e-Advisor: A Multi-agent System for Academic Advising

Fuhua Lin Steve Leung Dunwei Wen Frank Zhang Kinshuk Rory McGreal

Athabasca University, Athabasca, Alberta, Canada, T9S 3A3
Email: [oscarl, stevel, dunweiw, frankz, kinshuk, rory]@athabascau.ca
Tel: 1-866-53998664
Fax: 1-780-675-6186

Abstract: This paper presents an approach to tackle a dynamic and complex individualized study planning and scheduling problem utilizing multi-agent system approach and ontology-driven methodology. To support the approach a web-based multiple intelligent agents system called e-Advisor is developed and tested by users. The paper describes the various types of agents used in e-Advisor, the development of the ontologies and their applicability, system implementation issues and a preference-driven planning algorithm used by the agents. Empirical results show that the architecture and algorithm are effective.

Keywords: Academic Advising, Knowledge Modeling, Multi-Agent Systems, Intelligent Systems

1. Introduction

The purpose of academic advising is to help an advisee to develop and accomplish personalized educational plans [1]. Research shows that a high percentage of students are dissatisfied with the advising experience, which was in fact one of the greatest hindrances for them to attaining a degree [2]. For many universities, the ratio of students per advisor is high and further increasing as both the student population and faculty responsibilities increase, resulting in inadequate and potentially inappropriate advice to students. Inadequate and incorrect advice can cause dissatisfaction and frustration as students might take unnecessary courses or courses without proper prerequisites and could eventually lead to dropout or delayed graduation [2]. The issue of advising appears more prominent for institutions primarily offering distance credentials where academic face-to-face interviews are not feasible.

While academic advising is playing an increasingly important role in strengthening academic performance and student retention, giving advice about program study is difficult for academic advisors for the following reasons:

- All students in degree programs have to balance personal and career objectives, preferences, and financial and temporal constraints against degree requirements, course availability, and course interrelationships.
- An institution offering a program is subject to many unforeseeable constraints. For example many institutions cannot guarantee the availability of courses beyond a term or two into the future as course offerings depend on a number of factors including course curriculum projections based on uncertain and incomplete information.
- Courses delivery list and course prerequisites are *dynamically* changing as the program curriculum is developed and maintained via course revisions, additions, and closures.

The objective of this research is therefore to propose an approach to designing and developing intelligent systems which provide computer-supported cooperative educational environments to tackle a dynamic and complex individualized study planning problem.

Since 2003 we have been developing *e-Advisor* (http://www.e-advisor.org), an intelligent system which facilitates academic advising and program planning for students in the Master of Science in Information Systems (MSc IS) program of Athabasca University of Canada.

e-Advisor is a Web-based intelligent system that facilitates academic advising and program planning to meet MSc IS students' needs in seeking advice and therefore reducing the advisors' workload, and to support decision-making of MSc IS program administrators in scheduling courses. This paper focuses on the architecture and knowledge models of the system.

The rest of the paper is organized as follows. In Section 2, we will summarize the related work and in Section 3 we will present an architectural design of e-Advisor with focus on multiple intelligent agents approach. In Section 4 we will describe a Preference-Driven Planning Algorithm used by the Student and Advisor agents. Section 5 briefly introduces the implementation of e-Advisor. Section 6 presents the result of an empirical study on the overall usability and effectiveness of e-Advisor. Conclusion, discussion and future work will be summarized in Section 7.

2. Related work

To overcome these problems in a more comprehensive manner than possible with basic tools for student and advisor management, many Expert Systems and Decision Support Systems (DSS) for academic advising have been developed or reported since the 1980s [3] [4] [5] [6]. Turban *et al.* (1988) reviewed various approaches to the use of DSS in academic administration [3]. The use of a DSS for academic advising has been investigated by Murray and Le Blanc [4]. Carroll and Chappell (1996) describe a system to assist students at Sam Houston University in Texas [5].

We found that previous computer-based advising systems research suffers from two drawbacks related to limited scope and insufficient domain knowledge modeling and representation:

First, these systems are either research prototypes which focus on only one or two system components or are intended more as support tools that allow the user to browse through required courses than as truly intelligent advisors that can actively guide the students and react to changes. Second, effective computer-aided program planning and advising hinges on the modeling and representation of knowledge about the domain, program curriculum model, students, and the procedural knowledge of problem-solving. Yet the modeling of this kind of knowledge has not been fully addressed in previously proposed research.

Paquette & Tchounikine (1999) proposed a knowledge engineering method for the construction of advisor systems [7].

User modeling is a key factor in system personalization. Kay (1999) [8] and Chen & Mizoguchi (1999) [9] noted the advantage of using ontologies for learner/user models.

Mizoguchi and Bourdeau (2000) studied how ontologies can help to overcome problems in artificial intelligence in education [10]. Razmerita, et al., (2003) proposed a generic ontology-based user modeling architecture [11]. Ontology Language (OWL) proposed by W3C is used to publish and share ontologies, supporting advanced Web search, software agents and knowledge management. Protégé-OWL is a knowledge-based systems editor and browser allows domain experts to build knowledge-based systems by creating and modifying reusable ontologies. There is little effort to formalize the curriculum knowledge representation and manipulation. Zhou etc. (1996) proposed a topic association graph (TAG)-based formulation in which the precedence relations among topics are derived from the prerequisite relations over the domain concepts [12].

