E-Administrator: A Case Study of a Web-Enabled Student Administration System

Thomas Abraham, Kean University, 1000 Morris Ave, Union, NJ 07083, 908-527-2492

Abstract: E-Administrator is a web-enabled system, which collects data and generates reports on students in a graduate program. It is used by the staff to support program administration including maintaining student records and reporting on student progress. This paper describes the development life cycle of the system. It is intended as a case study for students in a systems development course. It provides enough detail so that it also can be used for administrators, such as department chairs, to develop their own systems. The system was built using widely available technologies. Also, the challenges facing our graduate program are probably very similar to ones faced by most academic departments. These features should make this paper useful both as a template for other similar system development efforts and as a case study in the classroom.

INTRODUCTION

The Master of Science in Management Information Systems (MSMIS) program is a graduate program in a medium-sized state university in New Jersey. It offers a unique degree, jointly sponsored by the Management and the Computer Science departments. There are 120 students enrolled in the program at this time. Two coordinators, one from each department, administer the program. The coordinators are full-time faculty who receive release time for this job. A secretary and two graduate assistants support the coordinators. Faculty from each of the departments, as well as several adjunct faculty, teach in the program. The coordinators schedule the classes and contact the department chairs to staff them.

The student body is very diverse both in terms of educational background and ethnicity. Students with an undergraduate degree in any discipline can apply for admission. About half of the student body is international, with China, India and Ghana being heavily represented. To ensure a reasonably uniform base of knowledge, there are 6 prerequisite courses that each student must have on their transcripts. They must also take the GMAT or GRE exam. Students may be conditionally admitted even if they don’t have all the prerequisites. Program administration must follow up on these students to make sure that the prerequisites are completed in a timely fashion.

Students must get 36 credits to qualify for the degree. Of these, 21 credits are from required courses. These courses follow a rigorous course sequence. For instance, the Systems Analysis course cannot be taken until a Data Structures class has been completed. Similarly, the Systems Design class follows both Analysis and Database Systems classes. Administrators have to approve student registration for three required classes to ensure that the proper sequence is maintained. Students must maintain a B average over the course of their enrollment. A grade below a B cannot be used for credit. If a student gets a C or lower in a required class, it must be repeated. Once again, administrators have to monitor student progress and contact students when they are in trouble.

During the course of a semester, several mailings are made to the students. From time to time phone calls are also made to students. College publications like Peterson’s send out an annual survey that asks for a great deal of data on students. This includes both academic and demographic data.

ANALYSIS

Requirements

The old system consisted of an Access database with some input screens and output reports. The database primarily consisted of a Student table. It was mostly used for printing mailing labels. Once a year some data analysis was done for reporting statistics to publications such as Peterson’s Guide or to the Deans of the two Colleges (Business and Math/Computer Science). Average GPAs, GMAT scores and demographic data was mined from the database. A university-wide system called Colleague was also available. This maintained student registration and grade information. However, the system provided a very poor interface and did not provide for ad-hoc reporting.

A major program need is the ability to monitor student progress. Students admitted conditionally into the program often are missing prerequisite courses. Coordinators need to identify and follow up with these students. The old system did not provide for this capability. Coordinators also need to identify students who are failing academic requirements such as the B average. This was also not available from the old system. Student advisement requires that student records, including transcripts, are readily available to the
coordinators. In particular, students have to obtain an approval from a coordinator to register for certain classes including Analysis, Design, Database Systems and the MIS Capstone. The coordinator has to verify that the prerequisite classes have been completed before granting the approvals. With the old system, that meant logging into a mainframe system called Colleague, which did not provide a graphical interface.

Another program need is course scheduling. Several months before each semester, the coordinators need to be able to create and update a schedule of classes, days, times and instructors. The schedule needs to be in the form of a grid or some other visual format. The scheduling process is iterative and the schedule will need to be generated several times.

The campus network was installed and is maintained by the Office of Computer Information Services (OCIS), which is a centralized group that services the entire University. The primary objective of the network and the group is to connect every desktop PC to the Internet and to support a set of standard applications. Local, program-specific systems are not supported. Thus, the PCs in the MSMIS office did not belong to a single Local Area Network (LAN) with a common server. Without network support, the office could not even use a common printer!

