A Programming Course For Business MIS Students

Kelvin Sung¹, Alan Leong², Rebecca Reed Rosenberg³

Abstract - The Business Program at the University of Washington, Bothell has recently established a new Management Information Systems Concentration. As an integral part of the technology foundation, students are required to take the typical software engineering course from the computer science department. This paper describes the motivations behind this decision, and the results of collaborative efforts between the Business and Computing faculty in creating a custom programming course to prepare MIS students for the software engineering course. This newly created programming course is designed to challenge students to apply and refine their basic programming and problem solving skills, and in doing so, to experience the often tedious software application development process. The implementation of the new course is based on programming the automation of popular commercial software systems. The results of our initial offering indicate success in building on the modest technical background of typical Business students in designing and implementing programming solutions to moderately complex practical problems.

Index Terms – Information Systems, Software Engineering, Curriculum Issues, Non-Majors, Supporting Courses.

INTRODUCTION

The Business Program at the University of Washington, Bothell (UWB) [1] has recently established a new undergraduate Management Information Systems (MIS) concentration [2]. The goals of the new MIS concentration are to provide students with the technical and business skills to analyze and design effective information systems, successfully manage software projects, and optimize user experience. It is recognized that, among other technology requirements, rigorous coverage of programming and software engineering issues is very important in accomplishing these goals. To ensure proper integration of technology, faculty members from the Computing and Software Systems (CSS) Program [3] are intimately involved throughout the MIS concentration development process. This paper focuses on our efforts in integrating programming and software engineering into the MIS concentration.

It was decided that MIS students should take the existing software engineering course [4] together with students from the CSS major. We came to this decision because the required software engineering topic areas are similar and we want to promote collaboration between Business and CSS students.

Our existing software engineering course is an intermediate junior-level course, similar to that described in [5]. To properly prepare for this course, students should be proficient in problem solving based on programming fundamentals and have first-hand experience with the entire application development life-cycle. For students in the CSS major, these preparations come from the prerequisite data structure and discrete mathematics courses. However, for Business MIS students, many technical topics covered in these pre-requisite courses are irrelevant. A new programming course is needed to be designed to refine students’ programming based problem solving skills and to prepare for the software engineering requirements.

Since the UWB is on a 10-week quarter system, it can be overwhelming to cover all the required topics in one quarter. As such, students are required to take a typical introductory programming course as the prerequisite for this course. We designed this new problem solving programming course to build on students’ knowledge in programming constructs. This new course is based on designing and implementing solutions to practical problems users may encounter while working in typical PC environments. To avoid overwhelming students with complex syntax and/or sophisticated tools, we chose to work with a simple scripting language with basic text editors. To maximize the impact of students’ solutions and to provide students with relevant technical skills for their future careers, we chose to work with programming the automations of commercial applications: writing driver programs to integrate/customize/automate existing commercial software systems for specific tasks [6].

This new course was first offered in the Fall quarter of 2003, where we implemented the syllabus around Microsoft Office Applications and the Visual Basic (VB) Scripting language. This choice was based on its ease for prototyping and its relevancy and familiarity to the student. We have received overwhelmingly positive feedback and observed students successfully applying their problem solving skills in developing moderately complex programming solutions. Currently we are teaching this class the second time and are instituting a systematic study of student learning outcomes [7].

Section 2 of the paper discusses technology and programming courses of typical existing MIS programs and details our reasons for creating a custom programming course. Section 3 examines the goals, implementation constraints, and design guidelines for the new programming courses. Section 4 presents the syllabus and our implementation of the first offerings of the course. Section 5 summarizes the paper with

¹ Kelvin Sung, Computing and Software Systems, University of Washington, Bothell, ksung@u.washington.edu
² Alan Leong, Business Program, University of Washington, Bothell, leonga@u.washington.edu
³ Rebecca Reed Rosenberg, Teaching and Learning Center, University of Washington, Bothell, brosenberg@uwb.edu

October 19 – 22, 2005, Indianapolis, IN
35th ASEE/IEEE Frontiers in Education Conference
S1J-24
a discussion of preliminary results and our current efforts in improvements and further studying the effectiveness of this new course.

**BACKGROUND AND MOTIVATION**

Many existing MIS programs recognized the importance of integrated software engineering competencies in their core curricula. Depending on the specific goals of each program, the offering of related technology courses varies drastically. For programming courses, this variation ranges from no specific requirements in programming skills (e.g. [8,9]); to basic CS1-type skills (e.g. [10-12]); to proficiency in programming with data structures (e.g. [13]). These programming courses are customized for Business MIS students based on selecting topic areas from model curricula [5,14]. It is important to note that, since the model curricula are designed for educating technology professionals, many of the topics (e.g. linked-list data structure) are low-level foundations for more advanced technology courses. Straightforward adaptation of these subjects may not be the best way to serve the needs of typical Business MIS programs/students. In a similar fashion, the delivery of software engineering knowledge also varies between dedicated custom courses ([8,9,11]) and distribution of the knowledge in other technology related courses (e.g. [10,12,13]). In all cases, these courses are customized and offered by MIS programs and, in most cases, specifically for Business students only.