3. Architectural Design

3.1 Situational analysis

A typical student usually completes a program within a set period of time, measured in semesters or years. Each semester the student plans or re-plans his/her study schedule and selects courses to be taken in that semester. The student may seek advice from his/her academic advisor during this planning process.

To help a student, the advisor may need to review the profile of the student to understand her/his progress and performance, and identify his/her goals of learning and preferences. Among other things, the advisor needs to know the program structure (e.g. prerequisite relationships among courses) and regulations, and the course delivery schedule.

The program structure and regulations are subject to constant changes such as new course additions, existing course revisions and cancellations. Course delivery schedules are usually determined by the program administrators through considering the availability and workload of instructors and predicting course enrollments. Another factor for consideration is the estimation of expected enrollments for potential courses through analysis of the student profiles or student surveys. Moreover, the course schedule of a program may also need to be changed or a course offering may need to be cancelled during the registration period for reasons such as low registration.

Many universities' and colleges' academic programs nowadays use online advising systems. However, most of these systems focus generally on student information management without addressing the preferences of students and the kinds of constraints discussed above.

3.2 Multi-Agent system architecture

Based on the preceding situational analysis and the fact that academic advising process consists of several interacting stakeholders, we decided to employ the approach of multi-agent systems (MAS) as the core architecture of e-Advisor. The reason for adopting a MAS architecture is that MAS technologies provide a new way of dealing with complexity, distribution, and interactivity. MAS technologies have been widely used to facilitate support for human-centered computing (HCC) [13], and advanced learning environments [14]. In particular, we identified the following characteristics of academic advising problem that land themselves to MAS approach:

First, academic advising is an iterative process due to the continuous change of the environment.

Second, the advising problem is too complex for a centralized algorithm to solve. For example, the academic advising module and program planning module are not self-contained but interact with each other and this is facilitated by cooperation, coordination and negotiation support usually found in multi-agent systems.

Third, the nature of academic advising in educational environments imposes requirements concerning the scalability of the underlying architecture and the integration of heterogeneous software. MAS technologies which have emerged from a convergence of technologies in distributed object systems and distributed artificial intelligence help manage this complexity because they support the seamless integration of heterogeneous components [15] [16] [17]. Web-based technologies in conjunction with multi-agent methodology form a new trend in modeling and development of learning environments [18]. It is demanded that distributed learning environments provide smarter or more intelligent learning functions that offer personal services with capabilities to learn, to reason, and to be autonomous and totally dynamic [19].

Figure 1 depicts the core architecture of e-Advisor. The system consists of four types of user agents, namely, Student Agent, Administrator Agent, Instructor Agent and Advisor Agent, and two types of resource agents: Ontology Agent and DB & KB Agent.

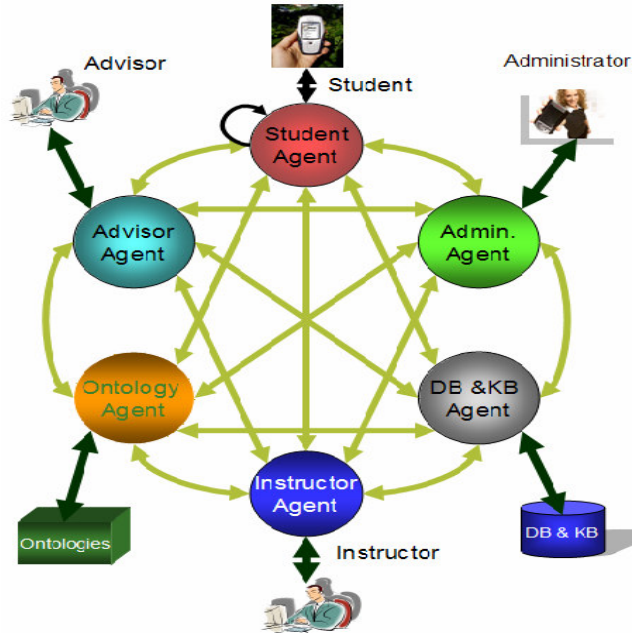

Figure 1: Overall Architecture of e-Advisor.

3.3. Ontologies

E-Advisor incorporates the use of multiple ontologies, including a course ontology, a topic ontology, a student ontology, and a program ontology. These ontologies are modeled using Protégé-OWL (http://protege.stanford.org).

A course in a program consists of various topics, defined explicitly or implicitly, and the prerequisite relationships between the topics. Higher-level courses often have lower-level prerequisite courses. The instructor of a course usually determines the prerequisite courses. As courses are changed by instructors, the prerequisite relationships among the courses in the program need to be updated accordingly.

Course Ontology: The Course Ontology describes the courses of the programs of MSc Information Systems. The are represented in OWL has the following properties: hasCourseDescription, hasCourseTitle, isWritteninLanguage, hasTopicCovered, hasCoursePrerequisite, hasCourseSyllabusURL, isCoordinatedby, and isDeliveredThrough.

Topic Ontology: The Topic Ontology describes the topics at the program level in the domain of MSc Information Systems. An example of the topics is "*Programming*". It also includes the relationships between the topics and some basic properties. An example of a relationship among the topics is '*is prerequisite of*'. For example, topic "*Data and Object Structures*' is a prerequisite of topic '*Data Management*'.

Student Ontology: The Student Ontology includes the following information:

- Student's basic information (such as home address, office phone, email, occupation, job title etc.);
- Courses taken and passed by the student;
- Preferences;
- Planning parameters and plan selected.

