International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies

http://www.TuEngr.com, http://go.to/Research

CS0: A Project Based, Active Learning Course

Hossein Hakimzadeh [a*], Raman Adaikkalavan [a], and James Wolfer [a]

[a] *Computer Science and Informatics, Indiana University South Bend, USA*

ARTICLEINFO

Article history:
Accepted 26 August 2011
Available online
01 November 2011

Keywords:
Active Learning;
CS0;
Nifty Projects;
Active Learning Classroom;
Retention.

ABSTRACT

The recruitment and retention of students in early computer programming classes has been the focus of many Computer Science and Informatics programs. This paper describes an initiative underway at Indiana University South Bend to improve the retention rate in computer science and informatics. The approach described in this work is inspired by the SCALE-UP project, and describes the design and implementation of an instructor-guided, active learning environment which allows students to gradually acquire the necessary critical thinking, problem solving, and programming skills required for success in computer science and informatics.

1 Introduction

At Indiana University South Bend (IUSB), the university admission policy allows a significant number of at-risk students to enter college and be given a second chance (IU South Bend Admission Rates, 2011). As a result, the overall success rate in lower level courses such as CS1 and CS2 can be as low as 50%, giving ample incentive to develop and implement initiatives that improve retention among computer science and informatics students. The lack of K-12 computer science preparation, inadequate mathematical preparation, poor critical thinking skills, and the lack of awareness of the global competition for high technology jobs

*Corresponding author (**H. Hakimzadeh**). Tel/Fax: +1-574 520 4517. E-mail addresses: hhakimza@iusb.edu.

only add to the reasons why students struggle in beginning science classes. One can hope that recent national initiatives such as STEM (STEM, 2007) which funds research and innovation in educational methods, and “Race to the Top” (Race to the Top, 2009) programs which reward schools that design and implement rigorous academic standards and high-quality assessments will continue to address some of these issues in the future.

In the meantime, to attract and retain students, the computer science education community has developed and experimented with a number of innovative approaches for teaching introductory computer science courses. These approaches include thematic courses such as those using games (Barnes, et al. 2007), computer security (Taylor, et al. 2008), simulation (Stone, 2006), computer graphics (Matzko, et al. 2006 and Batzinger, et al. 2011), and robotics (Pamela, et al. 2003 and Imberman, et al. 2005). Others (D'Souza, et al. 2008) have obtained good results by introducing supplemental instruction and student mentors in early computer programming courses. Finally, a number of initiatives have used active learning approaches which promote working in small groups on hands-on activity (McConnell, 2006 and Whittington, et al. 2008).

At IU South Bend, our response to student retention has been to design and develop a new CS0 course (“CSCI B100 - Problem Solving Using Computers”). This course provides students interested in Computer Science or Informatics the opportunity to explore the required skills needed for the discipline in a controlled “incubator” like setting. The new course incorporates ideas from thematic courses as well as active learning to introduce the students to structured problem solving, working within small groups, research, data collection, reflective thinking, stepwise refinement, dialogue and discussion. Additionally, the students are exposed to introductory programming concepts such as variables and memory manipulation, conditionals, loops, modularization and code reuse, arrays, and objects.

2 Course Design

The new CS0 course is divided into two phases. During the first phase (approximately the first 8 weeks of the semester), students are actively engaged in a series of problem solving exercises. Each exercise will require approximately 3 to 4 class sessions. During these sessions, students (in small groups) work jointly with their instructor and lab assistant to analyze and solve an assigned problem. The exercises also require group work outside of class (data collection, meetings, etc.) The goal of these supervised exercises is to provide a hands-on demonstration of how an experienced practitioner approaches a problem, how alternative solutions are considered and developed, and how each solution is evaluated in order to select the best solution. The solution is then implemented, revealing the process of identifying and correcting errors in design and implementation and illustrating how the solution is incrementally refined until the problem is solved. This process of problem specification, analysis, design, implementation, refinement, and documentation is repeated with four problems from different domains, helping the students recognize the natural patterns which exist in solving many problems.

During the second phase of the course, assignments become more traditional and the instructor limits his or her contribution to the solution. The instructor serves as a coach and helps the student groups focus their thinking and activities. Assignments during the second phase are small extensions of the assignments given in the first phase; however, each assignment will provide an opportunity for deeper understanding of programming concepts.

