Intelligence Student Advising System - An Implementation using Object-Oriented C++

Dyg. Norhayati bte. Abg. Jawawi

Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Locked Bag 791, 80990 Johor Bahru, Johor

Abstract

This paper present an approach for developing a consistent student course-advising system for undergraduate students using knowledge-based technology. A prototype system has been implemented in object-oriented technique using C++. The prototype system was designed for undergraduate Computing students. The prototype is able to give consultation and advice on some important aspect of student advising problems. Knowledgeable behaviour was produced where the 'expert' and 'knowledge' is stored separately from the inference engine. Object-oriented programming technique was found to enhance the development of the system.

Keywords : Intelligence Advising system, knowledge based system and object-oriented programming

1.0 Introduction

Most academic institutions offers their students a variety of programs from which a student can select one and satisfy the school requirements in order to be eligible for a degree. This usually consists of choosing the required courses within the student's major and minor, elective courses, and satisfying general university requirements. Typically a student has a certain amount of freedom to choose the right course and subjects which suit them.

The problem arise when the students are not sure which course to choose, in order to complete her/his degree successfully. This is because the student are not aware of the degree requirements, course descriptions and prerequisites. Advising is a very important component in retention of students, since good advising detects problems in early stages and prevents problems that would otherwise happen at the later stage.

The advisory process must address two areas: degree check requirements and the student's special interests. The role of the advisor is to suggest the offered courses when the student is likely to pass the subjects in the course taking into consideration the prerequisite requirements and minimizing the time to get a degree. Mature students can specify their interest at the advanced level and take courses from the set of elective courses.

The advising system often suffers from a number of shortcomings such as:

i. it creates a heavy burden on the academic staffs;
ii. the requirements often change, consequently it is difficult for the advisors to update their own knowledge with sufficient frequency;
iii. the existing set of manual or guidance is often incomplete or inconsistent, which may lead to misinterpretation of information;
iv. the overall process of advising is time consuming and complex ;
v. transfer student has different academic background and knowledge;

The layout of this paper is as follows. In Section 2 previous works related to student advising system are reviewed. Section 3 describes the advantage of applying knowledge-based technology to solve student advising problems. Section 4 describes an implementation and testing of a student course advising system using object-oriented

C++ for undergraduate students at the School of Computing, Sheffield Hallam University (SCAS).

2.0 Previous Works Related to Student Advising System

In recent years, several automated advising systems have been developed. These systems have been developed for both undergraduate and post-graduate education and have a wide variety of functionality. These systems have been developed by utilizing various technologies including artificial intelligence, spreadsheets, as well as traditional programming languages. A summary of these systems and their functionality is illustrated in Table 2.1 and Table 2.2. From these works, it was found that the student advising domain is amenable to the knowledge-based system method.

	Graduate Course Advisor - (Valtoria, 1984)	A Student Advisor (Golumbic, 1986)	A Course Advisor (Crooke, 1987)
Academic Institution	Duke Univ. & Univ. of Carolina	Bar Ilan Univ.	Texas Christian Univ.
Department	Computer Science	-	Computer Science
Student Status	Post-graduate	Undergraduate	Undergraduate
Language used	C-Prolog	Prolog	C-Prolog
Main Functions	Number of courses to enroll Type of courses to enroll "Best courses" for student Schedule of appropriate length	Check degree requirement Suggest courses based on course interest and Univ. requirement	Recommend possible major areas of study to undecided majors within the university
Checks Required Course?	No	Yes	No
Suggests Courses to Take	Yes	Yes	N/A

A Student Advisor (Frank, 1988)	Graduate Student Advisor (Chan, 1988)	Schedule Advisement System (Kawalski, 1991)	A Course Advisor - (Occena, 1993)
Univ. of Arkansas	Arizona State Univ.	California State Univ.	Univ. of Massouri-Columbia
Computer Sciences	Industrial Eng.	Computer Science	Industrial Eng.
Undergraduate	Post-graduate	Undergraduate	Undergraduate
C-Prolog	Personal Consultant	PC Scheme	Expert System Env. Shell
Check degree requirement Suggest on the best courses for student based on interest and progress	Requirement for degree Option for major fields of study Available courses and faculty	Check degree requirements Suggest one term schedule based on student's availability	Check degree requirement Suggest on the best courses for student based on interest and progress
Yes	N/A	N/A	Yes
Yes	No	Yes	Yes

Table 2.1 : Published student advisor systems with AI technology

In this paper a student course advising system using knowledge technique is presented. The key feature of the developed system is the separation of 'expert' or 'knowledge' from the inference engine and the reasoning strategy. This paper will demonstrate how C++ and character of object oriented programming (OOP) can help in development of such system.

