Assessment of Student Attitudes and its Impact in a Hands-On Programming Model for the Introductory Programming Course

Sheikh Ghafoor, Stephen Canfield, Michael Kelley, Tristan Hill
Tennessee Technological University, Cookeville, Tennessee

STEPHEN CANFIELD

Stephen Canfield is a professor in the Department of Mechanical Engineering at Tennessee Technological University. He received his Ph.D. in mechanical engineering at Virginia Tech in the field of parallel architecture robotics. His research interests include robot kinematics and dynamics, topological optimization of compliant manipulators and in-space mechanisms active student learning and undergraduate student research

SHEIKH GHAFOOR

Sheikh Ghafoor is an assistant professor in the Department of Computer Science at Tennessee Technological University. He received his Ph.D. in Computer Science from Mississippi State University. His current research is in autonomic resource management for high performance computing environment, programming model for parallel adaptive applications, and fault tolerant computing. He is also engaged in research in of computer science education.

TRISTAN HILL

Tristan Hill is master's student in the Department of Mechanical Engineering at Tennessee Technological University.

MICHAEL KELLEY

Michael Kelley is a master's student in the Department of Computer Science at Tennessee Technological University.

Assessment of Student Attitudes and its Impact in a Hands-On Programming Model for the Introductory Programming Course

Sheikh Ghafoor[1], Stephen Canfield[2], Michael Kelley, Tristan Hill
Tennessee Technological University, Cookeville, Tennessee

Abstract:

Many students enter engineering programs as a result of a hands-on experiences that they have had directly or indirectly in their past. The freshman-level programming course provides an ideal opportunity to build students' perceptions of engineering, creativity, and notions of how things work. Ideally, students will begin learning programming in an environment that matches their notions of engineering, that engineers design systems to control the world around them, and then later move to solving advanced models that describe how the world works. A recent model has been implemented in the college of engineering at Tennessee Tech (TTU) to base the initial programming experience on hardware in the loop approach where the programming target is a micro-controller. This course has been offered in both C/C++ and Matlab programming language.

From multiple previous implementations, we see that the students that engaged in the hands-on, hardware-based programming activities reported a more positive early experience with programming and its relation to the engineering curriculum relative to their comparison-group peers. The students participating in the project also reported improved confidence in their ability to learn and use programming and note its importance in their engineering studies. However, we have observed in both the treatment and control group that the students' change in attitude toward programming in some cases is neutral or negative. This result was not expected and did not correlate directly with the degree of engagement with the model. This paper will explore these findings in greater detail. It will provide an overview of the model and the expected outcomes in student attitudes towards programming. It will present the findings in student attitude resulting from three semesters-worth of project implementation. Several potential factors that led to these results will be presented. The paper will conclude with the implications of these findings on planning assessment tools for entry-level programming experiences based on student attitudes.

1. INTRODUCTION

Many students enter engineering programs as a result of hands-on experiences that they have had in the past. However, engineering programs often do not provide enough practical experiences early in the curriculum [1]. The freshman-level programming course provides an opportunity to build on incoming student's perceptions of engineering and the tools engineers use. The traditional entry-level programming course for engineers is based on learning C, Fortran or Matlab to solve numerical algorithms associated with common engineering models. Any use of a computer as a device to control physical events is generally contained in upper

[1] Author for correspondence: Department of Computer Science, Tennessee Technological University, 110 University Drive, Cookeville, TN 38505, 931-372-3482, sghafoor@tntech.edu

[2] Author for correspondence: Department of Mechanical Engineering, Tennessee Technological University, 115 West 10th St., Cookeville, TN 38505, 931-372-6359, SCanfield@tntech.edu

level courses. While creating programs to solve numerical analysis problems is an important tool for engineers, we contend that the current model is inverted based on a pedagogical basis. Ideally, students would begin learning programming in an environment that matches their notions of engineering; that engineers design systems that control the world around them; and then later move to solving advanced models that describe how the world works. Based on recent advances in microcontroller hardware, associated programming environments and many examples of integrating programming with hardware in the loop for upper classman engineering, the authors propose to alter the context in which programming is taught to engineering students at TTU. The course has been implemented as an initial programming experience based on a hardware-in-the-loop model, retaining the C or Matlab programming standard but using as a programming target a micro-controller (a computer designed to interface with the outside world) to interface to simple physical systems. This is intended to result in a programming experience that will demonstrate one way in which engineers use computers and be appropriate for early understanding of engineering.

