
*Note: This is the final draft version of this paper. Please cite this paper (or this final draft) as above. You can download this final draft from the following websites:*

http://research.sabanciuniv.edu

http://ertekprojects.com/gurdal-ertek-publications/
Rule-based expert systems for supporting university students

Gökhan Engin\textsuperscript{a}, Burak Aksoyer\textsuperscript{b}, Melike Avdagic\textsuperscript{b}, Damla Bozanli\textsuperscript{b},
Umutcan Hanay\textsuperscript{b}, Deniz Maden\textsuperscript{b}, Gurdal Ertek\textsuperscript{b}\textsuperscript{*}

\textsuperscript{a} Oracle Turkey, Istanbul, 34398, Turkey
\textsuperscript{b} Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, 34956, Turkey

Abstract

There are more than 15 million college students in the US alone. Academic advising for courses and scholarships is typically performed by human advisors, bringing an immense managerial workload to faculty members, as well as other staff at universities. This paper reports and discusses the development of two educational expert systems at a private international university. The first expert system is a course advising system which recommends courses to undergraduate students. The second system suggests scholarships to undergraduate students based on their eligibility. While there have been reported systems for course advising, the literature does not seem to contain any references to expert systems for scholarship recommendation and eligibility checking. Therefore the scholarship recommender that we developed is first of its kind. Both systems have been implemented and tested using Oracle Policy Automation (OPA) software.

© 2014 The Authors. Published by Elsevier B.V.
Selection and peer-review under responsibility of the Organizing Committee of ITQM 2014.

Keywords: rule-based systems; expert systems; recommender systems; decision support systems; higher education; university education.
1. Introduction

1.1. Rule-Based Expert Systems

Rule-based expert systems have the ability to emulate the decision making ability of human experts. They are designed to solve problems as humans do, by exploiting encoded human knowledge or expertise. This knowledge can be extracted and acquired directly through interaction with humans, as well as from printed and electronic resources such as books, magazines and websites. The extracted knowledge forms the knowledge base of the rule-based system. The other major component of rule-based systems draws conclusions from this knowledge, and is referred to as the inference engine. The conclusions and suggestions offered by the rule-based system satisfy users’ needs for expertise within the chosen domain.

Rule-based systems are designed to solve problems in a selected domain. Every domain has its own knowledge and reasoning humans, which can be emulated and even replaced through automated rule-based systems. Many domains contain a large amount of knowledge that can be captured fully only through an information system, since humans may not access or immediately retrieve fully the needed information. There are many advantages of rule-based expert systems: They decrease costs since they reduce the need for human experts; they are permanent; they can be used for different knowledge systems, which increases functionality; they increase reliability since they minimize errors that humans are prone to; and if designed by multiple experts, can increase confidence. Finally, they lack human emotions, which are sources of mistakes in human based systems. The advantages of rule-based expert systems are multifold and they can considerably facilitate human life for the better. In this paper, we present and describe two rule-based recommender systems projects, both in the domain of university education.

1.2. Oracle Policy Automation (OPA)

The software used in the reported projects is Oracle Policy Automation (OPA). OPA is capable of reading rules conveniently from spreadsheet and word-processor file formats, where the rules are written in a format akin to natural language. OPA automatically forms a
problem solving or decision making application based on the acquired rule base. In OPA, 
the rules can be written in more than 25 languages, including Turkish, the language selected 
for the expert system in our second case study. Users codify the information or knowledge 
via Microsoft Word and/or Microsoft Excel in their native (natural) language and form rules 
without having to know any programming languages (such as Java or C#) or expert system 
languages (such as CLIPS). This makes the testing and maintenance of, as well as the 
changes to the rules easier. If the rules are detailed and need frequent modifications, and/or 
if there will be auditing and the rules need to be transparent, OPA is especially suitable 
and convenient. After the program is executed and a suggestion is presented by the program, the 
program also shows why and how that suggestion is selected. This way, the rules can be 
checked and the reasons behind the suggestions can be provided to users.