The Advisor's Assistant - (Batchelder, 1989)	Student Advising (Malasri, 1988)	Semi-Auto Advising System (Malasri, 1990)	Student Advising system (Billo, 1993)
South Dakota School of Mines & Technology	Univ. Miami, Coral Gables	Christian Brother College	Univ. of Pittsburgh
Electrical Eng.	Civil & Architecture Eng.	Civil Eng.	Industrial Eng.
Undergraduate	Undergraduate	Undergraduate	Undergraduate
Turbo Pascal V5.0	Lotus 1-2-3	QUATTRO	Quick Basic
List student's courses completed and credits detail Check and list all courses to be completed before graduation	Check student credits and GPA Check transfer and prerequisite courses list all core courses which the student can take	List all required courses Keep all student records Display student progress Recommend what courses the student can take for one semester Suggest trial schedule	Develop student long-term schedule
Yes	Yes	Yes	No
All Remaining Courses	All Remaining Core	N/A	No

Table 2.2 : Published student advisor system with other technology

3.0 Intelligence Knowledge-based System

The methods of knowledge-based systems are now being applied in many diverse areas. This technology is useful for tasks where the knowledge of highly skilled human experts can be represented explicitly and acted upon by inference procedures. The result is intended to be a system which exhibits intelligent, knowledgeable behavior (Golumbic, 1986).

For the student advising problem, the knowledge-based system is preferred over a conventional algorithmic system for several reasons :

I. It was desired to manage knowledge and not data. The is system required to advise undergraduate students based on some domain expert's specific rules and heuristics.

II. The system need to contain explanation facilities available to other faculty members, thereby to understand the university advising process for undergraduate computing students.

III. The system needed sufficient flexibility to be easily maintained and modified to conform more to a particular discipline.

IV. A thorough analysis and understanding of the undergraduate advising process that would come about as a by product of the knowledge-based system development.

V. The permanent storage of course-advising information.

4.0 Implementation of Student Course Advising System (SCAS)

There are three groups of peoples involved in the student course advising problem: student advisor, student and administration. Figure 4.1 shows the relationship between the groups.

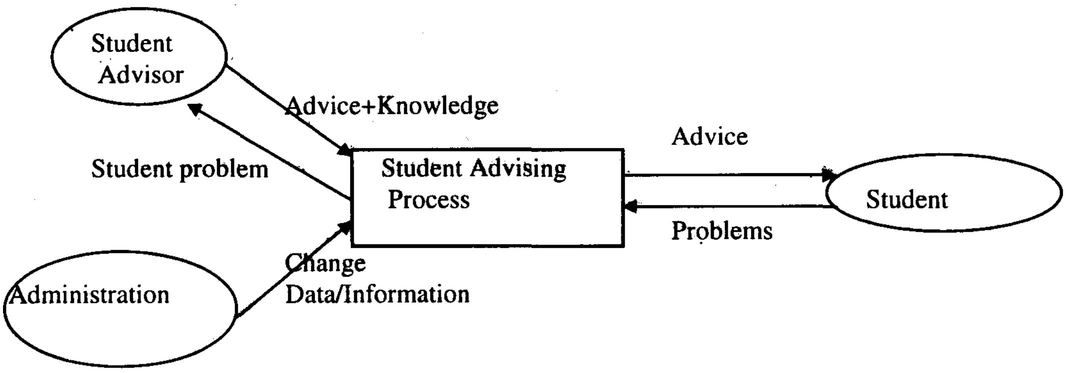

Figure 4.1 : The relationship between groups involved in student advising process

There are three main problem areas that need to be solved in SCAS. These areas are:

1) Advise student to select the right course. This involves first year student, second year direct entry student and final year direct entry student.

