ABSTRACT

Modeling classroom experiences after “real-world” experiences is a challenge in all disciplines. Currently, computer systems are complex in nature and require diverse talents and large teams to complete. This paper describes a process by which teams of college level students can develop the skills necessary to write programs as a unit, develop program delighters, and ultimately write a major system to self-imposed specifications. Through this process the students develop interpersonal skills and simultaneously create programs that adhere to stringent program specifications. This multi-semester developmental experiment outlines the advantages of team programming and presents a model based on grounded instructional theory.

Keywords: Programming, Pedagogy, Visual Basic .NET, Andragogy, Adult Learner.

INTRODUCTION

Development of programming systems in today’s environment requires the collective efforts of many designers, analysts, and programmers. Unfortunately, students customarily learn to program in an environment in which collaboration is less than a noble endeavor. In order to better prepare students for work experiences they will face upon graduation, they need to be in an environment that parallels the work environment that awaits them. This paper outlines a model that can be used to create such an environment. The model is based on grounded instructional design and tested over a period of six semesters.

In addition to addressing team programming, the concept of contract programming is introduced. Contract programming allows the team to come one step closer to real-world experiences. The students, in teams, decide on a system they want to create. They develop specifications for that system and come to agreement with the instructor on what reward, in the form of a grade, will be earned for various stages of the system they create.

The students’ creativity is challenged through the introduction of the delighter. A delighter goes beyond what is expected in a system or program. It is that extra functionality, interface design, or other design feature that gives the application an “extra flair” that differentiates it from the norm. Collectively, the real-world based assignments prepare students for a major project in team based programming. They are required to develop documentation and programming techniques that enable team design and development. Emphasis is placed on team responsibility. Creativity is challenged in every assignment. This process is all wrapped in a solid instructional design model. Through this process the students develop interpersonal skills and create programs that adhere to stringent program specifications.
INSTRUCTIONAL DESIGN

One of the problems faced in college classrooms is the lack of team based instructional designs to assist instructors’ with the development of real world environments. Textbooks are typically focused on individual learning (4, 6 & 9). The examples and assignments are not based on team involvement. There are very few models for team based learning of programming concepts available. Subsequently, there are very few opportunities for the students to experience “I do and I understand” when it comes to team development (8).

Instructional design for adult learners is significantly different from that of the younger learner. Pedagogy typically refers to the teaching of the young while andragogy refers to the teaching of the adult learner. This paper is written from the perspective of andragogy. Andragogy is premised on critical assumptions about the characteristics of adult learners. The characteristics are different from the assumptions about child learners on which traditional pedagogy is premised. Andragogy assumes as a person matures there is (1):

- change in self-concept which moves from one of being a dependent personality toward one of being a self-directed human being,
- an accumulation of a growing reservoir of experience that becomes an increasing resource for learning,
- readiness to learn which becomes increasingly oriented to the developmental tasks of social roles,
- change in time perspective from one of postponing application of knowledge to immediacy of application, and
- a focus on problem solving instead of subject centered learning.

It is the responsibility of the college instructor to insure that an intellectually stimulating environment is provided for students (8). When developing a learning experience, they must be aware of the differences between the younger and the more mature learners in their classrooms. This model outlines and emphasizes the self-directed needs of the adult learner through the encouragement of independent thought and creative programming. Also, the model acknowledges the higher level of background experience that the college level students bring with them. Additionally it emphasizes a shift from subject centeredness to problem solving centeredness.

Diana Laurillard stated, “Knowledge has a contextualized character, which means that we cannot separate knowledge to be learned from the situations in which it is used” (8). A corollary of this argument is that the acquisition of inert concepts will be of no use if the student cannot apply them. This model places emphasis on presenting an environment as close to a real-world experience as possible within the constraints of a college classroom. Through adoption of the model the instructor can emulate situations of everyday learning by contextualizing real-world activities.

An effective instructional design requires careful selection of learning opportunities. Gagne and Merrill stated, “The procedure of working backwards from goals to the requirements of instructional events is one of the most effective and widely employed techniques” (3). The implementation of their theory was utilized in most of the assignments where the final product
was given to the students in an executable form. The students must reverse engineer the program. The goals were outlined by the executable’s functionality. That functionality was not typically intuitively obvious. This approach takes Gagne’s and Merrill’s approach one step further by making the goals available to students only through exploration and experimentation. By requiring students to explore the executable’s full functionality they gained significant insight into problem resolution (8). The goal is presented by the instructor in the form of the executable. The students must develop an experimental framework to understand the question.

The assignments and project provide a roadmap for the implementation of team based experiences for advanced programming classes. It does not address the instructional methods used to support the assignments. To better support the development of solutions for the assignments, a Lao Tzu Approach of “keeping the process simple and the instruction clear” and avoiding the stigma of failure was taken. As stated in Renner’s book *The Art of Teaching Adults*, students “would rather avoid a new behavior than risk being seen as a fool” (11). While emphasis should be placed upon making the student an active participant of the learning experience, extreme care should be taken to not belittle their approach to a problem.