The program has been implemented over multiple semesters at TTU. An evaluation of the first implementation of this course is presented in [2]. The evaluation methods review in part student attitudes and perceptions about computer programming and how engineers use computers and programs to solve real-world problems. During the early implementation of the project and first assessment results, the student attitude data appeared to reinforce the authors' opinions of students' attitudes before and after the course; that students would be more positive about programming, its importance in engineering and the need to continue to learn programming during their education after the course. However, as survey data from subsequent courses based on this model were obtained, a different trend was observed that goes. This trend is illustrated and discussed further in this paper.

The remainder of this paper will proceed as follows. Section 2 will discuss some related work for the proposed model and assessment results. Section 3 will summarize the model and organization of the course. Section 4 will summarize the attitude survey findings and present a discussion on these results. The paper will end with concluding remarks in section 5.

2. DISCUSSION OF RELATED LITERATURE

A review of related literature is presented in two areas: the first applies to the underlying basis of the model, and the second applies to an outcome from attitude surveys in similar contexts. First, a background for the model is presented. Applying pedagogically-based improvements to the engineering programming experience throughout the undergraduate program has seen significant attention in the literature. The greatest focus has been on modifying the approach to teaching engineering computing to freshmen (see for example [4-7]). One common theme is the use of computing tools (such as spreadsheets, Matlab or MathCAD [3-5,8]) as an alternative to high-level programming languages (C or Fortran). Other popular approaches have been in involving students to perform activities in a more realistic environment [9], and programming within the context of gaming [10]. The proposed hands-on programming model presented by Canfield et al., [2] focuses on a programming experience that is in a context appropriate for early engineering students by introducing MCU hardware as the direct programming target.

The proposed model [2] for improving the programming experience is designed around a principle of learning that are highlighted in How People Learn [11], and emphasized within a context related to STEM education [12]. This principle is described in within the framework of the programming model as follows:

> Students enter the engineering curriculum in general and the early programming course in particular with preconceptions about how engineering and computers work. In order to effectively develop them as successful engineers, their initial understanding must be engaged and developed to see the full picture of computers in engineering, which goes beyond the traditional desktop picture.

To illustrate this point consider the simple output control of an MCU (microcontroller unit). This programming experience allows students to relate many fundamental concepts in programming (variable definition, discrete nature of variables, memory type, and memory control) in an immediate fashion to the computer, the computer components and the program. For example, defining a variable (creating that variable in ram) can be tied to direct control of a series of LEDs attached to an output port of the MCU. Furthermore, the early programming experiences can scaffold on students early expectations for physical cause-and-effect, direct control of memory through a program. Finally, the third principle is addressed through several aspects. In teaching students the basic process of creating and debugging computer programs, the meta-cognitive processes involved in solving open-ended problems should be demonstrated and reinforced. The programming target hardware provides students the tools and roadmap to direct learning beyond the classroom. The repetition of programming applications throughout the course and follow-on curriculum moves toward advanced, open-ended problems ending with the capstone design that will encourage computer integration in design.

This principle indicates that learning requires engagement. Therefore, measures for engagement and attitude are employed in evaluating the effectiveness of the model. Measuring of attitude toward different engineering disciplines has received attention in literatures [13-17]. These literatures eludes that in general, attitude is measured by assessing students' perceptions of the engineering professions and of their preparation of study the field. The students' perceptions are measured through surveys, often times using pre and post course surveys. Generally the questions in the surveys are in form of statements which express a view point and attitude towards the discipline. The students are asked to give their level of agreement or disagreement on each statement using some form of Likert[17] scale. Most of these surveys use five point Likert scale with higher values indicating greater levels of agreement with the statements. The scale is designated as 5- strongly agree, 4 - moderately agree, 3- no opinion, 2 -moderately disagree, and 1- strongly disagree.

