DEVELOPING A MORE EFFECTIVE COURSE TO DELIVER CIS EDUCATION

Thom Luce, Ohio University, luce@ohio.edu
Vic Matta, Ohio University, matta@ohio.edu
Corrine Brown, Ohio University, brownc@ohio.edu

ABSTRACT

Students' engagement in learning has consistently been linked to their retention rates in the Management Information Systems (MIS) program, their academic success (^{2}), and interest in the MIS major. Increasing student engagement was seen as vital to increasing enrolment in the MIS major, which began a rapid decline in 2001. One of the steps taken to curb this was to make changes to the curriculum, some in its programming constituent. We found that many institutions offer beginning programming courses using legacy character mode editors to develop static applications. Most MIS programming courses and textbooks are oriented towards teaching programming and application development with Windows applications. This paper describes changes made in our programming courses with respect to teaching development of interactive web pages instead of windows applications while using the same Integrated Development Environment (IDE). We discuss the rationale for the change, the challenges we faced, and the results of this transformation.

Keywords: Programming, Engagement, ASP.NET, VB.NET, C#.NET, Curriculum

RATIONALE

Are your students excited about their first or second programming class? Are they actively engaged in learning new programming languages? How many freshmen, sophomore or even junior students have any idea what a command prompt is or how to run programs from the command line? How often will the typical IS major ever need to do that? How many of your students, born and raised in the age of interactive video games and the Internet, are turned-on by programs that read and write console data? How many students say "Wow, that is cool" when they complete the proverbial "Hello World" program? How many of your students know what a desktop or Windows application is? How many will be actively involved in creating desktop/Windows applications in their careers?

The Association to Advance Collegiate Schools of Business (AACSB) Standards challenge us to "generate transformational learning" which requires an "investment of significant time in learning experiences" and "that time includes contact between students and faculty members, contact among students, and individual and personal *engagement* of students in learning and *applying* (emphasis added) knowledge and skills " [1, pg 52] AACSB goes on to say that "the most effective learning takes place when students are involved in their educational experiences. Passive learning is ineffective and of short duration. Faculty members should develop techniques and styles that engage students and make students responsible for meeting learning goals." [1, pg 54-55].

The requirements stated in the second paragraph are relevant to the questions asked in the first paragraph. Almost every introductory programming class known to the authors starts with, and many stay with, desktop applications. Just look at the available textbooks and trade books. Almost to a volume they teach VB, VB.NET, C, C++, C#, Java, or whatever with desktop applications, either using the Console for input/output or creating Windows (or the UNIX equivalent) forms. Even most Java books start with desktop applications and eventually get around to applets. How exciting is that? How many of your students call home (email home) bubbling over with excitement because they were able to type something at the command line and get an answer? How do these students show their work to their friends, relatives or prospective employers?

PROCESS

What do students know when they come to your beginning programming class? They know about video games and they know how to use the Web. Some may also know at least a little about the creation of static web pages, or maybe even how to use Flash® or Photoshop® on web pages. Based on this, and a desire to build more continuity into the MIS major, we moved both of our programming courses to the ASP.NET 2.0 environment starting with the 2004-2005 school year (yes, we were using beta software). The first course in the major introduces programming concepts using Visual Basic. This course formerly used VB 6, moved to

VB.NET with version 1.0 of the .NET framework and moved to VB.NET in an ASP.NET 2.0 environment. The second programming class was Java until the advent of .NET 1.0, when we moved it to C#.NET, and we now teach the course in an ASP.NET 2.0 environment.

Why did we make these changes? We did it for excitement, for engagement and because the students had previous knowledge they could bring to bear. Students get excited about developing web pages that they can show their friends, relatives and future employers. They are actively engaged in the process because they understand the goal and the reason for doing the programming. They have experience in interacting with things like text boxes, checkboxes, radio buttons, dropdown lists, buttons and links, so they have a hook upon which to hang new knowledge.

Our first programming course, the first required course in the MIS major, teaches VB.NET as a way to create dynamic web pages and places an emphasis on basic programming constructs with some emphasis on comprehending and implementing presentation layer components (textboxes, radio buttons, etc.) [3]. Students learn how to write and apply loops and decisions in business programming. They learn how to accept and validate user data (both through code and with Validator controls). Students learn how to acquire and render data from data access objects (again, both through code and with data bound controls such as the GridView and DetailView) and they learn about event-driven programming. The course also teaches some of the components necessary to orchestrate the flow of data between pages and during the postback cycle.