Program Ontology: The Program Ontology is used to define the logical structure of the program, describing the vocabulary that conceptualizes the curriculum. Some examples of the properties of Class Program are hasMaxSpanTimeInYear, hasMinimumNumberofCredit, hasSemestersEachYear, hasTotalCreditRequirement, MinimumNumberOfCoursesPerYear.

3.4 Ontology Agent

The ontologies are accessed through an Ontology Agent which is designed to extract and maintain planning knowledge from existing repositories using information retrieval methods. The planning knowledge that can be extracted includes topics and their prerequisite relationships, and the relationship between job objectives and topics. The sources of data used to obtain the planning knowledge include the course catalogs and students' profiles in the program.

The Ontology Agent is also responsible for accessing the ontologies, performing querying and reasoning across multiple ontologies, and providing an integrated ontology service to ontology clients through an *OWL-Connector*. The *OWL-Connector* allows the agents in the system to retrieve ontology knowledge represented in OWL from Ontology Agent. For example, the Instructor Agent needs to get the topics and the prerequisite topics from an OWL file of the Ontology Agent to run the matching algorithm and drive the Web interface. The OWL-connector is a Java class called *EadvisorOWLConnector.java* that uses methods from the Protégé-OWL Java API, which is based on the Jena API (http://jena.sourceforge.net/).

3.5 Student Agent

The Student Agent is mainly responsible for identifying students' planning requirements and preferences, and maintaining the students' models. Once a student enters a program, the student can activate an instance of a Student Agent via a Web-based interface to

- provide the basic information needed to build the profile of the student; and
- configure the scheduling requirements and preferences.

Figure 2 shows a snapshot of a typical student agent interface.

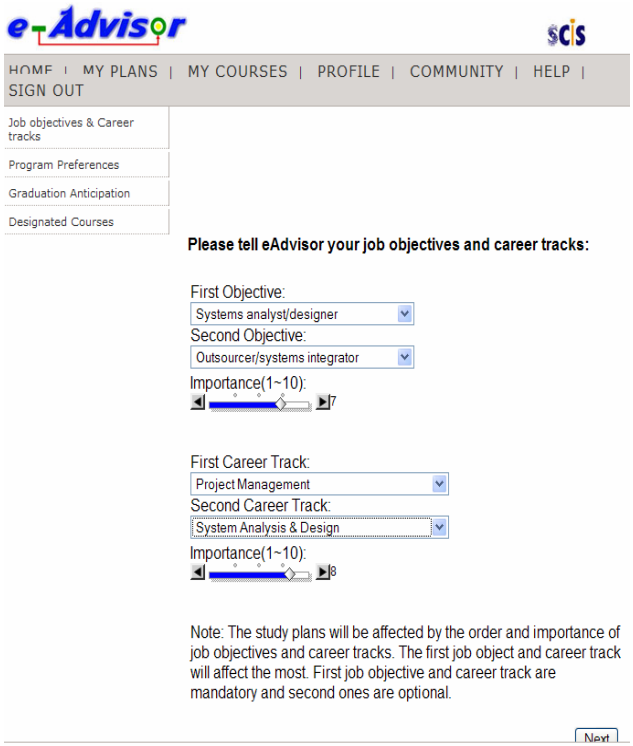

Figure 2: A snapshot of a typical student agent interface.

The preferences for course selection and study planning of a student include job objectives, career tracks, assessment style, thesis route, and the degrees of importance of these preferences. The student can also select how he/she receives notifications and alerts from the agent. The scheduling requirements and constraints of a student include desired graduation year or semester, the number of courses to be taken for each of the remaining semesters, and some particular courses to be taken in a specific semester.

As an intelligent agent, the Student Agent is able to negotiate with the Administrator Agent about the course delivery schedule and communicate with the Advisor Agent to generate the optimal study plans.

3.6 Instructor Agent

The task of the Instructor Agent is to construct and maintain the content-based prerequisites of the courses the instructors teach. An algorithm, called the Course Prerequisite Matching Algorithm (CPMA), has been developed. The basic idea behind the algorithm is that courses and their prerequisites can be represented by a common set of topics. Prerequisites thus can be best determined by examining the topics covered by the course and their relationships to the topics covered by other courses.

Instructors are responsible for maintaining the metadata for their courses during the life-cycle of the course.

To determine the topics and prerequisites, the Instructor Agent requests for a list of topics of the domain. The instructor can view, add or remove topics from the list returned before invoking CPMA. If the instructor feels that the results are appropriate, he/she can ask the agent to send a message to the Administrator Agent to add the selected prerequisite courses into the program model. If the content in the program model is changed, the Administrator Agent will send a message to the Instructor Agent to update the course's prerequisites.

3.7 Administrator Agent

The Administrator Agent has two responsibilities: maintaining the curriculum model and conducting course curriculum projection. New course entry is created by the administrator through a web-based interface. The administrator chooses options such as the course's code, title, instructor, and building block type. After the entry is created, the Administrator Agent sends a message to the corresponding instructor telling him or her that the course is ready for topics to be added and for the content-based prerequisites to be specified. Figure 3 is as a snapshot of the interface for the administrator.