The current set of projects were designed, developed and implemented by the authors to complement the background of the student body entering this course. Additional projects, based on “nifty assignments” (McGuire, et al. 2008; Franke, 2008; Reed, 2002; and Shiflet, 2007), have been implemented but are yet to be tested in the classroom. Currently, the course includes five contact hours per week, three hours of active-learning sessions including short lectures and group activities, and two hours of hands-on laboratory. Five sections of this course were taught during fall 2009, spring 2010, and fall 2010 semesters. We expect the students taking this course to have basic algebra skills. Students were asked to form teams of two or three, and assignments and labs were assigned to groups. To receive full credit, each student was responsible for submitting the entire assignment and/or lab. Also, after each assignment, group members were required to submit a “Self and Peer” evaluation form. The course did not require a text book, but lecture notes as well as video lectures were made available to the students via the class web site and the IU course management system. Additional information including assignments, labs, lecture notes, videos, etc., is available on the course web site at www.cs.iusb.edu/~A290.

2.1 Assignments

An important aspect of developing this course was the construction of appropriate problem solving and programming exercises. Our goal was to develop exercises that rely on basic mathematical skills, are relevant to student life and experiences, draw from diverse and interesting problem domains (attracting students with different background to the discipline), be limited in their scope so that it can be defined, analyzed, designed, and implemented in approximately five hours of lecture and laboratory, can be easily understood by beginning students, foster social interaction and team work, and can train students in understanding problem solving patterns and using programming concepts and elementary data structures. At the same time the exercises should be extendible, so that a more complex version can be assigned during the second phase of the course.

Depending on the faculty offering the course, a total of three or four major assignments were given to the students. Each assignment consisted of 3 distinct parts with their own due dates. The initial two parts were used in the first phase of the course and the third part was used in the second phase. Below, we will describe one assignment in depth and the other three briefly.

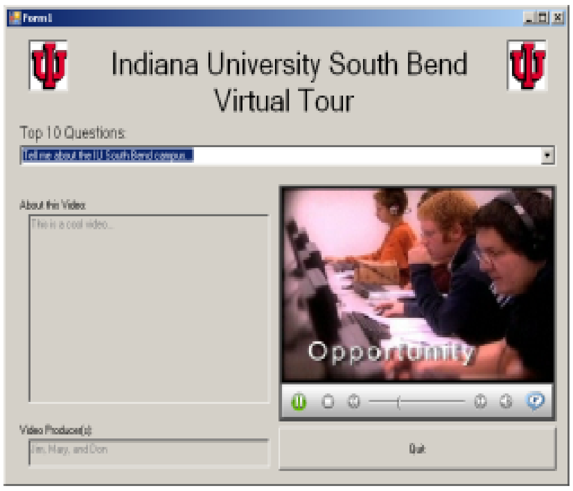
2.1.1 Assignment 1: Virtual Tour

In this assignment, students engage in the production, design and development of an application that can load and play videos. This exercise requires the use of conditionals, and Graphical User Interface (GUI) controls such as combo-boxes, text-boxes, buttons, and video player. It is meant to be simple, since it is assigned during the first class period, and student groups can begin working on the analysis and design component of the program immediately. Despite its simplicity, the problem is presented as a plausible real world application and students find it enjoyable to work with. We first discuss the variations to this basic program:

- Virtual Campus Tour: In this exercise (Figure 1) potential questions about the university campus are compiled, and videos answering each question is shot and edited. The program is then developed to provide a virtual campus tour by allowing the user to select a question and watch the corresponding video.
- Virtual Public Place Tour: Similar to the above, but this tour requires students to visit a public place and perform the same activities.
- Faculty Profile System: Textual and video information about faculty are compiled.

The user is then able to select a faculty and view his or her profile, learning about their teaching and research interests.

- Virtual Product Tour: Information about one or more products (e.g., computer hardware, software, car, flat screen TV, cell phone, etc.) are compiled and presented.
- Degree Profile System: Textual and video information about degree programs available at the university are compiled. The user is then able to select a degree program and view relevant information and interviews about that discipline.



Tell me about the IU South Bend campus.

This is a cool video...

