intelligent Flexible Learning Model					
• Interactive multimedia (IMM)	Yes	Yes	Yes	Yes	Yes
• Internet-based access to WWW resources	Yes	Yes	Yes	Yes	Yes
• Computer mediated communication, using automated response systems.	Yes	Yes	Yes	Yes	**Yes**

2.2.3 ADVANTAGES & DISADVANTAGES

The chief advantages of E-Education programs are that learners can learn at their convenience thus accommodating work and personal life and that it can be accessed by those who do not live near or who cannot attend traditional training centers and universities. This is tempered, however, by some of the costs and personal motivation needed to complete programs. For faculty, teaching at a distance requires a large shift in what is normally performed from being just a teacher to being a combination of facilitator, coach, and mentor. Last-minute preparation in isolation cannot happen since one needs to work with a team of professionals. Typically, teaching at a distance requires more time and faculty workload (Billings 1997). Cravener (1999) found in her review of 185 articles that having learners at a distance increased faculty time demands when compared with the classroom courses. For example, in a graduate epidemiology course, administrators complained of the number of e-mails and feedback needed to make learners feel less isolated and supported (Rose et al. 2000).

In e-Education, the learner is usually isolated. The motivational factors arising from the contact and competition with other learners are absent. The learner also lacks the immediate support of a teacher who is present and able to motivate and, if necessary, give attention to actual needs and difficulties that crop up during study. Distant learners and their teachers often have little in common in terms of background and day-to-day experiences and therefore, it takes longer for learner-teacher rapport to develop. Without face-to-face contact distant learners may feel ill at ease with their teacher as an “individual” and uncomfortable with their learning situation. In e-Education settings, technology is typically the conduit through which information and communication flow. Until the teacher and learners become comfortable with the technical delivery system, communication will be greatly inhibited.

Other advantages and disadvantages have been identified from numerous studies of e-Education in diverse fields (see Table 2-2).

Table 2-2: Advantage and Disadvantage of E-Education

Advantages		**Disadvantages**	
Convenience	Ability to participate in learning activities at the learners' convenience, at work or at home.	**Team approach**	Need a team of technical and pedagogical experts to develop course and content.
Accessible	Learners in rural areas can learn without incurring lengthy transportation costs. Women in traditional societies can learn at home.	**Faculty workload**	Need new teaching methods to offer same content; Typing comments or corrections makes grading slower. No chance for improvisation. Learners need more support than in traditional courses. Volume of communications increase.
Cost savings	Can be realized by decreasing learning time for learners and saving travel time and expenses to send faculty or learners to remote sites. School buildings are not required.	**Cultural differences**	Wider attendance means difficulty in addressing curriculum to different segments of learners.
Just-in-time	Access to more material for wider audience. Access to training means workers can immediately applies knowledge and skills to the job.	**New technology**	Must teach learners e-mail, computer skills, and networking. User guides have to be developed.
Computer proficiency	Those that use computers in e-Education programs often gain high computer proficiency.	**Lack of visual and nonverbal cues**	Written communications are more structured and formal than verbal. Isolation and alienation is an issue.
Instructional quality	A team of professionals often crafts e-Education programs. Many programs go through extensive quality control.	**Higher room for error**	The increased number of people on the development team needs heightened coordination.
Teamwork	Distance learners tend to support each other more and develop strong networks.	**Over reliance on technology**	Often depends on control of institution. Service failures, power losses, malfunctioning of computers or audiocassette players.
Inexpensive	Costs saving increase over time as up-front development costs are absorbed and more learners enroll.	**Expense of technology**	Programs that rely on satellites and/or computers cost a great deal.
		High degree of motivation	Dropout rates are very high due to the high degree of self-directedness required to finish.

2.2.4 TREND OF E-EDUCATION

Literatures show that the future of e-Education imposes: the development of adaptive e-Education environments have to take into account the pedagogical and psychological theories of learning, enhancement of the standards and specifications for e-learning as it concerns some didactical issues, and investigations concerning the elaboration of the methodology for developing learning content and conducting e-Education with the technological tools being worked out. The emerging trends in e-Education identified by Ohio(2004) are shown as table 2-3.

Table 2-3: Current and Emerging Trends in e-Education from (Ohio 2004)

Traditional	Emerging
Learners are considered receptacles	Learners share responsibility for their learning; self-directed learning
Faculty own content	Faculty act as director of learning team
Institutions act independently	Institutions act through partnerships for learning
Learning is confined by semesters, quarters, etc.	Learning happens in varying timeframes, open entry, open exit, etc.
Degrees based on credit hours	Degrees based on competency exams
Learning in the classroom	Learning takes place in multiple areas
Courses packaged one way	Courses packaged multiple ways: i.e. modules
Faculty-centered	Learner-centered
Relatively homogeneous learner population	Varied learner population: diversity in culture, age, etc.
Emphasis on college experience	Emphasis on learning
Faculty to learner model	Customer service model: demands for immediate information
Less competition for higher education	More competition: corporate and for-profit institutions
Technology played a small role in delivery of coursework and is less accessible to the masses	Technology plays a major role in course delivery and is more readily accessible to the masses
Technology evolved but not at rapid pace	Technology innovations change daily and have high cost

2.3 COGNITIVE THEORY IN EDUCATION

Cognition can be defined as “the act or process of knowing”; or more specifically, “an intellectual process which transforms perception or ideas into knowledge” (Web 1913). Cognition was crucial in the development of psychology as a scientific discipline. There are a variety of perspectives and emphases within cognitive psychology that are currently

impacting educators' concern about how to improve the teaching and learning process. For instance, the Cognitive Processing focuses on the study of the structure and function of mental processing within specific contexts or environments (Rau 1995). The Taxonomy of the Cognitive Domain (Bloom 1956) is widely used to classify the variety of educational objectives related to what and how people know. The Cognitive/Learning Style Theory may be used to predict the most effective instructional strategies or methods for a given individual learning task. The Constructivist Learning Theory may be used as a guide to build modern learner-centered constructivist learning environment.

Consequently, it is necessary to discuss the aforementioned cognitive theories and their implications to e-Education as there is an increasing recognition that successful e-Education requires not only rational technologies but sound cognitive learning theories which provide both theoretical foundation and practical opportunity for this thesis to move towards building MAS based constructivist learning environments in support of developing learner's individual and social ability from different perspectives.

2.3.1 COGNITIVE PROCESS

Cognitive learning theory explains how mental processes transform information received by the sensory organs into knowledge and skills in human memory. In general, the human being has a three-level memory structure: sensory memory, working memory, and long-term memory. Cognitive learning theory explains the human being's cognitive process of learning with several key ideas (Clark and Mayer 2002): (1) human memory has two channels, visual and auditory, and a limited capacity for processing information, (2) learning occurs by active processing in the memory system, and (3) new knowledge and skills must be retrieved from long-term memory for transfer to the job. As Clark pointed out, when a learner interacts with a tutoring system, a lesson's visual and auditory information enters the learner's eyes and ears. This information is briefly stored in the sensory memory. It then enters working memory and is finally stored in long-term memory. The working memory is the center of cognition where all active thinking takes place but its capacity is very limited. Learning requires new knowledge and skills stored in working memory to be integrated with existing knowledge in long-term memory. The integration process is an encoding procedure, which requires active processing of the information in working memory. The active processing in working memory is called rehearsal. Later, the learner must be able to retrieve those skills from long term memory back into working memory for solving problems. It is obvious that the human being's learning is active information processing procedure. Thus, it is very natural that more learner control strategies should be used in tutoring system in order to optimize the learning outcome and

increase the involvement and autonomy of learners in learning.

Because the learner's cognitive system has limited capacity competed by many sources of information, the learner must select those that best match his/her goals. Thus the tutoring systems should provide learners with information selection opportunities as well as selection guidance. The integration work requires that the limited working memory capacity be not overloaded, so the systems should enable learners to reduce their cognitive loads. To support the knowledge retrieval process, the systems must provide a job context during learning that will contain the needed retrieval links for learners' use. In addition, just as a computer has an operating system to manage data transfer, the human processor has metacognition to manage the learning processes. Metacognition refers to the mental management processes that monitor those information processing. A learner with effective metacognitive skills is able to set learning goals, decide in effective ways to reach those goals, oversee the progress, and make necessary adjustments. A good tutoring system should provide some of the management processes to enable high metacognitive learners to take advantage of them and also help learners with low metacognitive skills for successful learning.

In (Clark and Mayer 2002), there are multiple options identified for the learner control strategy:

- Content and content sequencing. Learners can select instructional goals and contents, and control the display order of the courses, topics, and screens within a lesson.
- Pacing. Learners can control the time spent on each lesson. Except for short video or audio sequences, learners can allocate different time periods to lessons for mastery according to their own needs.
- Access to learning support. Learners can control when to access which instructional components of lessons such as helps, examples, practice items, and coaches.
- Instructional interaction method. Learners can decide whether to learn individually or collaboratively, select peers as learning partners, and negotiate with systems on instructional goals and course contents.

The last option of instructional interaction method is generated with the emergence of collaborative learning and constructive learning. All these control options represent the different tutoring and learning needs. In tutoring systems, they can be controlled by learners to different degrees through corresponding software mechanisms. Note that although learners can control tutoring and learning, the learner control strategies are eventually implemented by software tutoring systems.

2.3.2 TAXONOMY OF COGNITIVE DOMAIN

Beginning in 1984, a group of educators undertook the task of classifying educational goal and objectives. The intent was to develop a classification system for three domains: the cognitive, the affective, and psychomotor. Work on the cognitive domain was completed in 1956 by Bloom and commonly referred to as Bloom's Taxonomy of the Cognitive Domain that is now most widely used ways of categorizing level of abstraction of questions that commonly occur in educational settings. The major idea of the taxonomy is that educator may organize knowledge following a complexity hierarchy from lesser to more complex concepts. The taxonomy is demonstrated in table 2-4 with sample verbs for each level.

Table 2-4: Taxonomy of Cognitive Objectives/Educational Objective in Cognitive

LEVEL	DEFINITION	SAMPLE VERBS
KNOWLEDGE	Learner recalls or recognizes information, ideas, and principles in the approximate form in which they were learned.	Write List Label Name State Define
COMPREHENSION	Learner translates, comprehends, or interprets information based on prior learning.	Explain Summarize Paraphrase Describe Illustrate
APPLICATION	Learner selects, transfers, and uses data and principles to complete a problem or task with a minimum of direction.	Use Compute Solve Demonstrate Apply Construct
ANALYSIS	Learner distinguishes, classifies, and relates the assumptions, hypotheses, evidence, or structure of a statement or question.	Analyze Categorize Compare Contrast Separate
SYNTHESIS	Learner originates, integrates, and combines ideas into a product, plan or proposal that is new to him or her.	Create Design Hypothesize Invent Develop
EVALUATION	Learner appraises, assesses, or analyzes on a basis of specific standards and criteria.	Judge Recommend Critique Justify

Bloom's Taxonomy is a convenient way to describe the degree to which we want our learners to understand and use concepts, to demonstrate particular skills, and to have their values, attitudes, and interests affected. Therefore, it permits teachers to introduce a pedagogical decision rule in their teaching material, based on learner abilities, to determine the learner's state of knowledge and adapt learning materials. One of its applications in this

thesis can be found in chapter 6

2.3.3 LEARNING STYLE

The ways in which an individual characteristically acquires, retains, and retrieves information are collectively termed the individual's learning style (Felder 1995). As Felder pointed out that people never learn in the same way, but always in many ways—"by seeing and hearing; reflecting and acting; reasoning logically and intuitively; memorizing and visualizing". The notion of learning style has been introduced by educationalists as a "description of the attitudes and behaviors that determine our preferred way of learning" (Honey 2001). In the past decades, researchers from different disciplines have intended to define and classify learning styles aiming at facilitating the individualized teaching and learning process. Literatures show that learning style theory has been important pedagogic foundation of intelligent tutoring system. However, there exist many different ways of categorizing learning styles. For example, Pask's (1975) Serialist Versus Holist indicates serialists prefer to learn in a sequential fashion while holists prefer to learning in a hierarchal manner. Kolb's Learning Style Inventory describes learning styles on a continuum running from concrete experience, through reflective observation, to abstract conceptualization, and finally active experimentation (Kolb and Fry 1975; Kolb 1984). Gardener's Multiple Intelligences (Gardner 1993) divides learning styles as dealing with words (Vernal/Linguistic), questions (Logical/Mathematical), pictures (Visual/Spatial), music (Music/Rhythmic), moving (Body/Kinesthetic), socializing (Interpersonal), and alone (Intrapersonal). The other popular learning style theories include the Felder-Silverman Learning Style Theory (Felder and Silverman 1988; Felder 1993), Litzinger and Osif Theory of Learning Styles (Litzinger and Osif 1993), Myers-Briggs Type Indicator (MBTI) (Briggs and Myers 1977; Myers and McCaulley 1985).

Especially in the recent years, some instructional practitioners have started to adopt learning style theories and explore the delivery of learning contents adapted to the learner's learning style preference in Web-based systems. For instance: the system developed by Carver et al. (1996, 1999) relates learning styles based on Felder- Silverman Learning Style Theory to course components, this system presents a list of links by order to each learner according to their learning style, leaving the individual learner to explore the course by clicking these links. The Web-based system created by Paredes and Rodriguez (2002) uses Felder-Silverman Learning Style Theory and Index of Learning Styles to assess learner's learning styles. Then the assessment result is used to automatically adapt Web-based educational systems' content sequencing for learner. However, the system only supports two dimensions of four dimensions in the Felder-Silverman Learning Style Theory. The

Arthur system (Gilbert and Han 2002) assumes four leaning styles: auditory, visual, tactile or a combination of these styles, and there is respective course material for each style. When learner first time enters the system, the course content is delivered to learner randomly. Then the system monitors learner's learning process and base on learner's evaluation to update learner's learning styles (auditory, visual, tactile or a combination of them). According to learner's latest learning styles, the system provides the suitable course content. The learning styles supported by the system are not based on any educational learning style theory, so its learning styles are more or less like preference. The other reportable applications and contribution in such direction can be observed in (Specht and Oppermann 1998; and Hong and Kinshuk 2004). Different systems have various ways to collect learner's learning styles, such as interview, questionnaire, and monitor learner's behavior. However, an important point that has to be kept in mind is to how to get a useful learner's learning style actually is a psychological test process that specially designed, and not by a simple interview (Brusilovsky, 2001).

Although most of the systems mentioned above have incorporated learning style theory into the learning material design, the main existing problem consists in their pedagogies and technologies are not suited to dynamic adjustment to learners' learning styles. The knowledge is still delivered in a static way and the learning materials are more or less preset for a certain type of learning style preference, and will not be changed or adjusted according to a change of learning style of the user over time. The pedagogy that integrates learning object and learning style, which we have applied in this thesis, is able to dynamically organize and deliver learning materials to satisfy individual learning preference requirements. The key support for dynamic adaptivity is assured by the incorporation of multi-agent technology into our system. More details on this application can be referred to chapter 4. The other situation of application of learning style theory to dynamically grouping learners for collaborative learning in the context of this thesis can be found in chapter 5

From the existing learning style theory, the Felder-Silverman Learning Style Model is chosen to be implemented in this thesis. The reasons to choose this learning style theory are:

- Its Index of Learning Style (ILS) questionnaire (Felder and Soloman, 2003) provides a convenient and practical approach to establish the dominant learning style of each learner.
- The results of ILS can be linked easily to adaptive environments (Paredes and Rodriguez, 2002).

- It is most appropriate and feasible to be implemented for WEB-based courseware (Carver and Howard, 1999).

Consequently, in the following sections, we describe The FSLSM (Felder-Silverman Learning Style Model). This model was developed on the basis of theories of psychological types (Jun 1921) and theory of experiential learning (Kolb 1984) and are now used widely in the science education and computer assisted learning system. FSLSM categorizes an individual's preferred learning style by a sliding scale of five dimensions: sensing-intuitive, visual-verbal, inductive-deductive, active-reflective and sequential-global. Currently, the inductive-deductive dimension has been deleted from the previous theory, because of pedagogical reasons. As shown in table 1, this theory situates a learner's learning style preference within a four-dimensional space (see table 2-5), with the following four independent descriptors:

Table 2-5: Felder's learning dimensions (Carver, et al., 1999)

Definition	Dimension		Definitions
Do it	Active	Reflective	Think about it
Learn facts	Sensing	Intuitive	Learning concepts
Require Pictures	Visual	Verbal	Require reading or lecture
Step by step	Sequential	Global	Big picture

The FSLSM learning style dimensions were partially defined in terms of the answers to the following four questions:

- What type of information does the learner preferentially perceive: sensory—sights, sounds, physical sensations, or intuitive— memories, ideas, insights?
- Through which modality is sensory information most effectively perceived: visual— pictures, diagrams, graphs, demonstrations, or verbal—written and spoken words and formulas?
- How does the learner prefer to process information: actively—through engagement in physical activity or discussion, or reflectively— through introspection?
- How does the learner progress toward understanding: sequentially—in a logical progression of small incremental steps, or globally—in large jumps, holistically?

In order to detect the learners learning style, a questionnaire –Index of Learning Styles (ILS), is developed by Felder and Soloman (2003). The aim of the ILS questionnaire is to help learners to identify their own dominant learning styles. Currently, the questionnaire consists of 44 questions that each comes with two possible answers, a or b. All question are classified correspond to four pairs in the Felder and Silverman Learning Style theory. The results of questionnaire are explained as follows:

- If your score on a scale is 1-3, you have a mild preference for one or the other dimension but you are essentially well balanced. (For example, a 3a in the ACT/REF category indicates a mild preference for active learning.).
- If your score on a scale is 5-7, you have a moderate preference for one dimension of the scale and will learn more easily in a teaching environment, which favors that dimension.
- If your score on a scale is 9-11, you have a strong preference for one dimension of the scale. You may have real difficulty learning in an environment, which does not support that preference.

2.3.4 CONSTRUCTIVISM

Theoretically, the commonly used instructional design models fall into objectivism-based instructional design models (Gagne et al. 1992), and constructivism-based instructional design models (Spiro 1992; Jonnassen 1998 & Hannafin 1999). The objectivist models are associated with behaviorism and cognitivism. They are both governed by an objective view of the nature of knowledge and what it means to know something. Behaviorism influenced traditional design models by providing prescriptions about the correlation between learning conditions and learning outcomes. Cognitive science has also contributed to traditional models by emphasizing the learner's schema as an organized knowledge structure. Objectivists believe that knowledge and truth exist outside the mind of the individual and are, therefore, objective. Learners may be told about the world and be expected to replicate its content and structure in their thinking (Jonnassen, 1991). Constructivists, on the other hand, believe that knowledge and truth are constructed by the learner and do not exist outside of his mind (Duffy and Jonassen, 1992). Therefore, according to constructivists, "learners construct their own reality or at least interpret it based upon their perceptions of experiences, so an individual's knowledge is a function of one's prior experiences, mental structures, and beliefs that are used to interpret objects and events."(Jonasson 1991).

As a theory of learning, Constructivism is a philosophical view about knowledge, understanding and learning that has roots both philosophy and psychology. The essential core of constructivism is that learners actively construct their own knowledge and meaning from their experiences (Fosnot 1996; Steffe and Gale 1995). Constructivism holds that learning is a process of building up structures of experience. By contrast with the traditional view of education as a process involving the transmission of knowledge from teachers to learners, a constructivist view believes that learning occurs through a process in which learners play active roles in constructing the set of conceptual structures that constitute their

own knowledge base.

Constructivist learning theory acknowledges that learners encode their understandings in language. Hence, communication is essential for all learning. Learning is a social act that cannot occur in isolation from others, even if they are not physically present. Knowledge, therefore, is something that belongs to both the individual and the community. Learning is set in context so that learners are aware of the purposes and applications of their learning, and recognize the effect that the context itself has on their learning.

Most of constructivist models are based on a set of philosophical assumptions and provide designers with a set of very general guidelines and principles that can facilitate designing a constructivist-learning environment. Among several constructivist instructional design models (e.g., Bednar et al.1995; Doolittle and Virginia 1999, Hannafin et al. 1999 & Jonassen 1999), Doolittle et al. (1999) put forward eight primary pedagogical recommendations in online education environment:

- Learning should take place in authentic and real-world environments.
- Learning should involve social negotiation and mediation.
- Content and skills should be made relevant to the learner.
- Content and skills should be understood within the framework of the learner's prior knowledge.
- Learners should be assessed formatively, serving to inform future learning experiences.
- Learners should be encouraged to become self-regulatory, self-mediated, and self-aware.
- Teachers serve primarily as guides and facilitators of learning, not instructors.
- Teachers should provide for and encourage multiple perspectives and representations of content.

Doolittle discussed each of the eight principles and gave each pedagogic statement a "grade" that reflects the online education's ability to meet or implement these statements. For example, Doolittle thought that the fourth pedagogic statement is the most difficult for online education to handle as probing and responding learner's prior knowledge in an asynchronous environment is less fluid and flexible than in synchronous environment. In the same period, Jonassen (1999), in his model, also presented eight design principles that can be used to design and develop what he calls the "constructivist learning environment". These design principles are as follows:

- Create real world environments that employ the context in which learning is relevant;

- Focus on realistic approaches to solving real-world problems;
- The instructor is a coach and analyzer of the strategies used to solve these problems;
- Stress conceptual interrelatedness, providing multiple representations or perspectives on the content;
- Instructional goals and objectives should be negotiated and not imposed;
- Evaluation should serve as a self-analysis tool;
- Provide tools and environments that help learners interpret the multiple perspectives of the world;
- Learning should be internally controlled and mediated by the learner.

It is not hard to find that these learning design principles for constructivism both emphasize on the fact that the designer should produces a learning environment that is much more facilitative in nature than prescriptive, the content should not be prescribed beforehand, direction should be determined by the learner and evaluation should be much more formative assessment and systems should provide more individual and collective learning supporting agency for facilitating learner to develop understanding through observation, reflection, experimentation, and interactions with the surrounding environment. These aspects are also what we consider very important in this thesis. As Doottle stated that the key to online education and constructivism is not whether or not the potential exists, but rather, whether or not the potential will be actualized.

Literature on the state of current online education market shows that most of today's e-Education systems are dominated by the objectivist school and the use of technology as a substitute for a teacher delivering instruction. Current approaches to the online learning environment usually transfer traditional classroom instruction to an online setting, recasting reading materials as web-based materials. These are basically mere Internet-based correspondence courses which rely on information acquisition and reflect low-level learning. As we see in most existing learning systems, the learners learn individually through computer-mediated communication. They interact with web-based instructional materials stored at remote locations and have minimal interaction with teachers and peers. The aim of such systems is to develop a planned online learning environment with structured, guided but often rigorous study courses and tasks for individual reflection and problem solving. These courses contain learning objectives, methods, materials and an evaluation scheme defined by the tutor him/herself. It contains the idea that there is a body of objective knowledge that can be delivered to learners through presentation and explanation. Obviously, The online learning environment based on objectivism has a

number of drawbacks, limitations and shortcomings (Mangal 1990), as it does not encourage learners to develop higher-order complex skills like creativity, problem-solving, designing and decision-making abilities and the acquisition of knowledge through social interaction.

It is worth noting that while the traditional objectivism has suffered much criticism, they still play important role in some context for example, training learners to perform a task the same way to enable consistency.

With the rapid evolution of information and communication, the advantages of constructivism are now attracting more and more instructional designers and practitioners to engage in learning design based on constructivism. Pedagogical methods using the constructivist approach include collaborative learning and creating learning situations that enable learners to engage in active exploration and social collaboration. The learners actively construct knowledge by formulating ideas into words, and these ideas are developed through the reactions and responses of others. The conversation (verbalizing), multiple perspectives (cognitive restructuring) and arguments (conceptual conflict resolutions) that arise in cooperative groups may explain why collaborative groups encourage a greater cognitive development than the same individuals achieve when working alone (Sharon 1980). According to the constructivist standpoint that knowledge has to be discovered, constructed, practiced, and validated by each learner, a collective learning environment is intended to develop complex skills like creativity, problem-solving, designing and decision-making abilities (McDonald et al. 1998). The best example supporting such collaborate learning is the current emerging CSCL (Computer-supported collaborative learning) paradigm that provides a framework to bring individual learners together to achieve a shared learning goal by managing their learning processes, using asynchronous and synchronous communication tools. Based on constructivist theories of learning, the online education environment can be designed on the assumption that learners themselves are an active agent and that they use social skills to undertake and complete group tasks and the tutor should foster the learner's constructive process and the role of tutor should be a guide or facilitator to provide expert feedback during knowledge building through structured collaborative learning tasks. One of the limitations of constructivist-based environment is that it does not always produce predictable learning outcomes. For example, in a situation where conformity is essential, divergent thinking and action may cause problems. Consequently, while the constructivism is becoming the trend of modern learning theory, there still exists some debate between objectivist and constructivist instructional design models (Bednar et al. 1995; Dick 1995 & Rowland 1995).

Especially some instructional design practitioners tend to believe that instructional designers and developers must allow circumstances surrounding the learning situation to help them decide which approach to learning is most appropriate. In addition, Schwier (1995) pointed out that it is necessary to realize that some learning problems may require prescriptive solutions, whereas others are more suited to learner control of the environment.

As we see it, an efficient distributed constructivist learning environment (CLE) is a technology-rich, open learning space where a learner can use a variety of tools and information or human resources in her/his pursuit of learning goals and problem-solving activities. Since CLEs are constructed from the perspective of learners, sensitive to their learning styles, needs, paces, preferences, prior knowledge, and therefore, such system should provide learners with scaffoldings that contain tools, strategies, and guides, which enable learners to interact with construction tools in ways that best enable them to build the learning systems at different levels of knowledge structure and technological sophistication.

2.4 INTELLIGENT TUTORING SYSTEM

This section briefly examines the state of the art on the main components of traditional intelligent tutoring system (ITS) and gives some summary analysis for its limitations as well as its implication to MAS based ITS.

Computer has been used in education for over 20 years, Computer-based training (CBT) and computer-aided instruction (CAI) were the first such systems deployed as an attempt to teach using computers. While both CBT and CAI may be somewhat effective in helping learners, they do not provide the same kind of individualized attention that a learner would receive from a human tutor (Bloom 1984). For a computer based educational system to provide such attention, it must reason about the domain and the learner. This has prompted research in the field of intelligent tutoring systems (ITSs). Especially, the Carbonell's (1970) proposal SCHOLAR system created an historical framework for ITSs that are computer-based instructional systems with models of instructional content that specify what to teach, and teaching strategies that specify how to teach (Wenger 1987, Ohlsson 1987).

The concept known as ITS or ICAI (Intelligent Computer-Aided Instruction) has many roots in Education, Psychology, and Artificial Intelligence (see figure 2-1). Nowadays, prototype and operational ITS provide practice-based instruction to support corporate training, college education, and military training.

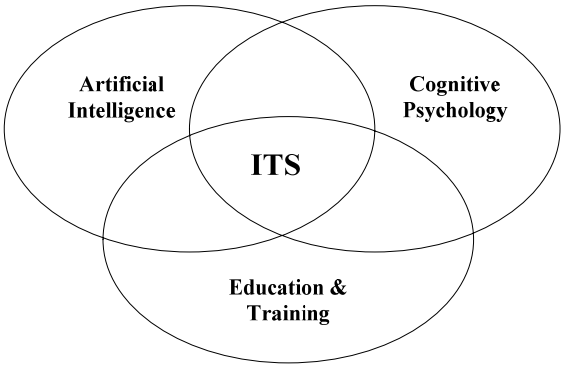


Fig 2-1. ITS Domain

The goal of an ITS is to provide the benefits of one-to-one instruction automatically and cost effectively. Unlike other computer-based training technologies, ITSs assess each learner's knowledge, skills, and expertise. Based on the learner model, ITS can tailor instructional strategies, in terms of both the content and style, and provide explanation, hints, examples, demonstration, and practice problems as needed. By contrast, to CBT, ITSs offer more considerable flexibility in presentation of material and a greater intelligent mechanism to adapt to learner individual needs. These systems achieve their intelligence by making inferences about a learner's mastery of topics or tasks in order to dynamically adapt the content or style of instruction. Content models (or knowledge bases, or expert systems, or simulations) give ITSs depth so that learners can "learn by doing" in realistic and meaningful contexts. Models allow content to be generated on the fly. ITSs allow "mixed-initiative" tutorial interactions, where learners can ask questions and have more control over their learning. Instructional models allow the computer tutor to more closely approach the benefits of individualized instruction by a competent pedagogue.

However, an ITS will typically constrain the learner to learn by a predetermined method or strategy (Rid 1989 & Kin 1997). ITS uses a model of the learner's knowledge (learner model) so that the learner is presented with new information only when he/she requires it. This is carried out in order to reinforce a point, to progress in the learning and/or to identify misconceptions and wrong-rules (Sle 1982). Such systems have been criticized for constraining the learner to solving a problem in a particular way (Rid 1989). In most complex problem domains, there can be many methods to achieve a correct solution. Some people may find one particular method that suits their way of thinking better than others, it has been argued that learners should be able to experiment with their own ideas and find

methods that naturally suit them.

Literature shows that a number of ITSs implement the cognitive tutoring strategy (Koedinger 2001; Anderson 1995). While most of today's ITSs may appear to be monolithic systems, for the purposes of conceptualization and design, it is often easier to think about them as consisting of several interdependent components. Previous research by Wanger (1984), Woolf (1994) and Oliveira (1994) has identified four major components: (1) the expert module containing the domain knowledge, (2) the learner module, which accumulates information about the learner's knowledge, misconceptions and behavior, (3) the pedagogic module, which includes the pedagogical expertise, and (4) the interface module (Figure 2-2).

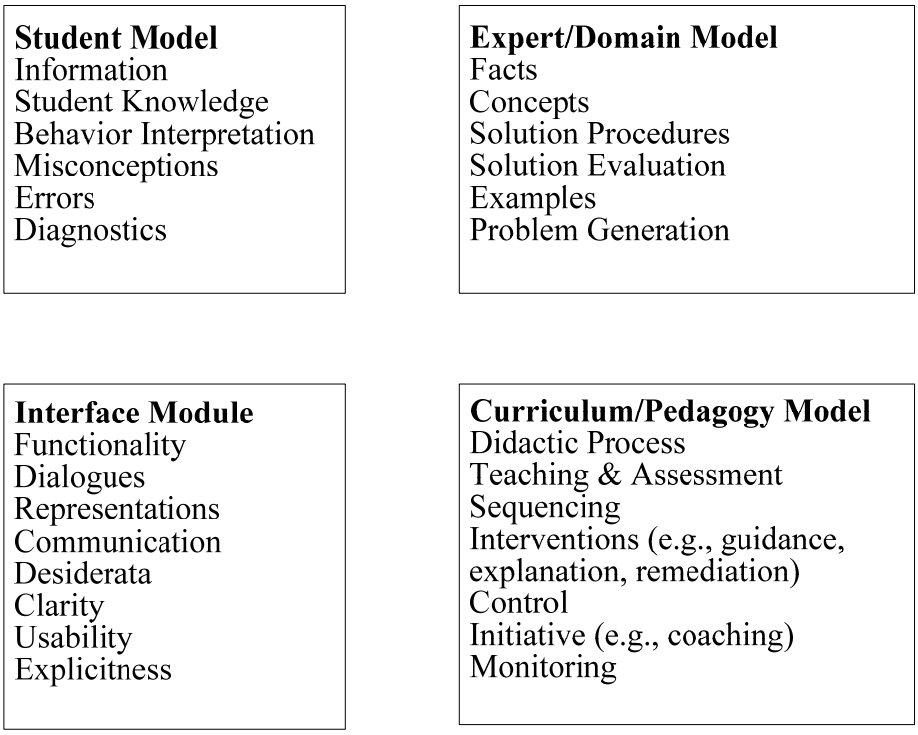


Fig 2-2. Intelligent Tutoring System Architecture adapted from (Wenger 1987)

- The expert module (domain model) is a dynamic representation of the knowledge domain. It contains domain knowledge, such as facts, concepts, processes, productions, and procedures required to solve problems. The model allows evaluation of the learner's solutions, and can provide examples of correct problem solutions.
- The pedagogy model (curriculum module) sequences the curriculum by comparing

the expert the learner model, has testing procedures that indicate the extent of the learner's knowledge, contains strategies that focus on how to teach best, and controls feedback. If the system is sensitive to misconceptions, this module contains remediation procedures.

- The interface module provides a uniform view of the environment to the user. It allows the user to interact with the system (as the curriculum module prescribes) by accessing the tutorial discourse, problems, examples, scores, progress summaries, representations, and resources (e.g., examples, diagrams, lectures).
- The learner model captures the knowledge status of the learner at any point during the teaching process. Simple learner models only keep track of topics that the learner has mastered, and which topics have not yet been covered (Knowledge Spaces, Doignon and Falmagne 1999). More advanced learner models record misconceptions, build bug libraries, and implementing VanLehn's repair theory (VanLehn 1990; Brown and VanLehn 1980), or record which teaching strategies work best for a specific learner. Unlike other modules, the learner model deals within formation that is specific to individual learners. Sison (1998) states in his learner model survey that every learner modeling system is obviously limited to observable responses of the learner to a stimulus in a domain. He calls this learner behavior, which can be plain input, an action, a result, or intermediate scratch work. Each of these options entails a different behavioral complexity and, thus, requires different strategies to extract useful modeling information. Sison categorizes systems by how many atomic behaviors they need to gain some information about the learner. He states that systems, such as Anderson's tutors (Anderson 1995) build one extreme of the spectrum, as they verify each keystroke of the user and, thus, use single behaviors to build the user model. Other modelers derive higher-level structures through rule or decision tree induction by using multiple behaviors as input.

While ITSs have proven to be successful on a small scale, several problems must be overcome before they have widespread impact on computer-based learning. Literatures from Wenger (1987), Bloom (1995), and Mutter (1988) among others have identified a wide range of limitations associated with the expert module, the learner model, the pedagogical module, and the interface. They stated that some fundamental shortcomings of ITS may not by overcome simply through incremental improvements to various ITS components. These limitations are summarized in table 2-6.

Table 2-6: Limitations of ITS

LIMITATIONS	EXPLANATION
In user modeling	
In educational design	Self (1990) states that the philosophy behind ITS is only concerned with "knowledge communication" (Wenger 1987) and that the goal of teaching is to "transmit a particular subject matter to the learner" (Ohlsson 1986). Therefore, this view neglects the general constructivist educational philosophy that holds that knowledge cannot be communicated or transmitted, but has to be actively constructed by learners themselves. However, the research community developing ITSs proved to be resistant to the adaptation of ideas from educational psychology for a long time (Self 1990).
In teaching and pedagogic expertise	Most of ITSs have very impoverished pedagogic component, Such components often comprise a collection of rules that seem to work reasonably well in practice, and there is no scientific encyclopedia of good tutoring heuristics to consult. To improve the pedagogic capability of a ITS, one needs to enhance the rule-base coaching of learners, this has to enrich the pedagogic knowledge base (Ohl 1991).
authoring, architecture and delivery platform,	bloom (1995) identified four problems in ITS: 1. ITS authoring is complex, requiring special domain authors. 2. The real world communities (the end users) seem unwilling to accept the ITS paradigm, mainly because they do not understand the technology and nothing is offered to the human teacher. 3. There is very little reuse of ITS architecture across applications, this argument corresponds with the criticism of Kinshuk (1997) in that a new ITS is generally developed for each new domain of applications. Bloom pointed out that an ITS mut have the ability to reuse the learner model, the instructional model and the knowledge base inference mechanism. 4. Most ITSs require specialist delivery platform, i.e., they may not suit the systems that the end-user may already have, although this is less likely to be case today as computer systems are more and cheaper.

These limitations may be account for the phenomena that very few ITSs are available over WWW. Besides the limitations mentioned above, one of the main difficulties in

designing intelligent tutoring systems is the time and cost required. A large team, including computer programmers, domain experts, and educational theorists, is needed to create just one ITS. Estimates of construction time indicate that 100 hours of development translates into 1 hour of instruction (Murray and Woolf 1992). One approach to alleviate the difficulties for developing ITSs is to provide Authoring Tool so that fewer developers would be needed for the construction of educational software. Another approach to simplifying ITS construction is to take advantage of the modularity of each system. Despite the natural breakdown of an ITS into the four components discussed previously, there has been little effort towards reusing components from one system in the development of another. This should not only involve developers just reusing their own components, but should also mean sharing components among different designers. Currently, several difficulties impede such modularization. First, tutors are written in many different programming languages that are incompatible. Second, this component view of ITSs is more of an ideal situation than a reality. Frequently, implementers intertwine the components into one monolithic system. Furthermore, since the field of ITS is relatively young, there are not accepted standards for kinds of components nor for their contents. Finally, there is no protocol for communication between the various parts of an ITS. In this thesis, we address such issues through incorporation of Multi-agent System. These distinct advantages of MAS such as reusability, modularity, standard agent communication language and interactive protocol make easy to solve most of the aforementioned problems in a natural and graceful fashion. As the reader can see, ITSs match very well with MAS technology, where each module could be an agent or a set of agents. The related MAS technology will be discussed in detail in the next section as it is the underpinning and scaffolding in support of building a personalized intelligent learning environment in this thesis.

2.5 MULTI-AGENT SYSTEM

Multi-agent system is a relatively new strand of research stemming from distributed artificial intelligence. This technology has long been perceived as a viable solution for large complex applications in dynamic environments such as the Internet and the Web. In particular, in the last few years, a plethora of models, toolkits, methodologies, modeling languages and so on have appeared in a very short period of time. In the domain of e-learning, MAS technology is now attracting more and more attention from researchers and practitioners because of its promise as a new paradigm for conceptualizing, designing, and implementing increasingly complex distributed learning environments in which various

resources (i.e., online learning resources, human resources, tools) are typically distributed in different locations in a dynamical fashion, system behaviors are often unpredictable and system requirements are always changeable. Annex 2 lists several recent applications of MAS to e-Education, which shows that MAS theory has begun to walk into internet applications although they are still far away from the mature phase.

In order to better reveal the features of MAS and provide sound theoretic foundation for subsequent chapters, the following sections will concentrate on reviewing the literature on topics closely relevant to this dissertation. Consequently, if necessary, some key points may be explained or demonstrated through some specific agents developed in our system.

2.5.1 BACKGROUND

In the past a few years, the Internet and Web have rapidly developed into the main platform of software applications in a wide range of domains. In addition to the common features of distributed systems such as concurrency, distributed, hypermedia, etc., the Web and Internet based systems have the new features of autonomy, evolutionary life-cycle, collaboration, etc. Consequently, software engineering of such systems is confronted with a number of challenges, such as to deal with service-oriented computing, dynamic integration of autonomous components, distributed and mobile computing, etc. Furthermore, real problems involve distributed, open systems (Hewitt 1986). An open system is one in which the structure of the system itself is capable of dynamically changing. The characteristics of such a system are that its components are not known in advance; can change over time; and can consist of highly heterogeneous agents implemented by different people, at different times, with different software tools and techniques. Perhaps the best-known example of a highly open software environment is the internet. The internet can be viewed as a large, distributed information resource, with nodes on the network designed and implemented by different organizations and individuals. In an open environment, information sources, communication links, and agents could appear and disappear unexpectedly. Currently, agents on the internet mostly perform information retrieval and filtering. The next generation of agent technology will perform information gathering in context and sophisticated reasoning in support of user problem-solving tasks. These capabilities require that agents be able to interoperate and coordinate with each other in peer-to-peer interactions. In addition, these capabilities will allow agents to increase the problem-solving scope of single agents. Such functions will require techniques based on negotiation or cooperation, which lie firmly in the domain of MASs (Jennings et al. 1998; O'Hare and Jennings 1996; Bond and Gasser 1988). The growth of the MAS field is indisputable. Research in MASs is concerned with the study, behavior, and construction of a collection

of possibly preexisting autonomous agents that interact with each other and their environments.

2.5.2 AGENT

In general, intelligent software agents have their root in three domains (i.e., Artificial Intelligence, Software Engineering and Human Interface). According to Russel and Norvig (1996), architecture and program compose an agent. From the architecture perspective, an agent is anything that can be viewed as perceiving its environment through sensor and acting upon that environment through effectors. However, from the software perspective, an agent is substantially a program, which has a specific plan of action, defined in a limited domain and a behavior pattern, which allows it to change at the right moment, its own interaction with the world depending on stimuli from the environment (Colazzo and Silvestri 1997).

Wooldridge and Jennings (1995) divided the notions of agents in two ways: a weak and a strong approach. The weak notion of agent refers to the following properties:

- *autonomy*: agents work by their own and have some kind of control over their actions and internal state;
- *social ability*: agents interact with other agents (and humans beings) via some kind of agent communication language and common ontology;
- *reactivity*: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or all of these combined), and respond in a timely fashion to changes that occur in it;
- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.

The strong notion of agent, in addition to having the properties identified above, is either conceptualized or implemented using concepts that are more usually applied to humans. For example, it is quite common in Artificial Intelligence to characterize an agent using mental notions, such as knowledge, belief, intention, and obligation.

In AI literature, some researchers have further pointed out other interesting properties as follow:

- Agents must be subservient, i.e., must act on behalf of someone else. It is the originalsense of the term in AI, where an agent is to perform other's instructions explicitly (Shoham 1993);
- Agents are persistent software entities (they work all the time during program execution) dedicated to a specific purpose (so they are distinguished from subroutines) (Simith et al. 1994);

- They have many functions, but three are essential: perception of dynamic conditions of the environment action to affect these conditions, reasoning to interpret perceptions, solve problems, draw inferences and determine actions (Hayes-Roth 1995);
- Agents must be rational - a rational agent is one that does the right thing. The agent rationality depends on the performance measure that defines degree of success and perception history (the actions that the agent can perform) (Russel and Norvig 1996);

2.5.3 MULTI-AGENT SYSTEM

Today's distributed systems are facing to more and more complex, realistic, and large-scale problems. Such problems are obviously beyond the capabilities of an individual agent. The capacity of a single intelligent agent is limited by its knowledge, its computing resources, and its perspective. Just as in human society we have the adage that "no man is an island", indicating the fact that no one person is sufficient on her own so also it applies in the agent universe. A multi-agent system is a collection of agents; each agent is situated in some environment and is able to interact with its environment and with other agents. As seen from Distributing Artificial Intelligent, a multi-agent system is a loosely coupled network of problem solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity. More recently, the term multi-agent system has been given a more general meaning, and it is now used for all types of systems composed of multiple autonomous components showing the following characteristics (Flores-Méndez, 1999):

- Each agent has incomplete information or capabilities to solve a problem and thus a limited viewpoint.
- There is no global system control.
- Data is decentralized.
- Computation is asynchronous.

Therefore, a multi-agent system should have the following skills (Camacho, 2002):

- Social Organization
- Coordination
- Cooperation
- Negotiation
- Communication

Obviously, such skills differentiate MAS from other related disciplines such as distributed computing, object-oriented systems, and expert systems. In particular, the motivations or the increasing interest in MAS research include the ability of MASs to do the following:

- First is to solve problems that are too large for a centralized agent to solve because of resource limitations or the risk of having one centralized system that could be a performance bottleneck or could fail at critical times. Such problem is increasingly obvious in today's client-server distributed computing paradigm.
- Second is to allow for the interconnection and interoperation of multiple existing legacy systems. To keep pace with changing business needs, legacy systems must periodically be updated. Completely rewriting such software tends to be prohibitively expensive and is often simply impossible. Therefore, in the short to medium term, the only way that such legacy systems can remain useful is to incorporate them into a wider cooperating agent community in which they can be exploited by other pieces of software. Incorporating legacy systems into an agent society can be done, for example, by building an agent wrapper around the software to enable it to interoperate with other systems (Genesereth 1994).
- Third is to provide solutions to problems that can naturally be regarded as a society of autonomous interacting components-agents. For example, in meeting scheduling, a scheduling agent that manages the calendar of its user can be regarded as autonomous and interacting with other similar agents that manage calendars of different users (Garrido and Sycara 1996; Dent et al. 1992). Such agents also can be customized to reflect the preferences and constraints of their users. Other examples include air-traffic control (Kinny et al. 1992; Cammarata, McArthur, and Steeb 1983) and multi-agent bargaining for buying and selling goods on the internet.
- Fourth is to provide solutions that efficiently use information sources that are spatially distributed. Examples of such domains include sensor networks (Corkill and Lesser 1983), seismic monitoring (Mason and Johnson 1989), and information gathering from the internet (Sycara et al. 1996).
- Fifth is to provide solutions in situations where expertise is distributed. Examples of such problems include concurrent engineering (Lewis and Sycara 1993), health care, and manufacturing.
- Sixth is to enhance performance along the dimensions of (1) computational efficiency because concurrency of computation is exploited (as long as communication is kept minimal, for example, by transmitting high level information and results rather than low level data); (2) reliability, that is, graceful recovery of component failures, because agents with redundant capabilities or appropriate inter agent coordination are found dynamically; (3) extensibility because the number and the capabilities of agents working on a problem can be altered; (4) robustness, the system's ability to tolerate uncertainty,

because suitable information is exchanged among agents; (5) maintainability because a system composed of multiple components-agents is easier to maintain because of its modularity; (6) responsiveness because modularity can handle anomalies locally, not propagate them to the whole system; (7) flexibility because agents with different abilities can adaptively organize to solve the current problem; and (8) reuse because functionally specific agents can be reused in different agent teams to solve different problems.

2.5.4 MOBILE AGENT

Today's most widespread paradigm for distributed computing follows the client-server paradigm. In this paradigm, the server is defined as a computational entity that provides some services. The client requests the execution of these services by interacting with the server. After the service is executed, the result is delivered back to the client. The server therefore provides the knowledge of how to handle the request as well as the necessary resources. A limitation of the client-server model is that the client is limited to the operations provided at the server. Therefore, if a client needs a service that a particular server does not provide, the client must find a server that can satisfy the request by sending out messages to all servers. This clearly is an inefficient use of network bandwidth. As a valuable alternative to the traditional programming model, the mobile agent paradigm involves the mobility of an entire computational entity, along with its code, state and probably the potential resources (e.g., ontology schemas) from host to host on a network. Dag (1994) implemented a computational metaphor that is analogous to how most people conduct business in their daily lives: visit a place, use a service, and then move on. Generally, the itinerary map whereby to travel through different network nodes can either be predefined or determined by the mobile agent on the fly, based on the its current state or its computing logic.

As compared with the traditional client-server model, the mobile agent paradigm has several advantages shown as below (Lange 1999, Wong 1999, and Chess 1998):

- *Communication latency and bandwidth*: If the communication between two interacting entities involves a considerable amount of data, it may be beneficial to move one of them close to the server instead of moving the data between them. The locality of the two entities allows them to decrease the latency and save bandwidth in the communication. Based on the locality information the agent may decide to move to the server location instead of invoking the server functions remotely. Since the interaction is carried out locally, it is independent of the network traffic.
- *Asynchronous execution*: Instead of interacting with a server in many RPC-type communications, a client can bundle the requests within a mobile agent. Having reached the

server the agent starts interacting with the services locally. This allows the user to disconnect from the network when delegating agent to perform certain task on the remote machine. Particularly in mobile computing, roaming devices such as PDA or laptops are often disconnected from the network. For instance, a simple scenario may be happen in MAGE: one morning, Professor Julier switches on her computer, logs on to her campus portal, her assistant agent named Angie appears as animated character on her screen. “Good morning, Prof. Julier, here is a coming exam activity taking place next week Thursday.” Prof. Julier arouses her Test Assistant Agent named Louis, “MAEAm, What can I help you?” Prof. Julier begins to type in the exam paper control parameters, requests Louis to go for a trip, and eventually turns off her computer and leaves her office. The next day, when Prof. Julier log on again, holding coffer on hand, Louis is coming back with a set of exam paper……

- *Dynamic adaptation*: Mobile agents have the ability to adapt dynamically to changes in their environment. They can, for instance, react autonomously to balance the load in the network or move on to a replica of a current node that is failing. This behavior also provides mobile agents with robustness and a degree of fault tolerance. For example, in automatic test generation subsystem of MAGE, when the destination server where the item pool resides collapse, it is possible for the Control Agent to automatically choose the nearby mirror servers for continued execution.
- *Protocol encapsulation*: Today's networks consist of many legacy applications. As their protocols evolve, legacy problems often occur. Mobile agents can move to the remote legacy application and encapsulate its protocol. Thus, other applications can communicate with this application via the agent.
- *Parallel execution*: since mobile has the ability to clone itself to perform parallel tasks. The GAMASTP in MAGE is a typical application of parallel execution.

The disadvantage of mobile agents is the need to install a support agent platform at each host the agents need to visit. Besides, agent code and data must be as short as possible in order to achieve the benefits of the technology. Furthermore, it is important to mention security. This hard problem is faced by all technologies of developing distributed systems (Chess 1998; Tahara 2000).