2) Advise on to change course for student already began their course but decided to change their course. It also involves providing information to students who have problem in particular units
3) Advise to select the electives subjects for final year student.

4.1 The Design of SCAS

The design of SCAS involves two main areas: the knowledge design and the inference engine design. The knowledge obtained through knowledge acquisition process can be categorized into three main groups:

1) Course knowledge i.e. information related to the course which includes subjects, course and unit prerequisites.
2) Evaluation knowledge i.e. knowledge related to evaluation process which includes questions asked by the advisor to identify the student ability, the weighting factors and reasoning statements.
3) Rule knowledge i.e. heuristics and decision rules.

For the purpose of flexibility and maintainability these three knowledge are stored in three difference data files. Any changes in the information and knowledge will not affect other components of the SCAS, especially the inference engine. Changes can be made in these data files and therefore avoiding recompiling the whole system when there are changes in the knowledge. Figure 4.2 shows how the above knowledge fit with the other components in the SCAS.

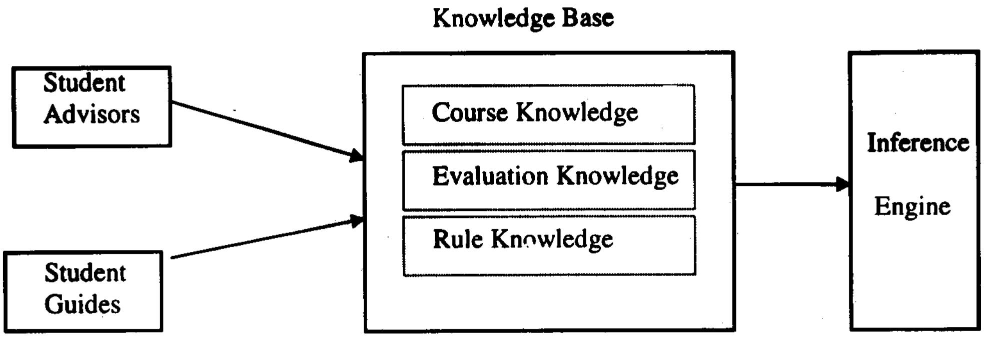

Figure 4.2 : Course, evaluation and rule knowledge as components of the SCAS

Course and evaluation knowledge represented in frame, refer table 4.1 and 4.2. Rule knowledge represented in procedural rules, example of the rules are:

i. IF the student overall performance is suitable for software engineering course THEN check for software engineering detail subject

ii. IF the student have no problem with all detail software engineering subject THEN the student really suitable for the course.

Refer to figure 4.3 on how this rules used in SCAS.

Slot	Fillers
Unit/subject	205
Title	Software Tools and Evironment
Prerequisite	103
Course_taken	SE, IT, EI

Table 4.1 : Unit Frame for Course Information

Slot	Fillers
Question_number	103_1
Question	Do you have basic understanding of software principle?
Weight_Factor	5
Reason	Basic understanding of software engineering principles

Table 4.2 : Evaluation skill represented in frame

rule_SE0 : if	rule_SE01 : if
level=2 and	103=60 and
sl = 50	105 = 50
then	then
check = rule_SE01	check = done

Figure 4.3 : Rules used in SCAS

The ability of advisor in evaluating a student is presented in evaluation knowledge. Within a unit/subject, certain attributes are more important than others in determining whether a student understand a particular unit/subject or not. Weighting factor is used to give indication or measurement associated with the attributes. The higher the weighting factor the more important is the attribute.

Consider an example shown below for a subject named 'Programming'. This subject has four attributes, each attribute has a weighting factor represented by a number in the bracket.

I. Unit 103 - Programming

 A. Understand software engineering basic principles. (10)

 B. Familiar with general idea about programming methods. (5)

 C. Familiar with general idea about programming style. (5)

 D. Know how to design program. (3)

Considering the unit named 'Programming', the attribute A 'Understand software engineering basic principles' is more important than other attributes B - D in determining whether a student understand the programming subject. The higher the weighting factor, the greater is the importance. The weighting factor figures were determined by the student advisor.