Jim, Mary, and Don

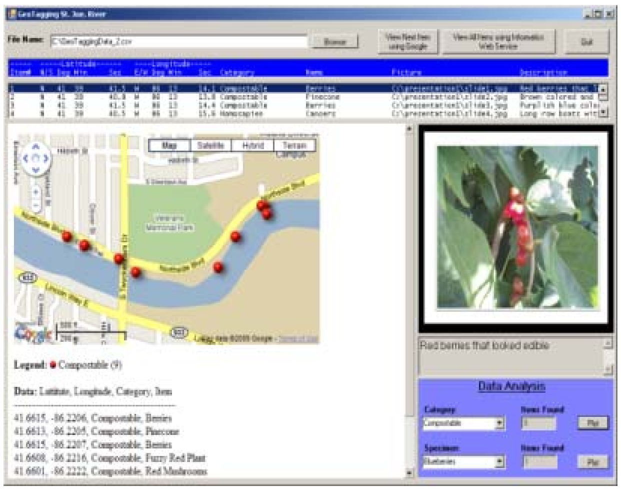
Figure 1: Virtual Tour

Below we discuss the three phases of the "Virtual Public Place Tour". In the first phase, students were asked to research about virtual tours offered by businesses, select a public place, and create a set of questions to be answered about that place. After submission, the instructor typically discusses all the student submissions in the class. Based on the discussions, questions by the groups are refined. In the second phase, students create short edited videos answering those questions and also create a GUI design for their tour. Sometimes their GUI design is very complex and students will refine their design post submission. In the third phase, students are asked to implement their design using Visual Basic. Students are asked to load the information (questions, video file location, etc.) needed for the program from a CSV or XML file.

Programming principles explored in all the three phases include: Variables, operators, conditionals, duplicate elimination, file handling, documentation, and GUI design and controls.

2.1.2 Assignment 2: Geo-Tagging

This next assignment is slightly more complex. The basic idea is to identify items of interest, classify them, tag their latitude and longitude, take pictures or videos of the items, and possibly collect other data about each item. Then the program is designed to allow the user to navigate through the data set, and display the data on a map using Google or Yahoo map via a web service. A later extension to the assignment will allow the user to view the items by category, count the items in the category, etc. Variations to this assignment can include:



```
-----  -----Latitude------  ----Longitude-----
Item#  N/S Deg Min   Sec   E/W Deg Min   Sec  Category      Name       Picture                       Description
1      N   41  39    41.5  W   86  13    14.1 Compostable   Berries    C:\presentation1\slide1.jpg   Red berries that l
2      N   41  39    40.8  W   86  13    13.8 Compostable   Pinecone   C:\presentation1\slide2.jpg   Brown colored and
3      N   41  39    41.5  W   86  13    14.4 Compostable   Berries    C:\presentation1\slide3.jpg   Purplish blue colo
4      N   41  39    40.5  W   86  13    15.6 [illegible]   [illegible] C:\presentation1\slide4.jpg  Long row boats wit
```

Legend: ● Compostable (9)

Data: Latitude, Longitude, Category, Item

```
41.6615, -86.2206, Compostable, Berries
41.6613, -86.2205, Compostable, Pinecone
41.6615, -86.2207, Compostable, Berries
41.6608, -86.2216, Compostable, Fuzzy Red Plant
41.6601, -86.2222, Compostable, Red Mushrooms
```

Figure 2: Geo-Tagging

- Geo-Tagging the local St. Joseph River: which allows the user to select a segment of the St. Joseph River (a river near the university), then virtually navigate the banks of the river, viewing items of interest (
- Figure 2). Clicking the icon for each item will reveal some additional information about the item.
- Geo-Tagging for Realtors: which allows the user to select a location on the map and click on the houses that are for-sale. Clicking the icon for the house will reveal basic information about the house such as it street address, price, image, etc. (perhaps some other data typically found on a Realtor Multiple Listing Service (MLS) sheet)
- Fast-Food Finder: which allows the users to virtually navigate the neighborhood around the campus and identify the nearby fast-food restaurants. Clicking the icon for the restaurant will reveal some basic information about the business such as it street address, type of food or beverage sold, student related specials or discounts.