Although the mobile agent technology has not yet found its way into today's more prominent applications, whereas the potential benefit of the mobile agent paradigm has been widely accepted, especially in the past few years, several mobile agent platforms such as (JADE 2006), Agent Tcl (Tcl 2005), Aglets (IBM 2005), Mole (Mole 2005), and Voyager (Voyer 2006) have emerged. They greatly stimulate and promote the development

of mobile agent based applications

In MAGE, within the framework of JADE, we designed and implemented a couple of mobile agents running on online examination system. Each of them is responsible for distinct particular tasks corresponding to generating test, deliver test and evaluate test, respectively. More details can be referred to chapter 7.

2.5.5 FIPA STANDARD

The Foundation for Intelligent Physical Agents (FIPA) is an international non-profit organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications in order to fulfill the requirements of today's dynamic, heterogeneous and distributed service provisioning applications. Within the arena of distributed software infrastructures, FIPA promotes a landscape where agent platforms provide life support to communities of agents, which in turn cooperate to enable services and application. FIPA tries to support both agent-level and platform-level interoperability through a comprehensive set of specification. Nowadays, it has widely been accepted as a de facto standard by most agent community. Since MAGE intends to develop MAS based intelligent learning environment conforming to FIPA specification in order to enhance the interoperability and reusability with other FIPA-compliant MAS learning systems, it is worth examining the state of the art on agent management reference model, agent communication language, agent protocol library. As a matter of fact, the aforementioned aspects form the underpinning of the agents running on MAGE for efficient agent management, maintenance, and communication, cooperation, etc.

2.5.5.1 AGENT MANAGEMENT REFERENCE MODEL

Agent management provides the normative framework within which FIPA agents exist and operate. It establishes the logical reference model (see Figure 3) for the creation, registration, location, communication, migration and retirement of agents.

From figure 2-3, it can be seen that an *Agent Platform* (AP) provides the physical infrastructure in which agents can be deployed. The AP consists of the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and MTS) and agents. Such logical components, each representing a capability set (i.e., services) are further described as follows:

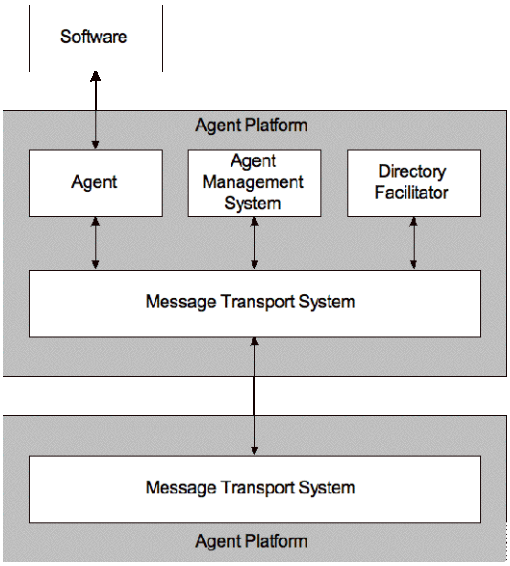


Fig 2-3. Agent Management Reference Model

• An *agent* is a computational process that implements the autonomous, communicating functionality of an application. Agents communicate using an *Agent Communication Language* (ACL) and *Agent Ontology*. For example, In MAGE, most of agents can speak both XML and SL content languages, besides, to smooth the communication and mutual understanding for specific application domains, several ontologies have also been developed including *Leaner-Model-Ontology*, *Learning-Content-Management-Ontology*, *Test-Ontology* and *CSCL-Ontology*. Of course, the *System-Management-Ontology* and *Mobility-Ontology* are ready-made and available anytime thanks to adoption of the modern FIPA-Compliant middleware-JADE. According to FIPA, an agent is the fundamental actor on an *AP* which combines one or more service capabilities, as published in a service description, into a unified and integrated execution model. An agent must have at least one owner, for example, based on organisational affiliation or human user ownership, and an agent must support at least one notion of identity. This notion of identity is the *Agent Identifier* (AID) that labels an agent so that it may be distinguished unambiguously within the agent universe. In fact, sometime it is not easy to give agent an appropriate name. Consider, in automatic test generation subsystem of MAGE, a population of agents are dynamically being created and killed during the evolution process, in such situation, how to maintain unique and unambiguous AIDs is one key task.

- A *Directory Facilitator* (DF) is an optional component of the AP, but if it is present, it must be implemented as a DF service. The DF provides yellow pages services to other agents. Agents may register their services with the DF or query the DF to find out what services are offered by other agents. For example, the LCMDFAgent (Learning Content Management DF Agent) and HelperDFAgent in MAGE, are implemented as DFs which provides learning content finding service and peer learner finding service, respectively. Actually, Multiple LCMDFs exist within MAGE and they are federated in order to cater for possible different Learning Object providers.
- An *Agent Management System* (AMS) is a mandatory component of the AP. The AMS exerts supervisory control over access to and use of the AP. Only one AMS will exist in a single AP. The AMS maintains a directory of AIDs which contain transport addresses (amongst other things) for agents registered with the AP. The AMS offers white pages services to other agents. Each agent must register with an AMS in order to get a valid AID.
- An *Message Transport Service* (MTS) is the default communication channel between agents on different APs.

It should be noted that the concept of an AP does not mean that all agents reside on an AP have to be co-located on the same host computer. FIPA envisages a variety of different APs from single processes containing lightweight agent threads, to fully distributed APs built around proprietary or open middleware standards.

2.5.5.2 AGENT COMMUNICATION LANGUAGE

The message types (communicative act types) are central to ACL specifications, which impart a meaning to the whole messages, so it is worth having a look at the structure of an ACL message. Figure 2-4 is an example of the structure of a message by the Learner Model Agent sending to the Pedagogic Agent.

Figure 2-4 summarizes the main structural elements of an ACL message represented as s-expressions. The first element of the message is a word that identifies the communicative act being communicated, which defines the principal meaning of the message. There then follows a sequence of message parameters, introduced by parameter keywords beginning with a colon character. One of the parameters contains the content of the message, encoded as an expression in some formalism. Other parameters help the message transport service to deliver the message correctly (e.g. sender and receiver), help the receiver to interpret the meaning of the message (e.g. language and ontology), or help the receiver to respond in a conversion (e.g. reply-with, reply-by). This transport form is

serialized as a byte stream and transmitted by the message transport service. The receiving agent is then responsible for decoding the byte stream, parsing the components message and processing it correctly.

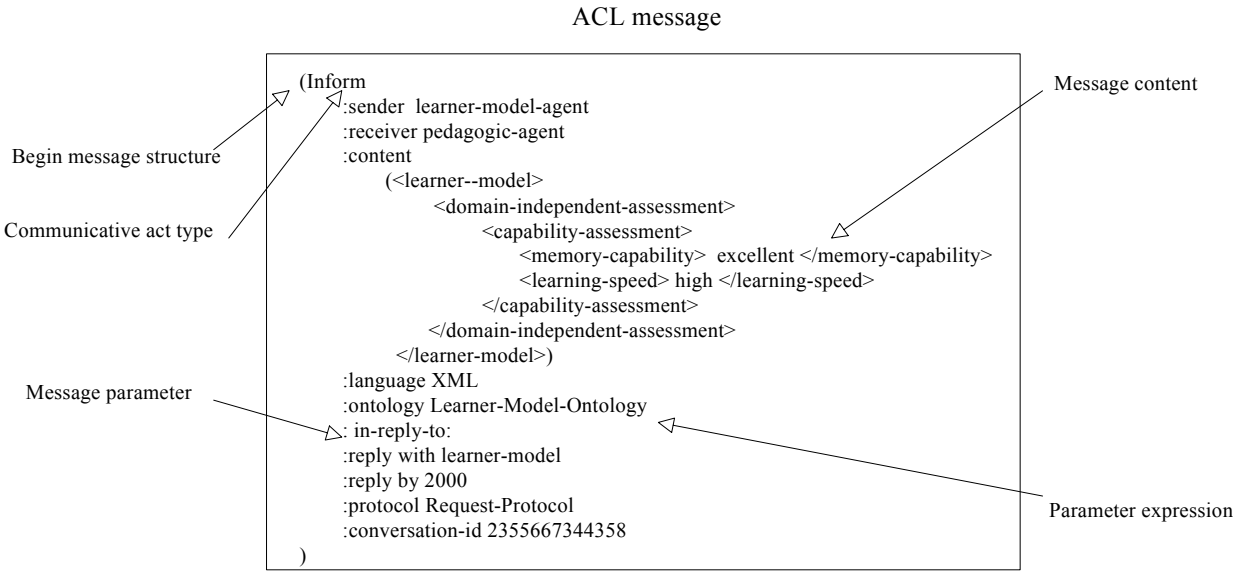


Fig 2-4. Components of a message

As noted above, the message contains a set of parameters. Parameters may occur in any order in the message. The only parameter that is mandatory in all messages is the *receiver* parameter, so that the message delivery service can correctly deliver the message. Clearly, no useful message will contain only the receiver. However, which other parameters are needed for effective communication will vary according to the situation. Further interpretations of the message parameters are shown as follows combined with the above sample:

- *Sender*: Denotes the identity of the sender of the message, i.e. the name of the agent of the communicative act. e.g., learner-model-agent
- *Receiver*: denotes the identity of the intended recipient of the message. e.g., pedagogy-agent.
- *Content*: denotes the content of the message; equivalently denotes the object of the action. In general, the content can be encoded in any language, and that language will be denoted by the *language* parameter.
- *Reply-with*: introduces an expression, which will be used by the agent responding to this message to identify the original message. Can be used to follow a conversation

thread in a situation where multiple dialogues occur simultaneously.

- *In-reply-to*: denotes an expression that references an earlier action to which this message is a reply.
- *Language*: denotes the encoding scheme of the content of the action. e.g., as XML in this sample.
- *Ontology*: denotes the ontology that is used to give a meaning to the symbols in the content expression. e.g., as Learner-Model-Ontology in this sample.
- *Reply-by*: denotes a time and/or date expression, which indicates a deadline on the latest time by which the sending agent would like a reply. e.g., as 2s in this sample.
- *Protocol*: introduces an identifier that denotes the *protocol* that the sending agent is employing. The protocol serves to give additional context for the interpretation of the message. e.g., as FIPA Request-Protocol in this sample, more details can be referred to the next section.
- *Conversation-id*: Introduces an expression, which is used to identify an ongoing sequence of communicative acts which together form a conversation. A conversation may be used by an agent to manage its communication strategies and activities. In addition, the conversation may provide additional context for the interpretation of the meaning of a message. e.g., as 235566734358 in this sample referring to the conversation identifier used to identify such kind of communication.

The available message types in FIPA ACL are grouped into the communicative act library, which gives an informal and a formal explanation of the meaning of each communicative act, grounding the whole library in a semantic framework using the speech act theory as the communicative model, and the Beliefs-Desires-Intentions (BDI) model as the formal logic framework. The detailed interpretation can be referred to annex 1.

2.5.5.3 INTERACTION PROTOCOL LIBRARY

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, which are used to design agent interaction providing a sequence of acceptable messages and a semantic for those messages. While each FIPA ACL message is given a formal semantics based on the speech-act theory, this is still not enough to satisfy the need for sociality of agent system. This is because a typical agent interaction encompasses more than a single message, so more comprehensive abstractions are needed.

FIPA specifications provide this abstraction as a collection of interaction protocols. Some of the most significant FIPA interaction protocols are:

- FIPA-request
- FIPA-query
- FIPA-request-when
- FIPA-contract-net
- FIPA-iterated-contract-net
- FIPA-auction-english
- FIPA-auction-ducth

FIPA-request protocol allows an agent to request another agent to perform some action. This is similar to ordinary request/response protocols used in client/server systems, but with a significant difference. Since software agents are autonomous entities, an agent can refuse to perform the requested action even if able to do so. So, while a client/server call either succeeds or fails raising an exception, a FIPA-request interaction can succeed, can fail for a capability lack of the receiver agent, but can also fail for the unwillingness of receiver to perform the task at hand. This FIPA protocol supports the whole set of outcomes arising from the interaction: the agent starting protocol sends a request message to its peer, containing the action it wants its peer to perform, if the receiving agent knows nothing about the action requested it could answer with a not-understood message. Then, the responder agent decides if it want to try to satisfy the request; if so, the answer will be an agree message, otherwise a refuse message is sent back. After the agree communicative act, the responder agent actually tries to execute the action, if all goes smoothly, the initiator is notified with an inform message, otherwise a failure is sent. The other details of interaction protocols can be referred to (FIPA 2000).

2.6 SUMMARY

This chapter examined the state of art on several related topics on e-Education. Through the review and analysis on the present status and future tendency of e-Education, it is found that in the current e-Education paradigm exist several disadvantages simply pointed out as follows:

- From instructional design model perspective, most of e-Education systems are based on objectivism rather than constructivism. Current approaches to the online learning environment usually transfer traditional classroom instruction to an online setting, recasting reading materials as web-based materials. These are basically mere Internet-based correspondence courses which rely on information acquisition and reflect low-level

learning;

- From cognitive control strategy perspective, most of e-Education systems are system-centered rather than learner-centered, the learner cannot control the content sequencing and learning pacing according to his/her preference and status;
- From network computing paradigm perspective, the client-server paradigm is predominant in the existing e-Education systems. Although this paradigm greatly promotes the development of e-Education, it has several drawbacks in terms of bandwidth; parallel computing, resource distribution, etc. (see 2.5.4).
- From design and implementation perspective, the main modules in e-Education systems are intertwined and thus interdependent.
- From user modeling perspective, the traditional e-Education systems apply centralized server to modeling learner. It is obvious that such paradigm makes difficult to manage and coordinate the data that are distributed, dynamical and sometime even unpredictable.
- From personalization and adaptation perspective, the existing e-Education systems just present the same interface and prescribed learning content to all the users, thus apparently, these systems cannot personalize the learning process (e.g.,. according to learning style and learning preference) and adapt to the learner's knowledge status and cognitive level.
- From interactive process perspective, the existing e-Education systems make hard to communicate among tutors, learners, and other actors involved.

Based on the analysis on the state of art on the e-Education domain, it is apparent that it is hard for a single discipline to address all the issues mentioned above. This is a typical cross-discipline issue. In this dissertation, some new solutions to the existing issues have been put forward and implemented taking advantage of some new methodology, theory and tools. Especially, we apply MAS to the design and implementation of an e-Education system; this is a significant shift from both software programming and network computing paradigm perspective. This shift introduces a peer-to-peer network-computing paradigm instead of the traditional client-server paradigm. These distinct advantages of MAS such as reusability, modularity, standard agent communication language and interactive protocol make promising to solve most of the aforementioned problems in a more natural and graceful fashion. Besides the advantages mentioned in section 2.5, the most encouraging feature of MAS consists in its capability as a container that can encapsulate any possible tools and methodology involve in related disciplines. Therefore, when designing a MAS based e-Education system, one possibility is to replace the components of ITS implemented

as a monolithic system by a set of intelligent agents on behalf of learners, tutors, tools, pedagogic tactic or electronic resources, respectively. These agents could model learners in just-in-time manner, offer learners on-demand suggestions, build dynamical adaptive learning group, motivate learners as needed, recommend desired peer helpers, personalized learning materials, or administer adaptive test etc. Obviously, theses functionalities require that we not only take into considerations the technology behind agents but the learner traits such as different backgrounds, status of knowledge and learning styles in order to build a fully-fledged individualized constructivist learning environment that facilitates learning collaboration, learner autonomy, reflectivity and active engagement.

In the following chapters, we will concentrate on the research on how to model and implement such a personalized and constructivist e-Education system taking advantage of MAS and other relevant methodology, theory, and standards discussed in this chapter. In particular, the focus of this paper will be aimed at how to develop an efficient MAS-based e-Education system in order to

- personalize the learner's learning process according to his/her learning preference, learning style;
- offer adaptive course sequencing and navigation according to the learner model and domain model;
- facilitate the developing process of courses and learning objects combined the LO standard and MAS ;
- facilitate the test generation, delivery, evaluation and publication applying mobile agent technology;
- provide seamless access for learners to a variety of distributed help resource including human resources as well as electronic resources;
- facilitate collective thinking, collaborative endeavor, knowledge sharing.

CHAPTER 3 ARCHITECTURE OF MAGE

This chapter firstly analyzes e-Education reference model based on LTSA of IEEE LTSC. The result shows that it is convenient for us to map the abstraction functionality of LTSA to each sub modular of MAGE, thus we could eventually build an e-Education system that fully comply with the LTSA of IEE LTSC. Secondly, this chapter presents a recommended architecture of multi-agent e-Education system, which consists of three types of agents: learner-side agents, server-side agents and learning content-side agents. Finally, several learning scenarios (i.e., intelligent tutoring system without the inversion of teachers; teacher-centered learning paradigm; collective learning paradigm based learning task) are also illustrated by the corresponding UML sequence schemas.

3.1 INTRODUCTION TO E-EDUCATION REFERENCE MODEL

As described in chapter 2, multi-agent technology will possibly bring us a powerful and flexible solution to e-Education context thanks to its anthropopathic “intelligence” feature such as autonomy, communication, cooperation, coordination and negotiation among agents, however, from the system engineering perspective, it is, initially, imperative that we build a flexible, robust reference model at the abstract level before implementing this system. In general, this reference model should be a top level of functional abstract regardless of the specific platform, content or technology. Through corporate effort, a recommended e-Education reference model (REERM) (see figure 3-1) has already been identified, which will act as the prototype model for guiding the concrete design of MAGE. However, what surprises us most is that our recommended reference model is inconceivably similar to the Leaning Technology Systems Architecture (LTSA 2004) of IEEE LTSC (IEEE Learning Technology Standards Committee). So it is first worth spending some effort on bringing REERM and LTSA into comparison. The comparison result shows that the two models are basically identical though the latter considers more aspects. As such, more reinforced confidence and encouragement will urge us to eventually achieve a pro forma implementation conformance to the LTSA of IEEE LTSC through incorporating the agent technology into our e-Education system.

3.1.1 A RECOMMENDED E-EDUCATION REFERENCE MODEL

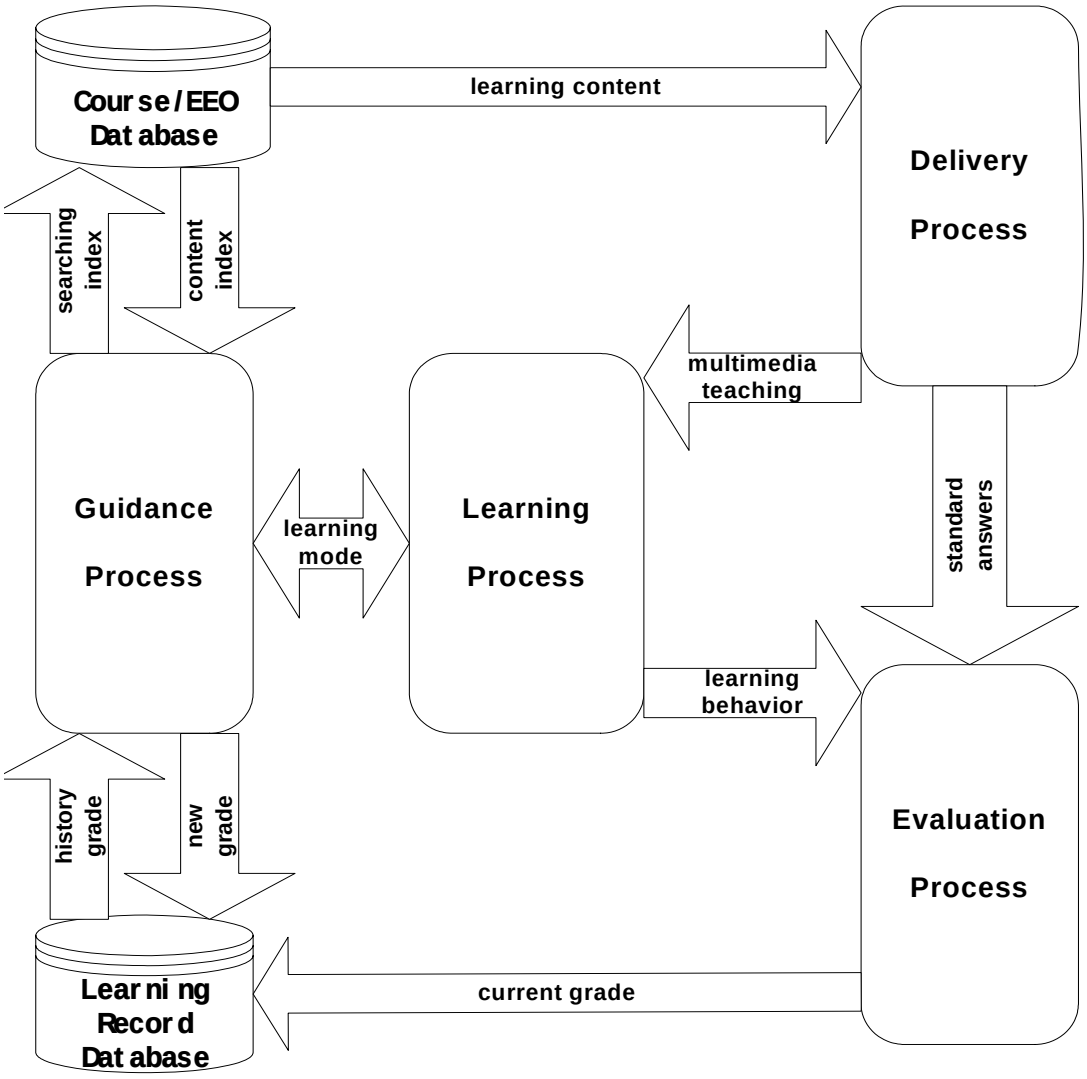


Fig. 3-1 A recommended e-Education reference model (REERM)

As shown above, the REERM identifies four processes: learner process, evaluation process, guidance process, and delivery process; two stores: learner records database and learning course/EEO database; and ten information flows among these components: learning behavior, multimedia, learning content, content index, searching index, history grade, new grade, current grade, standard answers and learning mode.

1) Learning Process

- Receives learning contents from Delivery Process;
- Moreover, the learner's learning behavior is sent to the Evaluation Process;
- The learner can select the appropriate learning mode with help of Guiding System.

2) Evaluation Process

- Receive learning behavior from the learner and code it;
- Obtain evaluation information according to the correct answers from Delivering Process and send it to Guidance Process;
- Store current grades and individual's learning progress chart into Learning Record Database.

2) Guidance Process

- Help select some appropriate learning mode for each learner;
- Mode choice may be determined by either this module or the learner、the teacher or the learner's parents.
- According the selected learning mode, this module can produce content index through Searching Index Engine and send it to Delivery Process.

3) Delivery Process

- Retrieve learning contents, which should include a variety of resource such as audio、video、documents、PowerPoint and so on.
- Transform the learning contents into the format of multimedia and present them to learner.
- Send corresponding standard answers to Evaluation Process.

3.1.2 LEANING TECHNOLOGY SYSTEMS ARCHITECTURE (LTSA) OF IEEE LTSC

LTSA specifies a high-level architecture for information technology-supported learning, education, and training systems that describes the high-level system design and the components of these systems. This Standard covers a wide range of systems, commonly known as learning technology, education and training technology, computer-based training, computer assisted instruction, intelligent tutoring, metadata, etc. This Standard is pedagogically neutral, content-neutral, culturally neutral, and platform-neutral. This Standard (1) provides a framework for understanding existing and future systems, (2) promotes interoperability and portability by identifying critical system interfaces, and (3) incorporates a technical horizon (applicability) of at least 5-10 years while remaining adaptable to new technologies and learning technology systems.

3.1.3 LTSA OVERVIEW

Five refinement layers of architecture are specified, but only layer 3 (system components) is normative in this Standard. This architecture is applicable to a broad range of learning scenarios. These refinement layers are called, from highest to lowest levels (see figure 3-2):

- Learner and Environment Interactions (informative): Concerns the learner's acquisition, transfer, exchange, formulation, discovery, etc. of knowledge and/or information through interaction with the environment.
- Learner-Related Design Features (informative): Concerns the effect learners have on the design of learning technology systems.
- System Components (normative): Describes the component-based architecture, as identified in human-centered and pervasive features.
- Implementation Perspectives and Priorities (informative): Describes learning technology systems from a variety of perspectives by reference to subsets of the system components layer.
- Operational Components and Interoperability — coding, APIs, protocols (informative): Describes the generic "plug-n-play" (interoperable) components and interfaces of information technology-based learning technology architecture, as identified in the stakeholder perspectives.

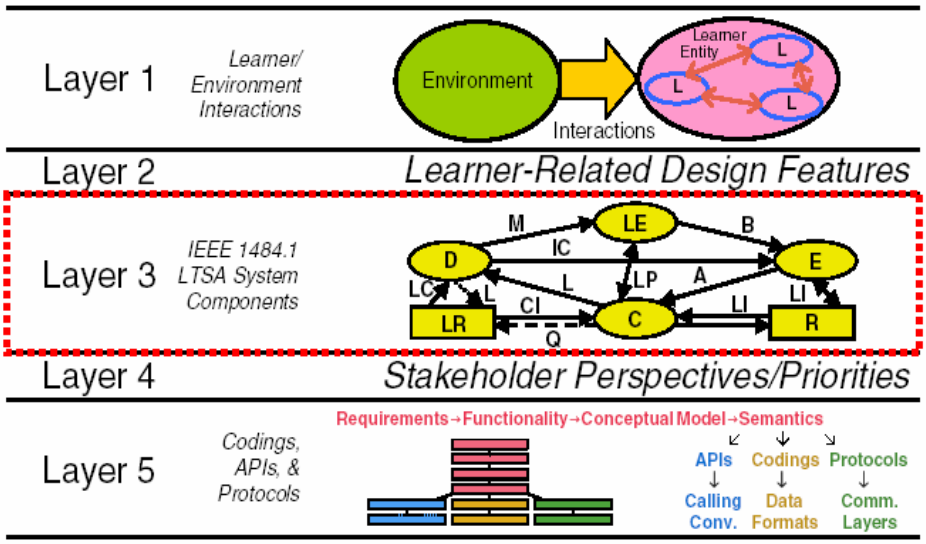


Fig. 3-2 The LTSA abstraction-implementation layers.

Note: Only layer 3 (system components) is normative in this Standard.

The LTSA is described in five successive refinement layers from highest to lowest. Each layer describes a system at a different level. The lower layers are implementations of the higher layers; the higher layers are abstractions of the lower layers. The five layers represent five independent areas of technical analysis. For example, it is possible to discuss an abstraction (e.g., the LTSA system components — layer 3, similar to REERM), independently of an implementation (e.g., the coding, APIs, and protocols of an actual implementation —layer 5). In other words, even though layer 3 contains components such as "evaluation" and "coach", these components are "conceptual" in that there is no

requirement for separable, identifiable components called "evaluation" and "coach" in actual implementations. Because layer 3 (i.e. System Components) is normative and the core of LTSA, so it needs us to take some extra effort to anatomize layer 3 (see figure 3-3)

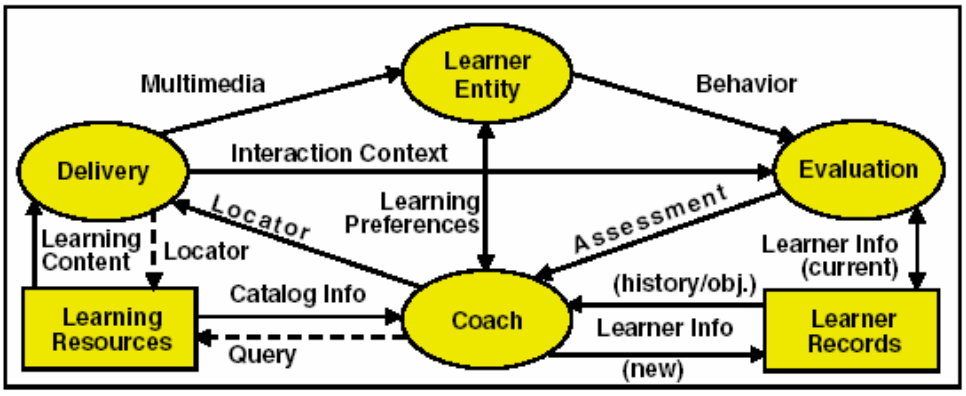


Fig. 3-3 LTSA system components.

The LTSA identifies four processes: learner entity, evaluation, coach, and delivery process; two stores: learner records and learning resources; and thirteen information flows among these components: behavioral observations, assessment information, performance and preference information (three times), query, catalog info, locator (twice), learning content, multimedia, interaction context, and learning preferences. Briefly, the overall operation has the following form:

- The learning styles, strategies, methods, etc., are negotiated among the learner and other stakeholders and are communicated as learning preferences;
- The learner is observed and evaluated in the context of multimedia interactions;
- The evaluation produces assessments and/or performance information;
- The performance information is stored in the learner history database;
- The coach reviews the learner's assessment, preferences, past performance history, and, possibly, future learning objectives;
- The coach searches the learning resources, via query and catalog info, for appropriate learning content;
- The coach extracts the locators from the available catalog info and passes the locators to the delivery process, e.g., a lesson plan; and
- The delivery process extracts the learning content from the learning resources, based on locators, and transforms the learning content to an interactive multimedia presentation to the learner.

Compared figure 7 with figure 5, we can see they are basically analogous. For example,

- LTSA Coach can be mapped to REERM Guidance Process, similarly,
- LTSA Learner Entity vs. REERM Learning Process;

- LTSA Delivery vs. REERM Delivery Process;
- LTSA Evaluation vs. REERM Evaluation Process;
- LTSA Learner Records vs. REERM Learning Record Database;
- LTSA Learning Resources vs. REERM Course/Learning Object.

3.1.4 STAKEHOLDER PERSPECTIVES

In LTSA, Stakeholder represents a group of persons, organizations, or entities that have a common interest. For example, a "learning object (EEO) developer" stakeholder represents all those who have material interest in learning object development; an “e-Education system developer” stakeholder represents all those who have material interest in e-Education system development, and so on. In LTSA, The stakeholder perspectives layer is considered a separate refinement layer because this layer of granularity addresses a particular design issue: which perspective, view, or subset is relevant to the lower-level design.

In LTSA Each stakeholder may identify its stakeholder perspective by using an existing LTSA stakeholder diagram. The annex of LTSA (2001) contains an informative summary of over 120 stakeholder perspectives (e.g., learner centered, assessment centered, content developer, learning object,). It seems to be exhausting for us to understand so many perspectives. In fact, it is not the case. The results of this analysis have been: (1) verification and validation of the LTSA components in significant systems, stakeholders, and industries, (2) discovery of which LTSA components are emphasized and de-emphasized in different systems, stakeholders, and industries, and (3) indication of varying priorities among higher-level and lower-level design issues.

As for MAGE, it is easy for us to find many appropriate “stakeholders” relevant to MAGE (e.g., “learner centered”, “learning object”, “distributed learning”,” simulation”, “content packaging”, “Intelligent tutoring tools”, “Learning tool to tool communication”, etc.). Since MAGE is such a complicated distributed system, which may include several subsystems that correlate yet have different emphasis/priority, that we should find an effective analytical tool/method to determine which functionalities are more important or less important in each subsystem (e.g., learning object development, e-assessment subsystem, etc.). Fortunately, at the level of abstraction, these stakeholders perspectives of LTSA may possibly provide an alternative means for us to analyze and identify these priority functionalities of a specific subsystem or system though they can not provide any recommendations of the actual implementation at all. In annex 1, consequently, several related stakeholders from LTSA are enumerated for guiding the functional analysis at the abstraction design phase and more importantly, at the time when introducing multi-agent technology into e-Education system, they can facilitate the mapping of agent(s) to the corresponding abstraction level.

3.2 FRAMEWORK OF MULTI-AGENT E-EDUCATION SYSTEM (MAGE)

As described above, when accomplishing the generalized functional analysis at the abstraction layer (i.e., system components layer), the next step that we should take into consideration will be focused on the issues of further implementation. Namely, select some appropriate technologies as the supporting tools/how to map these "intelligent" agents to corresponding components compliant with LTSA and its stakeholder perspectives. Figure 3-4 is a recommended framework of multi-agent e-Education system.

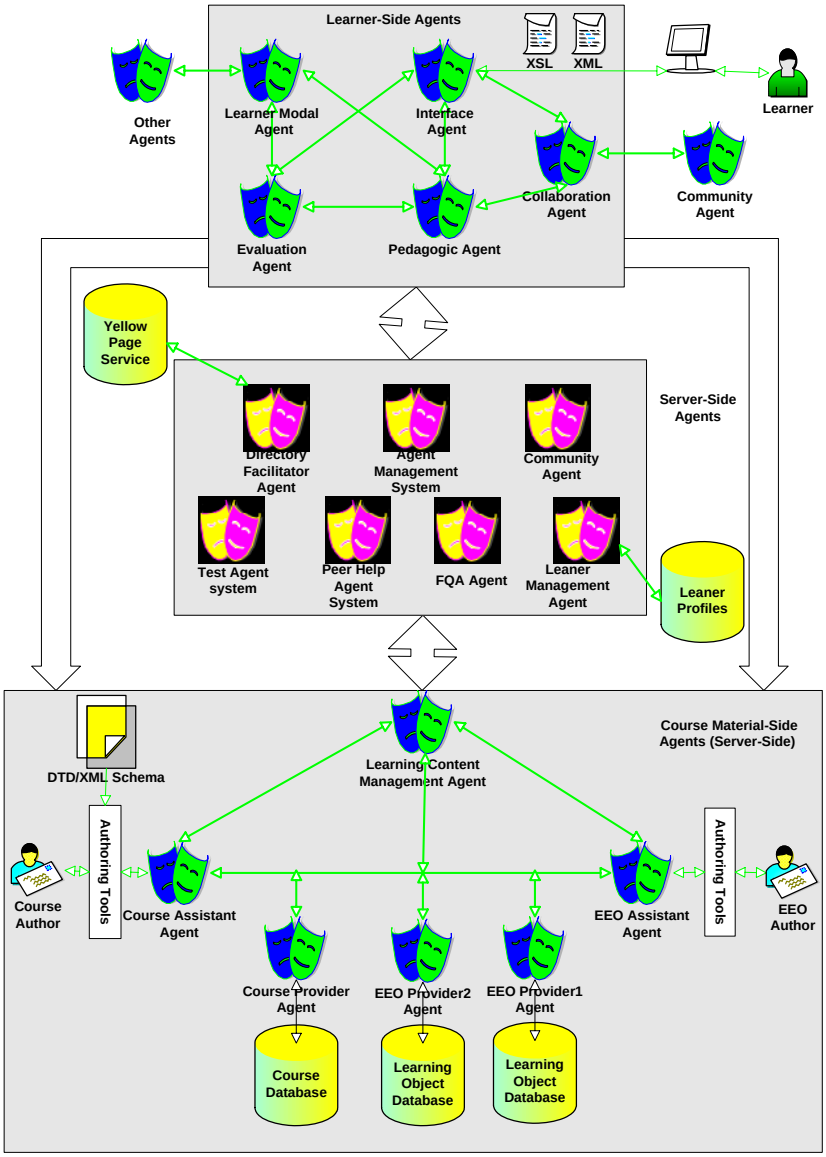


Fig. 3-4 Multi-agent e-Education system

3.2.1 CLASSIFICATIONS OF AGENTS IN MAGE

According to the functionality of agents In MAGE, we classify all the agents into three types: Learner-side agents, Server-side agents and Course content-side agents. Learner-side agents are responsible for the provision of high quality learning service to a specific learner; Server-side agents are responsible for the efficient management and maintenance of the whole system; Course content-side agents are responsible for the management, maintenance, authoring of course and learning object. All the agents in MAGE have a common runtime environment so that they can communicate, negotiate and cooperate in an efficient and effective manner.

3.2.1.1 LEARNER-SIDE AGENTS

- **Learner Assistant agent (LAA)**

LAA provides the learner with a personalized learning interface according to his/her preference and performance. Its main function is responsible for the interaction between a learner and MAGE so that it can actively react to the learner's request and present tailored learning content to learners.

- **Learner model Agent (LMA)**

LMA is responsible for the maintenance of the learner model including static and dynamic information that may be domain-dependent and domain-independent.

- **Learning evaluation Agent (LEA)**

LEA is responsible for the learner's performance assessment including pro-assessment, process-assessment and post-assessment during the whole learning process. All assessment information is used to update the learner's model or provide aid for PA to make appropriate learning strategy or learning path.

- **Pedagogic agent (PA)**

LPA takes charge of making tailored learning strategy and appropriate learning path, and so on.

- **Learning collaboration agent (LCA)**

LCA can help the learner create a collective learning environment so that a learning group can be formed.

- **FAQ agent (FAQA)**

FAQA is responsible for answering the learner's common questions.

- **Peer help Agent system(LHAS)**

This sub system is responsible for providing peer help.

- **E-assessment agent system (EAAS)**

This sub system is responsible for providing the whole services involved in the e-exam.

3.2.1.2 SERVER-SIDE AGENTS

Server-side agents can be classified into two types: System management agents and learning service agents. System management agents is responsible for the management and maintenance of MAGE itself while learning service agents provides learning services to client-side (learner-side) agents.

A). System Management Agents

System management agents are composed of AMS, RMA, DFA and ACC. All the agents will be designed as conforming to the Foundation for Intelligent Physical Association (FIPA) standard and specification.

- **Agent Management System (AMS)**

AMS provides a "white page" service/naming service (i.e. maintains a directory of agent identifiers (AID) and ensures that each agent in MAGE/System platform has a unique name). AMS also provides life-cycle service (i.e., manages the creation, deletion, suspension, resumption, authentication and migration of agents in MAGE platform)

- **Remote Monitoring Agent (RMA)**

RMA agent can offer a graphical interface to platform administration, this agent shows the state of the Agent Platform it belongs to (agents and agent containers) and offers various tools to request administrative action from the AMS agent and to debug and test the applications.

- **Directory Facilitator agent (DFA)**

DFA provides a "yellow page" service by means of which agents may register their services with the DFA and an agent can find other agents providing the services he requires in order to achieve his goals. In addition, a DFA can be allowed to federate with other DFAs and to control (i.e. register, deregister, modify and search for agent descriptions) all the network of federated DFAs.

- **Agent Communicative Channel (ACC)**

ACC is an agent that uses the information provided by the AMS to control all the exchange of messages within MAGE platform, namely, all the agents residing on MAGE

have a common runtime environment.

B). Learning Service Agents

- **FQA agent (FQAA)**

FQAA is responsible for handling the Question and Answer (Q&A) requests from learner-side FQA agents.

- **Community Agent (CA)**

A community agent is responsible for online learning community management and services. In MAGE, all the leaner-side collaboration agents are under the supervision of server-side CA. with CA, it makes possible for learners and teachers to create a collective learning environment.

- **Learner profile agent (LPA)**

Learner profile agent is responsible for the management and maintenance of the profile database of all the learners. When initiating, learner-side learner model agent is created in Applet within the browser and it can obtain its own learner model from LPA.

3.2.1.3 LEARNING RESOURCE-SIDE AGENTS

In general, learning resource-side agents also reside on server-side. However, the course interface agent, course assistant agent and EEO agent can move to or be downloaded on the client-side so that all the learning object authors and course authors can conveniently develop their EEOs or course at dispersed places. Note that since MAGE is peer-to-peer distributed system that provides a common environment, so the boundary of server-side and client-side has become somewhat blurry in many cases.

- **Course interface agent (CIA)**

CIA serves as a GUI of the course author. CIA provides a template-based course-authoring tool that facilitates developing consistent courses composed of EEOs, RIOs or “raw assets”. Besides, as an autonomous agent CIA has all the characteristics of an agent.

- **Course assistant agent (CAA)**

CAA helps course author perform many specific tasks such as searching existing EEOs, subscribing services (e.g., EEO) to LCMA, negotiating with other EEO authors, etc.

- **EEO interface agent (EEOIA)**

Like CIA, EEOIA provide a EEO development environment facilitating EEO authors to develop consistent EEO/RIO

- **EEO assistant agent (EEOAA)**

Like CAA, EEOAA perform many specific tasks such as searching existing EEOs, subscribing services (e.g., EEO) to LCMA, negotiating with other course authors, etc.

- **EEO provider agent (EEOPA)**

EEOPA is responsible for the communication with EEO/RIO/Raw asset repositories/databases

- **Course provider agent (CPA)**

CPA is responsible for the communication with course repositories/databases

- **Learning content management agent (LCMA)**

LCMA responsible for the management of courses and learning objects, and the maintenance of the XML-based metadata files, which include the metadata description of courses, EEOs, RIOs or Raw assets, etc. LCMA also provides the yellow page service. For example, when an EEO accomplished, it can be registered with LCMA agent so that Course author can find it.

3.3 LEARNING SCENARIOS

Based on MAGE, it makes possible for learners to learn in many learning scenarios such as intelligent tutoring system, teacher-intervened learning, collective learning, etc.

3.3.1 SCENARIO 1—AGENT ENABLED INTELLIGENT TUTORING SYSTEM (AITS)

In this scenario, AITS consists of several agent performing different tasks. By means of that, the learner can acquire personalized learning experience according to his/her preference. Since all of the functions of LTSA components can be executed by the corresponding agents, so AITS can be claimed fully compliant to IEEE LTSA. In fact, we can see that there exists an apparent mapping relation between AITS and IEEE LTSA from figure 3-5.

3.3.1.1 AGENTS INVOLVED

- Learning interface agent (LIA)
- Learning model agent (LMA)
- Evaluation agent (EA)
- Pedagogic agent (PA)

- Learning content management agent (LCMA)
- Course provider agent (CPA)
- EEO provider agent (EEOPA)

3.3.1.2 UML SEQUENCE SCHEMA

- Step 1: the learner states a learning objective to learning interface agent;
- Step 2: LIA checks the learning objective and send ACL message to pedagogic agent;
- Step 3: pedagogic agent request learning model agent for help;
- Step 4: learning model agent returns the learner's preference information;
- Step 5: pedagogic agent makes some analysis and inference according to the learner's profiles and predefined strategy rule and then forms a piece of specific service requirement ACL message which will transferred to learning content management agent;
- Step 6: learning content management agent will match its registered services and response with a list of names of course/EEO provider agents providing such service. If no match is available, learning content management agent returns null to pedagogic agent;
- Step 7: according to the list returned from LCMA, pedagogic agent requests the corresponding course/EEO providers agents to provide services
- Step 8: course/EEO provider agents returns all the required learning contents to learning interface agent
- Step 9: before presenting these learning contents to the learner, learning interface agent informs evaluation agent of the arrival of learning contents and sends relevant context to EA so that EA can monitor the learner's behaviors during the learning process;
- Step 10: learning interface agent present the learning materials to learner in the form that conforms to the learner's preference with the help of a specific XML style sheet;
- Step 11, step 13 and step 16 represented by three blue lines with double arrows indicate that the evaluation agent keeps monitoring the behaviors;
- Step 12, step 14 and step 15 represented by three lines with single arrow indicate that the evaluation agent updates the learner's model at an appropriate time;
- Step 17 implies that the evaluation agent considers that the learner cannot achieve the expected learning objective according certain assessment algorithm. So EA reports the learner's current learning performance to the pedagogic agent. In this case, PA may be readjust the learning strategy and make a new criteria for finding more tailored learning

contents for the learner;

- Step 18 →step 22 repeats the step 5→step 9, a new cycle will begin again.

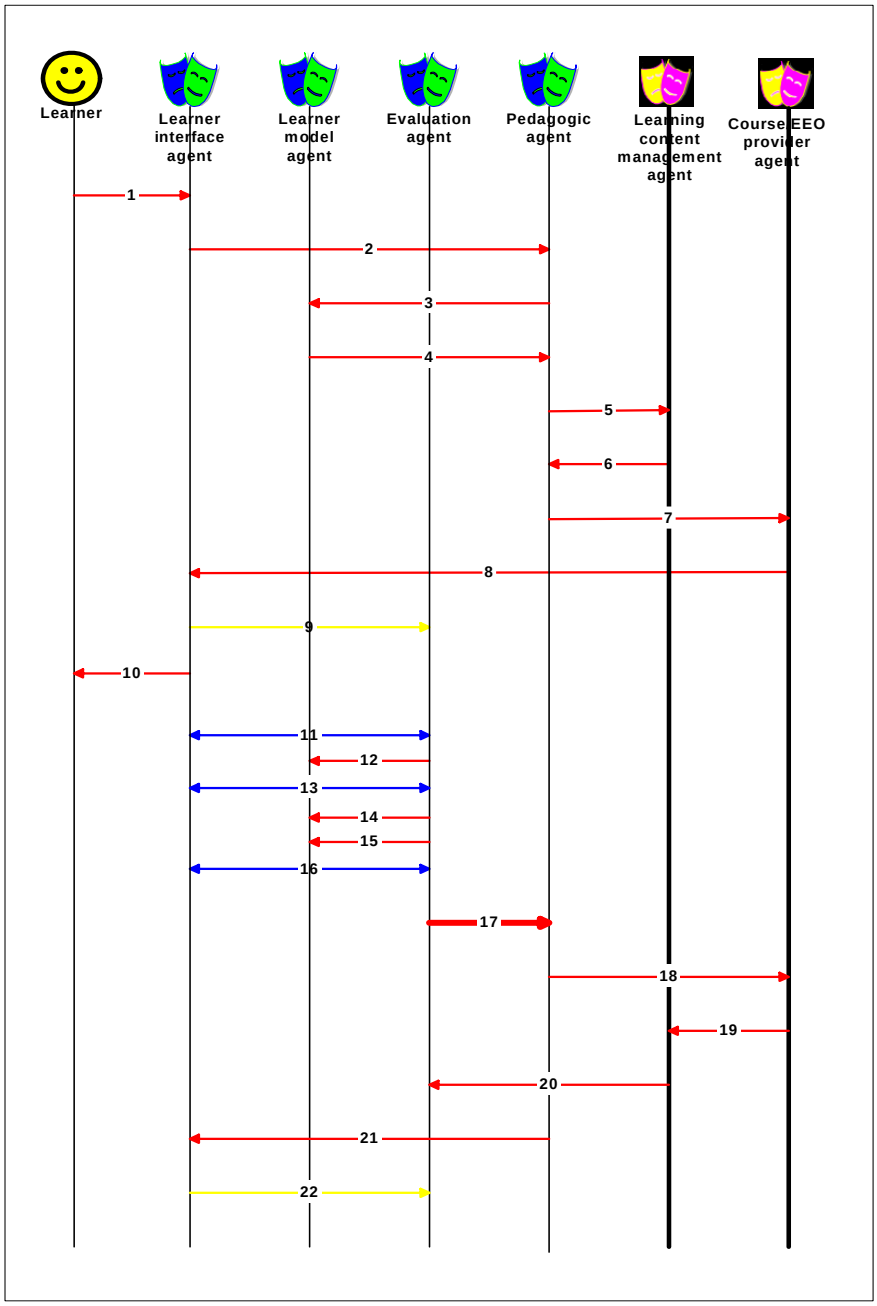


Fig.3-5 Intelligent tutoring UML sequence schema

3.3.2 LEARNING SCENARIO 2—TEACHER INTERVENED LEARNING

This scenario (see figure 3-6) provides an environment that supports the "face to face" communication between the learner and the teacher who can help so that they can discuss together some topic. When the teacher makes sense of the problem that takes place on the

learner, he can identify some learning task appropriate to the learner and find some tailored learning contents for the leaner.