When approaching the integration of programming and software engineering topics into our own curriculum, the following factors are considered:

**Opportunity for interaction.** There are two important reasons to maximize the interaction of students from MIS concentration and students/professors from technology departments. First, taking technology courses with technology-major students means MIS students would be learning the subject matters with future professionals and from experts in the technology fields. Second, interacting with technology-major students in such close proximity will expose MIS students to subtle discipline-specific culture differences. These are delicate issues MIS professionals must understand.

**The programming course.** This course should concentrate on challenging students in applying programming based critical thinking skills; and in experiencing the design and implementation of programming solutions. We believe the topic areas identified by the model curricula for technology professionals (e.g. [5,14]) should be consulted but not necessarily adopted.

**The Software Engineering course.** The topics areas required by the MIS concentration are similar to the course content of the existing software engineering course for CSS major students. Because of the largely qualitative content, this is an excellent venue for MIS and CSS students to learn and collaborate with each other. We are modeling their future professional relationships and creating an environment for healthy interdisciplinary exchange.

**DESIGN CONSIDERATIONS**

The new programming course should accomplish three major goals:

**Critical thinking skills.** This course should refine students’ critical thinking and problem solving skills in the context of computer programming. Students must be able to bridge the gap from qualitative description of solutions to a structured design and implementation of solutions based on programming fundamentals.

**Hands-on development experience.** This course should provide students with a hands-on experience and understanding of the software application development process. As preparation for the software engineering course, students must understand the nature of the end-work product that is being produced before trying to understand the support services that are necessary to produce it (e.g. project planning and tracking, selection of life-cycle model and ways to get customer feedback, quality assurance techniques, need for requirements gathering and definition, etc) [15]. In addition, MIS professionals must work closely with and manage technology workers throughout their careers. It is important for them to experience and understand the fundamental nature of technology professionals’ job.

**Practical skills.** This course should teach practical technical skills that are relevant to the students’ future career. For the majority of Business MIS students, this is probably the last programming course they will take in their academic careers. It would be desirable that, besides the foundational concepts, students gain technical skills that are directly applicable in their future careers.

When evaluating the above goals against the ones specified in the technical curricula [5,14], we see that the first two goals overlap somewhat with introductory programming courses, while the third goal is irrelevant for technology professionals. This observation suggests that the new programming course should adapt topics from programming fundamentals as suggested by the model curricula. However, the last goal for our course demands different approaches to the delivery of the underlying topics. For example, although studying and programming tree traversal algorithms in C++ would contribute to accomplish the first two goals, after the class, it is highly unlikely that MIS students would have the opportunity to apply the technical skills learned (e.g. C++ syntax, pointer operations, integrated development environment tools, etc.). When compared to designing and implementing search operations for typical hierarchical file-system structures in PC environments with HTML driven VBScripts, we notice similar concepts (solution modeling, problem solving skills) are conveyed. While in the second case, there are much greater potentials for applying the technical tools and skills involved (Scripting and Markup Language coding, and operating/file system interfaces).
We are on a 10-week quarter system. Many of our undergraduate students are transfers from partner lower-division community colleges. This combination presents interesting challenges. It is difficult to design a 10-week course to accomplish the above goals for non-technical students with no prior programming experience. On the other hand, a two-quarter programming sequence is over-kill for our requirements and presents an unnecessary barrier. For these reasons, our class requires one previous introductory programming course.

With the above defined goals and understanding of implementation constraints, we approach the designing of our course with the following guiding principles:

**Concepts vs. Tools** – It is important to clearly distinguish the concepts from the tools that are used to reinforce the concepts. In addition, it is of utmost importance to avoid overwhelming the non-technical students with difficult to learn/use technical tools.

**Relevancy Of Examples/Assignments** – It is important for students to appreciate that the concepts they are learning have practical relevancy in their daily lives. Whenever possible, in-class case studies and homework assignments should leverage the students' typical interaction with computers. Coupled with the goal of teaching practical tools, these can be powerful motivating factors.

**Future Self Learning** – Students must be able to adapt to future technological tools and apply the same concepts. Problem definition and solution design must be clearly separated from technology implementations. One approach to clearly accomplishing this separation is by demonstrating multiple implementations of the same solutions based on different technologies.

These guidelines led to our choice of designing the course based on simple scripting language automating popular commercial applications.