<u>Programming principles explored:</u> Conditional, loops, array, functions, procedures, file

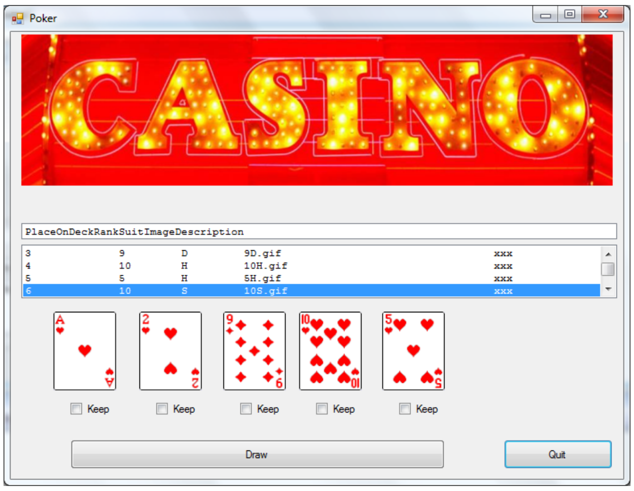
I/O, classes, documentation, and GUI design.

2.1.3 Assignment 3: Game (Video Poker)

This assignment is quite popular with students and creates a fully functioning Video Poker game (**Error! Reference source not found.**). It allows the player to draw from a deck of cards, keep the cards they want and draw up to four new cards. After which the program has to evaluate if the player has a winning hand and calculate the payout. Variations to this assignment can include:

- Black-Jack: in which one is playing against the dealer.
- Matching / Concentration game, where the player tries to match a pair of cards. The board can be 4x4, or 6x6 (even number of cards), all the cards are taken from the same deck of cards. The game is timed to allow the player to evaluate his or her performance.

Programming principles explored: Conditional, loops, array, functions, procedures, random numbers, file I/O, classes, documentation, and GUI design.



```
PlaceOnDeckRankSuitImageDescription
3            9        D        9D.gif                              xxx
4            10       H        10H.gif                             xxx
5            5        H        5H.gif                              xxx
6            10       S        10S.gif                             xxx
```

Figure 3: Video Poker.

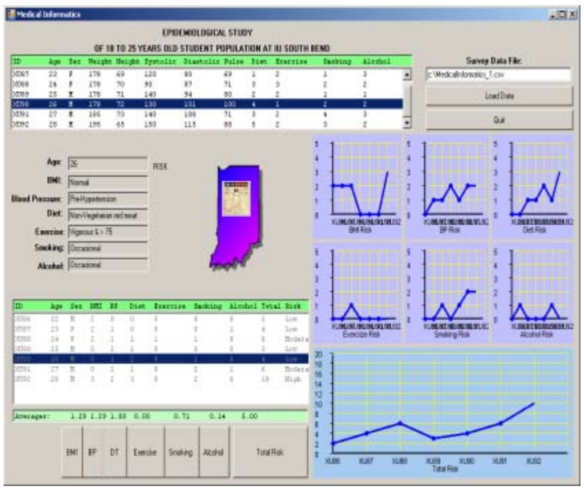
2.1.4 Assignment 4: Medical Informatics

This assignment requires considerable up-front research and team activity. The goal of this assignment is to familiarize students with the data collection, measurement, and analysis, needed in many areas of research. Students are asked to first develop health survey, then are asked to conduct an anonymous survey of (18 to 25 year old college students), measuring their height, weight and blood pressure. The collected information is then brought into the program and analyzed, then displayed in a variety of different forms. The user can select a specific subject and see their individual data points, they can see the average for each category, and finally they can see the data graphically using a line or bar chart (Figure 4). Variations to this assignment can include:

- An interactive version of the software which allows individuals to walk up to a kiosk, enter their own information and see how they compare to others in the population.

In this assignment we intentionally stayed away from recommending behavioral changes, or providing medical (or even common sense) advice to survey subjects. Instead, in consultation with the school of nursing each student was provided with an information sheet, pointing them to appropriate medical resources.

Programming principles explored: Conditional, loops, array, functions, procedures, file I/O, classes, documentation, and GUI design.



Medical Informatics

EPIDEMIOLOGICAL STUDY

OF 18 TO 25 YEARS OLD STUDENT POPULATION AT IU SOUTH BEND

ID	Age	Sex	Weight	Height	Systolic	Diastolic	Pulse	Diet	Exercise	Smoking	Alcohol
XU87	22	F	170	69	120	80	69	1	2	1	3
XU88	24	F	170	70	90	87	71	3	3	2	2
XU89	25	M	170	71	140	94	80	2	2	1	1
XU90	26	M	170	72	130	101	100	4	1	2	2
XU91	27	M	186	73	140	108	71	3	2	4	3
XU92	28	M	195	65	150	115	88	5	2	3	2