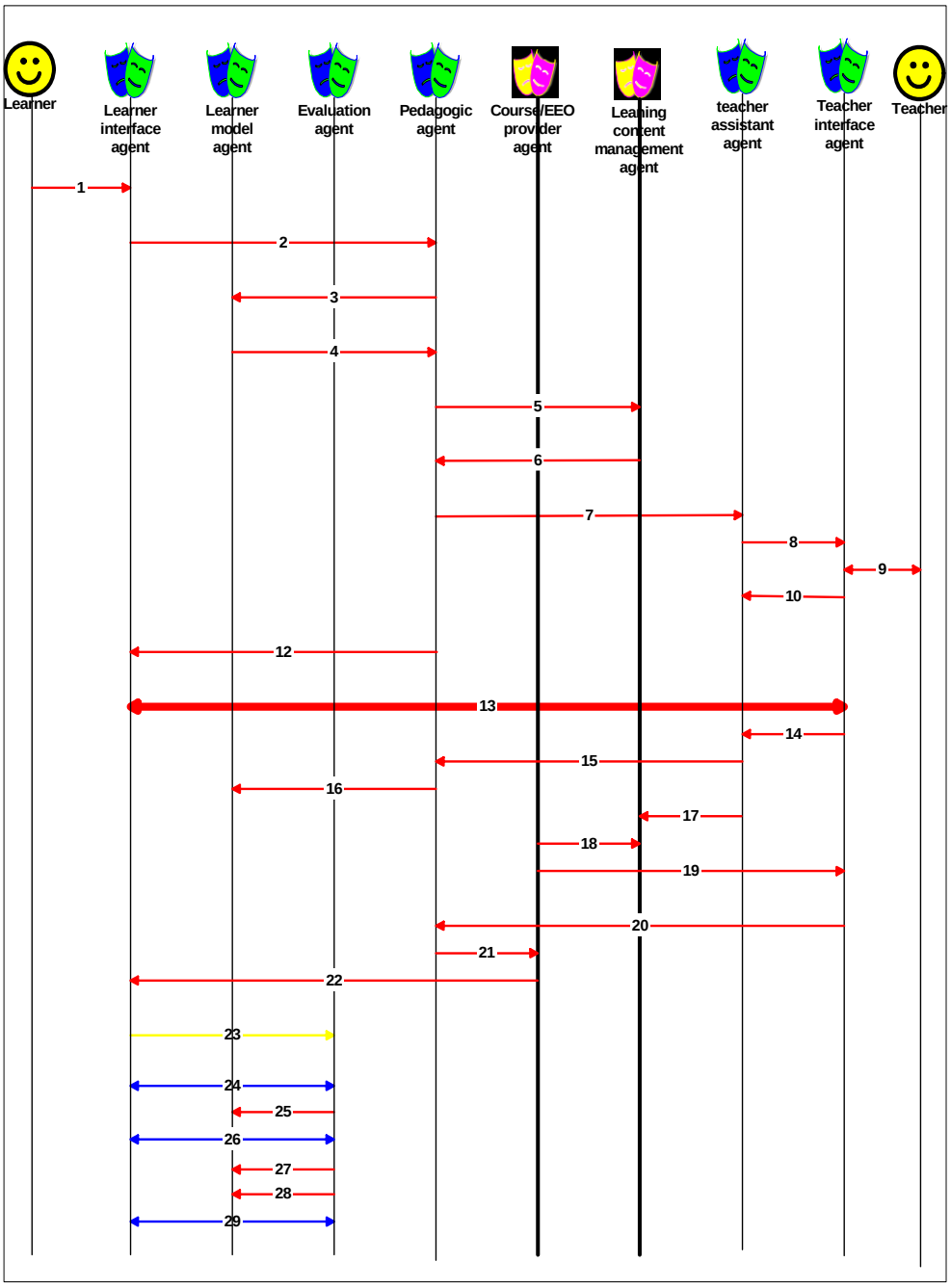


Fig. 3-6 Teacher intervened learning UML sequence schema

3.3.2.1 LEARNING SCENARIO 3—COLLECTIVE LEARNING BASED ON TASK

This learning scenario enables several learners to learn collectively and collaboratively. First the teacher assigns a specific research task to a leaning group, and then the learners in the

group discuss the subject together, identify their learning objective and sub task needed to be accomplished by each learner, and eventually elect a group leader so that he/she can contact and coordinate with the teacher on behalf of the learning group. In the collective learning context, the group learners can exchange views each other. When accomplishing the given task, the group leader will report the final results to the teacher. Figure 3-7 illustrates the process.

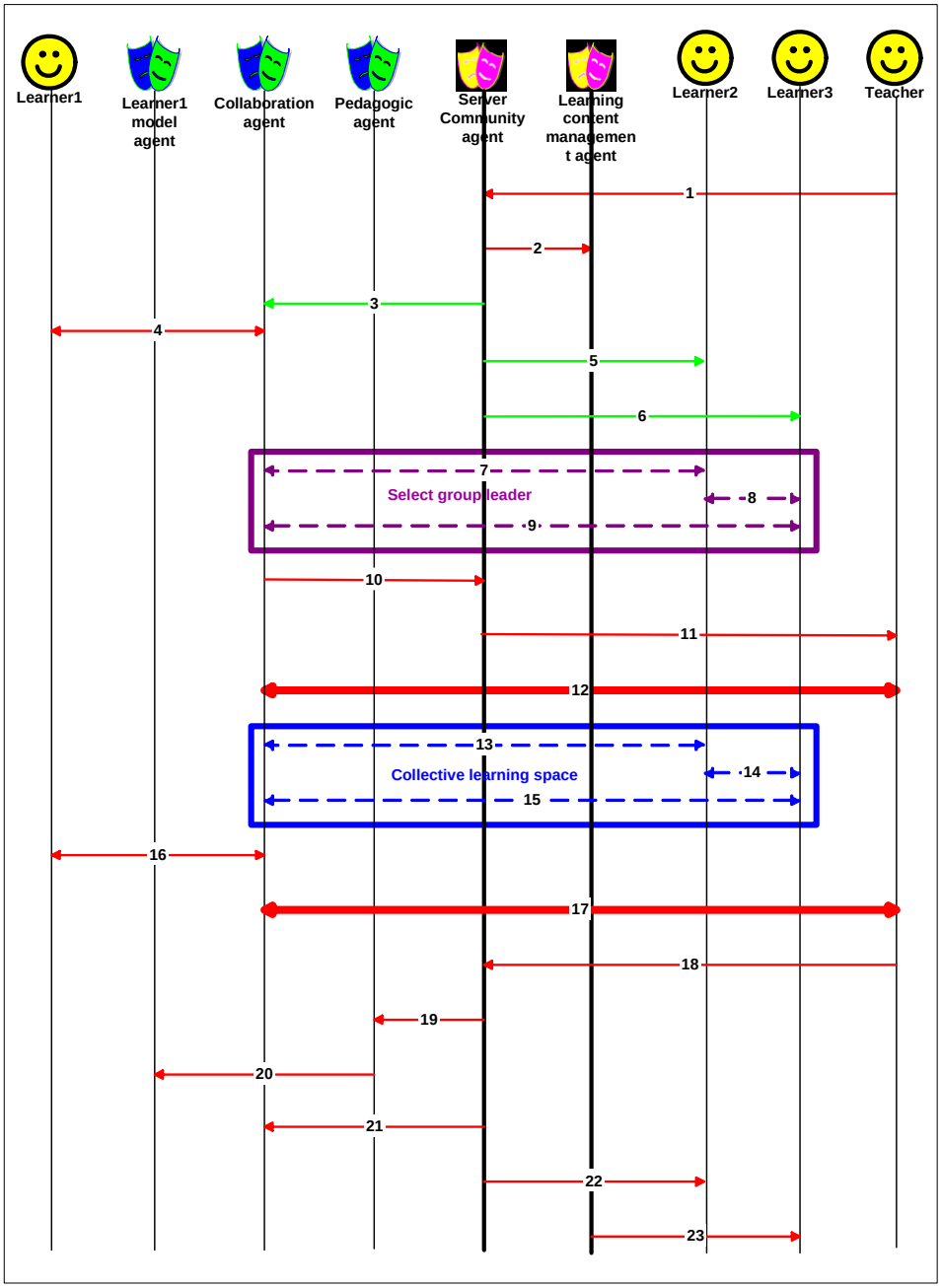


Fig. 3-7 Collective learning UML sequence schema

3.4 SUMMARY

This chapter proposed an agent enabled e-Education architecture, which can provide the learner with flexible learning styles and personalized learning process, as well as the course creator with facilitation of developing courses taking advantage of these existing EEOs defined in MAGE. Compared with the conventional e-learning system (CELS), MAGE has the following features:

- MAGE is a peer-to-peer distributed system, each agent in which acts as not only server but also client, and yet in CELS, client-server based paradigm predominates, consequently, there often exists the load balance problem on the server side.
- It is convenient for MAGE to personalize the learning process according to the learner's performance and preference thanks to the *anthropopathic* feature of agents, and yet CELS, in most cases, just offers numerous static web pages that lack the mechanism of tracking the learner's dynamic information.
- In MAGE, certain agents are designed as mobile agents that can not only reduce bandwidth usage but also allow the users to resume their work, even when the network is disconnected.
- MAGE has the capability to develop courses by assembling or decomposing numerous reusable EEOs or courses so that it is possible to build a self-improving and increasingly accumulated e-Education resource library, and yet CELS lacks the mechanism to reuse learning resources in other contexts.

Compared with other existing similar MAS based e-Education system, MAGE has the following advantages:

- MAGE is large-scale MAS based peer-to-peer e-Education system in which involve several group of agents or agent subsystems that speak specific agent communication language with particular ontologies. From the literature, it can be found that, in most MAS based e-Education system, there exist only single or few agents that are locally embedded in traditional client-server paradigm in order to improve certain module designed in the e-Education system. Therefore, to certain degree, this paradigm is a mixed network-computing paradigm. However, MAGE is completely designed as FIPA-compliant e-Education system although agents are purposely separated in client-side, server-side, and resource-side for the purpose of easier analysis.
- Besides taking into consideration the MAS technology itself, we also incorporate international learning standards, learning theory and some engineering view into the relevant agents. Instead, the other MAS based e-Education systems focuses more on the MAS technology itself and often ignore the pedagogic theory and learning standards.

CHAPTER 4 MAS BASED EEO & COURSE AUTHORING SYSTEM (MEEOCAS)

In this chapter, we first presented the philosophy of MEEOCAS in which all the participant roles are both consumers and contributors. Based on the principle mentioned above, we proposed the developing process of courses and learning objects by virtue of the conceptual model of MEEOCAS. It provides an efficient and powerful mechanism to facilitate corporation of software agents and learning object technologies. Secondly, for our purpose, a new definition of learning object—EEO, which stems from the idea of Object-Oriented Programming (OOP) and the model of Cisco's reusable learning object (RLO), was put forward. Furthermore, an XML based EEO packaging model was also described. Thirdly, the implementation model of MEEOCAS based on JADE (Java Agent DEvelopment Framework) platform fully complying with the FIPA specifications was presented. Relying on the common platform, all the involved agents can conveniently communicate, collaborate, and negotiate with each other, in order to perform some specified tasks using the common domain ontology and XML content language. Finally, some application scenarios based on UML schema, such as searching the appropriate EEOs, subscription to LMMA and the negotiation between course and EEO developers, were demonstrated. In conclusion, the aim of the MEEOCAS project is to make easy the development and deployment of learning contents and to build a self-improving and increasingly accumulated e-Education resource library.

4.1 INTRODUCTION

As is known that "content is king", therefore, as far as e-learning is concerned, one key issue is how to develop instructional materials of high quality that could be reused and applied to different contexts. Unfortunately, these instructional contents are, traditionally, expensive and time consuming to produce. In most situations, course authors have to create their new course from the scratch, even though numerous online instructional materials are conveniently accessible: course developers have to break them down into smaller constituent components at the beginning, and then modify or reassemble them in their own way in support of their individual instructional goals. Such repeated creating process, without doubt, is tedious and time-consuming. The more important point is the difficulty in

sharing and reusing of these courses even when they are available, how to share and reuse them is still a big problem. Since the traditional courses generally fixed in length, sequence, and scope, are built as such a large monolithic structure that it is difficult to re-purpose them into other contexts at such a course-grained level via Internet. In fact, this large and inflexible structure misses out on most of the benefits to authors and learners.

Fortunately, the recent emergence of learning object technology seems to be a promising solution to the problem due to the potential of its reusability, interoperability, adaptability, and scalability. However, another issue arises when taking into consideration how to apply these ready-made learning objects to course authoring system and deploy them to the learning process in an efficient and flexible manner. To address this issue, the agent technology has been introduced in MEEOCAS. The combination of learning object and agent technology makes it possible for MEEOCAS to facilitate the development of course and learning object itself, and furthermore, to personalize learner's learning process and patterns.

4.2 DESIGN PRINCIPL AND CONCEPT MODEL

When considering the design of MEEOCAS, to overcome the inflexible monolithic structure of traditional course and the low efficiency of creating process, as well as, to comply with the functional requirements of different occasions, our philosophy is: Learners, course materials designers and teachers are both consumers and producers when using MEEOCAS. That is to say, learners, course materials designers (i.e. learning object designers and course designers), teachers not only directly or indirectly benefit from MEEOCAS but also contribute to MEEOCAS and thus a self-improvement, and increasingly accumulated learning resource library can probably be worked up. Figure 4-1 is the concept model of MEEOCAS, in which, we introduced the well-known concept of learning object as supporting groundwork, which can be re-purposed for many use and we emphasize more on the performance support than simply an information system composed of learning objects. From learning activity-enabled perspective, learning object will play a key role in several aspects in MEEOCAS (see figure 4-1) as follows:

- Facilitating the development of courseware, thanks to the reusability, deliverability and discoverability of learning object, mainly through the support by Course Assistant Agent (CAA) in cooperation with both Learning Content Management Agent (LCMA) and EEO Assistant Agent (EEOAA).

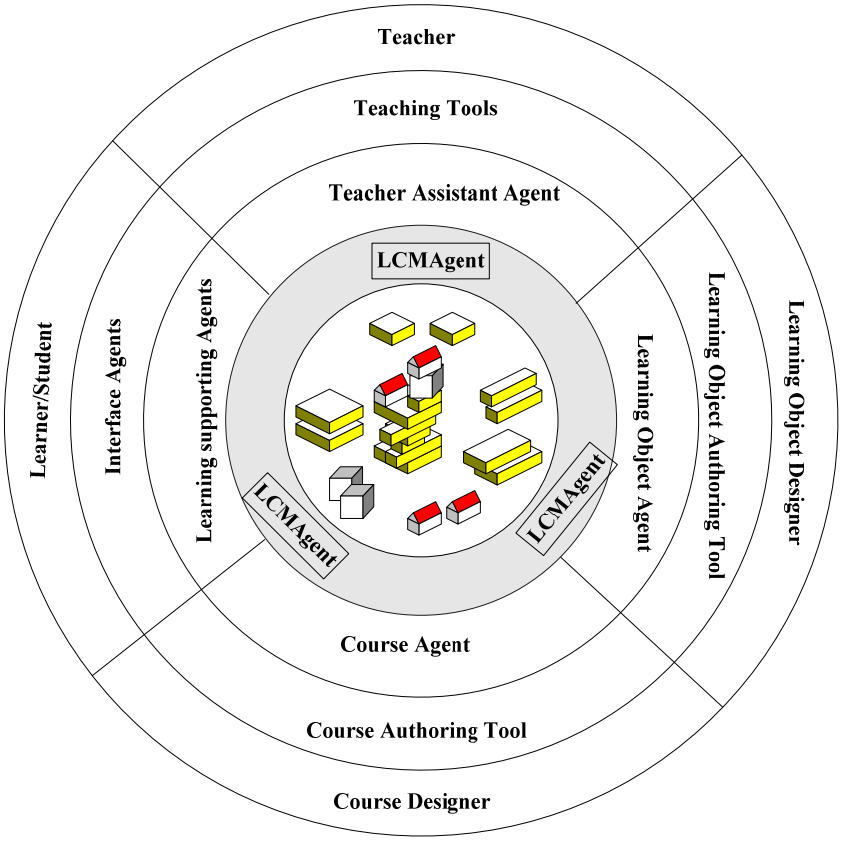


Fig. 4-1 Concept model of MEEOCAS

- Facilitating the development of learning object in virtue of its modifiability, inheritability, and discoverability etc. This process is chiefly assisted by OLAA and LMMA.
- Providing learners with personalized learning experiences and tailored learning services through intelligent navigation and dynamic learning paths modification according to their own learning performances and profiles in automatic tutoring mode, which is supported by the cooperation and coordination between Pedagogic Agent (PA), Learner Model Agent (LMA), and Evaluation Agent (EA) etc.
- Teachers can gain access to learning object libraries and make full use of them in teacher-centered learning mode with support from Teacher Assistant Agent (TAA).

4.3 LEARNING OBJECT DESIGN

4.3.1 DEFINITION LEARNING OBJECT

In the e-Education context, the emergence of learning object is assuredly exciting and encouraging. As [Wiley 2000] says Reusable Learning Objects (RLOs) are emerging as the

"technology of choice in the next generation of instructional design, development, and delivery, due to its potential for reusability, generativity, adaptability, and scalability." It is certain that having a library of learning objects to draw on will sharply shorten the course development time when allowing for faster deployment of the learning and personalize the learning process.

However, learning object is a relatively new term, thus we are not surprised to find numerous versions of different definitions of learning object: Learning Object Metadata (2001)-IEEE 1484.12.1 defines a learning object as "any entity, digital or not-digital, which can used, re-used or referenced during technology supported learning". Cisco (2001) defines a learning object as "a granular, reusable chunk of information that is media independent". Wiley (2001) defines learning objects as ''elements of a new type of computer-based instruction grounded in the object-oriented paradigm of computer science".

Besides various definitions of learning objects, there are also a large amount of terms relevant to it, like "Reusable Learning Object (RLO)", "Reusable Information Object (RIO)" (Cisco 2001), "Assignable unit", "Sharable content object (SCO)" (ADL SCORM 2001), "knowledge objects" (Merrill, Li, &Jones 1991), "online learning materials" (MERLOT 2000), "educational software components" (ESCOT 2001) and etc. This dissertation uses the term E-Education Object (EEO) to describe its purpose and functionalities. Here gives the definition of EEO conforming to the functionalities requirement of learning object in MEEOCAS:

An E-Education Object (EEO) is a reusable, modifiable, scalable, inheritable, polymorphous and multipurpose component that encapsulates well-organized "raw assets" (i.e. contents, practices and assessments) as well as a common interface attached to it.

This definition is derived from some concepts of object-oriented programming, and also references to the models of ADL's SCORM and Cisco's RLO. It is found to be an effective way to describe and construct an EEO. The more detailed interpretation is given as follows:

- Reusability indicates that an EEO can be reused in certain context at random times without any modification.
- Inheritability implies that an EEO can be entirely or partially (contents, practices or assessments) inherited by other ones. By this means, a wonderful experience of creating a new EEO applied to other contexts can be gained.
- Modifiability denotes that an EEO can be modified and then form a new EEO, e.g. when an EEO becomes out of date or not appropriate for most learners, this EEO should be updated, modified or even deleted.

- Scalability refers to the granularity of a learning object that can range from as small as a section to as large as a lesson. (Note: in MEEOCASS, the hierarchy of a course is like this: course->unit->lesson->section. In principal, the largest learning object is constrained within a lesson consisting of units, several of lessons constitute a course intended to deliver in MEEOCAS to learners for accomplishing their knowledge, skill, competence etc.)
- Multipurpose implies that an EEO can be applied to several contexts. E.g. in MEEOCAS, an EEO can be used by several actors/participants such as learners, teachers, EEO designers, Course designers and related agents.
- Polymorphous implies that the same subject matter may possibly own several versions of representing forms, e.g. simulations, demonstration, experiments, animations, html, text, video clip and games etc, which are likely to point to a single learning objective in order to accommodate to different learning style.
- EEO is a structural component that consists of well-organized "raw assets" such as content, practice and assessment.
- EEO has a common interface, which makes it possible to be accessed, discovered and connected to the outside world.

Of course, the issue of intellectual property has to be taken into account when an EEO needs to be modified or reused by others rather than the original creator, an imaginable approach to this problem is to get the permission of the original copyright holder or else pay for the reuse of it.

4.3.2 STRUCTURE MODEL OF LEARNING OBJECT

Firstly, we introduce and discuss an influential model —Cisco Reusable Learning Object (RLO) Model— and then present a more flexible and practical structural model of EEO on the basis of it.

As a worldwide leader in networking for the Internet and one of the forerunners in learning object design, creation, and deployment, Cisco is also actively participating in standards groups such as IMS and ADL, whose RLO strategy has been attracting extensive attention all the while. As such, it is worth looking at their RLO structure. The Cisco RLO is created by combining an overview, a summary, and from five to nine (7±2) Reusable Information Objects (RIO) (see figure 4-2). Each RIO is built upon a single objective. Several RIOs are combined together to create a Reusable Learning Object (RLO). If a RIO can be equated with an individual component of a learning objective, a RLO is the sum of RIOs needed to fulfill that objective. To aid in content standardization, each RIO is further

classified as concept, fact, procedure, process, or principle. Each of these RIO types has a recommended template that authors can follow to build the RIO.

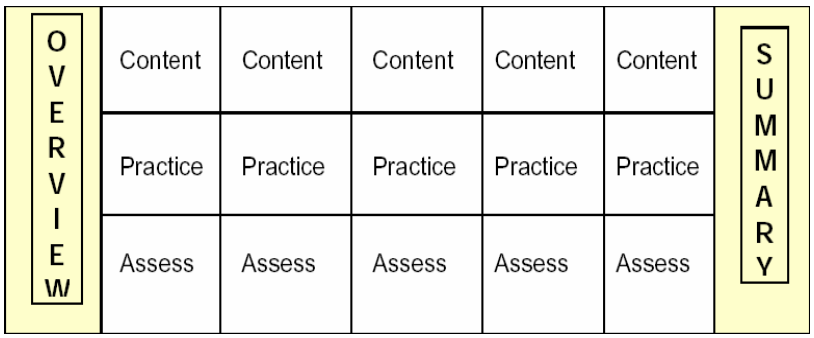


Fig. 4-2 Cisco's RLO and RIO (Note: each column represents a single RIO)

As above-mentioned, the reusability of Cisco's RLO model occurs at the level of RIO. Its internal elements (i.e. content items, practices items and assessment) cannot be reused in other contexts. Consequently, it is inconvenient and inflexible in the situation in which we only need reuse or inherit part of a RIO. According to ADL SCORM, any deliverable "raw medias", such as illustrations, documents or media streams, can be seen as "assets" ready to be reused in other "content object". Although a single "asset" cannot be used as a learning object alone, multiple learning objects can reuse these assets for gaining their objective. Consequently, from our standpoint, each content, practice or assessment in Cisco's RIO is also a valuable asset which seems to be a more "meaningful chunk" than "raw media" mentioned in ADL SCORM.

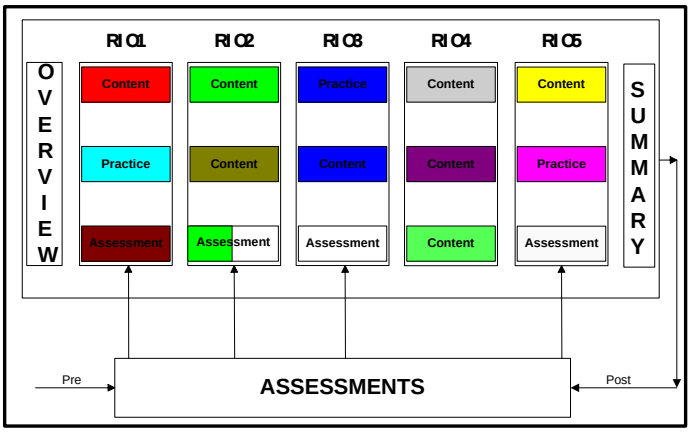


Fig. 4-3 Recommended E-Education Object (EEO) Model

From figure 4-3, we can see that there are mainly two differences between the Cisco RLO model and the recommended EEO model: Firstly, the recommended model still contains several RIOs, but the structure of each RIO included in an EEO is not exactly the same. In Cisco RLO model, and yet all RIOs are as like as two peas, i.e. the internal structure and sequence (content-> practice->assessment) of each RIO is unchangeable. In recommended EEO model, however, the inside of different RIOs may manifest different

structures and sequences. An example is that, in RIO3, the component “practice” can possibly appear before the component “content” for some specific need. In another example, RIO4 may only contain the elements “content”. The second and also the most different point is that in the recommended EEO model, all the “assets” such as contents, practices or assessments in an EEO can be entirely or partially reused /inherited by other EEO. To be more exactly, all the boxes of different colors representing different assets in figure 3 can possibly be reused in other contexts. An example is used here to illustrate this idea. In Figure 4, assuming that an EEO designer is creating a new EEO3, like a course designer does, it is not necessary for him/her to develop this EEO from the scratch since there is a library of existing EEOs “waiting” for reuse. Therefore, the creation of EEO3 is relatively an easy and comforting process.

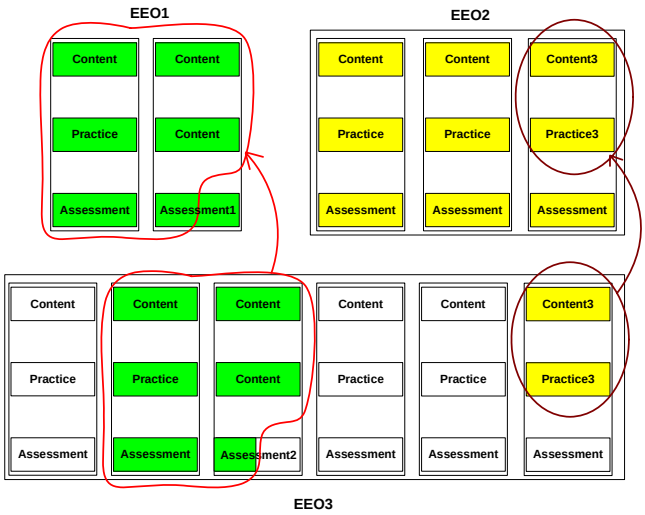


Fig. 4-4 is an example of demonstrating how to inherit assets from existing EEOs.

As you see in figure 4-4, the accomplished EEO3 has inherited content3 and practice3 from EEO2 and furthermore almost all the assets in EEO1 with the exception of component “Assessment1” that has modified into Assessment2 in EEO3. Of course, the remainder represented by white boxes is created by the EEO designer himself/herself.

Using the recommended EEO model can bring us many benefits: On the one hand, it provides more flexibility and greater return on investment from the perspective of reusability and commerce. For example, in learning object-oriented learning pattern, a learner can make use of an EEO as a stand-alone performance support tool. The EEO gives learners the learning context, knowledge and skills needed to perform the given objective, and a method to assess mastery. EEOs and RIOs can also appear as offerings on a “road map” that is customized to the needs of each learner. Learners can see from this road map what they need to take, what they have completed, and what their learning destination is.

On the other hand, we can adequately "borrow" from Cisco other advanced ideas in building a learning object, such as its strategies of taxonomy and sequencing. What excites us most is that, based on our multi-agent system, a flexible, personalized and dynamical e-Education environment can be formed because of the "participation" of EEO.

4.3.3 EXTENSION OF LO METADATA TO ENHANCE ADAPTIVIEY

Table 4-1: Metadata table of a learning object

General	**Technical**	**Educational**	**Management**	**Relation**
Identifier	Format	Pedagogic Type	Description	Kind
Title	Size	Interactivity	Name of author	Identifier of
Language	Location	Type	Cost	resource
Description	Requirements	Interactivity	Restriction	Description
Keywords	Type	Level		
Domain	Name	Semantic		
Structure	Maximum	density	**Life Cycle**	**Annotation**
Aggregation Level	Version	Educational	Version	Person
	Minimum	Context	Status	Date
Meta-metadata	Version	Duration	Contribute	Description
Identifier	Duration	Difficulty Level		
Contribute		Age Range		
Scheme metadata		Learning Time		
Language		Use Description		

4.3.4 PACKAGE MODEL

After an EEO is well accomplished, it has to be packaged and stored into database so that it can be reusable and accessible. When considering encapsulating an EEO into the package, it is a good practice for us to comply with some international specifications or standards (e.g. the specifications and standards of IMS, IEEE and SCORM). In the package, the key is to clearly describe the structure of an EEO and its corresponding resource. Fortunately, metadata has this ability to provide a common means to describe things so that EEOs can be self-describing and can be searched, found and applied to a specific context. Here gives a recommended package model of EEO (see figure 4-5), which includes: manifest XML, EEO metadata XML, RIO metadata XML, Raw asset metadata XML, Physical files and Package interchange file (PIF). Their functions are described as follows:

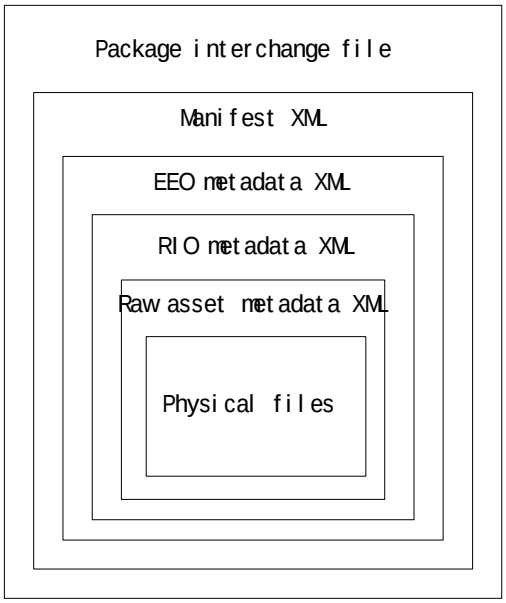


Fig. 4-5 A recommended EEO package model

- PIF: it can be a zip file or other archive format allowing for standalone external use.
- Manifest XML file: it describes the file list needed when using the EEO.
- EEO metadata XML file: it describes the structure and other metadata information such as General, Lifecycle, Metadata, Technical, Education, Relation and Classification etc.
- RIO metadata XML files: it describes the content construct.
- Raw asset files: it describes itself.
- Physical files: The physical files are the actual resource files.

4.4 ARCHITECTURE STRUCTURE

As mentioned above, it is sure that having a library of read-made EEOs to draw on will sharply shorten the course developing time. However, when allowing for how to make them to be deployed conveniently in a system and to be found easier by EEO and course developers, obviously, we still need a common communication environment to support it. Naturally, we introduce multi-agent technology to realize our system—MEEOCAS, which is based on Java Agent Development Environment (JADE 2003). JADE is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with FIPA specifications and through a set of tools that support the debugging and deployment phase. The agent platform can be distributed across network and the configuration can be controlled via a remote GUI. According to our experience of using JADE, it justifies this feasibility of

building a flexible and powerful MEEOCAS (see figure 4-6) with JADE.

From figure 4-6, we can see that learning material-side agents reside on server-side. However, the course interface agent, course assistant agent, EEO interface agent and EEO assistant agent can move to or be downloaded on the client-side so that all the EEO and course authors can conveniently develop their EEOs or courses at dispersed places. It is worth of note that MEEOCAS is a peer-to-peer distributed system that provides a common channel for communication, so the boundary of server side and client side has become somewhat blurry in most cases. The following is their functionality description of all involved agent in MEEOCAS and their further applications will be presented in the next section.

- *Course interface agent (CIA)*

CIA serves as a GUI of the course author. CIA provides a template-based course-authoring tool that facilitates developing consistent courses composed of EEOs. Besides, as an autonomous agent, CIA has all the characteristics of an agent.

- *Course assistant agent (CAA)*

CAA helps course author perform many specific tasks such as searching existing EEOs, subscribing services (e.g., EEO) to LCMA, negotiating with other EEO authors, etc.

- *EEO interface agent (EEOIA)*

Like CIA, EEOIA provides an EEO development environment facilitating EEO authors to develop consistent EEO/RIO

- *EEO assistant agent (EEOAA)*

Like CAA, EEOAA performs many specific tasks such as searching existing EEOs, subscribing services (e.g., EEO) to LCMA, negotiating with other course authors, etc.

- *EEO provider agent (EEOPA)*

EEOPA is responsible for the communication with EEO repositories/databases

- *Course provider agent (CPA)*

CPA is responsible for the communication with course repositories/databases

- *Learning material management agent (LCMA)*

LCMA is responsible for the management of life cycle of all involved agent in MEEOCAS such as the creation, deletion, suspension, resumption, authentication and migration of agents. LCMA also provides the yellow page service. For example, when an EEO accomplished, it can be registered with LCMA agent so that course author can find it.

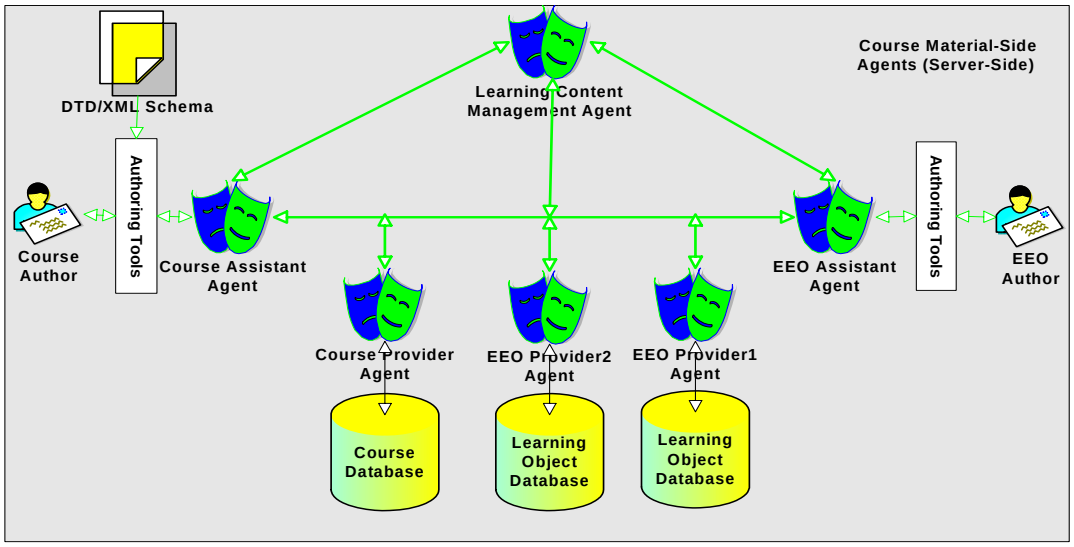


Fig. 4-6 Architecture of agent enabled course-authoring model based on learning object

4.5 COURSE AUTHORING SCENARIOS

4.5.1 SERCHING LEARNING OBJECTS

With MEEOCAS, several application scenarios can be easily applied, such as searching the appropriate EEOs, the subscription to LCMA and the negotiation between course and EEO developers etc. To save spaces, only one UML sequence schema, used to demonstrate the conversation protocols when searching an EEO, is given here. Figure 4-7 shows that course developers can search the EEOs that he/she prefers and subscribe to the LCMA when the search fails. These agents involved include CIA, CAA, LCMA, EEOPA and Federal LCMA. The detailed interaction steps are shown below.

- **Step 1:** course author states desired EEO as a goal to course interface agent; CIA may provide a particular form consisting of learning object metadata (LOM) to facilitate the course developer to customize his/her desired EEO (e.g., if the course developer needs such an EEO as Context='continuous formation', Interactive Level='middle', Difficulty='high', Keyword='agent, communication', Language='French', etc., s/he may fill out the ready-made form compliant to the IEEE LOM standard (2002) to customize his/her required EEO).
- **Step 2:** CIA sends an ACL message to CAA (note: the ontology of the ACL message is called 'EEO LOM Ontology' in MEEOCAS, which, in fact, is the LOM schema that represents the element concepts and their relations of LOM, and its content codec language may be XML or SL language).
- **Step 3:** CAA responses to CIA, which refers to the acceptance of CIA's request

- **Step 4:** CAA asks for help from LCMA that can identify whether there exist EEO provider agents providing such kind of service as described in the ACL message of CAA (note: in MEEOCAS, this CAA message applies a template to searching such service). In MEEOCAS, there maybe exist several LCMA agents that compose the federation enabling the flexibility and distribution data storage. As such, when a local LCMA cannot find appropriate EEO providers, it can deliver the CAA's request to other federal LCMAs for help.
- **Step 5:** LCMA returns a list of names of agents that match the template, if no match is satisfied, LCMA sends a message inquiring whether CAA is willing to search further.
- **Step 6:** CAA agrees with LCMA.
- **Step 7:** LCMA asks for help from the federal LMCA agents
- **Step 8:** Implying that the federal LMCA agent has found the intended EEO provider.
- **Step 9:** According to the returned list of provider agents from federal LCMA, CAA sends corresponding ACL messages to all of the EEO provider agents, which first match the request with their own EEO metadata info. If these matches are satisfied, the locators (e.g., a URL or URI pointing to the desired EEO) will be sent to CAA. Or else, failed message may be presented to the course developer and CIA may inquire further whether the course developer is willing to modify his/her form or subscribe to such service from LCMA which will automatically notify CAA as soon as such service is available in MEEOCAS.
- **Step 10 and step 11:** Course assistant informs CIA to present the desired EEO with a XML type sheet to which the course developer prefers.
- **Step 12:** Once having accomplished a courseware by taking advantage of the template-based course authoring tools, course developer can store his/her works into the course repository. This process is performed through **step 13 and step 14.**
- **Step 15:** When a new course is stored into the course repository/database, course provider agent will register the new service to LCMA agent.
- **Step 16 and step 17:** Return the message of success registration.

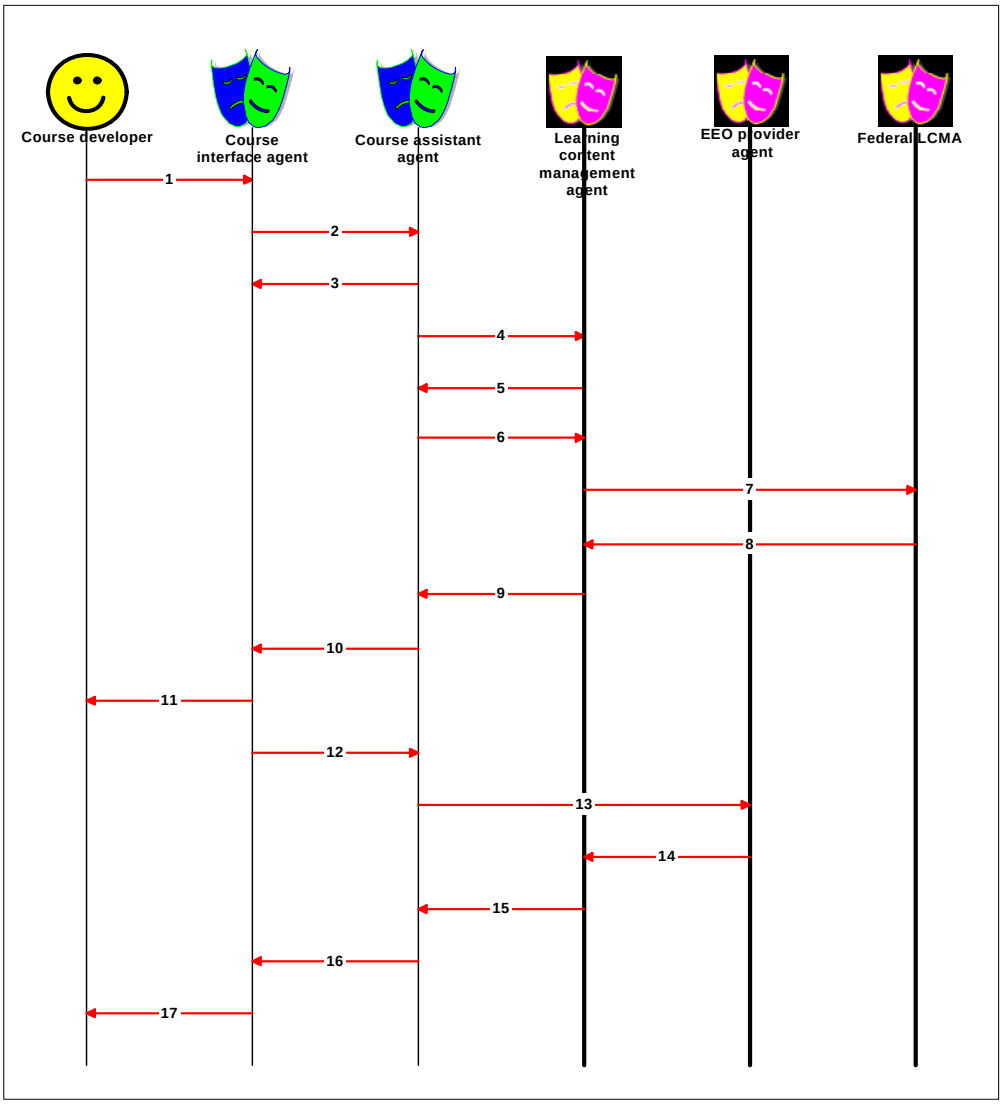


Fig. 4-7 UML sequence schema of searching target EEO

4.5.2 SUBSCRIPTION

Figure 4-8 aims at illustrating how to subscribe to LCMA.

Agents involved:

The agents involved in scenario 2 are similar to scenario 1.

UML sequence schema

The distinction between scenario 1 and scenario 2 is that when the local LCMA and its federal LCMAs all have not registered services conforming to the request. Through from step 11 to step 14, the subscription task can be performed. As such, once the subscribed service is available in MAEES, LCMA will automatically notify the learner's course assistant agent. This process is illustrated from step 15 to step 22.

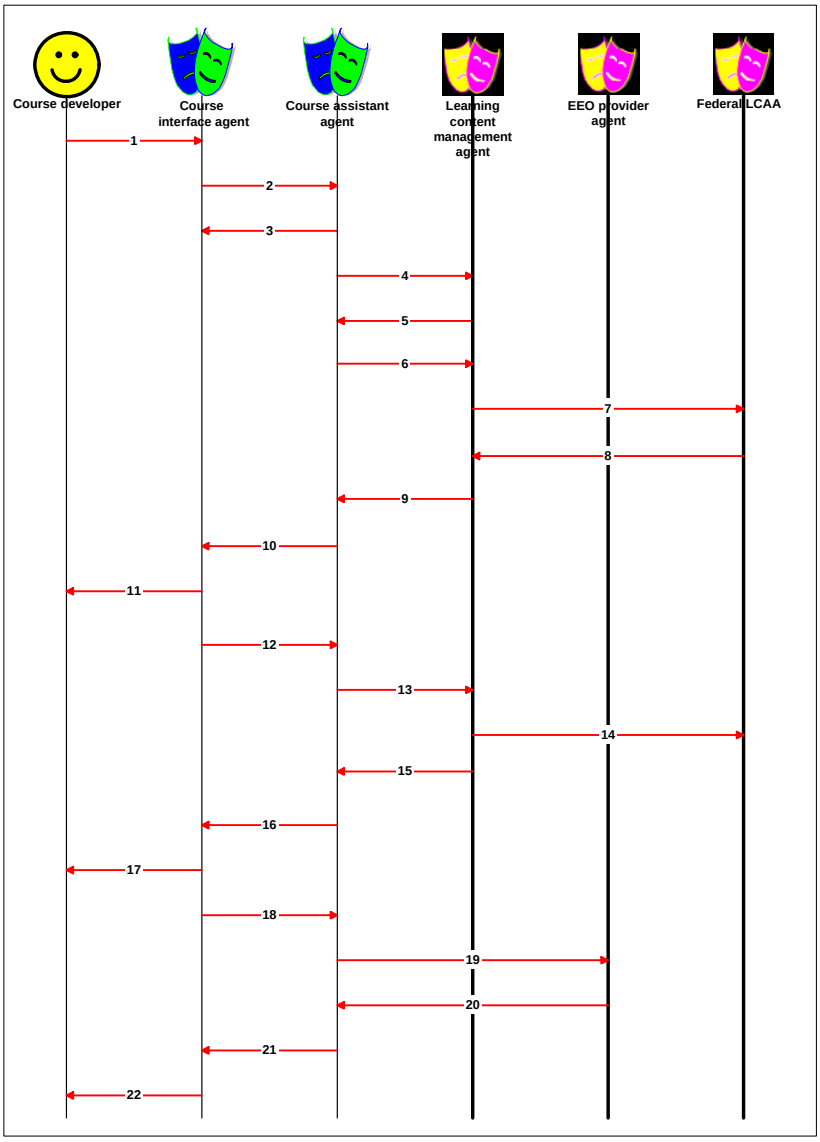


Fig. 4-8 Subscription to LCMA and the federal LCMA

4.5.3 NEGOCIATON WITH LO CREATORS

Figure 4-9 aims at illustrating how to negotiate between course and EEO authors.

Agents involved:

Course interface agent

Course assistant agent

Learning content management agent

OOE assistant agent

EEO interface agent

UML sequence schema:

- Step1 → step 5 represent the process of searching a EEO developer who provides

the EEO that the course developer is interested in

- Step 6 → step 13 represent the connection process between course and EEO developer. In figure 12, the EEO developer agrees to accept the negotiation with course developer.
- Step 14: the bold red lines with double arrows represents the actual negotiation process is taking place.

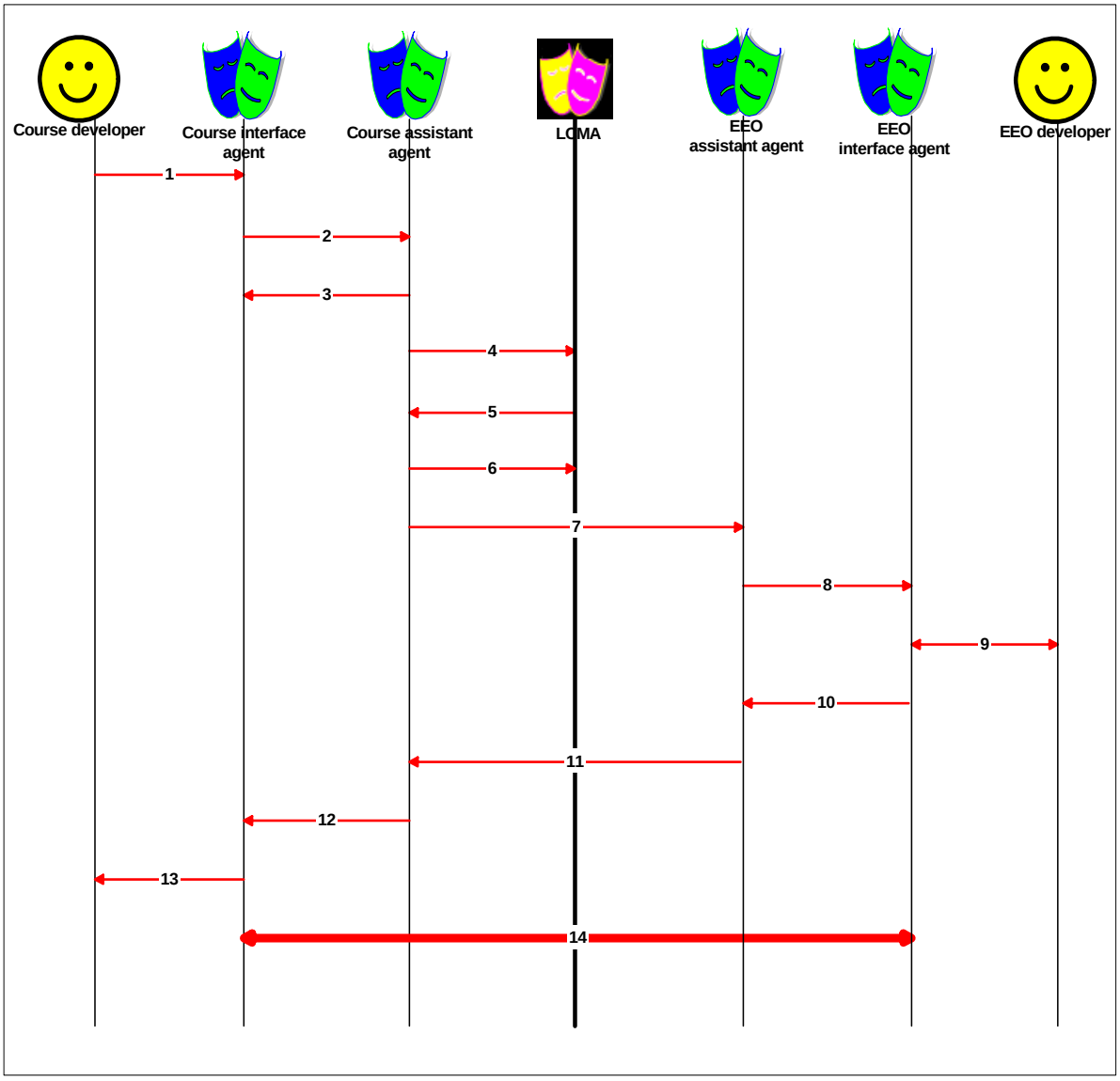


Fig. 4-9 Negotiation between EEO and Course developer

4.6 SUMMARY

This chapter presented the architecture of a multi-agent based learning object and course authoring system. With the introduction of EEO into MEEOCAS, we can find out

that it brings many advantages as follows:

- Course designers may conveniently develop their courses through assembling the ready-made EEOs instead of creating them from the scratch. It is sure that, when contributing courses to MEEOCAS, they could also make full use of the existing courses, which contain well-organized and well-tagged EEOs provided by MEEOCAS.
- From the standpoint of EEO designers, it is convenient for them to create EEOs adapting to corresponding templates in accordance with an appropriate taxonomy, and to update the old version of EEOs according to the feedbacks from all the other actors (e.g. teachers, course designers, learners or the agents in MEEOCAS). In addition, EEO designers can also formulate different EEOs satisfying diversified levels of learning objectives by inheriting, modifying and reassembling the existing EEOs.
- From the perspective of teachers, they could take advantage of these ready-made EEOs and courses to organize their teaching process or offer learners some recommendations of useful learning recourses during their interactive pedagogic activities.
- As far as learners are concerned, in MEEOCAS-enabled environment, they may choose among several available learning patterns (e.g. course-oriented learning, EEO-oriented learning, self-paced learning, teacher-centered learning or collective learning pattern etc.). In converse, the feedbacks from learners are also important references for course and EEO designer to modify and improve their works.