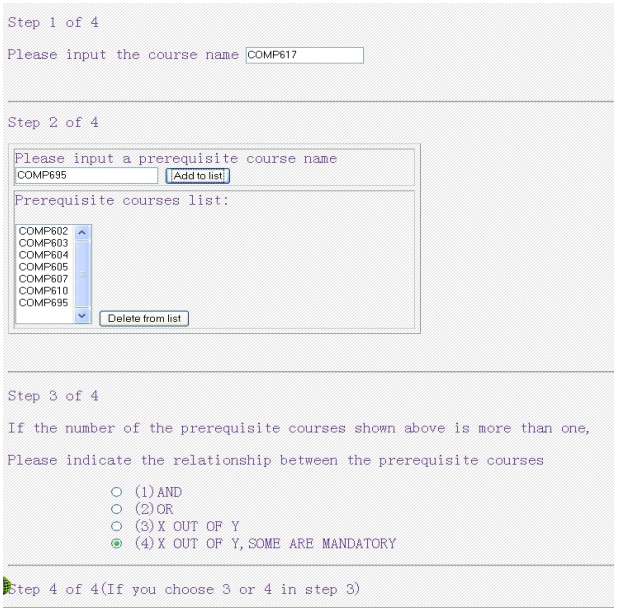

Figure 3: Interface for the Administrator.

The Administrator Agent performs negotiation process with Student Agents, maximizes the enrollments while meeting the faculty workload constraints, and works out a course delivery schedule before the registrations start for a particular semester.

3.8 Advisor Agent

The Advisor Agent is responsible for (1) generating the course-taking plans for students, and (2) evaluating the status of the students.

Evaluation of a student's status primarily consists of determining to what extent the student has executed against his or her goals and plans, and whether the student has met the program regulations.

The Advisor Agent constructs an optimized course-taking plan for the student, using the planning requirements and preferences, student profile, course delivery schedules, program regulations and structure, and knowledge provided by student's advisor and program regulations. The Advisor Agent requests these data by sending requests to the appropriate DB & KB Agents.

4. Preference-Driven Planning Algorithm

The Advisor Agent uses a utility-driven scheduling algorithm to generate plans. In program scheduling, a feasible schedule for a given set of scheduling parameters for a student must satisfy the constraints about course availability, program regulations, and course prerequisite relationships. Usually, many feasible schedules can be produced for a given set of preferences and constraints. The objective is to provide the student with an optimal schedule from these feasible schedules.

4.1 Petri Net-represented Curriculum Model

A curriculum model consists of a set of courses and their prerequisite relationships, and a set of regulations and rules governing the program. The courses are indexed with metadata supplied by course authors or instructors. Two of the most important metadata of a course are the topics the course covers and the course's prerequisite courses. All courses deal with one or more domain topics in the program Topic Ontology of the MSc IS program. The dependency relations among the courses, such as prerequisites and co-requisites, are used for two purposes:

- To ensure that the plans generated by the Student Agent are correct in that they fulfill the prerequisite requirements of the program; and
- To capture the natural parallelism among the courses so that the planning agent can exploit inherent concurrent characteristics of the curriculum to optimize the span time of the program of study of a student.

There are two types of prerequisite relationships: (1) Content-based prerequisites; and (2) Curriculum-based prerequisites. The content-based prerequisites are provided by the Instructor Agent as described in the Instructor Agent section of this paper.

The results from CPMA are one or more alternative groups $G_1, G_2, \ldots, G_r$ of prerequisite courses of course c as follows:

$$G_i(c) = R, \quad i = 1, 2, \ldots, r.$$

So the form of the prerequisites of course c is

$$G_1(c) \text{ or } \ldots \text{ or } G_r(c) \qquad (5)$$

We call this *content-based prerequisites representation*.

Curriculum-based prerequisites: The regulations of MSc IS curriculum classify two types of qualifications: *IS Foundation* and *IS Core*. To complete *IS Foundation*, a student must pass three of the five foundation courses. To complete *IS Core*, a student must complete four out of seven core courses. This definition can be expressed by an *'n OUT OF m'* condition expression formalized as follows:

$$P(c)=n \text{ OUT OF } m:\{n; C_1, C_2, \ldots, C_m\} \ (n>0, n\leq m) \qquad (6)$$

In other words, to meet the prerequisite condition, a student must complete n courses out of m courses $C_1, C_2, \ldots, C_m$.

The prerequisites of a course c can consist of one or more such expressions. For example, course COMP667 (Intelligent Software Agents) has the prerequisite "IS Foundation and IS Core". Therefore, we can formalize *the curriculum-based prerequisites* of a course as a union set of condition expressions: $P_1(c)$ and $P_2(c)$ and … and $P_l(c)$, denoted as $\bigcap_{j=1}^{l} P_j(C)$ ($l > 0$). Figure 3 shows a snapshot for the interface for the administrator to enter curriculum-based prerequisites of courses.

Combining (5) and (6), the prerequisites Pr(c) of a course can be expressed as:

$$G_1(c) \cap \bigcap_{j=1}^{l} P_j(c) \text{ or } \ldots G_r(c) \cap \bigcap_{j=1}^{l} P_j(c) \qquad (7)$$

We adopt Petri nets [20] to represent the curriculum model, where places represent the status of a student's progress through the program and transitions represent the courses, and Pr(c) in (7) corresponds to the firing rules for a transition. The mandatory prerequisite relations are expressed as *arcs* of the Curriculum Petri nets. The Curriculum Petri nets can be used to check the correctness, consistency and completeness of a program.