3. COURSE DESCRIPTION

The course is design as a traditional "Introduction to Programming for Engineers" course in a two, three or four hour arrangement with a lab opportunity. The course follows a traditional syllabus and lecture series. The programming examples labs and assignments however are developed for a microcontroller (MCU) based target. The MCU programming examples and assignments provide a context-appropriate demonstration of engineering practice. Transparency in the programming applications can be achieved through programming the MCU in C that gave direct control of memory and I/O registers, making it a good fit for electrical or computer engineering students. When the programming platform is Matlab, a greater level of abstraction for the input/output registers is easily achieved through simple functions, while allowing the students to directly control input and output devices. For the C programming approach, a commercial integrated development environment (IDE) [17] serves as the program editor and cross compiler to interfaces with the MCU. For the Matlab programming, students write their program in the usual fashion as a script in an m-file, and then cross-compile and load the program using a single Matlab command at the command line. The cross compiling routine as well as MCU-specific functions are contained in a MCU toolbox for matlab.

3.1 Course Content:

The course content was based on the pre-existing syllabi, with the primary topics presented in Table I.

Table I. Course Topics

Topic #	Topic
1	Introduction to programming, program design, process
2	Data types:
3	Std libraries, math libraries, math operations, text operations
4	Std libraries, math libraries, math operations, text operations
5	Conditional statements: if, else, else if
6	Repetition: while, for
7	User defined functions
8	Multi-dimensional arrays

3.2 Course Hardware:

A primary distinguishing feature of this work is to implement a microcontroller rather than traditional PC as the initial programming target. Further, the authors contend that the MCU selected should be appropriate for engineering practice, and at the same time be readily accessible interface through on an evaluation board. One such product is the Dragon12 plus board [18] which is based on the Motorola HCS12 MCU and was implemented in this project. This MCU is widely used in engineered products, and the Dragon12 evaluation board, shown in fig. 1 below, has numerous input / output functions integrated directly with the HCS12 MCU Table II provides a summary of the primary features of the Dragon12 evaluation board.

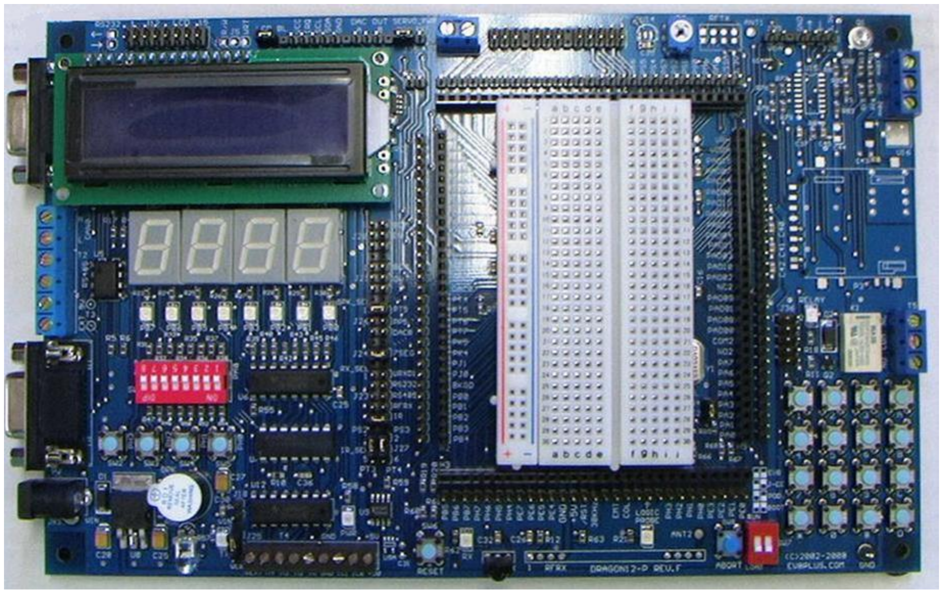
Fig. 1. Dragon 12 Plus MCU board

Table II. Summary of MHCS12/Dragon12 features

Product	Capability
Processor: MC9S12	16bit CPU, 24Mhz 256K Flash EEPROM, 12K RAM Serial communication, 10-bit ATD, timer channels, PWM, discrete I/O, interrupt I/O
Evaluation Board: Dragon12, www.wytec.com	Output Devices: 2x16 digit LCD, single-row LEDs, 4 – 7 segment LEDS, Piezo speaker
	Input Devices: 8 dip switches, 4 momentary switches, 16-key keypad, IR proximity sensor, Photoresister
	Other: Motor driver (H-bridge)

3.2 Course Assignments:

Course assignments were provided weekly (over a 14 week semester) and consisted of selected problems from the course textbook, and nine programming assignments uniformly distributed over the semester. Table III summarizes the topics of these programming assignments and links them to the desired programming skills. Each programming activity was assigned with a problem statement and a required set of deliverables tied to the program performance. The assignment provided some additional support through simple examples of useful functions or a brief discussion of any physical interface issues involved. To fully complete each assignment, the student was required to implement their programmed device in a setting outside of the classroom or lab, and provide a short, written assessment of how their observations of this experience.