Survey Data File: c:\MedicalInformatics_T.csv

Age: 26

BMI: Normal

Blood Pressure: Pre-Hypertension

Diet: Non-Vegetarian red meat

Exercise: Vigorous & > 75

Smoking: Occasional

Alcohol: Occasional

ID	Age	Sex	BMI	BP	Diet	Exercise	Smoking	Alcohol	Total	Risk
XU86	22	M	1	0	0	0	0	0	1	Low
XU87	23	F	1	1	0	0	0	1	4	Low
XU88	24	F	1	1	1	1	1	0	6	Modera
XU89	25	M	0	1	1	0	0	0	2	Low
XU90	26	M	0	1	1	0	1	0	4	Low
XU91	27	M	0	1	1	0	1	1	6	Modera
XU92	28	M	1	2	3	0	2	0	13	High

Averages: 1.29 1.29 1.83 0.00 0.71 0.14 5.00

Figure 4: Medical Informatics.

2.1.5 Other Assignments

Additional, less involved, assignments were assigned by the instructor(s) as necessary. These assignments were designed to reinforce a specific topic or provide additional exposure to new topics.

Table 1: Assignment to Curriculum Mapping.

	Variables, Operators	Conditionals	Loops	Procedures, Functions	Arrays, Collections	Classes, Objects	File I/O	GUI
Assign 1 (Virtual Tour)	1	1	0	0	0	0	0	1
Assign 2 (Geo-Tagging)	2	2	1	0	0	1	0	2
Assign 3 (Video Poker)	2	2	1	1	1	1	1	2
Assign 4 (Medical Informatics)	2	2	1	1	1	1	1	2
Assign 5 (Virtual Tour)	3	2	2	2	2	1	2	3
Assign 6 (Geo-Tagging)	3	2	3	2.5	2.5	1.5	2	3
Assign 7 (Video Poker)	3	3	3	2.5	3	1.5	2	3
Assign 8 (Medical Informatics)	3	3	3	2.5	3	1.5	2	3

2.1.6 Evaluation Rubrics for Assignments

Tables 1 and 2 represent the instructors' view of how each assignment contributes to the overall learning goals of the class. Table 1 maps each assignment to the course curriculum learning objectives. The learning objectives for the course are: understanding variables and memory, I/O, conditionals, loops, modules, parameter passing, arrays, classes, files, and GUI. Although the code that is provided to the student for each assignments include all of the curriculum learning objectives, each assignment (and its corresponding laboratories) focuses on a few of the objectives. For example, assignment 1 concentrates on variables, conditionals and GUI, yet the code provided to the students also includes procedures, and classes.

Table 2: Overall Course Objectives to Assignment Mapping.

	Virtual Tour	Geo-Tagging	Video Poker	Medical Informatics
Dealing with Complexity	1	2	3	3
Fostering Creativity	2	3	3	3
Fostering Group Work	3	3	3	3
Showing Initiative	2	3	2	3
Research	1	2	2	3
Problem Solving	1	2	3	3
Understanding code	1	2	3	3

Table 2 maps each assignment to the overall course objectives. Each assignment is rated by the instructor in terms of overall complexity, required creativity, group interaction, initiative, research, problem solving, reading and understanding programs, and developing new code.

2.2 Skill Building Labs

While working on PART-1 and PART-2 of each assignment, the students are also required to complete a number of "Skill Building" labs (approximately 3 labs per assignment). These hands-on laboratory exercises provide the essential programming skills necessary to understand and complete their future assignments. The challenge in doing the assignments is most often related to integration of skills in order to solve a larger and more complex problem.

2.3 Active Learning: Classroom and Group Activities

The traditional lecture-style classrooms are not suitable for group interaction or activities and having an appropriate classroom is essential to the success of this course. In order to foster easier group interaction and small community based learning, we have designed and constructed a new active-learning classroom inspired by the SCALE-UP (Beichner, 2007) classroom design. The classroom is specifically designed for group activities. Students sit around a collaborative station (each station can have up to five members). In this class, groups usually have two or three members. Each station is equipped with a large display monitor attached to a computer. The classroom walls are used as whiteboards for group discussions. We have developed various timed group activities specifically for this course to enhance student learning. Below, we discuss one of those group activities.