Writing and running an OPA program has five stages. First one is to gather the 
information and data for the rules that will be implemented. Second step is to write the rules 
using Microsoft Excel and/or Word. Afterwards, the rule base is checked for language 
mistakes, rule mistakes and other errors that might have occurred. After the rule base is 
finalized, the users start answering the related questions to obtain a suggestion. In the first 
project, the suggestion is the list of courses that can be taken by a student, given the courses 
he has taken so far. In the second project, the suggestion is the eligibility of (a student for) a 
specific scholarship. After all the questions are answered, the rule-based expert system 
presents its suggestion. For example, in the second project, the resulting output is in the 
form “The student/user is eligible for the Sabancı Vakfı Scholarship” or “The student/user is 
not eligible for the Sabancı Vakfı Scholarship”, corresponding to the suggestions of applying 
and not applying to the Sabancı Vakfı Scholarship, respectively. The user can see the reason 
why the program gave the result it gave by clicking the underlined result. If the user is 
eligible for the scholarship, clicking the underlined hyperlinked text will show all the criteria 
satisfied by the user. If the user is not eligible for the scholarship, then the underlined 
hyperlinked text will state which criteria was not satisfied. This part of OPA is very useful for 
auditing reasons because it clearly states the reasons for the results. In our application this 
is highly relevant because the school systems, entrance exams and the rules for the eligibility 
change frequently.
There are more than 6000 colleges and more than 15 million college students in the US alone, and academic advising is typically performed by human advisors. Engaging with students is becoming more complex and student-demand driven, bringing an immense managerial workload to faculty members, as well as other staff at universities. There is also an information overload on the student side, as the information that needs to be interpreted, clarified, and captured is showing an ever increasing trend. Expert systems developed for university education can serve a variety of goals, including advising, tutoring, instruction, and evaluation. We will discuss the studies in literature under two groups, namely expert systems for course advising and expert systems for other education-related goals. Within the first group we will also discuss course advising systems that are built using methodologies other than expert systems.

Dunstan, Nambiar and Dutta, Laghari & Khuwaja, and Al Ahmar report expert systems for advising undergraduate students on course selection. Al-Ghamdi et al. describe a system for advising postgraduate students and presents a detailed review of expert systems for course selection. Khanna et al. also provide an extensive review, besides listing the main areas where expert systems can be used in education. Onyeka et al. propose combining rule-based reasoning with case based reasoning in course advising expert systems. Goodarzi and Rafe incorporate fuzzy reasoning into its advisory expert system. Koutrika et al. develop a flexible system for generating advisory expert systems. Parameswaran et al. present CourseRank, a course recommendation and planning system developed at Stanford University. The authors report that more than 170 universities throughout the world have adopted CourseRank, making the software in the most widely used course advising system in the world.

Besides expert systems for course advising, the literature contains a multitude of studies that describe course advising systems built using other methodologies. Studies of Vialardi et al., Taha, Cho and Kang, and Park et al. are examples where course recommendation is based on data mining. Among these three studies, Vialardi et al. reports using real world data from a Spanish university; Taha uses collaborative filtering; Cho and Kang uses hybrid filtering. Sobecki uses nature inspired methods, again for course recommendation.
Besides course recommendation, expert systems have been developed for other purposes, as well. Buche and Querrec\textsuperscript{18} describe an expert system for helping human learners in virtual reality environments. Hwang \textit{et al.}\textsuperscript{19} introduce an expert system that works as instructor for improving problem solving skills of students. Jaryani \textit{et al.}\textsuperscript{20} present an expert system that provides customized learning for each student based on his/her background and realities. The fuzzy expert system in the work of Van Hecke\textsuperscript{21} identifies the personal profile of the students as one of the three types. Grupe\textsuperscript{22} presents a web based expert system for selecting academic major, where the student user is directed to one of the most suitable six majors from among 60. Deorah \textit{et al.}\textsuperscript{23} also recommend academic majors. Fong and Biuk-Aghai\textsuperscript{24} describe a system that recommends the university that matches the student’s profile. The system developed by Aslam and Khan\textsuperscript{25} is one for recommending faculties to students. Finally, Cline and Brewster\textsuperscript{26} present an expert system for automatically evaluating student concept maps.

2. Case 1: Supporting Course Selection Decisions

2.1. Motivation

For this project our aim was to create a rule-based system for Manufacturing Systems Engineering Program students at Sabanci University to prepare their schedules using Oracle Policy Automation (OPA). When preparing the rule base, we investigated rules related to prerequisite courses, student’s GPA (Grade Point Average), which courses open which semester in, which area the student is specializing in, and which courses the student has readily registered to.

2.2. Project Progress

In order to reach our aim we collected data from the university’s database of courses, namely BannerWeb. First of all we listed all courses in MS Excel and we connected them with their pre-requisites. This way the obtained a lists of the courses and the relations between courses. Furthermore, we represented this data in the yEd graph visualization software. It is possible to assign colors to nodes in yEd. This way, connection between
courses and three main course types as area electives, core electives and required courses became clearly distinguishable (Figure 1).