Table III. Summary of Lab exercises and Project

#	**Programming Skill**	**Description**	**hands-on activity**
1	Introduction to programming environment, creating, compiling, building, executing	Create a simple teleprompter	display characters on a 2-line 16 char. LCD screen
2	Define/use data types	Display a running pattern of lights on leds	read digital inputs / define digital outputs
3	Use of standard library functions, math operations	Create a simple song using a piezo-speaker	Drive a speaker with a square wave of specific frequency, duration
4	Conditional statements: if, else, else if	Create a two-player game to test reaction speeds	coordinate digital inputs, display outputs based on inputs
5	Repetition: while, for	Students create a stopwatch using the speaker, LCD or 7-segment LEds, and the push button switches.	coordinate digital inputs, display outputs based on inputs
6	Combining conditional statements, repetition	Create a device that monitors how long a refrigerator door is open and sets of an alarm if it is open for a defined length of time.	Read analog input (from a function), use this data to coordinate output functions
7	one-dimensional arrays	Create a system that requires a security code before operating a motor/light	Read digital inputs and compare to a lock-code sequence, when correct, operate a motor/light
8	Multi-dimensional arrays	Create an electronic address book that lets a user input and store a name using the push button switches, LCD and an array.	Use digital inputs to enter data into the system, recall and display this data on the LCD
9	User defined functions	Create a system that demonstrates basic elements of a servo motor system	Drive motors at different speeds using a pwm function based on analog input

4. STUDENT ATTITUDE SURVEY

The programming course was implemented at Tennessee Technological University in fall 2008, spring 2009, and spring 2011. The fall 2008 and spring 09 offering was in C and spring 2011 offering was in Matlab. An assessment plan was developed to measure the impact of the model on students' attitude towards programming. The attitude evaluation was done through pre and post survey. These survey was given to both the model group (those students taking the new hardware based programming class) and the control group (students taking the traditional non hardware based programming class). After spring 2009 implementation, the questionnaire was modified based on several existing validated survey [13-16]. Table IV shows the questions in the survey related to the attitude towards programming. Questions 1-8 are common to all implementations and question 9 through 12 is added to spring 11 implantation. The students answer the questions on a 1-5 (1 – strongly disagree, 5 – strongly agree) Likert scale. Figure 2 shows the result of the pre and post course survey from Fall 08 semester. Figure 3 shows the difference between pre and post survey. Similarly figure 4 and 5 shows results from Spring 09 and Figure 6 and 7 shows results from Spring 11 semester.

Table IV. Attitude Survey Questions

1	I am sure that I can learn programming
2	Generally I feel secure about attempting programming problems
3	If I could avoid programming to get an engineering degree, I would
4	I have a lot of self-confidence when it comes to programming
5	I will use programming in many ways throughout my life
6	I am sure that I can help others use programming to solve problems
7	To be interesting to me programming needs to be connected to real world problems or applications
8	I am excited to learn programming skills that will allow me to control real world devices
9	I see myself joining a professional society related to computer programming or applications (for example ACM, Association for computer machinery) in the future
10	I am interested in joining a club that makes use of programming (for example the robotics club, Unix user group) in the future
11	I would write a program outside of the required class work
12	I would write a program to solve an assignment in another class, even if it was not required.