The old system was not even based on a file server architecture since the client PCs in the program were not part of a single LAN. Instead, MS Access was installed on all the clients and copies were made frequently to each client and a backup to a mainframe. Several people including the secretary, graduate assistants, and program coordinators entered data into the system. It became a nightmare trying to keep all the changes on several machines synchronized. Only one person could update the database at any moment and obviously the others could not view the latest information until the updated database was copied to their machine.

The program coordinators are also faculty in the two sponsoring departments. Like most faculty, they often work from home or from offices at other locations on campus. Ideally, they could advise students and discuss faculty schedules over the phone or via E-mail. This was difficult since the old system did not support remote access.

The old system did not meet the needs of the program. There was a need for multiple users to remotely access a single database. In addition to printing mailing labels, there was a need to create a semester course schedule and to view student transcripts. There was a need for reporting on student progress on completing prerequisites. There was the need for exception reporting on students failing to meet program standards.

The MSMIS Program Administration Process

Graduate Studies sends over a batch of student applications on paper forms. The coordinators review these applications and accept them or reject them or place them on hold. Applicants are often conditionally accepted into the program subject to completing several prerequisites within a year of acceptance. The secretary then records the application and the decision in the computer database. She generates letters of acceptance or rejection and mails them to the applicants.

The registrar’s office sends out course scheduling forms well in advance of the semester. The coordinators use a long-range plan to help them schedule the courses for each upcoming semester. Instructors are also assigned to the sections during this process. The coordinators create a grid indicating days and times for each section. The grid provides a visual guide to ensure that the sections are well balanced. This grid is called a tentative course schedule. It is refined and updated right up to the start of the semester. Sections may be added or dropped or moved to a different time slot. Instructors may be changed. Building and room numbers (provided by the Registrar) are added or modified. The tentative schedule is posted on the program website to help students plan their registration.

Students request approval to register for designated classes over the phone or via E-mail. The secretary or the graduate assistants record these requests. Coordinators review the requests on a daily basis. If the student meets the requirements, they enter the approval on the Colleague system.

Students register for classes online or in person using the University’s registration system (Colleague). The old system did not capture any of this data. However, the MSMIS program has unique needs for tracking students’ progress, which the Colleague system does not meet. The new system will need to capture this data in a batch process using the class rosters provided by the Colleague system. At the end of each semester, a batch of grade rosters will be used to enter the grades into the system too. This will allow coordinators to monitor student progress through transcripts and exception reports.

Students inform the secretary when prerequisites are completed or when contact information changes. She enters these changes into the system. From time to time the coordinators schedule new faculty and courses. They will also sometimes modify prerequisites as well.
The key outputs of the system are the Student Transcript, Course Schedule, Missing Prerequisites Report and the Failing Grades Report. The transcript contains student identification and a list of courses taken, with semester and grade information and a GPA. The schedule contains course identification, instructor, day, time, and location information for each section offered in a semester. The missing prerequisite report contains a list of students with the prerequisites each has not completed and the required completion date. The failing grades report contains a list of students with the courses, semesters and the failing grades that each one obtained.

CASE QUESTIONS

1. Develop a logical process model of the MSMIS system.
2. Develop a logical data model of the MSMIS system.
3. Develop a Network Architecture DFD.
5. Develop a prototype of the Reports
6. Develop a prototype of the Input Forms
7. Develop a prototype of the User Interface
8. Describe some of the tools used in the construction of a system.

TEACHING NOTE

The required courses in the MSMIS degree include a sequence of 3 systems development lifecycle courses – Systems Analysis, Database Systems and Systems Design. Students are expected to develop an operational prototype of a system. They must turn in a detailed report documenting the models and prototypes they develop. At the start of the Design class, we provide students with a sample case study that explores the process and clearly documents the solution. This case study is used in class to illustrate the problem and the process used to solve it. It also clearly lays out what is expected from the student report. The report is expected to contain data, process and object models at both logical and physical levels. It must also document the prototype throughout, interface and input screens.

Students use a tool such as System Architect (Popkin Software) for creating the models. The database management system is usually Microsoft Access. This is widely available and easy to use. Traditionally, the front end was created using Access or Powerbuilder. Today, however, systems development is moving away from the 2-tier client-server architecture and towards a web-based or network computing architecture. This trend is not restricted to just E-Business. Even traditional applications are now being web-enabled. It is important that students are exposed to the tools underlying this architecture. This paper describes a web-enabled system that is constructed using relatively user-friendly tools, which are widely available to most students.