After completing the first course, students take two courses combined and integrated into an eight hour block that must be taken as a unit. These courses provide an introduction to systems analysis and database concepts. ASP.NET development is used to demonstrate both system design concepts and database concepts. Students in this combined course build knowledge based on the framework created in the programming course.

Next in the sequence is the second programming course, a course that teaches C#.NET in an ASP.NET environment. This course introduces students to a different programming syntax (C# is a C family language), to Object Oriented Programming concepts and to Event Driven Programming concepts. While this course does introduce a new programming language, it places a major emphasis on the creation of systems and linkages between different types of systems in order to achieve business systems integration [5]. Students learn how to create and consume web services [7] to achieve integration across disparate systems. The learning curve for web service creation is relatively low because in the ASP.NET environment, web services look very similar to normal user interfacing web pages.

The next course in our sequence is a systems development/project management course that takes students through the complete system development life cycle and its management. Students create web-based applications for real clients using ASP.NET and the language of their choice (see Reference #5 for additional discussion of tools used in this course). This course typically involves several tutorial sessions on advanced database concepts, programming events related to data bound controls, accessing data on data bound controls, etc.

RESULTS

Why do this? Students get excited. They are engaged in the learning process because they can relate to the goal, even if they have never programmed before. Students produce products that they can share with friends and future employers. They produce products that are easily displayed in Electronic Student Portfolios. They produce projects that are easy to assess to determine if desired learning outcomes were attained.

Each class builds on the previous classes and uses the same integrated development environment (IDE). This minimizes the learning curve associated with new IDEs and allows for development of more sophisticated systems, more comprehensive project management principles, and better web application development by the time students finish the last class.

All the courses mentioned are taught using Microsoft's Visual Web Developer Express 2005 (VWD). They could also be taught using Visual Studio 2005 (VS) but VS is a huge program and supports both web and application development in VB.NET, C#.NET, J#.NET and managed C++. VS 2005 is available as part of the Microsoft Academic Alliance so the price is reasonable. VWD is a much smaller product and only supports web development, but it supports web development in VB.NET, C#.NET and J#.NET. VWD is currently free and even if Microsoft starts charging for it next year, the cost should be low and it will most likely be included in the Academic Alliance program.

VS 2005 and VWD both offer several advantages over previous versions of Visual Studio. Students can develop and test web applications without needing to deploy the application to a server. VS and VWD both contain a built-in version of Internet Information Server (IIS) and both treat a web site project as an Application. This means students can develop and test applications without copying files to a web server and without needing the administrative access to the server required to create an application. Since web pages run in this restricted web server environment, it is possible to use the full debugging and tracing facilities built into both products from student machines, without the need for administrative access to the system.

VS 2005 and VWD also ship with SQL Server Express, a scaled down version of SQL Server 2005. Using SQL Server Express allows students to develop and test SQL Server applications using the SqlDataSource control rather than creating Microsoft Access applications using the AccessDataSource control (but this is still available if you want to use it). The advantage of this is scalability. In many cases the only thing necessary to move an application from SQL Server Express to SQL Server 2000 or 2005 is changing the data source's ConnectionString.

The approach we have taken is not without its pitfalls and problems. First, there are essentially no textbooks currently on the market that support this approach. If you wish to try this, you must either try to adapt a book that focuses on desktop applications (that hasn't worked very well for us), create materials, handouts, on-line references on your own, or even write your own textbook.

One interesting and unexpected pitfall of the approach presented here is that lesson plans and technology don't necessarily become simpler in the ASP.NET environment. If fact, there are a number of issues to consider. First is the persistence of application data. In an interactive desktop application you enter data, the data is stored in memory, you enter more data, the program manipulates and stores the data, and you continue until the application ends. Web pages are, by original design, stateless [4]. This means that both instructor and student must be aware of the problem and learn ways to save data between postbacks. In the web environment, by design the web application does not handle or track the data after posting it back to the server; i.e., after you submit data to the server, the server processes the data and everything is lost when the server posts the page back to the client.