Throughout the course the Locke and Atomistic Model was used (10). In an object-oriented programming class, the model is nicely paired with the subject material. Simply stated, with a hand held calculator there are prepackaged functions. They do nothing until acted upon. The programmer must invoke these functions to create a program. If each of the functions is explained individually, such as the objects from the Tool Box or math library, then students will be better able to formulate a solution to a problem. Students will better grasp the functionality that is available to them. Unfortunately, this functionality is typically given out of context. It is viewed as many scattered pieces to the problem and not as a solution a specific real-world problem. It is the instructor’s responsibility to put the functionality into a meaningful context for the solution of a complex problem.

**Assignments**

The assignments outlined below are indicative of the complexity of the nine assignments used each semester. Assignments are both individual and team based. Emphasis is placed on team solutions by encouraging students to share experiences even with individual based assignments. Many of the assignments were developed from actual business problems. The individual based assignments allow students to demonstrate their individual capabilities, while the team based assignments provide students with opportunities to work together to resolve more and more complex problems. If a delighter is required for an assignment, 90% of the grade is earned as the student or team properly programs the functionality. The last 10% can be earned by implementing delighters. A delighter is defined different for each of the assignments.

**Order Interface:** This assignment is individually based and is used to determine the basic skill set the students bring with them to the class. This assignment draws on the student’s ability to develop multi-form applications, with data entry and formatted output fields, and draws upon their knowledge of the math library. This assignment and the List Box assignment were part of a consulting project for a small local business that sold tractors and maintenance parts. The student develops an application, with a splash screen and an interface screen, for the ordering of
product from a business of their choice. This aspect of the assignment encourages creativity by allowing the student to select an area of business they are interested in. Applications have been developed to sell gourds, guns, and automobiles. For this assignment, students must compute various taxes and provide basic sub-total and total calculations. Delighters for this assignment usually include images of the product and printed output. With this assignment, the basic ground rules for minimal features are set. All programs are required to include splash screens and menu structures that include Exit, verification of Exit, Help, and About features.

**List Box**: This individually based assignment concentrates on the student’s understanding of the list box functions such as add, delete, and clear. In addition, file manipulation functions such as select, create and read are needed. The student must develop a program that has a text entry field and two List Boxes (A and B). Data can be entered into List Box A through the text entry field or through the importing feature of a user selectable text file. Items can be selected from List Box A and moved or copied to List Box B. Items from List Box B can be removed, deleted, or stored in a user selectable data file. This assignment is the first real challenge in developing a delighter. The ease of use and the basic navigation design are the most common delighters.

**Digital/Analog Clock Stopwatch**: The Digital/Analog Clock is the first team based assignment. The teams develop a clock program with the ability to display the time in analog and digital format. This assignment was part of a usability test tool that was developed for a local manufacturing company. In the analog format, the clock should have hour, minute, and second hands of appropriate length and width. Also, the program must display the date. The application must include a stop watch function that keeps time to the 1/100th of a second. Typically, the students break up the assignment into the digital components of the clock and the analog components of the clock. From a coding perspective, this is typically inefficient and results in the development of dual timers, one for digital and one for analog. Some examples of the delighters include Mickey Mouse shaped forms for the analog clock and 12 or 24 hour format selection for the display of the digital time. Additional delighters have enhanced stop watch features, such as day of the week display, and interfacing with the system to resolve current date and time.

**Pong**: This is the second team based assignment in which the students create a program, similar in function, to the traditional Pong game. It is an academic exercise to hone the students’ math programming skills. The program is designed for one player against the computer. The objective of the game is for the player to stay active for as long as possible. Each time the ball bounces off the computer, the speed of the ball is increased. The program must keep track of play time and keep a log file of the best 10 times. Through this assignment the students are introduced to programming mouse and keyboard events, advanced logic, and math functions. The student is given a sample of an executable program from which they must reverse engineer program functionality, functionality that is not intuitively obvious. Delighters for this assignment have included the multiple balls in play and introducing an “English” on the ball. From a team perspective, the sharing of development responsibilities is less evident and typically results in one student taking a lead roll.

**Screen Saver**: This assignment was developed after a student organization created a basic screen saver as a fund raising project. Very little detail is given to the students for this individually
based assignment. The student develops an animated screen saver. The screen saver must be parameter driven and function like a typical Windows screen saver. This assignment challenges the creative skills of the students. They have explored the use of Bezier curves, timers, and videos. Since this assignment is fundamentally a creative exercise, no delighters are required.

**Many Words Database:** This individual assignment challenges the student to create a list of passwords based on words stored in an Access database. The assignment paralleled a project to create random passwords for a student computer lab at the university. A sample database with several thousand words is made available to the teams. The database contains words of various lengths, from one to eight characters long. The program gives the user the ability to add words to the data base. Logic must be developed to eliminate duplicate words. All stored words should be in capital letters. This feature requires the students to explore character manipulation. The program uses concatenations of the words in the database to create passwords so that the resulting password is, depending on the user’s desire, from 1 to 24 characters long. The passwords are to be displayed in a List Box. No duplicate passwords are allowed. The user should be able to scan the list box and eliminate passwords or to add passwords of their own creation. The number of passwords desired and the length of the password is user selectable. If the database does not contain enough words to create the selected number of passwords, the program should notify the user. The program will have the ability to create a text file with the passwords in it. The text file’s name should include the last name of the student, as well as an indicator of the length of the passwords, and the number of passwords. Delighters, for this assignment, include animated interfaces and adding numbers to letter combinations if the number of passwords cannot be met with the existing database. This assignment challenges the students’ ability to solve complex logic problems and to develop interfaces for databases.