Peer Learning: In this activity a problem is given to teams consisting of two students. Each team is required to solve the problem using the software development life cycle model. Teams have to analyze the problem and develop algorithms using the white board. The instructor serves as a consultant to the teams. Teams that have completed their algorithm are asked to visit another team and evaluate their algorithm. After the evaluation period is complete, the instructor discusses the algorithms developed by each team, and demonstrates the pattern involved in solving the problem. This activity allows peer learning and also helps the instructor to demonstrate the various ways in which a problem can be solved. The next step in this activity is to translate the algorithm to a computer program. This allows the students to see how their algorithm works in the computer.

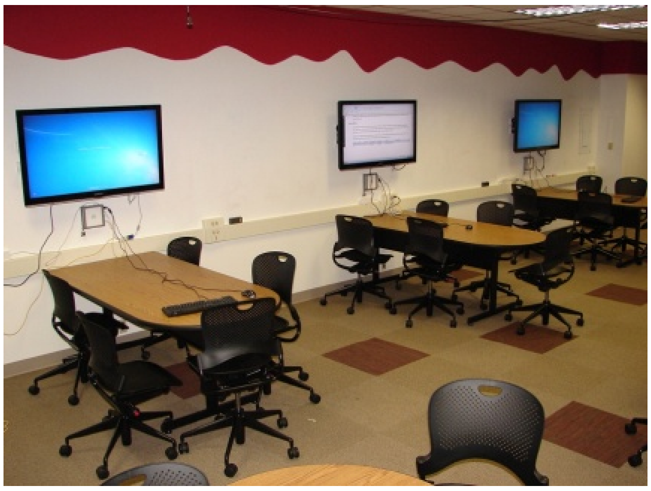


Figure 5: Active Learning Classroom

2.4 Self and Peer Evaluation

Having students work within a team environment makes it more challenging to maintain accountability. In order to promote individual accountability, we have developed separate forms for self and peer evaluation. After each assignment, each student must report his or her contribution (as well as their team members) to the overall effort. The reporting categories include group participation, brain storming, data collection, data analysis, design activities, and overall contribution.

3 Reflections and Student Feedback

The student feedback seems to support the primary course objectives of improving retention by systematically introducing problem solving and basic programming skills. Anecdotally, the students rated the course as good to excellent. Informal comments indicated that they were initially anxious about learning problem solving and programming, however as the course progressed, they started to gradually understand the patterns of problem solving. Many seemed to feel that the course was at times overwhelming, and some found that working together as a team was stressful.

Positive:

- Pace of the course was good.
- Labs were helpful to understand the material.

- Lecture handouts and videos were useful in the learning process.
- Teamwork helped to work and learn more effectively.
- Some of our programming labs were designed to walk the student through the problem solving step in detail, causing some students to ask for labs without any walk through.
- Course was found to be helpful, would recommend the course to other students as their first computer science class.

Negative:

- More lab time would have been helpful.
- Some students felt that programming should have been introduced much earlier.
- Despite efforts to design interesting assignments, encouraging peer learning, providing ample online resources, some students still found the course challenging.
- At this point the course does not require a text book. Some students indicated preference for having a text book, but the majority felt that the online resources as well as the handouts provided in class were sufficient.

Reflections

- Significant planning and development has been put toward crafting assignments that are both interesting and represent real-world problems. Although anecdotal student comments are generally positive, since this is only the third semester that this class has been offered there is insufficient quantitative data to report. In order to assess student experiences in the course, the authors have created a survey instrument which can be accessed online at *http://www.cs.iusb.edu/~A290/B100_Student_Survey.pdf*
- As of the fall of 2010 semester, this course is being conducted in a newly constructed active learning classroom designed to support group interaction. Based on our observations of student interaction, we believe that the majority found the classroom to be an effective learning environment. We will be soliciting student feedback with respect to the classroom activities, such as peer learning and group exercises in the upcoming semesters.

4 Conclusion

This paper describes a new approach for teaching CS0. The goal of this approach is to make computer science and informatics more accessible to aspiring majors and to improve retention. The approach employs guided exercises using an active learning classroom and laboratory environment. The course engages the students in teams of two or three, working on real-world problems, collecting and cleansing data, conducting research, performing hands-on labs, completing in-class active learning exercises, reading, and writing code. Compared with the traditional CS1, student motivation was high, with higher than normal attendance, and increased student retention. We anticipate that the majority of these students will make the transition to CS1, and we look forward to evaluating student success in CS1. Although the results are preliminary, they appear to be promising. Based upon our experience and student feedback, we believe that the new CS0 course successfully assists students with the gradual development of the critical thinking, and problem solving skills, necessary to succeed in Computer Science and Informatics.