4.2 Preference Fitness

We denote the set of preference types as **P** = { **Pi** }. In e-Advisor, we identified four *preference types* as shown in Table 1.

Table 1: Preference Types and Preference Items in e-Advisor

Preference Type	*Preference Name*	*Preference Items*
P1	Job Objective	JO01, JO02, …, JO14
P2	Career Track	CT01, CT02, …, CT15
P3	Specialization	SP01, …, SP05
P4	Assessment Style	AS01, AS02, AS03

Each preference type P_i contains a group of *preference items*, denoted as $\mathbf{PI_i}$ *(i = 1, 2, 3, 4)*.

For each preference type, a student is allowed to select one or more preference items. If multiple preference items are selected for a preference type, the selection order indicates relative importance to the student.

Different preference types have different degrees of importance in the scheduling depending on the individual student. For example, a learner might have a weak sensitivity to course '*Assessment Style*' but a strong sensitivity to the difference in '*Job Objectives*'. To rank them quantitatively, a set of weights $\mathbf{W_P} = \{w_i: w_i \in [0, 1], i = 1, 2, 3, 4; \sum_{1}^{4} w_i = 1.\}$ are assigned to each preference type by the student, indicating the degrees of importance of the preferences in the planning. The larger the weight, the more important the preference type is considered in course selection.

The agent can assist the student via prompts. With the help of the Ontology Agent (see Section 3.3), the Student Agent can detect any conflict in the requirements and preferences input by the student.

In order to decide what courses should be selected first for a given preference item, we calculate the relationships between the preference items about job objectives, career tracks, and specializations and courses in the program. As courses are made up of topics and topics have a preference item relationship, this is done using the relationship between the preference items and the topics that make up the courses.

For a preference item p and course c in a course set C in the program, we denote the degree of preference of c towards p as $m(p, c)$. For a preference item p in Preference P_4

(assessment style) and course c, $m(p, c)$ is the percentage λ of the marking scheme of the course. For a preference item p in preference type P_1, or P_2, or P_3, $m(p, c)$ is the sum of the degree of preference between p and each topic that the course c covers. Thus,

$$m(p,c)=\begin{cases}\sum_{i=1}^{h} m(p,t_i), & if\ p\in P_1\cup P_2\cup P_3\\ \lambda & if\quad p\in P_4\end{cases} \qquad (1)$$

Where $m(p, t_i)$ is the degree of preference between a preference item p and a topic t_i; h is the number of topics that course c covers. For a given p, the larger the value of $m(p, c)$, the more preferable course c is considered.

The mapping relations $m(p, t)$ are represented as a group of relational tables and stored in the knowledge base of the Advisor Agent. The human advisor can login to a Web-based GUI to maintain this mapping knowledge. The knowledge can also be obtained from student profiles through text mining.

After this, we calculate the degree of preference between a course c and the preference set **P** using weighted sum of the degrees of all preferences. The average degree of preferences between a course c and **P**, denoted as tp_c, is calculated as follows:

$$tp_c = \frac{\sum_{i=1}^{4} w_i \times m(p_i, c)}{\sum_{i=1}^{4} w_i} \qquad (2)$$

Here w_i is the weight of preference type P_i, defined in $\mathbf{W_P}$; p_i is a preference item with preference type P_i; $m(\ .\ )$ is the function defined in (1).

For example, assuming that the student has selected p_1 = JO02 (A PhD program leading to teaching), p_2 = CT03 (Decision Making), p_3 = SP01 (System Development), and p_4 = Project; we further assume that $w_1 = 0.9$, $w_2 = 0.7$, $w_3 = 0.2$, $w_4 = 0.8$. To know the average degree of preference value for course c = COMP667 (Intelligent Software Agents) and P, we retrieve the preference-course mapping knowledge: $m(p_1, c) = 2.0$, $m(p_2, c) = 1.9$, $m(p_3, c) = 1.0$, $m(p_4, c) = .8$, then we calculate: $tp_{COMP667}$ = (.9×2.+.7×1.9+.2×1.+.8 ×.8)/(.9+.7+.2+.8) = 1.53.

4.3 Course Combination

A course combination is a set of program courses that are available and the student is qualified to take. For each semester a student may have a number of possible course combinations.

For a student s and a given semester t, a course combination is denoted by $B(s, t)$. The aggregate of all course combinations is denoted as a set of course combinations $\mathbf{B}(s, t) = \{B_1(s, t), B_2(s, t), \ldots, B_l(s, t)\}$, where $B_i(s, t)$ is the ith course combination, and l is the total number of all different course combinations in $\mathbf{B}(s, t)$. If there are v courses that are available and ready-to-take, then $l = C_v^u$, where u is the number of courses per semester the student would like to take.

Given a student s and a semester t to be planned, we need to select the most suitable course combination from $\boldsymbol{B}(s, t)$ based on the preferences of the student, we need to sort $\boldsymbol{B}(s, t)$ by preference. To this end, we introduce the concept of *Degree of Preference* of a course combination, denoted as DoP. The DoP of the i^{th} course combination $B_i(s, t)$ in course combination $\mathbf{B(s, t)}$ is denoted as $b_i(s, t)$ and calculated as:

$$b_i(s,t) = \frac{1}{N_i}\sum_{c\in B_i(s,t)} tp_c \qquad (i = 1,2,\ldots,m) \qquad (3)$$

where N_i is the number of courses in $B_i(s, t)$, c is a course belonging to the course combination B_i(s, t) and tp_c is defined in (2).