ASP.NET maintains data associated with a page and its controls in the ViewState [10]—an encrypted set of data that is passed between the client and server in the page header. Controls with ViewState enabled "remember" their content from one postback to the next. Data entered in textboxes, data selected from checkboxes and radio buttons and results stored in labels and textboxes are maintained (assuming the controls ViewState property is set to true) from one postback to another. The programmer can also store and retrieve data directly from the ViewState with statements such as: `ViewState["username"] = "Jon Smith";` The primary limitations to preserving data this way are the fact that the data is passed back and forth between client and server and the fact that only data that can be serialized, or converted to a string of characters, may be stored in the ViewState.

What about intermediate results and other data that you don't want the user to see? Some data could be stored in hidden textboxes, but a knowledgeable user could still view it by examining the page source. There is also a problem because only data that can be converted to a string can be moved into a textbox. Another option, and the one we teach in our first class, is to place data in Session variables. Session variables can hold any valid .NET type [9] and exist as long as the user is actively using the web application. However, they do disappear if too much time elapses between round-trips to the server or if the user has disabled session cookies. Session variables are a good way to pass data from one web page to another while data in the ViewState are normally associated with one page. The important point here is the need to address this issue almost from the beginning of the class.

We find it very difficult to teach array processing concepts in the ASP.NET environment, especially because of the issues discussed above. We have found, however, that we can discuss array concepts while dealing with substrings and collection objects. For example, you may determine which items in a ListBox control are selected by looping through a ListBox control's Items collection where each item is accessed as a subscripted component of the collection. Similar processing is possible with records returned by a DataReader. Both of these examples also serve as good, practical places to demonstrate the use of loops of different types (for loops when the number of items in the ListBox are known, while loops to access records from a DataReader until the end-of-file is reached, for-each loops to access data from certain collection objects).

Some Object Oriented Programming concepts are easy—a simple web page is a class file. Controls have methods and properties used or called in normal interactions with the control. Information hiding can be mentioned because while we use the properties and controls, we don't know how they work or what they look like at the code level. Creating new classes and demonstrating inheritance is more difficult, largely because of statelessness issues. Objects instantiated from a class have to be saved during the post back cycle and the whole process of doing that feels very artificial, because it is. However, one good solution to this to create web services because web services exist in their own class file and messages are used to communicate between the main web page and the web service. Another possibility in a more advanced class would be the creation and use of DataAccessObjects to implement true n-tier architecture applications.

SUMMARY

How is the role of IS Education Changing? Active engagement in interesting, relevant learning keeps students interested in the topic, the course and the major. Gone are the days when business schools were looking for ways to turn students away from IS courses. Gone are the days of weed-out classes and high grade point requirements to enter and remain in the major. Here instead are days when we have to work to make our classes relevant and interesting for our students. We must work to attract students to our majors and then to keep them. Part of this can be accomplished by revamping our service courses and business core courses to highlight what IS is all about (certainly it's about more than how to use word processing, spreadsheets and presentation graphics packages).

Teaching programming in a web based environment is one way to keep students interested and engaged. Students come to us with a comfort level in using the web. They get excited when they can manipulate and make it work in a way that they want and they just may stay around to try another class. Figure 1 shows enrolment trends in our major courses. MIS 220 is the first class in the major. MIS 225 was the second class in the sequence until the 04-05 school year when MIS 320 became the second class and MIS225 was replace by MIS 400, but as the third class in the sequence. MIS 220 and MIS 400 are the two main programming courses discussed in this paper.

As you can see, our enrolments in the first class hit bottom in the 03-04 school year, the same year we started using ASP.NET 1.0 in MIS 220. We switched to the beta version of ASP.NET 2.0 for the 04-05 year and to the release product for the 05-06 school year. Enrollments in MIS 320 and MIS 400, while lagging because of sequencing issues, are also in the rise. The changes described in this paper are certainly not the only reason for the turn in enrollment, but they are a contributing factor.