Despite these potential advantages of EEO, it is still difficult to make them function without a flexible supporting environment. Just because of the incorporation of agent technology in MEEOCAS, which makes possible to provide a common channel for the communication, cooperation and negotiation among all the agents. As such, it is convenient to deliver and handle EEOs throughout MEEOCAS with the help of relevant agents. For MEEOCAS, it not only facilitates the development of courses and EEOs, but also, when MEEOCAS to the whole e-Education system, makes easy to personalize the learner's learning process and diversify their learning styles because of the wonderful features of EEO and agent. With the advancement of multi-agent and Internet technology, it is believed that similar systems dedicating to the development of e-Education resource library will surely be emerged more and more.

CHAPTER 5 MAS BASED ADAPTIVE & ACTIVE LEARNING FRAMEWORK

This chapter proposed a MAS based integrated framework in support of adaptive and active learning in both individual and collective learning spaces. In the adaptive individual space, the key issue is how to dynamically generate personalized learning path consisting of domain concepts and present associated learning objects catering for a learner's knowledge state and learning preference. As to this, this chapter put forward an efficient searching algorithm for the presentation generation based on the proposed domain ontology model. In the collective learning space, our focus is on the issue how to find appropriate help resources (e.g. peer learners, learning materials, or other applications) and how to dynamically build a tailored learning group on behalf of learners in a distributed network according to their need. In this regard, this chapter proposed two corresponding architectures: One is the peer help system; another is architecture of the learning group forming system.

5.1 INTRODUCTION

As far as personalization and adaptation is concerned, we have to consider such issues as how to dynamically generate learning path and present tailored learning objects (EEO) catering for a learner's knowledge state and learning preference; how to find appropriate help resources (e.g. peer learners, learning materials, or other applications) for a learner when s/he encounters difficulty in learning certain domain concept or topic and how to build collective learning environment in support of constructivist learning, e.g. a learner can take the initiative to construct a desired learning group for her own particular purpose.

To achieve the above goals, we have to carefully design the domain model which are absolutely necessary modules for an adaptive learning system. The issue of how to design and implement them is obviously dependent on the given domain and specific application. Thus, what we contribute in these regards mainly consists in the proposition of some generic modeling method, strategy and process rather than specific ones.

As far as adaptivity is concerned, MAGE provides learners with several adaptive learning experiences available in both individual and collective learning spaces as shown in figure 5-1. the distinct feature that is different than other adaptive learning systems is that we distinguish between two types of adaptive mechanisms: **individual adaptive learning**

(IAL)and **collective adaptive learning** (CAL). In MAGE, the IAL indicates the situation in which the system helps to choose adaptive learning path consisting of domain concepts and associated learning objects while the CAL means that the system searches for appropriate peer learners in distributed network for the purpose of either facilitating the help session or the collective learning by dynamically building learning groups according to the learner's preference. It is worth noting that these adaptive process can be controlled by the learner himself or herself. That means it is the learner who decides to whether or not accept the adaptive recommendations. For instance, for advanced learners, they are inclined to navigate through the concept map (structure of a course) by their own desired ways while the novice learners prefer to be guided by the system as they have few knowledge about the course to be learned.

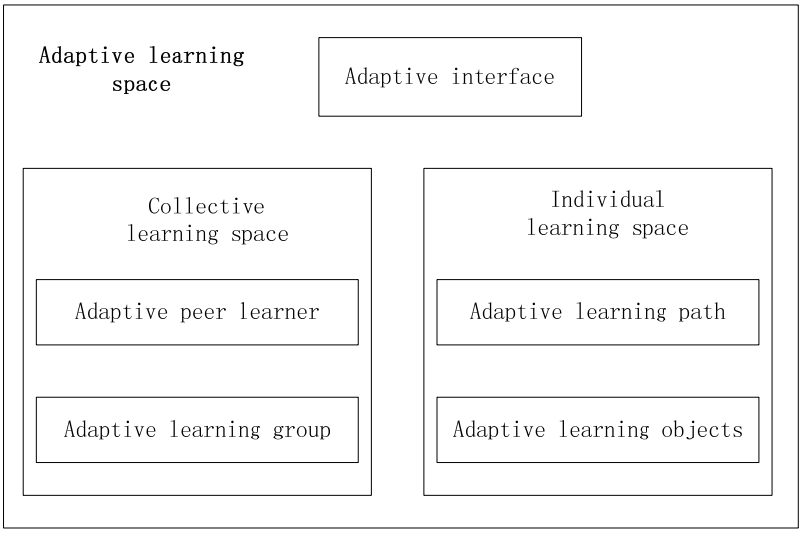


Fig. 5-1. Adaptive learning architecture

Obviously, this situation will probably change with the development of a learner's learning process. From the social-constructivist perspective, online collective learning has important signification and influence on the development of social ability, human interrelationship and learning motivation for active engagement, especially for virtual learning environment without the chance of face-to-face contacts between tutors and learners. To enhance such learning experience, we put forward a adaptive peer help model and a adaptive collaborative learning model on the proposed domain model basis. From my personal point of view, they have important contribution toward the constructivist learning. At length, we summarize these adaptive considerations in table 5-1.

Table 5-1: Category of adaptive strategy recommended in MAGE

Aspects of adaptivity	Main agents involved	Explanation
Adaptive interface	**Leaner model agent Learner assistant agent**	The system allows learners to choose the preferred user interface or content display style (e.g. font, color, style etc.)
Adaptive learning path	**Leaner model agent Pedagogic Agent LCMAgent Learner assistant agent**	The system can automatically recommend learning path consisting of abstract domain concepts, this adaptation is realized according to the learner's knowledge status
Adaptive learning objects	**Leaner model agent Pedagogic Agent LCMAgent**	The system can select adaptive learning objects in a set of candidate EEOs associated with a unique concept in support of personalized learning. We achieve this goal considering both the learner's learning style as well as his/her cognitive ability
Adaptive peer learners	**Helper Agent Matchmaker Agent DFAgent Learner assistant agent**	The subsystem can search for tailored peer learners according to a learner's preference. This adaptativity is obviously constructivism-oriented, and it has important significance in terms of supporting the development of social ability, self-reflection and meta knowledge in distributed online learning environment.
Adaptive learning group	**Collaborative Agent Broker Agent Learner assistant agent**	The subsystem can dynamically search and form a learning group matching his preference, and this system support the knowledge internalization, externalization, socialization, and combination from the Knowledge Management perspective

From table 5-1, we can see that these adaptive strategies are respectively achieved by cooperation among a set of agents with different goals and behaviors. Consequently, we give attention to not only the models themselves not also the modeling process.

5.2 DOMAIN MODELING

The representation of domain knowledge is a key starting point to implementing flexible and adaptive mechanism. Literatures show that most of the existing online learning support systems represent the tree-like domain course consisting of predetermined learning materials that cannot dynamically adapt to the learner's knowledge state and learning preference. The domain model we proposed in this dissertation is composed of three layers: ontology layer (of course structure), metadata layer and learning objects layer (see figure 5-3). We explicitly define the domain model as D_{model}={abstract concepts, semantic links, metadata, learning objects}. In particular, the ontology layer is represented by a set of abstract concepts/topics and semantic links instead of actual learning materials, in this proposed model, we identified several different semantic links between domain concepts (DC) and in this paper, they are defined as follows:

- *IsPartOf*(C1,C2) indicates that the concept of C1 is a child element of its parent node C2. Obviously, C2 is a composite concept that has no direct learning objects associated with it. Consequently, when C2 is chosen as the learning target, the actually learning contents are its child concepts.
- *IsRequiredBy*(C1,C2) indicates that to learn C1 needs C2 as a prerequisite. This relation poses a constraint on the delivery order of the DCs to the learner.
- *SuggestedOrder*(C1,C2) means that it is preferable to learn C1 and C2 in this order. Note that also this relation poses a constraint on the DCs' order but now it is not necessary to learn C2 if the learner is interested in only C1.
- *IsReferencedBy*(C1,C2) indicates that there exist a similar concept C2 that provides the chance to the learner refers to.
- *IsTestedBy*(C,TP_i) indicates that the C has a test TP_i for the evaluation of the learner's mastery of this concept.

All the proposed relations can be easily represented by arcs in a graph data structure (where each node represents a DC). Besides the above relationships among DCs, we need a link between the Ontology and the Metadata levels (see figure 5-2). This link is represented as *IsTaughtby*(C, EEO), which indicates that the concept C can be taught by means of learning object EEO which is described by the metadata M.

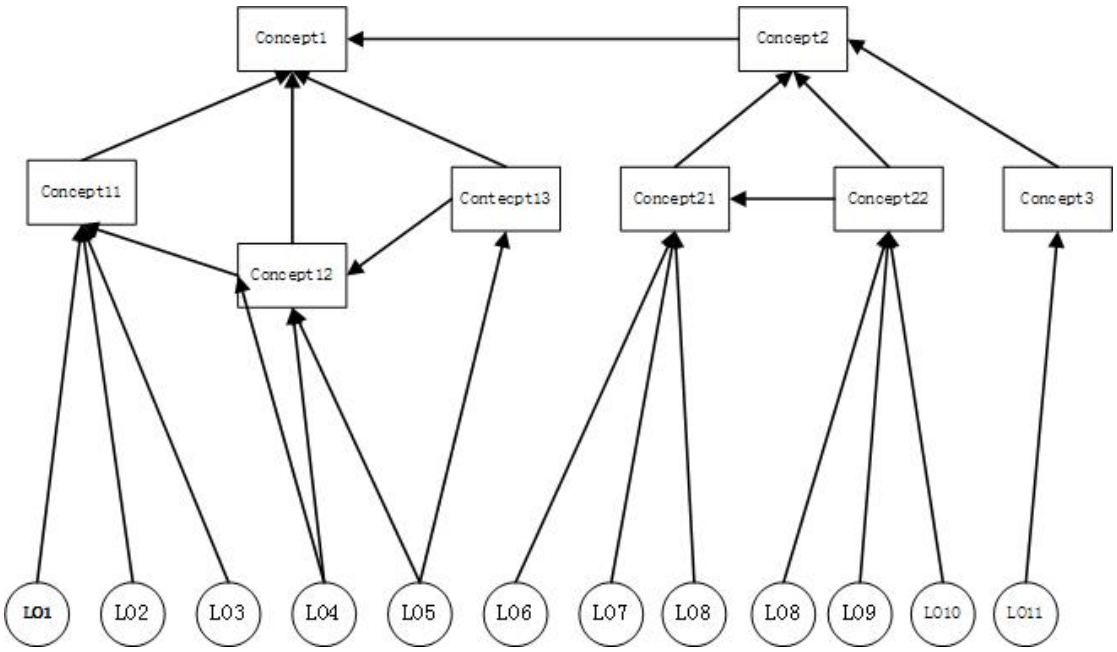


Fig 5-2. Domain concept model

Besides, to enhance adaptability and flexibility, the following assumptions with regard to the learning object should be met.

1. Defining different types of assets (e.g. text, picture, audio, video, hyperlink etc.)
2. Supporting different types of learning objects (e.g. content, exercises, assessment, etc. and any combination of these types)
3. Providing different levels of detail for a learning object (e.g. novice, medium and advanced learner)
4. Mapping a learning object to a learner's characteristics (e.g. language, accessibility, learning style, etc.)
5. Mapping a learning object to one concept specific to certain domain

From the domain model, it can be found that three remarkable feathers can be identified as follows:

1. The course structure (ontology) and actual learning contents (learning objects) are separated thus ensure that different learner can be delivered tailored learning objects according to the knowledge state and learning preference. Note that these learning objects to be delivered to learners are dynamically generated through learning object-searching system described in chapter 4.
2. When to start to learn, the learner can control whether to learning through the course map or be guided by the system by automatically generating learning path and learning content.
3. This domain model is not isolated, it has direct link channel with MAS-supported

agents in support of peer help and collective learning environment.

5.3 ADAPTIVE INDIVIDUAL LEARNING

5.3.1 AGENT ARCHITECTURE

In the proposed architecture, there is a distinct feature that distinguishes the other learning frameworks. From figure 5-3, it is clear that individual learning and collective learning are not isolated, instead they are efficiently combined into an learning framework as a whole in support of personalized learning experiences in both individual and collective online learning setting. As for individual learning, this architecture provides personalized learning path and tailored learning objects closely associated with the concepts on the path in order to explain the target domain concepts that a learner requested. This task is delegated to the *presentation generation agent*, which is responsible for the generation of a set of learning objects (EEOs) that explain (teach) the target

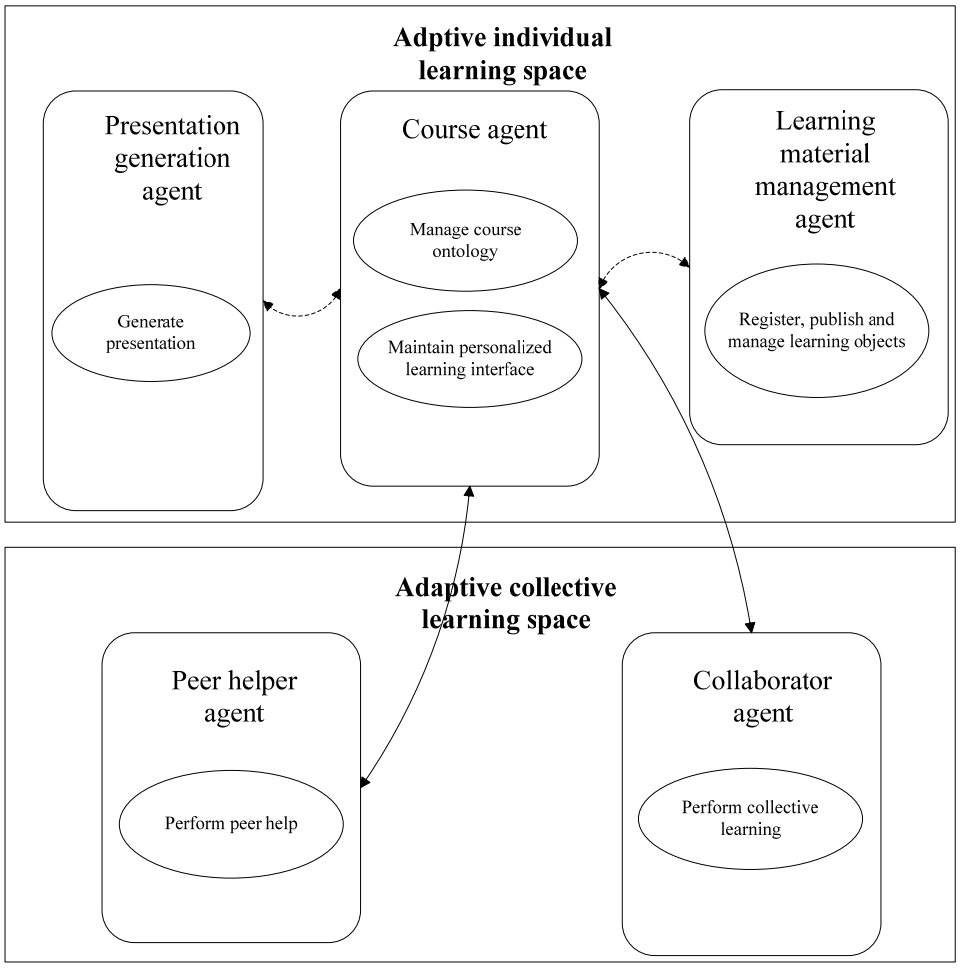


Fig. 5-3. Architecture of adaptive individual learning

concepts stated by the learner. To enhance flexibility, improve learning efficiency and develop the learner's value-added reflection, mutual knowledge sharing. This architecture also introduced the learning objects finding system (see chapter 4) and collective learning space (see section 5.4) involving peer help system and learning group forming, respectively. Image that if a learner is not satisfied with the learning contents recommended by the system, how can we do in this case? To address such issue, this architecture provides several channels for the learner to choose the next learning strategy. One possibility is that the learner request the *course agent* to offer more suitable learning objects that are in fact not included in the specified course, in this situation, the *course agent* can send a mobile agent to the server that the *learning content management agent* resides. If no appropriate results returns, the course agent even can subscribe to such service. The deeper details can be referred to chapter 4. Another possibility is that the learner can ask for peer help. Since when the learners choose to register the same course, they naturally share the same course ontology. Of course, their capability of the course concepts can also be detected and recorded. Thus, it is possible to find candidate learners who are competent in requested domain concepts though the learner's *peer helper agent* (see section 5.4.2). a third possibility is that the learner wants to learning in a learning group. In this situation, the learner can formulate the group profile (i.e., requirements for the group member) and delegate his *collaborator agent* to perform the searching and invitation tasks. Of course, the tutor can also participate in the group learning activity through beneficial suggestions to certain member of the learning group or as a whole (see section 5.4.3).

From the above description, it can see that the proposed architecture is flexible enough to address adaptive learning issue. Especially the incorporation of collective learning space, to certain extent, compensates the lack of direct contact among learners and tutor compared to the face-to-face class paradigm.

The next section, we will focus on the algorithm of automatic learning path generation. This algorithm is performed by the *presentation generation agent* (PGA).

5.3.2 AUTOMATIC LEARNING PATH GENERATION

In this section we describe the automatic learning path generation algorithm that performs the *presentation* generation including both adaptive learning path and learning objects. In this paper, a Presentation(*PR*) is a list of EEOs delivered to a learner in order to meet her/his learning requirements as much as possible. To obtain the PR, the learner has to state a list of target concepts (*TC*) belonging to the learner's learning objectives, and then PGA will request Learner Model Agent (LMA) for this learner's learning preference and

knowledge state. Given them as input, PGA builds a Presentation, namely a list of EEOs which satisfy all the *TC* taking into consideration the learner's present state (i.e., Cognitive State and the Learning Preferences). More precisely, a *PR* is an ordered list of EEOs (*PR*= {*l1,…,ln*}) with the following properties:

1. The union of the EEOs ($\bigcup_{i=1,\ldots,n} EEO_i$) of *PR* is sufficient to explain to the learner all the Target Concepts belonging to *TC*.

2. For each EEO_i, $EEO_j \in PR$, **if** : *IsTaughtby*(C_1,*EEOi,)* **and** *IsTaughtby* (C_2,*EEOj,)* **and** $C_1 \prec C_2$, **then** $i<j$, where the partial order relation $\prec$ between DCs is recursively defined in the following manner:

a. **if** *IsRequiredBy(x, y)* **then** $y \prec x$

b. **if** *SuggestedOrder (x, y)* **then** $x \prec y$

c. **if** *IsReferencedBy*(x,y) **then** $x \prec y$

d. **if** *IsTestedBy (x, y)* **then** $x \prec y$

e. **if** *IsPartOf (x, z)* **and** *IsPartOf (y, w)* **and** $z \prec w$ **then** $x \prec y \ \wedge \ x \prec w \ \wedge \ z \prec y$.

3. *PR* should meet the learner's learning preference(LP)and present knowledge state

While Points 1 and 3 of the above definition are self-explaining, Point 2 needs some remarks. The relations among DCs pose a partial order on the elements (i.e., domain concepts) of a didactic domain. Consequently, EEOs belonging to a same Presentation have to respect this partial order. If, for instance, a Presentation contains l_i and l_j, which explain, respectively, the concepts Derivatives and Limits, then l_j has to precede l_i. The same situation occurs when EEO_i and EEO_j explain DCs not directly linked to each other by an order relation (i.e., *IsRequiredBy*, *SuggestedOrder, IsReferencedBy* or *IsTestedBy*) but their child components of DCs directly linked by an order relation.

The Presentation generation algorithm collects in a Concepts' list named AtomicList all those atomic DCs which can be reached starting from the Target Concepts and following the links *IsRequiredBy*, *SuggestedOrder, IsReferencedBy* and *IsTestedBy*. Then AtomicList is subsequently linearized. Finally, for each of the AtomicList DCs, the algorithm looks for the EEO whose Metadata best match the learner's learning preferences. The main searching algorithm is shown in the table 5-2.

Table 5-2 Presentation generation algorithm

Input: Target Concepts: DC list, KS: Knowledge State, LP: Learning Preferences.
Output: EEO list.
1. Check the course Ontology consistence.
2. Q:= TargetC, AtomicList:= Nil, PR:= Nil.
3. **For** each x $\in$ Q s.t. $\neg$Known(x, KS), **do**:
    a. **If** [*IsPartOf*(y,x) **or** *IsRequiredBy*(x,y) **or** *IsReferencedBy*(x,y) **or** *IsTestedBy*(x,y)] **and** y $\notin$ Q, **then** insert y in Q.
b. **If** $\neg\exists$ y s.t. *IsPartOf*(y,x), **then** insert x in AtomicList.

4. AtomicList:= Sort(AtomicList, TargetC).
5. **For** each x $\in$ AtomicList **do**:
a. Let LOList be the list of all the EEOs e s.t. *IsTaughtBy*(x, e) and *Consistent*(e, LP).
b. BLO:= Choose_the_best_of (LOList,LP).
c. Insert BLO in *PR*.
d. **For** each x' AtomicList s.t. *IsTaughtBy*(x', BLO), delete x' from AtomicList.
6. Return *PR*.

In the presentation generation algorithm, line 1 checks if the ontology that represents course structure is consistent. This is done to avoid loops in the operations performed in Lines 3 and 4. An Ontology is inconsistent when, for example, $x \prec y$ and $y \prec z$ and $z \prec x$. The check is easily realized through looking for a loop in the oriented graph representing the Ontology. For each Target Concept, lines 3.a recursively collects all the DCs needed to learn it. In Line 3.b, a DC is added to the list of atomic concepts if it has not child components. Line 4 linearizes the atomic concept list just obtained.

Line 5, for each DC *x* of the atomic list, selects a most appropriate EEO among all those EEOs able to explain *x* (let us call them *LOList(x)*). It is important to note that the choice is local to *LOList(x)*. Indeed, if we look to the example of Figure 1, a learning object EEO can be linked to two or more DCs. For instance, *EEO3* explains both *Derivatives* and *Integrals*, thus, if in Line 5.b the function:

Choose_the_best_of (LOList(Derivatives),LP)

returns *EEO3*, then in Line 5.d *Integrals* is excluded. Nevertheless, in Line 5.b we can have *BLO = EEO2* because the function *Choose_the_best_of* depends only on *LOList(Derivatives)* and on *LP*. In other words, *EEO2* could be judged better filling the learner preferences with respect to *LO3*, without taking into account any global

optimization or minimization of the final EEO list. The reason of this choice arises from the fact that a global optimization would lead to a combinatorial explosion, while we generally have only very few EEOs linked to more then one DC. Finally, the functions *Known*, *Consistent* and *Choose_the_best_of* are easily realized comparing DCs or EEOs with the Knowledge State or Learning Preferences' facts.

5.4 ADAPTIVE COLLECTIVE LEARNING

5.4.1 INTRODUCTION

One of the basic requirements for education in the future is to prepare learners for participation in a networked, information society in which knowledge will be the most critical resource for social and economic development. Computer-supported collaborative learning (CSCL) is one of the most promising innovations to improve teaching and learning with the help of modern information and communication technology. Collaborative or group learning refers to instructional methods whereby learners are encouraged or required to work together on learning tasks. It is widely agreed to distinguish collaborative learning from the traditional direct transfer model in which the instructor is assumed to be the distributor of knowledge and skills. The shift from face-to-face learning groups to asynchronous DLGs introduces specific requirements for the design of CSCL environments to overcome the constraints of space and time, and to compensate for many of the elements that typically occur in face-to-face learning groups. One of these elements is the seemingly effortless social interaction that takes place and has been recognized as the crucial element underlying the current interactive learning perspectives meant to encourage shared understanding (Mulder, Swaak, & Kessels, 2002), critical thinking (Bullen, 1998; Garrison, Anderson, & Archer, 2001), social construction of knowledge (Jonassen, 1991a, 1991b, 1994) and the acquisition of competencies (Jochems, 1999; Keen, 1992). According to Kearsley (1995) one of the most important instructional elements of contemporary distance education is interaction due to its positive affects on the effectiveness of distance educational courses. Social interaction appears to be particularly important for achieving shared understanding and the construction of knowledge based on the social negotiation of views and meanings. Hiltz (1984) underlined this when she stated that "the social process of developing shared understanding through interaction is the 'natural' way for people to learn". CSCL environments embracing group learning, critical thinking, constructivist learning, and competency-based learning emphasize social interaction.

In the following two subsections, we will focus on two approaches to facilitating the

constructivist learning.

5.4.2 PEER HELP MODELING

5.4.2.1 INTRODUCTION

To illustrate the functionality of peer help system (PHS), we will use an example scenario. Imagine that a learner working on a programming assignment in a Computer Science course still has problem on certain domain concept presented by the system. She may delegate the task of finding help to her personal assistant agent. The personal agent tries to find another agent (either application agent or another personal agent) that offers information resources related to the help request. These resources can be electronic resources, for example web pages created by the instructor or other learners (and represented by their application agents in the system), or threads / postings in a discussion forum (represented by discussion forum application agents). The agent can also find "human help resources", i.e. learners who are currently on line and competent in the concept of specified course. The agents share a common taxonomy for indexing the information resources, based on the topics/concepts taught in the class. Usually the course instructor creates the taxonomy (represented as the course ontology in this dissertation) from the course outline when the system is configured for a given course. To find a most appropriate peer helper, the learner's assistant agent has to request a special agent named matchmaker agent to perform such matching task. The key issue is how the matchmaker agent locates the agents that possess information resources or represent users knowledgeable on certain domain concept is facilitated by DFAgent agents that maintain profiles of the knowledge and some other characteristics of users and applications.

Back to the scenario: if there is no appropriate electronic resource for the learner's question, the matchmaker creates a ranked list of the learners who are on line and who know something about the domain concept. The matchmaker sends this list to the personal assistant agent of the learner who asked for help. The personal assistant agent starts negotiation with each of the personal agents of the potential helpers on the list, trying to find one that would agree to help at a satisfactory price. Once the negotiation process has succeeded, the agent of the potential helper notifies its learner and asks her if she would be willing to help. If not, the personal agent has to negotiate with other agents from the list of suggested helpers. If the learner is willing to help, a communication channel is opened between the two users (e.g. a simple chat tool), and a help session starts. After one of the parties terminates the chat, an evaluation form (specific for the matchmaker that recommended the helper) pops up allowing each learner to evaluate the other learner. This

information is stored by the personal agents (i.e. each personal agent contains a model of the other user); it is also forwarded back to the matchmaker to update profiles of users in its database.

5.4.2.2 ARCHITECTURE OF PEER HELP

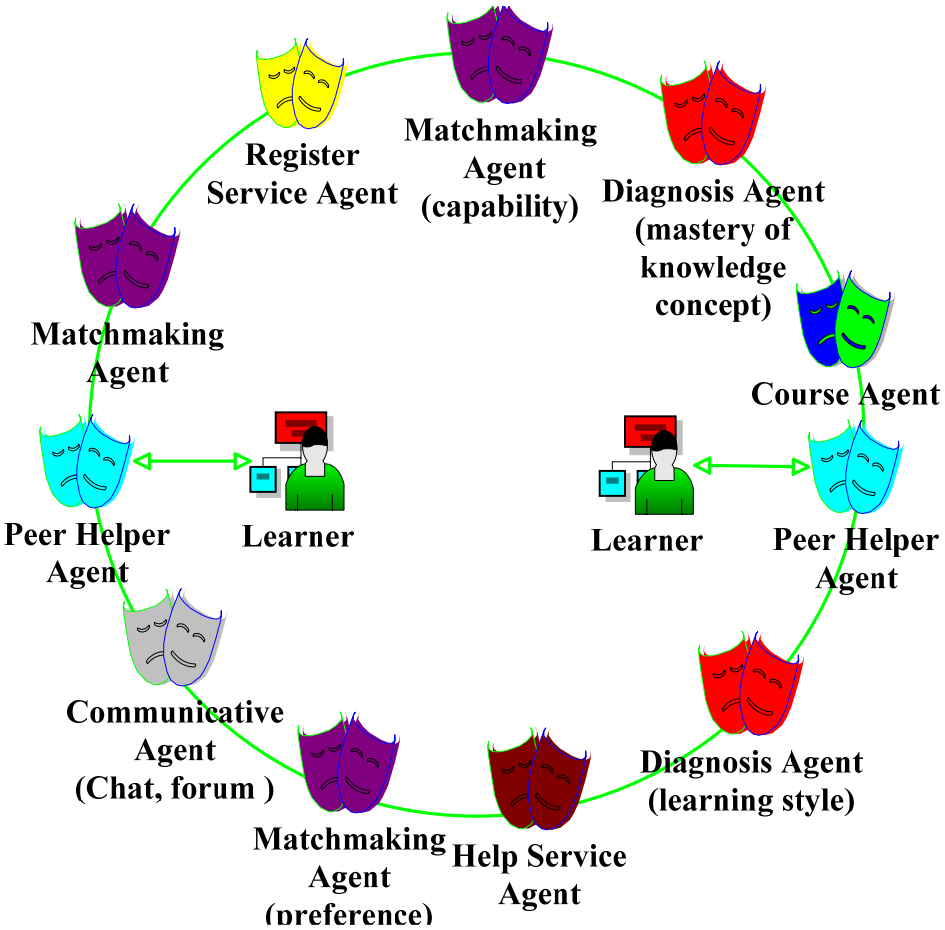


Fig. 5-4. The architecture of peer help system (PHS)

The proposed architecture of peer help system is illustrated in figure 5-4. It can be found that this architecture is composed of several agents that perform different task through cooperation, negotiation among them. These agents share the common communication language (e.g., ACL) and ontology (e.g., HelperOntology). Each agent manages specific resource of the user or the application it represents, for example the knowledge resources of the user about the domain concepts, or the instructional materials belonging to an application. Agents trade these resources when they need resources that they do not possess. For this, they negotiate and establish long-term inter-agent relationships, some of which reflect relationships between learners. In this way, we achieve a complex multi-user, multi-application, adaptive, self-organizing system that supports

learners in locating and using resources (other learners, applications, and information).

Most of the involved agents are briefly described and discussed as below:

Peer helper agent,(PHAgent), in the context of the peer help system, maintains partial learner models containing certain basic learner characteristics. Examples of such characteristics are lists of the learner's friends and foes, preferences about how the agent should negotiate on the user's behalf, taking into consideration the subjective importance to the learner of certain resources like time or money, and the user's egoism or altruism. These characteristics are set explicitly by the learner. They reflect the way the learner wishes to be perceived by the "world" through his/her personal agent; therefore, indirectly, they also represent a kind of model of the user. During negotiation with other agents, the PHAgent acts as a representative of the user. The agents try to optimize their actions and to predict the "opponent's" actions. For this purpose, they create models of the other agent's "character". Thus, each PHAgent models the character that the other user wants his/her agent to represent in the agent community. LHAgent also take into account relationships that may have previously formed between the learners, for example, by changing the negotiation strategy (offering a discount for friends or an extra high price for "enemies"). After repeated successful negotiations followed by successful help sessions between the learners, the agents offer to add a new relationship between the learners in their models, thus increasing the number of "friends". In addition, PHAgent collects references to other agents who keep information about the learner, e.g. *learner model agent* (LMA) that records the knowledge mastery in the domain of a specified course, and different *diagnosis agent*s (DA) that are responsible for developing model of the learner's knowledge in various domain.

Course agent (CAgent) is an important application agent with GUI that represents the course, which is matched to the taxonomy of domain concepts to be taught, when being requested it, this agent can display on the screen this course map consisting of domain concepts. It can also store user preferences with respect to the interface of the web-based course. Of course, it keeps references to LCMAgent (see chapter 4) in support of locating more appropriate learning objects when all learning objects provided by this course can not meet the learner's requirement. Besides, this agent also has links with the *Peer helper agent* and *Collaborate agent* in order to get peer help or learn in a collective learning setting as needed, for example, when a learner feels confused about certain course concept or desires to discuss one domain concept collectively.

*Communicative agent*s (CommAgent) are mainly responsible for the management of communication being taking place (help session) between the learners

Register Service Agent (RSAgent) performs a single task, i.e. to register the help service information with *Help service agent* (see the following paragraphs). It is worth nothing that in the service information also involves preference information such as the preferred language, learning style, desired communication tools and so on besides the domain concepts that have been well mastered.

Diagnostic agents (agents representing test items, questionnaires etc.) represent a special type of application agent that creates learner models for a particular purpose. For example, a learning style diagnostic agent detects the learner's learning style by a questionnaire composed of set of ILS questions according to the Felder-Silverman Learning Style Model (see section 2.3.3). a monitor agent monitors user activities (browsing, reading and posting in the discussion forum), checks time-stamps and updates the level of user eagerness. There also exist agent that models users' mastery of domain knowledge. some diagnostic agent allows the learner to fill a self-evaluation form to initialize knowledge model of the learner. Another kind of diagnostic agent translates learner assignment grades into probabilities about the learner's level of knowledge about course concepts. There is no integration of the different models of the learner's knowledge at a central place, which is a crucial difference with centralized user modeling approaches. Data is retrieved and integrated "on the fly" from the various agents only when it is needed.

Help service agents (HSAgent) is extended from the FIPA DF agent (see FIPA 2000) that has capability to publish and maintain any service. In this paper, especially we regard the learner's capability of mastery of the domain concepts as a kind of help service resource. In fact, there are two possibilities to register help service with the *Help service agents*. One is that the peer help system itself automatically performs such operation, in particular, during the course learning, the diagnosis agent may assess the learner's mastery of some specified domain concepts at appropriate time and reports the results to the course agent, if level of mastery of the tested concepts excess over a threshold, it makes clear that the learner is competent in helping other peer learners understand such concepts. In this case, the course agent commits the registration task to the *Register service agent* that registers the peer help service with HSAgent. Another possibility is that the learner herself/himself takes the initiative to declares or modify her/his help service information directly in manual manner (see 7.5).

Matchmaking agents act as a special broker that is responsible for performing matchmaking for specific purposes (e.g. locating the peer helper who is most knowledgeable on certain domain concept, or locating a helper who has a compatible learning style, etc.). In fact, there may exist many types of matchmaking agents, specialized according to various purposes. Each agent finds suitable peer helpers according to different

criteria. For example, one broker finds the most competent peers on the domain concept of the help request. Another broker finds peers that are currently available (on-line). A third broker finds peers that have particular social characteristics (e.g. eagerness, helpfulness, class ranking) that might be beneficial for helpers. A fourth broker finds peers that have a similar learning style to that of the learner asking for help. A fifth broker finds peer-helpers only among the friends of a learner. Thus, each matchmaking agent takes a different set of learner characteristics and calculates a score using a simple ranking algorithm. Typically several broker agents work together and pipeline their results to produce ranked shortlists of helpers that are optimal according to some combination of criteria.

Obviously, the proposed architecture is based on the cooperation and negotiation between the involved agents; we will illustrate the peer help process in the next section.

5.4.2.3 MODELING APPROACH IN PEER HELP SYSTEM

As can be seen from the discussion above most of the communication and reasoning about the learner's knowledge is distributed among the agents in the system. Learner modeling and adaptation is thus fundamentally fragmented and localized. In the peer help system, there are several situations with regards to the multi-learner multi-agent modeling approach. We describe them as below, respectively.

A. Learner Modeling Agents

The *peer help agent*, (as a representative of the learner), is allowed to modeled by its owner, i.e., the learner can model this agent's character and strategy, i.e. how cooperative this agent will be, and how it will engage in negotiation. Since the agent represents the learner in the system, in some sense, this is how the learner is perceived by the other agents and (indirectly) by the other learners. The "character" imprinted in the agent by the learner is somewhat related to the notion of explicit user modeling (Rich, 1983) and open user models (Bull & Pain, 1995; Paiva, et al., 1995). However, in the traditional notion of "inspectability" of user models it is assumed that the learner can view the model that the system has created of him/her and the learner corrects misrepresentations that the system may hold about him/her. In our case the learner can create a personalized agent, which deliberately differs from the learner's.

B. Agent Modeling Other Agents

To negotiate better, the *peer helper agent* (as personal agent) needs to be able to predict the next move of the other agent. This move depends on the strategy and preferences of the opponent. Since all the personal agents are self-interested (they work to satisfy best the needs of their users), it cannot be expected that the agents will reveal their priorities. Creating and maintaining a model of the opponent and sharing this model with other agents may help the agents overcome this problem. It is possible to use probabilistic

influence diagrams to model the preferences of the opponent agent.

C. Users Modeling Other Users

A learner can instruct an agent about other users, either by providing evaluation in some form to be interpreted by the agent or by explicitly setting values for certain features in the models of other users (or their agents) maintained by the learner's agents. For example, in PHS, learners evaluate each other after a help-session. The peer helper agent of each user gives a short evaluation form to fill; in effect, a simple model of the other user's competence and helpfulness is constructed by the personal agent. Learners also can instruct their agents about who their friends are and what their friends can be contacted about, what domain concepts (or topics) they are good at, topics for which they should not be contacted. Thus, the user creates simple models of other users; the personal agent of the learner utilizes these models to navigate better in the social space of the environment.

D. Agents Modeling Users

This case comes closest to the traditional process of a system modeling the user. In PHS, we hope that the diagnostic agents have the ability to use specific rules to infer values of particular learner characteristics from raw data and from other agents. For example, the diagnostic agent computing the eagerness of a given learner receives data about the number of times the learner has logged into the system and the number of postings that the learner has read and posted from the application agent of the discussion forum. Matchmakers collect user model information about the knowledge of all learners that are in a given group (class) from diagnostic agents specific to certain topics of the class taxonomy and from the personal agents of the learners, from which they receive both results of learners' self evaluations and peer evaluations (after a help session). In integrating this information they use specific rules (e.g. give more weight to more recent information, to diagnostic agents over peer helper agents or to peer evaluation over self evaluation).

5.4.2.4 AGENT INTERACTIVE DIAGRAM

To clearly demonstrate the agent interactive process among the proposed peer help system (PHS). the UML sequence diagram is used to illustrate a common scenario (see figure 5-5) in which, a learner A can choose to learn a set of domain concepts in adaptive individual learning space, when the system detects that the target concepts (all or partial) that learner A has learned have been mastered, it will automatically register such capability as a help service with the *help service agent*. Of course learner A can also specify the scope, condition and strategy of service. While another learner B still cannot understand certain concept after a period of learning time, he is willing to get peer help in collective learning space. In this case, it is possible for him to commit the peer help searching task to his *peer helper agent*. Once a list of peer helpers on line are returned, this agent will select a most

appropriate peer helper according to the criteria “indoctrinated” by learner B. the detailed steps are describe as below:

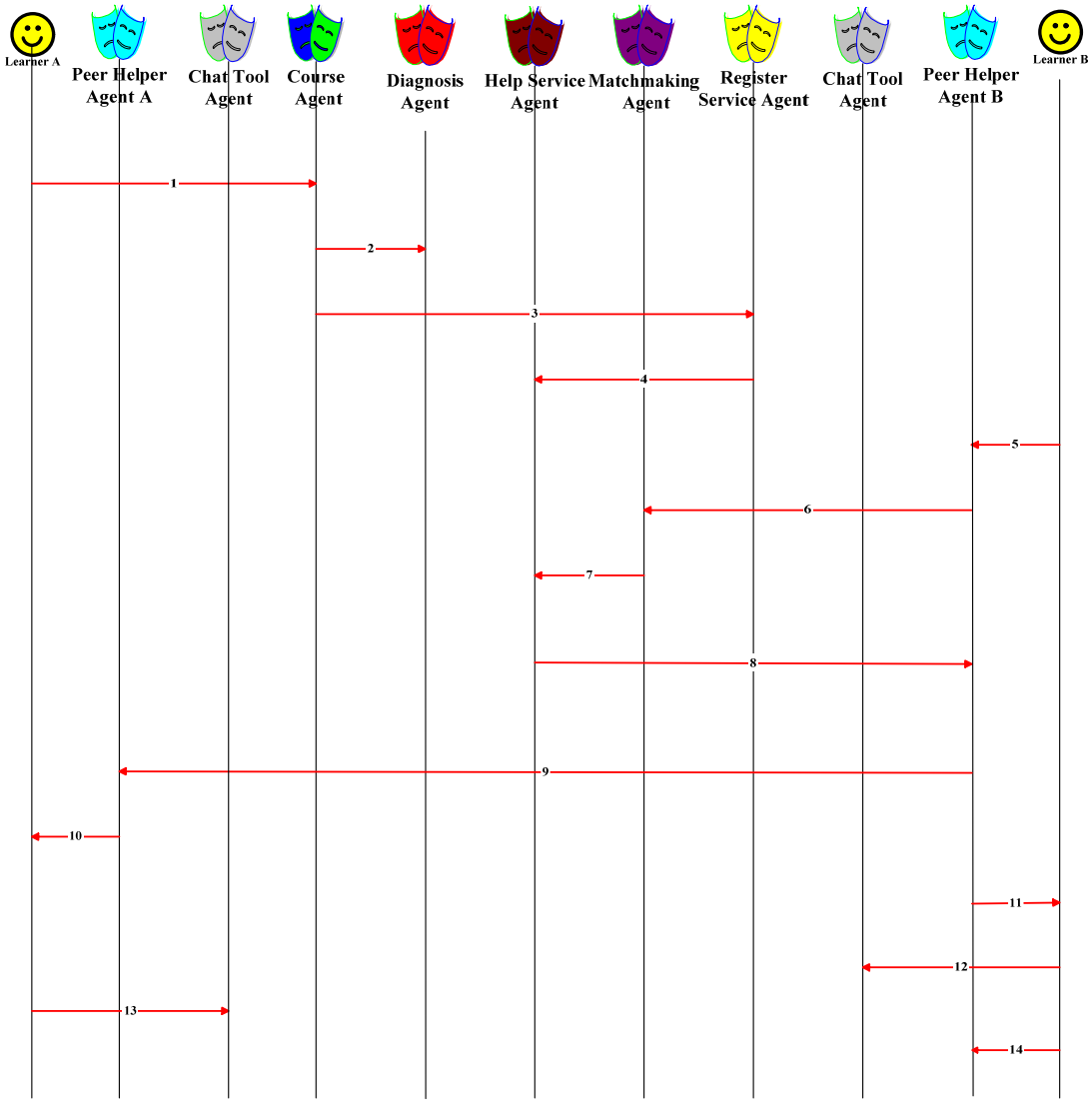


Fig. 5-5. Peer help sequence diagram

Step 1. Learner A request the course agent to display the specified course map consisting of the domain concepts. Once this learner states the target concepts, the course agent will send a request message to the learner model agent in order to initiate the learner’s present knowledge state and learning preference. Given the chosen target concepts, knowledge state and learning preference as input, the course agent has the ability to generate an adaptive learning path (all the concepts necessary to explain the target concepts) and associated learning objects.

Step 2. When a specified time arrives or learner A takes the initiative to take a informal performance test so as to validate his learning performance on the target concepts,

in this situation, the course agent requests the diagnosis agent to compose a test and records the final results.

Step 3 and step 4 . Once there are tested concepts that have passed a give threshold, the course agent asks the *register service agent* to register the help service information with the *help service agent*.

Step 5. learner B would like to learner certain target concept though peer help system, thus, he start his *peer helper agent.*

Step 6. the *peer help agent* sends necessary matching information to the corresponding *matchmaking agent* in order to carry out peer helper searching.

Step 7. The *matchmaking agent* requests the *help service agent* to locate the peer helpers who meet the matching requirement.

Step 8. The *help service agent* return a list of peer helpers, who are knowledgeable about the requested domain concepts and satisfy other specified matching condition, to learner B's *peer helper agent*.

Step 9. The *peer helper agent* of learner B begins to negotiate with that of learner A (supposed learner A is among the candidates of peer helpers). when this negotiation is successfully reached, they will notify their learner, respectively. Otherwise, the *peer helper agent* of learner B has to continue to negotiate with other *peer helper agent*s that represent the potential peer helpers.

Step 10 and step 11. Both of helper and helpee's *peer helper agent*s inform the negotiation result to their learners

Step 12 and step 13. If successful negotiation is reached, the helper and the helpee respectively start their communicative tools (as chat tool) and begin the help session.

Step 14. When the help session is terminated, learner B can evaluate the performance of the peer helper.

5.5 LEARNER GROUP FORMING MODELING

5.5.1 INTRODUCTION

Most educators agree that advances in computational technology are going to have a positive impact on educational activities. However, it is not so clear whether the emphasis should be placed on individual or on collaborative learning. Collaborative learning supported by computers seems to be very promising, since advances in computational technology enable the widespread use of tools such as bulletin boards, chats, whiteboards, and videoconferences. Individual learning provides benefits such as self-pacing and

establishment of learning goals by the learner.

It would be interesting, if possible, to combine the two learning paradigms into a common learning framework. However, this combination is not easily attainable. Usually, collaborative learning environments emphasize Computer-Mediated Communication (CMC), with tools to integrate e-mail, bulletin boards, whiteboard and chat rooms into HTML pages (Collins-Brown, 1999). In this context, groups have to be previously assigned to work together, and the administrator must create the corresponding e-groups. It is assumed that the group will work together in order to achieve the common understanding which should result from this community of learners. Nevertheless, with the incorporation of multi-agent, similar to the peer help system, it is easy to construct the collaborative learning environment still on the individual learning space basis.

This section presents architecture to dynamically establish collaboration groups for individual learners sharing a common learning goal. In this architecture, individual learners can establish a collaboration profile. Indicating the characteristics of the group, they would like to participate. The proposed collective learning architecture is based on several agents, which perform functions such as seeking for potential collaboration partners, expressing which collaboration services are to be used, and monitoring collaborative learning activities.

5.5.1.1 GROUP FORMING MODEL

From the technological point of view, collaboration among users of an on-line learning environment depends on two tasks, which are the group definition and the establishment of (synchronous or asynchronous) communication sessions. Therefore, a collaboration framework should at least provide the tools to perform both tasks.

The definition of a study group is a trivial task when the required negotiation for composing the group occurs off-line. However, in an on-line learning environment in which a teacher may not be present, the learner ignores who are the other learners in the environment and what they are studying, this task become quite complex.

In order to support learning groups in this kind of environment, the definition of a learner collaboration profile is proposed. The adoption of this profile enables to find collaborators with desirable levels of skill and knowledge, thus making possible the composition of groups with common interests, which would potentially improve the performance.

The key issue of the proposed framework, which aims at facilitating the establishment of collaborative settings in an on-line learning environment, is the concept of group, a dynamic set of learners who are brought together to discuss about some domain concept or

special topic.

A group has a life cycle that starts with the identification of group profile by any learner in the environment. This learner, who acts as the group owner, establishes the set of desirable features for recognizing potential members for this group . i.e., the group profile (see figure 5-6, a).

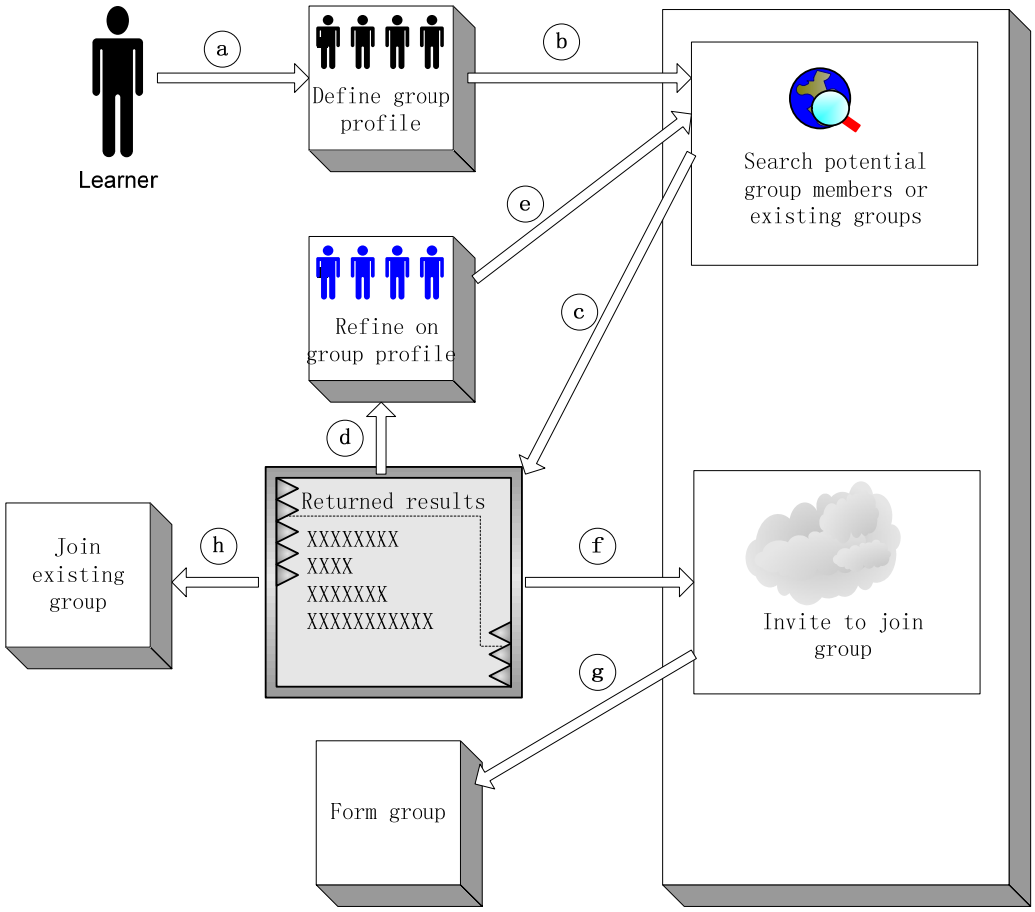


Fig. 5-6. Group Forming Process model

The group profile is composed of a list of conditions expressing the goal to achieve, and optionally, a required degree of knowledge on the domain concept (expressed as a real value between 0 and 1) that members of the group should have. For example, in order to start learning group to study the domain concept “agent interactive protocol” with members that already have a reasonable knowledge on agent theory, a learner can possibly customize a group profile as below:

(Agent Theory, 0.6) AND (Agent interactive protocol)

The adoption of a mechanism to search for collaborators, based on conditions connected by AND and OR operators, enables the owner to determine the degree of homogeneity or heterogeneity for the group being created. For example, conditions related

by OR operators sets a group in which its members competences are heterogeneous. This flexibility enables to get higher educational benefits in defining heterogeneous groups as stated in Johnson and Johnson (1996), or to define other learning conditions in which homogeneity is more suitable.