**Four-color Map Problem:** This team based assignment is modeled after an academic problem outlined in the VNR Concise Encyclopedia of Mathematics (5). Gefen & Govindarjulu stated, "Mapmakers know that any political map can be drawn with four colors in such a way that any two countries with a common frontier (not just a point) are colored differently" (4). The teams are to create a program that demonstrates this assumption. The teams need to develop a data structure that represents the 48 contiguous states in the United States, Canada, Mexico, and the oceans that surround the states. The development of an appropriate data structure is the key to the solution of this problem. The program should be able to create the data structure, save it, retrieve it, and test it for the color map assumption. Delighters typically include animation and the presentation of the results in a graphic form.

**Chat Room:** This team based assignment introduces students to the network objects available in the .NET environment. The idea for this assignment came from the need of a small company to have an internal chat program for their employees. The teams are to create a program that enables two individuals to chat with each other across the Internet. The program can be either peer-to-peer based or client/server based. The students are given some rudimentary networking code and expected to enhance the code through research and experimentation. This assignment sets the foundations for the game contract. The most common delighter is expanding the number of users in the chat session and adding emoticons.
Vector Intercept Problem: This team based assignment relies heavily on the student’s understanding of sine and cosine functions, timers, graphic manipulation, and complex logic in an object-oriented environment. This assignment was spawned from an actual Department of Defense project. This assignment exemplifies a military or business situation where many variables are needed to meet a moving corporate goal. The teams write a simulation for the following scenario: Up to 128 airplanes are airborne. There is one movable aircraft carrier on which they can land. The program simulates the return of the aircraft to the aircraft carrier. Initially, a selection of the number of planes is made, and then each plane is given a random heading, speed, and altitude. A sample executable program is given for students to reverse engineer the solution. This problem expands the students’ use of interactive creation of objects and the assignment of numerous properties and methods to those objects.

Game Contract

About four weeks into the semester the students must begin this team based assignment. It is worth 25% of the student’s total grade. It simulates a business situation in which the team is hired to create a product, a game. The game must be based on a totally new set of rules or on an existing game with significant modifications to the existing rules. The games must be network based and must include animation or graphics. As with the Chat Room Assignment, the teams can base the network implementation of the game on a peer-to-peer model or a client/server model.

The first phase of the assignment requires the students to develop a preliminary design of the game. The design document starts of as one page overview of what the students’ plan to create. The students are encouraged to concentrate on board games without significant movement of screen objects. Projects have included advanced forms of many card games, Monopoly®, Battleship®, Minesweeper® and three-dimensional chess. Through a series of interactions with the instructor, the teams then create a detailed design document outlining the rules and playing environment. Without exception, the initial ideas penned by the students are too complex. The process of developing the design document is a matter of defining what can be accomplished in the limited time remaining. Once the design document is completed, the student must post their document on the web for other teams to view. The last attachment to the design document is one defining rewards. The student and the instructor come to agreement on what is necessary to earn an A, B or C for the assignment. At this point the design document is binding and work begins.

Throughout the remainder of the semester the students give informal in-class updates on their progress and problems. They are free to share ideas with each other. Very little changing of the design document is allowed. The final portion of the process is a demonstration of the product. The students must create an installation disk for the game and demonstrate the game to the faculty and their peers. During the demonstration, the students must be prepared to field questions and explain how the various parts of the application work.
SUMMARY AND CONCLUSIONS

At the end of each semester the students were asked “What would you have done differently?” and “What have you learned about team programming?” The following are common responses to these questions.

1. It still takes nine months to have a baby. Having more people working on the program does not mean you can get it down in less time.
2. We had differing techniques for coding. At times it was difficult to integrate the different styles without having to rewrite one part or another.
3. Splitting up the work was the difficult part. Many times one of us would just do it all.
4. Documentation is a necessity.
5. We should have started sooner. We waited too long to start coding.
6. We did not define the common procedures on which to build from.

At the end of the semester the students evaluate the course. The evaluation score for the spring 2004 semester was 4.92 out of 5. Previous semesters were as high as 5 out of 5. Using this approach, the students were challenged but not overwhelmed with the work. From the first assignment the expectations were high. A focus on the “delighter” concept forced the student to think beyond the stated scope of the problem and explore not only additional program features but additional coding techniques not covered in lectures. The format of the course demonstrated that the students will learn and participate if they are properly motivated and learning is made interesting. Interjection of real-world assignments in a real-world environment gave the students appreciation for what they could accomplish. One student put it succinctly: “He made us learn more than I thought I could.”

REFERENCES