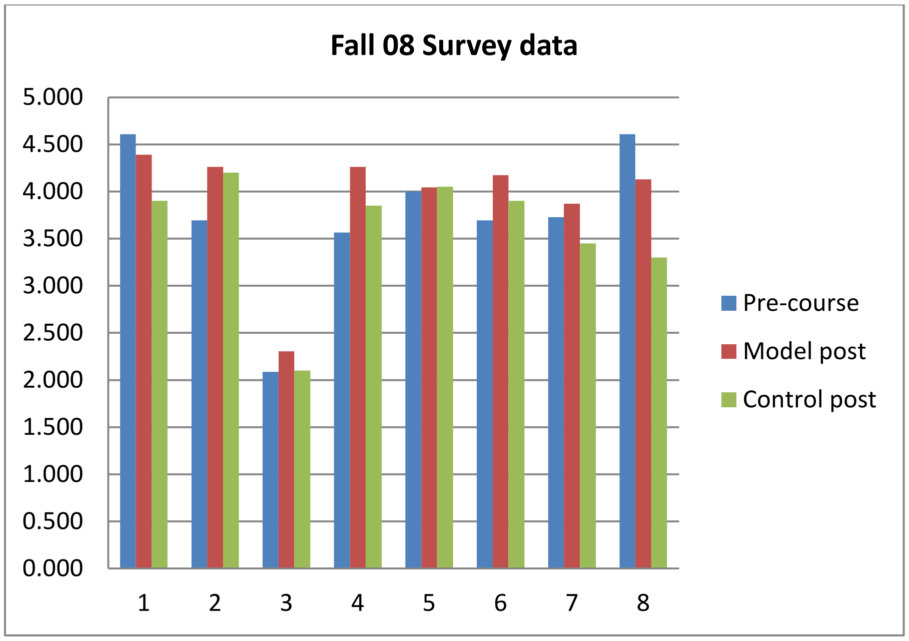

Figure 2

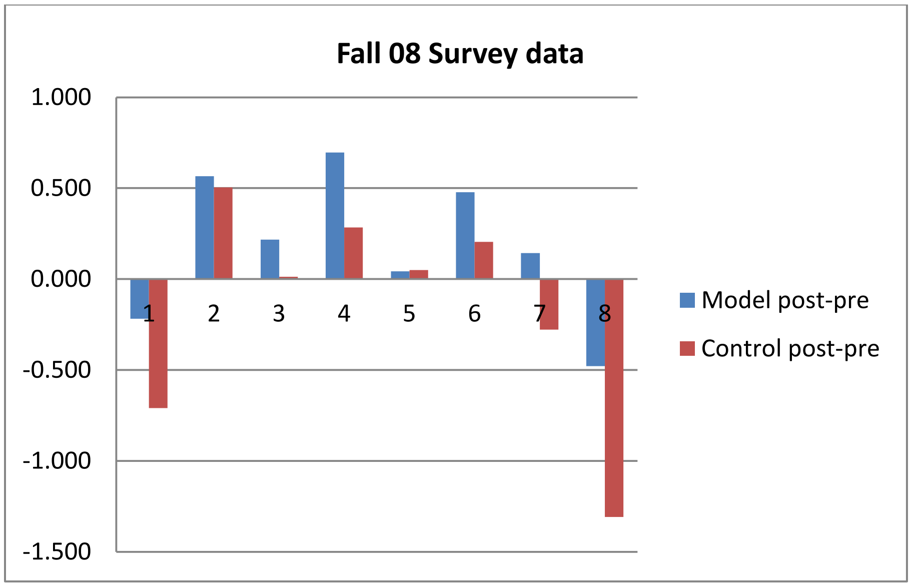

Figure 3

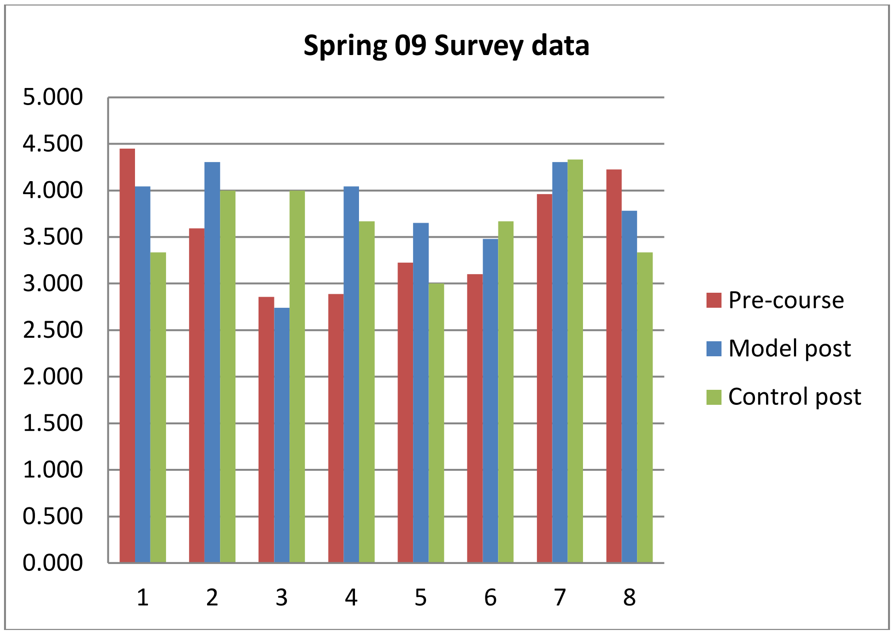

Figure 4

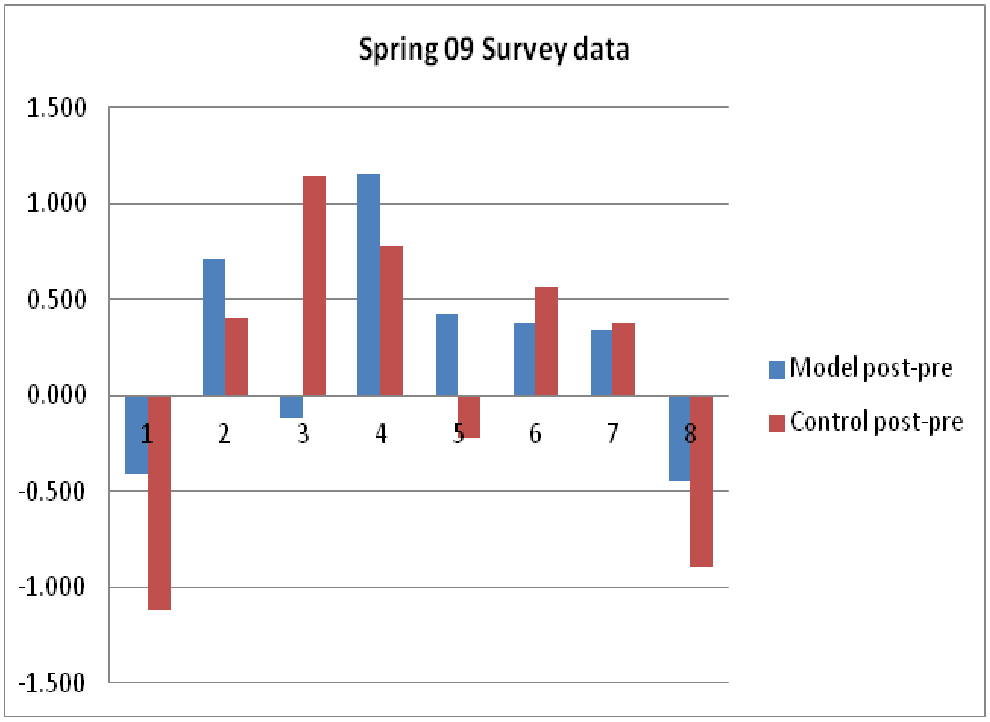

Figure 5

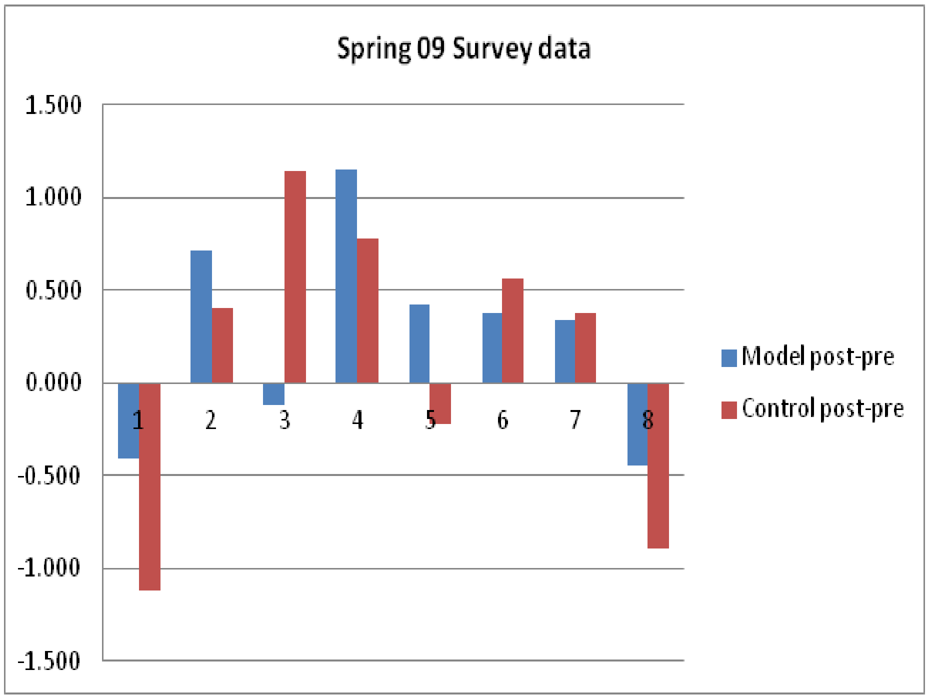

Figure 6

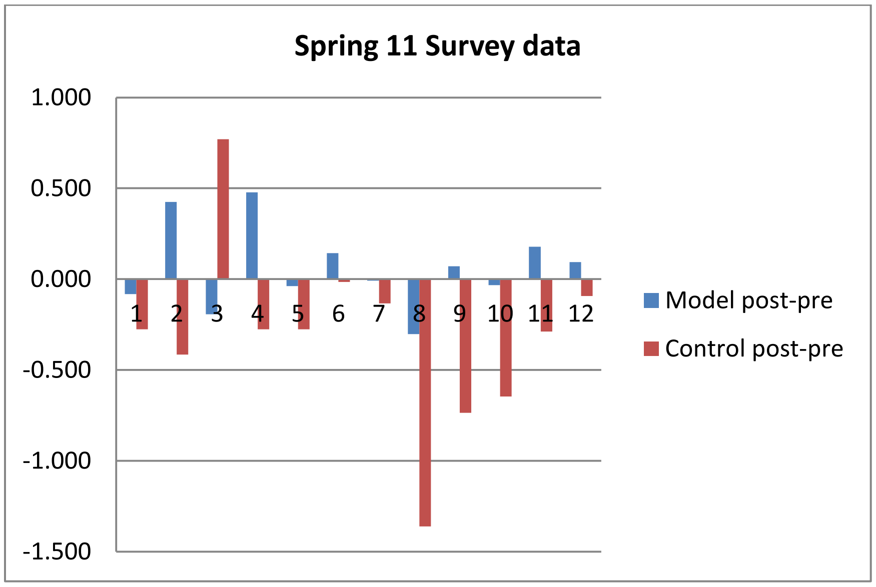

Figure 7

These results show several trends common in freshmen engineering students. They generally feel confident about their ability to learn programming, and believe it is an important skill in engineering. If we compare the attitude of the students at the beginning of the course and at the end of the course in most cases their attitude changes positively (Q2 – Q7: all semesters, Q9 – Q12 spring 11) for the both model group and the control group. In case of model group the change is more positive than the control group. In few cases (Q1 and Q8) it can be seen that the students' attitude has been changed negatively. However, the change in model group is less negative than the change in control group. One potential reason for the