![Visualization of co- and pre-requisite relationship between courses](image)

**Fig. 1.** Visualization of co- and pre-requisite relationship between courses

We used the logical connectors “and” and “or” to establish logical connection between a course and its pre-requisites. If a user should take all pre-requisites of a course, “and” is the operator to identify relationships. On the other hand, if one of pre-requisites of a course is sufficient to take that course, “or” is the operator to describe the relationship. Also, through the “exists” command, system associates one entity with another entity. We established the course definition as an excel table with four columns, which are CourseID, CourseName, and CourseCredit and CourseCategory. Another Excel document used by OPA is the student database, which receives student number as input. By this single input, OPA matches student number with student name from the student list. Also, OPA saves all historical data of a user for the next session with the user. This way, a user can easily observe her/his historical movement on OPA as a user. For instance, a student can easily follow his/her credit and course level at past semesters. In addition to this, OPA calculates total credits for all sub-groups of courses is required, core and area electives. And then, OPA gives the user total credit by summation of all units. We finally planned and executed various scenarios and test cases for testing the developed export system.

In addition to the mentioned rules, several other types of rules have been defined:
Global Controls and Determinations: These are rules that globally determine student and credit attributes, such as the following:

- Eligibility check is completed if
  - total credit is known and
  - student name is known
- The student is sophomore if
  - total credit > 47 and
  - total credit < 81

Individual Course Eligibility Checks: These are rules that determine student eligibility for each course such as the following:

- The student is eligible for ENS204 if
  - The student has taken NS101
  - And
  - Either
    - The student has taken MATH101 or
    - The student has taken MATH102

Course technical rules: These are technical rules that tell which courses are taken, such as the following:

- The student has taken CS201
  - Exists(all courses of the student, course id = “CS201”)
- The student has taken ENS208
  - Exists(all courses of the student, course id = “ENS208”)
- The student has taken ENS491
2.3. Results

The developed rule-based expert system meets the user with a welcome screen. Next screen takes as input the student number and in order to match student name and record action logs of the user. Third screen is designed for taking course list from the user (Figure 2). The user chooses his/her previously taken courses from a list, where the list is read from a source file by OPA. Once the user submits the courses taken, the lists of courses that s/he can take is listed (Figure 3).

If a user wants to know the reason why s/he is not eligible for a course, the user should click “Why?” button on the results screen to reach that information, and learn the reason (Figure 4).

3. Case 2: Supporting Scholarship Decisions

3.1. Motivation

The goal of the second project was also to assist university students by providing recommendations. After discussions in the early stages of our project, we decided to implement a scholarship advisor that will present, with a student’s criteria, which scholarship s/he is eligible for. There are many institutions that offer scholarships to university students, however every foundation’s criteria and budget is different. Our aim was to match universities students with the scholarships that they are eligible for. This can help students in easily finding what they are looking for and it can decrease a task that would take weeks to minutes.

We used OPA again for this project. In our case, the rules would be based on the students’ GPA, their target university, their income, their skills, and other relevant criteria. OPA asks multiple choice questions that are related with the criteria of the scholarships. From the
home screen, the student clicks on the scholarship and then answers the related questions, and finally the results screen shows if s/he is eligible for the scholarship or not.

Fig. 2. Screen where the user (student, advisor, staff member) enters the courses taken so far

Fig. 3. Screen where the courses that the student can take are listed
3.2. Project Progress

To realize the project, we firstly gathered the data of all the scholarships in Turkey and their criteria. Finding all the rules wasn’t easy since some of the scholarships did not have a website but had just a phone number. Criteria for some of the scholarships were not available on the websites since the application period was over. After scanning through scholarships, we found approximately 200 scholarships, half of which were only reachable by phone and didn’t provide a website. We eliminated those scholarships from further consideration. From among the other 100 scholarships, only 51 had the criteria publicly available on their websites. From that point on, we deducted the rules of eligibility for each scholarship and separated every one of them in an Excel sheet so that while writing the rules for OPA, it would be easier and faster.

Furthermore, after gathering all the data we parsed their requirements and pre-requisites. For one institution, there may be an income requirement, while for another, meeting just a GPA threshold is sufficient, and thus we divided all the information into subcategories. To make it more organized, every scholarship’s criteria is listed in an Excel file so that every rule can be seen and plugged into OPA easily. With this system the students’ search and information gathering time can decrease significantly, and they can filter through significant amount of information and work to achieve on their goals. Moreover, we believe students would not be the only people who would appreciate this program. Also, foundations can receive applications from students who are more likely to be selected, and consequently their workload can be reduced, as well.
The second stage of the project consisted of writing all the rules in OPA. Figure 5 illustrates rules for a particular scholarship. This rule writing was done for each of the 51 scholarships.