After calculating $\{b_i(s, t)\}$, we sort the course combination set $\mathbf{B}(s, t)$ by $\{b_i(s, t)\}$. We call the new set the sorted course combination vector, denoted as $\overline{B}$ (s, t). The course combinations are in descending order of DoP so that the course combination with the largest value of DoP is located in the first position of the set.

4.4 Search Tree

Course combinations relate to a given semester but plans need to be generated across multiple semesters to be useful to the student. Search trees allow for the generation of multiple plans that incorporate course combinations across multiple semesters. A Search Tree for student s is a tree type data structure consisting of a set of nodes, each of which corresponds to a course combination as defined above. Each Search Tree starts with a root node which corresponds to the current semester. From the current node, each potential course combination in the next semester generates a child node in the next layer. So, m course combinations from a node will generate m child nodes in the next layer. The depth of the Search Tree is d +1, where d is the maximum number of possible remaining semesters to be planned and can be estimated using the regulation rules. The nodes in the ith layer of a Search Tree are all the possible course combinations in the (i-1)th remaining semester to be planned. Figure 4 illustrates the Search Tree.

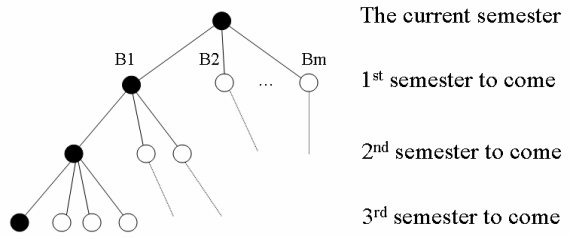

Figure 4: Search Tree.

4. 5 The planning algorithm

To generate feasible plans, the agent travels the Search Tree depth-first. A path from the root to a node in which the total credit requirements are met is a feasible schedule and is represented by a set of course combinations with their $\{b_i(s, t)\}$ as shown below:

$Path = \{(B^{(1)}, b^{(1)}(s, t)), (B^{(2)}, b^{(2)}(s, t)), \ldots, (B^{(r)}, b^{(r)}(s, t))\}$.

Here $B^{(i)}$ is the course combination selected for the i-th remaining semester; $b^{(i)}$ (s, t) is the DoP for $B^{(i)}$; r is the number of remaining semesters for the student.

Any path from the root to a node where the total credit requirements are met is a feasible plan, the first path through being the optimal path in terms of the preference.

Due to the difference in preferences of Student and Advisor Agents such as course availability and course prerequisites, it is very likely that the Advisor Agent cannot generate the students' most desired plan. Under these circumstances the Advisor Agent will request the Student Agent to relax the credit constraints or span-time until a feasible plan can be generated.

The planning algorithm is summarized as follows:

Step 1: Read the student ***s***'s preferences ***P***, Requirements, ***R***, and student profile ***F***, and initialize the curriculum Petri Net, ***PN***, for the student;

Step 2: Starting from the current semester ***t***, construct and travel the Search Tree ***T*** depth-first,

2.1. Given a semester, travel through ***PN*** to find the "ready-to-take" course set ***C1*** ⊆ ***PN***;

2.2. Get available course set ***C2*** of the next semester;

2.3. For all the possible Course Combinations ***B*** ⊆ ***C1, C2***; and calculate their degrees of preference-matching to the preferences ***DOP***(***B***, ***P***);

2.4. Sort ***B*** by **DOP**(***B***, ***P***) and construct the child nodes of the current node;

2.5. Select the most-left and unvisited node in the next layer; update ***PN***,

2.6. If "credit requirements are met", record the newly-generated plan and calculate the overall optimality value. If not, go to **Step 2.1** for the next semester.

Step 3: If no feasible plan generated, go to Step 4; otherwise, sort the plans generated by the overall optimality values ***f***(***s***, ***t***) in descending order.

Step 4: Estimate the graduation semester ***yr*** and the credit numbers for each remaining semester {cr_i} and negotiate with the Student Agent.

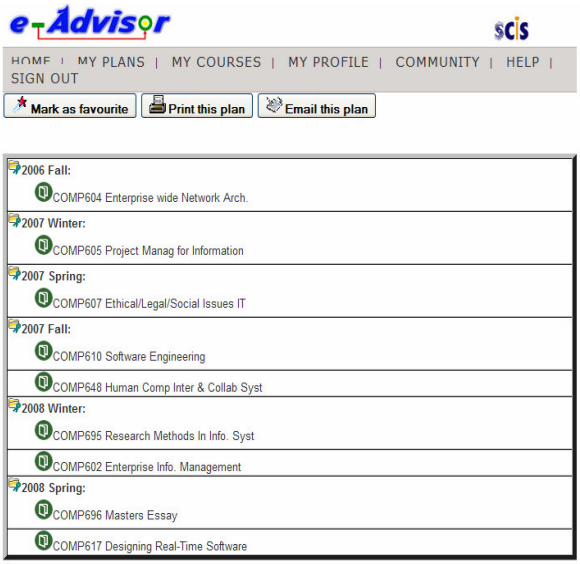

Figure 5: An example of plans generated by a Student Agent.

Figure 5 shows an example of study plans generated for a student.