negative change in attitude is the difference between students' perception about programming before entering the course and the reality they face during the course. Many freshman students entering in the programming course have the perception that if they successfully complete the course they will be able to develop video games or applications like face book etc. During the course they see that programming requires hard work, study, and lot of hands on practice. They realize developing any significant computer program would require further study, in programming, math, and lot more practice. As a result their initial attitude and excitement about programming decreases. However, the overall change in students' attitude towards programming is positive. The results indicate in general, students that engaged in the hands-on, hardware-based programming activities feel more confident in their ability to learn and use advanced programming and they have more positive attitude towards programming.

5. CONCLUSIONS

This paper has presented a model for the restructuring of the traditional "Introduction to Programming" course for engineering students with an emphasis on hands-on application of programming assignments. The underlying pedagogical foundations of this activity are to engage incoming students' notions of engineering and to build on this early knowledge in a progressive fashion with real-world programming applications that are relevant to engineering and appropriately selected for the target group. The redesigned programming course was implemented at TTU. Students that engaged in the hands-on, hardware-based programming activities reported a more positive change in their attitude towards programming relative to their comparison-group peers. In some cased negative change in attitude has been observed in both groups of students. However, the change is less in case of model group. The negative change in student's attitude can be attributed to difference between students' perception about programming before the course and the reality they learned during the course.