![Rules for a particular scholarship](image)

**Fig. 5. Rules for a particular scholarship**

- 71 Yüzyıl Eğitim ve Kültür Vakfı Bursuna başvurumaı hak kazanır mı?
- Adana Ticaret Odası Bursuna başvurumaı hak kazanır mı?
- Aydın Karahisar Sanayi ve Ticaret Odası Bursuna başvurumaı hak kazanır mı?
- Alanya Eğitim ve Kültür Vakfı Bursuna başvurumaı hak kazanır mı?
- Alanya Eğitim ve Kültür Vakfı - Dr. Üzeyir Gahri Bursuna başvurumaı hak kazanır mı?
- Anadolu Eğitim ve Sosyal Yardım Vakfı Bursuna başvurumaı hak kazanır mı?
- Anadolu Vakfı Bursuna başvurumaı hak kazanır mı?
- Antalya Eğitim Vakfı Bursuna başvurumaı hak kazanır mı?
- Antalya Üniversitesi Destekleme Vakfı Bursuna başvurumaı hak kazanır mı?
- Başbakanlık Bursuna başvurumaı hak kazanır mı?
- Bereket Vakfı Bursuna başvurumaı hak kazanır mı?
- Bosra Kocabiyik Vakfı: Bursuna başvurumaı hak kazanır mı?
- Diyanet Vakfı Bursuna başvurumaı hak kazanır mı?

**Fig. 6. The first screen that the user sees in Case Study 2**
3.3. Results

The lists of possible scholarships (in this program there are 51 scholarships) appear on the user’s first interface. The user starts the program by clicking on the scholarship she/he wants to test the eligibility for the first. There are some common rules required for most of the scholarships, the question is asked only once and used in other scholarships as well. Examples for these common rules are the school student wants to attend or is attending, students major, and is whether the student is a Turkish citizen or not (most frequent criterion).

In Figure 6, the first interface that the user sees is presented. Here in our example, let us assume that the student clicks the link “Anadolu Eğitim ve Sosyal Yardım Vakfı Bursuna başvurmaya hak kazanır mı?” (“Is the student eligible for the Anadolu Eğitim ve Sosyal Yardım Vakfı scholarship?”) and starts answering the questions in order. Questions appear successively as they are answered in a way that preserves eligibility. If the answer violates eligibility, then the expert system automatically stops and prompts that “The student is not eligible for this scholarship”. For the specific scholarship mentioned, the correct sequence of answers would be (yes, yes, yes, yes, no, no, no), and if this sequence is entered the reason page would look like Figure 7. If, for example, the second question is answered no instead of yes, the reason page would look like Figure 8.

![Anadolu Eğitim ve Sosyal Yardım Vakfı Bursuna başvurmaya hak kazanır.](image)

Fig. 7. Result screen displayed when the student is eligible, with the rationale for the suggestion (eligible, apply) is explained
Fig. 8. Result screen displayed when the student is *not* eligible, with the rationale for the suggestion (not eligible, do not apply) is explained

3.4. Discussions

The biggest problem this rule-based system presented was with the questions where variables were used. Rather than the system asking whether someone’s GPA is greater than 3.5, we decided it would be better if the program asked what the GPA was and then decided if it satisfies the rules. The problem arose when for example we wanted to know the students major. It would have been impractical to ask the user to answer questions like “Are you a Business major?” and if the answer is no asking “Are you a Industrial Engineering major?” and so on. So again we decided to use the variables. But since the program is written in Turkish, the user could enter the major using capital letter, small letter, Turkish letters or English letters. This makes a huge number of possibilities. This is where we decided to search for a pull-down menu for these variables.

The student using this automation doesn’t have to answer all the questions for the 51 scholarships; by clicking on the few scholarships they want to apply they can answer the related questions. The system also doesn’t require the user to answer the same question more than once even if the question’s answer is required for different scholarships. For example the citizenship criteria is one of the popular ones, so in the first eligibility test the user answers it and the program remembers the answer and does not ask the same question for the other scholarships.

Our first thought was for the student to write down all his or her information and then all the available scholarships would have been presented. Unfortunately due to the fact that every scholarship has different rules and the importance of the answers change in different
scholarships, that kind of system would not have worked. In the end, we have assembled the information and the available software and completed the case study successfully.

4. Conclusions

This paper reported and discussed the development of two educational expert systems: The first expert system is that course advising system which recommends courses to undergraduate students. The second system suggests scholarships to undergraduate students based on their eligibility. While there have been reported systems for course advising, the literature does not seem to contain any references to expert systems for scholarship recommendation and eligible checking. Therefore, the scholarship recommender that we developed is the first such system in the literature and can provide a frame of reference for following studies. Both systems have been implemented and tested using Oracle Policy Automation (OPA) software, a convenient integrated development environment for expert systems development. Future stages of the project can involve the deployment of the developed expert systems on the university intranet and/or the internet.

Acknowledgements

The authors thank Mete Sevinç for creating the visualization of co- and pre-requisite relationships between the courses.

References