The group profile is used by the collaborative learning framework to start a search for users of the on-line learning environment whose individual profiles satisfy the proposed set of conditions (see Figure 5-6, b). The result of this search is a list that may contain learners and existing groups fitting the proposed group profile (see figure 5-6, c).

The learner proposing the group may then decide either to suspend the creation of a new group and try to join an existing group (see figure 5-6, h), or to proceed with group creation. The proponent learner may also not be satisfied by the search results. e.g., the list of potential group members might be too long or too small. In this case, the learner might choose to review the set of conditions in the group profile, either by restricting or by relaxing the required degree of knowledge for the target concept or by introducing or removing some conditions (see figure 5-6, d). The system can again perform the searching process applying the new group profile (see figure 5-6, e).

When the learner receives a list of potential group members that satisfy the proposed group profile, the searching phase ends and the framework starts the invitation phase (see figure 5-6, f). First, the proponent learner (now the group owner) may select learners within the list of group members and then the group owner then assigns the task of inviting the selected potential group members to join the new group. Upon acknowledgment of all invited learners, the framework concludes the creation of the group (see figure 5-6 g).

Besides enabling the creation of new groups, the framework shall provide functionalities to monitor activities and participation of group members. A practical use of this module is to detect the degree of individual participation of a learner within a group or even the activity level of the group. This information could be used by the learner, in the case of the participation monitoring, to change her/his behavior within the group in terms of participation and, in the case of activities monitoring, to decide whether she/he is going to continue to work, or not, with the group. Another possible module that can be provided is a knowledge-monitoring module that may be used to evaluate the quality of the knowledge achieved by the group and its members. Knowledge here is used in a broader sense, meaning not only the acquired concepts on a given subject but also the methodology used to get these concepts. The modules above mentioned will be used by the tutor who is responsible for monitor whether the learning group has achieved a valid knowledge in the subject being discussed.

5.5.1.2 AGENT ARCHITECTURE

It is obvious that the proposed model is naturally implemented with MAS since the task related to searching and invitation of peer collaborators is easily performed by agents. The group creation process is time consuming, since the learner may be involved in successive steps in profiles refinement process and in the searching and selection of the appropriate group members. With software agents, the tasks related to the collaboration framework can be easily done on behalf of the learners. The *group agent* performs the searching task, considering a given group profile, and proceed to invite the selected partners or even to request the inclusion in an existing group. This agent is also responsible for verify the matching between the specified group profile against the individual learner profiles.

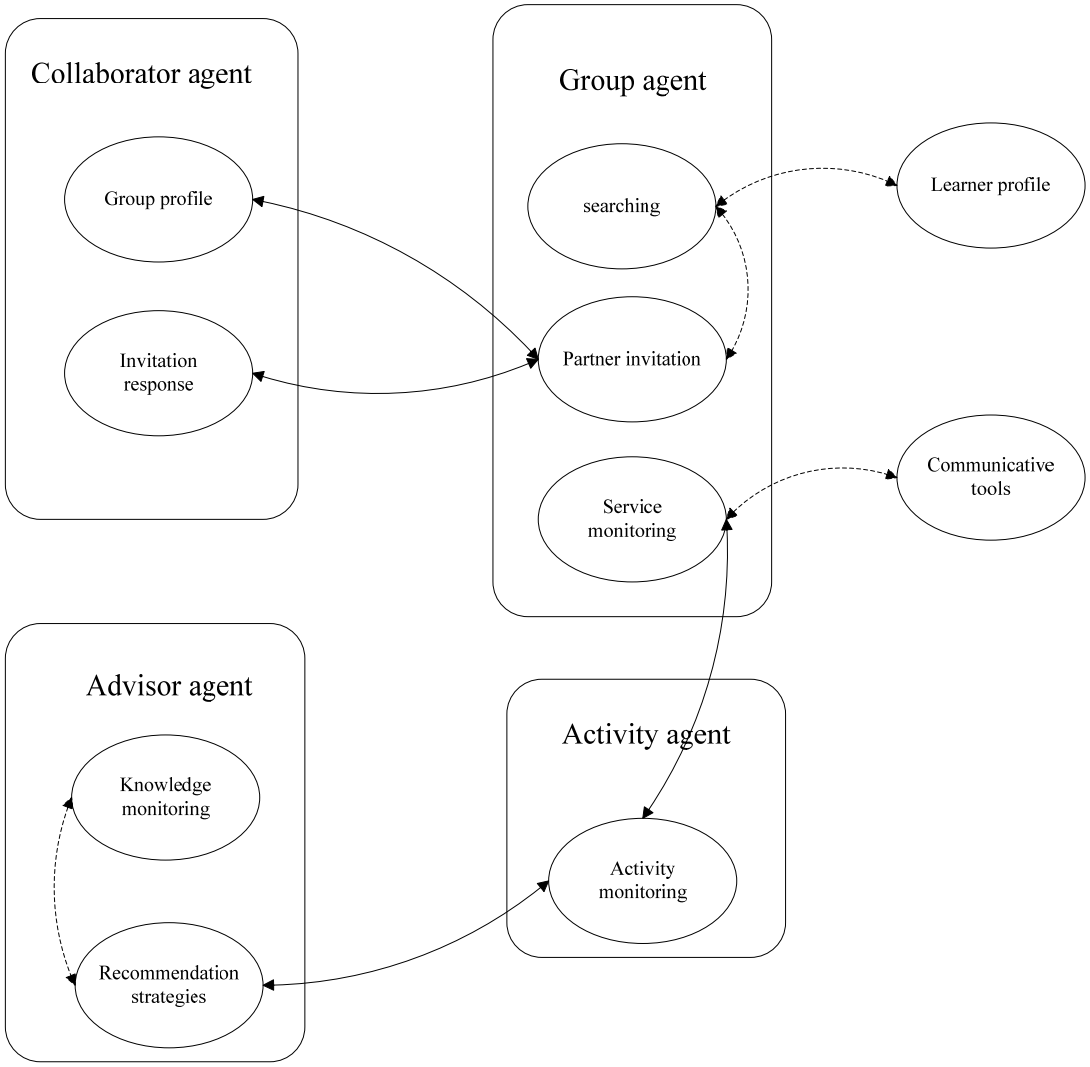


Fig. 5-7. Multi-agent architecture of learning group forming system

From figure 5-7, it can seen that this architecture is composed of a set of agents which jointly implement the collaborative learning framework. The *collaborator agent* on behalf of it learner is responsible for receiving users goals specifications and performs actions to fulfill these goals, this is also the point of access for learners to establish a group session or to receive information of other agents to be passed to a learner.

The *group agent* performs the tasks related to searching and invitation processes. It is the this agent that is responsible for keeping all the information about the groups opened by a learner, including member list and control data from collaboration services. This agent has some important properties. It must be active while there are active groups on line, ensure the persistence and security of groups information, and be able to locate or to be located by an owner.

In the searching and invitation process, mobile agents are useful. Lange and Oshima (1999) has pointed that mobility is one desirable property of software agents, depending on the tasks to be performed, the volume of the data to be manipulated and the characteristics of the networks in terms of performance, quality of services (QoS), and topology. The search for partners involves the query of one or more database servers which keep learner profiles. These learner profiles contain information about characteristics of learner in terms of theirs competences and skills in a set of subjects, and in terms of performance in group activities. Based on some measure of network performance and data amount to be retrieved, or on the evaluation of distribution of learners in the network, it may be useful to send an agent (i.e., *searching agent*) through an itinerary of such servers, rather than retrieving the data for local processing.

The *collaborator agent* is associated with any learner in the learning environment, group owner or not. This agent is responsible to interact with *searching agent*s sent by group agent. During searching process, it can be important to know which are the resources available in the computer platform used by the potential collaborator. Also, the agent interacts with *invitation agent*s in order to receive invitations and reply to them on behalf of its owner, eventually with some human intervention.

The other two agents are the *activity agent* and *advisor agent*. Once a group activity goes into effect, some collaboration services (i.e., communicative tools) are used. These services are not part of the architecture, but they provide information that is collected by the *group agent*. This information can be used by the activity group members, this may be important to evaluate the effectiveness of a group.

The *advisor agent* performs most of the tasks related to group and members knowledge monitoring. A group advisor (maybe a online tutor) must use this agent to send

recommendations to a group as a whole or to one of its members. These recommendations will be based on the members profiles and on the information related to the group activities, as provided by the *activities agent*.

5.6 SUMMARY

This chapter proposed a distinct constructivist-learning framework that distinguish other counterpart in the following aspects:

- The domain learning concepts and it concrete learning objects associated with it are separated;
- How to teach lies on the learner preference and learning status and how to learn lies on the learner's control on his/her own;
- When difficulty occurs among domain concepts, a learner can directly request peer help online or form a learning group in collective learning environment;
- The leaner and agent model is not static but dynamic and distributed modeling process

Owing to the time and page constraints, some other issues are not profoundly considered in the proposed learning framework. However, they are definitely worth taking into consideration in the future work. For example, one issue on group learning that must be considered is how to deal with group-based decisions. The voting criteria might be important since, in terms of degree of relevance and acceptance, it can improve the evaluation of an educational agent about the evolution and understanding of the subject by the members of a group, both collectively and individually. This must be better studied in order to implement a software agent. Another issue on the peer help system is how to build the learner's mental and social model, how to efficiently negotiate between the learners' personal agents still deserves the further research, and eventually there maybe exist better mechanism used to encourage the engagement of peer help. This work will be our focus in the next work.

CHAPTER 6 AN INNOVATIVE E- ASSESSMENT APPROACH: MOBILE AGENT BASED PARADIGM

This thesis put forward an innovative holistic solution to modeling large-scale on-line assessment system by applying the new generation of mobile agent based distributed computing paradigm. In particular, the most significant innovative point consists in that we proposed and designed an innovative model of automatic test generation by seamlessly integrating genetic algorithm, mobile agent, and MAS.

6.1 INTRODUCTION

As is well known that whether in e-Education programs, e-learning portals, or the traditional education environment, teaching and assessment cooperate as a complete learning cycle. In traditional classroom based paradigm, the assessment is employed to identify the strengths and weaknesses in the teaching process, determine whether that a course has accomplished its objectives, or measure learners' performance for purpose of rating, feedback, or readjustment of teaching strategies and reorganization of teaching materials. In e-Education context, the e-assessment (i.e. online assessment) plays the same role but has the potential to provide more flexible and innovative assessment paradigms and approaches. Some significant benefits or potentials of e-assessment are identified as bellows:

- Greater flexibility for tutors, learners, evaluators with any place and any time assessments.
- Saving time of tutors and administrative staff through simplified and quicker examination procedures.
- Reduction in the workload of tutors when feedback is provided automatically or evaluation work is done by other test evaluators.
- Monitoring the progress of large learner numbers is easier with e-assessments.
- Rapid feedback of individual and group results via result publishing system.
- More flexible and imaginative assessments possible. for instance, objective test items submitted online can be evaluated by automatic evaluation engine while subjective test items can be readily distributed among evaluators for marking or grading;
- Improved access for learners who are unable to attend for assessment on-campus

due to physically disabilities or family commitments.

- Test result data can be captured and analyzed by the tutor or particular automatic diagnostic components (i.e. agents), thus enabling personal learning experience.

Although the above benefits are exciting and attractive from both learner and tutor perspective, its potentials is far from being brought into play. As Ryan (2000) pointed out that "preliminary searches for good examples of online assessment reveal no mainstream examples of the potential of the new media to construct authentic, flexible and meaningful evaluation of the range of learner learning." Some of this lack of innovation in assessment may stem from the perception that e-Education is somehow "second class learning." Hence, universities undertaking e-Education assessments will not stray far from the assessment practices of face-to-face traditional teaching as they recognize that "Only by meeting such normal quality standards you will be recognized as being serious and the results of your teaching effort may result in accreditation in the university context." (Fritsch 2003). However, a study of e-assessment confirms that the principles for quality e-assessment are the same regardless of the delivery mode. In other words, validity, reliability, fairness and flexibility are the key measures for quality assessment (Booth et al. 2003).

From the literature, it is found that the traditional computer based evaluation mechanisms, such as Web Based Testing, rely predominately on the client-server model. Generally, in Web Based Testing (WBT) systems, the learners on the client side download a questionnaire as a web page and submit the answers back to the server. The server evaluates the answers and returns the results to the client. Java Applets and scripting languages like Java Script etc. are the frequently used techniques to enable front-end client processing. Common Gateway Interface (CGI) scripts or Java Servlets are the most often used techniques for server side processing. Such mechanisms usually do not scale well and do not fully support features like automatic test generation, evaluation of subjective questions, delivery of dynamic content, off-line examinations, flexible communication between online evaluation components, and proactive event notification etc. Obviously, these features are extremely desirable for e-assessment and there is a need for alternate ways of designing such applications. In particular, there is a need to provide the following features:

- Comprehensive solution: automatic test generation, delivery, evaluation, and result compilation as well as publishing are important components of the e-assessment application and should be well integrated with each other and the rest of evaluation system.
- Support for subjective questions: Answers that involve written text or graphical schematics would normally require manual evaluation by one or more evaluators. The system should support a workflow of answer papers among these evaluators.

- Delivery of dynamic content: Questions may need to be presented to the learners using dynamic content in the form of audio, video, multimedia etc. Sometimes it might also be necessary to send a tool (e.g. a compiler for client-side code compilation and testing) to the learners.
- Offline examinations/operations: Unreliable links, security and other reasons might require that learners, tutors, and evaluators work offline for certain durations.
- Support for push: There are cases where pushing information to the users is a better alternative than the users pulling the information from the servers. E.g., such a need may arise when some run-time notices are to be communicated to the learners. Since the existing WBT mechanisms primarily use the client-server and pull model of distributing information, we feel that it would be cumbersome to extend them to provide the above features. Hence, there is a need for alternate mechanisms.

Under this context, this chapter put forward an innovative holistic solution to modeling large-scale on-line assessment system by applying the new generation of mobile agent based distributed computing paradigm. The promise of mobile agent paradigm makes possible to address the above issues in a natural and elegant fashion. Consequently, we will exhibit in this chapter the advantage and use of various mobile agents in four typical e-assessment process: test generation, delivery, evaluation and result publishing. In particular, the most significant contribution in this chapter is that we proposed and designed an innovative model of automatic test generation by seamlessly integrating genetic algorithm, mobile agent, and MAS. Since mobile agents are autonomous and dynamic entities that have the ability to migrate between various nodes in the network, they offer many advantages over traditional design methodologies like reduction in network load, overcoming network latency and disconnected operations etc.

6.2 OVERALL FUNCTION STRUCTURE

In this section, we will discuss and describe the overall architecture of the proposed mobile agent based e-assessment system (MAEA), and then elaborate on four assessment services components necessary to e-assessment process. As shown in figure 6-1, MAEA is built on four layers from functionality perspective, i.e., GUI layer, Test Service Layer, Data Access Layer, and Agent Communication & Management Layer. It is worth noting that, in MAEA, different agents on behalf of particular services, applications, or human actors, either mobile or static, reside on client-side or server-side machines that may be dispersed over internet physically. This is natural because of the characteristic of e-assessment environment where different actors such as learners, tutors, test evaluators, administrators

are typically located around the world.

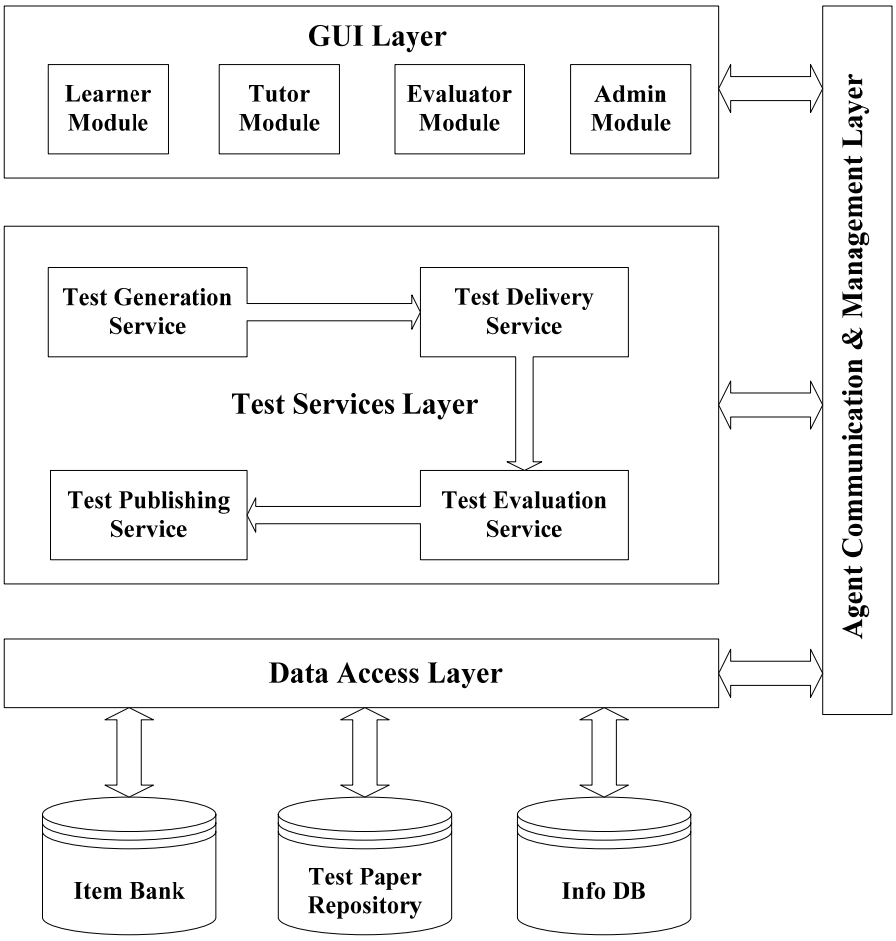


Fig. 6-1 General Service Architecture of MAEA

In particular, in GUI layer, there exist four types of users who participate in the e-assessment process: learners, tutors, evaluators and administrators. They play different roles and undertake different tasks. More specifically, A learner is the candidate who takes the formal exam (summary assessment). Besides, for specific purpose (e.g., diagnostic or reflection), the learner may be persuaded by their personal agents, or even takes the initiative to perform some self-evaluations (formative assessment) during learning process. A tutor is the online course author or the teaching organizer who is responsible for preparing/setting the paper by taking advantage of the read-made test generation service in the test service layer, which has the capability of automatically generating test papers according to the requirements specified by the tutor. Without doubt, this process is significant and can largely alleviate the tutor's workload. In this dissertation, we differentiate between the two roles of tutor and evaluator. The reason is apparent that assessing hundreds of test papers is a burdensome work. To address such issue, an efficient

practice is to distribute these tests among different evaluators for scoring. Besides, in a large-scale online examination environment, several exam delivery centers are often staffed with administrators who are responsible for both the registration, authentication of all roles as well as the invigilation of examinations. All the role mentioned above are supported by their personal assistant agents (personal secretary) assisting in performing corresponding tasks.

In test services layer, the four core components: test generation, delivery, evaluation and publishing are integrated with each other and however designed as autonomous service providers that may run on different servers. In particular, the main feature of this architecture is that all the online assessment components are designed as several independent services represented by corresponding autonomous mobile agents that have the capability to migrate to the appropriate destinations and carry out local computation. This innovative paradigm and design philosophy bring several benefits such as bandwidth reduction, independent of connectivity of network, and enhance of robustness and flexibility, for instance, once the breakdown of one service has no affect to another services, as they are autonomous and distributed on different servers.

The data access layer provides the bridge to connect databases and agents that need particular data access services, which are, in fact, also implemented as utility agents.

The agent communication and management layer is the runtime environment that provides the common communication channel for interaction among agents as well as provides management service (e.g., white or yellow service).

6.3 PROTOTYPE DESIGN OF GENETIC ALGORITHM BASED MAS TEST GENERATION SYSTEM (GAMASTG)

Automatic test generating system in distributed computing context is one of the most important links in on-line evaluation system. Although the issue has been argued long since, there is not a perfect solution to it so far. This section proposed an innovative approach to successfully addressing such issue by the seamless integration of genetic algorithm (GA) and multi-agent system (MAS). In the design phase, test ontology was firstly defined for smoothing the communication among agents. For the implementation of GA, The fitness function and the structure of chromosome were identified on the basis of the analysis of constraint conditions associated with a test. To demonstrate the task execution flow and messages passing among agents, the activity diagram and sequence diagram were also shown on the AUML basis. In addition, the state chart is used to clearly illustrate state

transitions within the core agent GACtrlAgent that regulates the direction of evolution for each generation of population. On these models basis, we implemented the prototype in the next chapter, using JADE middle ware. The authentic, reliable simulation results reflect the feasibility and reasonability of the proposed models designed in this section.

6.3.1 INTRODUCTION

It is well known that in order to verify the learner's level of understanding and select corresponding educational strategy, the most popular measurement tool of learners' knowledge is a test. Therefore, the key issue is how to assure the efficiency and quality of a test. Obviously, besides the quality of item bank itself containing a large number of test question items, it also depends on an appropriate algorithm design. The test generation is a typical multiple variable and multiple objective optimization problem. Naturally, the genetic algorithm can be considered as the preferred alternative due to its capabilities of adaptive global optimization and intelligent parallel searching in non-linear solution space (Hammel et al. 1999). As compared with the traditional searching and optimization methods, GAs search a population of points in parallel without requiring derivative information or other auxiliary knowledge. Consequently, GA is highly suitable for such solving problems as automatic test generation.

Another subsequent issue worth taking into account is how to automatically generate tests in a distributed environment where diverse resources typically reside on different network nodes. This obviously relates to the network-computing paradigm, the traditional solutions to distributed applications are mostly based on client-server paradigm. Although this model has been dominant in networks for many years, there still exist some inevitable issues such as resources control, network bandwidth, latency, connectivity etc. Fortunately, the emerging MAS, specially the mobile agent system, seems to be a promising alternative. Compared with a client-server centralized system, the advantages of MAS include distribution of processing, support for a more flexible peer-to-peer model, decentralization of control, the reduction of network bandwidth use, etc.

The main contribution of this section is to propose a new methodology of integration of GA and MAS in order to address the issue of automatically generating tests. In GAMASTG (Genetic Algorithm Based Multi-Agent System Applied to Test Generation), the involved agents have been beforehand indoctrinated with the experience knowledge of GA and all the population individuals in GA are mapped into a group of individual agents representing a group of potential solutions. In particular, a special agent called GACtrlAgent was designed as a mobile agent that is able to migrate via Internet to the destination node where the item pool exists. More specifically, this agent has the capability

of constructing the evolutional environment and the authority for the determination on how the population evolves.

6.3.2 GENETIC ALGORITHM

GA is a computational model inspired by evolution in the search for solutions to complex problems. GAs operate on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. GAs assess the performance of individual members of a population with fitness function. This is done through a fitness function that characterizes an individual's performance in the problem domain. Thus, the fitness function establishes the basis for selection of pairs of individuals that will be mated together during reproduction. During the reproduction phase, each individual is assigned a fitness value derived from its raw performance measure given by the fitness function. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating whereas less fit individuals have a correspondingly low probability of being selected. Once the individuals have been assigned a fitness value, they can be chosen from the population, with a probability according to their relative fitness, and recombined to produce the next generation. Genetic operators manipulate the characters (genes) of the chromosomes directly. The crossover operator is used to exchange genetic information between pairs of individuals. It is applied with a probability when the pairs are chosen for breeding. A further genetic operator, called mutation, is then applied to the new chromosomes, again with a set probability. Mutation causes the individual genetic representation to be changed according to some probabilistic rule. Mutation is generally considered to be a background operator that ensures that the probability of searching a particular subspace of the problem space is never zero. This has the effect of tending to inhibit the possibility of converging to a local optimum, rather than the global optimum. After recombination and mutation, the individual strings are then, if necessary, decoded, the fitness function evaluated, a fitness value assigned to each individual and individuals selected for mating according to their fitness, and so the process continues through subsequent generations. Although the underlying mechanisms are simple, GA has proven itself as a general, robust and powerful search mechanism (Adeli et al. 1994; Cai et al. 1996;

Zitzler et al.2000).

6.3.3 TEST ONTOLOGY DESIGN

In recent years the development of ontology—explicit formal specifications of the terms in the domain and relations among them (Gruber 1993)—has been moving from the realm of artificial intelligence laboratories to the desktops of domain experts (Guarino et al. 1999; Holsapple et al. 2002; Rothenfluhh et al. 1996; Valente et al. 1999). It is well accepted that a common ontology holds the key to fluent communication between agents (Genesereth et al. 1994). As is known that, for agents to be able to communicate in a way that makes sense for them, they must share the same language and vocabulary. For a specific domain (e.g., e-examination), one must define its own vocabulary and semantic for the content of the communication between agents. This requires ultimately the definition of ontology.

In order to define the ontology of test generation, the key point is how to identify these application-specific concepts and relations among them so that agents in GAMASTG can ascribe the common meaning to them. Apparently, this is relevant to the constraint conditions associated with a test paper. Therefore, it is necessary to analyze the parameters of composing a test before the definition of ontology. Typically, the following procedures should be taken into account when considering setting a paper in a certain discipline:

- To identify the scope of knowledge points and their distribution proportions in a test.
- To identify the distribution of teaching requirements (can be represented by Bloom's (1956) taxonomy: knowledge, comprehension, application, analysis, synthesis, and evaluation).
- To identify the difficulty distribution, e.g., a test paper consisting of 10% of very difficult test items ,20% of r difficult ones, 40% of medium ones, 20% of easy ones and 10% of very easy Ones, respectively.
- To identify the structure of a test paper, i.e., the types of test items, the quantity of each type as well as its score.

Obviously, this is a multiple variables solving problem. It can be represented by a graphical mathematical model as shown in figure 6-2.

The axis X, Y and Z respectively represent knowledge point, difficulty level and teaching requirement, while the cubes with different colors and different weights respectively stand for different item types and different scores. The issue of constructing a test is just to select some appropriate cubes meeting the constraint conditions in each dimension in this mathematic model.

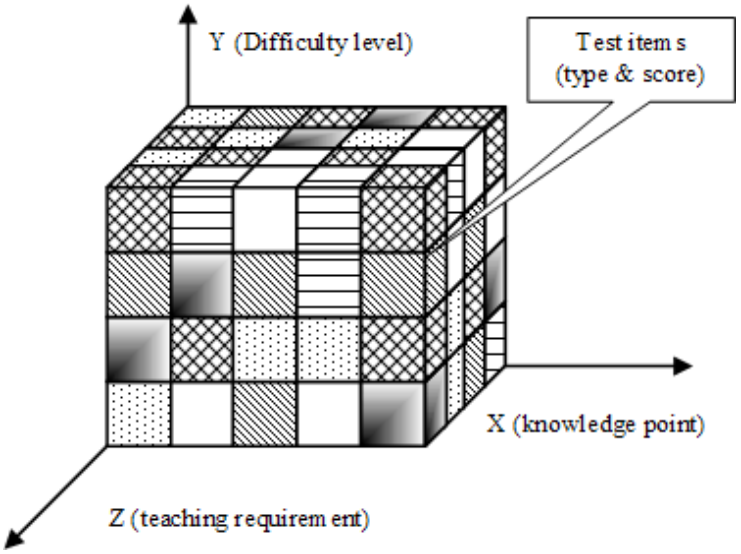


Fig. 6-2 Mathematical model of solutions space

Based on the above analysis, we defined the test ontology as shown in figure 6-3.

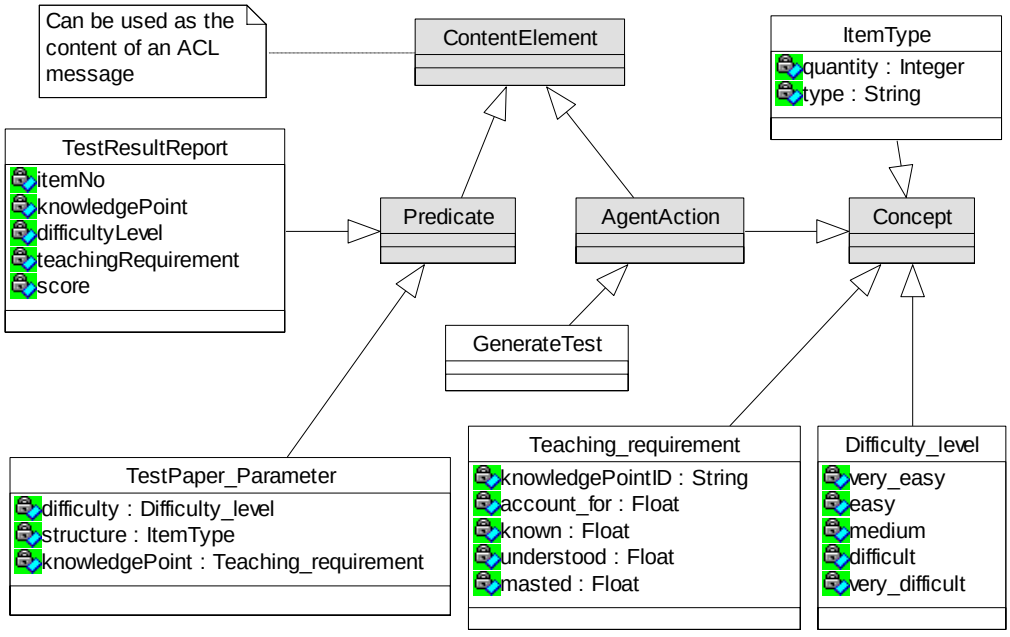


Fig. 6-3 Class diagram of Test ontology

This ontology describes the main elements that agents use to create the content of messages, e.g., application-specific predicates and actions. Predicates are expressions that indicate something about the status of the world and can be true or false. While agent actions indicate actions that can be performed by some agents. This ontology defines these

concepts such as test structure, score proportions of different types of test items with different difficult levels, knowledge points and teaching requirements etc. Once this ontology has been explicitly incorporated into the communication, it can be used to construct the message content, one of the key points for communication.

6.3.4 DESIGN OF GA

As far as GA is concerned, there are mainly two aspects worth consideration. One is the representation of a chromosome; the other is fitness function. As a result, the following discussion will focus on the strategies for them.

6.3.4.1 STRUCTURE OF CHROMOSOME

Individual chromosome represents a potential solution (i.e. a test paper in GAMASTG). Each test item is identified by its unique number. Consequently, each chromosome consists of some ordering number of items. From figure 6-4, we can see that such numbers with the same type are placed in the same segment; all the chromosomes have the same sequence and length.

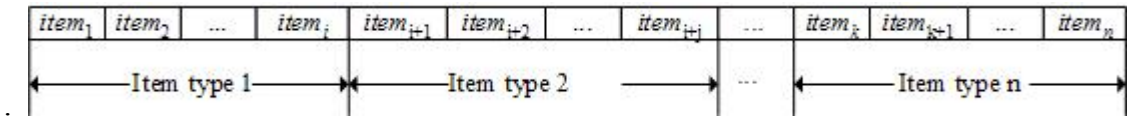


Fig. 6-4 Structure of chromosome

Apparently, such way of sequencing a chromosome makes easy to operate on the genetic operators (i.e. crossover and mutation). With regard to crossover, it is the most genetic operator contributing to GA. thanks to structure of the chromosome mentioned above; it is convenient to exchange crossover information between two chromosomes.

For example, given two original chromosomes A and B consisting of n items as shown below, they perform two-point crossover through exchanging part of test items.

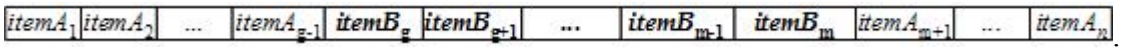
Original chromosome A:

Original chromosome B:

Suppose that two crossover points randomly selected from the parent chromosomes are g and m, respectively (where 1<g<m<n). All items between the two points should be

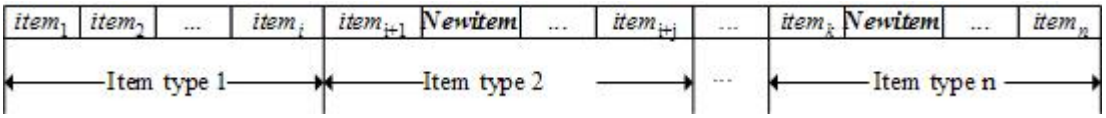
swapped according to standard two-point crossover operation, thus rendering two child chromosomes as follows:

Chromosome A after crossover



Chromosome B after crossover

For genetic operator mutation, the strategy adopted in this paper is to use a new created random item to replace the old corresponding one according as the mutation probability. The following chromosome is used to illustrate this operator.



6.3.4.2 OBJECTIVE FUNCTION AND FITNESS FUNCTION

The objective functions can be obtained as follows:

$$f_i(item_1, item_2, ..., item_n) = DL_i \quad i=1, 2\ldots p \qquad 6\text{-}1$$

$$g_i(item_1, item_2, ..., item_n) = KP_i \quad i=1, 2\ldots m \qquad 6\text{-}2$$

$$h_{ij}(item_1, item_2, ..., item_n) = KT_{ij} \quad i=1, 2, m; j=1, 2\ldots k \qquad 6\text{-}3$$

Where:

DLi: the percentage of difficulty level i in relation to the total score of a test.

KPi: the percentage of knowledge point i in relation to the total score of a test.

KTij: the percentage of teaching requirement j belonging to knowledge point i in relation to the score of knowledge point i.

p, m, k: denote the number of difficulty levels, knowledge points and teaching requirements, respectively.

The constraint conditions of DLi, KPi and KTij are shown below:

$$DLi=DLCi \quad KPi=KPCi \quad KTij=KTCij \qquad 6\text{-}4$$

Where:

DLCi, KPCi, KTLCij: are the expected value of DLi, KPi, KTij, respectively. Generally, they are constants specified by teachers or persons who set a test.

Evidently, these objective functions reflect a multiple variables optimization problem with multiple constraint conditions. They have to be converted into fitness function in order

to control the direction of evolution. To obtain the fitness function, two definitions are firstly made in formula (5) and (6) respectively.

$$E_1 = \sum_{i}^{p} \left| DLC_i - DL_i \right| \qquad 6\text{-}5$$

$$E_2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \left| KPC_i \times KTC_{ij} - KP_i \times KT_{ij} \right| \qquad 6\text{-}6$$

Where:

E_1: the sum of error between the given value DLC_i and the calculated value DL_i:

E_2: the sum of error between the given value $KPC_i \times KTC_{ij}$, and its corresponding calculated value.

Finally, the fitness function is defined as follows:

$$F(item_1, item_2, ..., item_n) = \begin{cases} \theta \quad if \ E_1 \geq 1 \quad or \quad E_2 \geq 1 \\ k_1(1-E_1) + k_2(1-E_2) \end{cases} \qquad 6\text{-}7$$

The most interesting feature in function F is the use of parameters k1, k2 respectively representing the weighting factors associated with E1 and E2 to which k1, k2 reflect the extent of importance attached. Based on large numbers of experiments, it is found that it is relatively more difficult to meet the constraint conditions on the teaching requirements and knowledge points, so more importance should be given to k2 so as to protect these individual chromosomes with good performance in teaching requirements and knowledge points. The other interesting point is that, in function F, the fitness is set as a positive real number approaching zero when a chromosome (i.e. a test paper) has too bad performance.

6.3.5 ARCHITECTURE

Figure 6-5 shows the proposed agent system architecture of GAMASTG, in which there exist several types of agents (mobile or static) that might be geographically dispersed on different internet nodes.

Tutor Personal Agent (TAA) is a personal assistant agent that performs some particular tasks on behalf of a teacher. For instance, it can help a teacher set up initial parameters, search other agents providing such service as composing tests, and eventually display or edit the final result etc. In order to facilitate the access to GAMASTG through common browsers, this agent can be downloaded on remote machines together with an applet and then run in local Java Virtual Machine.

Learner Self-evaluation Agent (LSA) provides the opportunity for a learner to take formative assessment. In this case, LSA will request Leaner Model Agent (see chapter 3) to

inform the prior cognitive status of relevant knowledge points as so to identify the blind spot of this learner. On this basis, LSA can ask for *Test Generation Service Agent* to compose a tailor-made test that adapt to the learner's knowledge status.

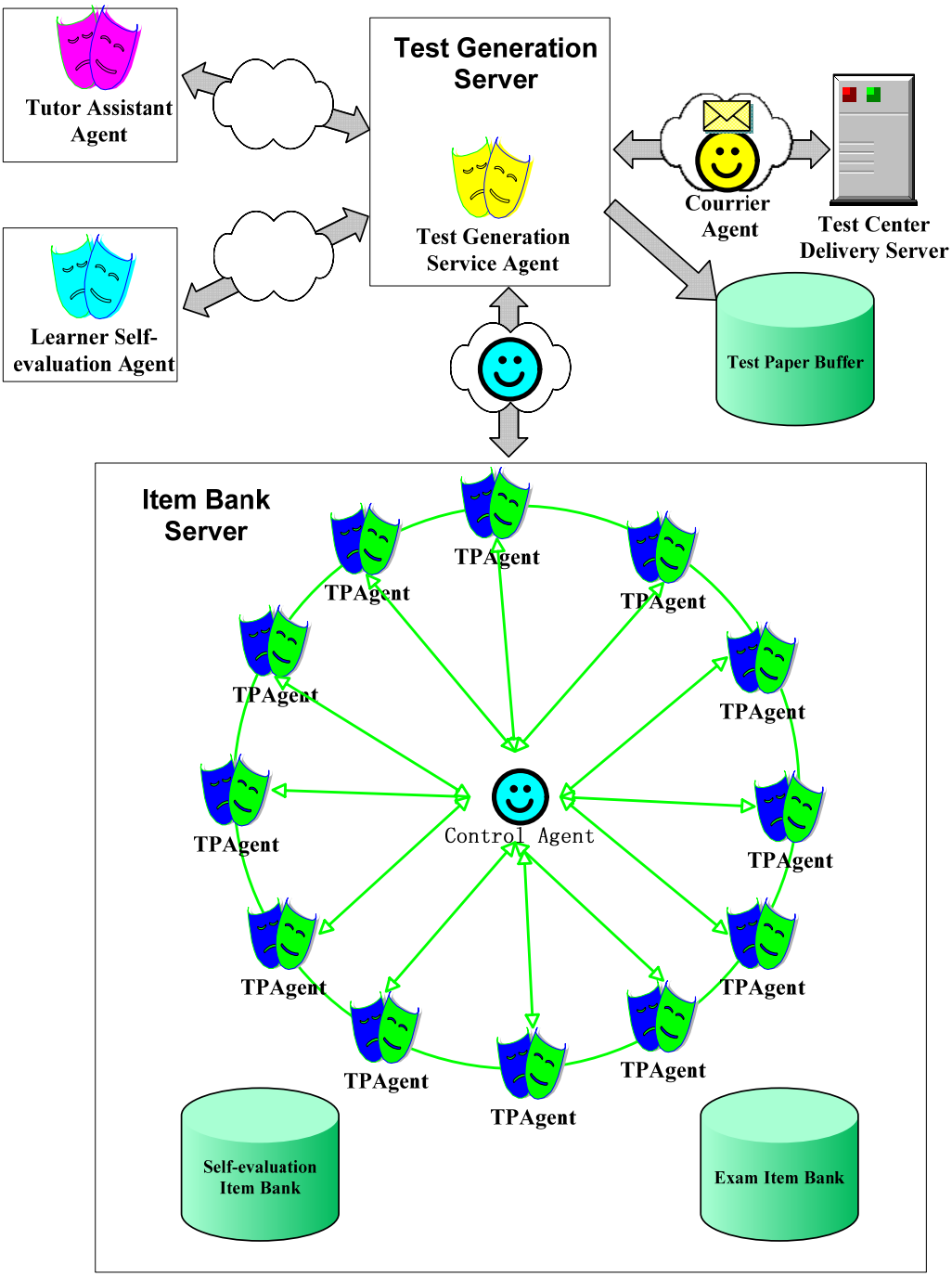


Fig. 6-5 Test Generation Architecture

Test Generation Service Agent (TGSA) is responsible for offering automatic test

generation service. It runs persistently on an test generation server that provides a runtime environment for all agents as well as manages their lifecycle. In fact, TGSA plays the role of agent factory, which possesses the ability of dynamically produce mobile agents that migrate to remote servers to perform particular tasks. Thus most of the concrete work will be delegated to those mobile agents.

Courier Agent (CA) is a mobile agent that is responsible for dispatching test papers to different Test Delivery Center Servers (TDCS). This agent is created by TGSA after the generated test paper is verified, edited, and confirmed by the test setter (i.e. tutor). Once TDCS is not available for some reasons(e.g, busy, or disconnection of network), such test papers can buffered in temporary database so as to wait for appropriate time to send CA to corresponding destinations.

GACtrlAgent is the core agent participating in the implementation of GA. It is created by TGSA when a request of generating tests arrives (from either TPA or LSA). The interesting point is that this agent is also designed as a mobile agent. Once obtaining initial parameters from TGSA, it will migrate via Internet to the nodes where the item banks reside. On arrival at the destination, it begins to construct the evolution environment by creating a population of individual agents called TPAgent. These TPAgents respectively represent a single potential solution (i.e. a test paper consisting of a set of test items with different types, difficulty, knowledge points, and test objective) in problem domain. After the evolution environment is successfully deployed, GACtrlAgent will gain the control (not entirely) over the genetic operations of each generation of population. In other word, GACtrlAgent begins to take charge of scheduling the genetic operations: selection, crossover and mutation. In particular, it can make some strategies to control how the next generation will evolve according to the evaluation of each TPAgent's performance. More exactly, for these TPAgents with good performance, they will win chance to reproduce themselves and survive in the next generation, while others with bad performance might be killed. During the process of evolution, all the TPAgents are the actual undertakers of GA, that is to say, they can perceive needed information from the outer world (both from GACtrlAgent and other TPAgents) carry out the genetic operators such as crossover, mutation or even suicide for instance. It is worth noting that TPAgents may interact with different databases, depending on the different requests. In particular, for requests from LSA, TPAgents retrieve test items from self-evaluation database, however, for request from TPA, the exam item bank applies. With regard to the detailed design and implementation of GAMASGT can be referred to the section 5 and chapter 7.

To clearly describe the activities among agents, a UML based activity diagram is

shown in figure 6-6.

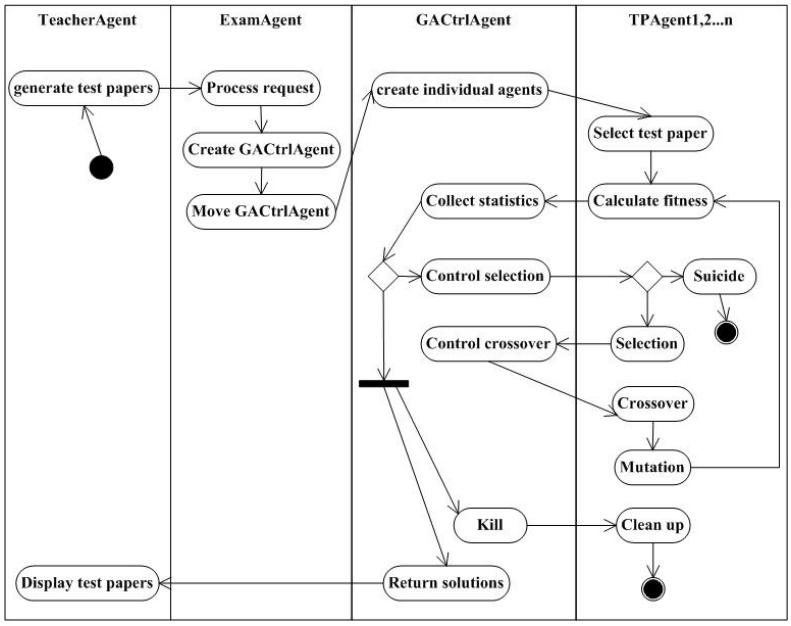


Fig. 6-6 Agent UML activity diagram of GAMASGT

6.3.6 STATE CHART

Agents have behaviors and state. The state of an agent depends on its current activity or condition. A state chart diagram shows the possible states of the agent during its life period and the transitions that cause a change in state. State Diagrams view agent objects as *finite* state machines that can be in one of a set of finite states and that can change its state via one of a finite set of stimuli. Figure 6-7 are used to clearly exibit the internal states and their transitions of GACtrlAgent and TPAgent.

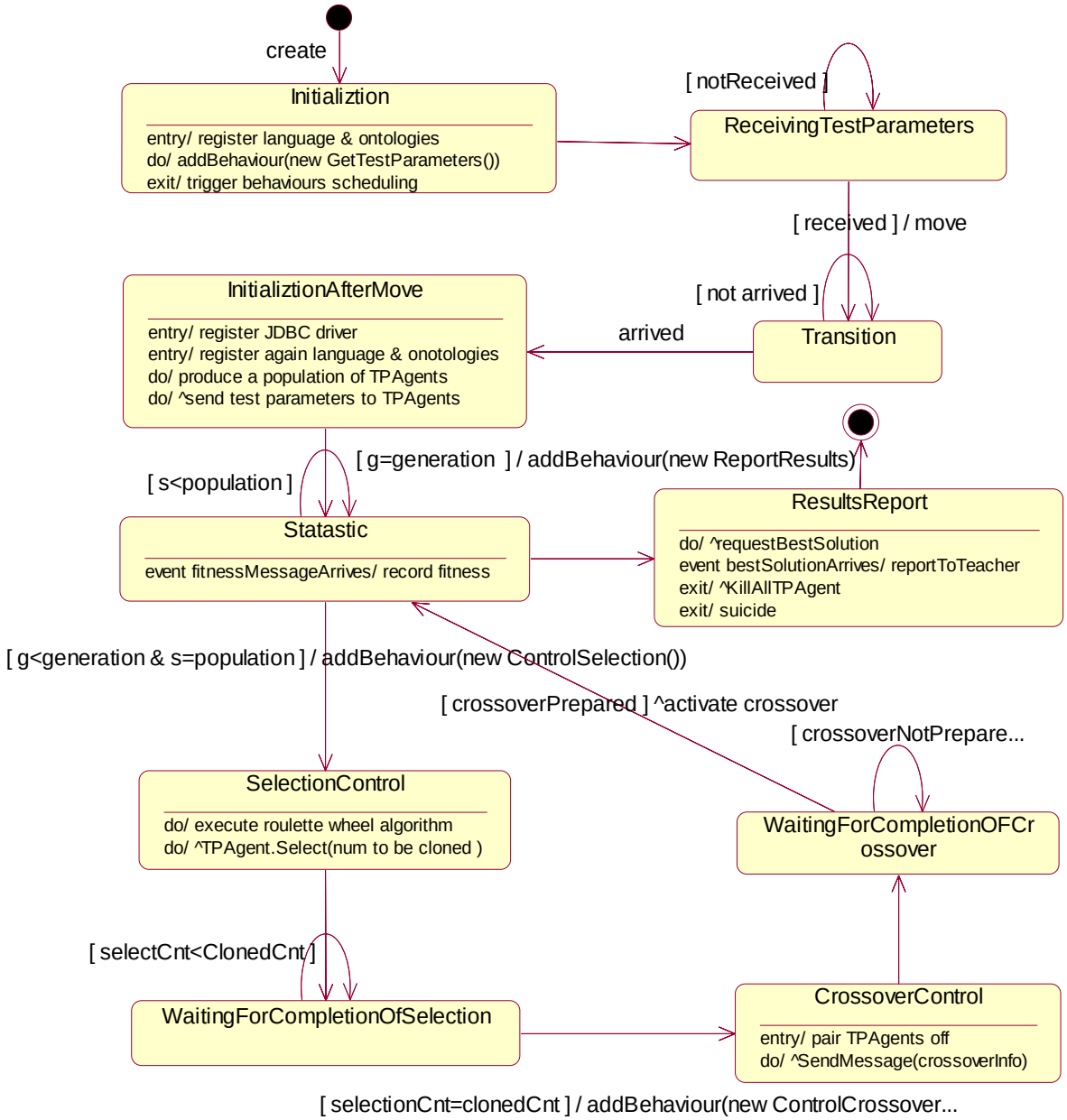


Fig. 6-7 Statechart of GACtrlAgent

6.3.7 INTERACTIVE MODEL

To illustrate the message passing among agents, an AUML based sequence diagram is shown (see figure 6-8).