**The New Programming Class**

As discussed, the contents of this class must address three major areas: refining critical thinking problem solving skills in the context of computer programming; experiencing complete application development process; and learning relevant/practical technical skills/tools. With students’ prior knowledge in programming constructs and junior-level maturity, we took an integrated approach where we challenge students on all three fronts throughout the quarter. The course begins with demands on straightforward efforts and progresses into requirements on careful multi-week design, planning, and implementation exercises.

The design of our course is structured around the Case Study approach (e.g. [16-18]) where all concepts are presented based on example solutions to practical problems. The balance between abstract concepts in problem solving and learning of technical skills is achieved by examining a typical office productivity software system (e.g. web browser, word processor, spreadsheet, database, etc.) and exploiting the lack of integration and need for customization. We identify shortfalls of current systems, describe solution approaches, design solutions with programming constructs, and finally demonstrate implementation based on current automation technology. With this strategy, the following are the topic areas covered in the 10-week course.

| Introduction and Programming Tools                                                                 |
| Working with Web Browsers                                                                            |
| Conditional Statements and Positive/Negative Logics                                                  |
| Application Automations + Word Processors                                                             |
| Loops and Problem solving with iteration                                                              |
| Working with Spreadsheets                                                                              |
| Functions, Scope Rules and Modular Design                                                            |
| Automation with multiple applications                                                                |
| Data Structures: Strings, Arrays                                                                      |
| Working with libraries: Strings and Regular Expressions                                              |
| Object Based Design                                                                                   |
| Working with pre-defined libraries: Dictionary Objects                                               |
| Algorithm Analysis: Sorting and Searching                                                              |
| Working with email systems                                                                            |
| Algorithms Design: Divide and Conquer                                                                  |
| Working with Databases                                                                              |

The first half of the course aims at relating programming constructs to solution modeling strategies while introducing application automation concepts. During this time, students work with complete software solutions, learn new programming language/environment, new technology (application automation), while refining their critical thinking problem solving skills and re-familiarizing themselves with basic programming fundamentals. In this way, we avoid overwhelming students with too many new concepts and technologies while still challenging them on all three aspects of the course. The more complex programming elements and approaches to analyze/design algorithms are introduced only after students become comfortable with the new technologies. By the second half of the class, students are expected to examine realistically-complex practical problems and design/implementation solutions based on customizing/integrating multiple commercial applications. Example Case Studies are included in the Appendix Section at the end of this paper. The three cases are, respectively, from beginning, middle, and final part of the quarter. Explicit commercial software systems are referenced because these are actual Case Studies used in our first offering of the course.

We adopted Sprankle [19] as our textbook for its systematic approach to logical analysis and the discussion of productivity software systems in the context of basic programming constructs. As discussed, we have chosen the VB Scripting language and Microsoft Office Applications as our targets for automation because of availability and students’ familiarity. Similar to [20,21] our choice of language is motivated by simplicity for non-major students and our experience agrees with [22]: that students find the language easy to learn and the language accomplishes the required task. Typically, lecture times are equally split between studying of solution design strategies and discussion of implementation with automation technology. Approximately one programming assignment is designed for each of the topic areas. The initial assignments are due weekly and are meant for reviewing programming constructs and...
practicing the automation technology. The final two assignments are more significant-sized 2-week projects (e.g. Case 3 in the Appendix). We have designed two examinations (a mid-term and a final) to assess students’ learning. These are open book/notes exams where students are expected to demonstrate ability in applying critical thinking skills and technology learned in solving practical problems.

RESULTS AND CURRENT STATUS

This course [23] was first offered in Fall 2003 and we have observed initial success. Our observations are based on three aspects: students’ performance and ability in implementing the final project (similar to Case 3 in the Appendix); students’ course evaluation (overwhelmingly positive), and students’ continued interests (many students decided to minor in CSS IT [24]). Examples of student feedback from the Fall 2003 offering include: "[This course] allowed me to understand difficult practical problems, and even solving some of these problems." "VB Script coding is applicable to many current jobs? to automate the business process." In other informal conversations, students described how while working with Business faculty on collecting on-line survey information, they were able to automate the extraction of large amounts of survey data into spreadsheets based on knowledge gained from this class.

While we are encouraged by these results, we have also learned some important lessons. As expected, students were much more motivated and interested in the automation technology when compared to the more abstract problem solving portion of the lecture. Another observation is that students tend to start late with their assignments. A typical CSS student understands that the risk involved in starting late is that one’s solution may not compile/run and thus would accept significantly less credit for the assignment. However, Business students seemed to have no concept of such risks and accept significantly less credit for the assignment. However, Business students seemed to have no concept of such risks and expected credit for their time spent on thinking/coding. One other important lesson we learned is that Business students valued and thrive in group-based problem solving opportunities much more so than typical CSS students. These are precisely the subtle discipline-based culture differences we need to understand and experience.