5. Implementation

E-advisor is implemented on the multi-agent development platform JADE (http://jade.tilab.com/) and runs on two Agent Platforms running Linux operating system. A series of services are situated on the first Agent Platform, including a data service (PostgreSQL), knowledge service (Ontology implemented in Protégé-OWL (http://protege.stanford.edu/), curriculum modeling using Petri Net Kernel (http://www2.informatik.hu-berlin.de/top/pnk/)) and Web service (Tomcat). The user agents run primarily on the second JADE Agent Platform (see Figure 6). The experimental version of the system was completed in year 2005 and students of Athabasca University's MSc IS program have been using it in trials since that time.

Several databases and knowledge bases are included to support all agents. They include the Student Models DB, Courses DB, and Planning Knowledge Base. The database schemas in e-Advisor are derived from the ontologies by adding data type information and translating the knowledge representation formalism into the relational database management paradigm.

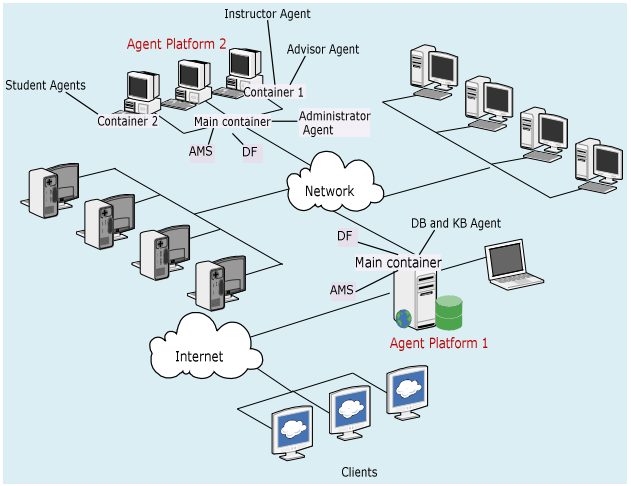

Figure 6: Agent deployment platforms of e-Advisor.

6. Empirical Results

To explore the effectiveness of e-Advisor we conducted an online survey study.

The study was an explorative study that aimed to discover how effective e-Advisor is in the actual application. We further divided the concept of effectiveness into three dimensions: (1) Are the plans generated by e-Advisor realistic and useful? (2) Does the interface of e-Advisor usable? (3) What is the overall user experience?

A total of 15 closed end questions on a 5-point scale were designed to measure the dimensions (see Appendix A). In particular, the following elements were measured:

- Effectiveness: correctness and flexibility of plans.
- Usability: visual appeal of the user interface, comprehensiveness of instructions and easiness of system navigation.
- Overall experience: system stability and availability, and general view of the system.

6.1 Study Method, Population and Sample

As the objective of the survey was not to test a hypothesis but to explore the effectiveness of e-Advisor, all existing MSc IS students were invited to participate in the survey. Consequently the study focused on the effectiveness of

Student Agent and the Student Interface only. However, the final participation rate was more than 60%. To avoid perceived bias from the respondents on the researchers (two of the authors are also instructors in the MSc IS program), a research assistant was hired to invite participants, manage the questionnaires, collect data and report the summary results. Since the researchers themselves cannot identify individual participants, we expect the results to reflect the true opinion of the students. Figure 7 summarizes the results of the survey.

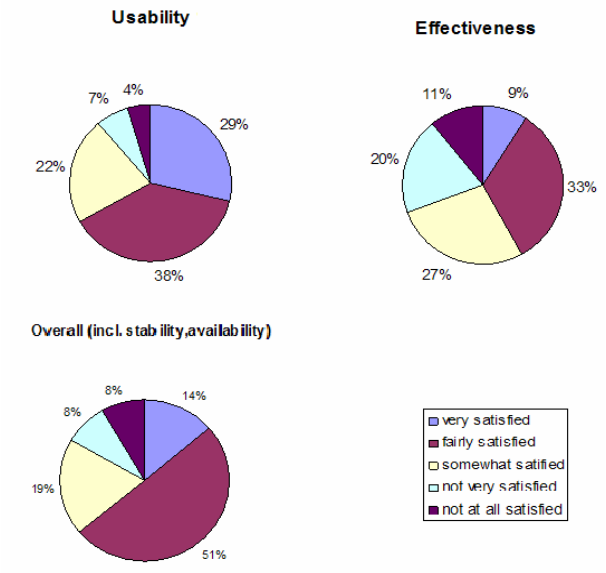

Figure 7: e-Advisor Usability Survey Results.

The results seem to be positive and encouraging; a majority of the participants considered e-Advisor helpful. Approximately 60% of the respondents considered e-Advisor effective and usable. More than 50% of the respondents reported a positive user experience with the system.

7. Conclusions

In this paper we proposed a multiple intelligent agents approach and ontology driven methodology to tackle a dynamic and complex individualized study planning problem. We developed a system called e-Advisor and tested with students registered with the MSc IS program offered by the School of Computing and Information System of Athabasca University. Although there is room for improvement, the results are positive and encouraging.

We will continue to work on empirical studies on other users to test of utility functions of agents and the scalability of the system. We will enhance the system, in particular the ontology maintenance and the agent negotiation algorithm, and improve inter-agent collaboration. Further analysis of data accumulated in planning and knowledge maintenance processes can be done to assist in determining student profiles, communications, and decision-making behavior. This will make our intelligent agents more adaptive and responsive, and therefore we can expect better services from the system.

Acknowledgements

We thank the National Science and Engineering of Research Council of Canada (NSERC) for sponsoring this project and Xiaofeng Liang, Benjamin Blanchard, and Qin Li for their programming work for the project.