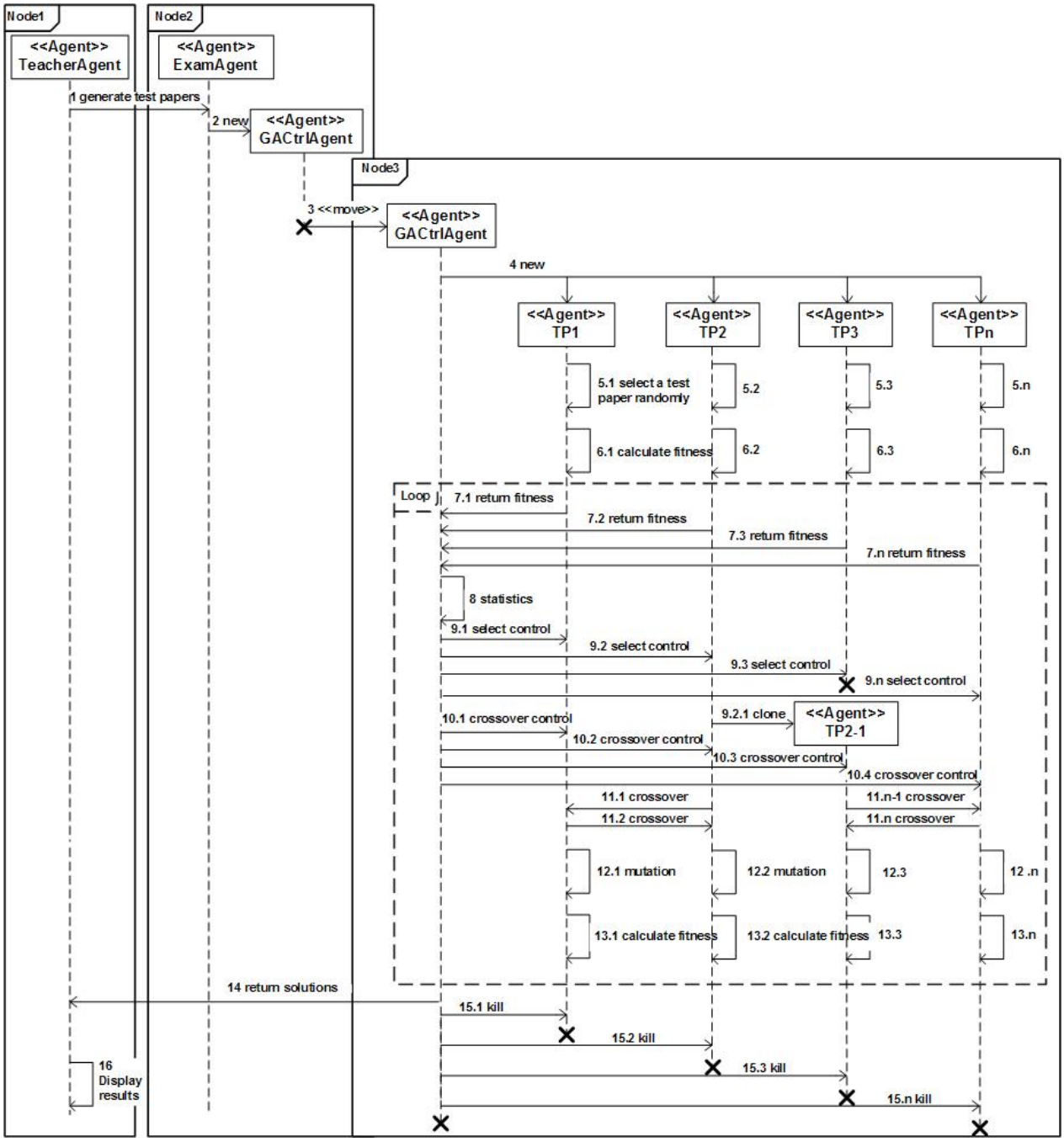


Fig. 6-8 Agent UML sequence diagram of GAMASTG

This diagram focuses on depicting the sequence of messages exchanged among agents, along with their corresponding event occurrences. The detailed interpretation of messages is as follows:

1. The TeacherAgent states these constraint conditions such as teaching requirements, knowledge points and difficulty levels to ExamAgent on the basis of TestOnology.

2. ExamCenterAgent creates a mobile agent called GACtrlAgent which acts as the role of GA dispatcher

3. ExamCenterAgent assigns GACtrlAgent to migrate through internet to the remote machine (i.e. node 3 where the item pool exists)

4. On arrival, GACtrlAgent begins to create a certain numbers of individual agents called TPAgents each representing a test paper, i.e., a potential solution.

5. TPAgents composes a test represented by a set of random item numbers according as test structure specified by the teacher and then extract item parameters from the local database

6. TPAgents calculates their fitness values respectively using the formula (7).

7. TPAgents reports their own fitness to GACtrlAgent

8. Having collected all fitness values from TPAgents, GACtrlAgent performs some essential statistic tasks such as the calculation of total fitness, average fitness, maximal fitness and minimal fitness etc.

9. GACtrlAgent determines the survival chance of the old generation of TPAgents in the new one applying the roulette algorithm [12].It is to say: those with the worst performance will be ordered to suicide themselves, while others with better performance will win opportunity to clone several times themselves.

10. After the selection operation is over, GACtrlAgent has to be responsible for the crossover genetic operation, more in detail, it has authority to pair TPAgents off and respectively send them the message contents with the crossover information, i.e., the agent name to exchange chromosome information with and the crossover position specifying the crossover point from where the latter of a chromosome will be used to exchange with the mating agent.

11. When requested to perform the crossover genetic operation, the mating TPAgents have to retrieve the test items to be exchanged from its own chromosome and send them to each other. Eventually a new chromosome will be reassembled with the exchanged test items.

12. Individual agents do the mutation operation according to their own mutation probability. In practical operation, when one item needs to be mutated, it is sufficient to replace this item with a new random item.

13. Once both the three genetic operations are over, the new generation of TPAgents again calculates their fitness values and the subsequent messages passing process will

repeat from step 7 to step 14 until the solutions eventually meet the desired expectation or the total generation arrives.

14. When obtaining a set of test papers, GACtrlAgent sends the best solutions to the teacher.

15. GACtrlAgent kills all the TPAgents including itself.

16. TeacherAgent displays the final results to the teacher.

It is worth noting that there are mainly two points of difference between the proposed notation of agent-oriented UML and the object-oriented UML sequence diagrams. First, the UML messages are sent from one Object to another in the form of method call while the AUML messages are sent from one Agent to another in the form of ACL. Second, most of the UML messages are synchronous where control is passed to called object until that method has finished running while the AUML messages are absolute asynchronous where the agent-readable message passing is somewhat similar to the human-readable e-mail mechanism. Besides, in this dissertation, we use the package *node* to represent a separate machine, *loop* to represent the iterative process of interaction, and the extended stereotype <<Agent>> to represent Agent object as well as <<move>> to represent one agent's migration from one machine to another.

The concrete design, implementation and simulation of GAMASTG can be referred to the next chapter.

6.4 DESIGN OF TEST DELIVERY

As shown in figure 6-9, this stage of test delivery mainly involves: (1) distribution of test paper to different test delivery center servers, (2) creation of Delivery Agents that are responsible for delivering test paper to distance learners (examinees), and (3) creation of Answer Agent that is responsible for compiling answers collected by delivery agents (note that these compiled answer paper will further be sent to Evaluation Server for final assessment).

It is obvious that the architecture shown below exhibits to great extent the flexibility as most of the activities in this stage are performed by mobile agents (such as Answer Agent, Courier Agent, and Delivery Agent). In particular, we describe in detail the three stages:

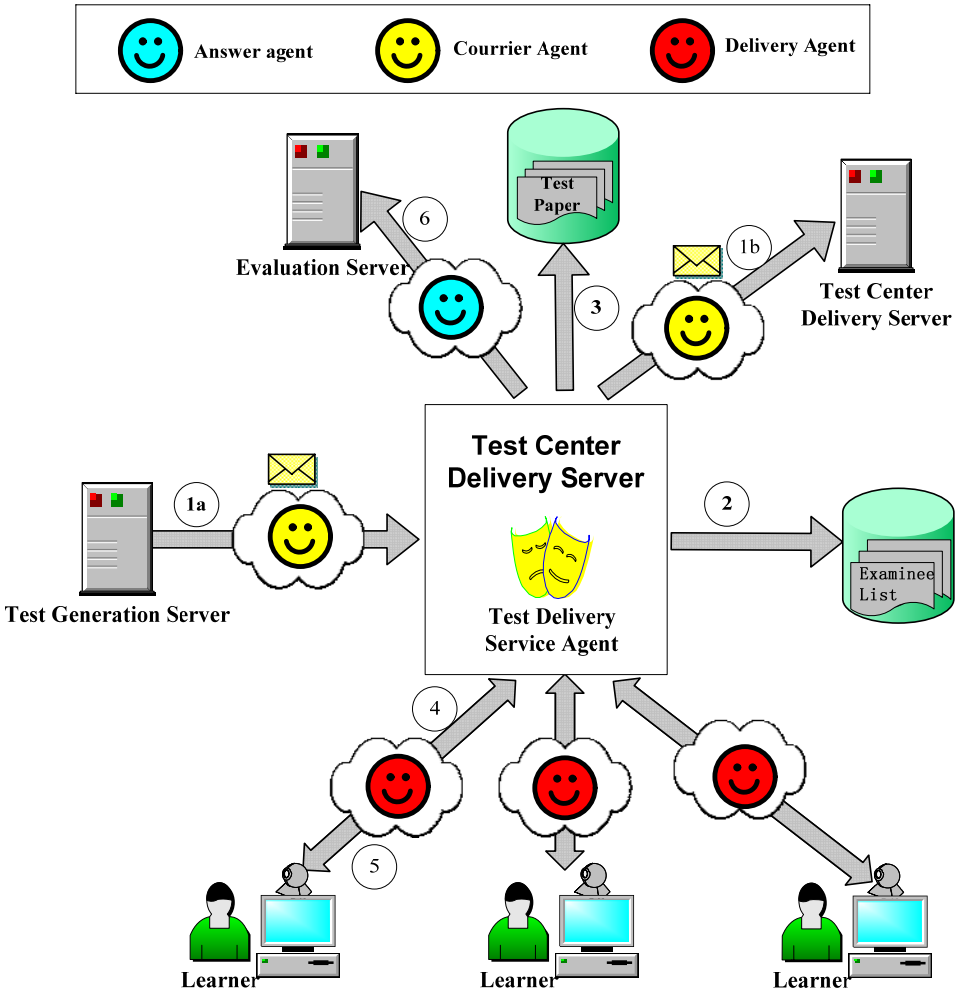


Fig. 6-9 Architecture of Test Delivery Service

(1) Distribution of test papers among Delivery Centers

In large-scale online test environment, there may exist several test delivery centers distributed on different network nodes at which learners register with the vicinal servers. As described in the previous section, the Courier Agent is created by the Test Generation Service Agent (TGSA). Upon supplied with prepared test paper and itinerary of the specified test delivery centers, this agent will migrate to the first Test Center Delivery Server (see step 1a) as long as this operation is permitted and authorized by Test Delivery Service Agent (TDSA) running within this server. After leaving over a copy of the test paper in the local databases (see step 3), the Courier Agent moves on to the next location (see step 1b). Upon completion of the itinerary, it returns to the Test Generation Server and terminates if no any abnormity occurs. Or else, the above process needs to be repeated. If the number of test delivery centers is large, more than one Courier Agent may be launched

in parallel.

(2) Creation of Delivery agents and testing

One of the tasks to be performed by Test Delivery Service Agent (TDSA) is to detect the exact moment when a scheduled assessment should be triggered. Of course, the time of assessment can be predetermined by tutors, administrators (in case of compulsory summary assessment), or even the learner himself (in case of self-evaluation). Once a compulsory assessment needs to be launched, the personal agent of each learner who has to take the assessment (according to the examinee list (see step 2) registered at this test delivery center) will receive one notification of information about the upcoming assessment. When any learner prepares well for taking the test, his/her personal agent has to respond with a message requesting the migration of Delivery Agent. Then TDSA begins to create and initiate one Delivery Agent per examinee. Once these Delivery Agents have extracted the specified test paper, they start to migrate via network to the corresponding learner's machine. If necessary, the Delivery Agent may carry with utility tools for purpose of facilitating the process of answering question items. Thanks to the mechanism of mobile agent paradigm, the learners can take offline test. During the testing process, the Delivery Agent presents the questions to a learner and records his/her answers. When the designated examination duration terminates or if the learner finishes ahead of schedule and wants to submit the results, the Delivery Agent returns to the Test Delivery Server with the answers (see step 5). It is worth noting that when the type of assessment is self-evaluation, for rapid feedback we assume that the test just consists of objective question items without any subjective item. In this situation, the test can be automatically evaluated by Delivery Agent and the learner can see the test result immediately.

(3) Creation of Answer Agents

The Test Delivery Service Agent needs to create another type of mobile agent ─Answer Agent— that is used to extract answers from the Delivery Agent and is later sent to the Evaluation Server. Note that while the Delivery Agent itself could be sent to the Evaluation Server, we use a separate Answer Agent to ensure security and anonymity. For example, learner machines may not be trusted hosts and the use of Delivery Agent hides information about the evaluation process from the learner Similarly, the Answer Agent hides learner details from the evaluators.

6.5 DESIGN OF EVALUATION & RESULT PUBLISHING

This stage involves: (1) evaluation of answer papers and (2) compilation and publication of test results (see figure 6-10).

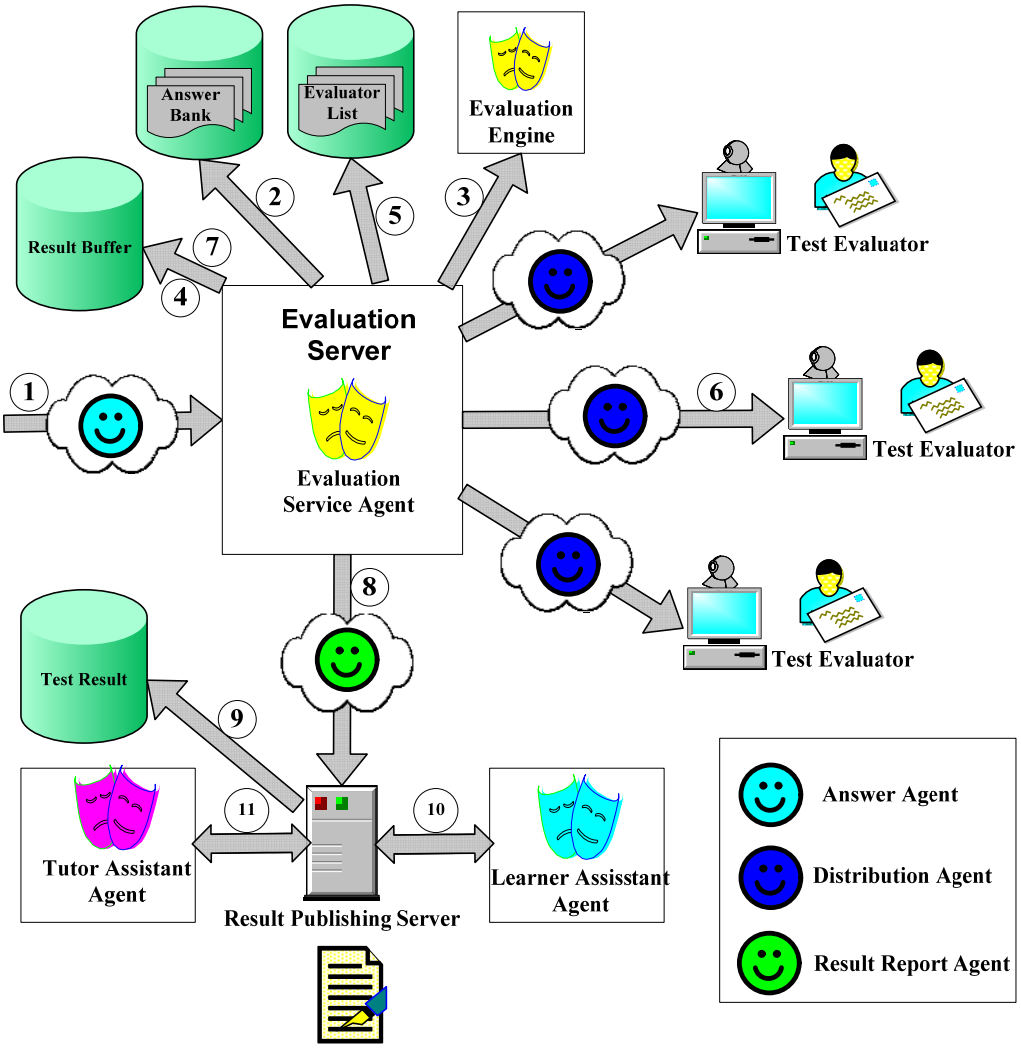


Fig. 6-10 Architecture of Evaluation and Result Publishing

(1) Evaluation of Answer Papers

If the type of assessment is mandatory formal examination, we assume that the test paper consists of both objective and subjective question items. As a result, the part of objective question items can be evaluated automatically by machine while the part of subjective items must be evaluated by human (i.e., evaluators). Accordingly, we describe the two types of evaluation process as follows:

- **Evaluation of objective question items:** the Answer Agent together with the "answers paper" arrives at the Evaluation Server (see step 1) with the permission of Evaluation Service Agent. The part of subjective question items (e.g., writing essay) in the answer paper will be saved into local database (see step 2) for the preparation of subsequent evaluation by distance test evaluators where appropriate. However, for the objective question items (e.g., true/false, multiple choice, matching etc.), they can be sent to the

evaluation engine (see step 3) where the correct answers associated with these question items have beforehand been stored. After Evaluation Engine finishes this evaluation process, the final results/scores need to be stored in result buffer database (see step 4).

- **Evaluation of subjective question items**: the division of work among several evaluators is especially of signification as it is unimaginable that a single tutor can evaluate hundreds of test papers. This situation is closely similar to the traditional pencil-pen-paper based large-scale examination. Whereas, the incorporation of mobile agent paradigm largely enhances the flexibility, independently of time, space, and connectivity of network. With regard to the issue how to distribute among evaluators which separate part within a test paper to be evaluated, it can be negotiated beforehand through their personal assistant agents. Once they come to consistent agreement, all relevant information will be registered with Evaluation Server. When everything goes smoothly, the manual evaluation process begins to happen. More specifically, Evaluation Service Agent sends messages to all the evaluators who are responsible for evaluation of one specified test paper, notifying that the answers to be evaluated are already ready. If any evaluator is available online and request to launch his/her evaluation process, Evaluation Service Agent start to create Allocatee Agent that is responsible for extracting, from the local answer bank, the predetermined portion of question items belonging to the share that the evaluator should do. After obtaining the destination address of the evaluator from registration info database (see step 5), the Allocatee Agent then moves to the evaluator's machine (see step 6), Once arrival, it presents a Graphical User Interface to the evaluator and prompts her to evaluate its answers. When the evaluator completes relevant evaluation work within allowable duration, Allocatee Agent will carry the evaluation result and returns to Evaluation Server. Eventually, the result is likewise stored into the result buffer database temporarily (see step 7), waiting for the appropriate time to later be sent to the Result Publishing Server. In particular, When all the answers have been evaluated, Evaluation Service Agent creates Report Agent that is responsible for assembling all the answers and then move to the Result Publishing Server (see step 8) with the permission of Result Publishing Service Agent.

(2) Publication of Results

After Report Agent arrives at Result Publishing Server, result data are persisted in local database. Eventually Publish Service Agent compiles analyses and publishes the final results. In particular, it sends messages including final test results to relevant personal assistant agents (see step 10, 11) on half of different users (tutor, or learner).

6.6 SUMMARY

The chapter put forward a new approach to building an e-assessment system applying MAS, especially Mobile Agent. Some innovative points can be summarized below:

1. This chapter proposed a new methodology and a prototype of GAMASTG., more specifically,

- The items numbering method inside chromosome assure the facilitation of operate on genetic operators.
- The seamless integration of GA and MAS can not only address the multi variable optimization problems of convergence speed and multiple constraint conditions, but also makes full use of social agents capabilities to solve distributed computing problems in an elegant manner.
- The introduction of test ontology makes all involved agent understand well each other in term of composing test requirements.
- GACtrlAgent is designed as a mobile agent; this brings many advantages such as reducing network bandwidth, latency, etc.

2. For the other links (i.e., test delivery, evaluation and result publishing) in the e-assessments, the core functionality is mostly carried out by relative mobile agents, as compared with the traditional client-server computing paradigm, the advantage is obvious as describe in chapter 2 such as:

- Communication latency and bandwidth:
- Asynchronous execution
- Protocol encapsulation
- Parallel execution

CHAPTER 7 MAS IMPLEMENTATION & SIMULATION BASED ON JADE FRAMEWORK

To verify and validate the feasibility and efficiency of the models proposed in this thesis, this chapter implemented and simulated part of the models with the JADE framework. Especially, three typical applications are implemented. First, we implemented the simplified prototype of GAMASTP for the purpose of synthetically revealing how to concretely implement a complex multi-agent system, which is concerned with several key issue: how to implement test ontology and apply to the communication among agents; how to design and implement agent behavior model according to the previous models; how to deploy agents over different network nodes. The second application is implemented for the purpose of how the learner model agent updates the learner model upon receiving the refresh data as well as how to answer any questions from external agents, this example also showed the application of interactive protocols such as FIPA request and FIPA query. The third example is used to implement part of the peer help system aiming at demonstrating the process how to find appropriate competent peer learners. The simulation results show the feasibility and efficiency of the models proposed in this dissertation.

7.1 INTRODUCTION

The purpose of this chapter is to implement part of models proposed in previous chapters in order to verify and validate their feasibility and efficiency. In the phase of implementation, an important decision is to find the most suitable programming language. Since we need a platform independent and on-line accessible system we choose Java, so we needed a Java based agent system. Moreover, we want the agent system to be FIPA -compliant since FIPA is the international organization responsible for standardizing the agent based technology. We also wanted a system that is in growing and has possibilities to be extended in the future. We found JADE framework (JADE 2005; Chmiel et al. 2004) to be the best choice and the most suitable solution for our needs. The main reason for this selection is that JADE is a widely adopted platform within the software agent development and research communities. It is open-source and full FIPA compliant and runs on a variety of operating systems including Windows and Linux. With support from JADE, it simplifies the implementation and deployment of MAS based applications to a great extent. In the next subsequent sections, we will pay more attention to presenting our experience of how to

use JADE features to implement these models (i.e., GAMASTP, peer help system, and the learner model agent) proposed in previous chapters in order to exhibit different approaches to the practical agent design.

7.2 JADE

The section will introduce JADE middle ware that we chose to develop and validate our applications.

7.2.1 INTRODUCTION

JADE (Java Agent Development Framework) is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase. JADE agent platform tries to keep high the performance of a distributed agent system implemented with the Java language. In particular, its communication architecture tries to offer flexible and efficient messaging, transparently choosing the best transport available and leveraging state-of-the-art distributed object technology embedded within Java runtime environment. All agent communication is performed through message passing and the FIPA ACL is the language that is used to represent the messages. Each agent is equipped with an incoming message box and message polling can be blocking or non-blocking with an optional timeout. Moreover, JADE provides methods for message filtering. The developer can apply advanced filters on the various fields of the incoming message such as sender, performative or ontology. FIPA specifies a set of standard interaction protocols such as FIPA-request, FIPA-query, etc. that can be used as standard templates to build agent conversations. For every conversation among agents, JADE distinguishes the role of the agent that starts the conversation (initiator) and the role of the agent that engages in a conversation started by another agent (responder). According to the structure of these protocols, the initiator sends a message and the responder can subsequently reply by sending a not understood or a refuse message indicating the inability to achieve the rational effect of the communicative act, or an agree message indicating the agreement to perform the communicative act. When the responder performs the action he must send an inform message. A failure message indicates that the action was not successful. JADE provides ready-made behavior classes for both roles, following most of the FIPA specified interaction protocols. In JADE, agent tasks or agent intentions are implemented through the use of behaviors. Behaviors are logical execution threads that can be composed in various ways to achieve complex execution patterns and can be initialized, suspended and spawned at any given time. The agent core

keeps a task list that contains the active behaviors. JADE uses one thread per agent instead of one thread per behavior to limit the number of threads running in the agent platform. A scheduler, hidden to the developer, carries out a round robin policy among all behaviors available in the queue. The behavior can release the execution control with the use of blocking mechanisms, or it can permanently remove itself from the queue in run time. Beside, JADE uses an agent model and a Java implementation that offer a good runtime efficiency and software reuse. The following is the list of features that JADE supports the agent development:

- Distributed agent platform. The agent platform can be split among several hosts (provided they can be connected via RMI). Only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as Java threads and live within *Agent Containers* that provide the runtime support to the agent execution.
- Graphical user interface to manage several agents and agent containers from a remote host.
- Debugging tools to help in developing multi agents applications based on JADE.
- Intra-platform agent mobility, including transfer of both the state and the code (when necessary) of the agent.
- Support to the execution of multiple, parallel and concurrent agent activities via the behavior model. JADE schedules the agent behaviors in a non-preemptive fashion.
- FIPA-compliant Agent Platform, which includes the *AMS* (*Agent Management System*), the *DF* (*Directory Facilitator*), and the *ACC* (*Agent Communication Channel*). All these three components are automatically activated at the agent platform start-up.
- Many FIPA-compliant DFs can be started at run time in order to implement multi-domain applications, where a domain is a logical set of agents, whose services are advertised through a common facilitator. Each DF inherits a GUI and all the standard capabilities defined by FIPA (i.e. capability of registering, deregistering, modifying and searching for agent descriptions; and capability of federating within a network of DF's).
- Efficient transport of ACL messages inside the same agent platform. In fact, messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When crossing platform boundaries, the message is automatically converted to/from the FIPA compliant syntax, encoding, and transport protocol. This conversion is transparent to the agent implementers that only need to deal with Java objects.
- Automatic registration and deregistration of agents with the AMS.
- FIPA-compliant naming service: At start-up, agents obtain their GUID (Globally Unique IDentifier) from the platform.

- Support for application-defined content languages and ontologies.

7.2.2 JADE ARCHITECURE

The JADE Agent Platform complies with FIPA specifications and includes all those mandatory agents that manage the platform, that is the ACC, the AMS, and the DF. All agent communication is performed through message passing, where FIPA ACL is the language to represent messages. The software architecture is based on the coexistence of several Java Virtual Machines (VM) and communication relies on Java RMI (Remote Method Invocation) between different VMs and event signaling within a single VM. Each VM is a basic *container* of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. In principle, the architecture allows also several VMs to be executed on the same host; however, this is discouraged because of the increase in overhead and the lack of whatever benefit. Each agent container is a multithreaded execution environment composed of one thread for every agent plus system threads spawned by RMI runtime system for message dispatching. A special container plays the front-end role, running management agents and representing the whole platform to the outside world. A complete Agent Platform (AP) is then composed of several agent containers as shown in Figure 7-1. Distribution of containers across a computer network is allowed, provided that RMI communication between their hosts is preserved.

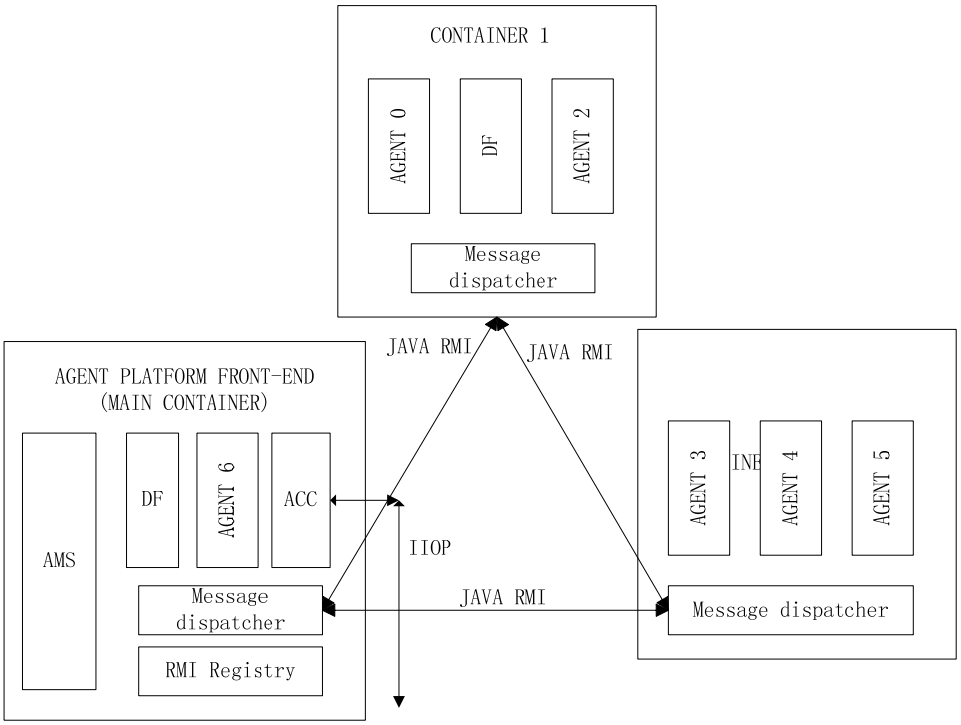


Fig. 7-1 JADE architecture

Each Agent Container is an RMI server object that locally manages a set of agents. It controls the life cycle of agents by creating, suspending, resuming and killing them. Besides, it deals with all the communication aspects by dispatching incoming ACL messages, routing them according to the destination field (:receiver) and putting them into private agent message queues; for outgoing messages, instead, the Agent Container maintains enough information to look up receiver agent location and choose a suitable transport to forward the ACL message.

The agent platform provides a Graphical User Interface (GUI) for the remote management, monitoring and controlling of the status of agents, allowing, for example, to stop and restart agents. The GUI allows also to create and start the execution of an agent on a remote host, provided that an agent container is already running. The GUI itself has been implemented as an agent, called RMA (Remote Monitoring Agent). All the communication between agents and this GUI and all the communication between this GUI and the AMS is done through ACL via an ad hoc extension of the FIPA-agent-management ontology.

7.3 IMPLEMENTATION OF GAMASGT

7.4 IMPLEMENTATION OF TEST ONTOLOGY WHITH PROTEGE

As described in chapter 7, for agents to be able to communicate in a way that makes sense for them, they must share the common ontology within the content language. To handle easily inside an agent, it is obvious that the information content needs to represented as JAVA objects. However, theses easily manipulated objects have to be conversed into a string or a sequence of bytes within the content slot for easy to transfer. Fortunately, This conversion process is automatically performed in JADE (see figure 7-2)

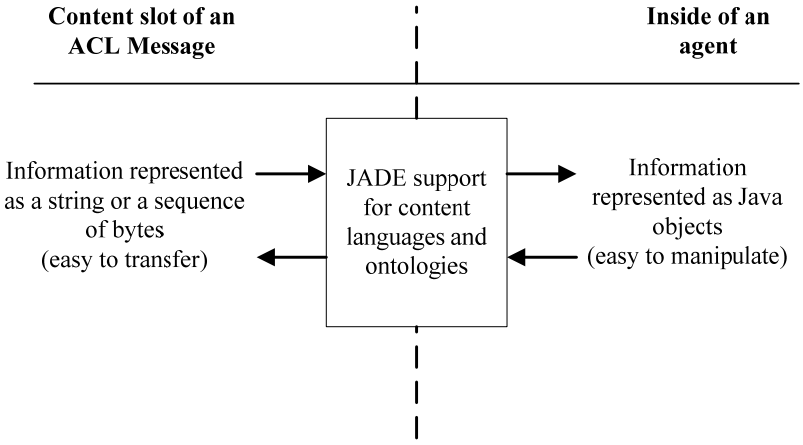


Fig. 7-2 The conversion performed by the JADE support for content languages and ontologies

A boring problem is that it is rather time consuming when implementing ontology since we have to develop all definition classes (i.e., the schemas) for each predicate, action and concept included in the test ontology. Fortunately, thanks to a plug-in called Beangenerator developed by the University of Amsterdam (C.J. Van Aart 2000), it is possible to define the test ontology using Protégé and make the well-defined ontology classes to be created automatically. Figure 7-3 shows the screenshot applying Protégé to the implementation of test ontology. With the Protégé tool, it is possible to allow developers to reuse domain ontologies and problem-solving methods, thereby shortening the time needed for development and program maintenance.

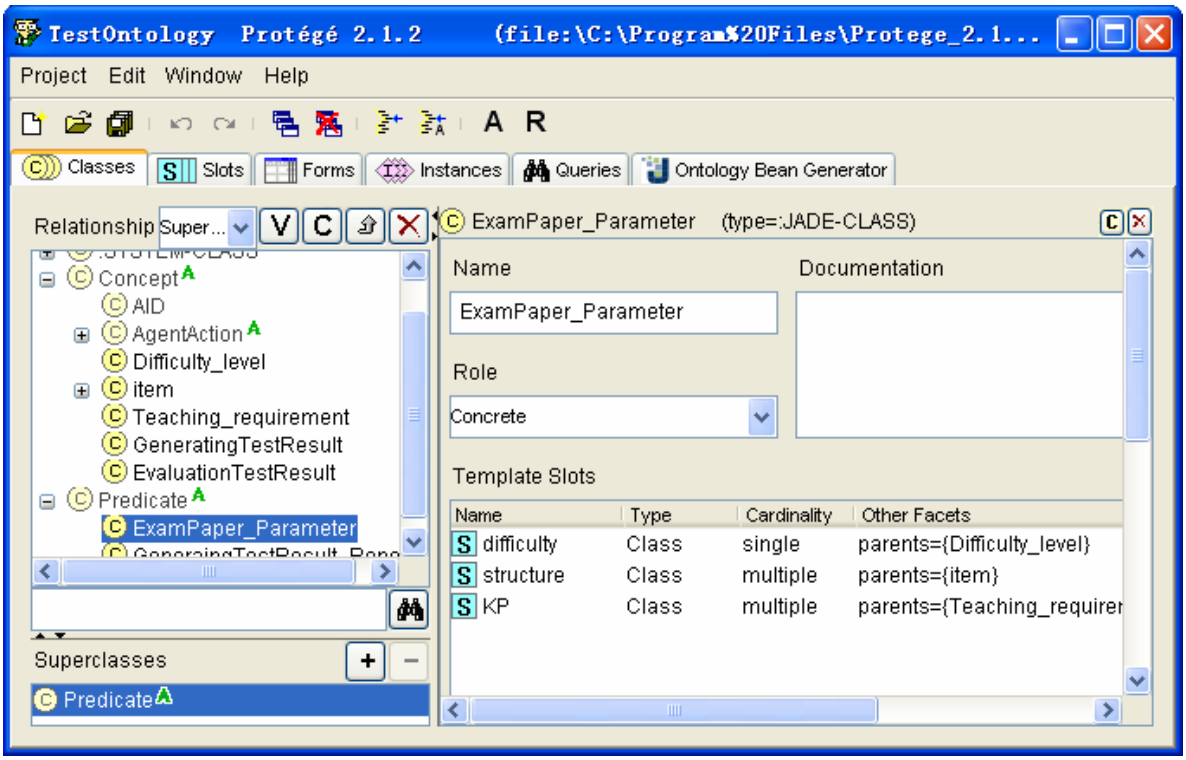
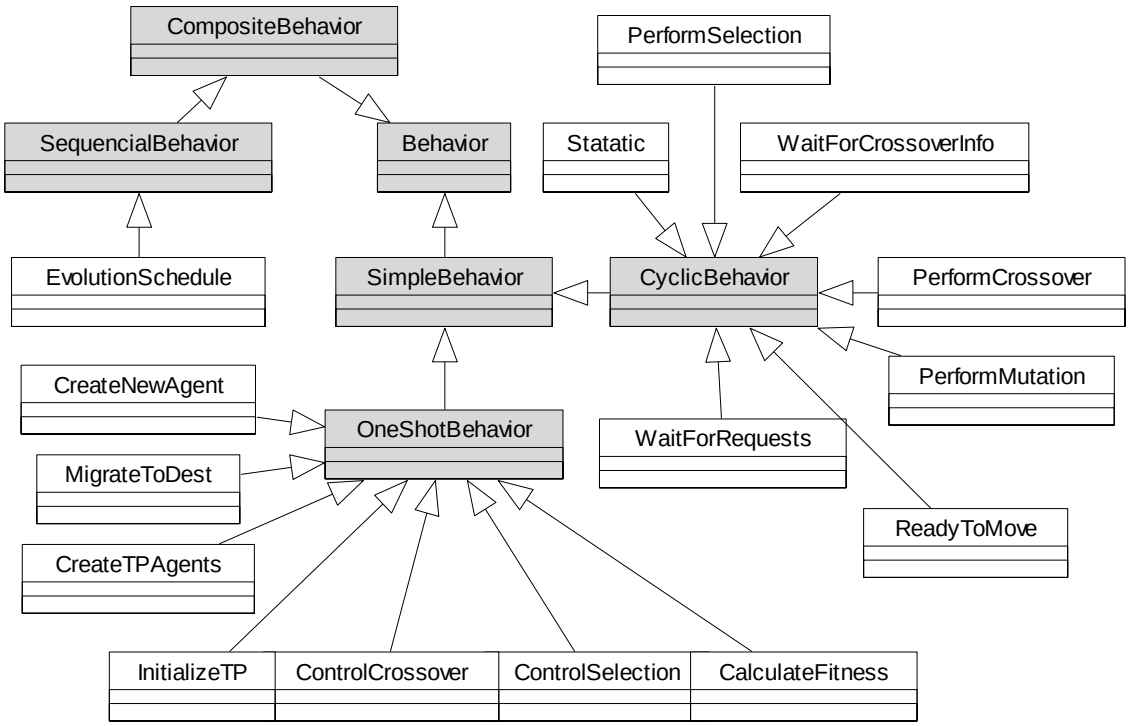


Fig. 7-3 A screenshot applying Protégé to the implementation of test ontology

7.4.1 7.4.1 DESIGN OF AGENT BEHAVIOR MODEL

The JADE behavior model allows an agent to execute several parallel tasks in response to different external events. In order to make agent management efficient, every JADE agent is composed of a single execution thread and its hidden scheduler carries out a round-robin non-preemptive policy among all behaviors in the active behaviors queue.

Figure 7-4 shows the behavior class hierarchy of GAMASTG agents based on the .JADE behavior model (i.e. these classes in gray) in which the SequentialBehavior is a composite behavior that executes its sub-behaviors sequentially and terminates when all sub behaviors are done. Therefore, the actual operations performed by executing this behavior are defined in its children behaviors. The abstract class CyclicBehavior models atomic behaviors that must be executed forever. The abstract class OneShotBehavior models atomic behaviors that must be executed only once and cannot be blocked.



.Fig. 7-4 UML model of behavior class hierarchy

The following table is the function descriptions of each agent in GAMASTG and all the tasks have been implemented as the JADE behaviors.

Table 7-1: Behavior types of each agent in GAMASTG

Agent name	Behavior name & type	General function descriptions
ExamCenterAgent	WaitForRequests : CyclicBehviour	This behavior waits for the requests from TeacherAgents that may want to compose a test paper on behalf of a teacher. Besides, it is responsible for retrieving control parameters associated to a test according to TestOntology.
	CreateNewAgent: OneshotBehaviour	This behavior is dynamically created within the end method of WaitRequest behavior. It is responsible for creating a new agent, i.e., the core agent GACtrlAgent.
GACtrlAgent	ReadyToMove :CyclicBehavior	Before GACtrlAgent migrates, this behavior is responsible for some initialization work.

	MigrateToDest: :OneshotBehavior	This behavior will make GACtrlAgent migrate via Internet to the destination node where the item pool exists upon receiving request from ExamCenterAgent.
	CreateTPAgents : OneshotBehavior	This behavior produces a population of TPAgents in order to construct the evolution environment.
	Statatic : CyclicBehvior	This behavior performs some statistic work such as the calculations of total fitness, average fitness, maximal fitness and minimal fitness in each generation. According to these statistic, it has the authority of determine the whole lifecycle of GA
	EvolutionSchedule :SequentialBehavior	This behavior is used to schedule its sub-behaviors in order to make them execute sequentially.
	ControlSelection :OneShotBehavior	A sub behavior of EvolutionSchedule behavior, it uses the Roulette Wheel Algorithm to direct the survival probability of old TPAgents in the next generation.
	ControlCrossover :OneShotBehavior	A sub behavior of EvolutionSchedule behavior, it is executed in such way that it randomly pairs all the TPAgents off and determines the crossover location. It is worth noting that not all TPAgents need to mate, this depends on the crossover probability.
TPAgents	InitializeTP :OneShotBehavior	This behavior randomly extracts a test paper from the local database. The test paper represents an initial solution to problem domain.
	CalculateFitness : OneShotBehvior	This behavior is responsible for the calculation of fitness value of each TPAgent according to formula (7)
	PerformSelection : CyclicBehaviour	This behavior concretely performs the selection genetic operation. It might clone or kill its owner agent (i.e. TPAgent). This depends on the order from GACtrlAgent
	WaitForCrossoverInfo :CyclicBehavior	This behavior receives crossover information from GACtrlAgent and prepares the test items to be exchanged as well as sends them to the mating TPAgent.
	PerformCrossover : CyclicBehaviour	This behavior concretely performs the crossover genetic operation. That is to say, it is responsible for reassembling the chromosome.
	PerformMutation : CyclicBehaviour	This behavior performs the mutation genetic operation according to it own mutation probability

7.4.2 7.4.2 AGENT IMPLEMENTATION

7.4.3 7.4.2.1 GENERIC AGENT INTERNAL ARCHITECTURE

All the agents that have been implemented in MAGE have the same internal architecture as shown below. From figure 7-5, we can see that all the MAGE agents on JADE basis have the following characteristics:

- agents are autonomous since each agent controls its own thread of execution and has a private proxy of the life-cycle manager.
- Each agent decides itself when to read the incoming messages and which messages to read
- Other agents have no way to get the agent object reference because of the mechanism of asynchronous message communication
- Different behaviors can be executed concurrently; the scheduler of behaviors carries out a round- robin non-preemptive policy among all the behaviors in the ready behavior queue.
- The MAGE agents create their capabilities through the decomposition of task into subtasks and implementation of them within particular behaviors. It is worth noting that some behaviors are put into behavior queue in the initialization phase while others are dynamically created according to the change of an agent's belief or in case of the need of behaviors cooperation

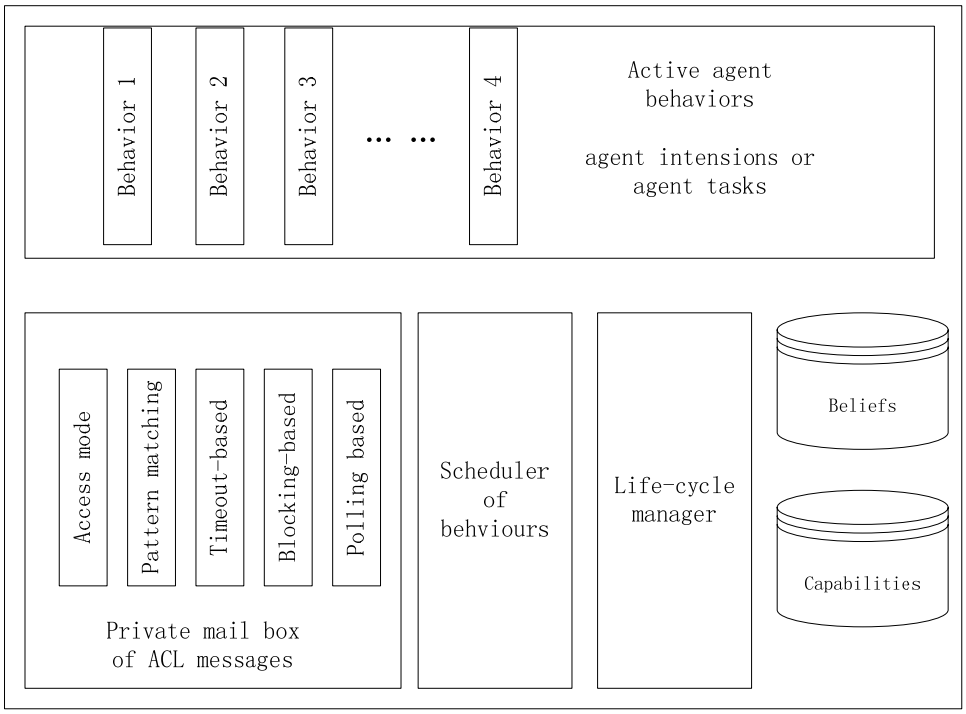


Fig. 7-5 Generic agent internal architecture

7.4.4 7.4.2.2 IMPLEMENTATION OF TEACHER AGENT

(1) Building Teacher Agent with integrated GUI

As is known that, in the Java programming language, a GUI runs on its own thread (the event-dispatching thread) that allows it to handle and react promptly to events that are generated whenever the user interacts with the GUI via a component such as pressing a button or resizing the window. On the other hand, an agent program runs on its own execution thread, which allows it to handle its behaviors. Because it is not efficient to let one thread call directly the methods of the other thread, JADE has provided an appropriate mechanism to manage interactions between the two threads when integrating a GUI with an agent.

The mechanism is simply based on event passing. Let us see how this mechanism works by considering the two directions of the interaction beween a GUI and an agent.

- **The agent interacting with the GUI** - A GUI has already a built-in mechanism of handling event which is implemented via the *actionPerformed()* method of every component that is registered with an *ActionListener* object. To register a component of the GUI with an *ActionListener* object, you either make your GUI implements the *ActionListener* interface and then register all interactive components of your GUI such as buttons with this *ActionListener* via the method *addActionListener* or for each of the interactive components, we can anonymously create an *ActionListener* object and add it to the component by passing it in argument to the same method *addActionListener()*. Whenever a call to the GUI is made, an *ActionEvent* is generated by the source component, that invokes the *actionPerformed()* method. And according to the code provided within the *actionPerformed()* method, the GUI responds by processing the event. When your agent program interacts with the GUI, it just calls the method provided within the GUI program that activates this mechanism.

- **The GUI interacting with the agent** - JADE has provided the abstract class *GuiAgent* that extends the Agent class. This class has two specific methods: *postGuiEvent()* and *onGuiEvent()*. These are the two methods that allow to handle the interactions between a GUI and an agent program. To be able to use these methods, the agent program must extend the *GuiAgent* class. Then we must provide the necessary code within the *onGuiEvent()* method that the agent will use to receive and process events that are posted by the GUI via the method *postGuiEvent()*. we may view the *onGuiEvent()* method as the equivalent of the *actionPerformed()* method in the GUI. When an agent program extending the *GuiAgent* class starts, it launches a specific behavior - the *GuiHandlerBehaviour* - that

handles incoming events from the GUI and dispatches them to the appropriate handlers, following exactly the same mechanism as in the GUI. To post an event to the agent, the GUI simply creates a GuiEvent object, adds the required parameters and passes it in argument to the method *postGuiEvent()*. Since this method belongs to the GuiAgent class, it is necessary to provide the GUI with a reference to the agent class on which the GUI can invoke that method.

Figure 7-6 shows GUI of Teacher Agent, which facilitate the teacher to configure his or her preferred test paper parameters, the details can be referred to the previous chapter.