When an answer to a question is yes, the fact associated with the question also become yes. In order to get a total score for each student , the sum of those weighting factors initiated to yes is calculated. The score is called the *total weighting factor*. If all the facts are initiated to yes, the total weighting factor is called *the maximum sum weighting factor*. Therefore the suitability level for each student can be calculated using the equation

Suitability level = Total weighting factor / total maximum sum weighting factor

4.1.1 Inference Engine Design

The inference engine was designed using object-oriented approach.

Object-Oriented design

Using the object-oriented design approach as specified by (Booch, 1991), the major steps involves are defines as:

Step 1: Identify the object and their attributes

From the problem domain a number of class can be identified:

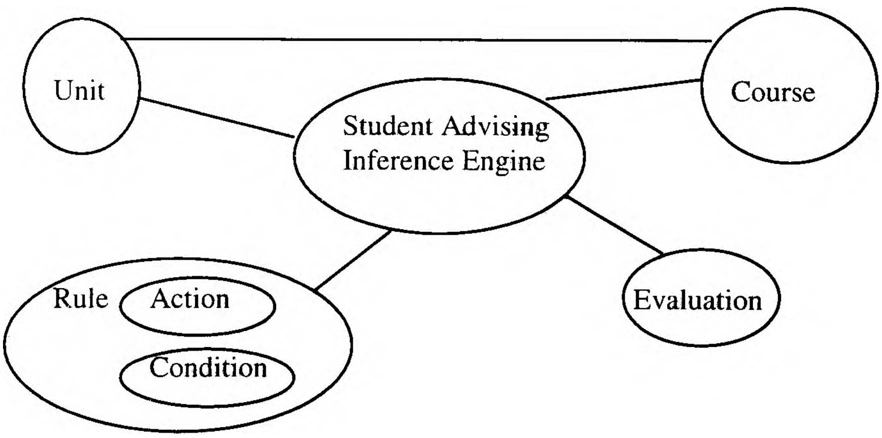

Figure 4.4 : Objects and attributes

From Figure 4.4 of SACS, five distinct objects can be found.

1) Student Advising Inference Engine (SAIE). This provide service for building the advise process, reasoning process and user interface display.
2) Unit. This gives information about each unit/subject in the programme.
3) Course. This gives information about all courses in the programme.
4) Evaluation. Representing evaluation knowledge.
5) Rule. Representing advice knowledge.

Step 2: Identify the operations suffered by and required of each object

This stage is concerned with identifying the operation that may be performed on an object as listed in Table 4.3.

Object	Operation
SAIE	Get course and unit relevant for UNIT and COURSE object, Get the right evaluation object from EVALUATION, Calculate suitability level, Get the right rule from RULE object, Select the right reason from EVALUATION, Get fact or input from student and display reason and other display facility
Unit	Insert all unit information into working memory, search for specific unit, display specific unit
Course	Read unit and course data, compare course involve in UNIT object with existing course offered
Evaluation	Insert all object frame into working memory, search for specific object, insert user/student answer for each question,
Rule	Insert rule (include rule condition and action) into working memory, search for specific rule

Table 4.3: Operation performed on objects

Step 3 : Establish the visibility of each object in relation of other object

From Figure 4.15, it is clear that object SAIE establishing relation to other objects in the system. Unit object need to refer to course object to identify whether a course is a valid course in the system.

Step 4 : Establish the interface to each object: this step involves writing a structure for the object

It can be seen at this point that there is one major object in the system which is the inference engine. This object will use all other objects as part of it component, but SAIE object cannot change any value in other object (can only use or read the object). This provides both data hiding and encapsulation of the whole system.

Step 5 : Implementation of each object

The implementation step involves the use of a pseudo-code approach to design the respective procedure for each object.

4.2 Implementation of Knowledge-base and Inference Engine Subsystem

The artificial intelligence (AI) programming language such as prolog provide some building facilities for building inference engine, therefore developing an inference engine using this type of language is trivial. Writing inference engine in C++ requires more afford.