Logical Modeling

The process is modeled using Data Flow Diagrams (DFDs). The top level DFD is the Context Diagram, shown as Figure 1, and it lays out the net inputs and outputs to the E-Administrator system. Typically, a decomposition diagram would be drawn to identify the structure of the system. It is clear from the description above that the system consists of 5 functions – Admissions, Registration, Scheduling, Reporting and Data Maintenance. The Admissions function is used to record, review and act on applications. The Registration function is used to capture data on student registration transactions as well as student grades. Student Transcripts and Class Rosters can be viewed at the end of this process. The Scheduling function is used to add semesters and schedule sections of various courses for the semester. It is also used to assign times, faculty and rooms to each section. A graphical schedule can be generated at the end of this process. The Reporting function is used to generate exception reports such as the failing grade report and missing prerequisite report. It also generates a summary report on student demographics. Finally, the Data Maintenance function is used to add and edit student, faculty, course and prerequisite data. The next level DFD is the System DFD, shown as Figure 2, and it lays out the events associated with the functions. Students are encouraged to model the process themselves before viewing our models.
Clearly, to meet the needs of the program, the database had to expand to handle course and registration data. The existing Access database contained a large Student table, which contained student contact information as well as their academic history. It was enough to generate mailing labels and most of the demographic report. However, it did not contain any data that could be used to produce reports on student progress in the program. A course can be offered over many semesters and several courses are scheduled each semester. A student can register for many courses and a course has many registered students. Figure 3 shows the Entity Relationship Diagram (ERD) developed for the new system. Once again, students are encouraged to develop their own data model before viewing the one presented.
DESIGN

Given the absence of a common LAN and the lack of network support staff, the typical File server and Client/Server distributed architectures were not feasible. However, every client was now connected to the Internet. It was now possible to leapfrog the other architectures and go directly to a network computing architecture. A web-enabled system would meet the need for remote access to a common database. There is no need for any special network support. In terms of hardware, the clients already existed in the form of the PCs on each desk. A server with its own IP address would be needed. In terms of software, the system would need a web server, a database management system, an integrated development environment (IDE) for program construction, and browsers on the clients. The College of Business already had a license for Microsoft products. It was decided that the simplest and lowest cost solution would be to adopt these products for the system. We would continue to use the Access database management system. Microsoft supports web development through a server-side technology called Active Server Pages (ASP). ASP has built in objects for connecting to the
database and for handling communication with the browser. ASP is available on any of the Microsoft web servers. Windows 2000 contains IIS, which is a powerful web server. Personal Web Server (PWS), which is built into the earlier Windows operating systems, is sufficient for a system of this size and scope. Unfortunately, browsers do not follow a common standard and supporting several standards was beyond the scope of this project. So, we opted for Internet Explorer 5 and above as the browser we would support. All the code was to be written in either VBScript (server-side) or JavaScript (client-side) and HTML, using a Microsoft software development environment (SDE) called Visual InterDev. Figure 4 shows the Network Architecture DFD for the web-enabled system.

The Database Design is laid out in Figure 5 as the Physical Schema. The prototypes of the Output Reports are Figures 6 - 8. The prototypes of the Input Screens are shown in Figures 9 - 12. Finally the User Interface prototypes are shown in Figures 13 - 17.

![Figure 4: Network Architecture DFD](image)

CONSTRUCTION

The PC on one coordinator’s desk was designated as the server. Computer Services set up the machine with its own IP address. Personal Web Server (PWS), which is built into the Windows operating system, was enabled on the server. PWS automatically supports Active Server Pages (ASP). The Access database was also located on this server.

Programming was done on Visual Interdev. However, several other tools such as HomeSite, are just as good. All the code was written in either VBScript (server-side) or JavaScript (client-side). The scripts were also located on the same server. The system was built with existing hardware and software and labor. Therefore the system was constructed entirely in-house and at no additional cost.

CONCLUSION

E-Administrator has worked pretty well so far. It can be accessed remotely from anywhere in the world. It allows the staff to simultaneously access and modify student data. It provides the program coordinators with a way to track student progress and monitor potential problem students. It has significantly reduced the reliance on paper-based filing system.

Due to space limitations, only selected figures are included.