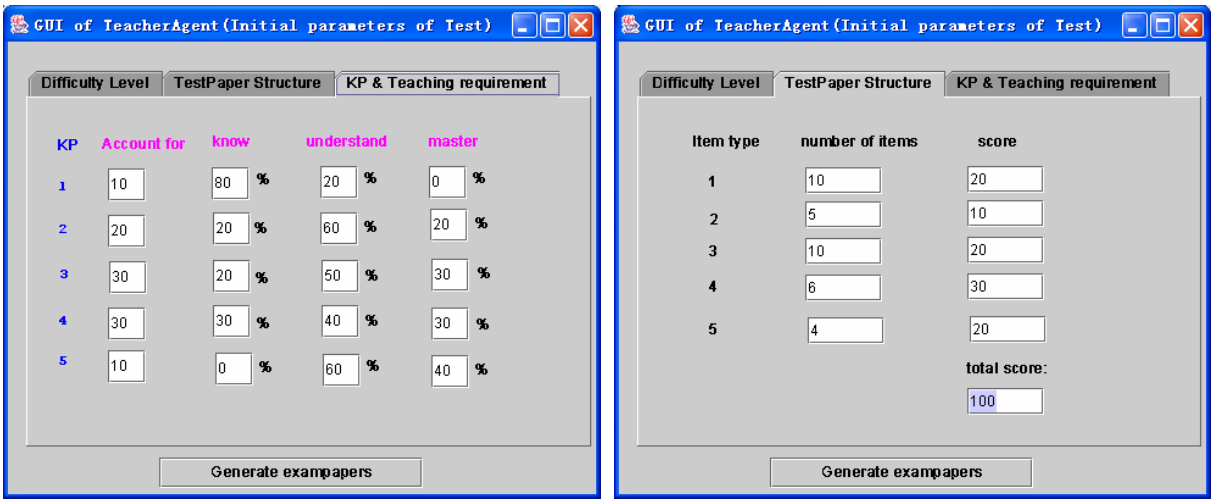


Fig. 7-6 The GUI of a Teacher Agent

(2) Task Implementation

Table 1 described in detail the main behaviors and core methods of Teacher Agent designed based on JADE framework. All the behaviors and methods listed in this table are triggered or activated by either particular events or incoming messages from other agents. It is worth noting that some behaviors are launched in the initialization phase while some behaviors are added to the behavior scheduler of this agent in dynamical fashion, depending on the agent's belief or task. The column task decomposition indicates in detail the main subtasks designed in the corresponding behavior or automatic called methods. Note that the implementation of other agents in GAMASTP is also represented as similar format of task decomposition.

Table 7-2: Task implementation of Teacher Agent

Behaviors Methods	Triggered Event	Task decomposition
Setup()	When TeacherAgent is created, this method is automatically triggered.	This method performs initiations and addition of initial behaviors. For GACtrlAgent, it is used to 1. register content language and ontologies. in particular, we have to register: • agent content language, i.e. SLCodec • domain ontology i.e. ExamPaperOntology 2. add to behavior scheduler the following agent behaviors: ResultShowBehavior
onGuiEvent()	When GUI event arrives, this method is triggered.	1. Extract the test paper parameters from the incoming event object 2. send message containing the test paper parameter to Test Generation Service Agent and request it to compose a test paper according to the submitted constraint requirement
ResultShowBehavior	When the incoming messages satisfy the following condition: Sender=Test Generation Service Agent ConservatonID= ”result”	1. Extract message content and obtain the final test paper according to the constraint requirements specified by the teacher 2. display the compiled test papers in the teacher’s browser in order to assist the teacher in viewing, modifying, and saving the final test paper on delivery Servers by courier agent created by Test Generation Service Agent

(3) Deploy in Browser

To allow the service to run on client’s machine with a minimum (or no) installation required. The JADE system proved to be a very good solution since it offers the possibility for agents to live on the client's machine without them needing more than a Java enabled web browser to be installed on that machine. The adopted solution was to create an applet on the client’s browser. Using the JVM, created by the browser to run the applet, a new agent container is created on the client’s machine using JADE API calls; figure 7-7 illustrates position of a Teacher Agent in the browser.

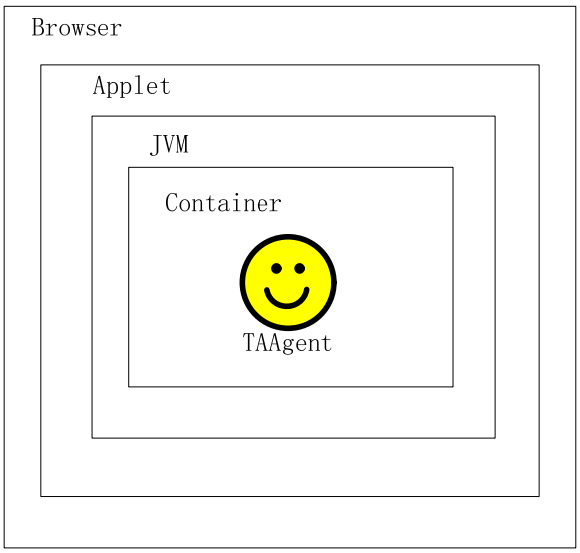


Fig. 7-7 TAAgent in the Browser

7.4.5 7.4.2.3 IMPLEMENTATION OF TEST GENERARATION SERVICE AGENT (TGSAGENT)

Table 7-3: Task implementation of Test Generation Service Agent

Behaviors Methods	Triggered Event	Task decomposition
Setup()	When GACtrlAgent is created, this method is automatically triggered.	This method performs initiations and addition of initial behaviors. For TGSAgent, it is used to 1. register content language and ontologies. in particular, we have to register: • agent content language, i.e. SLCodec • mobility ontology i.e. MobilityOntology • domain ontology i.e. ExamPaperOntology 2. add to behavior scheduler the following agent behaviors: WaitForRequest
WaitForRequest	if any incoming messages that match: Sender=authorized teacher agents, ConversationID=”Generate Test Paper”	1. extract message content and obtain the test paper 2. Add to behavior scheduler the following agent behaviors: SQBehavior
SQBehavior	when WaitForRequest behavior terminates	Control its sub behaviors to execute sequentially. In particular, this composite behavior contains two sub behaviors: CreateNewAgent, and PassTestParameter. When SQBehavior is triggered, it makes the first sub behavior available in the behavior scheduler

CreateNewAgent	When SQBehavior is triggered	1. create a new container in order to make the preparation for the creation of a new agent (i.e., GACtrlAgent) 2. create and start GACtrlAgent
PassTestParameter	When CreateNewAgent behavior terminates	Send message containg the test paper parameters to GACtrlAgent
ReturnResult	if any incoming messages that match: Sender=GACtrlAgent, ConversationID=”ResultReport”	Send message containing the final generated test paper to the requesting agent
CreateCourierAgent	if any incoming messages that match: Sender=TeacherAgent, ConversationID=”ConfirmedTestPaper”	Create courier agent that has the capability to carry the proved test paper to the specified Test Delivery Servers

7.4.6 7.4.2.4 IMPLEMENTATION OF GACTRLAGENT

Table 7-4: Task implementation of GACtrlAgent

Behaviors Methods	Triggered event	Task decomposition
Setup()	When gactrlagent is created, this method is automatically triggered.	This method performs initiations and addition of initial behaviors. For gactrlagent, it is used to 3. Register content language and ontologies. In particular, we have to register: • Agent content language, i.e. Slcodec • Mobility ontology i.e. Mobilityontology • Domain ontology i.e. Exampaperontology 4. Add to behavior scheduler the following agent behaviors: Receiveexampaperparameter, Statastic, startcrossover, and isclonefinishedinfo
ReceiveExamPaperParam	If incoming message matches the following conditions: Sender=tgsagent ,and Conversation ID= “exampaper_Parameter”	1. To extract test parameters from message content, including item type, number, teaching requirement, and difficult requirement; 2. To locate the GUID of the container (to migrate on) that runs on the Item Bank Server. This is realized through sending query to AMS. 3. To perform the operation of migration by calling the domove method. It should be noted that this method just changes the agent state to TRANSIT. The actual migration takes place asynchronously.

afterMove()	After gactrlagent arrives at the remote destination agent container in Item Bank Server, this method is triggered	We overrode this placeholder method to perform the following actions: 1. Reregister content language and ontology as they can be migrated with the agent itself 2. Create a set of tpagents according to the predefined number of population 3. Broadcast message containing test parameters to all the created tpagents
Stastic	When any incoming message that matches: Sender=any tpagent, conversationid=”Fitness”	1. Record the fitness values sent by tpagents 2. After each generation, this behavior is responsible for calculating the maximal, minimal and average fitness value and saving them. 3. When having collected all the fitness values for each generation and the current count of generation is lower than the preset total generation, this behavior will add the behavior controlselection to behavior scheduler
CtrolSelction	After each generation is over	1. Calculate the number of each tpagent that survives in the next generation, applying Roulette Wheel Algorithm. 2. Send messages to all of the tpagents of the current generation in order to order them to perform corresponding actions according to 1. In particular, • If number>1, the tpagent can clone itself by number-1 • If number=1, the tpagent does nothing • If number=0, the tpagent has to suicide.
IsCloneIFinishednfo	When the incoming messages satisfy the following condition: Sender=any tpagent Conservatonid= “confirmifcopyisfinished”	1. Count the number of messages coming from the tpctrlagents that clone themselves 2. Once the count arrives at the specified number, this behavior adds new behavior controlcrossover for the transmission of crossover information
ControlCrossover	After the task transfercrossoverinfo is finished	1. Randomly pairs all the tpagents off and produces the crossover positions. 2. Send messages containing the crossover position information to the paired tpagents that need to use that for the crossover operation
StartCrossover	When the incoming messages satisfy the following condition: Sender=any tpagent Conservatonid= ”crossoverprepared”	1. Count the number of messages coming from the tpagents that have received the crossover information 2. Send messages to the paired tpagents in order to really start crossover operation when they have prepared well the question items that need to be exchanged according to the crossover position

7.4.7 7.4.2.5 TPAGENT

Table 7-5: Task implementation of TPAgent

Behaviors Methods	Triggered event	Purpose and task decomposition
Setup()	When tpagent is created, this method is automatically triggered.	This method performs initiations and addition of initial behaviors. For gactrlagent, it is used to 1. Register content language and ontologies. In particular, we have to register: • Agent content language, i.e. Slcodec • Domain ontology i.e. Exampaperontology 2. Add to behavior scheduler the following agent behavior: Sqbehavior Selectionbehaviour; Waitcrossoverparameterbehaviour; Crossoverbehaviour; Reportresultbehaviour
SQBehavior	When setup() finishes its initiation	Control its sub behaviors to execute sequentially. In particular, this composite behavior contains two sub behaviors: getexampaperparameterbehaviour, and initializepopulationbehaviour. When sqbehavior is triggered, it makes the first sub behavior available in the behavior scheduler
GetExamPaperParameterBehaviour	If incoming message matches: Sender=gactrlagent ,and Conversation ID= "exampaper_parameterfromctrlagent"	Receive test paper parameters by extracting message content
InitializePopulation Behaviour	When getexampaperparameter Behaviour terminates	1. Separate the test paper parameters in order to obtain each item type and its number 2. Obtain the connection of local database containing item bank 3. Randomly produce the specified number of items within the preset scope for each item type 4. Extract item characteristics by SQL statements 5. Calculate the fitness value according to formule (7) in the previous chapter 6. Send message containing fitness information to gactrlagent

Selection Behaviour	If incoming messages match: Sender=gactrlagent ,and Conversation ID= “clone”	This behavior is used to perform the selection genetic operator Extract the message content and obtain the number that determines the time needing to be cloned by tpagent If number=0 then suicide If number=1 then do nothing If number>1 then clone itself number-1 time by calling doclone method
afterClone()	When the clone operation terminates	Send message with conversationid= “confirmifcopyisfinished “ to inform gactrlagent that the operation selection has terminated. It is worth noting the message content is empty.
WaitCrossoverParameterBehaviour	If incoming message matches: Sender=gactrlagent ,and Conversation ID= “crossoverparameter”	1. Extract message content and obtain crossover information including the agent identifier to which the receiving tpagent will send the question items to be exchanged, and crossover point 2. Prepare the crossover content according to the crossover point indicated by the message content 3. Send message to gactrlagent in order to notify that the crossover content has well been prepared 4. Wait for the confirmation information from gactrlagent in order to ensure that all tpagents that need to perform crossover operation, have made advance preparation for the forthcoming crossover operation 5. When the confirmation information arrives, then really launch the crossover operation by sending message containing the items to be exchanged to the specified tpagent
Crossover Behaviour	If incoming message matches: Sender=gactrlagent ,and Conversation ID= “crossovermsg”	1. Extract message content and obtain the question items that are used to reassemble the test paper 2. Perform the mutation operation according to the specified mutation probability. After this operation, a new test paper (i.e., a new chromosome) is created 3. Obtain the connection of local database containing item bank 4. Extract item characteristics by SQL statements according to the new created test paper represented by a set of item numbers 5. Calculate the new fitness value according to formule (7) in the previous chapter 6. Send message containing fitness information to gactrlagent

7.4.8 7.4.3 PLATFORM IMPLEMENTATION

7.4.9 7.4.3.1 SIMULATION

To perform the simulation studies, we firstly created five tables respectively representing five different types of test items in a database of MySQL. Each table contains 1000 records created in a random manner and has the same table structure as follows:

Item No	Knowledge point	Teaching requirement	Difficulty level	Score/item

Where: $\text{Knowledge point(KP)} \in \{kp1, kp2, kp3, kp4, kp5\}$ kp1 - kp5 indicates five different knowledge points.

$\text{Teaching requirement(TR)} \in \{know, understand, master\}$

$\text{Difficulty level} \in \{very\ easy, easy, medium, difficult, very\ difficult\}$

Score/item : represents the score value for each test item

The GA initial parameters mainly include the size of population (set by 30), the number of generations (set by 15), the crossover probability (set by 90%), mutation probability (set by 5%), the length of a chromosome (this depends on the total quantity of test items) as well as the weighting factors: k1 (set by 2) and k2 (set by 3) defined in formula 6-7.

Besides, the definitions of initial parameters for the test composition are specified by a teacher as shown from table 7-6 to table 7-9. It is worth noticing that

Table 7-6: the definition of a test structure

Item type	Number of items	Score/item	Subtotal
1	10	2	20
2	5	2	10
3	10	2	20
4	6	5	30
5	4	5	20
Total score of this test:			100

Table 7-7: the definition of difficulty levels

Difficulty level	very easy	easy	medium	difficult	very difficult
Score percentage	10%	20%	40%	20%	10%

Table 7-8: the definition of knowledge points

Knowledge point	kp1	kp2	kp3	kp4	kp5
Score percentage	10%	20%	30%	30%	10%

Table 7-9: the definition of teaching requirements

KP\TR	know	understand	master
kp1	80%	20%	0
kp2	20%	60%	20%
kp3	20%	50%	30%
kp4	30%	40%	30%
kp5	0	60%	40%

Figure 7-8 is a snapshot that illustrates the interactive conversation process among GAMASTP agents

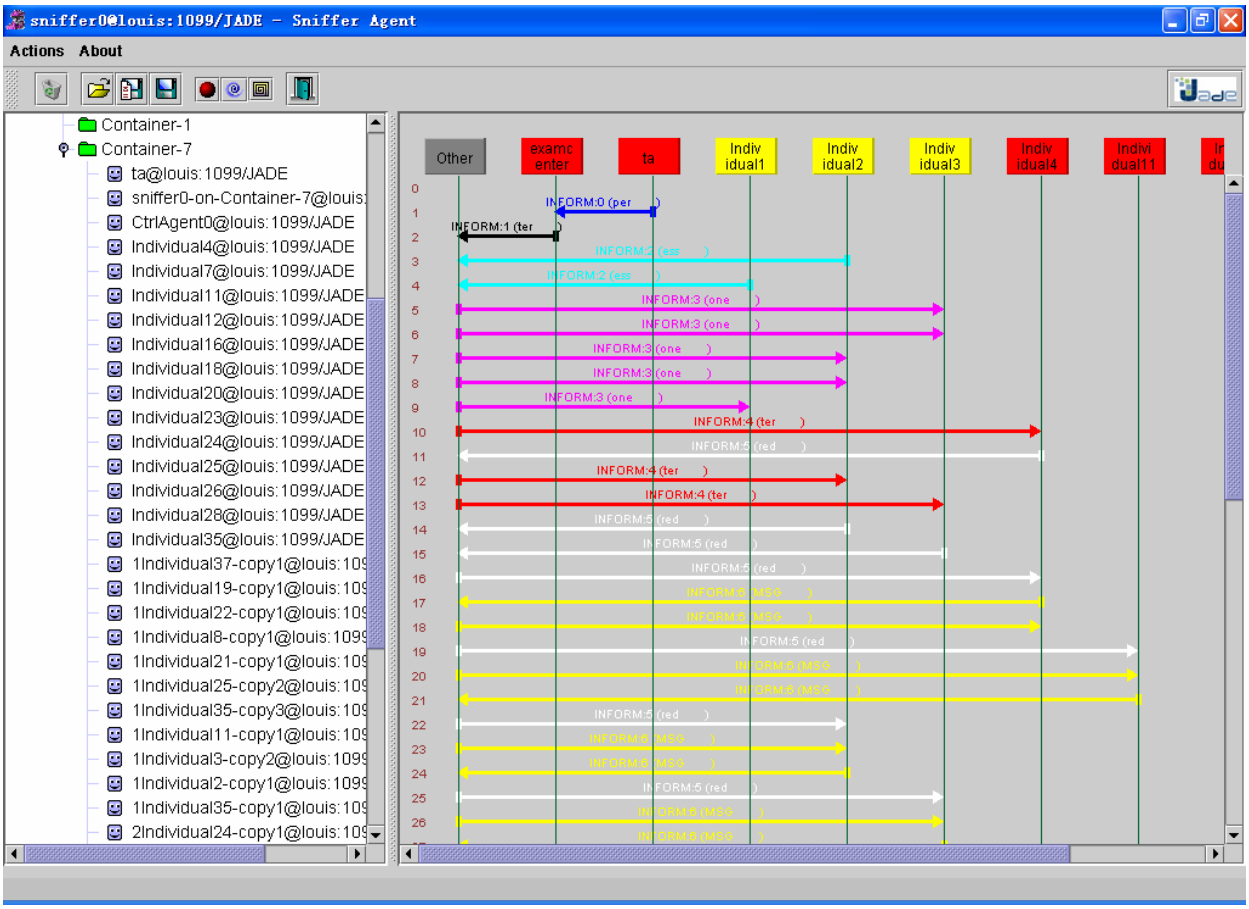


Fig. 7-8 Interactive process among GAMASTP agents

The final simulation results are shown as follows:

The best solution (i.e. the best chromosome consisting of a collection of test items)={114, 382, 542, 219, 600, 878, 196, 767, 197, 898, 1169, 1069, 1163, 1487, 1286, 2511, 2894, 2577, 2721, 2191, 2270, 2855, 2583, 2976, 2770, 3681, 3745, 3323, 3942, 3204, 3721, 4712, 4687, 4901, 4323}. Thanks to the chromosome structure defined in section 6.3.4.1, consequently, the requirements defined in table 7-6 are naturally met.

Figure 7-9 records the average fitness of each generation. It shows that the population of individual agents always evolves towards a better solution.

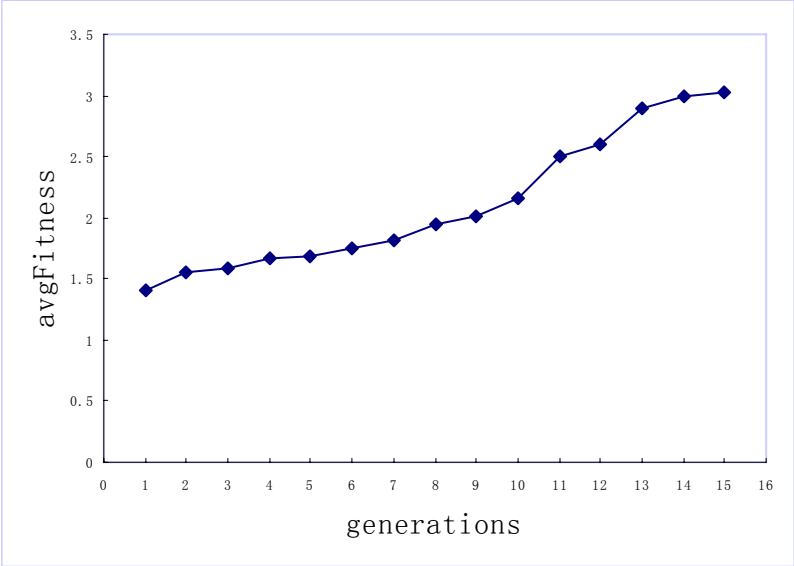


Fig. 7-9 Evolution of the average fitness of each generation

The simulation results with respect to difficulty levels, knowledge points and teaching requirements are shown as table 7-11, table 7-12 and table 7-13.

Table 7-11: the simulation results for difficulty levels

Difficult level	Very easy	easy	medium	difficult	Very difficult
Score percentage	8%	23%	42%	18%	7%

Table 7-12: the simulation results for knowledge points

Knowledge point	kp1	kp2	kp3	kp4	kp5
Score percentage	10%	14%	36%	32%	8%

Table 7-13: the simulation results for teaching requirements

KP\TR	know	understand	master
kp1	60%	20%	20%
kp2	14.3%	71.4%	14.3%
kp3	16.7%	50%	33.3%
kp4	37.4%	31.3%	31.3%
kp5	0	50%	50%

Compared with table 7-7, table 7-8 and table 7-9, we can obtain E1=10%, E2=24% and Fmax=4.08 according to the formulas 7-5, 7-6 and 7-7. They basically meet the requirements specified in table 7-7, table 7-8 and table 7-9. In order to verify the stability, we carried out hundreds of tests on the same problem, what make us excited is that the simulation results are still very stable. This shows the recommending prototype and its implementation model is feasible and robust.

7.5 IMPLEMENTATION OF LEARNER MODEL AGENT

The learner model agent is a key to personalize the learner's learning experience and adapt the system behaviors to the learner's personal profile. As we see from the previous chapter, This agent is responsible for updating the specific learner model, and answering any relative query from outside agents in terms of the learner's user profile such as personal information, interactive history, knowledge status etc. the implementation process involves the ontology generation, database creation, and the interaction protocol implementation. From our experience, to realize the second function, the key is to how to identify and understand the query content and achieve rational effect. This process is realized through implementing the standard FIPA query and request interaction protocol. Therefore, this example is used to demonstrate how to implement agent interaction protocol (IP) in JADE setting.

7.5.1 PROTOCOAL IMPLEMENTATION

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. A designer of agent systems has the choice to make the agents sufficiently aware of the meanings of the messages and the goals, beliefs and other mental

attitudes the agent possesses, and that the agent's planning process causes such IPs to arise spontaneously from the agents' choices. This, however, places a heavy burden of capability and complexity on the agent implementation, though it is not an uncommon choice in the agent community at large. An alternative, and very pragmatic, view is to pre-specify the IPs, so that a simpler agent implementation can nevertheless engage in meaningful conversation with other agents, simply by carefully following the known IP. Therefore, this section will introduce two IPs (i.e., FIPA Query IP and FIPA Request IP) involved in the Learner Model Agent conversations and furthermore shows the state chart applying FSM.

A. FIPA Request IP AUML diagram—update learner model

We apply this protocol to updating the learner's model. The FIPA Request Interaction Protocol (IP) allows one agent to request another to perform some action. The representation of this protocol is given in Figure 7-10 which is based on extensions to UML 1.x. (Odell 2001). This protocol is identified by the token FIPA-request as the value of the protocol parameter of the ACL message. The Participant processes the request and makes a decision whether to accept or refuse the request. If a refuse decision is made, then "refused" becomes true and the Participant communicates a refuse. Otherwise, "agreed" becomes true.

If conditions indicate that an explicit agreement is required (that is, "notification necessary" is true), then the Participant communicates an agree. The agree may be optional depending on circumstances, for example, if the requested action is very quick and can happen before a time specified in the reply-by parameter. Once the request has been agreed upon, then the Participant must communicate either:

- . A failure if it fails in its attempt to fill the request,
- An inform-done if it successfully completes the request and only wishes to indicate that it is done, or,
- An inform-result if it wishes to indicate both that it is done and notify the initiator of the results.

Any interaction using this interaction protocol is identified by a globally unique, non-null conversation-id parameter, assigned by the Initiator. The agents involved in the interaction must tag all of its ACL messages with this conversation identifier. This enables each agent to manage its communication strategies and activities, for example, it allows an agent to identify individual conversations and to reason across historical records of conversations.

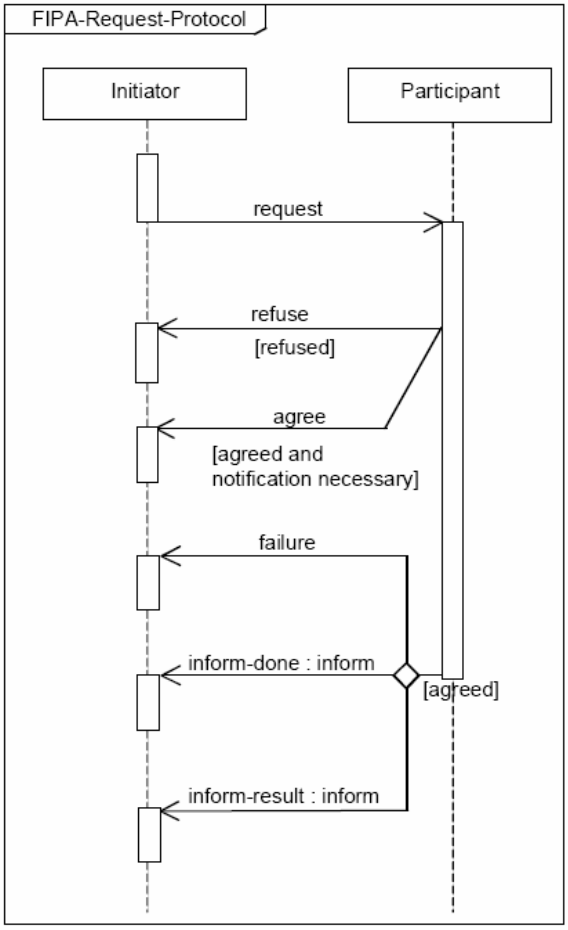


Fig. 7-10 FIPA request protocol

B. FIPA query IP AUML diagram —querying learner model

The Initiator requests the Learner Model Agent to perform some kind of inform action using one of two query communicative acts, query-if or query-ref (see (FIPA00037)). The query-if communication is used when the Initiator wants to query whether a particular proposition is true or false and the query-ref communication is used when the Initiator wants to query for some identified objects. The Learner Model Agent processes the query-if or query-ref and makes a decision whether to accept or refuse the query request. If the LMA makes a refuse decision, then "refused" becomes true and the Learner Model Agent communicates a refuse. Otherwise, "agreed" becomes true.

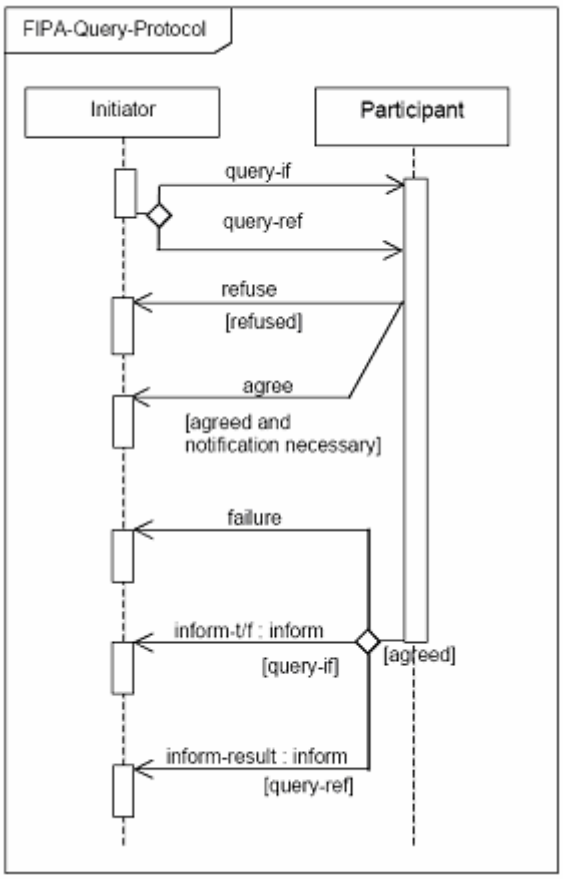


Fig. 7-11 FIPA query protocol

If conditions indicate that an explicit agreement is required (that is, "notification necessary" is true), then the Participant communicates an agree. The agree may be optional depending on circumstances, for example, if the requested action is very quick and can happen before a time specified in the reply-by parameter. If the Participant fails, then it communicates a failure. In a successful response, the Participant replies with one of two versions of inform:

- The Participant uses an inform-t/f communication in response to a query-if where the content of the inform-t/f asserts the truth or falsehood of the proposition, or,
- The Participant returns an inform-result communication in response to a query-ref and the content of the inform-result contains a referring expression to the objects for which the query was specified.

Any interaction using this interaction protocol is identified by a globally unique, non-null conversation-id parameter, assigned by the Initiator. The agents involved in the interaction must tag all of its ACL messages with this conversation identifier. This enables

each agent to manage its communication strategies and activities, for example, it allows an agent to identify individual conversations and to reason across historical records of conversations.

C. Implementation of Query IP using Finite State Machine

For every interaction protocol mandated by FIPA specifications, two roles can be played by an agent:

- Initiator role: the agent contacts one or more other agents to start up a new conversation, evolving according a specific interaction protocol.
- Responder role: in response to a message received from some other agent, the agent carries on a new conversation following a specific interaction protocol. In this example, it is obvious that the Learner Model Agent plays such role.

JADE provides a Behavior object for each one of these two protocol roles; these behaviors often are abstract classes that application programmers must extend in order to provide application specific code to handle the various protocol steps. Figure 7-12 and figure 7-13 shows the state chart of the initiator and responder agent, respectively.

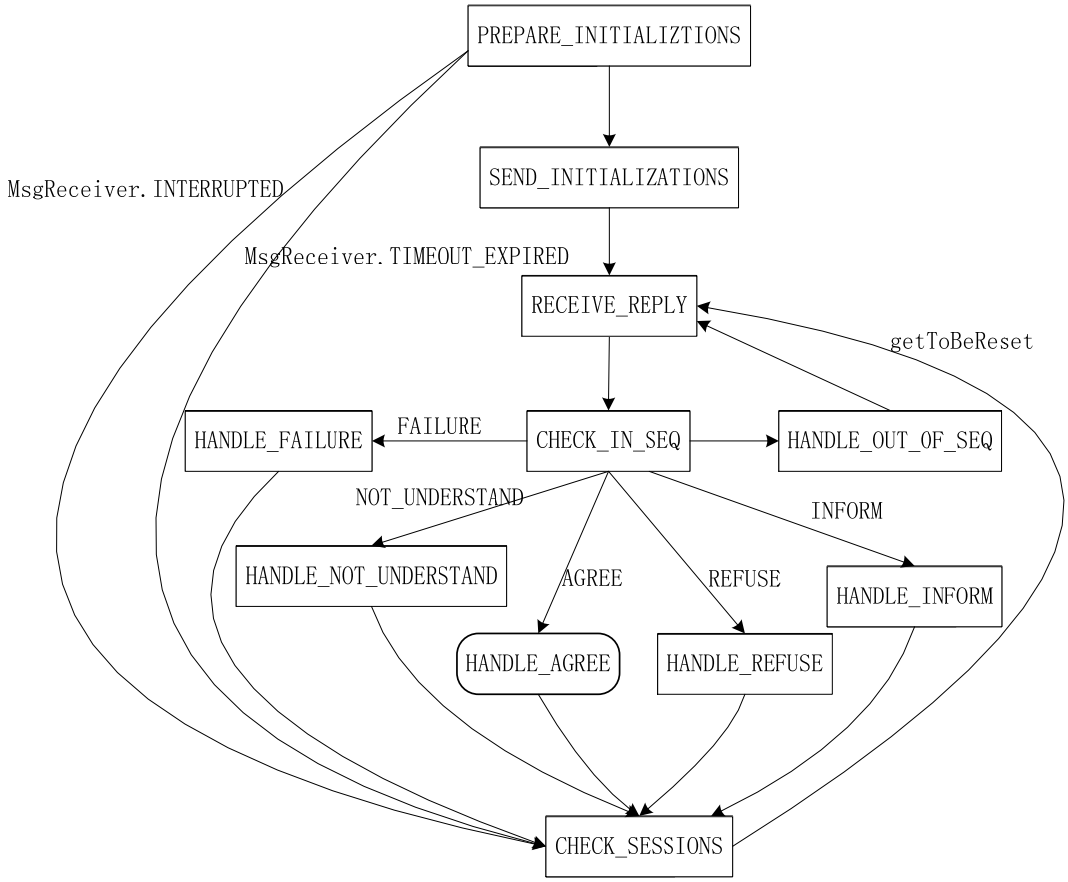


Fig. 7-12 FSM state chart of Initiator

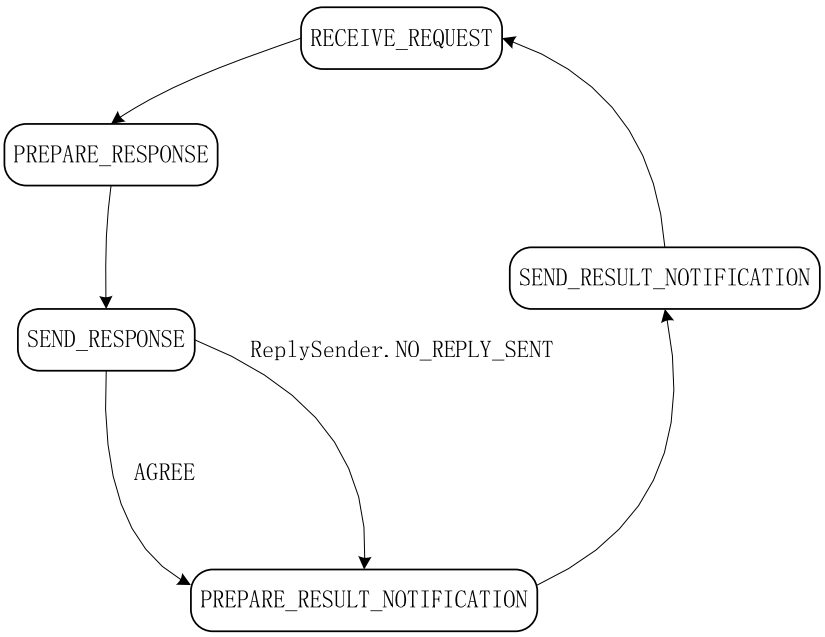


Fig. 7-13 FSM state chart of Learner Model Agent

7.5.2 SCENARIO

This section demonstrates two concrete application scenarios accompanied by a sequence of screenshots intercepted from our simulation environment developed in JBuilder. The scenario is described blow:

Case 1: A Pedagogic Agent named into wants to query for the education status of a learner named Louis from his learner profile database.

Query structure from Pedagogic Agent:

```
((iota ?myedustatus
(EducationStatusIs ?myedustatus (Learner :name louis))))
```

Answer results from Learner Model Agent:

```
((=
    (iota ?myedustatus
      (EducationStatusIs ?myedustatus (Learner
         :name louis)))
    (EducationStatus
      :major
       (sequence automation "system analysis and integration")
      :preferredDomain MAS
      :area e-learning
      :obtainedDegrees
       (sequence automation "system analysis and integration"))))
```

Figure 7-14 show the conversation ACL message between the two agents more clearly.

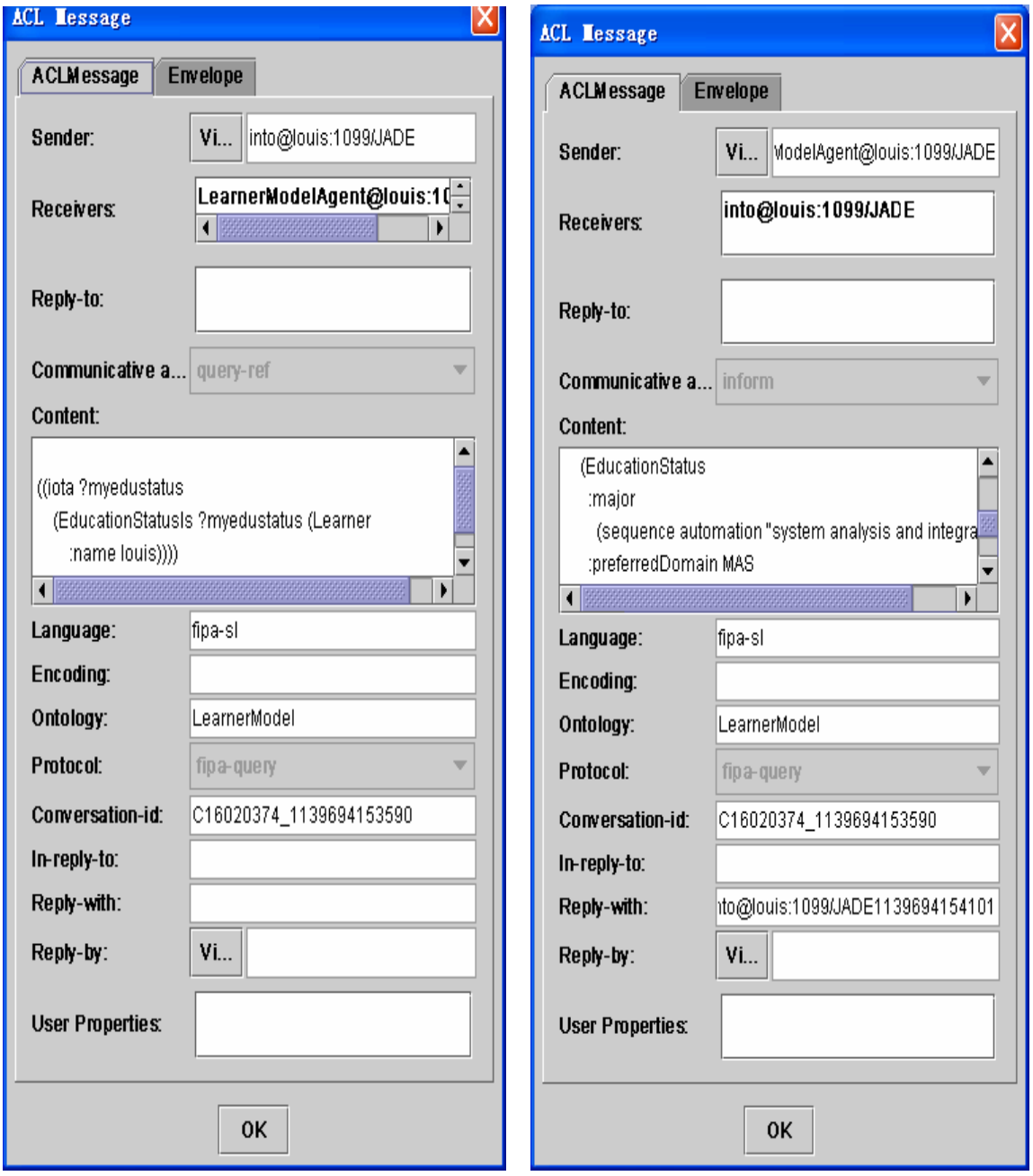


Fig. 7-14 Snapshot of ACL Messages for Learn Model Agent and Pedagogic Agent

Figure 7-15 illustrates the incoming message and outgoing message boxes within Learning Model Agent; furthermore, we can see the dynamic behavior pool at the bottom of figure 7-15.

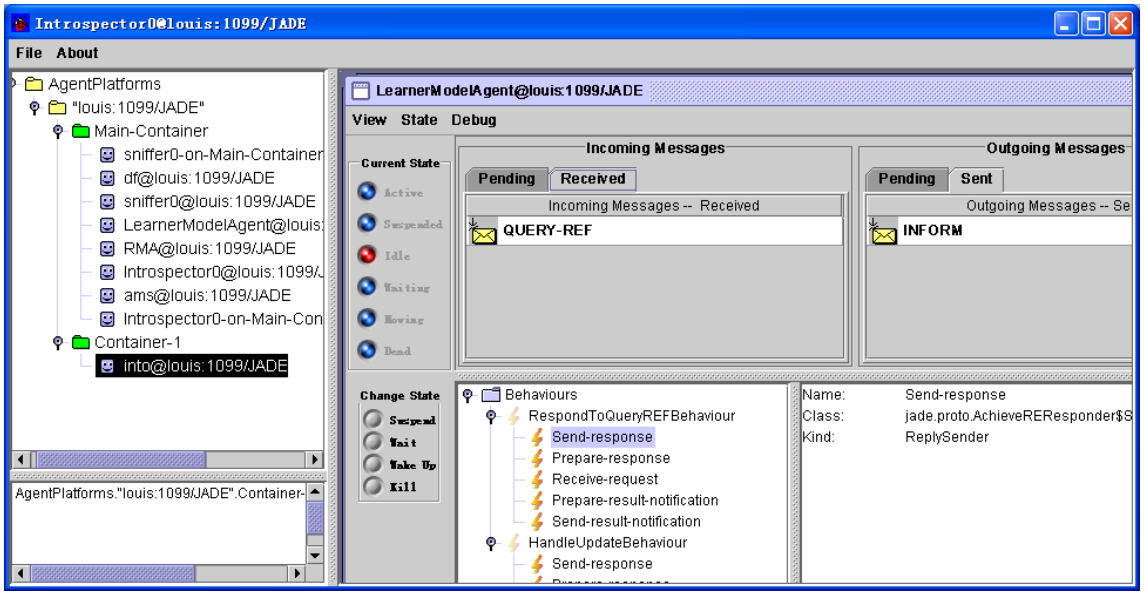


Fig. 7-15 snapshot of the message box and behavior pool for Learner Model Agent

Case 2: The Evaluation Agent send a message to Learner Model Agent and request it to update a learner's cognitive ability. Different from the prior case, the message content is represented as XML language instead of SL expression. This shows that an agent has the capability to speak more than one language although the agents in ongoing conversation have to speak the same language respect the same protocol.

```
<CONTENT_ELEMENT type="NotifyInfo">
<assessInfo type="CognitiveConstruct">
    <cognitiveAbility>
    <avgScoreForEasyKP>0.0</avgScoreForEasyKP>
    <knowledgeLevel>justsoso</knowledgeLevel>
    <analysisLevel>good</analysisLevel>
    <evaluationLevel>double</evaluationLevel>
    <rank>0</rank>
    <avgScoreForMediumKP>0.0</avgScoreForMediumKP>
    <applicationLevel>poor</applicationLevel>
    <avgScoreForHardKP>0.0</avgScoreForHardKP>
<COMPREHENSIONLEVEL>TRES

BIEN</COMPREHENSIONLEVEL>

    <totalScore>0.0</totalScore>
</COGNITIVEABILITY>

    <DomainID>computer</DomainID>
</assessInfo>
</CONTENT_ELEMENT>
```

7.6 IMPLEMENTATON OF PEER HELP SYSTEM

For the sake of saving pages, we focus on the scenario description involved in this example, concerning the agent implementation details; we do not list them here. Thus, we demonstrate this subsystem in the light of the sequence of a whole help session

1. Register service with Help Service Agent

Figure 7-16 illustrates the user interface for the service registration (i.e., capability and preference publishing) with Help Service Agent by a Register Service Agent. Actually, this process can not only performed by filling in the form manually according to the learner's willing, but also by the peer help system automatically. This mechanism makes more flexible for the learner to publish his/her capability.

Fig. 7-16 Register service with DFAgent

2. Request matchmaker agent to find desired peer helpers

Table 7-14: Message slots for Learner Assistant Agent

Purpose	Search online peer learners from DFAgent, who are knowledgeable about topic 5 and are willing to speak English with chat tool
Sender	PeerHelpertAgent
Receiver	MatchmakingAgent
Communicative act	Request
Language	XML
Ontology	Help
Protocol	FIPA-Request
Conversaton-id	pa@louis:1099/JADE321344553553
Reply-with	pa@louis:1099/JADE321344553553
Content	See below

Communication content is represented in the form of XML as below:

```
<CONTENT_ELEMENT type="help">
<topic>topic5</topic>
<communicationTool>chat</communicationTool>
<competence>very high</competence>
<language>English</language>
</CONTENT_ELEMENT>
```

3. Search prospective peer learners

Table 7-15: Message slots for Matchmaking Agent

Purpose	Return a list of online learners who meet the specified constraint
imitator	Learner 1
Sender	matchmakingAgent
Receiver	DFAgent
Communicative act	Request
Language	fipa-sl0
Ontology	FIPA-Agent-Management
Protocol	FIPA-Request
Conversaton-id	matchmaker@louis:1099/JADE1139931568312
Reply-with	matchmaker@louis:1099/JADE1139931568312
Content	See below

Communication content is represented as below in SL:

```
((action
    (agent-identifier
      :name df@louis:1099/JADE
      :addresses (sequence http://louis:7778/acc))
    (search
      (df-agent-description
        :languages (set English)
        :services
          (set (service-description
              :name topic5
              :properties (set (property :name competence :value "very high")))))
      (search-constraints :max-results -1))))
```

4 Return a list of online learners who meet the specified constraint

Table 7-16: Message slots for Help Service Agent

Purpose	Return a list of online learners who meet the specified constraint
imitator	Learner 1
Sender	HelpServiceAgent
Receiver	matchmakingAgent
Communicative act	Request
Language	fipa-sl0
Ontology	FIPA-Agent-Management
Protocol	FIPA-Request
Conversaton-id	matchmaker@louis:1099/JADE1139931568312
Reply-with	matchmaker@louis:1099/JADE1139931568312
Content	See below

Communication content is shown below:

```
((result
    (action
     (agent-identifier
       :name df@louis:1099/JADE
       :addresses (sequence http://louis:7778/acc))
     (search
       (df-agent-description
         :services (set (service-description
                  :name topic5
                  :properties (set (property :name competence :value "very high"))))
         :languages (set English))
       (search-constraints :max-results -1)))
    (sequence
     (df-agent-description
       :name (agent-identifier :name learner2@louis:1099/JADE :addresses (sequence
http://louis:7778/acc))
       :services
        (set (service-description
           :name topic5
           :ownership learner2
           :properties
            (set (property :name competence    :value "very high")
               (property :name communication :value chat)
               (property :name rank :value help.ontology.Help@5c1eae))))
       :protocols (set "")
```

```
        :languages (set English)
        :ontologies (set MASCourse))
      (df-agent-description
        :name (agent-identifier :name learner1@louis:1099/JADE :addresses (sequence
http://louis:7778/acc))
        :services
          (set (service-description
             :name topic5
             :ownership learner1
             :properties
              (set (property :name competence :value "very high")
                 (property :name communication :value chat)
                 (property :name rank:value help.ontology.Help@1909385))))
        :protocols (set "")
        :languages (set English)
        :ontologies (set MASCourse)))))
```

5 Invite chosen peer helper to participate in help session

Fig. 7-17 UI of help invitation

6 Start the help session

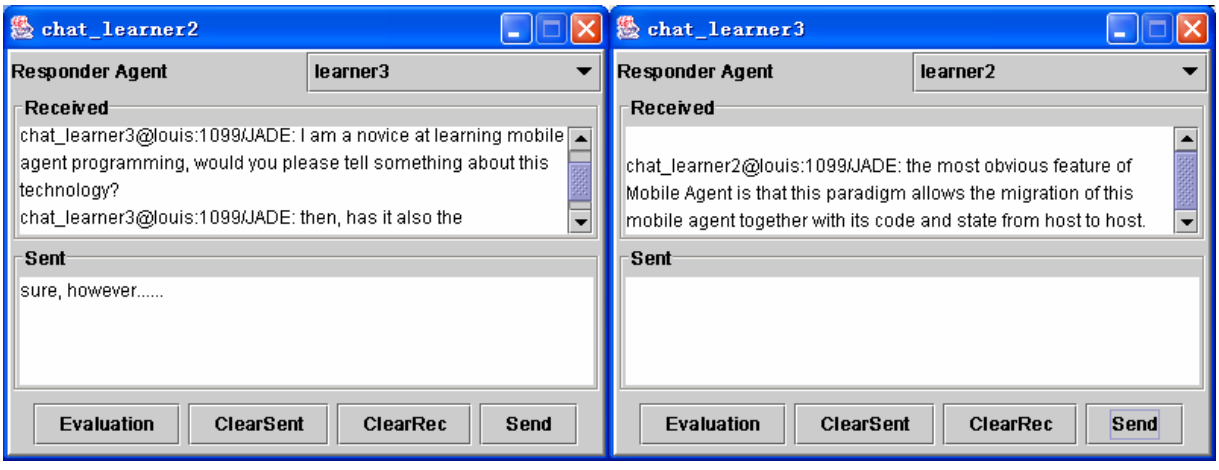


Fig. 7-18 UI of help session

7.7 SUMMARY

In this chapter, we implemented three applications. First, we implemented the simplified prototype of GAMASTP for the purpose of synthetically revealing how to concretely implement a complex multi-agent system, which is concerned with several key issue: how to implement test ontology and apply to the communication among agents; how to design and implement agent behavior model according to the previous models; how to deploy agents over different network nodes. the second application is implemented for the purpose of how the learner model agent updates the learner model upon receiving the refresh data as well as how to answer any questions from external agents, this example also showed the application of interactive protocols such as FIPA request and FIPA query. The third example is used to implement part of the peer help system aiming at demonstrating the process how to find appropriate competent peer learners. In this example, for purpose of the simplification, we focus on implementing several particular agents such as DFAgent that makes possible for learners to publish their services, here indicating the cognitive abilities on different topics/concepts shared by learners, learning styles, or other personal profiles, MatchMaking Agent that is responsible for search for prospective peer learners who meet the specified help constraint, and the ChatAgent that is used to communicate messages in a help session. This agent is activated when a successful negotiation between helper and helpee has been arrived at. The results of simulation show that the models proposed in this thesis is feasible and reasonable..