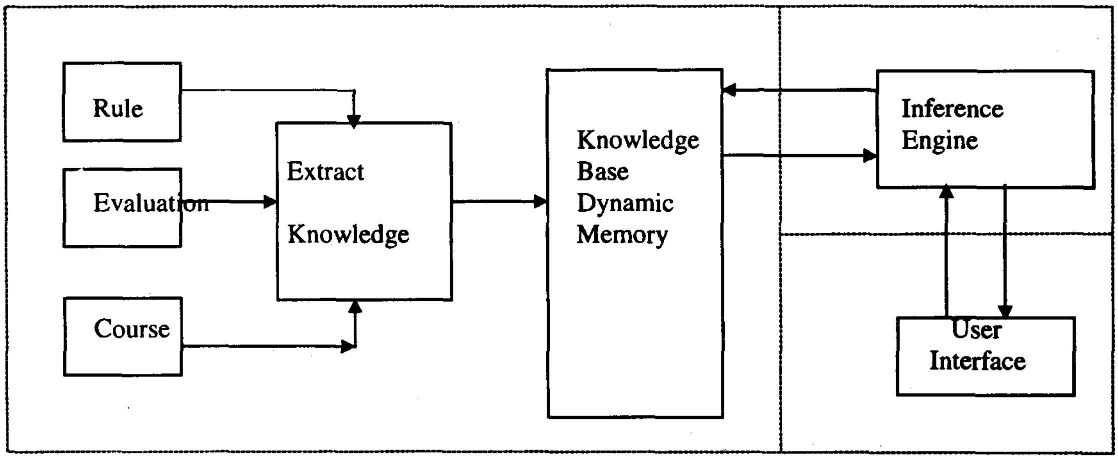

Figure 4.5: Subsystems and their relation in the SCAS

As can be seen from Figure 4.5, there are five blocks in this subsystem. The blocks *'Rule'*, *'Evaluation'*, and *'Course'* are configuration files which store the information, rules and facts about courses and subjects in School of Computing. This three files are implemented as plain ASCII files to store the information, facts and rules. Ideally, these files should be protected to avoid unauthorised persons from tempering with the information in the files. It should only allows the administration or the advisor to change the files.

The task of the block named *'Extract Knowledge'* is to read the three configuration files , extract and store the information in the appropriate knowledge-base dynamic memory. This feature make the system very flexible, where any changes in the knowledge block does not need any changes of system or recompilation of the whole inference engine.

In this system the two main tasks of inference engine are giving requested information about the courses to students, and advise student on selecting suitable course and on changing to another course. In the advising tasks the major steps performed by the inference engine are:

1) Ask questions to get information from student
2) From the answers calculate student suitability level
3) From the suitability level check rules
4) Derive conclusion from rules
5) Give explanation/reasoning for the conclusion

The inference engine is implemented using forward chaning method and object-oriented programming technique in C++ is used to write the implementation of the inference engine. The knowledge and the information in the application environment of the student advising system, can be easily represented as *object, attribute and value.* The used of object-oriented language in this problem can make the representation of objects more efficient. Object-oriented language like C++ has features built into the language to support objects. An instances of class is a data object that can be manipulated. The code fragments shown in Figure 4.6 gives the definition of the class for the unit data structure as was implemented in C++. The evaluation information and rules data structures are defined similarly to the code fragments in Figure 4.6.

There are various other object-oriented programming languages such as Prolog, Smalltalk, Object Pascal, Common Lisp Object system(CLOS) and Ada. However, what distinguishes C++ from many other languages is the ability to define new data types in a way that their use cannot be distinguished from the built-in types (Hekmatpour 1990). Borland C++ was chosen for the development of the SCAS because it offers a good development environment and it is widely available and easily accessible at the School of Computing.

```
typedef struct UnitCourse
{
  int unit;                //unit code
  char unittitle[SHORTSTR];//unit full title
  int nocourseinv;//number course involve might be not necessery
  int prerequisite[MAXPRESUB]; //prerequisite for a particular unit
  int courseinv[MAXCOURSE]; //all course involve for a
                             particular unit
}UnitStruct;

struct UnitFrame
{
    UnitStruct unitinfo;
    struct UnitFrame *next;   //point to the next node
};

typedef struct UnitFrame* UnitFramePtr;

class Unit
{
public:
  Unit();
  void sls_insert(UnitStruct ui); //insert a node to the list
  void display();                //Display all info/node
  void dispaunit(UnitFramePtr aunit); //display one node
  UnitFramePtr search(int unitcode);  //search for particular unit
  int scourse(int coursecode,int* arrcourse); //search all unit
                                                given a course
  void testfile(ofstream& out_stream);   //display info to file

private:
  UnitFramePtr start,last;
};
```

Figure 4.6: Code fragments showing the class definition of unit information.