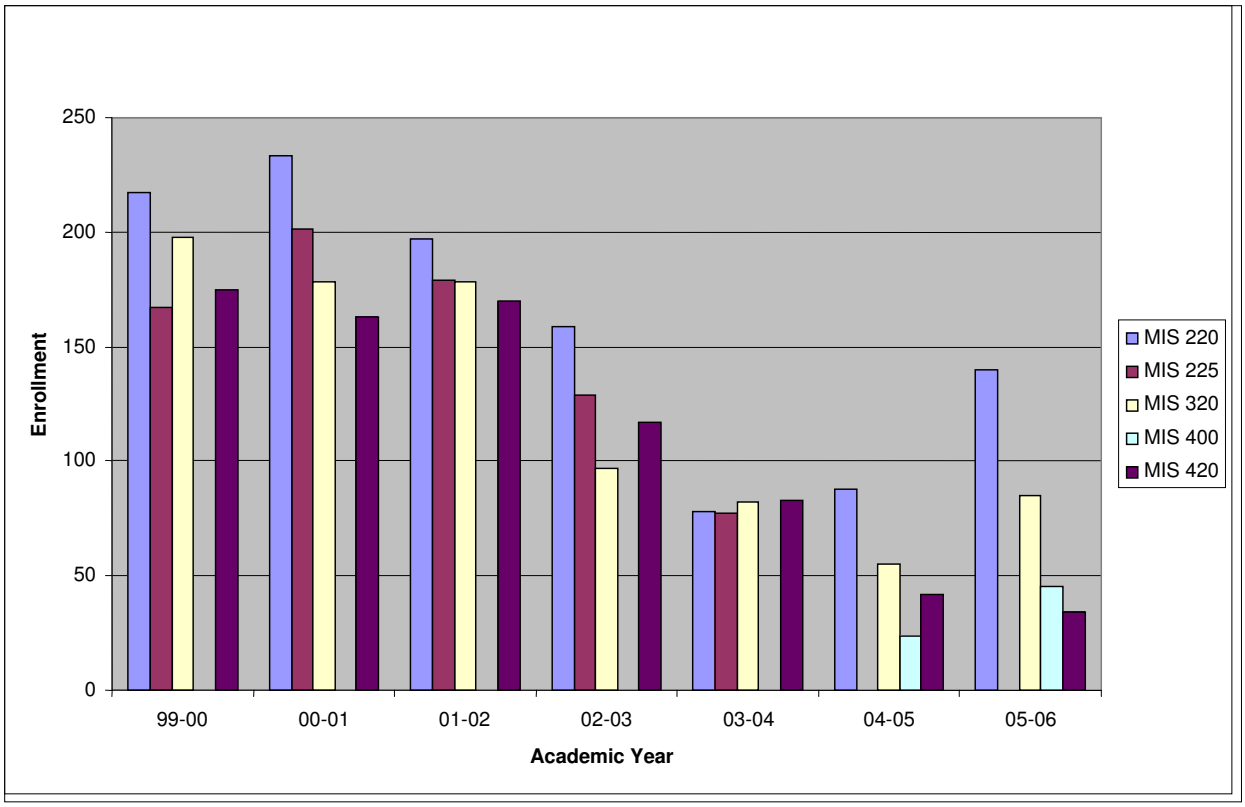

Figure 2. Academic Year Enrollments

REFERENCES

1. Eligibility Procedures and Accreditation Standards for Business Accreditation. *AACSB International* (2006).
2. Blank, W. (1997). Authentic instruction. In W.E. Blank & S. Harwell (Eds.), *Promising practices for connecting high school to the real world* (15-21). Tampa, FL: University of South Florida. (ERIC Document Reproduction Service No. ED 407 586)
3. Jezierski, E., Yajaman, N., Olsen, A., Gonzalez, D., Cibraro, P., & Cazzulino, D. (2003). Design and implementation guidelines for web clients. *Patterns and Practices Developer Center*. Available online from http://www.only4gurus.net/microsoft/DIGWC.pdf
4. Lobel, L. (2003). Managing session state on the client. Available online from http://www.ftponline.com/vsm/2003%5F09/magazine/columns/aspnet/
5. Luce, T. (2005). Moving the senior development class from web development to life cycle development—A case for Visual Studio 2005, *Issues in Information Systems, VI*, (1), 114-120.
6. Markus, M. L. (2000). Paradigm shifts—e-business and business / systems integration. *Communications of the AIS, 4*(10), 1-45.
7. Skonnard, A. (2005). The birth of web services. from http://msdn.microsoft.com/webservices/understanding/default.aspx?pull=/msdnmag/issues/02/10/xmlfiles/default.aspx
8. Tiobe. (2006). *Programming community index for February 2006.*
9. Visual Studio 2005 Documentation for Session Variables.
10. Visual Studio 2005 Documentation for View State.
11. Vorobiov, G., Dichter, C., Benninghoff, J., & Hewett, C. (2003). Developing and optimizing web applications on the asp.Net platform. *Intel Technology Journal, 7*(1), 47-59.