In our current (second) offering of this class (Winter 2005) we have made some modifications designed to enhance students’ learning experience. To further engage students in abstract problem solving, we have designed/integrated closed-lab sessions with lectures. In these lab sessions, students have to program simple and technology-independent solutions based on lectured problem solving strategies. For example, when discussing recursive divide-and-conquer problem solving strategy, the accompanying closed-lab session requires students to implement a simple recursive algorithm to merge two fixed size sorted arrays. In this way, we reduce the amount of materials presented in lecture by engaging students in actively understanding the concepts taught. In addition, learning from our previous experience, students are encouraged to work in 2-person groups. Finally, we constantly remind students to start on their assignments early. We understand this is less than an ideal solution. In the next offering of this course, we will partition assignments into smaller units with multiple due dates.

The adopted Case Study approach is well understood and widely endorsed in the field of Computer Science Education (e.g. [16,18]). However, our integration of specific commercial applications (i.e. Microsoft Office Programs) into the practical problem and solution formulation is a deviation from the general practice of abstract solution formulation based on foundational concepts. It is important to formally assess and verify student learning outcomes. Such an assessment will validate our approach and identify weaknesses for improvements.

We have designed and are currently administering a mixed-method approach student assessment program [26]. Our assessment instruments include pre- and post-tests, staged assignments, student perceptions (through logs students completed along with the staged assignments), and faculty self-assessment (based on reflections on in-class activities, student work and student feedback). We are collecting this information and will proceed to analyze the data once this quarter comes to an end.

In all cases, one important future work is in continual emphasis of the separation of concepts from implementations. It is important to define the problem statements and design solutions independent from specific technologies. In this way, it would be possible to demonstrate to students multiple implementations of the same solutions based on different technologies. We have ported all the existing Case Studies to Jscript and C#. Currently, we are investigating the possibilities of porting the automation target to a new productivity software environment (e.g. Star Office) under different operating systems (e.g. Linux).

Finally, it is interesting to notice the similarities of this course to other practical, problem solving, programming courses designed for non-technical students (e.g. [20,21,27-32]). Although in our case the course is designed specifically for serving MIS students, due to the many similar issues with non-technical students, the resulting class contains many similar characteristics. For example, the choice for scripting language (e.g. [20,21]); programming solutions with significant and visible impacts (e.g. [21,28,30]); students’ familiarity with the problem domain (e.g. [27-29], etc. The major difference of this course is in its role as a preparation class for a software engineering course where we want students to "experience the entire software development process." Based on our limited experience with Business students, it seems possible that this class could be suitable for general student populations. In a 16-week semester system, one can dedicate more time to the initial coverage of programming fundamentals and turn this into a CS0-type programming course suitable for general student populations. In our case, it is interesting to investigate the possibilities of shifting the emphasis of the class from in-depth problem solving to programming construct and eliminate the programming knowledge prerequisite and turn this into a 10-week CS0 programming course.
ACKNOWLEDGMENT

Thanks to all CSS341 students for working so hard with the authors on constantly providing feedback; to Charles Jackels, Frank Cioch, Steve Holland, and Connie Pace, the key administrators for their encouragements and support. Wilson Kam ported all the VBScript case studies to Jscript and C#. This work is supported in part by the University of Washington, Bothell Worthington Technology Award (2005).

REFERENCES


0-7803-9077-6/05/$20.00 © 2005 IEEE 35th ASEE/IEEE Frontiers in Education Conference S1J-28
APPENDIX

Example case studies: Here are three example case studies, presented in increasing complexity. Our experience is that, at the beginning of the quarter our students are somewhat able to verbalize solutions to these problems with high-level programming constructs, while they are able to implement automation solutions at the end of the quarter. Notice the goals of all these case studies are to develop software programs to automatically solve the problems. Currently there are about 50 such case studies available off our course website [23].

Case 1: Problem solving with conditional statements working with word processor: I have a hard drive containing thousands of Microsoft Word documents. Many of these documents contain the misspelled word “colour”. Write a program to fix all the misspellings.

Case 2: Problem solving with loop/conditional statements working with spreadsheet and web-html page: I am interested in investigating different loan and re-payment options, with different loan amounts (e.g. $20,000 vs. $30,000), interest rates (e.g. 5% vs. 6%), and monthly payments (e.g. $400 vs. $500). I would like to find out how long it will take to repay the loan, and how much total interests I will pay in the end. Please write a HTML document implementing simple user interface to invoke your program so that you can prepare reports using Microsoft Excel to automate my investigation.

Case 3: Problem solving with requirements in design/analysis working with multiple software systems: I would like to analyze all the junk emails I received by computing statistics and preparing reports on who are, and when are they, sending me junk emails. Please write a program to automatically retrieve all the junk emails from my Microsoft Outlook system; export sender information and receiving date to create a database using Microsoft Access; analyze the receiving statistics using Microsoft Excel; and prepare a report of the findings using Microsoft Word.