4.3 System Testing SCAS

The testing process is intended to validate the working of the SCAS and to verify the SCAS meets its specification. The main goals of the system testing are :

1) To verify the information and knowledge to be supplied to system are correct according to the Student Guideline and expert/advisor knowledge. This process done by human expert.
2) To check that the system read the knowledge, rules and facts in the configuration files correctly and any changes in the configuration files will be registered by the system. Figure 4.7 show example of correct facts print by the system.
3) To validate the advising functions of the system according to the SCAS functional specification. Refer to example Test 3.

```
List of units for Software Engineering Unit for Year 1 :
101: SYSTEM
102: BUSINESS ORGANISATION, PLANNING AND CHANGE
103: PROGRAMMING
104: TECHNOLOGY
105: BASIC MATHEMATICS
108: HUMAN COMMUNICATIONS AND PSQ A
Unit for Year 2 :
204: DATABASES
205: SOFTWARE TOOLS AND ENVIRONMENTS
206: SOFTWARE ENGINEERING DESIGN
207: SOFTWARE ENGINEERING METHOD A
208: MATHEMATICS AND COMPUTING
Unit for Year 3 :
305: SOFTWARE ENGINEERING MANAGEMENT
306: HUMAN COMPUTER INTERACTION AND KNOWLEDGE BASED SYSTEMS
310: PROCESS MANAGEMENT

Select the unit code for more information
or type p(unit code) for list of prerequisites unit e.g.<p201>:
```

Figure 4.7 : Subject detail screen

Test 3: Test 3 was conducted to test the advising function for the case of a second year Software Engineering student who want to change her/his course to Information Technology. The test data used in this test is shown in table 4.4. The result of the consultation screen is shown in Figure 4.8. The result of the consultation is than verified by the student advisor.

Data field	Data value
Current course	Software Engineering
New course	Information Technology
Student level	Second year
Problems in subject	Software Engineering Design Software Engineering Method A
Academic background	1. Software Engineering 2. Programming 3. Discrete Mathematics 4. Introduction to computer system

Table 4.4 : Test data for use in test 3 in validating advising function

```
      ADVISE SYSTEM - Reasons for system conclusion
      *********************************************

      Conclusion : Not suitable for course - Information Technology
Reasons:
You score 19.84127% and the minimum score required is 50%
You have problem with 210      COMMUNICATION ENGINEERING II
      1. Understand the use of digital communication in local and wide
         area networks to quantify the description and processing of signals?
      2. Ability to describe the limitations imposed upon LAN and WAN
          by noise encoding and the methods of modulation.
 You have problem with 211      MICROELECTRONICS
      1. Familiar with the various microprocessor technologies.
      2. Familiar with the various microprocessor circuits types.
      3. Familiar with the various microprocessor levels of integration.
      4. Ability to design simple integrated circuit systems for amplification
          and signal processing.
      5. Ability to synthesize combinational logic.
      6. Familiar with the architecture and use of typical microprocessors.

      Press any key to continue
```

```
      ADVISE SYSTEM - Reasons for system conclusion
      *********************************************
      Conclusion : Not suitable for course - Information Technology
Reasons:
You score 19.84127% and the minimum score required is 50%
You have problem with 213      SOFTWARE ENGINEERING METHODS B
      1. Understand the stages in the software development process.
      2. Understand the relationship between the stages in the software
         development process.

Below are all subjects you need to know for new root(not in current root):
      1. 210 :          COMMUNICATION ENGINEERING II
      2. 211 :          MICROELECTRONICS
      3. 213 :          SOFTWARE ENGINEERING METHODS B

      Press any key to continue
```

Figure 4.8 : The result of consultation screen for Test 3

There is a substantial amount of code in the SCAS to be tested and effective strategy is needed to make sure it is tested thoroughly. In this system the bottom-up testing strategy was adopted across the whole stages of testing process during the development period. Incremental approach was used in the testing process. Drivers were written in order to test the unit, module and subsystems.