References

1. Bykat, A., *Design of a Knowledge-Intensive Consultant: SAIKIC.* Journal of Artificial Intelligence in Education, 1997. **8**(1): p. 3-20.
2. Younis, N., R. Salman, *The Shifting Sands of Advising.* The Mentor, 2006. **8**(2).
3. Turban, E., J. C. Fisher, S. Altman, *Decision Support Systems in Academic Administration.* The Journal of Educational Administration, 1988. **26**(1): p. 97-113.
4. Murray, W.S., L. Le Blanc. *A Decision Support System for Academic Advising*. in *1995 ACM Symposium on Applied Computing*. 1995. Nashville, TN, USA.
5. Carroll, J., G. Chappell *An Intelligent Universal Advisor*. in *The 1996 ACM symposium on Applied Computing*. 1996. Philadelphia, Pennsylvania, United States: ACM Press New York, NY, USA.
6. Hamdi, M.S., *MASACAD: A Multiagent-Based Approach to Information Customization.* IEEE Intelligent Systems, 2006. JANUARY/FEBRUARY 2006: p. 60-67.
7. Paquette, G., P. Tchounikine. *Towards a knowledge engineering method for the construction of advisor systems*. in *the International Conference on AI in Education*. 1999. Le Mans, France.
8. Kay, J. *Ontologies for reusable and scrutable student model*. in *In Proceedings of AIED99 Workshop on Ontologies for Intelligent Educational Systems*. 1999.
9. Chen, W., R. Mizoguchi. *Communication Content Ontology for Learner Model Agent in Multi-Agent Architecture*. in *AIED99 Workshop on Ontologies for Intelligent Educational Systems*. 1999.
10. Mizoguchi, R., J. Bourdeau, *Using ontological engineering to overcome AI-ED problems.* International Jounal of Engineering Education, 2000. **11**(2): p. 107-121.
11. Razmerita, L., A. Angehrn, A. Maedche. *Ontology based user modeling for Knowledge Management Systems*. in *the User Modeling Conference*. 2003. Pittsburgh, USA.
12. Zhou, G., J. T. L. Wang, P. A. Ng, *Curriculum Knowledge Representation and Manipulation in Knowledge-Based Tutoring Systems.* IEEE Trans. on Knowledge and Data Engineering, 1996. **8**(5): p. 679-689.
13. Hoffman, R.R., P. J. Hayes, K. M. Ford, *Human-Centered Computing: Thinking In and Out of the Box.* IEEE Intelligent Systems, 2001. **16**(5): p. 76-78.
14. Greer, J., G. McCalla, J. Vassileva, R. Deters, S. Bull, L. Kettel *Lessons Learned in Deploying a Multi-Agent Learning Support System: The I-Help Experience*. in *International AI and Education Conference AIED'2001*. 2001. San Antonio, USA: IOS Press, Amsterdam.
15. Genesereth, M.R.K., Steven P, *Software Agents.* Communications of the ACM, 1994. **37**(7): p. 48-53.
16. Sycara, K., K. Decker, A. Pannu, M. Williamson, and D. Zeng, *Distributed Intelligent Agents.* IEEE Expert 1996(Dec.): p. 36-45.
17. Jennings, N.R., K. Sycara, M. Wooldridge, *A Roadmap of Agent Research and Development.* Journal of

Autonomous Agents and Multi-Agent Systems, 1998. **1**(1): p. 3-78.
18. Webber, C.L.B., S. Pesty, N. Balacheff. *The Baghera project: a multi-agent architecture for human learning*. in *Workshop - Multi-Agent Architectures for Distributed Learning Environments, International Conference on AI and Education*. 2001. San Antonio, Texas.
19. Jafari, A., *Conceptualizing intelligent agents for teaching and learning.* Educause Quarterly, 2002: p. 28-34.
20. Murata, T. *Petri Nets: Properties, Analysis and Applications* in *Proceedings of the IEEE*. 1989.
21. Karampiperis, P., D. Sampson. *Designing Learning Services for Open Learning Systems Utilizing IMS Learning Design*. in *the 4th IASTED International Conference on Web-Based Education (WBE 2005)*. 2005. Grindelwald, Switzerland: ACTA Press.

Appendix A e-Advisor Evaluation Form

Please help us to develop a better e-Advisor by sharing your experience! **Questionnaire**

Please rate the following statements:

(1- Strongly disagree; 2 - disagree; 3 - neutral; 4 - agree; 5 - Strongly agree)

1. I can use e-Advisor without referring to any written instruction.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

2. The layout, fonts and colors of e-Advisor look good.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

3. The default plan generated by e-Advisor is exactly what I am planning.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

4. I am happy with e-Advisor.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

5. There are too many plans generated by e-Advisor.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

6. I go to the e-Advisor website every week.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

7. I will follow the plans generated by e-Advisor with no or minor modification.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

8. I usually finish using e-Advisor in 30 minutes.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

9. I cannot find my intended plan in all of the plans generated by e-Advisor.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

10. Sometimes e-Advisor is not available when I want to use it.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

11. I can always undo any mistakes I entered in e-Advisor.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

12. e-Advisor is easy to use.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

13. There is too much information provided by e-Advisor.

◉ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

14. e-Advisor is useful to my study.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

15. I usually refer to one or more of the plans generated by e-Advisor.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5