5 References

Barnes, T., et al. (2007) Game2Learn: Building CS1 Learning Games for Retention, *In Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE)*, pp. 121-125.

Batzinger, R., Hakimzadeh, H., (2011) Laboratory Exercises for A201 Introduction to computing. *Online:* *http://mypage.iusb.edu/~hhakimza/201_505/index.html*

Beichner, R. (2007). The Student-Centered Activities for Large Enrollment Undergraduate Programs (SCALE-UP) Project. *In Research-Based Reform of University Physics (1). Online:* *http://www.ncsu.edu/PER/SCALEUP/Classrooms.html*, *http://scaleup.ncsu.edu/*

D'Souza, D., et al. (2008) Transforming Learning of Programming: a Mentoring Project, *ACE '08: Proceedings of the tenth conference on Australasian computing education* - Volume 78, pp. 75-84.

Franke, B., Catching Plagiarists (2008). *Online:* *http://nifty.stanford.edu/2008/franke-catch-plagiarists/#introduction*

Imberman, S., P., Klibaner, R. (2005) A Robotics Lab for CS1, *Journal of Computing Sciences in Colleges, 21(2), Consortium for Computing Sciences in Colleges*, pp. 131-137.

IU South Bend Admission Rates (2011), *IU Institutional Research and Reporting* *http://www.iu.edu/~uirr/reports/compliance/common/index.shtml and* *http://www.iu.edu/~uirr/reports/compliance/doc/common_dataset/CDS_2010/IUSB_2010.pdf (Sections C and D)*

Matzko, S., Davis, T., A. (2006) Teaching CS1 with Graphics and C, *inroads – SIGCSE Bulletin*, 38(3), pp. 168-172.

McConnell, J., J. (2006) Active and Cooperative Learning: Final Tips and Tricks (Part IV), *inroads – SIGCSE Bulletin*, 38(4), pp. 25-28

McGuire, M., Murtagh, T., P. (2008) Huff(man)ing and Puffing – Huffman Image Compression. *Online:* http://nifty.stanford.edu/2008/mcguire-murtagh-huffman-image/

Pamela B. Lawhead, P.B., et al. (2003) A Road Map for Teaching Introductory Programming using LEGO© Mindstorms Robots, *SIGCSE Bulletin*, 35(2), pp. 191-201.

Race to the Top (2009): http://www.whitehouse.gov/the-press-office/fact-sheet-race-top

Reed, D. (2002) Nifty Assignments. *Online:* http://www.dave-reed.com/Nifty

Shiflet, A. B. (2007) Spreading of Fire. *Online:* http://nifty.stanford.edu/2007/shiflet-fire/

STEM - Science Technology Engineering and Mathematics initiative (2007), The Department of Education and the National Science Foundation. http://www.nsf.gov/nsb/stem/

Stone, J, A. (2006) Using a Machine Language Simulator to Teach CS1 Concepts, *inroads –SIGCSE Bulletin*, 38(4), pp. 43-45

Taylor, B., Azadegan, S. (2008) Moving Beyond Security Tracks: Integrating Security in CS0 and CS1, *In Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE)*, pp. 320-324.

Whittington, K., Yacci, M. (2008) Active Learning for Classroom Management Model, *Proceedings of the Informing Science & IT Education Conference*, pp. 231-239. *Online:* http://proceedings.informingscience.org/InSITE2008/InSITE08p231-239Whit484.pdf

Dr.Hossein Hakimzadeh is an Associate Professor of Computer Science and Informatics at Indiana University South Bend, USA. His current research interests include database systems, software engineering, and computer science education.

Dr.Raman Adaikkalavan is an Assistant Professor of Computer Science and Informatics at Indiana University South Bend, USA. He received his Ph.D. in Computer Science and Engineering from The University of Texas at Arlington in 2006. His current research interests include security in computing, data stream processing, complex event processing, and elements of teaching and pedagogy.

Dr.James Wolfer is a Professor of Computer Science and Informatics at Indiana University South Bend, USA. His research interests include naturally inspired computing, medical imaging, graphics, and education.

Peer Review: This article has been peer reviewed and accepted by IAJC-ASEE International Joint Conference on Engineering and Related Technologies sponsored by IAJC, ASEE, IEEE, venue at University of Hartford, USA during April 29-30, 2011.