4.4 Sample consultation sessions with SCAS

The implemented SCAS provides the user with facilities which includes provide information related to courses and giving advice to student for selecting and changing course.

Figure 4.9 shows a screen shot of the main menu of SCAS.

```
*******************************************************************
* SHEFFIELD HALLAM UNIV. UNDERGRADUATE COMPUTING PROGRAMME ADVICE SYSTEM *
* Version 0.78 By Dayang Norhayati Abg. Jawawi 1996 *
*******************************************************************
1. Course Advice System
2. Quick Information
0. EXIT system



Select an option :
-------------------------------------------------------------------
```

Figure 4.9 : SCAS Main Menu

Option '1' offers the advising function for both new students and existing students. Option '2' lists all the subjects details of a particular courses offered.

Table 4.5 shows an example data fed to the SCAS for a second year direct entry software engineering student. Figure 4.10, shows the result of the consultation and this result has been verified to be correct by the human expert.

Data field	Data value
Course interested	Software Engineering
Student status	Second year direct entry
Academic background	1. Software Engineering 2. Programming 3. Discrete Mathematics 4. Introduction to computer system

Table 4.5 : Example data 1

```
     ADVISE SYSTEM - Reasons for system conclusion
     ***********************************************
Conclusion : Suitable for course - Software Engineering


Reason:
 You score 80% and the minimum score required is 50%
 Advise : But you still have some problem with following subjects
 You have problem with 105      BASIC MATHEMATICS
     1. Understand the nature of the binary number system.
     2. Ability to describe data using statistical measures and
     distributions.
     3. Ability to apply and use a range of statistical distributions i

        hypothesis testing.
```

Figure 4.10 : The result of consultation screen for data 1

4.0 Conclusion

In this paper the used of knowledge based system in solving student course advising problem has been described. An implementation of a Student Course Advising system (SCAS) using object-oriented C++ has been discussed. Knowledgeable behaviour was produced where the 'expert' and 'knowledge' is stored separately from the inference engine. Object-oriented programming technique was found to enhance the development of the system.

5.0 References

1) Batchelder M. J.:"The Advisor's Assistant", *Proceedings of the IEEE Frontiers in Education Conference*, 1989, 255

2) Billo R. E. and Bidanda B.: "A Student Advising System for Undergraduate Engineering Curricular Scheduling", *Computer Education*, 1994, 22(3), 205-213

3) Booch, G., *Object Oriented Design with Applications*, The Bejamin/Cummings Publishing Company, inc., 1992

4) Chan D. and Cochran J. K.: "Using Expert-system Shells for Graduate Student Advising", *Engineering Education*, 1988, 310

5) Crooke D. E., Starks S. A. and Thorp D. S., "CSAD: A Course Advisor", *ASEE Annual Conference Proceedings*, 1987, 658

6) Frank J. L., Duffield C. A. and Swearingen C. A., "Mentor-I: An Expert Database System for Student Guidance", *IEEE Expert*, 1988, 3, 40-46

7) Golumbic, M. C., Markovich M., Tsur S. and Schild U. J.: "A Knowledge-Based Expert System for Student Advising", *IEEE Transactions on Education*, 1986(5), E-29(2), 120

8) Hekmatpour S., *C++ A Guide for C Programmers*, Prentice Hall,New Jersey, 1990

9) Kawalski K. and Ealy D.: "Schedule Advisement Expert System", *Computer Education*, 1991, 17(4), 259-265

10) Malasri S.: "Student Advising Using Spreadsheet Program", *International Journal Appl. Engineering Education*, 1988, 559

11) Malasri S.: "Semi-Automated Student Advising System Using Quattro Spreadsheet Software", *Computer Education*, 1990, 14(4), 317-324

12) Occena L. G. and Miller S. L.: "IEADVISE - An Undergraduate Course-Advising Expert System in Industrial Engineering", *Expert Systems*, 1993(8), 10(3), 139

13) Valtoria M., Smith B. and Loveland D.: "The Graduate Course Advisor: A Multi-Phase Rule-Based Expert System", *Proceeding of the IEEE Workshop on Principles of Knowledge-Based System*, 1984, 53