REFERENCES

[1] Shallcross, Lynne, "Piecing It All Together", ASEE Prism, November 2006, Volume 16, Number 3, http://www.prism-magazine.org/nov06/tt_01.cfm.

[2] Canfield, S. L., and M. A. Abdelrahman, "Enhancing the Programming Experience for Engineering Students through Hands-On Integrated Computer Experiences," *Proceedings of 2009 ASEE Southeastern Section Annual Conference,* Marietta, GA, Apr. 5-7, 2009.

[3] Herniter, M. E., Scott, D. R., and R Pangasa, "Teaching programming skills with MATLAB", 2001 ASEE Annual Conference and Exposition, Jun 24-27, Albuquerque, NM.

[4] Clough, D. E., Chapra, S. C. and G. S. Huvard, "A Change in Approach to Engineering Computing for Freshmen, - Similar Directions at Three Dissimilar Institutions," 2001 ASEE Annual Conference and Exposition, Jun 24-27, Albuquerque, NM.

[5] M. H. Naraghi and B. Litkouhi, "An effective approach for teaching computer programming to freshman engineering students," 2001 ASEE Annual Conference and Exposition, Jun 24-27, Albuquerque, NM.

[6] Adamchik, V. and A. Gunawardena, "Adaptive book: Teaching and learning environment for programming education", Proceedings ITCC 2005 – International Conference on Information Technology: Coding and Computing, Las Vegas, NV, Apr. 4-6 2005, p 488-492.

[7] Calloni, B. A. and D. J. Bagert, "Iconic programming for teaching the first year programming sequence," Proceedings – Frontiers in Education Conference, v 1, Atlanta, GA, Nov. 1-4 1995, p 99-102.

[8] Colombo, M. A., M. R. Hernandez and J. E. Gatica, "Combining high-level programming languages and spreadsheets an alternative route for teaching process synthesis and design," 2000 ASEE Annual Conference and Exposition, St. Louis, MO, June 18-21, 2000, p 773-784.

[9] W. Campbell, "Teaching programming by immersion reading and writing," Proceedings – Frontiers in Education Conference v1, Boston MA, Nov 6-9 2002, p T4G/23-T4G/28.
[10] K. Scott, "Teaching Graphical Interface Programmin in Java with the Game of Wari," Proceedings for the 8th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Tessaloniki, Greece, Jun 30-Jul 2, 2003, p 254.
[11] Bransford, J. D., Brown, A., & Cocking, R., How People Learn: Mind, Brain, Experience and School, Expanded Edition, Washington, DC: National Academy Press, 2000.
[12] Committee on How People Learn, A Targeted Report for Teachers, How Students Learn: History, Mathematics, and Science in the Classroom, M. Suzanne Donovan and John D. Bransford, *Editors,* THE NATIONAL ACADEMIES PRESS Washington, D.C., 2005.
[13] Annalisa Weigel, Survey of Aerospace Student Attitudes. http://web.mit.edu/caspar/aerosurvey.htm
[14] Jean-Claude Thomassian, Anoop Desai, and Patrick Kinnicut, " A Study of Student Attitude towards Media Based Instruction in Introductory Engineering Courses",Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference, October 22 – 25, 2008, Saratoga Springs, NY
[15] Nocito-Gobel, J. M. Collura, S. Daniels, and I. Orabi, "Are Attitudes Toward Engineering Influenced by a Project- Based Introductory Course?", Proceedings, 2005 American Society for Engineering Education Annual Conference and Exposition, Portland, Oregon, June 12 - 15, 2005.
[16] Besterfield-Sacre, M.E., C.J. Atman, and L.J. Schuman, "Engineering Student Attitudes Assessment", *Journal of Engineering Education*, 87(2), April 1998, pp. 133-141.
[17] Rensis Likert**,** A Technique for the Measurement of Attitudes," Archives of Psychology, No.140, 1932.
[18] Dragon12 plus, www.evbplus.com
[19] Freescale Codewarrior, www.freescale.com