CHAPTER 8 CONCLUSIONS AND PERSPECTIVES

8.1 CONCLUSIONS

An integrated solution to the MAS based e-Education is proposed in this dissertation. The main contributions of this dissertation can be concluded in the following points:

1. Contribution to the modeling of multi-user & multi-agent in the distributed context. In particular, this thesis addressed the agent communication issue, we designed a set of FIPA-compliant ontologies such as learner model ontology, learning content management ontology, test ontology so that all the interacting agent can speak the same language and apply the same set of vocabulary. Instead of adopting the traditional server-centered monolithic user modeling method, we put forward a just-in-time multi-user multi-agent modeling method that allows different agent to use different modeling method and take different forms of model representation so as to enhance the robustness and flexibility and modularity in large-scale, complicated and open e-Education environment.

2. Proposed a powerful and flexible multi-agent based framework to facilitate the development of course contents and support just-in-time individualized learning experience for learners with different needs and cognitive ability by making full use of a repository of reusable learning objects;

3. Designed a efficient learning path generation algorithm aiming at recommending a tailored learning path consisting of domain concepts in support of explaining the target concepts specified by the learner according to both the learner's knowledge state and learning preference.

4. Proposed a flexible and innovative collaborative learning reference model according to the principle of constructivism, in support of learners to create tailored learning group by virtue of different matchmaking agents on different criteria basis, moreover, this model makes easy for teachers to monitor and organize the activity of learning groups.

5. Addressed the issue how to find appropriate electronic resources, peer learners, tutors in the distributed learning space through the smart help system.

6. Put forward an innovative holistic solution to modeling large-scale on-line assessment system involving four typical e-assessment process: test generation, delivery, evaluation and result publishing. by applying the new generation of mobile agent based distributed computing paradigm. In particular, the most significant contribution of this

model is that we proposed and designed an innovative model of automatic test generation by seamlessly integrating genetic algorithm, mobile agent, and MAS. The incorporation of mobile agent paradigm not only addressed the issues such as bandwidth latency, network traffic, fault tolerance, load balance etc. but also greatly enhanced the flexibility in terms of generating, delivering and assessing test.

8.2 PERSPECTIVES

When the old problems are solved, new problems will appear. The history of the science and technologies always run in this way. The further study will be conducted as below:

1. From development and system perspective, e-Education is increasingly becoming more and more dynamic, open, unpredicted, and complex distributed system. As a new generation of solution to the distributed computing paradigm, MAS exhibits its powerful application perspective in the domain of e-Education. However, the predominant and popular network computing paradigm at present is on the client-server paradigm basis. Therefore, we will pay more attention to explore how to combine the MAS system with the traditional network service architecture (e.g. J2EE). Obviously, there are two directions to be expected: one possibility is to encapsulate the latter with multi-agent, another is that the MAS system can be regarded as a standalone service within the former.
2. From standard view, as is known that it brings us several benefits with regard to exchangeability, reusability, communication, etc. however, most of the learning standards seems not to put more efforts on the adaptive and personalized learning. Inevitably, how to further extend them will be a focus in the near future. Besides, how to develop interfaces that facilitate the application of these standards is another aspect that we are interested.
3. From MAS perspective, so far, few environments can provide the capability of developing practical MAS based applications. Our experience in the development of MAGE shows that a whole cycle for the MAS development has to be involved in several phases including analysis, modeling, programming, debugging and deployment. The development work will be a very burdensome and rather complex process without a MAS development environment. Thus, our future research will also be on how to implement a FIPA-compliant MAS development environment based on the object-oriented modeling language UML.

ACKNOLEDGEMENT

In the last few years, I have been guided and supported by a number of people without whom; it would have been very difficult to navigate the Ph.D. journey.

To start with, I want to thank my advisor professors Luqing YE, Pierre PADILLIA and Daniel ROY. I am very grateful to them for having provided me with ample scope to freely explore research avenues and learn from my adventures! They have also taught me by example to give attention to detail without loosing sight of the bigger picture.

Dr. Romuald STOCK has also greatly guided my work and influenced my thinking. I am grateful to him for having introduced me to the fascinating world of human cognition and learning. Needless to say, his friendship, understanding and encouragement have provided me with great strength and inspiration.

I am grateful to Latifa REZG for her friendly reception and kind consideration.

I must specially thank everybody at the LGIPM and HUST Laboratory for having provided me with an intellectually rich and supporting environment in conducting my research.

I also take this opportunity to thank my roommate Chen PENG for his time and encouragement.

I thank all my colleagues Yongqing LIU, Chuang FU, Xiuliang LIU, Hongzhu LIANG, Zhengrong LI, Hongxia XIONG, Ren YU, Yuanchu CHENG for their feedback and for supporting me through my dissertation.

Specially, I wish to express sincere appreciation to my Chinese supervisor professor Luqing YE for his assistance and insight through my dissertation research. He has provided me with invaluable feedback, advice and criticism and has been an excellent role model. He has given me the opportunity and guidance to grow up as a researcher and a person.

My parents give me the gift of life and natured me who I am. They are always there for me. Everything I am and will be is a complex combination of their unconditional love, patience and unique ways. I dedicate this effort to them and hope to be worthy of the lives they live. Finally, thanks to my family and friends who have provided me with all the

support and encouragement I needed.

Finally, I must also mention that through the long sleepless nights spent working on my dissertation, I have been inspired by the love of my wife. I would like to thank for her support and patience during the very difficult time.

I am also grateful for the financial support from the DUO-France, and I would like to thank the authors of free software applications especially the JADE open source project team.

DEDICATION To My Dear Parents and My Dear Wife

REFERENCES

Agent Tcl. (2005). R.S. Gray. Agent Tcl: A flexible and secure mobile-agent system. In Forth Annual UsenixTcl/Tk Workshop, 1996.

IBM. (2005). Aglets Workbench. http://www.trl.ibm.co.jp/aglets/index.html.

M. Strasser, J. Baumann, and F. Hohl. Mole (1997).- a java based mobile agent system. In M. M uhlh auser, editor, Special Issues in Object Oriented Programming, pages 301-308. dpunkt Verlag, 1997.

Voyager, (2005). ORB 3.0 Developer Guide, 1999.http://www.objectspace.com.

ADL (2004) Advanced Distributed Learning: SCORM, 2004. Last visit: 2004-03-26, URL: http://www.adlnet.org/index.cfm?fuseaction=scormabt

Anderson, John R. (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences* 4 (2):167-207.

Andrade, Adja F. de; Jaques, Patrícia A.; Jung, João Luiz; Bordini, Rafael H.; Vicari, Rosa M.. (2001). A Computational Model of distance Learning Based on Vygotsky´s Socio-cultural Approach. Workshop Multi- Agent Architectures for Distributed Learning Environments. Proceedings International Conference on AI and Education, San Antonio, Texas, May, 2001. P.33-40.

Ayala, G., Yano, Y. (1996). Intelligent Agents to Support the Effective Collaboration in a CSCL Environment, Proceedings of the ED-TELECOM 96 World Conference on Educational. Communications, June 17 - 22, 1996, Boston, Mass. AACE, Patricia Carlson and Fillia Makedon (Eds.) pp. 19-24

Ayala, G., Yano, Y. A. (1998). Collaborative Learning Environment Based on Intelligent Agents, Expert Systems with Applications, Pergamon Press, pp. 129-137. 1998

Bednar, A. K., Cunningham, D. Duffy, T. M. & Perry, J. D. (1998). Theory into practice: How do we link? In T. M Duffy and D. H. Jonassen (Eds.) *Constructivism and technology of instruction: A conversation*, Hillsdale, NJ: Lawrence Erlbaum Associates, 17-35.

Bloom, B.S. (1984). The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, 13, 3-16.

Bloom, C.P. (1995) Roadblocks to successful ITS Authoring in Industy. In *Proceedings of AI-ED-95 Workshop on Authoring Shell for Intelligent Tutoring System*. Washionton.DC, 16, 1-6.

Bloom, M.D. (1956) Engelhart, E.J. Furst, W.H. Hill, & D.R. Krathwohl *Taxonomy of*

Educational Objectives, Handbook I: Cognitive Domain (McKay, New York).

Bond, A. H., Gasser, L. (Eds.) (1988) Readings in Distributed Artificial Intelligence.

Bradshaw, J.M., Dutfield, S., Benoit, P. and Woolley, J.D. KAoS: Toward An Industrial-Strength Open Agent Architecture. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 375-418.

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligenc*e, 4:349–355.

Brauer, W. and Weiss, G. Multi-machine scheduling, a multi-agent approach. In *Proceedings of the Third International Conference on Multi Agent Systems*, pages 42-48, 1998.

Briggs, K. C. and Myers, I. B. (1977). *Myers-Briggs Type Indicator*, Palo Alto, CA: Consulting Psychologist Press, Inc.

Brown, J. S., and K. VanLehn. (1980). Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science* (4):379-426.

Brown, J. S., and K. VanLehn. (1980). Repair Theory: A Generative Theory of Bugs in Procedural Skills. *Cognitive Science* (4):379-426.

Brusilovsky, P. (2001). Adaptive hypermedia. User Modeling and User Adapted Interaction, Ten Year Anniversary Issue (Alfred Kobsa, ed.) 11 (1/2), 87-110

Camacho Fernández, D. Recuperación e Integración de Información disponible en Web para la Resolución de Problemas, Universidad Carlos III de Madrid, 2002.

Cammarata, S.; McArthur, D.; and Steeb, R. 1983. Strategies of Cooperation in Distributed Problem Solving. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), 767–770. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Capuano, N.; Marsella, M.; Salerno, S. ABITS: An Agent Based Intelligent Tutoring System for Distance Learning. In : Proceedings of the International Workshop in Adaptative and Intelligent Web-based Educational Systems. Available at http://virtcampus.cl-ki.uni-osnabrueck.de/its-2000/.

Carbonell, J. R. AI in CAI: na artificial intelligence approach to computer assisted instruction. IEEE Transactions on Man Machine Systems, v.11, n.4, p.190-202, 1970.

Carver, C. A., Howard R. A. and Lavelle E. (1996). *Enhancing learner learning by incorporating learner learning styles into adaptive hypermedia.* Proceedings of EDMEDIA 96 - World Conference on Educational Multimedia and Hypermedia, Boston, MA, pp. 118-123.

Carver, C. A., Howard, R. A. and Lane, W. D (1999). *Enhancing Learner Learning*

Through Hypermedia Courseware and Incorporation of Learner Learning Styles, IEEE Transactions on Education, vol. 42, no. 1, 1999, pp. 33-38.

Chess D., Harrison C.G., and Kershenbaum A. 1998, Mobile agents: Are they a good idea? In G. Vigna, editor, Mobile Agents and Security, LCNS 1419, pages 25-47. Springer Verlag,.

Chess D.M.. 1998, Security issues in mobile code systems. In *Mobile Agents and Security*.

Cohen, P.R., Cheyer, A., Wang, M., and Baeg, S.C. An open agent architecture. In: *Proceedings of the AAAI Spring Symposium*. 1994.

Colazzo, L.; Silvestri, 1997, L. The pragmatics of the Tutor:A proposal of modelling. AI-ED97 : Eighth World Conference on Artificial Intelligence in Education - Workshop V : Pedagogical Agents, 8.,1997. Proceedings... Kobe: Japan, 1997.

Corkill, D. D., and Lesser, V. R. 1983. The Use of Metalevel Control for Coordination in a Distributed Problem- Solving Network. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), 767–770. Menlo Park, Calif.: International Joint

Corkill, D. D., and Lesser, V. R. 1983. The Use of Metalevel Control for Coordination in a Distributed Problem- Solving Network. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), 767–770. Menlo Park, Calif.: International Joint

Dag Johansen, Robert van Renesse and Fred B. Schneider, “Operating System Support for Mobile Agents”, Department of Computer Science, Cornell University, USA, November 1994.

D'Amico, C.B.; Viccari, R.M.; Alvares, L.O. (1977) A Framework for Teaching and Learning Environments. In: Simpósio de informática Na educaçao, VIII, 1977, Sao Paulo, SP.

D'Amico, C.B.; Pereira, A.S.; Geyer, C.F.R., Viccari, R.M. (1998) Adapting Teaching Strategies in a Learning Environment on WW. In: Proceedings of the WebNet World Conference of the WWW, Internet & Intranet, Florida, USA. 1998.

Davidson, K. (1998). *Education in the Internet--linking theory to reality*, http://www.oise.on.ca/~kdavidson/cons.html

Decker, K., Sycara, K. and Williamson, (1998) M. Middle-Agents for the Internet. In: *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI-97)*, January, 1997.

Decker, K., Williamson, M. and Sycara, K. Matchmaking and Brokering. (1996) In:

Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), December, 1996.

Dent, L.; Boticario, J.; McDermott, J.; Mitchell, T.; and Zabowski, D. 1992. A Personal Learning Apprentice. In Proceedings of the Tenth National Conference on Artificial Intelligence, 96–103. Menlo Park, Calif.: American Association for Artificial Intelligence.

DerekStockley (2004) E-learning Definition and Explanation (Elearning, Online Training, Online Learning), 2004. Last visit: 2004-04-29, URL: http://derekstockley.com.au/elearningdefinition. html

Dick, W. (1995). Instructional design and creativity: A response to the critics. *Educational Technology*, 35 (4), 5-11.

Dietinger T. (2003) Aspects of E-Learning Environments. Graz University of Technology, Dissertation, 2003.

Doignon, J.-P., and J.-C. Falmagne. (1999). *Knowledge Spaces*: Springer-Verlag.

Doolittle P. E., Virginia T. (1999). Constructivism and Online Education, (http://edpsychserver.ed.vt.edu/workshops/tohe1999/pedagogy.html)

Duffy, T. M. & Jonassen, D. H. (1992). *Constructivist and the technology of instruction: A conversation*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Ertmer, P. A. & Newby, T. J. (1993). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly*, 6 (40), 50-72.

Felder R.M. 1995, Learning and Teaching Styles In Foreign and Second Language Education *Foreign Language Annals,* 28,1, 21–31

Felder, R. M. and Soloman, B. A. (2003). Index of Learning Styles Questionnaire, Available online in: http://www.ncsu.edu/felder-public/ILSdir/ilsweb.html

Felder, R.M., and L.K. Silverman. 1988. Learning and Teaching Styles in Engineering Education. *Engineering Education* 78,7, 674-681.

Felder, R.M.1993. Reaching the Second Tier: earning and Teaching Styles in College Science Education. *J. Coll. Sci. Teaching* 23,5, 286-290.

Ferguson, I. A. (1992a). *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agent*s. PhD thesis, Clare Hall, University of Cambridge, UK. (Also available as Technical Report No. 273, University of Cambridge Computer Laboratory).

Ferguson, I. A. (1992b). Towards an architecture for adaptive, rational, mobile agents. In Werner, E. and Demazeau, Y., editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds*

(MAAMAW-91), pages 249–262. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands.

Fikes, R. E. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligenc*e, 5(2):189–208.

Flores-Mendez R.A. (1999) Standardization of Multi-Agent System Frameworks, University of Calgary 1999.

Fosnot, C. T. (1996), Constructivism: Theory, perspective, and practice, New York: Teachers College Press.

Frasson, Claude; Martin, Louise; Gouarderes, Guy; Aimeur, Esma. (1998) LANCA: A Distance Learning Architecture Based on Networked cognitive Agents. In lecture Notes in Computer Science. Intelligent Tutoring Systems. Proceedings of 4th International Conference, ITS 1998, San Antonio, Texas, August 1998. P. 594-603.

Gagné, R. M., Wager, W. W. & Briggs, L. J. (1992). *Principles of instructional design* (4th ed.), New York: Holt, Rinehart and Winston.

Garcia Barrios, V.M.; Gütl, Ch.; Preis, A.M.; Andrews, K.; Pivec, M.; Mödritscher, F.; Trummer, Ch.: AdeLE (2004) A Framework for Adaptive E-Learning through Eye Tracking. In proceedings of IKnow' 04, 2004.

Gardner, H. (1993). *Multiple Intelligences: The Theory in Practice*. New York: Basic books.

Garrido, L., and Sycara, K. 1996. Multiagent Meeting Scheduling: Preliminary Experimental Results. In Proceedings of the Second International Conference on Multiagent Systems, 95–102. Menlo Park, Calif.: AAAI Press.

Genesereth, M. R., and Ketchpel, S. P. 1994. Software Agents. Communications of the ACM 37(7): 48–53

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87*), pages 677–682, Seattle, WA.

Giráldez Betrón, J.I. (1999) Modelo de Toma de Decisiones y Aprendizaje en Sistemas Multi-Agente, Universidad Politécnica de Madrid,.

Gilbert, J. E. and Han C. Y. (1999). Arthur: Adapting Instruction to Accommodate Learning Style. *Proceedings ofWebNet 99, World Conference of the WWW and Internet, Honolulu,* HI, 433-438.

Gilbert, J. E. and Han, C. Y. (2002). Arthur: A Personalized Instructional System..*Journal of Computing In Higher Education. 14, 1,* Norris Publishers, Amherst, MA.

Johnson, C. and Orwig, C. (1998). *What is learning style.* http://www.sil.org/lingualinks/library/Llearning/CJ0625/CJ0676.html

Hall, B. (1997). Web-based training cookbook. New York: Wiley.

Hannafin, M., Land, S. & Oliver, K. (1999). Open learning environments: Foundations, methods, and models. In C. M. Reigeluth (Ed.), *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II), New Jersey: Lawrence Erlbaum Associates, 115-142.

Hasebrook J.P. (2001) Learning in the Learning Organisation. In journal *J.UCS*, vol. 7, pp. 472-487, 2001.

Hayes-Roth,B. 1995, Directed Improvisation with animated puppets. CHI'95: Conference on Human-Computer Interaction, 1.,1995.Proceedings... Denver: 1995.

Honey, P. (2001). Honey and Mumford Learning Styles Questionnaire. *PeterHoney Learning,* http://www.peterhoney.com/product/learningstyles. Accessed Apr 2004.

Hong, H. & Kinshuk (2004). Adaptation to Learner Learning Styles in Web Based Educational Systems, *ED-MEDIA 2004*, 491-496.

Jennings, N. R. (1993). Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving. *Journal of Intelligent and Cooperative Information System*s, 2(3):289–318.

Jonassen, D. H. & Grabowski, B. L. (1993). *Handbook of individual differences, learning and instruction*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Jonassen, D. H. (1999). Designing consructivist learning environments. In C. M. Reigeluth (Ed.) *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II), New Jersey: Lawrence Erlbaum Associates, 215-239.

Jonnassen, D. H. (1991). Objectivist vs. constructivist: Do we need a new philosophical paradigm? *Educational Technology: Research and Development*, 39 (3), 5-14.

Kinny, D.; Ljungberg, M.; Rao, A.; Sonenberg, E.; Tidhard, G.; and Werner, E. 1992. Planned Team Activity. In Artificial Social Systems, eds. C. Castelfranchi and E. Werner. New York: Springer-Verlag.

Kinshuk, Patel,A. (1997). A Conceptual Framework for Internet based Intelligent Tutoring Systems. *Knowledge Transfer* 2,117-124.

Koedinger, Kenneth R. (2001). Cognitive tutors as modeling tool and instructional model. In *Smart Machines in Education*, edited by Feltovich. Menlo Park, CA: AAAI Press.

Kolb, D. & Fry, R. (1975). Towards an applied theory of experiential learning, *in*

Theories of group processes, ed. C.L. Copper London: John Wiley, 33-58.

Kolb, D. (1984). *Experiential learning: Experience as the source of learning and development*, Englewood Cliffs, NJ: Prentice-Hall.

Lange D.B. and Oshima M. (1999), Seven good reasons for mobile agents. Communciations of the ACM, 45(3):88-89, March.

Lennon, and J., Maurer, H. (2003): Why it is Difficult to Introduce e-Learning into Schools And Some New Solutions. In journal *J.UCS*, vol. 9, pp. 1244-1257, 2003.

Lewis, C. M., and Sycara, K. (1993). Reaching Informed Agreement in Multispecialist Cooperation. Group Decision and Negotiation 2(3): 279–300.

Litzinger, M.E., & Osif. B. (1993). Accommodating diverse learning styles: Designing instruction for electronic information sources. *In What is Good Instruction Now? Library Instruction for the 90s. ed.* Linda Shirato. Ann Arbor, MI: Pierian Press.

Maes, P. (1995) "Artificial Life meets Entertainment: Interacting with Lifelike Autonomous Agents." Special Issue on New Horizons of Commercial and Industrial AI, Vol. 38, No. 11, pp. 108-114, Communications of the ACM, ACM Press.

Maes, P. (1989). The dynamics of action selection. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), pages 991–997, Detroit, MI.

Maes, P. (1990). Situated agents can have goals. In Maes, P., editor, *Designing Autonomous Agent*s, pages 49–70. The MIT Press: Cambridge, MA.

Maes, P. (1991). The agent network architecture (ANA). *SIGART Bulleti*n, 2(4):115–120.

Mangal S. K.(1990).General psychology, *Sterling Publishers Private Ltd.*

Mason, C., and Johnson, R. 1989. DATMS: A Framework for Distributed Assumption- Based Reasoning. In Distributed Artificial Intelligence, Volume 2, eds. M. Huhns and L. Gasser, 293–318. San Francisco, Calif.: Morgan Kaufmann.

Mason, R., & Kaye, A.R.(1990) Toward a new paradigm for distance education. In L. M. Harasim (Ed.), *Online education: Perspectives on a new environment*. New York, NY: Praeger Publishers

McDonald J.; Gibson, C. C. (1998) 'Interpersonal dynamics and group development in computer conferencing', *The American Journal of Distance Education*, 12(1), p. 6-21,.

Morgan Kaufmann. Bradshaw, J.M. An Introduction to Software Agents. In: *Software Agents*, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 3-46.

Müller, J. P. (1994). A conceptual model for agent interaction. In Deen, S. M., editor, *Proceedings of the Second International Working Conference on Cooperating Knowledge*

Based Systems (CKBS-94), pages 213–234, DAKE Centre, University of Keele, UK.

Müller, J. P. and Pischel, M. (1994). Modelling interacting agents in dynamic environments. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94*), pages 709–713, Amsterdam, The Netherlands.

Müller, J. P., Pischel, M., and Thiel, M. (1995). Modelling reactive behaviour in vertically layered agent architectures. In Wooldridge, M. and Jennings, N. R., editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890*), pages 261–276. Springer-Verlag: Heidelberg, Germany.

Murray, T. and B. Woolf. 1992. Results of Encoding Knowledge with Tutor Construction Tools. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, July, pp. 17-23.

Myers, I.B. and McCaulley, M.H. (1985). A guide to the development and use of the Myers-Briggs Type Indicator. *Consulting Psychologists Press.*

Nabil, A., Awerbuch, B., Slonim, J. Wegner, P. & Yesha, Y. (1997). Globalizing business, education, and culture through the Internet. Communications of the ACM, 40(2).

Nii, H.P. Blackboard Systems. In: *The Handbook of Artificial Intelligence*, A. Barr, P.R. Cohen and E.A. Feingenbaum (Eds.), Addison-Wesley, New York, 1989, Volume IV, chapter XVI, pages 1-82.

Nwana, H. S. and Ndumu, D. T. 2000 A Perspective on Software Agents Research, Nwana, H. S. and Ndumu, D. T. IN: The Knowledge Engineering Review, 14(2), pages 125-142, Cambridge University Press.

Ohio (2004) Future of Distance and e-Learning in Ohio: A Report of the Ohio Learning Network Task Force on the Future of Distance and e-Learning in Ohio

Ohlsson, S. (1986). Some principles of intelligent tutoring. *Instructional Science* 14:293-326.

Ohlsson, S. (1987). Some Principles of Intelligent Tutoring. In Lawler & Yazdani (Eds.), *Artificial Intelligence and Education, Volume 1*. Ablex: Norwood, NJ, pp. 203-238.

Oliveira, F. M.. Critérios de equilibração para Sistemas Tutores Inteligentes. Porto Alegre: CPGCC da UFRGS, 1994. Tese de Doutorado.

Paredes, P., and Rodriguez, P. (2002). Considering Learning Styles in Adaptive Web-based Education. *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics en Orlando*, Florida, 481-485.

Pask G. (1975) Conversation, Cognition, and Learning. New York:Elsevier,1975.

Pereira, A.S.; D'Amico, C.B.; Geyer C.F.R. Gerenciamiento do Conhecimiento do ambiente AME-A. http://www.inf.ufrgs.br/~adriana/vcied.doc

Porter, L. R. (1997) Creating virtual classroom: Distance learning with the internet.

New York: J. Wiley & Sons.

Robert S. Gray. "Agent Tcl: A transportable agent system", Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95), Baltimore, Maryland, December 1995.

Rosatelli, M.C.; Self, J.A.; Thiry, M. LeCS: a collaborative case study system, in G. Gauthier, C. Frasson and K. VanLehn(eds.), Intelligent Tutoring Systems, Berlin: Springer-Verlag, p232-241, 2000.

Rowland, G. (1995). Instructional Design and Creativity: A Response to the Criticized. *Educational Technology*, 35 (5), 17-22.

Russel, S.; Norvig,P. 1996, Artificial Intelligence: A modern Approach. New Jersey:Prentice Hall,1996.

Schwier, R. A. (1995). Issues in emerging interactive technologies. In G. J. Anglin (Ed.) *Instructional technology: Past, present and future* (2nd Ed.), Englewood, CO: Libraries Unlimited, 119-127.

Self, John. (1990). Theoretical Foundations for Intelligent Tutoring Systems. *Journal of Artificial Intelligence in Education* 1 (4):3-14.

Sharon, S.(1980) Cooperative learning in small groups: recent methods and effects on achievement, attitudes and ethnic relations, *Review of Educational Research*, 50(2), 241-247.

Shoham,Y. (1993), Agent-oriented programming. Artificial Intelligence, 60,1,1993.

SILVEIRA, Ricardo. 2000 Modelagem Orientada a Agentes Aplicada a Ambientes Inteligentes Distribuídos de Ensino - JADE - Java Agent framework for Distance learning Environments Porto Alegre: CPGCC da UFRGS,. Dissertação de Mestrado. http://www.inf.ufrgs.br/~rsilv/Jade/jade.html

Sison. 1998. Learner Modelling and Machine Learning. *International Journal of Artificial Intelligence in Education* 9:128-158.

Smith, D.C.; Cypher,A.; Spohrer,J. KindSim, 1994,: Programming Agents Without a Programming Language. Communication of the ACM, 37,7,1994.

Specht, M. and Oppermann, R. (1998). ACE - Adaptive Courseware Environment. The *New Review of Hypermedia and Multimedia 4*, 141-161.

Spiro, R. J., Feltovich, P. J., Jacobson, M. J. & Coulson, R. L. (1992). Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domain. In T. Duffy & D. Jonassen (Eds.) *Constructivist and the technology of instruction*, Hillsdale, N.J.: Lawwrence Erlbaum Association Publishers, 57-75.

Steffe, L. P., & Gale, J. (Eds.) (1995). Constructivism in education. Hillsdale, NJ: Earlbaum.

Sycara, K.. Multiagent Systems. AI Magazine, vol. 18, nº 2, 1998.

Sycara, K.; Decker, K.; Pannu, A.; Williamson, M.; and Zeng, D. 1996. Distributed Intelligent Agents. IEEE Expert 11(6): 36–46.

Tahara Y., Ohsuga A., and Honiden S.. 2000, Safety and security in mobile agents. In *Draft Proceedings of AOSE2000*.

Thibodeau, Marc-André; Bélanger, Simon; Frasson, Claude. 2000 Matchmaking of User Profiles Based on discussion analysis Using Intelligent Agents. In Lectures Notes in Computer Science. Intelligent Tutoring Systems. Proceedings of 5th International Conference, ITS 2000, Montreal, Canada, June 2000. P.113-122.

VanLehn, K. (1990). *Mind Bugs*. Cambridge, MA: MIT Press.

Vassileva, J., Greer, J., McCalla, G., Deters, R., Zapata, D., Mudgal, C. And Grant, S 1999. *ARIES* Lab, A Multiagent Approach to the Design of Peer´help Environments. AIED'99 Submission,.

Vassileva, Julita; Detters, Ralph; Geer, Jim; Maccalla, Gord; Bull, Susan; Kettel, Lori. (2001) Lessons from Deploying I-Help. Workshop - Multi-Agent Architectures for Distributed Learning Environments. Proceedings of International Conference on AI and Education, San Antonio, Texas, May, 2001. P.3-11.

Vere, S. and Bickmore, T. (1990). A basic agent. *Computational Intelligenc*e, 6:41–60.

Vosniadou, S., and W. F. Brewer. (1989). The Concept of the Earth's Shape: A Study of Conceptual Change in Childhood. *Unpublished paper*.

Webber, Carine; Bergia, Loris; Pesty, Sylvie; Balacheff, Nicolas. (2001) The Baghera project: a multi-agent architecture for human learning. Workshop - Multia-Agent Architectures for Distributed Learning Environments. Proceedings International Conference on AI and Education, San Antonio, Texas, May, 2001.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann.

Wiederhold, G. Mediators in the Architecture of Future Information Systems. In: *IEEE Computer*, March 1992, pages 38-49.

Wiley, David A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D.A. Wiley (Ed.). *The Instructional Use of Learning Objects* [on-line]. Available: http://reusability.org/ read/.

Willis, B. (1994) (Ed.) Distance education: Strategies and tools. EnglewoodCliffs, NJ: Educational Technology Publications.

Wilson, B. G. & Cole, P. (1991). Cognitive dissonance as an instructional variable.

Ohio Media Spectrum, 43(4), 11-21.

Wong D., Paciorek N., and Moore D.1999, Java-based mobile agents. Communication of the ACM, 42(3):93-102, March.

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

APPENDIX A: FIPA AGENT COMMUNICATIVE ACT LIBRARY

Since communicative acts are central to FIPA ACL specification, a brief informal description of the different communicative acts is given as follows. To facilitate understanding better, each communicative act is followed by a simple example derived from partial agent performatives in MAGE.

- *Accept-proposal* is a general-purpose acceptance of a proposal that was previously submitted (typically through a *propose* act). The agent sending the acceptance informs the receiver that it intends that (at some point in the future) the receiving agent will perform the action, once the given precondition is, or becomes, true. E.g., the Pedagogic Agent informs the Teacher Agent that it accepts an offer from the Teacher Agent to communicate with each other by telephone.
- *Agree* is a general-purpose agreement to a previously submitted *request* to perform some action. The agent sending the agreement informs the receiver that it does intend to perform the action, but not until the given precondition is true. E.g. the Teacher Assistant Agent agrees to the request from Learner Agent that help the learner make a learning style or leaning strategy.
- *Cancel* allows an agent to stop another agent from continuing to perform (or expecting to perform) an action, which was previously requested. Note that the action that is the object of the act of cancellation should be believed by the sender to be ongoing or to be planned but not yet executed. E.g. the Course Assistant Agent asks Learning Content Management Agent to cancel the service of subscription to a specific leaning object because the course developer thinks that s/he no longer needs it.
- *CFP* is a general-purpose action to initiate a negotiation process by making a call for proposals to perform the given action. The actual protocol under which the negotiation process is established is known either by prior agreement, or is explicitly stated in the protocol parameter of the message. E.g. the Course Assistant Agent asks EEO Provider Agent to submit its proposal to provide a learning object that should meet the following conditions: Context=’continuous formation’, Interactive Level=’middle’, Difficulty=’high’, Keyword=’agent, communication’, Language=’French’, etc.
- *Confirm* allows the sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition. E.g. the Learner Agent

confirms to the EEO Provider Agent that the version of learner's Browser is 6.0.

- *Disconfirm* allows an agent to disconfirm the value of a proposition; the sender informs the receiver that a given proposition is false, where the receiver is known to believe, or believe it likely that, the proposition is true. E.g. the Learner Agent confirms to the EEO Provider Agent that the version of learner's Browser is not 6.0.
- *Failure* is an action of telling another agent that an action was attempted but the attempt failed. E.g. The DF Agent informs the Collaboration Agent that it cannot find some learners who have the common interests.
- *Inform* allows the sender to inform the receiver that a given proposition is true. E.g. the Learner Modal Agent informs the Pedagogic Agent that the learner is an active one.
- *Inform-if* is an abbreviation for informing whether or not a given proposition is believed. The agent which enacts an *inform-if* macro-act will actually perform a standard *inform* act. The content of the inform act will depend on the informing agent's beliefs. E.g. the Pedagogic Agent request the Lerner Model Agent to inform whether or not the learner is bad in memory.
- *Inform-ref* macro action allows the sender to inform the receiver some object that the sender believes corresponds to a definite descriptor, such as a name or other identifying description. E.g. the Pedagogic Agent requests Learner Model Agent to tell it the current learning progress of a specific course.
- *Not-understood* is a general-purpose action where the sender informs the receive that it does not understand the actions this agent did. The sender of the act (e.g. i) informs the receiver (e.g. j) that it perceived that j performed some action, but that i did not understand what j just did. A particular common case is that i tells j that i did not understand the message that j has just sent to i. E.g. the Leaner Modal Agent did not understand the *query-if* message from the Pedagogic Agent because it did not recognize the ontology.
- *Propose* is a general-purpose action to make a proposal or respond to an existing proposal during a negotiation process by proposing to perform a given action subject to certain conditions being true. The actual protocol under which the negotiation process is being conducted is known either by prior agreement, or is explicitly stated in the *protocol* parameter of the message. E.g. the Teacher Agent proposes to the Pedagogic Agent of a leaner that s/he wants to discuss with the learner by telephone.
- *Query-if* is the act of asking another agent whether (it believes that) a given proposition is true. The sending agent is requesting the receiver to *inform* it of the truth of the proposition. The Pedagogic Agent asks Evaluation Agent if the learner has completed

the post-assessment.

- *Query-ref* is the act of asking another agent to inform the requestor of the object identified by a definite descriptor. The sending agent is requesting the receiver to perform an *inform* act, containing the object that corresponds to the definite descriptor. E.g. The Course Assistant Agent asks the Learning Content Agent the available service.
- *Refuse* is performed when the agent cannot meet all of the preconditions for the action to be carried out, both implicit and explicit. For example, the agent may not know something it is being asked for, or another agent requested an action for which it has insufficient privilege. E.g. An EEO Assistant Agent refuses the Course Assistant Agent to modify a specific EEO because the EEO developer has no time at the moment.
- *Reject-proposal* is a general-purpose rejection to a previously submitted proposal. The agent sending the rejection informs the receiver that it has no intention that the recipient performs the given action under the given preconditions. E.g. the Pedagogic Agent informs the Teacher Agent that it refuses an offer from the Teacher Agent to communicate with each other by telephone.
- *Request* denotes that the sender is requesting the receiver to perform some action. The content of the message is a description of the action to be performed, in some language the receiver understands. The action can be any action the receiver is capable of performing: pick up a box, book a plane flight, and change a password etc. e.g. the Pedagogic Agent request the Learner Model Agent to offer the score of test 5.
- *Request-when* allows an agent to inform another agent that a certain action should be performed as soon as a given precondition, expressed as a proposition, becomes true. E.g. when the pre-assessment is completed.
- *Request-whenever* allows an agent to inform another agent that a certain action should be performed as soon as a given precondition, expressed as a proposition, becomes true, and that, furthermore, if the proposition should subsequently become false, the action will be repeated as soon as it once more becomes true. E.g. the Pedagogic Agent requests the Evaluation Agent to notify it whenever the case that the score of any assessment part is blow 20 occurs.
- *Subscribe* is the act of requesting a persistent intention to notify the sender of the value of a reference, and to notify again whenever the object identified by the reference changes. E.g. Course Assistant Agent tells Learning Content Management Agent to notify it whenever an intended learning object appears in MAGE.

APPENDIX B: MAS BASED E-EDUCATION SYSTEM

This annex briefly describes some of the most recent approaches of MAS based systems oriented to education:

- White Rabbit (Thibodeau et al, 2000). The White Rabbit system intends to enhance cooperation among a group of people by analyzing their conversation. Each user is assisted by an intelligent agent, which establishes a profile of his or her interests. Next, with its autonomous and mobile behavior, the agent will reach the personal agents of other users to be introduced and presented to the ones that seem to have similar interests. A mediator agent is used to facilitate communication among personal agents and to perform clustering on the profiles that they have collected. Conversation between users takes place in a chat environment adapted to the needs of the system.
- LeCS (Learning from Case Studies). This is an intelligent system for remote education that has, according to Rosatelli et al. (2000), an architecture based on a Federal System of agents. LeCS supports web-based distance learning from case studies, allowing collaborative learning between a group of learners that is geographically dispersed. It provides the necessary tools to carry out the case solution development and accomplishes functions that altogether assist the learning process. LeCS is used to give CSCL (Computer Supported Collaborative Learning) through the Web. The method of machine used Learning is based on CBR (Case Based Reasoning). In this agent-agent architecture, direct communication does not exist, but all is done by a special agent called Facilitator. This Facilitator is in charge to store all the information needed for the communication. Three types of agents exist in the system:

1. Agent Interface: stores the individual interactions of each user
2. Information Agent: stores information regarding didactic materials (HTML page, images, interactions on the chat, etc) and keeps a knowledge base for the solutions of the developed cases
3. Advisor Agent: has mechanisms to guess situations in which an aid to the user is needed

- Baghera (Webber et al., 2001). The Baghera platform is founded on the principle that the educational function of a system is an emerging property of the interactions organized between its components: agents and humans, and not a mere functionality of one of its parts. Their first achievements include a web-based multi-agent architecture for learning environments and an operational prototype for the learning of geometry. Learners and teachers interact with different agents, according to the activities they will carry on and

the educational approach of Baghera. Each learner is supported by three artificial agents:

1. Learner's Personal Interface Agent: associated with the learner's interface
2. Tutor Agents: can interact with mediator agents, assistant agents and other tutors
3. Mediator Agent: the aim of this agent is to choose an appropriate problem solver to send the learner's solutions

In a similarly way, two artificial agents give support to each teacher:

1. Teacher's Personal Interface Agent: associated with the teacher's interface
2. Assistant Agent: a kind of personal agent whose goals include assisting the teacher with the creation and distribution of new activities, which are kept in the teacher's electronic folder.

- Help (Vassileva et al., 2001). I-Help is based on a multi-agent architecture, consisting of personal agents (of human users) and application agents (of software applications). These agents use a common ontology and communication language. Each agent manages specific resources of the user (or application) it represents, including for example, the knowledge resources of the user about certain concepts, or the instructional materials belonging to an application. The agents use their resources to achieve the goals of their users, their own goals, and goals of other agents. Thus all the agents are autonomous and goal-driven. In their goal pursuit the agents can also use resources borrowed from other agents, i.e. they are collaborative. For this, they have to negotiate. Each agent possesses a model of its user and of other agents; it has encountered and negotiated with. The agents communicate with each other and with matchmaker agents to search for appropriate help resources for their users, depending on the topic of the help-request. If an electronic resource is found (represented by application agents), the personal agent "borrows" the resource and presents it to the user in a browser. However, if a human helper is located, the agents negotiate the price for help, since human help involves inherent costs (time and effort) for the helper. Help is arranged (negotiated) entirely by the personal agents, thus freeing the learner from the need to bargain and think about the currency spent / earned. In this way the personal agents trade the help of their users on a virtual help market. Thus the multi-agent architecture involves various levels of organization, including the negotiation between agents, an economical model and control / policing institutions. In this way, we achieve a distributed (multi-user, multi-application) adaptive (self-organized) system that supports users in locating and using help resources (other users, applications, and information) to achieve their goals.
- AME-A (D'Amico et al., 1997; D'Amico et al., 1998; Pereira et al., 2001). AME-A is an education-learning multi-agent, which sets out the study and the development of an interactive educational system for education. The proposal is generic and adapts education to the psico-pedagogical characteristics of the learner. The system uses both static learning

and dynamic one. The static learning corresponds to the first interaction of the learner with the environment, where an agent models the apprentice according to his/her affective characteristics, motivation and level of knowledge. The dynamic learning takes places during the interaction, when the learner model (like the pedagogical strategies in force) is validated.

- Electrotutor (D'Amico et al., 1997; D'Amico et al., 1998; Pereira et al., 2001). Electrotutor III implements distributed environments of intelligent education learning based on a multi-agent architecture for teaching Physics. Agents dynamically perceive the conditions of their environment and make decisions to change it. Seven agents, each one of them with a specific function compose the society of agents. In order to be able to act on the environment, each agent has an internal partial representation of the world that surrounds it. The metaphor of mental states is used to model this way the knowledge base that represents the states of the environment where the agent is living. The seven agents are:

1. Dominion Managing Agent: recovers information referring to the dominion on which the learner is going to work
2. Exercises Managing Agent: provides exercises and their answers to the learner
3. Examples Managing Agent: provides examples to the learner
4. Activities Managing Agent: in charge to provide extra activities to the learner
5. Learner Model Agent: in charge to construct and to maintain a knowledge base that models the state of the learners who are or been have connected to the system
6. Agent Interface: controls what appears on the Navigator (an agent interface exists per learner).
7. Communication Managing Agent: in charge of the communication of each agent Interface with the others.

- JADE (Silveira, 2000). This environment contains a special agent responsible for each teaching strategy developed, that is, for the domain knowledge retrieval over each point to be presented to the learner, for the task of proposing exercises and evaluating proposals, examples and extra activities. JADE architecture encompasses, therefore, a Multi-Agent environment composed of an agent responsible for the system general control (Learner s Model), and a Communication Manager and other agents (Pedagogical Agents), which are responsible for tasks related to their teaching tactics, where each agent may have its tasks specified according to its goal. All actions of learner's data accessing are taken by the Learner s Model, thus when a pedagogical agent is required to update the learner's historic, this agent sends to the Learner Model data to be updated, as well as any other change in the learner s state of teaching.
- GRACILE (Ayala-Yano, 1996; Ayala-Yano, 1998). For Computer-Supported Collaborative Learning (CSCL) environments, they propose intelligent agents that assist the

learners and cooperate in order to create possibilities of effective collaboration in a virtual community of practice. They have developed two kinds of software agents: mediator agents that play the role of facilitators that support the communication and collaboration among learners, and domain agents, which provide assistance concerning the appropriate application of domain knowledge in the network. Mediator agents cooperate exchanging their beliefs about the capabilities, commitments and goals of the learners. Doing this each mediator agent is able to construct a representation of its learner's collaboration possibilities in the group (referred as the learner's group-based knowledge frontier), considering the social and structural aspects of knowledge development. The mediator agent proposes the learner to commit to tasks that require the application of knowledge elements in the learner's group-based knowledge frontier, which results in an increment of the collaboration possibilities between learners, the creation of zones of proximal development, and therefore more learning possibilities.

- A Computational Model of Distance Learning Based on Vygotsky's Social-Cultural Approach (Andrade et al., 2001). This framework is based on Vygotsky's social-cultural theory and is designed as a multi-agent society supporting distance learning. The goal of this research is to propose an environment that privileges collaboration as form of social interaction, through the use of language, symbols and signs. To support collaborative learning, they present a society formed of the following artificial agents: ZPD agents, mediating agents, semiotic agent and social agent; it also involves human agents who have either the role of tutors or learners.

- ABITS: An Agent Based Intelligent Tutoring System for Distance Learning (Capuano, 2000). ABITS is able to support a Web-based Course Delivery Platform with a set of "intelligent" functions providing both learners modeling and automatic curriculum generation. Such functions found their effectiveness on a set of rules for knowledge indexing based on Metadata and Conceptual Graphs following the IEEE Learning Object Metadata (LOM) standard. Moreover, in order to ensure the maximal flexibility, ABITS is organized as a multi-agent system composed by pools of three different kinds of agents (evaluation, pedagogical and affective agents). Each agent is able to solve in autonomous way a specific task and they work together in order to improve the WBT learning effectiveness adapting the didactic materials to user skills and preferences.

APPENDIX C: ABBREVIATION

AAgent—Allocatee Agent
ACC—Agent Communicative Channel
ACL—Agent Communication Language
ADL —Advanced Distributed Learning
AID —Agent Identifier
AMS —Agent Management System
AP—Agent Platform
API — Application Programming Interface
AUML—Agent Unified Modeling Language
CAA—Course Assistant Agent
CA—Courier Agent
CAgent— Course Agent
CAI—computer aided instruction
CAL—collective adaptive learning
CBT— Computer-based training
CGI — Common Gateway Interface
CIA—Course Interface Agent
CMC—computer mediated communication
CommAgent—Communicative Agents
CPA— Course Provider Agent
CSCL— Computer-supported collaborative learning
DAgent— Diagnostic Sgents
DF — Directory Facilitator
EAAS—E-Assessment agent system
EEOAA —EEO Assistant Agent
EEOIA— EEO Interface Agent
EEOPA— EEO Provider Agent
ExamCenterAgent—Exam Center Agent
FAQ—AFAQ agent
FIPA—Foundation of International Physical Association
FSLSM —Felder-Silverman Learning Style Model

GACtrlAgent —Genetic Algorithm Control Agent
GA—genetic algorithm
GA—Genetic Algorithm
GA—Group Agent
GAMASTG—Genetic Algorithm Based MAS Test Generation system
GUI —Graphic User Interface
GUID —Globally Unique IDentifier
HSAgent— Help Service Agents
IAL—individual adaptive learning
ICAI—Intelligent Computer Aided Instruction
ICT—Information and Communication Technology
ILE—Intelligent Learning Environment
ILS—Index of Learning Style
IP — Interaction Protocol
ITS—Intelligent Tutoring Systems
J2EE —Java 2 Enterprise Edition
JADE —Java Agent DEvelopment Framework
JVM—Java Virtual Machines
KM— Knowledge Management
LAA —Learner Assistant Agent
LCA—Learning Collaboration Agent
LEA—Learning Evaluation Agent
LMA —Learner Model Agent
LO—Learning Object
LOM—Learning Object Metadata
LPA—Learner Profile Agent
MAEA—Mobile Agent E-Assessment System
MAGE— MAS Based e-Education System
MAgent—Matchmaking Agent
MAS—Multi-Agent System
MEEOCAS—Mas based EEO Course Authoring System
MTS —Message Transport Service
PA—Pedagogic Agent
PGA—presentation generation agent
PHAS—Peer Help Agent System

RIO —Reusable Information Object
RMA—Remote Monitoring Agent
RSAgent — Register Service Agent
SCORM—Sharable Content Object Reference Model
TAA—Tutor Personal Agent
TC— target concepts
TDSATest —Delivery Service Agent
TGSA—Test Generation Service Agent
TPAgent—Test Paper Agent
WBT—Web Based Testing
XML— eXtensible Markup Language XML

LIST OF PUBLICATIONS DURING DOCTORAL STUDY

1. MENG Anbo, Luqing Ye Daniel Roy, Pierre Padilla,"***Genetic Algorithm Based Multi-Agent System Applied to the Automatic Generation of Test Papers***", Computers & Education, 2006 (SCI, accepted).

2. MENG Anbo, Luqing Ye Romuald Stock Pierre Padilla, "***A Multi-agent enabled e-Education Object & Course Authoring System***", International Journal of Distance Education and Application, Vol. 2. No.8, p3-15, 2005.

3. MENG Anbo, YE Luqing,et al., "***Application of Genetic Algorithm in Adaptive Governor with Variable PID Parameters***", Control Theory and Application , Vol. 3. CN 44-1240/TP, p398-404, 2004 (EI).

4. MENG Anbo, YIN Hao et al., "***A Niw 2-D Fuzzy Control System in Traffic Lights Based on PLC***", Journal of Electrical Automation, Vol. 4. CN 32-176/TM, p8-11, 2004.

5. MENG Anbo, YIN hao et al., "***Application of Neural Network in the Modeling of Hydro Turbine***", Journal of.DADianji, vol. 6.CN 23-1253, p39-43, 2003.

6. MENG Anbo, YIN hao et al.,"***A New Method for Calculation of Water Hammer Based on Precise Model***", Journal of North China Institute of Water Conservancy and Hydroelectric Power, Vol.4. CN41-5058/TV, p44-50, 2003.

7. MENG Anbo, YIN hao et al.,"***Application of S-Function to the Non-Linear Modeling For Hydro-Turbine***", Journal of North China Institute of Water Conservancy and Hydroelectric Power, Vol.4. CN41-5058/TV, p38-40, 2003.

.